D

Data Envelopment Analysis
DEA

R. De Leone
Dip. Mat. e Fisica, University degli Studi di Camerino,
Camerino, Italy

MSC2000: 90B50, 90B30, 91B82, 90C05

Article Outline

Keywords

DEA; Comparative efficiency assessment; Linear programming

Data envelopment analysis (DEA) is a novel technique based on linear programming for evaluating the relative *performance* of similar units, referred to as decision making units (DMUs). The system under evaluation consists of n DMUs: each DMU consumes varying amount of m_1 different inputs (resources) to produce m_2 different outputs (products). Specifically, the jth DMU is characterized by the input vector $x^j > 0$ and the output vector $y^j > 0$. The aim of DEA is to discern, for each DMU, whether or not is operating in an efficient way, given its inputs and outputs, relative to all remaining DMUs under consideration. The measure of *efficiency* is the ratio of a weighted sum of the outputs to a weighted sum of the inputs. For each DMU, the weights are different and obtained by solving a linear programming problem with the objective of showing the DMU in the best possible light.

The ability to deal directly with incommensurable inputs and outputs, the possibility of each DMU of adopting a different set of weights and the focus on individual observation in contrast to averages are among the most appealing features of model based on DEA.

A process is defined *output-efficient* if there is no other process that, using the same or smaller amount of inputs, produces higher level of outputs. A process is defined *input-efficient* if there is no other process that produces the same or higher level of outputs, using smaller amount of inputs. For each orientation there are four possible models:

1) the 'constant returns' model;
2) the 'variable returns' model;
3) the 'increasing returns' model;
4) the 'decreasing returns' model.

Each model is defined by a specific set of economic assumptions regarding the relation between inputs and outputs [10,11]. Associated with each of the four DEA models, independent of the orientation, there is a *production possibility set*, that is, the set of all possible inputs and outputs for the entire system. This set consists of the n DMUs and of 'virtual' DMUs obtained as linear combination of the original data. The efficient frontier is a subset of the boundary points of this production set. The objective of DEA is to determine if the DMU under evaluation lies on the efficient frontier and to assign a score based on the distance from this frontier [6].

The production set for the 'constant returns' model is

$$T_1 = \left\{ (x, y) \geq 0 : \begin{array}{c} x \geq \mu \sum_{j=1}^{n} x^j \lambda_j, \\ y \leq \mu \sum_{j=1}^{n} y^j \lambda_j, \\ \forall \lambda_j \geq 0, \quad \sum_{j=1}^{n} \lambda_j = 1, \\ \mu > 0 \end{array} \right\},$$

while for the ‘variable returns’ model we have

$$T_2 = \left\{ (x,y) \geq 0 : \begin{array}{c} x \geq \mu \sum_{j=1}^{n} x^j \lambda_j, \\ y \leq \mu \sum_{j=1}^{n} y^j \lambda_j, \\ \forall \lambda_j \geq 0, \quad \sum_{j=1}^{n} \lambda_j = 1 \end{array} \right\}.$$

The production sets for the ‘increasing’ (resp. ‘decreasing’) returns models are similar to the set T_2 above with the equality constraint $\sum_{j=1}^{n} \lambda_j = 1$ replaced by the inequality $\sum_{j=1}^{n} \lambda_j \leq 1$ (resp. $\sum_{j=1}^{n} \lambda_j \geq 1$).

The ‘constant returns, input oriented’ envelopment LP is given next:

$$\begin{cases} \min\limits_{\theta,\lambda \geq 0} & \theta \\ \text{s.t.} & \sum\limits_{j=1}^{n} x^j \lambda_j - x^{j^*}\theta \leq 0 \\ & \sum\limits_{j=1}^{n} y^j \lambda_j \geq y^{j^*}. \end{cases} \tag{1}$$

For the ‘constant returns, output oriented’ case we have, instead, the following LP problem [4]:

$$\begin{cases} \max\limits_{\psi,\lambda \geq 0} & \psi \\ \text{s.t.} & \sum\limits_{j=1}^{n} x^j \lambda_j \leq x^{j^*} \\ & \sum\limits_{j=1}^{n} y^j \lambda_j - y^{j^*}\psi \geq 0. \end{cases} \tag{2}$$

In both cases the additional constraint

$$\sum_{j=1}^{n} \lambda_j \begin{pmatrix} = \\ \leq \\ \geq \end{pmatrix} 1$$

defines the LP for the variable, increasing and decreasing returns DEA models, respectively.

The corresponding dual problem for the input-oriented case is

$$\begin{cases} \max\limits_{\pi,\sigma,\beta} & \sigma^\top y^{j^*} + \beta \\ \text{s.t.} & -\pi^\top x^j + \sigma^\top y^j + \beta \leq 0 \\ & j = 1,\dots,n \\ & \pi^\top x^{j^*} \leq 1 \\ & \pi \geq 0, \quad \sigma \geq 0, \end{cases} \tag{3}$$

where $\beta = 0$, β unrestricted, $\beta \leq 0$ and $\beta \geq 0$ for the constant, variable, increasing and decreasing return DEA models.

For the output-oriented case the dual is:

$$\begin{cases} \min\limits_{\pi,\sigma,\beta} & \pi^\top x^{j^*} + \beta \\ \text{s.t.} & \pi^\top x^j - \sigma^\top y^j + \beta \geq 0 \\ & j = 1,\dots,n \\ & \sigma^\top y^{j^*} \geq 1 \\ & \pi \geq 0, \quad \sigma \geq 0 \end{cases} \tag{4}$$

with $\beta = 0$, β unrestricted $\beta \geq 0$ and $\beta \leq 0$ for the constant, variable, increasing and decreasing returns DEA models.

For the ‘input-oriented, constant returns’ case, the reference DMU j^* is

- *inefficient* if
  - the optimal value of problem (1) is different from 1, or
  - the optimal value of Problem (1) is equal to 1 but there exists an optimal solution with at least one slack variable strictly positive;
- *efficient* in the remaining cases.

Moreover the efficient DMU j^* can be

- *extreme-efficient* if Problem (1) has the unique solution $\lambda^*_{j^*} = 1, \lambda^*_j = 0, j = 1, \dots, n, j \neq j^*$;
- *nonextreme efficient* when Problem (1) has alternate optimal solutions.

The efficiency for the other models is defined in a similar manner.

The conditions $\theta \geq 0$ and $\psi \geq 0$ can be introduced without loss of generality in (1) and (2) since only nonnegative values for these variables are possible given our assumption on the data. Since $\lambda_{j*} = 1, \lambda_j = 0$ for $j \neq j^*$, $\theta^* = 1$, and $\lambda_{j*} = 1, \lambda_j = 0$ for $j \neq j^*$, $\psi^* = 1$ are always feasible for (1) and (2), respectively, the optimal objective function value lies in the interval $(0, 1]$ for the input orientation case and $[1, \infty)$ for the output orientation case.

The linear programs (1) and (3) above can be interpreted in the following way. In the input-oriented case, we compare the reference DMU j^* with a ‘virtual’ DMU obtained as linear combination of the original DMUs. Each input and output of this virtual DMU is a linear combination of the corresponding component of the inputs and outputs of all the DMUs. The optimal value is, in this case, always less than or equal to 1. If the optimal value is strictly less than 1, then it is possible to construct a virtual DMU that produces at least the same amount of outputs as the reference DMU using an amount of inputs that is strictly smaller than amount used by the j^*th DMU. When this happens we declare the DMU j^* inefficient. Instead, when the optimal value is equal to 1 there are three possible cases:

- there exists an optimal solution with at least one slack variable strictly positive;
- the optimal solution is unique;
- there exists multiple optimal solutions.

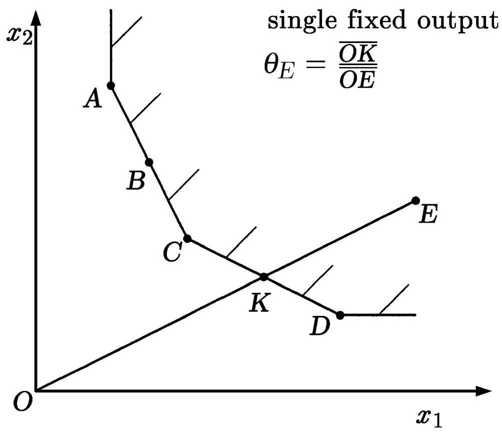
In the first case we declare the reference DMU inefficient. In the last two cases the j^*th DMU is efficient (extreme-efficient, respectively nonextreme efficient).

For Problem (3), the optimal solution π^* and σ^* represent the weights that are the most favorable for the reference DMU, i. e., the weights that produce the highest efficiency score under the hypothesis that, using the same weights for the other DMUs, the efficiency remains always below 1.

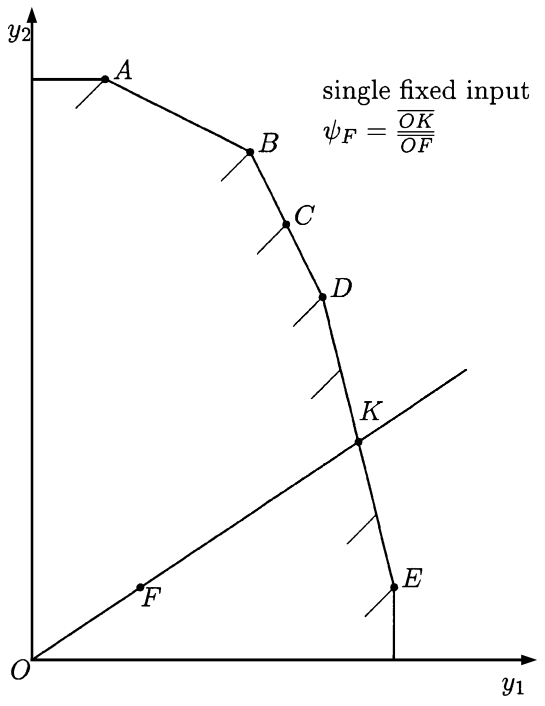
Similar interpretations can be given for the output-oriented case for Problems (2) and (4).

In Fig. 1 it is represented the production possibility set and the efficient frontier for the five DMUs 'A' to 'E'. These DMUs are characterized by two different inputs and a single output value set to some fixed value.

All the DMUs are efficient but the DMU 'E'. The DMU 'B' is efficient but nonextreme. The virtual DMU 'K', obtained as convex combination of the DMUs 'C' and 'D', is more efficient than the DMU 'E'. The optimal value θ^* for the linear programming problem (1) for the DMU 'E' is exactly the ratio of the lengths of the segments OE and OK.



Data Envelopment Analysis, Figure 1
Two-input, single output DMUs



Data Envelopment Analysis, Figure 2
Two-output, single input DMUs

Figure 2 shows the case of DMUs characterized by two distinct outputs and a single input set to a fixed value. All the DMUs are efficient except the DMU 'F' that is dominated by the virtual DMU 'K'. The optimal value ψ^* for the linear programming problem (2) for the DMU 'F' is the ratio of the lengths of the segments OE and OK.

The original 'constant returns' model was proposed in [4]. In [2] the *variable returns* model was proposed with the objective of discriminating between technical efficiency and scale efficiency. The bibliography published in [7] (part of [3]) contains more than 500 references to published article in the period 1978–1992 and many more articles appeared since.

In all the DEA models discussed above, all efficient DMUs receive an equal score of 1. An important modification proposed in [1] allows to rank efficient units. The main idea is to exclude the column being scored from the DEA envelopment LP technology matrix. The efficiency score is now a value between (0, +∞] in both orientations. In [5] are discussed in detail the issues (infeasibility, relationship between modified and standard formulation, degeneracy, interpretation of the optimal solutions) related to these DEA models.

In [8] and [9] the properties of 'unit invariance' (independence of the units in which inputs and outputs are measured) and 'translation invariance' (independence of an affine translation of the inputs and the outputs) of an efficiency DEA measure are discussed. The translation invariance property is particularly important when data contain zero or negative values. Standard DEA models are not unit invariant and translation invariant. In [8] it is proposed a weighted additive DEA model that satisfies these properties:

$$\begin{cases} \min\limits_{\lambda,s^+,s^-} & \sum\limits_{i=1}^{m_1} w_i^+ s_i^+ + \sum\limits_{r=1}^{m_2} w_r^- s_r^- \\ \text{s.t.} & \sum\limits_{j=1}^{n} x_i^j \lambda_j + s_i^+ = x_i^{j*} \\ & i = 1, \ldots, m_1 \\ & \sum\limits_{j=1}^{n} y_r^j \lambda_j - s_r^- = y_r^{j*} \\ & r = 1, \ldots, m_2 \\ & \sum\limits_{j=1n}^{n} \lambda_j = 1 \\ & \lambda_j \geq 0, \quad j = 1, \ldots, n, \\ & s_i^+ \geq 0, \quad i = 1, \ldots, m_1, \\ & s_r^- \geq 0, \quad r = 1, \ldots, m_2. \end{cases} \tag{5}$$

where w_i^+ and w_r^- are the sample standard deviation of the inputs and outputs variables respectively.

Models based on data envelopment analysis have been widely used in order to evaluate efficiency in both public and private sectors. [3, Part II] contains 15 application of DEA showing the 'range, power, elegance and insight obtainable via DEA analysis'. Banks, hospitals, and universities are among the most challenging sectors where models based on DEA have been able to assess efficiency and determine strength and weakness of the various units.

See also

► Optimization and Decision Support Systems

References

1. Andersen P, Petersen NC (1993) A procedure for ranking efficient units in data envelopment analysis. mansci 10:1261–1264
2. Banker RD, Charnes A, Cooper WW (1984) Some models for estimating technological and scale inefficiencies in data envelopment analysis. mansci 30:1078–1092
3. Charnes A, Cooper W, Lewin AY, Seiford LM (1994) Data envelopment analysis: Theory, methodology and applicastions. Kluwer, Dordrecht
4. Charnes A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision making units. ejor 2:429–444
5. Dulá JH, Hickman BL (1997) Effects of excluding the column being scored from the DEA envelopment LP technology matrix. jors 48:1001–1012
6. Farrell MJ (1957) The measurement of productive efficiency. J Royal Statist Soc A120:253–281
7. Seiford LM (1994) A DEA bibliography (1978–1992). In: Charnes A, Cooper W, Lewin AY, Seiford LM (eds) Data Envelopment Analysis: Theory, Methodology and Applicastions. Kluwer, Dordrecht, pp 437–469
8. Pastor JT (1994) New DEA additive model for handling zero and negative data. Techn Report Dept Estadistica e Invest. Oper Univ Alicante
9. Pastor JT, Knox Lovell CA (1995) Units invariant and translation invariant DEA models. orl 18:147–151
10. Shepard RW (1953) Cost and production functions. Princeton Univ. Press, Princeton
11. Shepard RW (1970) The theory of cost and production functions. Princeton Univ Press, Princeton

Data Mining

DAVID L. OLSON
Department of Management, University of Nebraska, Lincoln, USA

Article Outline

Synonyms

Data mining; Large-scale data analysis; Pattern recognition

Introduction

Data mining has proven valuable in almost every aspect of life involving large data sets. Data mining is made possible by the generation of masses of data from computer information systems. In engineering, satellites stream masses of data down to storage systems, yielding a mountain of data that needs some sort of data mining to enable humans to gain knowledge. Data mining has been applied in engineering applications such as quality [4], manufacturing and service [13], labor scheduling [17], and many other places. Medicine has been an extensive user of data mining, both in the technical area [21] and in health policy [6]. Pardalos [16] provide recent research in this area. Governmental operations have received support from data mining, primarily in the form of fraud detection [9].

In business, data mining has been instrumental in customer relationship management [5,8], financial analysis [3,12], credit card management [1], health service debt management [22], banking [19], insurance [18], and many other areas of business involving services. Kusiak [13] reviewed data mining applications to include service applications of operations. Recent reports of data mining applications in web service and technology include Tseng and Lin [20] and Hou and Yang [11]. In addition to Tseng and Lin, Lee et al. [14] discuss issues involving mobile technology and data mining. Data mining support is required to make sense of the masses of business data generated by computer technology. Understanding this information-generation system and tools available leading to analysis is fundamental for business in the 21st century. The major applications have been in customer segmentation (by banks and retail establishments wishing to focus on profitable customers) and in fraud and rare event detection (especially by insurance and government, as well as by banks for credit scoring). Data mining has been used by casinos in customer management, and by organizations evaluating personnel.

We will discuss data mining functions, data mining process, data systems often used in conjunction with data mining, and provide a quick review of software tools. Four prototypical applications are given to demonstrate data mining use in business. Ethical issues will also be discussed.

Definitions

There are a few basic functions that have been applied in business. Bose and Mahapatra [2] provided an extensive list of applications by area, technique, and problem type.

- **Classification** uses a training data set to identify classes or clusters, which then are used to categorize data. Typical applications include categorizing risk and return characteristics of investments, and credit risk of loan applicants. The Adams [1] case, for example, involved classification of loan applications into groups of expected repayment and expected problems.
- **Prediction** identifies key attributes from data to develop a formula for prediction of future cases, as in regression models. The Sung et al. [19] case predicted bankruptcy while the Drew et al. [5]) case and the customer retention part of the Smith et al. [18] case predicted churn.
- **Association** identifies rules that determine the relationships among entities, such as in market basket analysis, or the association of symptoms with diseases. IF–THEN rules were shown in the Sung et al. [19] case.
- **Detection** determines anomalies and irregularities, valuable in fraud detection. This was used in claims analysis by Smith et al. [18].

To provide analysis, data mining relies on some fundamental analytic approaches. Regression and neural network approaches are alternative ways to identify the best fit in a given set of data. Regression tends to have advantages with linear data, while neural network models do very well with irregular data. Software usually allows the user to apply variants of each, and lets the analyst select the model that fits best. Cluster analysis, discriminant analysis, and case-based reasoning seek to assign new cases to the closest cluster of past observations. Rule induction is the basis of decision tree methods of data mining. Genetic algorithms apply to special forms of data, and are often used to boost or improve the operation of other techniques.

In order to conduct data mining analyzes, a **data mining process** is useful. The Cross-Industry Standard Process for Data Mining (CRISP-DM) is widely used by industry members [15]. This model consists of six phases intended as a cyclical process:

- **Business understanding:** Business understanding includes determining business objectives, assessing the current situation, establishing data mining goals, and developing a project plan.
- **Data understanding:** Once business objectives and the project plan are established, data understanding considers data requirements. This step can include initial data collection, data description, data exploration, and the verification of data quality. Data exploration such as viewing summary statistics (which includes the visual display of categorical variables) can occur at the end of this phase. Models such as cluster analysis can also be applied during this phase, with the intent of identifying patterns in the data.
- **Data preparation:** Once the data resources available are identified, they need to be selected, cleaned, built into the form desired, and formatted. Data cleaning and data transformation in preparation for data modeling needs to occur in this phase. Data exploration at a greater depth can be applied during this phase, and additional models utilized, again providing the opportunity to see patterns based on business understanding.
- **Modeling:** Data mining software tools such as visualization (plotting data and establishing relationships) and cluster analysis (to identify which variables go well together) are useful for initial analysis. Tools such as generalized rule induction can develop initial association rules. Once greater data understanding is gained (often through pattern recognition triggered by viewing model output), more detailed models appropriate to the data type can be applied. The division of data into training and test sets is also needed for modeling (sometimes even more sets are needed for model refinement).
- **Evaluation:** Model results should be evaluated in the context of the business objectives established in the first phase (business understanding). This will lead to the identification of other needs (often through pattern recognition), frequently reverting to prior phases of CRISP-DM. Gaining business understanding is an iterative procedure in data mining, where the results of various visualization, statistical, and artificial intelligence tools show the user new relationships that provide a deeper understanding of organizational operations.
- **Deployment:** Data mining can be used both to verify previously held hypotheses, and for knowledge discovery (identification of unexpected and useful relationships). Through the knowledge discovered in the earlier phases of the CRISP-DM process, sound models can be obtained that may then be applied to business operations for many purposes, including prediction or identification of key situations. These models need to be monitored for changes in operating conditions, because what might be true today may not be true a year from now. If significant changes do occur, the model should be redone. It is also wise to record the results of data mining projects so documented evidence is available for future studies.

This six-phase process is not a rigid, by-the-numbers procedure. There is usually a great deal of backtracking. Additionally, experienced analysts may not need to apply each phase for every study. But CRISP-DM provides a useful framework for data mining.

There are many **database systems** that provide content needed for data mining. Database software is available to support individuals, allowing them to record information that they consider personally important. They can extract information provided by repetitive organizational reports, such as sales by region within their area of responsibility, and regularly add external data such as industry-wide sales, as well as keep records of detailed information such as sales representative expense account expenditure.

- **Data warehousing** is an orderly and accessible repository of known facts and related data that is used as a basis for making better management decisions. Data warehouses provide ready access to information about a company's business, products, and customers. This data can be from both internal and external sources. Data warehouses are used to store massive quantities of data in a manner that can be easily updated and allow quick retrieval of specific types of data. Data warehouses often integrate information from a variety of sources. Data needs to be identified and obtained, cleaned, catalogued, and stored in a fashion that expedites organizational decision making. Three general data warehouse processes exist. (1) Warehouse generation is the process of designing the warehouse and loading data. (2) Data management is the process of storing the data. (3) Information analysis is the process of using the data to support organizational decision making.

- **Data marts** are sometimes used to extract specific items of information for data mining analysis. Terminology in this field is dynamic, and definitions have evolved as new products have entered the market. Originally, many data marts were marketed as preliminary data warehouses. Currently, many data marts are used in conjunction with data warehouses rather than as competitive products. But also many data marts are being used independently in order to take advantage of lower-priced software and hardware. Data marts are usually used as repositories of data gathered to serve a particular set of users, providing data extracted from data warehouses and/or other sources. Designing a data mart tends to begin with the analysis of user needs. The information that pertains to the issue at hand is relevant. This may involve a specific time-frame and specific products, people, and locations. Data marts are available for data miners to transform information to create new variables (such as ratios, or coded data suitable for a specific application). In addition, only that information expected to be pertinent to the specific data mining analysis is extracted. This vastly reduces the computer time required to process the data, as data marts are expected to contain small subsets of the data warehouse's contents. Data marts are also expected to have ample space available to generate additional data by transformation.
- **Online analytical processing** (OLAP) is a multidimensional spreadsheet approach to shared data storage designed to allow users to extract data and generate reports on the dimensions important to them. Data is segregated into different dimensions and organized in a hierarchical manner. Many variants and extensions are generated by the OLAP vendor industry. A typical procedure is for OLAP products to take data from relational databases and store them in multidimensional form, often called a **hypercube**, to reflect the OLAP ability to access data on these multiple dimensions. Data can be analyzed locally within this structure. One function of OLAP is standard report generation, including financial performance analysis on selected dimensions (such as by department, geographical region, product, salesperson, time, or other dimensions desired by the analyst). Planning and forecasting are supported through spreadsheet analytic tools. Budgeting calculations can also be included through spreadsheet tools. Usually, pattern analysis tools are available.

There are many statistical and analytic **software tools** marketed to provide data mining. Many good data mining software products are being used, including the well-established (and expensive) Enterprise Miner by SAS and Intelligent Miner by IBM, CLEMENTINE by SPSS (a little more accessible by students), PolyAnalyst by Megaputer, and many others in a growing and dynamic industry. For instance, SQL Server 2005 has recently been vastly improved by Microsoft, making a more usable system focused on the database perspective.

These products use one or more of a number of analytic approaches, often as complementary tools that might involve initial cluster analysis to identify relationships and visual analysis to try to understand why data clustered as it did, followed by various prediction models. The major categories of methods applied are regression, decision trees, neural networks, cluster detection, and market basket analysis. The Web site www.KDnuggets.com gives information on many products, classified by function. In the category of overall data mining suites, they list 56 products in addition to 16 free or shareware products. Specialized software products were those using multiple approaches (15 commercial plus 3 free), decision tree (15 plus 10 free), rule-based (7 plus 4 free), neural network (12 plus 3 free), Bayesian (13 plus 11 free), support vector machines (3 plus 8 free), cluster analysis (8 plus 10 free), text mining (50 plus 4 free), and other software for functions such as statistical analysis, visualization, and Web usage analysis.

Example Applications

There are many applications of data mining. Here we present four short examples in the business world.

Customer Relationship Management (CRM)

The idea of customer relationship management is to target customers for special treatment based on their anticipated future value to the firm. This requires estimation of where in the customer life-cycle each subject is, as well as lifetime customer value, based on expected tenure with the company, monthly transactions by that customer, and the cost of providing service. Lifetime value of a customer is the discounted expected stream of cash flow generated by the customer.

Many companies applying CRM score each individual customer by their estimated lifetime value (LTV), stored in the firm's customer database [5]. This concept has been widely used in catalog marketing, newspaper publishing, retailing, insurance, and credit cards. LTV has been the basis for many marketing programs offering special treatment such as favorable pricing, better customer service, and equipment upgrades.

While CRM is very promising, it has often been found to be less effective than hoped [10]. CRM systems can cost up to $70 million to develop, with additional expenses incurred during implementation. Many of the problems in CRM expectations have been blamed on over-zealous sales pitches. CRM offers a lot of opportunities to operate more efficiently. However, they are not silver bullets, and benefits are not unlimited. As with any system, prior evaluation of benefits is very difficult, and investments in CRM systems need to be based on sound analysis and judgment.

Credit Scoring

Data mining can involve model building (extension of conventional statistical model building to very large data sets) and pattern recognition. Pattern recognition aims to identify groups of interesting observations. Interesting is defined as discovery of knowledge that is important and unexpected. Often experts are used to assist in pattern recognition. Adams et al. [1] compared data mining used for model building and pattern recognition on the behavior of customers over a one-year period. The data set involved bank accounts at a large British credit card company observed monthly. These accounts were revolving loans with credit limits. Borrowers were required to repay at least some minimum amount each month. Account holders who paid in full were charged no interest, and thus not attractive to the lender.

We have seen that clustering and pattern search are typically the first activities in data analysis. Then appropriate models are built. Credit scoring is a means to use the results of data mining modeling for two purposes. Application scoring was applied in the Adams et al. example to new cases, continuing an activity that had been done manually for half a century in this organization. Behavioral scoring monitors revolving credit accounts with the intent of gaining early warnings of accounts facing difficulties.

Bankruptcy Prediction

Corporate bankruptcy prediction is very important to management, stockholders, employees, customers, and other stakeholders. A number of data mining techniques have been applied to this problem, including multivariate discriminant analysis, logistical regression, probit, genetic algorithms, neural networks, and decision trees.

Sung et al. [19] applied decision analysis and decision tree models to a bankruptcy prediction case. Decision tree models provide a series of IF–THEN rules to predict bankruptcy. Pruning (raising the proportion of accurate fit required to keep a specific IF–THEN relationship) significantly increased overall prediction accuracy in the crisis period, indicating that data collected in the crisis period was more influenced by noise than data from the period with normal conditions. Example rules obtained were as shown in Table 1, giving an idea of how decision tree rules work.

For instance, in normal conditions, if the variable Productivity of capital (E6) was greater than 19.65, the model would predict firm survival with 86 percent confidence. Conversely, if Productivity of capital (E6) was less than or equal to 19.65, and if the Ratio of cash flow to total assets (C9) was less than or equal to 5.64, the model would predict bankruptcy with 84 percent confidence. These IF–THEN rules are stated in ways that are easy for management to see and use. Here the rules are quite simple, a desirable feature. With large data sets, it is common to generate hundreds of clauses in decision tree rules, making it difficult to implement (although gaining greater accuracy). The number of rules can be controlled through pruning rates within the software.

Fraud Detection

Data mining has successfully supported many aspects of the insurance business, to include fraud detection, underwriting, insolvency prediction, and customer segmentation. An insurance firm had a large data warehouse system recording details on every transac-

Data Mining, Table 1
Bankruptcy Prediction Rules

Condition	Rule	Prediction	Confidence level
Normal	E6>19.65	Nonbankrupt	0.86
Normal	C9>5.64	Nonbankrupt	0.95
Normal	C9≤5.64 & E6≤19.65	Bankrupt	0.84
Crisis	E6>20.61	Nonbankrupt	0.91
Crisis	C8>2.64	Nonbankrupt	0.85
Crisis	C3>87.23	Nonbankrupt	0.86
Crisis	C8≤2.64, E6≤20.61, & C3≤87.23	Bankrupt	0.82

Where C3 = Ratio of fixed assets to equity & long-term liabilities. C8 = Ratio of cash flow to liabilities. C9 = Ratio of cash flow to total assets. E6 = Productivity of capital. Based on Sung et al. [19]

tion and claim [18]. An aim of the analysis was to accurately predict average claim costs and frequency, and to examine the impact of pricing on profitability.

In evaluating claims, data analysis for hidden trends and patterns is needed. In this case, recent growth in the number of policy holders led to lower profitability for the company. Understanding the relationships between cause and effect is fundamental to understanding what business decisions would be appropriate.

Policy rates are based on statistical analysis assuming various distributions for claims and claim size. In this case, clustering was used to better model the performance of specific groups of insured.

Profitability in insurance is often expressed by the cost ratio, or sum of claim costs divided by sum of premiums. Claim frequency ratio is the number of claims divided by the number of policy units of risk (possible claims). Profitability would be improved by lowering the frequency of claims, or the costs of claims relative to premiums.

Data was extracted from the data warehouse for policies for which premiums were paid in the first quarter over a three-year period. This meant that policies were followed over the period, augmented by new policies, and diminished by terminations. Data on each policy holder was available as well as claim behavior over the preceding year. The key variables of cost ratio and claim frequency ratio were calculated for each observation. Sample sizes for each quarter were well above 100,000.

Descriptive statistics found exceptional growth in policies over the past two years for young people (under 22), and with cars insured for over $40,000. Clustering analysis led to the conclusion that the claim cost of each individual policy holder would be pointless, as the vast majority of claims could not be predicted. Af-

Data Mining, Table 2
General Ability of Data Mining Techniques to Deal with Data Features

Data characteristic	Rule induction	Neural networks	Case-based reasoning	Genetic algorithms
Handle noisy data	Good	Very good	Good	Very good
Handle missing data	Good	Good	Very good	Good
Process large data sets	Very good	Poor	Good	Good
Process different data types	Good	Transform to numerical	Very good	Transforma-tion needed
Predictive accuracy	High	Very high	High	High
Explanation capability	Very good	Poor	Very good	Good
Ease of integration	Good	Good	Good	Very good
Ease of operation	Easy	Difficult	Easy	Difficult

Extracted from Bose and Mahapatra [2]

ter experimentation, the study was based on 50 clusters. A basic k-means algorithm was used. This identified several clusters as having abnormal cost ratios or frequency sizes. By testing over a two-year gap, stability for each group was determined. Table 2 compares data mining techniques.

Ethical Issues in Data Mining

Data mining is a potentially useful tool, capable of doing a lot of good, not only for business but also for the medical field and for government. It does, however, bring with it some dangers. So, how can we best protect ourselves, especially in the area of business data mining?

A number of options exist. Strict control of data usage through governmental regulation was proposed by Garfinkel [7]. A number of large database projects that made a great deal of practical sense have ultimately been stopped. Those involving government agencies were successfully stopped due to public exposure, the negative outcry leading to cancellation of the National Data Center and the Social Security Administration projects. A system with closely held information by credit bureaus in the 1960s was only stopped after governmental intervention, which included the passage of new laws. Times have changed, with business adopting a more responsive attitude toward consumers. Innovative data mining efforts by Lotus/Equifax and by Lexis-Nexis were quickly stopped by public pressure alone.

Public pressure seems to be quite effective in providing some control over potential data mining abuses. If that fails, litigation is available (although slow in effect). It is necessary for us to realize what businesses can do with data. There will never be a perfect system to protect us, and we need to be vigilant. However, too much control can also be dangerous, inhibiting the ability of business to provide better products and services through data mining. Garfinkel prefers more governmental intervention, while we would prefer less governmental intervention and more reliance on publicity and, if necessary, the legal system.

Control would be best accomplished if it were naturally encouraged by systemic relationships. The first systemic means of control is publicity. Should those adopting questionable practices persist, litigation is a slow, costly, but ultimately effective means of system correction. However, before taking drastic action, a good rule is that if the system works, it is best not to fix it. The best measure that electronic retailers can take is to not do anything that will cause customers to suspect that their rights are being violated.

Conclusions

Data mining has evolved into a useful analytic tool in all aspects of human study, to include medicine, engineering, and science. It is a necessary means to cope with the masses of data that are produced in contemporary society. Within business, data mining has been especially useful in applications such as fraud detection, loan analysis, and customer segmentation. Such applications heavily impact the service industry. Data mining provides a way to quickly gain new understanding based upon large-scale data analysis.

This paper reviewed some of the applications that have been applied in services. It also briefly reviewed the data mining process, some of the analytic tools available, and some of the major software vendors of general data mining products. Specific tools for particular applications are appearing with astonishing rapidity.

References

1. Adams NM, Hand DJ, Till RJ (2001) Mining for classes and patterns in behavioural data. J Oper Res Soc 52(9):1017–1024
2. Bose I, Mahapatra RK (2001) Business data mining – a machine learning perspective. Inf Manage 39(3):211–225
3. Cowan AM (2002) Data mining in finance: Advances in relational and hybrid methods. Int J Forecasting 18(1):155–156
4. Da Cunha C, Agard B, Kusiak A (2006) Data mining for improvement of product quality. Int J Prod Res 44(18/19):4027–4041
5. Drew JH, Mani DR, Betz AL, Datta P (2001) Targeting customers with statistical and data-mining techniques. J Serv Res 3(3):205–219
6. Garfinkel MS, Sarewitz D, Porter AL (2006) A societal outcomes map for health research and policy. Am J Public Health 96 (3):441–446
7. Garfinkel S (2000) Database Nation: The Death of Privacy in the 21st Century. O'Reilly & Associates, Sebastopol CA
8. Garver MS (2002) Using data mining for customer satisfaction research. Mark Res 14(1):8–12
9. Government Accounting Office (2006) Hurricanes Katrina and Rita: Unprecedented challenges exposed the individ-

uals and households program to fraud and abuse: Actions needed to reduce such problems in future: GAO-06–1013, 9/27/2006, pp 1–110
10. Hart ML (2006) Customer relationship management: Are software applications aligned with business objectives? South African J Bus Manage 37(2):17–32
11. Hou J-L, Yang S-T (2006) Technology-mining model concerning operation characteristics of technology and service providers. Int J Prod Res 44(16):3345–3365
12. Hui W, Weigend AS (2004) Data mining for financial decision making. Decis Support Syst 37(4):457–460
13. Kusiak A (2006) Data mining: Manufacturing and service applications. Int J Prod Res 44(18/19):4175–4191
14. Lee S, Hwang C-S, Kitsuregawa M (2006) Efficient, energy conserving transaction processing in wireless data' broadcast. IEEE Trans Knowl Data Eng 18(9):1225–1238
15. Olson DL, Shi Y (2007) Introduction to Business Data Mining. McGraw-Hill/Irwin, Englewood Cliffs, NJ
16. Pardalos PM, Boginski VL, Vazacopoulos A (eds) (2007) Data Mining in Biomedicine. Springer, Heidelberg
17. Qi X, Bard JF (2006) Generating labor requirements and rosters for mail handlers. Comput Oper Res 33(9):2645–2666
18. Smith KA, Willis RJ, Brooks M (2000) An analysis of customer retention and insurance claim patterns using data mining: A case study. J Oper Res Soc 51(5):532–541
19. Sung TK, Chang N, Lee G (1999) Dynamics of modeling in data mining: Interpretive approach to bankruptcy prediction. J Manage Inf Sys 16(1):63–85
20. Tseng VS, Lin KW (2006) Efficient mining and prediction of user behavior patterns in mobile web systems. Inf Softw Technol 48(6):357–369
21. Yamaguchi M, Kaseda C, Yamazaki K, Kobayashi M (2006) Prediction of blood glucose level of type 1 diabetics using response surface methodology and data mining. Med Biol Eng Comput 44(6):451–457
22. Zurada J, Lonial S (2005) Comparison of the performance of several data mining methods for bad debt recovery in the healthcare industry. J Appl Bus Res 21(2):37–54

D.C. Programming

HOANG TUY
Institute of Mathematics, VAST, Hanoi, Vietnam

MSC2000: 90C26

Article Outline

As optimization techniques become widely used in engineering, economics, and other sciences, an increasing number of nonconvex optimization problems are encountered that can be described in terms of *dc functions* (differences of convex functions). These problems are called dc optimization problems, and the theory dealing with these problems is referred to as dc programming, or dc optimization ([3,4,5,6,13]; see also [1,8]).

Historically, the first dc optimization problem that was seriously studied is the concave minimization problem [11]. Subsequently, reverse convex programming and some other special dc optimization problems such as quadratic and, more generally, polynomial programming problems appeared before a unified theory was developed and the term dc optimization was introduced [12]. In fact, most global optimization problems of interest that have been studied so far can be identified as dc optimization problems, despite the diversity of the approaches used.

DC Structure in Optimization

Let Ω be a convex set in $\mathbb{R}^n$. A function $f\colon \Omega \to \mathbb{R}$ is said to be *dc on* Ω if it can be expressed as the difference of two convex functions on Ω: $f(x) = p(x) - q(x)$, where $p(x), q(x)\colon \Omega \to \mathbb{R}$ are convex. Denote the set of dc functions on Ω by $DC(\Omega)$.

Proposition 1 *$DC(\Omega)$ is a vector lattice with respect to the two operations of pointwise maximum and pointwise minimum.*

In other words, if $f_i(x) \in DC(\Omega)$, $i = 1, \dots, m$, then:
1. $\sum_{i=1}^m \alpha_i f_i(x) \in DC(\Omega)$, for any real numbers α_i;
2. $g(x) = \max\{f_1(x), \dots, f_m(x)\} \in DC(\Omega)$;
3. $h(x) = \min\{f_1(x), \dots, f_m(x)\} \in DC(\Omega)$.

From this property it follows in particular that if $f \in DC(\Omega)$, then $|f| \in DC(\Omega)$, and if $g, h \in DC(\Omega)$, then $gh \in DC(\Omega)$. But for the purpose of optimization the most important consequence is that

$$\begin{aligned} &g_i(x) \le 0, \ \forall i = 1, \dots, m \\ &\quad \Leftrightarrow \quad g(x) := \max\{g_1(x), \dots, g_m(x)\} \le 0\,, \\ &g_i(x) \le 0 \text{ for at least one } i = 1, \dots, m \\ &\quad \Leftrightarrow \quad g(x) := \min\{g_1(x), \dots, g_m(x)\} \le 0\,. \end{aligned}$$

Therefore, any finite system of dc inequalities, whether conjunctive or disjunctive, can be rewritten as a single dc inequality.

By easy manipulations it is then possible to reduce any dc optimization problem to the following *canonical form*:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad g(x) \le 0 \le h(x)\,, \end{aligned} \qquad \text{(CDC)}$$

where all functions f, g, h are convex.

Thus dc functions allow a very compact description of a wide class of nonconvex optimization problems.

Recognizing dc Functions

To exploit the dc structure in optimization problems, it is essential to be able to recognize dc functions that are still in hidden form (i. e., not yet presented as differences of convex functions). The next proposition addresses this question.

Proposition 2 *Every function $f \in C^2$ is dc on any compact convex set Ω.*

It follows that any polynomial function is dc, and hence, by the Weierstrass theorem, $DC(\Omega)$ is dense in the Banach space $C(\Omega)$ of continuous functions on Ω with the supnorm topology. In other words, any continuous function can be approximated as closely as desired by a dc function.

More surprisingly, any closed set S in $\mathbb{R}^n$ can be shown to be a *dc set*, i. e., a set that is the solution set of a dc inequality. Namely, given any closed set $S \subset \mathbb{R}^n$ and any strictly convex function $h\colon \mathbb{R}^n \to \mathbb{R}$, there exists a continuous convex function $g_S\colon \mathbb{R}^n \to \mathbb{R}$ such that $S = \{x \in \mathbb{R}^n\colon\ g_S(x) - h(x) \le 0\}$ [10].

In many situations we not only need to recognize a dc function but also to know how to represent it effectively as a difference of two convex functions. While several classes of functions have been recognized as dc functions [2], there are still few results about effective dc representations of these functions. For composite functions a useful result about dc representation is the following [13].

Proposition 3 *Let $h(x) = u(x) - v(x)$, where $u, v\colon \Omega \to \mathbb{R}_+$ are convex functions on a compact convex set $\Omega \subset \mathbb{R}^m$ such that $0 \le h(x) \le a\ \forall x \in \Omega$. If $q\colon [0, a] \to \mathbb{R}$ is a convex nondecreasing function such that $q'_-(a) < \infty$ ($q'_-(a)$ being the left derivative of $q(t)$ at a), then $q(h(x))$ is a dc function on Ω:*

$$q(h(x)) = g(x) - K[a + v(x) - u(x)]\,,$$

where $g(x) = q(h(x)) + K[a + v(x) - v(x)]$ is a convex function and K is any constant satisfying $K \ge q'_-(a)$.

For example, by writing $x^\alpha = e^{h(x)}$ with $h(x) = \sum_{i=1,\dots,n} \alpha_i \log x_i$ and applying the above proposition, it is easy to see that $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, with $\alpha \in \mathbb{R}^n_+$, is dc on any box $\Omega = [r, s] \subset \mathbb{R}^n_{++}$. Hence, any synomial function $f(x) = \sum_\alpha c_\alpha x^\alpha$, with $c_\alpha \in \mathbb{R}$, $\alpha \in \mathbb{R}^n_+$, is also dc on Ω.

Global Optimality Criterion

A key question in the theoretical as well as computational study of a global optimization problem is how to test a given feasible solution for global optimality.

Consider a pair of problems in some sense mutually obverse:

$$\inf\{f(x)\colon x \in \Omega\,,\ h(x) \ge \alpha\}\,, \qquad (\mathrm{P}_\alpha)$$

$$\sup\{h(x)\colon x \in \Omega\,,\ f(x) \le \gamma\}\,, \qquad (\mathrm{Q}_\gamma)$$

where $\alpha, \gamma \in \mathbb{R}$, Ω is a closed set in $\mathbb{R}^n$, and $f, g\colon \mathbb{R}^n \to \mathbb{R}$ are two arbitrary functions.

We say that problem (P_α) is *regular* if

$$\inf \mathrm{P}_\alpha = \inf\{f(x)\colon x \in \Omega, h(x) > \alpha\}\,. \qquad (1)$$

Analogously, problem (Q_γ) is regular if $\sup \mathrm{Q}_\gamma = \sup\{h(x)\colon x \in \Omega, f(x) < \gamma\}$.

Proposition 4 *Let $\bar{x}$ be a feasible solution of problem (P_α). If $\bar{x}$ is optimal to problem (P_α) and if problem (Q_γ) is regular for $\gamma = f(\bar{x})$, then*

$$\sup\{h(x)\colon\ x \in \Omega,\ f(x) \le \gamma\} = \alpha\,. \qquad (2)$$

Conversely, if (2) holds and if problem (P_α) is regular, then $\bar{x}$ is optimal to (P_α).

Turning now to the canonical dc optimization problem (CDC), let us set $\Omega = \{x \colon g(x) \le 0\}$ and without losing generality assume that the reverse convex constraint $h(x) \ge 0$ is essential, i. e.,

$$\inf\{f(x) \colon x \in \Omega\} < \inf\{f(x) \colon x \in \Omega, h(x) \ge 0\} . \quad (3)$$

Since CDC is a problem (P_α) with $\alpha = 0$, if $\bar{x}$ is a feasible solution to CDC, then condition (3) ensures the regularity of the associated problem (Q_γ) for $\gamma = f(\bar{x})$. Define

$$C = \{x \colon h(x) \le 0\}, \quad D(\gamma) = \{x \in \Omega \colon f(x) \le \gamma\} , \quad (4)$$

and for any set E denote its polar by E^*. As specialized for CDC, Proposition 4 yields:

Proposition 5 *In order that a feasible solution $\bar{x}$ of CDC may be a global minimizer, it is necessary that the following equivalent conditions hold for $\gamma = f(\bar{x})$:*

$$D(\gamma) \subset C , \quad (5)$$

$$0 = \max\{h(x) \colon x \in D(\gamma)\} , \quad (6)$$

$$C^* \subset [D(\gamma)]^* . \quad (7)$$

If the problem is regular, then any one of the above conditions is also sufficient.

An important special dc program is the following problem:

$$\text{minimize} \quad g(x) - h(x) \quad \text{subject to} \quad x \in \mathbb{R}^n , \quad (\text{DC})$$

where $g, h \colon \mathbb{R}^n \to \bar{\mathbb{R}}$ are convex functions ($\bar{R}$ denotes the set of extended real numbers). Writing this problem as $\min\{g(x) - t \colon x \in D, h(x) \ge t\}$ with $D = \operatorname{dom} g \cap \operatorname{dom} h$ and using (7), one can derive the following:

Proposition 6 *Let $g, h \colon \mathbb{R}^n \to \bar{\mathbb{R}}$ be two convex functions such that h(x) is proper and lsc. Let $\bar{x}$ be a point where $g(\bar{x})$ and $h(\bar{x})$ are finite. In order for $\bar{x}$ to be a global minimizer of $g(x) - h(x)$ over $\mathbb{R}^n$, it is necessary and sufficient that*

$$\partial_\varepsilon h(\bar{x}) \subset \partial_\varepsilon g(\bar{x}) \quad \forall \varepsilon > 0 , \quad (8)$$

where $\partial_\varepsilon f(a) = \{p \in \mathbb{R}^n \colon \langle p, x - a\rangle - \varepsilon \le f(x) - f(a) \quad \forall x \in \mathbb{R}^n\}$ is the ε-subdifferential of $f(x)$ at point a.

Solution Methods

Numerous solution methods have been proposed for different classes of dc optimization. Each of them proceeds either by outer approximation (OA) of the feasible set or by branch and bound (BB) or is of a hybrid type, combining OA with BB. Following are some typical dc algorithms.

An OA Method for (CDC)

Without losing generality, assume (3), i. e.,

$$\begin{aligned} &\exists w \quad \text{s.t. } g(w) \le 0 , \\ &\qquad f(w) < \min\{f(x) \colon x \in \Omega, h(x) \ge 0\} . \end{aligned} \quad (9)$$

where, as was defined above, $\Omega = \{x \colon g(x) \le 0\}$. In most cases checking the regularity of a problem is not easy while regularity is needed for the sufficiency of the optimality criteria in Proposition 5. Therefore the method to be presented below only makes use of the necessity part of this proposition and is independent of any regularity assumption.

In practice, what we usually need is not an exact solution but just an approximate solution of the problem. Given tolerances $\varepsilon > 0, \eta > 0$, we are interested in *$\varepsilon$-approximate solutions*, i. e., solutions $x \in \Omega$ satisfying $h(x) \ge -\varepsilon$. An ε-approximate solution x^* is then said to be *η-optimal* if $f(x^*) - \eta \le \min\{f(x) \colon x \in \Omega, h(x) \ge 0\}$.

With $\bar{x}$ now being a given ε-approximate solution and $\gamma = f(\bar{x}) - \eta$, consider the subproblem

$$\max\{h(x) \colon x \in \Omega, f(x) \le \gamma\} . \quad (Q_\gamma)$$

For simplicity assume that the set $D(\gamma) = \{x \in \Omega, f(x) \le \gamma\}$ is bounded. Then (Q_γ) is a convex maximization problem over a compact convex set and can be solved by an OA algorithm (see [13] or [3]) generating a sequence $\{x^k, y^k\}$ such that

$$x^k \in \Omega, f(x^k) \le \gamma, h(x^k) \le \max(Q_\gamma) \le h(y^k) \quad (10)$$

and, furthermore, $\|x^k - y^k\| \to 0$ as $k \to +\infty$. These relations imply that we must either have $h(y^k) < 0$ for some k (which implies that max $(Q_\gamma) < 0$), or else $h(x^k) \ge -\varepsilon$ for some k. In the former case, this means there is no $x \in \Omega$ with $h(x) \geq 0$ and $f(x) \leq f(\bar{x}) - \eta$, i. e., $\bar{x}$ is η-optimal to CDC, and we are done. In the latter case, x^k is an ε-approximate solution with $f(x^k) \leq f(\bar{x}) - \eta$. Using then a local search (or any inexpensive way available) one can improve x^k to $x' \in \Omega \cap \{x \colon h(x) = -\varepsilon\}$, and, after resetting $\gamma \leftarrow f(x') - \eta$ in (Q_γ), one can repeat the procedure with the new (Q_γ). And so on.

As is easily seen, the method consists essentially of a number of consecutive cycles in each of which, say the lth cycle, a convex maximization subproblem (Q_γ) is solved with $\gamma = f(x^l) - \eta$ for some ε-approximate solution x^l. This sequence of cycles can be organized into a unified procedure. For this, it suffices to start each new cycle from the result of the previous cycle: after resetting $\gamma \leftarrow \gamma' := f(x') - \eta$ in (Q_γ), we have $D(\gamma') \subset D(\gamma)$, with a point $x' \notin D(\gamma')$, so the algorithm can be continued by using a hyperplane separating x' from $D(\gamma')$ to form with the current polytope outer approximating $D(\gamma)$ a smaller polytope outer approximating $D(\gamma')$. Since each cycle decreases the objective function value by at least a quantity $\eta > 0$, and the objective function is bounded from below, the whole procedure must terminate after finitely many cycles, yielding an ε-approximate solution that is η-optimal to (CDC).

It is also possible to use a BB algorithm for solving the subproblem (Q_γ) in each cycle. The method then proceeds exactly as in the BB method for GDC to be presented next.

A BB Method for General DC Optimization

A general dc optimization problem can be formulated as

$$\min\{f(x) \colon g_i(x) \geq 0, \quad i = 1, \dots, m, x \in \Omega\}, \qquad \text{(GDC)}$$

where Ω is a compact convex subset of $\mathbb{R}^n$, and $f, g_1, \dots, g_m$ are dc functions on Ω. Although in principle GDC can be reduced to the canonical form and solved as a CDC problem, this may not be an efficient method as it does not take account of specific features of GDC. For instance, if the feasible set of GDC is highly nonconvex, computing a single feasible solution may be as hard as solving the problem itself. Under these conditions, a direct application of the OA or the BB strategies to GDC is fraught with pitfalls. Without adequate precautions, such approaches may lead to grossly incorrect results or to an unstable solution that may change drastically upon a small change of the data or the tolerances [15,16].

A safer approach is to reduce GDC to a sequence of problems with a convex feasible set in the following way. By simple manipulations it is always possible to arrange that the objective function $f(x)$ is convex. Let $g(x) = \min_{i=1,\dots,m} g_i(x)$, and for every $\gamma \in \mathbb{R} \cup \{+\infty\}$ consider the subproblem

$$\max\{g(x) \colon x \in \Omega, f(x) \leq \gamma\}. \qquad (\mathrm{R}_\gamma)$$

Assuming the set $D(\gamma) := \{x \in \Omega, f(x) \leq \gamma\}$ to be bounded, we have in (R_γ) a dc optimization over a compact convex set. Using a *BB procedure* to solve (R_γ) we generate a nested sequence of partition sets M_k (boxes, e. g., using a rectangular subdivision), together with a sequence $\alpha(M_k) \in \mathbb{R} \cup \{-\infty\}$, and $x^k \in \mathbb{R}^n$, $k = 1, 2, \dots$, such that

$$\operatorname{diam} M_k \to 0 \text{ as } k \to +\infty, \qquad (11)$$

$$\alpha(M_k) \searrow \max\{g(x) \colon x \in M_k \cap D(\gamma)\} (k \to +\infty), \qquad (12)$$

$$\alpha(M_k) \geq \max(\mathrm{R}_\gamma), x^k \in M_k \cap D(\gamma), \qquad (13)$$

where max(P) denotes, as usual, the optimal value of problem P. Condition (11) means that the subdivision rule used must be exhaustive, while (12) indicates that $\alpha(M_k)$ is an upper bound over the feasible solutions in M_k, and (13) follows from the fact that M_k is the partition set with the largest upper bound among all partition sets currently of interest.

As before, we say that x is an *ε-approximate solution* of GDC if $x \in \Omega$, $g(x) \geq -\varepsilon$ and x^* is η-optimal if $f(x^*) - \eta \leq \min\{f(x) \colon g(x) \geq 0, x \in \Omega\}$. From (11)–(13) it follows that $\alpha(M_k) - g(x^k) \to 0$ as $k \to +\infty$, and hence, for any given $\varepsilon > 0$, either $\alpha(M_k) < 0$ for some k or $g(x^k) \geq -\varepsilon$ for some k. In the former case, $\max(\mathrm{R}_\gamma) < 0$, hence $\max(\mathrm{GDC}) > \gamma$; in the latter case, x^k is an ε-approximate solution of GDC with $f(x^k) \leq \gamma$. So, given any ε-approximate solution $\bar{x}$ with $\gamma = f(\bar{x}) - \eta$, a finite number of iterations of this BB procedure will help to determine whether there is no feasible solution x to GDC with $f(x) \leq f(\bar{x}) - \eta$, i.e., $\bar{x}$ is η-*optimal* to GDC, or else there exists an ε-approximate solution x' to GDC with $f(x') \leq f(\bar{x}) - \eta$. In the latter case, we can reset $f(x') - \eta \leftarrow \gamma$ and repeat the procedure with the new γ, and so on. In this way the whole solution process consists of a number of cycles, each involving a finite BB procedure and giving a decrease in the incumbent value of $f(x)$ by at least $\eta > 0$. By starting each cycle right from the result of the previous one, the sequence of cycles forms a unified procedure. Since η is a positive constant, the number of cycles is finite and the procedure terminates with an ε-approximate solution that is η-optimal to GDC.

The efficiency of such a BB procedure depends on two basic operations: branching and bounding. Usually, branching is performed by means of an exhaustive subdivision rule, so as to satisfy condition (11). For rectangular partition, this condition can be achieved by the standard bisection rule: bisect the current box M into two equal subboxes by means of a hyperplane perpendicular to a longest edge of M at its midpoint. However, it has been observed that the convergence guaranteed by an exhaustive subdivision rule is rather slow, especially in high dimensions. To improve the situation, the idea is to use, instead of the standard bisection, an *adaptive subdivision rule* defined as follows. Let the upper bound $\alpha(M_k)$ in (12) be obtained as $\alpha(M_k) = \max\{\Gamma(x) : x \in M_k \cap D(\gamma)\}$, where $\Gamma(x)$ is some concave overestimator of $g(x)$ over M_k that is tight at some point $y^k \in M_k$, i.e., satisfies $\Gamma(y^k) = g(y^k)$. If $x^k \in \operatorname{argmax}\{\Gamma(x) | x \in M_k \cap D(\gamma)\}$, then the subdivision rule is to bisect M_k by means of the hyperplane $x_s = x_s^k + y_s^k/2$, where $s \in \operatorname{argmax}_{i=1,\dots,n} |y_i^k - x_i^k|$. As has been proved in [13], such an adaptive bisection rule ensures the existence of an infinite subsequence $\{k_\nu\}$ such that $y^{k_\nu} - x^{k_\nu} \to 0$ as $\nu \to +\infty$. The common limit x^* of x^{k_ν} and y^{k_ν} then yields an optimal solution of the problem (R_γ). Computational experience has effectively confirmed that convergence achieved with an adaptive subdivision rule is usually much faster than with the standard bisection. For such an adaptive subdivision to be possible, the constraint set $D(\gamma)$ of (12) must be convex, so that for each partition set M_k two points $x^k \in M_k \cap D(\gamma)$ and $y^k \in M_k$ can be defined such that $\alpha(M_k) - g(y^k) = o(\|x^k - y^k\|)$.

DCA–A Local Optimization Approach to (DC)

By rewriting DC as a canonical dc optimization problem

$$\min\{t - h(x) : x \in \mathbb{R}^n, t \in \mathbb{R}, g(x) - t \leq 0\} ,$$

we see that DC can be solved by the same method as CDC. Since, however, for some large-scale problems we are not so much interested in a global optimal solution as in a sufficiently good feasible solution, a local optimization approach to DC has been developed [9] that seems to perform quite satisfactorily in a number of applications. This method, referred to as DCA, is based on the well-known Toland equality:

$$\inf_{x \in \operatorname{dom} g} \{g(x) - h(x)\} = \inf_{y \in \operatorname{dom} h^*} \{g^*(y) - g^*(y)\}, \quad (14)$$

where $g, h : \mathbb{R}^n \to \mathbb{R}$ are lower semicontinuous proper convex functions, and the star denotes the conjugate, e.g., $g^*(y) = \sup\{\langle x, y\rangle - g(x) : x \in \operatorname{dom} g\}$. Taking account of this equality, DCA starts with $x^0 \in \operatorname{dom} g$ and for $k = 1, 2, \dots$, computes $y^k \in \partial h(x^k); x^{k+1} \in \partial g^*(y^k)$. As has been proved in [9], the thus generated sequence x^k, y^k satisfies the following conditions:

1. The sequences $g(x^k) - h(x^k)$ and $h^*(x^k) - g(^*(x^k)$ are decreasing.
2. Every accumulation point x^* (resp. y^*) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of the function $g(x) - h(x)$ (resp. $h^*(y) - g^*(y)$).

Though global optimality cannot be guaranteed by this method, it has been observed that in many cases of interest it yields a local minimizer that is also global.

Applications and Extensions

The above described dc methods are of a general-purpose type. For many special dc problems more efficient algorithms are needed to take full advantage of additional structures. Along this line, dc methods have been adapted to solve problems with separated nonconvexity, bilinear programming, multilevel programming, multiobjective programming, optimization problems over efficient sets, polynomial and synomial programming, fractional programming, continuous location problems, clustering and datamining problems, etc. [4]. In particular, quite efficient methods have been developed for a class of dc optimization problems important for applications called multiplicative programming [4,5]. Also, techniques for bounding, branching, and decomposition have been refined that have very much widened the range of applicability of dc methods. Most recently, *monotonic optimization*, also called *DM optimization*, has emerged as a new promising field of research dealing with a class of optimization problems important for applications whose structure, though different from the dc structure, shares many common features with the latter. To be specific, let C be a family of real valued functions on $\mathbb{R}^n$ such that (i) $g_1, g_2 \in C, \alpha_1, \alpha_2 \in \mathbb{R}_+ \Rightarrow \alpha_1 g_1 + \alpha_2 g_2 \in C$; (ii) $g_1, g_2 \in C \Rightarrow g(x) := \max\{g_1(x), g_2(x)\} \in C$. Then the family $\mathcal{D}(C) = C - C$ is a vector lattice with respect to the two operations of pointwise maximum and pointwise minimum. When C is the set of convex functions, $\mathcal{D}(C)$ is nothing but the vector lattice of dc functions. When C is the set of increasing functions on $\mathbb{R}^n$, i. e., the set of functions $f : \mathbb{R}^n \to \mathbb{R}$ such that $x' \geq x \Rightarrow f(x') \geq f(x)$, the vector lattice $\mathcal{D}(C)$ consists of *DM functions*, i. e., functions representable as the difference of two increasing functions. For the theory, methods, and algorihms of DM optimization, we refer the reader to [7,14,18].

References

1. Floudas CA (2000) Deterministic Global Optimization. Kluwer, Dordrecht
2. Hartman P (1959) On Functions Representable as a Difference of Convex Functions. Pacific J Math 9:707–713
3. Horst R, Tuy H (1996) Global Optimization (Deterministic Approaches), 3rd edn. Springer, Berlin
4. Horst R, Pardalos PM (eds) (1995) Handbook of Global Optimization. Kluwer, Dordrecht
5. Konno H, Thach PT, Tuy H (1997) Optimization on Low Rank Nonconvex Structures. Kluwer, Dordrecht
6. Pardalos PM, Rosen JB (1987) Constrained Global Optimization: Algorithms and Applications. Lecture Notes in Computer Sciences 268. Springer, Berlin
7. Rubinov A (1999) Abstract Convexity and Global Optimization. Kluwer, Dordrecht
8. Sherali HD, Adams WP (1999) A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Dordrecht
9. Tao PD, An LTH (1997) Convex analysis approach to D.C. Programmng: Theory, algorithms and applications. Acta Mathematica Vietnamica 22:289–356
10. Thach PT (1993) D.c. sets, dc functions and nonlinear equations. Math Programm 58:415–428
11. Tuy H (1964) Concave programming under linear constraints. Soviet Math 5:1437–1440
12. Tuy H (1985) A general deterministic approach to global optimization via dc programming. In: Hiriart-Urruty JB (ed) Fermat Days 1985: Mathematics for Optimization. North-Holland, Amsterdam, pp 137–162
13. Tuy H (1998) Convex Analysis and Global Optimization. Kluwer, Dordrecht
14. Tuy H (2000) Monotonic Optimization: Problems and Solution Approaches. SIAM J Optim 11(2):464–494
15. Tuy H (2005) Robust Solution of Nonconvex Global Optimization Problems. J Global Optim 32:307–323
16. Tuy H (2005) Polynomial Optimization: A Robust Approach. Pacific J Optim 1:357–373
17. Tuy H, Al-Khayyal FA, Thach PT (2005) Monotonic Optimization: Branch and Cuts Methods. In: Audet C, Hansen P, Savard G (eds) Essays and Surveys on Global Optimization. GERAD. Springer, Berlin, pp 39–78
18. Tuy H, Minoux M, NTH Phuong (2006) Discrete Monotonic Optimization with Application to A Discrete Location Problem. SIAM J Optim 1778–97

Decision Support Systems with Multiple Criteria

CONSTANTIN ZOPOUNIDIS, MICHAEL DOUMPOS
Department Production Engineering and Management
Financial Engineering Lab. Techn., University Crete,
Chania, Greece

MSC2000: 90C29

Article Outline

Keywords

Decision support system; Multicriteria analysis;
Multicriteria group decision support system;
Intelligent multicriteria decision support systems

In practical real-world situations the available time for making decisions is often limited, while the cost of investigation is increasing with time. Therefore, it would be enviable to exploit the increasing processing power provided by the modern computer technology, to save significant amounts of time and cost in decision making problems. Computationally intensive, but routine tasks, such as data management and calculations can be performed with remarkable speed by a common personal computer, compared to the time that a human would need to perform the same tasks. On the other hand, computers are unable to perform cognitive tasks, while their inference and reasoning capabilities are still very limited compared to the capabilities of the human brain. Thus, in decision making problems, computers can support decision makers by managing the data of the problem and performing computationally intensive calculations, based on a selected decision model, which could help in the analysis, while the decision makers themselves have to examine the obtained results of the models and conclude to the most appropriate decision.

This merging of human judgment and intuition together with computer systems constitutes the underlying philosophy, methodological framework and basic goal of *decision support systems* [17]. The term 'decision support system' (DSS) is already consolidated and it is used to describe any computer system that provides information on a specific decision problem using analytical decision models and access to databases, in order to support a decision maker in making decisions effectively in complex and ill-structured problems where no straightforward, algorithmic procedure can be employed [28].

The development of DSSs kept pace with the advances in computer and information technologies, and since the 1970s numerous DSSs have been designed by academic researchers and practitioners for the examination and analysis of several decision problems including finance and accounting, production management, marketing, transportation, human resources management, agriculture, education, etc. [17,19].

Except for the specific decision problems that DSSs address, these systems are also characterized by the type of decision models and techniques that they incorporate (i. e. statistical analysis tools, mathematical programming and optimization techniques, multicriteria decision aid methods, etc.). Some of these methodologies (optimization, statistical analysis, etc.) which have already been implemented in several DSSs, are based on the classical monocriterion approach. However, real world decision problems can be hardly considered through the examination of a single criterion, attribute or point of view that will lead to the 'optimum' decision. In fact such a monocriterion approach is merely an oversimplification of the actual nature of the problem at hand, that can lead into unrealistic decisions.

On the other hand, a more realistic and flexible approach would be the simultaneous consideration of all pertinent factors that may affect a decision. However, through this appealing approach a very essential issue emerges: how can several and often conflicting factors can be aggregated to make rational decisions? This issue constitutes the focal point of interest for all the *multicriteria decision aid* methods. The incorporation of multicriteria decision aid methods in DSSs provides the decision makers with a highly efficient tool to study complex real world decision problems where multiple criteria of conflicting nature are involved. Therefore, the subsequent sections of this paper will concentrate on this specific category of DSSs (*multicriteria DSSs*, MCDSSs).

The article is organized as follows. In section 2 some basic concepts, notions and principles of multicriteria decision aid are discussed. Section 3 presents the main features and characteristics of MCDSSs, along with a review of the research that has been conducted in this field, while some extensions of the classical MCDSSs framework in group decision making and intelligent decision support are also discussed. Finally, section 4 concludes the paper and outlines some possible future research directions in the design, development and implementation of MCDSSs.

Multicriteria Decision Aid

Multicriteria decision aid (MCDA, the European School) or multicriteria decision making (MCDM, the American School) [49,64] constitutes an advanced field of operations research which is devoted to the development and implementation of decision support methodologies to confront complex decision problems involving multiple criteria, goals or objectives of conflicting nature. The foundations of MCDA can be traced back in the works of J. von Neumann and O. Morgenstern [43], and P.C. Fishburn [20] on utility theory, A. Charnes and W.W. Cooper [10] on goal programming, and B. Roy [47] on the concept of outranking relations and the foundations of the ELECTRE methods. These pioneering works have affected the subsequent research in the field of MCDA that can be distinguished in two major groups: discrete and continuous MCDA. The former is involved with decision problems where there is a finite set of alternatives which should be considered in order to select the most appropriate one, to rank them from the best to the worst, or to classify them in predefined homogeneous classes. On the contrary in continuous MCDA problems the alternatives are not defined a priori, but instead one seeks to construct an alternative that meets his/her goals or objectives (for instance the construction of a portfolio of stocks).

There are different ways to address these two classes of problems in MCDA. Usually, a continuous MCDA problem is addressed through *multi-objective* or *goal programming* approaches. In the former case, the objectives of the decision maker are expressed as a set of linear or non linear functions which have to be 'optimized', whereas in the latter case the decision maker expresses his/her goals in the form of a reference or ideal point which should be achieved as close as possible. These two approaches extend the classical single-objective optimization framework, through the simultaneous consideration of more than one objectives or goals. Of course in this new context it seems illusory to speak of optimality, but instead the aim is initially to determine the set of efficient solutions (solutions which are not dominated by any other solution) and then to identify interactively a specific solution which is consistent with the preference structure of the decision maker. The books [54,57] and [63] provide an excellent and extensive discussion of both multi-objective and goal programming.

On the other hand, discrete MCDA problems are usually addressed through the *multi-attribute utility theory* (MAUT) [26], the *outranking relations approach* [48] or the *preference disaggregation approach* ([23,44]). These three approaches are mainly focused on the determination and modeling of the decision makers' preferences, in order to develop a global preference model which can be used in decision making. Their differences concern mainly the form of the global preference model that is developed, as well as the procedure that is used to estimate the parameters of the model. The developed preference model in both MAUT and preference disaggregation is a utility or value function either additive or multiplicative, whereas the outranking relations approach is based on pairwise comparisons of the form 'alternative a is at least as good as alternative b'. Concerning the procedure that is used to estimate the parameters of the global preference model, both in MAUT and outranking relations there is a direct interrogation of the decision maker. More precisely, in MAUT the decision maker is asked to determine the trade-offs among the several attributes or criteria, while in outranking relations the decision maker has to determine several parameters, such as the weights of the evaluation criteria, indifference, strict preference and veto thresholds for each criterion. On the contrary, in preference disaggregation, an ordinal regression procedure is used to estimate the global preference model. Based on a reference set of alternatives, which may consist either of past decisions or by a small subset of the alternatives under consideration, the decision maker is asked to provide a ranking or a classification of the alternatives according to his/her decision policy (global preferences). Then, using an ordinal regression procedure the global preference model is estimated so that the original ranking or classification (and consequently the global preference system of the decision maker) can be reproduced as consistently as possible.

Multicriteria Decision Support Systems

From the above brief discussion of the basic concepts and approaches of MCDA, it is clear that in any case the decision maker and his/her preferences constitute the focal point of the methodological framework of MCDA. This special characteristic of MCDA implies that a comprehensive model of a decision situation cannot be developed, but instead the model should be developed to meet the requirements of the decision maker [46]. The development of such a model can be only achieved through an iterative and interactive process, until the decision maker's preferences are consistently represented in the model. Both interactivity and iterative operation are two of the key characteristics of DSSs. Consequently, a DSS incorporating MCDA methods could provide essential support in structuring the decision problem, analyzing the preferences of the decision maker, and supporting the model building process.

The support provided by multicriteria DSSs (MCDSSs) is essential for the decision maker as well as for the decision analyst.

- The decision maker through the use of MCDSSs becomes familiar with sophisticated operations research techniques, he is supported in structuring the decision problem considering all possible points of view, attributes or criteria, and furthermore, he is able to analyze the conflicts between these points of view and consider the existing trade-offs. All these capabilities provided by MCDSSs serve the learning process of decision makers in resolving complex decision problems in a realistic context, and constitute a solid scientific basis for arguing upon the decisions taken.
- On the other hand, from the decision analyst point of view, MCDSSs provide a supportive tool which is necessary throughout the decision making process, enabling the decision analyst who usually acts as an intermediate between the system and the decision maker, to highlight the essential features of the problem to the decision maker, to introduce the preferences of the decision maker in the system, and to develop the corresponding model. Furthermore, through sensitivity and robustness analyses the decision analyst is able to examine several scenarios, concerning both the significance of the evaluation criteria as well as the changes in the decision environment.

The supportive operation of MCDSSs in making decisions in ill-structured complex decision problems was the basic motivation for computer scientists, management scientists and operations researchers in the development of such systems. Actually, MCDSSs are one of the major areas of DSSs research since the 1970s [19] and significant progress has been made both on the theoretical and the practical/implementation viewpoints.

The first MCDSSs to be developed in the 1970s where mainly oriented towards the study of multiobjective mathematical programming problems ([16,61]). These early pioneer systems, mainly due to the limited capabilities of computer technology during that period, were primarily developed for academic purposes, they were implemented in mainframe computers, with no documentation available, while they had no visual representation capabilities [31]. Today, after more than twenty years of research and advances in MCDA, DSSs, and computer science, most MCDSSs provide many advanced capabilities to decision makers including among others [46]:

1) Enhanced data management capabilities including interactive addition, deletion or modification of criteria.
2) Assessment and management of weights.
3) User-friendly interfaces based on visual representations of both alternatives and criteria to assist the interaction between the system and the decision maker.
4) Sensitivity analysis (what-if analysis) to determine how the changes in the weights of the evaluation criteria can affect the actual decision.

These capabilities are in accordance with the general characteristics of DSSs, that is interactivity, flexibility and adaptability to the changes of the decision environment, user oriented design and development, and combination of data base management with decision models. Although the aforementioned capabilities are common to most of the existing MCDSSs, one could provide a distinction of the MCDSSs according to the MCDA approaches that they employ:

- MCDSSs based on the multi-objective programming approach:
  - the TOMMIX system [2],
  - the TRIMAP system [11],
  - the VIG system ([29,32]),
  - the VIDMA system [30],
  - the DIDAS system [36],
  - the AIM system [37],
  - the ADBASE system [58], and
  - the STRANGE system [59].
- MCDSSs based on the MAUT approach:
  - the MACBETH system [5],
  - the VISA system [6], and
  - the EXPERT CHOICE system [21].
- MCDSSs based on the outranking relations approach:
  - the PROMCALC and GAIA systems [7],
  - the ELECCALC system [27],
  - the PRIAM system [34], and
  - the ELECTRE TRI system [62].
- MCDSSs based on the preference disaggregation approach:
  - the PEFCALC system [22],
  - the MINORA system [51],

- the MIIDAS system [52], and
- the PREFDIS system [66].

Most of the existing MCDSSs are designed for the study of general multicriteria decision problems. Although they provide advanced capabilities for modeling the decision makers' preferences in order to make a specific decision regarding the choice of an alternative and the ranking or the classification of the alternatives, MCDSSs do not consider the specific characteristics, as well as the nature of the decision that should be taken according to the specific decision problem that is considered.

To address the unique nature of some significant decision problems, where except for the application of MCDA methodology, some other type of analyses are necessary to consider the environment in which the decision is taken, several authors proposed domain specific MCDSSs. Some decision problems for which specific MCDSSs have been developed include the assessment of corporate performance and viability (the BANKADVISER system [39], the FINCLAS system [65], the FINEVA system [68], and the system proposed in [53]), bank evaluation (the BANKS system [40]), bank asset liability management [33], financial planning [18], portfolio selection [67], new product design (the MARKEX system [42]), urban planning (the system proposed in [1]), strategic planning [9], and computer system design [15].

Multicriteria Group Decision Support Systems

A common characteristic of all the aforementioned MCDSSs is that they refer to decisions that are taken by individual decision makers. However, in many cases the actual decision is not the responsibility of an individual, but instead there is a team of negotiating or cooperative participants who must conclude to a consensus decision. In this case, although the decision process and consequently the required decision support, remains the same, as far as each individual decision maker is concerned, the process that will lead the cooperative team or the negotiating parties to a consensus decision is completely different from the individual decision making process. Therefore, the type of support needed also differs.

Group DSSs (GDSSs) aim at supporting such decision processes, and since the tools provided by MCDA can be extended to generalized group decision process, several attempts have been made to design and develop such multicriteria systems. Some examples of *multicriteria GDSSs* include the Co-oP system [8], the JUDGES system [12], the WINGDSS system [13], the MEDIATOR system [24], and the SCDAS system [35].

Intelligent Multicriteria Decision Support Systems

Except for the extension of the MCDSSs framework in supporting group decision making, recently researchers have also investigated the extension of MCDSSs through the exploitation of the advances in the field of artificial intelligence. Scientific fields such as those of neural networks, expert systems, fuzzy sets, genetic algorithms, etc., provide promising features and new capabilities regarding the representation of expert knowledge, the development of intelligent and more friendly user interfaces, the reasoning and explanation abilities, as well as the handling of incomplete, uncertain and imprecise information.

These appealing new capabilities provided by artificial intelligence techniques can be incorporated in the existing MCDSSs framework to provide expert advice on the problem under consideration, assistance to the use of the several modules of the system, explanations concerning the results MCDA, models, support on structuring the decision making process, as well as recommendations and further guidance for the future actions that the decision maker should take in order to implement successfully his/her decisions. The terms '*intelligent multicriteria decision support systems*' or 'knowledge-based multicriteria decision support systems' have been used by several authors to describe MCDSSs which take advantage of artificial intelligence techniques in combination with MCDA methods.

Some representative examples of intelligent MCDSSs are, the system proposed in [3] for multiobjective linear programming, the MARKEX system for new product design [42], the CREDEX system [45] and the CGX system [55] for credit granting problems, the MIIDAS system for estimating additive utility functions based on the preference disaggregation approach [52], the INVEX system for investment analysis [60] based on the PROMETHEE method, as well as the FINEVA system [68] for the assessment of corporate performance and viability. All these systems incorporate in their structure one or more expert system components either to derive estimations regarding the problem under consideration (FINEVA, MARKEX, CREDEX, CGX, INVEX systems) or to support the use of the MCDA models which are incorporated in the system and generally support and improve the communication between the user and the system (MIIDAS and MARKEX systems). Furthermore, the INVEX system incorporates fuzzy sets to provide an initial distinction between good and bad investment projects, so that the number of alternatives to be considered latter on in the multicriteria analysis module is reduced.

The ongoing research on the integration of artificial intelligence with MCDA regarding the theoretical foundations of this integration and the related implementation issues ([4,25]), the construction of fuzzy outranking relations ([14,41,50]), and the applications of neural networks in preference modeling and utility assessment ([38,56]) constitutes a significant basis for the design and development of intelligent MCDSSs implementing the theoretical findings of this research.

Conclusions

This article investigated the potentials provided by MCDSSs in the decision making process. MCDSSs during the last two decades have consolidated their position within the operations research, information systems and management science communities as an efficient tool for supporting the whole decision making process beginning from problem structuring until the implementation of the final decision, in complex ill-structured problems.

The review which was presented in this paper reveals that recent advances in MCDSSs include systems for general use to solve both discrete and continuous MCDA problems, systems designed to study some specific real world decisions, as well as systems designed to support negotiation and group decision making.

As the computer science and technology progresses rapidly, new areas of applications of MCDSSs can be explored including their operation over the Internet to provide computer support to co-operative work of dispersed and asynchronous decision units. The incorporation of artificial intelligence techniques in the existing framework of MCDSSs also constitutes another significant area of future research. Although, as its has been illustrated in this paper, researchers have already tried to integrate these two approach in an integrated intelligent system, there is a lot of work to be done in order to take the most out of the capabilities of neural networks, fuzzy sets and expert systems to provide user-friendly support in decision problems where multiple criteria are involved.

See also

- ▶ Bi-objective Assignment Problem
- ▶ Estimating Data for Multicriteria Decision Making Problems: Optimization Techniques
- ▶ Financial Applications of Multicriteria Analysis
- ▶ Fuzzy Multi-objective Linear Programming
- ▶ Multicriteria Sorting Methods
- ▶ Multi-objective Combinatorial Optimization
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Optimization and Decision Support Systems
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multi-objective Optimization: Interactive Methods for Preference Value Functions
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multi-objective Optimization: Pareto Optimal Solutions, Properties
- ▶ Multiple Objective Programming Support
- ▶ Outranking Methods
- ▶ Portfolio Selection and Multicriteria Analysis
- ▶ Preference Disaggregation
- ▶ Preference Disaggregation Approach: Basic Features, Examples From Financial Decision Making
- ▶ Preference Modeling

References

1. Anselin L, Arias EG (1983) A multi-criteria framework as decision support system for urban growth management applications: Central city redevelopment. Europ J Oper Res 13:300–309
2. Antunes CH, Alves MJ, Silva AL, Climaco J (1992) An integrated MOLP method base package-A guided tour of TOMMIX. Comput Oper Res 1(4):609–625
3. Antunes CH, Melo MP, Climaco JN (1992) On the integration of an interactive MOLP procedure base and expert system techniques. Europ J Oper Res 61:135–144

4. Balestra G, Tsoukiàs A (1990) Multicriteria analysis represented by artificial intelligence techniques. J Oper Res Soc 41(5):419–430
5. Bana e Costa CA, Vansnick JC (1994) MACBETH-An interactive path towards the construction of cardinal value functions. Internat Trans Oper Res 1:489–500
6. Belton V, Vickers SP (1989) V.I.S.A.-VIM for MCDA. In: Lockett AG, Islei G (eds) Improving Decision Making in Organizations. Springer, Berlin, pp 319–334
7. Brans JP, Mareschal B (1994) The PROMCALC and GAIA decision support system for multicriteria decision aid. Decision Support Systems 12:297–310
8. Bui T (1994) Software architectures for negotiation support: Co-oP and Negotiator. Computer-Assisted Negotiation and Mediation Symposium, Program of Negotiation (May 26-27 1994), Harvard Law School, Cambridge, MA, pp 216–227
9. Chandrasekaran G, Ramesh R (1987) Microcomputer based multiple criteria decision support system for strategic planning. Inform and Management 12:163–172
10. Charnes A, Cooper WW (1961) Managem. models and industrial applications of linear programming. Wiley, New York
11. Climaco J, Antunes CH (1989) Implementation of a user friendly software package-A guided tour of TRIMAP. Math Comput Modelling 12(10–11):1299–1309
12. Colson G, Mareschal B (1994) JUDGES: A descriptive group decision support system for the ranking of items. Decision Support Systems 12:391–404
13. Csaki P, Rapcsak T, Turchanyi P, Vermes M (1995) R and D for group decision aid in Hungary by WINGDSS, a Microsoft Windows based group decision support system. Decision Support Systems 14:205–217
14. Czyzak P, Slowinski R (1996) Possibilistic construction of fuzzy outranking relation for multiple-criteria ranking. Fuzzy Sets and Systems 81:123–131
15. Dutta A, Jain HK (1985) A DSS for distributed computer system design in the presence of multiple conflicting objectives. Decision Support Systems 1:233–246
16. Dyer J (1973) A time-sharing computer program for the solution of the multiple criteria problem. Managem Sci 19:1379–1383
17. Eom HB, Lee SM (1990) Decision support systems applications research: A bibliography 1971–1988. Europ J Oper Res 46:333–342
18. Eom HB, Lee SM, Snyder CA, Ford FN (1987/8) A multiple criteria decision support system for global financial planning. J Management Information Systems 4(3):94–113
19. Eom SB, Lee SM, Kim JK (1993) The intellectual structure of decision support systems: 1971–1989. Decision Support Systems 10:19–35
20. Fishburn PC (1965) Independence in utility theory with whole product sets. Oper Res 13:28–45
21. Forman EH, Selly MA (2000) Decisions by objectives: How to convince others that you are right. World Sci., Singapore
22. Jacquet-Lagrèze E (1990) Interactive assessment of preferences using holistic judgments: The PREFCALC system. In: Bana e Costa CA (ed) Readings in Multiple Criteria Decision Making. Springer, Berlin, pp 335–350
23. Jacquet-Lagrèze E, Siskos J (1982) Assessing a set of additive utility functions for multicriteria decision-making: The UTA method. Europ J Oper Res 10:151–164
24. Jarke M, Jelassi MT, Shakun MF (1987) MEDIATOR: Toward a negotiation support system. Europ J Oper Res 31:314–334
25. Jelassi MT (1987) MCDM: From stand-alone methods to integrated and intelligent DSS. In: Sawaragi Y, Inoue K, Nakayama H (eds) Towards Interactive and Intelligent Decision Support Systems. Springer, Berlin, pp 575–584
26. Keeney RL, Raiffa H (1976) Decisions with multiple objectives: Preferences and value trade-offs. Wiley, New York
27. Kiss LN, Martel JM, Nadeau R (1994) ELECCALC-An interactive software for modelling the decision maker's preferences. Decision Support Systems 12:311–326
28. Klein MR, Methlie LB (1995) Knowledge based decision support systems with application in business. Wiley, New York
29. Korhonen P (1987) VIG-A visual interactive support system for multiple criteria decision making. Belgian J Oper Res Statist Computer Sci 27:3–15
30. Korhonen P (1988) A visual reference direction approach to solving discrete multiple criteria problems. Europ J Oper Res 34:152–159
31. Korhonen P, Moskowitz H, Wallenius J (1992) Multiple criteria decision support-A review. Europ J Oper Res 63:361–375
32. Korhonen P, Wallenius J (1988) A Pareto race. Naval Res Logist 35:615–623
33. Langen D (1989) An (interactive) decision support system for bank asset liability management. Decision Support Systems 5:389–401
34. Levine P, Pomerol JCh (1986) PRIAM, an interactive program for chosing among multiple attribute alternatives. Europ J Oper Res 25:272–280
35. Lewandowski A (1989) SCDAS-Decision support system for group decision making: Decision theoretic framework. Decision Support Systems 5:403–423
36. Lewandowski A, Kreglewski T, Rogowski T, Wierzbicki A (1989) Decision support systems of DIDAS family (Dynamic Interactive Decision Analysis & Support. In: Lewandowski A and Wierzbicki A (eds) Aspiration Based Decision Support Systems. Springer, Berlin, pp 21–27
37. Lofti V, Stewart TJ, Zionts S (1992) An aspiration-level interactive model for multiple criteria decision making. Comput Oper Res 19:677–681
38. Malakooti B, Zhou YQ (1994) Feedforward artificial neural networks for solving discrete multiple criteria decision making problems. Managem Sci 40(11):1542–1561
39. Mareschal B, Brans JP (1991) BANKADVISER: An industrial evaluation system. Europ J Oper Res 54:318–324

40. Mareschal B, Mertens D (1992) BANKS a multicriteria, PROMETHEE-based decision support system for the evaluation of the international banking sector. Revue des Systèmes de Décision 1(2):175–189
41. Martel JM, D'Avignon CR, Couillard J (1986) A fuzzy outranking relation in multicriteria decision making. Europ J Oper Res 25:258–271
42. Matsatsinis NF, Siskos Y (1999) MARKEX: An intelligent decision support system for product development decisions. Europ J Oper Res 113:336–354
43. Neumann J Von, Morgenstern O (1944) Theory of games and economic behavior. Princeton Univ. Press, Princeton
44. Pardalos PM, Siskos Y, Zopounidis C (1995) Advances in multicriteria analysis. Kluwer, Dordrecht
45. Pinson S (1992) A multi-expert architecture for credit risk assessment: The CREDEX system. In: O'Leary DE, Watkins PR (eds) Expert Systems in Finance. North-Holland, Amsterdam, pp 27–64
46. Pomerol JCh (1993) Multicriteria DSSs: State of the art and problems. Central Europ J Oper Res Econ 3(2):197–211
47. Roy B (1968) Classement et choix en présence de points de vue multiples: La méthode ELECTRE. RIRO 8:57–75
48. Roy B (1991) The outranking approach and the foundations of ELECTRE methods. Theory and Decision 31:49–73
49. Roy B, Vanderpooten D (1997) An overview on the European school of MCDA: Emergence, basic features and current works. Europ J Oper Res 99:26–27
50. Siskos J (1982) A way to deal with fuzzy preferences in multiple-criteria decision problems. Europ J Oper Res 10:614–324
51. Siskos Y, Spiridakos A, Yannacopoulos D (1993) MINORA: A multicriteria decision aiding system for discrete alternatives. J Inf Sci Techn 2:136–149
52. Siskos Y, Spiridakos A, Yannacopoulos D (1999) Using artificial intelligence and visual techniques into preference disaggregation analysis: The MIIDAS system. Europ J Oper Res 113:281–299
53. Siskos Y, Zopounidis C, Pouliezos A (1994) An integrated DSS for financing firms by an industrial development bank in Greece. Decision Support Systems 12:151–168
54. Spronk J (1981) Interactive multiple goal programming application to financial planning. Martinus Nijhoff, Boston, MA
55. Srinivasan V, Ruparel B (1990) CGX: An expert support system for credit granting. Europ J Oper Res 45:293–308
56. Stam A, Sun M, Haines M (1996) Artificial neural network representations for hierarchical preference structures. Comput Oper Res 23(12):1191–1201
57. Steuer RE (1986) Multiple criteria optimization: Theory, computation and application. Wiley, New York
58. Steuer RE (1992) Manual for the ADBASE multiple objective linear programming package. Dept Management Sci and Inform. Technol. Univ. Georgia, Athens, GA
59. Teghem J, Dufrane D, Thauvoye M, Kunsch P (1986) STRANGE: An interactive method for multi-objective linear programming under uncertainty. Europ J Oper Res 26:65–82
60. Vranes S, Stanojevic M, Stevanovic V, Lucin M (1996) INVEX: Investment advisory expert system. Expert Systems 13, no 2:105–119
61. Wallenius J, Zionts S (1976) Some tests of an interactive programming method for multicriteria optimization and an attempt at implementation. In: Thiriez H, Zionts S (eds) Multiple Criteria Decision Making. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 319–331
62. Yu W (1992) ELECTRE TRI: Aspects methodologiques et manuel d'utilisation. Document du Lamsade (Univ Paris-Dauphine) 74
63. Zeleny M (1982) Multiple criteria decision making. McGraw-Hill, New York
64. Zopounidis C (1997) The European school of MCDA: Some recent trends. In: Climaco J (ed) Multicriteria Analysis. Springer, Berlin, pp 608–616
65. Zopounidis C, Doumpos M (1998) Developing a multicriteria decision support system for financial classification problems: The FINCLAS system. Optim Methods Softw 8(3-4)
66. Zopounidis C, Doumpos M (2000) PREFDIS: A multicriteria decision support system for sorting decision problems. Comput Oper Res 27:779–797
67. Zopounidis C, Godefroid M, Hurson Ch (1995) Designing a multicriteria DSS for portfolio selection and management. In: Janssen J, Skiadas CH, Zopounidis C (eds) Advances in Stochastic Modeling and Data Analysis. Kluwer, Dordrecht, pp 261–292
68. Zopounidis C, Matsatsinis NF, Doumpos M (1996) Developing a multicriteria knowledge-based decision support system for the assessment of corporate performance and viability: The FINEVA system. Fuzzy Economic Rev 1(2):35–53

Decomposition Algorithms for the Solution of Multistage Mean-Variance Optimization Problems

PANOS PARPAS, BERÇ RUSTEM
Department of Computing, Imperial College, London, UK

MSC2000: 90C15, 90C90

Article Outline

Abstract

Stochastic multistage mean-variance optimization problems represent one of the most frequently used modeling tools for planning problems, especially financial. Decomposition algorithms represent a powerful tool for the solution of problems belonging to this class. The first aim of this article is to introduce multi-stage mean-variance models, explain their applications and structure. The second aim is the discussion of efficient solution methods of such problems using decomposition algorithms.

Background

Stochastic programming (SP) is becoming an increasingly popular tool for modeling decisions under uncertainty because of the flexible way uncertain events can be modeled, and real-world constraints can be imposed with relative ease. SP also injects robustness to the optimization process. Consider the following standard "deterministic" quadratic program:

$$\begin{aligned} \min_x \quad & \frac{1}{2}x'Hx + c'x \\ s.t \quad & Ax = b \\ & x^l \le x \le x^u \, . \end{aligned} \tag{1}$$

It is not always possible to know the exact values of the problem data of (1) given by H, A, c, and b. Instead, we may have some estimations in the form of data gathered either empirically or known to be approximated well by a probability distribution. The SP framework allows us to solve problems where the data of the problem are represented as functions of the randomness, yielding results that are more robust to deviations.

The power and flexibility of SP does, however, come at a cost. Realistic models include many possible events distributed across several periods, and the end result is a large-scale optimization problem with hundreds of thousands of variables and constraints. Models of this scale cannot be handled by general-purpose optimization algorithms, so special-purpose algorithms attempt to take advantage of the specific structure of SP models. We examine two decomposition algorithms that had encouraging results reported in linear SP; the first is based on the regularized version of Benders decomposition developed by [21], and the second on an augmented-lagrangian-based scheme developed by [4].

Others [9,24,27] formulated multistage SP as a problem in optimal control, where the current stage variables depend on the parent node variables, and used techniques from optimal control theory to solve the resulting problem. Another related method is the approximation algorithm by [11] where a sequence of scenario trees is generated whose solution produces lower and upper bounds on the solution of the true problem. Decomposition algorithms are not, however, the only approach to tackle the state explosion from which SPs suffer; approximation algorithms and stochastic methods are just two examples of other methods where research is very active [5]. In this study, we are concerned only with decomposition methods.

Problem Statement

We consider a quadratic multistage SP. In the linear case, SP was first proposed independently by [10] and [1]; for a more recent description see [7] and [13]. For two stages, the problem is:

$$\min_x \quad \frac{1}{2}x'Hx + c'x + \mathcal{Q}(x) \tag{2a}$$

$$s.t \quad Ax = b \tag{2b}$$

$$x^l \le x \le x^u \, . \tag{2c}$$

We use $'$ to denote the transpose of a vector or a matrix. c and $x^{u,l}$ are known vectors in $\Re^{n_1}$. Let A and H be known matrices in $\Re^{m_1 \times n_1}$ and $\Re^{n_1 \times n_1}$. These quantities represent the state of the world that is known. We assume that H is positive semidefinite. The first two terms in the objective function (2a) model the goals of the decision maker that do not depend on uncertain events. $\mathcal{Q}(x)$ represents the expected value of the second-stage objective function:

$$\mathcal{Q}(x) = E_\xi[Q(x, \xi(\omega))] \, ,$$

where

$$Q(x, \xi(\omega)) = \min_y \frac{1}{2}\alpha y'(\omega)H(\omega)y(\omega) - (1-\alpha)c'(\omega)y(\omega) \tag{3}$$

$$s.t \quad W(\omega)y(\omega) = h(\omega) - T(\omega)x \tag{4}$$

$$y(\omega)^l \leq y(\omega) \leq y(\omega)^u \,. \tag{5}$$

Let Ω be the set of all random events, and $\omega \in \Omega$ be the particular realization of an event so that when ω is known the random events are aggregated in the vector $\xi(\omega) = [y(\omega), H(\omega), W(\omega), h(\omega), T(\omega), y^{u,l}(\omega)]$, and let Ξ be the support of ξ. The uncertainty of the second stage is represented by the random data $H(\omega)$, $W(\omega)$, and $T(\omega)$, which are matrices in $\Re^{n_2 \times n_2}, \Re^{m_2 \times n_2}$, and $\Re^{m_2 \times n_1}$ respectively. The vectors $c(\omega)$, $h(\omega)$, and $y^{l,u}(\omega)$ are random vectors in $\Re^{n_2}, \Re^{m_2}$, and $\Re^{n_2}$ respectively. We assume that the number of possible realizations of ω is finite. Under this assumption, $\xi(\omega)$ is taken to mean that for different ω's the data of the problem change. The dependence of y on uncertainty is depicted as $y(\omega) \in \Re^{n_2}$. The vector $y(\omega)$ is still the decision variable but this notation is used to stress the point that for different realizations of ω we must have a different y. In the objective function (3), the quadratic term represents the risk of the decision measured by variance, while the linear term represents the expected outcome. The scalar $\alpha \in [0, 1]$ is used in (3) to describe the trade-off between risk expectation.

Deriving the multi-stage problem from the two-stage formulation is just a matter of applying the ideas described above recursively to attain the required number of stages. For the multistage problem with T_s periods, the first-stage decision remains the same but for $t = 2 \ldots T_s$ we have

$$\mathcal{Q}_t(x_{t-1}) = E_{\xi_t}\Big[Q_t\Big(x_{t-1}, \xi_t(\omega)\Big)\Big] \,, \tag{6}$$

where

$$\begin{aligned} Q_t(x_{t-1}, \xi_t(\omega)) = \min_y \alpha \frac{1}{2} y_t'(\omega) H_t(\omega) y_t(\omega) \\ - (1-\alpha) c_t' y_t(\omega) + \mathcal{Q}_{t+1}(y_t(\omega)) \\ s.t \; W_t(\omega) y_t(\omega) = h_t(\omega) - T_{t-1}(\omega) x_{t-1} \\ y_t(\omega)^l \leq y_t(\omega) \leq y_t(\omega)^u \,. \end{aligned} \tag{7}$$

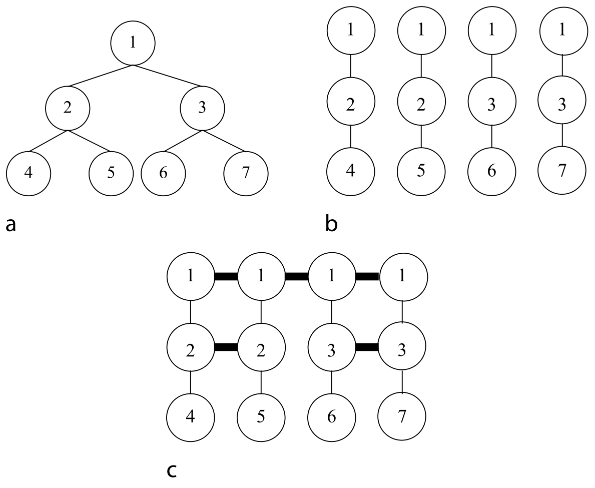
For the last time period $t = T_s$, the recourse function $\mathcal{Q}_{T_s+1}$ is zero.

Our principal concern involves decomposition algorithms for (7). For more insight into the properties of stochastic quadratic problems the reader is referred to [14], and [15]. Before we delve into decomposition algorithms, we introduce some terminology that will be used in the next section.

The dynamic programming model (7) is usually referred to as *non-anticipative*. This property means that decisions are based on the past and not the future. There are two ways this concept can be represented, namely compact and split-view formulations [20].

The compact variable formulation can be mapped directly onto a tree structure known as the *scenario tree*; see Fig. 1a. The root of the tree represents the state of the world that is deterministic. As we move down the scenario tree, different events represent different realizations of ω, each level of the tree represents a different time period, and the path from the root to a leaf node is known as a *scenario*. We use $\nu = (t, k)$ to denote the kth node in period t, $a(\nu)$ the ancestor node, and $d(\nu)$ the descendant nodes. Benders decomposition, to be introduced in the next section, assumes such a structure and the result is a decomposition of the large scale problem into several subproblems, each representing a node in the tree.

In a split-variable formulation for each scenario, from the set of possible scenarios, new decision variables are introduced so that the large-scale problem is decomposed into n subproblems, where n is the number of scenarios. Conceptually, using this approach, the non-anticipative constraints are completely relaxed; see



Decomposition Algorithms for the Solution of Multistage Mean-Variance Optimization Problems, Figure 1
Different views on non-anticipativity

Fig. 1b. To enforce these constraints, new constraints are introduced that "rebuild" the links between subproblems, usually through some penalty function (see Fig. 1c).

Methods

The importance of decomposition algorithms in SP was recognized early on, as results in the theory of stochastic programs are closely linked with their solution algorithms. The two algorithms described in this section represent two very promising approaches in decomposition of SPs.

Nested Benders Decomposition (NBD)

Benders decomposition was first proposed in [2], and it has been applied to SP by [26]; it is usually referred to as the *L-shaped method* due to the structure of the constraint matrix. The extension to the non-linear convex case has been done in [12], and the extension to the general convex SP appears in [8]. The algorithm has also been widely studied for multistage problems in a parallel environment[6]. More recent studies appear in [18]. In [15] the quadratic case is also studied.

It can easily be seen that (2a) is equivalent to:

$$\begin{aligned}\min_{x,\theta}\quad & \frac{1}{2}x'Hx + c'x + e'\theta \\ s.t\quad & Ax = b \\ & \theta \le p_\omega Q(x,\xi(\omega)) \\ & x^l \le x \le x^u\end{aligned} \tag{8}$$

where e is a vector of ones. The dimension of the latter vector is equal to the number of nodes in the next period. The expression $p_\omega Q(x,\xi(\omega))$ represents the value of the next stage decision if event ω occurs (with probability p_ω). The dimensions of the rest of the data are the same as in (2a). Even though it is possible to aggregate the θ vector to a single variable, computational studies [5,6] have shown that the reduction of variables did not enhance performance, possibly due to loss of information.

To represent the recourse function in (8), we construct an approximation using outer linearizations. This is achieved by computing cuts (cutting planes). There are two types of cuts: optimality and feasibility. Instead of solving the large-scale problem (8) we solve the relaxed version

$$\begin{aligned}\min_{x}\quad & \frac{1}{2}x'Hx + c'x + e'\theta \\ s.t\quad & Ax = b \\ & Dx \ge d && \text{(9a)} \\ & \theta \ge Gx + g && \text{(9b)} \\ & x^l \le x \le x^u\end{aligned}$$

where (9a) and (9b) represent feasibility and optimality cuts, respectively. The aim of these constraints is to approximate the feasible region of (8). Feasibility cuts are constructed as follows: Assuming that $t = T$ and for a fixed $\hat{\omega}$ the νth problem in (7) takes the following form:

$$\begin{aligned}Q(x) = \min_y\ & \frac{\alpha}{2}y'Hy - (1-\alpha)c'y \\ s.t\ & Wy = h - Tx_{a(\nu)} \\ & y^l \le y \le y^u\end{aligned} \tag{10}$$

Assume that this problem is infeasible due to the vector $x_{a(\nu)}$ generated in a subproblem of a previous stage. Consider the following problem:

$$\begin{aligned}& P(y, x_{a(\nu)}) = \min_y e'y^+ + e'y^- \\ & s.t\ Wy + y^+ - y^- = h - Tx_{a(\nu)} && (11) \\ & y^l \le y \le y^u && (12) \\ & y^{+,-} \ge 0\end{aligned}$$

Then since the original problem was infeasible due to $x_{a(\nu)}$ we must have that $P(\cdot, x_{a(\nu)}) > 0$. Let λ be the Lagrange multiplier of the constraint in (11), then by duality we must also have that $\lambda'(h - Tx_{a(\nu)}) \le 0$. Set $D = \lambda'T$ and $d = \lambda'h$ to obtain (9a), a supporting hyperplane to $Q(x)$. To apply this result when $t \neq T$ just note that the same procedure is recursively applied by taking under consideration the additional constraints from cuts of other subproblems.

For optimality cuts, one proceeds as follows: again we start with the problem in (10) and let x_k be the solution vector of a subproblem in the previous stage. By the gradient inequality we must have that (ω is dropped since it is clear from context):

$$Q(x) \ge Q(x_k) + \nabla Q(x_k)(x - x_k)$$

$$Q(x_k) = \frac{\alpha}{2}y'Hy - (1-\alpha)c'y + \lambda'(Wy - h + Tx_k)$$

$$\nabla Q(x_k) = \lambda'T$$

Thus

$$Q(x) \geq \lambda' Tx + \frac{a}{2} y' Hy - (1-\alpha)c' y + \lambda'(Wy - h)$$

Set

$$\begin{aligned} \theta &= Q(x) \\ G &= \lambda' T \\ g &= \frac{\alpha}{2} y' Hy - (1-\alpha)c' y + \lambda(Wy - h) \end{aligned}$$

to obtain (9b). Since we require a lower support for the expected value, we then multiply G and g by the probability of ω taking the particular realization of $\hat{\omega}$ for two-stage problems and the conditional probability for multistage problems. The application of optimality cuts when $t \neq T$ is again developed recursively just by taking into account the additional variables and constraints.

The algorithm proceeds by solving the relaxed problem (7) to obtain a solution vector, known as the *proposal vector*. The latter is then used to solve the subproblems in (10). If a subproblem is feasible then an optimality cut is appended to the constraint set of the ancestor problem (also called the *master problem*). Otherwise, only a feasibility cut is appended.

In the linear case, there are some well known drawbacks to the algorithmic framework developed above [5,21]. We expect issues similar to the following to manifest themselves in the quadratic case:

- The algorithm tends to be inefficient in early iterations due to the poor description of the original objective function provided by the cuts. Moreover, if a good warm-start is used, the algorithm may deviate significantly from this point, so any efficiency achieved by a good starting point is lost.
- The number of cuts for master problems may increase substantially, adding considerable computational burden to their solution.

For these reasons a regularized version of the algorithm was proposed in [21]; see also [22,23] for the multistage version. Ruszczynski's results, as well as a study performed in [28], indicate that the regularized version outperforms the original algorithm.

The basic idea is to add a quadratic term $\rho \parallel x - \hat{x} \parallel_2^2$ in the objective function, where $\hat{x}$ is chosen as the "best" current point, in a way to be made precise, and ρ is a penalty parameter. For a high value of ρ the algorithm is penalized from deviating from the current point. In [21], the convergence of the algorithm for $\rho = \frac{1}{2}$ was established for the convex case. The regularizing term stabilizes the behavior of the algorithm between iterations, enables valid deletion schemes of the cuts, and avoids degenerate iterations that would otherwise be possible.

The original problem is now decomposed into three types of subproblems. The first type is for the root node. The following problem is solved at each iteration:

$$\begin{aligned} \min_{x,\theta} \quad & \frac{1}{2}\alpha x' Hx - (1-\alpha)c' x + e'\theta + \frac{\rho}{2} \parallel x - \hat{x} \parallel_2^2 \\ s.t \quad & Ax = b \\ & Gx \geq \theta + g \\ & Dx \geq d \\ & x^l \leq x \leq x^u \,. \end{aligned}$$

The second type is for non-terminal nodes. The following subproblem needs to be considered:

$$\begin{aligned} \min_{y_\nu,\theta_\nu} \quad & \frac{a}{2} y_\nu' H_\nu y_\nu - (1-\alpha)c_\nu' y_\nu \\ & + e'\theta_\nu + \frac{\rho_\nu}{2} \parallel y_\nu - \hat{y}_\nu \parallel_2^2 \\ s.t \quad & W y_\nu = h_\nu - T_{a(\nu)} x_{a(\nu)} \\ & G_\nu y_\nu \geq \theta_\nu + g_\nu \\ & D_\nu y_\nu \geq d_\nu \\ & y_\nu^l \leq y_\nu \leq y_\nu^u \,. \end{aligned} \tag{13}$$

The third type is for terminal nodes. This type of subproblem is identical to (13) without, of course, the cuts in the constraint set and the regularizing term in the objective function.

The way cuts are recursively defined and the way subproblems are nested in each other has led this to be referred to as *nested Benders decomposition* (NBD). The algorithm can now be stated as follows:

Step 1: Set the iteration counter $i = 0$ and $t = k = 0$, and let $\hat{x}$ be a feasible point.

Step 2: Construct and solve $\nu(t, k)$ to find the solution vector x_ν^i.

Step 2.1: If the problem is infeasible and $t = 0$ then STOP: the problem is infeasible.

Step 2.2: If the problem is infeasible and $t > 0$, generate an optimality cut (9a) and append it to the constraint set of $a(\nu)$.

Step 2.3: If the problem was optimal and $t > 0$, generate an optimality cut (9b) and append it to the constraint set of $a(\nu)$.

Step 3: Compute

$$\hat{F}(x_\nu^i) = \frac{1}{2}x'Hx + c'x + \sum_{j=d(\nu)} \theta_j$$

$$F(\hat{x}_\nu^i) = \frac{1}{2}x'Hx + c'x + \sum_{j=d(\nu)} Q_j(x)\,.$$

If $\hat{F} = F$ and $t = k = 0$ then STOP: $\hat{x}$ is optimal; Else go to step 4

Step 4: Update the regularizing term:

4.1 If a subproblem returned a feasibility cut then $\hat{x}_\nu^{i+1} = \hat{x}_\nu^i$.

4.2 If $F(x_\nu^i) > F(\hat{x}_\nu^i)$ or $F(x_\nu^i) > \tau F(\hat{x}_\nu^i) + (1-\tau)\hat{F}$, then set $\hat{x}_\nu^{i+1} = \hat{x}_\nu^i$, and increase ρ.

4.3 If $F(x_\nu^i) < F(\hat{x}_\nu^i)$ or $F(x_\nu^i) < \tau F(\hat{x}_\nu^i) + (1-\tau)\hat{F}$, then set $\hat{x}_\nu^{i+1} = x_\nu^i$ and decrease ρ.

4.4 If $F(x_\nu^i) = \hat{F}$, then set $\hat{x}_\nu^{i+1} = x_\nu^i$, and decrease ρ.

Step 5: Set $i = i + 1$, find the next subproblem to solve (see below), and go to step 2.

Augmented Lagrangian Decomposition (ALD)

An alternative algorithm to Benders decomposition described in the previous section is based on the augmented lagrangian and the method of multipliers [3]. The fundamental difference between NBD and ALD is the way the two algorithms attack non-anticipativity constraints. NBD handles these constraints by having a master problem generating proposals to the subproblems further down the event tree; proposal vectors are affected by "future" nodes by feasibility and optimality cuts. In ALD a different approach is taken: non-anticipativity constraints are relaxed by expressing the large-scale problem in terms of smaller subproblems that are discouraged from violating the original constraints. The algorithm we use was developed in [4], so here we only sketch the main idea. ALD was developed and applied to the stochastic quadratic programming setting in [25] with encouraging results. Similar algorithms to ALD have been developed for linear stochastic programs [16,17].

The expectation in (6) for a given time period can also be written as

$$\begin{aligned} &\min_y \sum_{i=1}^m p_i\Big(\frac{\alpha}{2}y_i'H_iy_i - (1-\alpha)c_i'y_i\Big)\\ &s.t W_jy_i = h_j - T_jx_{a(i)} \qquad j = 1\dots r\\ &y_i^l \le y_i \le y_i^u\,. \end{aligned} \tag{14}$$

The problem in (14) is to be interpreted as follows: at the current time period there are m scenarios, each having different realizations for H, W, c, etc. There are r linking constraints (14) that are linked by the vector $x_{a(i)}$. In [4] the problem is decomposed by introducing a new variable z as follows

$$\begin{aligned} &\min_{y,z} \sum_{i=1}^m p_i\Big(\frac{\alpha}{2}y_i'H_iy_i - (1-\alpha)c_i'y_i\Big)\\ &s.t\ W_{ji}y_i = z_{ij} \qquad j = 1\dots r; i \in I(j)\\ &z_{ij} = h_j - T_jx_{a(i)} \qquad j = 1\dots r; i \in I(j)\\ &y_i^l \le y_i \le y_i^u\,, \end{aligned} \tag{15}$$

where $I(j)$ contains the indices of the subproblems that the jth constraint "crosses", i. e., $I(j) = \{i|w_{ji} \neq 0\}$. "Crosses" means that a constraint contains data from more than one subproblem. It is obvious that (14) and (15) are exactly the same problem, but the structure of (15) facilitates a decomposition algorithm via the relaxation of the constraints of (15). In [4] the method of multipliers is used for the general problem $\min\{f(x)|Ax = b\}$. Let $L_c(x,\lambda)$ denote the associated augmented lagrangian defined by $L_c(x,\lambda) = f(x) + \lambda'(Ax - b) + \frac{c}{2} \parallel Ax - b \parallel_2^2$, where λ is the vector of multipliers. The general algorithmic framework of the method of multipliers can be described as follows:

Step 1: Initialization: Set the iteration counter $k = 0$, and set $c(0) > 0$. Set $x(0)$, and $\lambda(0)$ as the starting point for the decision variables, and lagrange multipliers, respectively.

Step 2: Compute the next point $x(k+1) = \arg\min L_c(x, \lambda(k))$.

Step3 Update the Lagrange multiplier vector $\lambda(k+1) = \lambda(k) + c(k)\big(Ax(k+1) - b\big)$.

Step 4: Update the penalty parameter $c(k)$, and set $k = k + 1$. If some convergence criterion is not satisfied go to step 2.

Applying this general algorithmic framework to (15), the problem is decomposed into m subproblems and the non-anticipativity constraints are enforced through the penalty term in the augmented lagrangian.

The computation for the solution of (15) involves keeping z fixed in order to compute the next incumbent for y, and then keeping y fixed in order to compute the next incumbent for z. Thus, at the kth iteration the following subproblems are solved:

$$\begin{aligned} y_i(k+1) = \arg\min_{\xi} \Bigg\{ & p_i \left(\frac{\alpha}{2} \xi_i' H_i \xi_i - (1-\alpha) c_i' \xi_i \right) \\ & + \sum_{\{j \mid i \in I(j)\}} \Big(\lambda_{ji}'(k) W_{ji} \xi_i \\ & + \frac{c(k)}{2} (W_{ji}\xi_i - z_{ji})^2 \Big) \Bigg\} \\ & \forall i = 1, \dots, n \\ z_{ji}(k+1) = \arg\min_{\zeta_{ji}} \Bigg\{ & - \sum_{i \in I(j)} \lambda_{ji}'(k) \zeta_{ji} + \frac{c(k)}{2} \\ & \cdot \sum_{i \in I(j)} (W_{ji}\xi_i - \zeta_{ji})^2 \Big) \Bigg\} \quad \forall j = 1, \dots, r \\ & \text{s.t } \zeta_{ji} = h_j - T_j x_{a(i)} \quad i \in I(j) \end{aligned}$$

followed by an update of the lagrange-multiplier vector $\lambda_{ji}(k+1) = \lambda_{ji}(k) + c(k)(W_{ji} y_i(k+1) - z_{ji}(k+1))$. From a computational point of view the above iterative framework is inefficient because of the alternate minimizations required, making this algorithm unsuitable for a parallel environment. In our implementation we used the more efficient iteration proposed in [4]:

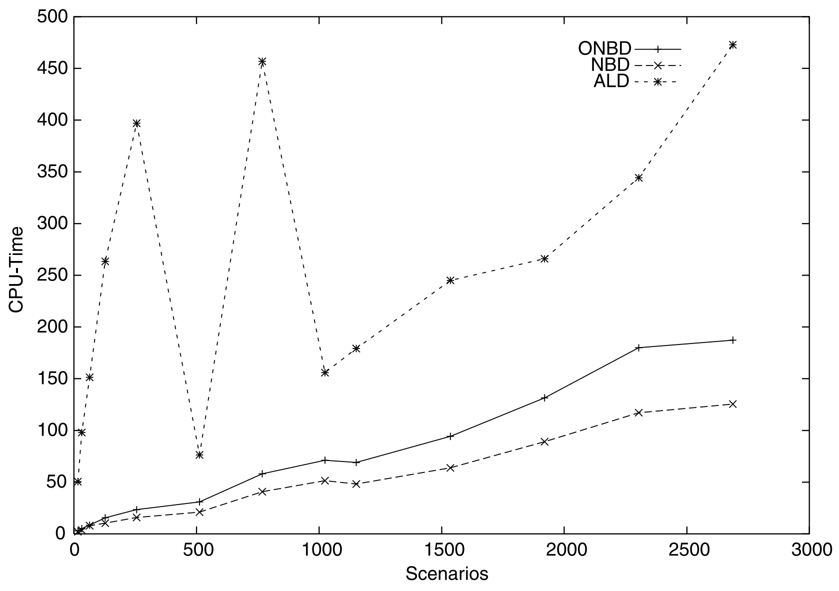
$$\begin{aligned} y_i(k+1) = \arg\min_{\xi} \Bigg\{ & p_i \Big(\frac{\alpha}{2} \xi_i' H_i \xi_i - (1-\alpha) c_i' \xi_i \Big) \\ & + \sum_{\{j \mid i \in I(j)\}} \Big(\lambda_j'(k) W_{ji} \xi_i \\ & + \frac{c(k)}{2} \big(W_{ji}(\xi_i - y_i(k)) + w_j \big)^2 \Big) \Bigg\} \end{aligned} \quad (16)$$

where $w_j = \frac{1}{m_j}(W_j y_i - h_j + T_j x_{a(i)})$, $\lambda_j(k+1) = \lambda_j(k) + \frac{c(k)}{m_j}(W_j y_i - h_j + T_j x_{a(i)})$, and m_j denotes the cardinality of $I(j)$. The derivation of this iteration is discussed in Bertsekas and Tsitsiklis ([4], p. 249). The expression in (16) forms the main iteration of the ALD algorithm. In order to have a complete description of the algorithm we need to specify how one can perform the updates of the penalty parameter $c(k)$ and how we tested for convergence.

The obvious convergence criteria for ALD are a test for feasibility and small changes in the objective function. However, it is possible, due to a poor selection of updates for $c(k)$, to reach a suboptimal solution. For this reason, it is vital to check the KKT conditions of the problem in addition to any other stopping criteria. If the KKT conditions are not satisfied while the change in the objective function is small (10^{-6} in our implementation), the update strategy for the penalty parameter appears to have been inappropriate. We performed various experiments with different update strategies for this penalty parameter and found that the strategy that works best on most problems is to start with a small value (0.001) and increase it at every iteration by another small factor (1.05); being more aggressive with the update of this parameter caused the algorithm to terminate prematurely. Note that an arbitrary starting point can be used to start the algorithm. If a feasible solution or the solution from a previous run is available it may be beneficial to start with a higher penalty term.

Numerical Experiments

The two algorithms were implemented and tested on a multistage financial planning problem. The detailed results can be found in [19]. Figure 2 summarizes the numerical performance (in terms of CPU time) as the number of scenario increases. ALD, and NBD stand for Augmented Lagrangian Decomposition, and Nested Benders Decomposition respectively. ONBD refers to Ordinary Nested Benders Decomposition, i. e. NBD without the regularizing term. From Fig. 2 it is clear that the regularized version of Benders decomposition is the most efficient of the algorithms we considered in this article. This result is in line with similar studies performed in the linear setting. One possible explanation is that the NBD algorithm takes advantage of the constraint structure of multistage stochastic programming problems more effectively. Note that the ALD algorithm can be applied to separable convex problems with more general constraint structure while NBD will need to be modified in order to be applicable to other types of separable problems. SP problems are one of the



Decomposition Algorithms for the Solution of Multistage Mean-Variance Optimization Problems, Figure 2
Solution times vs. number of scenarios

most frequently occurring class of large scale problems, so it is important to know whether cutting plane type algorithms or Lagrangian based algorithms take advantage of this structure more effectively. Based on the results of our experiments it seems that the NBD algorithm appears to be substantially better. Furthermore, we found that the penalty parameter often caused notable changes to the convergence times of both NBD and ALD. Finding an update scheme that works for all problems is a difficult task. In ALD the penalty parameter has two goals, one is forcing feasibility and the other of keeping iterations close to each other, thus a 'suboptimal' penalty update scheme may be more damaging than in NBD, this may give some insight to the difference in performance of the two algorithms. More detailed numerical experiments can be found in [19].

References

1. Beale EML (1955) On minimizing a convex function subject to linear inequalities. J R Stat Soc 17:173–184
2. Benders JF (1962) Partitioning procedures for solving mixed-variables problems. Numerische Mathematik 4: 238–252
3. Bertsekas DP (1999) Nonlinear Programming, 2nd edn. Athena Scientific, Belmont
4. Bertsekas DP, Tsitsiklis JN (1989) Parallel and Distributed Computation. Prentice-Hall, Englewood Cliffs
5. Birge JR (1997) Stochastic programming computation and applications. INFORMS J Comput 9:111–133
6. Birge JR, Donohue CJ, Holmes DF, Svintsitski OG (1996) A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Math Program 75(2):327–352
7. Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, New York
8. Birge JR, Rosa CH (1996) Parallel decomposition of large-scale stochastic nonlinear programs. Ann Oper Res 64: 39–65
9. Blomvall J, Lindberg P (2002) A Riccati-based primal interior point solver for multistage stochastic programming. Eur J Oper Res 143(2):452–461
10. Dantzig GB (1955) Linear programming under uncertainty. Manag Sci 1:197–206
11. Frauendorfer K (1996) Barycentric scenario trees in convex multistage stochastic programming. Math Program 75(2B):277–293
12. Geoffrion AM (1972) Generalized Benders decomposition. J Optim Theory Appl 10(4):237–260
13. Kall P, Wallace SW (1994) Stochastic Programming. Wiley, Chichester
14. Lau K, Womersley RS (2001) Multistage quadratic stochastic programming. J Comput Appl Math 129(1-2):105–138
15. Louveaux F (1978) Piecewise convex programs. Math Program 15:53–62
16. Mulvey JM, Ruszczynski A (1992) A diagonal quadratic approximation method for linear multistage stochastic programming problems. In: System modeling and optimiza-

tion, Lecture Notes in Control and Inform Sci, vol 180. Springer, Berlin, pp 588–597

17. Mulvey JM, Ruszczynski A (1995) A new scenario decomposition method for large-scale stochastic optimization. Oper Res 43(3):477–490
18. Nielsen SS, Zenios SA (1997) Scalable parallel Benders decomposition for stochastic linear programming. Parallel Comput 23(8):1069–1088
19. Parpas P, Rustem B (2005) Computational assessment of nested benders and augmented lagrangian decomposition for mean-variance multistage stochastic problems. INFORMS J Comput 19(2):239–247
20. Rockafellar T, Wets R (1991) Scenarios and policy aggregation in optimization under uncertainty. Math Oper Res 16(1):119–147
21. Ruszczynski A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Math Program 35:309–333
22. Ruszczynski A (1993) Parallel decomposition of multistage stochastic programming problems. Math Program 58(2A):201–228
23. Ruszczynski A (1995) On the regularized decomposition method for stochastic programming problems. In: Stochastic programming, Lecture Notes in Econom and Math Systems, vol 423. Springer, Berlin, pp 93–108
24. Salinger DH, Rockafellar T (2003) Dynamic splitting: an algorithm for deterministic and stochastic multiperiod optimization. Stochastic Programming E-Print Series (SPEPS). http://www.speps.info/
25. Settergren R (2001) Decomposition of financial engineering problems. Technical report. Department of Computing, Imperial College, London
26. Van Slyke R, Wets RJ-B (1969) L-shaped linear programs with applications to control and stochastic programming. SIAM J Appl Math 17:638–663
27. Steinbach MC (1998) Recursive direct algorithms for multistage stochastic programs in financial engineering. In: Kall P, Luthi HJ (eds) Papers of the International Conference on Operations Research. Springer, New York, pp 241–250
28. Vladimirou H (1998) Computational assessment of distributed decomposition methods for stochastic linear programs. Eur J Oper Res 108:653–670

Decomposition in Global Optimization

HOANG TUY
VAST, Institute of Mathematics,
Hanoi, Vietnam

MSC2000: 90C26, 90C25

Article Outline

In many nonconvex optimization problems the set of variables is partitioned into two groups such that the problem becomes much easier to solve when the variables in one group are held temporarily fixed. To exploit this structure, a method which has proved to be efficient is to *decompose* the problem into a sequence of easier subproblems involving only variables of the other group. The basic tool for this decomposition is the branch and bound (BB) concept.

BB Procedure for Decomposition

Consider the nonconvex global optimization problem

$$\min\{F(x,y)\colon G(x,y)\preceq_K 0,\ x\in X,\ y\in Y\}\,, \qquad \text{(P)}$$

where X is a compact convex subset of $\mathbb{R}^n$, Y is a closed convex subset of $\mathbb{R}^p$, $F\colon X\times Y\to\mathbb{R}$, $G\colon X\times Y\to\mathbb{R}^m$, K is a closed convex cone in $\mathbb{R}^m$ and $\preceq_K$ is the partial ordering in $\mathbb{R}^m$ induced by the cone K, i. e., such that $y\preceq_K y'\Leftrightarrow y'-y\in K$.

Problems of this form abound in applications such as pooling and blending in oil refining, optimal design of water distribution, structural design, signal processing, robust stability analysis and design of chips.

Suppose that by fixing $x\in X$ problem (P) becomes an easier problem in $y\in Y$. Then, to take advantage of this property on can solve (P) by a BB algorithm with branching performed in the x-space.

Specifically, at iteration k of of the BB procedure a collection S_k of partition sets in the x-space is considered, where for each partition set $M\in S_k$ a number (lower bound) $\beta(M)\in\mathbb{R}\cup\{+\infty\}$ has been computed such that

$$\beta(M)\le\inf\{F(x,y)\colon G(x,y)\preceq_K 0, x\in M\cap X, y\in Y\}. \qquad (1)$$

A partition set $M_k \in \operatorname{argmin}\{\beta(M)\colon M \in S_k\}$ is then further subdivided according to an exhaustive subdivision rule (e. g., the standard bisection rule, if rectangular subdivision is used), while the best feasible solution available $(\bar{x}^k, \bar{y}^k)$ is recorded. By removing every M such that $\beta(M) \geq F(\bar{x}^k, \bar{y}^k)$ the new collection S_{k+1} is formed. If $S_{k+1} = \emptyset$, the procedure terminates, concluding that the problem is infeasible if no feasible solution is available, or else that the current best feasible solution is actually an optimal one. Otherwise, the next iteration is started.

A key operation in this procedure is bounding: $M \mapsto \beta(M)$. It is assumed that this operation satisfies the following natural conditions:

$$\begin{aligned} &\text{(a)} \quad M' \subset M \Rightarrow \beta(M') \geq \beta(M)\,;\\ &\text{(b)} \quad \beta(M) < +\infty \Rightarrow M \cap X \neq \emptyset\,. \end{aligned} \tag{2}$$

When the BB procedure is infinite it generates a filter (an infinite nested sequence of partition sets) $M_{k_\nu}, \nu = 1, 2, \ldots,$ such that

$$\begin{aligned} &\beta(M_{k_\nu}) \leq \min(\mathrm{P}) \quad \forall \nu,\\ &M_{k_\nu} \cap X \neq \emptyset \quad \forall \nu,\\ &\bigcap_{\nu=1}^{+\infty} M_{k_\nu} = \{x^*\}\,. \end{aligned} \tag{3}$$

The algorithm is said to be *convergent* if $x^* \in X$ and

$$\min(\mathrm{P}) = \min\{F(x^*, y)\colon G(x^*, y) \preceq_K 0,\ y \in Y\}, \tag{4}$$

so any optimal solution y^* of this problem yields an optimal solution (x^*, y^*) of (P).

The basic issue of this decomposition scheme is under which conditions the BB procedure described above is guaranteed to converge in sense (4).

First observe that, since $M_{k_{\nu+1}} \subset M_{k_\nu}$ and hence, $\beta(M_{k_{\nu+1}}) \geq \beta(M_{k_\nu})$, we have from (3)

$$\beta(M_{k_\nu}) \nearrow \beta^* \leq \min(\mathrm{P})\,. \tag{5}$$

Theorem 1 *If $\beta(M_k) = +\infty$ for some k then (P) is infeasible and the algorithm terminates. If $\beta(M_k) < +\infty\ \forall k$, then there is an infinite subsequence $M_{k_\nu}, \nu = 1, 2, \ldots,$ satisfying (3) and such that $x^* \in X$. If in addition*

$$\lim_{\nu\to+\infty} \beta(M_{k_\nu}) = \min\{F(x^*, y)\colon G(x^*, y) \preceq_K 0, y \in Y\}, \tag{6}$$

then the BB decomposition algorithm is convergent.

Condition (6) simply says that the lower bound must be *eventually exact* as $k \to +\infty$. Also note that for ensuring that $x^* \in X$ the condition $x \in M \cap X$ in (1) is essential and cannot be omitted.

Convergence Achieved with Lagrangian Bounds

In many important cases *Lagrangian bounds* can be used throughout the decomposition algorithm, so that for every partition set M:

$$\beta(M) = \sup_{\lambda\in K^*} \inf\{F(x, y) + \langle\lambda, G(x, y)\rangle\colon x \in M \cap X,\ y \in Y\}\,, \tag{7}$$

where $K^* = \{\lambda \in \mathbb{R}^m\colon \langle\lambda, u\rangle \geq 0\ \forall u \in K\}$ is the dual cone of K.

For every $t \geq 0$ define

$$v(t) = \sup_{\lambda\in K^*} \inf_{\substack{y\in Y\\ \|x-x^*\|\leq t, x\in X}} \{F(x, y)+\langle\lambda, G(x, y)\rangle\}, \tag{8}$$

where, as throughout in what follows, x^* denotes the limit point of an exhaustive filter of partition sets generated by the BB algorithm, i. e., an infinite nested sequence $\{M_{k_\nu}\}$ such that $\cap_{\nu=1}^{+\infty} M_{k_\nu} = \{x^*\}$.

Theorem 2 *Assume Lagrangian bounds are used throughout the BB decomposition algorithm, and:*
(A1) $v(t) \to v(0)$ as $t \searrow 0$.
(A2) $\sup_{\lambda\in K^*} \inf_{y\in Y}\{F(x^*, y) + \langle\lambda, G(x^*, y)\rangle\} = \min_{y\in Y} \sup_{\lambda\in K^*}\{F(x^*, y) + \langle\lambda, G(x^*, y)\rangle\}$.
Then the BB decomposition algorithm is convergent.

Condition A1 expresses the continuity of $v(t)$ at $t = 0$. Condition A2 requires that the duality gap be zero for the subproblem $\min_{y\in Y}\{F(x^*, y)\colon G(x^*, y) \preceq_K 0\}$.

Theorem 3 *Assume that $F(x, y), G_i(x, y), i = 1, \ldots, m$, are lower semi-continuous and:*
(i) There exists a compact set $Y^0 \subset Y$ such that

$$\begin{aligned} &(\forall x \in X)(\forall\lambda \in K^*)\\ &Y^0 \cap \operatorname{argmin}_{y\in Y}\{F(x, y)+\langle\lambda, G(x, y)\rangle\} \neq \emptyset; \end{aligned} \tag{9}$$

(ii) $\sup_{\lambda \in K^*} \inf_{y \in Y}\{F(x^*, y) + \langle \lambda, G(x^*, y)\rangle\} = \min_{y \in Y} \sup_{\lambda \in K^*}\{F(x^*, y) + \langle \lambda, G(x^*, y)\rangle\}$.

Then the BB decomposition algorithm using Lagrangian bounds is convergent. Furthermore, the function $x \mapsto \sigma(x) := \min\{F(x, y)\colon G(x, y) \preceq_K 0,\ y \in Y\}$ *is lower semicontinuous at* x^* *and satisfies*

$$\underset{\substack{x \in X \\ x \to x^*}}{\underline{\lim}} \ \sigma(x) = \min(\text{P}) . \tag{10}$$

Remark 1 Condition (i) cannot be replaced by the following weaker one

(*) *There exists a compact set* $Y^0 \subset Y$ *such that for each* $\lambda \in \mathbb{R}^m_+$ *and for each* $x \in X$ *either the set of optimal solutions of the problem*

$$\min\{F(x, y) + \sum_{i=1}^{m} \lambda_i G_i(x, y)\colon y \in Y\} \tag{11}$$

is empty or it has a nonempty intersection with Y^0.

Partly Convex Optimization Problems

An important class of problems (P) is constituted by *partly convex problems*, i. e., problems (P) with the following assumption:

(PCA) *For every fixed* $x \in X$ *the function* $y \mapsto F(x, y)$ *is convex, while the mapping* $y \mapsto G(x, y)$ *is K-convex.* The latter means that $G(x, \alpha y^1 + (1 - \alpha(y^2 \preceq_K \alpha G(x, y^1) + (1-\alpha)G(x, y^2)$ whenever $y^1, y^2 \in \mathbb{R}^p, 0 \le \alpha \le 1$.

Owing to (PCA), for every $\lambda \in K^*$ and fixed $x \in X$ the function $\langle \lambda, G(x, y)\rangle$ is convex and the problem $\min\{F(x, y) + \langle \lambda, G(x, y)\rangle :\ y \in Y\}$ is a convex optimization problem. Specific decomposition methods for this class of problems were developed earlier in [3,4], and more recently in [1]. Within the present framework, the convergence conditions can be specialized as follows.

A function $f\colon Y \to \mathbb{R}$ is said to be *coercive* on Y if $\lim_{y \in Y,\, y \to +\infty} f(y) = +\infty$. Clearly this is equivalent to saying that for any $\eta \in \mathbb{R}$ the set $\{y \in Y\colon f(y) \le \eta\}$ is bounded.

Theorem 4 *Assume (PCA) with* $F(x, y), G_i(x, y),\ i = 1, \dots, m$, *is lower semi-continuous on* $X \times Y$ *and continuous in x for fixed* $y \in Y$. *Assume further that:*

(S) For some $\lambda^* \in K^*$ *the function* $y \mapsto F(x^*, y) + \langle \lambda^*, G(x^*, y)\rangle$ *is coercive on* Y.

Then the BB decomposition algorithm using Lagrangian bounds is convergent and the function $\sigma(x) := \min\{F(x, y)\colon G(x, y) \preceq_K 0,\ y \in Y\}$ *is lower semicontinuous at* x^* *and satisfies*

$$\underset{\substack{x \in X \\ x \to x^*}}{\underline{\lim}} \ \sigma(x) = \min(\text{P}) .$$

Remark 2 Condition A2, sometimes referred to as *dual properness* at x^*, means that the subproblem $\min\{F(x^*, y)\colon G(x^*, y) \preceq_K 0,\ y \in Y\}$ has zero duality gap. When Y is bounded, condition (S) obviously holds, so by Theorem 4, both conditions A1 and A2 follow from (PCA) and the lower semicontinuity of $F(x, y), G_i(x, y), i = 1, \dots, m$. On the other hand, when Y is unbounded, dual properness (i. e., condition A2), even coupled with continuity of the functions involved, is not sufficient to guarantee condition A1. These results suggest that several methods developed in the literature for problems of the form (P) should be revised for validity.

Partly Linear Optimization

A subclass of the class of partly convex optimization problems is formed by partly linear optimization problems which have the general formulation

$$\min\{\langle c(x), y\rangle + \langle c^0, x\rangle \\ : A(x)y + B(x) \le b, r \le x \le s,\ y \ge 0\} , \tag{GPL}$$

where $x \in \mathbb{R}^n, y \in \mathbb{R}^p, c : \mathbb{R}^n \to \mathbb{R}^p,\ c^0 \in \mathbb{R}^n,\ A := \mathbb{R}^n \to \mathbb{R}^{m \times p},\ B \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m,\ r, s \in \mathbb{R}^n_+$.

A special case of interest is the "pooling and blending problem" from the petrochemical industry which can be stated as

$$\min\{c^T y : A(x)y \le b,\ y \ge 0,\ x \in X\} ,$$

where X is a box in $\mathbb{R}^n$ and $A(x)$ is an $m \times p$ matrix whose elements $a_{ij}(x)$ are continuous functions of x. Condition (S) in Theorem 4 now reads

For some $\lambda^* \in \mathbb{R}^m_+$
we have $c^T y + \lambda^*, A(x^*)y - b\rangle \to +\infty$
as $y \to +\infty$,

which clearly holds if and only if $\langle A(x^*), \lambda^*\rangle + c > 0$. For example this condition is fulfilled by the partly linear problems considered in [1], and also by the bilinear matrix inequalities problem studied in [5].

Extensions

The above decomposition method can be extended to a number of important nonconvex global optimization problems.

Partly Monotonic Optimization

A function $f(x)\colon \mathbb{R}^n \to \mathbb{R}$ is said to be increasing (decreasing, respectively) if $f(x) \le f(x')$ ($f(x) \ge f(x')$, respectively) whenever $x \le x'$ [7].

Theorem 5 *In problem (P) assume that* $X = [a, b] \subset \mathbb{R}^m_+, > F(x, y), G(x, y)$ *are continuous, and*

$$\begin{aligned}&(\text{PMA})\ F(x, y), G_i(x, y), i = 1, \dots, m,\\ &\textit{are increasing in } x \in [a, b] \textit{ for every fixed } y \in Y\,.\end{aligned}$$

Assume further that the set $\{y \in Y\colon G(b, y) \preceq_K 0\}$ *is contained in some box* Y^0. *Then, with lower bounds defined as*

$$\begin{aligned}M = [r, s] \subset [a, b] &\mapsto \beta(M)\\ &= \min\{F(r, y)\colon G(s, y) \preceq_K 0,\ y \in Y\}\,, \quad (12)\end{aligned}$$

the BB decomposition algorithm is convergent.

If $F(x, y), G_i(x, y), i = 1, \dots, m$, are monotonic in $y \in Y^0$ (or more generally, dm functions in $y \in Y^0$ [7]) then the subproblems in (12) are standard monotonic (or dm) optimization problems and can be solved by currently available algorithms [7,10].

Remark 3 Theorem 5 still holds if $F(x, y), G_i(x, y)$, $i = 1 \dots, m$ are decreasing in $x \in [a, b]$ for fixed $y \in Y$ and we define

$$\beta(M) = \min\{F(s, y)\colon G(r, y) \preceq_K 0\}\,.$$

Monotonic/Convex Optimization

Theorem 6 *In problem (P) assume* $X \subset [a, b]$, $F(x, y), G_i(x, y), i = 1, \dots, m$, *are continuous in (x,y), increasing in* $x \in [a, b]$ *for fixed* $y \in Y$, *and convex (affine, respectively) in* y *for fixed* $x \in [a, b]$. *Assume, in addition, that*

$$\begin{aligned}&(\text{ST})\ \textit{For some } \lambda^* \in K^*\\ &\textit{the function } F(a, y) + \langle\lambda^*, G(a, y)\rangle \textit{ is coercive on } Y\,.\end{aligned}$$

Then for every $M = [r, s] \subset [a, b]$, *the Lagrangian bound problem*

$$\sup_{\lambda\in K^*} \inf_{\substack{x\in M\cap X\\ y\in Y}} F(x, y) + \langle\lambda, G(x, y)\rangle \qquad (13)$$

is a convex (linear, respectively) program and the associated BB decomposition algorithm is convergent.

References

1. Ben-Tal A et al (1994) Global minimization by reducing the duality gap. Math Programm 63:193–212
2. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam, American Elsevier, New York
3. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Theory Appl 78:187–225
4. Geoffrion AM (1972) Generalized Benders Decomposition. J Optim Theory Appl 10:237–260
5. Tuan HD, Apkarian P, Nakashima Y (2000) A new Lagrangian dual global optimization algorithm for solving bilinear matrix inequalities. Int J Robust Nonlin Control 10:561–578
6. Tuy H (1998) Convex Analysis and Global Optimization. Kluwer, Dordrecht
7. Tuy H (2000) Monotonic Optimization: Problems and Solution Methods. SIAM J Optim 11(2):464–494
8. Tuy H (2005) On solving nonconvex global optimization by reducing the duality gap. J Global Optim 32:349–365
9. Tuy H (2007) On a Decomposition Method for Nonconvex Global Optimization. Optim Lett 1:245–258, doi:10.1007/s11590-006-0025-2
10. Tuy H, Minoux M, Hoai Phuong NT (2006) Discrete Monotonic Optimization With Application to a Discrete Location Problem. SIAM J Optim 17:78–97

Decomposition Principle of Linear Programming

Jørgen Tind
University Copenhagen, Copenhagen, Denmark

MSC2000: 90C06

Article Outline

Keywords

Optimization; Decomposition

A frequently applied approach in the history of optimization is that of decomposition, by which a large problem is decomposed into smaller problems. The principle of decomposition goes back to the seminal paper by G.B. Dantzig and P. Wolfe [2].

The basic model is a linear programming problem with two sets of constraints to be stated as follows:

$$\begin{cases} \max & cx \\ \text{s.t.} & A_1 x \le b_1 \\ & A_2 x \le b_2 \\ & x \ge 0. \end{cases} \tag{1}$$

where $c \in \mathbf{R}^n$, $A_1 \in \mathbf{R}^{m\times n}$, $b_1 \in \mathbf{R}^m$, $A_2 \in \mathbf{R}^{q\times n}$ and $b_2 \in \mathbf{R}^q$ are given constants and $x \in \mathbf{R}^n$ is a vector of variables.

The fundamental idea is to solve (1) by interaction between two optimization problems, one of which is subject to the first set of constraints and the other subject to the second set of constraints. Denote the second set by

$$X = \{x \ge 0\colon\ A_2 x \le b_2\}\,.$$

For simplicity we assume that X is bounded and nonempty. Hence X is a polytope. Let x^i denote an extreme point of X for $i \in P$ where P is the index set of all extreme points. According to the Minkowski representation theorem (see [1]), the polytope X can alternatively be represented as the convex hull of the extreme points, i. e.

$$X = \left\{ x = \sum_{i\in P} \lambda_i x^i\colon \quad \begin{matrix} \sum_{i\in P} \lambda_i = 1, \\ \lambda_i \ge 0 \text{ for } i \in P \end{matrix} \right\}.$$

If X is unbounded extreme rays are introduced in the representation of X leading to a straightforward extension of the subsequent considerations.

Hence (1) is equivalent to

$$\begin{cases} \max & \sum_{i\in P} cx^i\lambda_i \\ \text{s.t.} & \sum_{i\in P} A_1 x^i \lambda_i \le b_1 \\ & \sum_{i\in P} \lambda_i = 1 \\ & \lambda_i \ge 0. \end{cases} \tag{2}$$

Problem (2) operates with fewer rows than the original formulation (1). The variable x has been substituted by the variables λ_i. However, since the number of extreme points is usually very large in comparison with the dimension n of the problem, the number of λ-variables may also be very large, and it requires a big effort to enumerate and calculate all extreme points. Fortunately, this is unnecessary. In fact, by the Caratheodory theorem, at most $n + 1$ extreme points need to be considered, see for example [1]. The trouble is to find the correct ones.

Problem (2) is called the *full master problem* since all extreme points are introduced in the formulation. As already indicated we shall consider formulations dealing with only a subset of extreme points. For this purpose let $\overline{P}$ denote a subset of the index set P leading to a tightening of (2), called the *restricted master problem*.

$$\begin{cases} \max & \sum_{i\in \overline{P}} cx^i\lambda_i \\ \text{s.t.} & \sum_{i\in \overline{P}} A_1 x^i \lambda_i \le b_1 \\ & \sum_{i\in \overline{P}} \lambda_i = 1 \\ & \lambda_i \ge 0 \text{ for } i \in \overline{P}. \end{cases} \tag{3}$$

Assume here for simplicity that (3) is feasible. If not, additional techniques exist and may be applied to make the problem feasible. So an optimal basic solution exists together with optimal dual variables to be denoted by $y \in \mathbf{R}^m$ and $v \in \mathbf{R}$ according to the $m + 1$ rows of (3). By linear programming duality there exists a dual linear programming problem of the full master problem (2) with the variables (y, v) and with constraints

$$yA_1x^i + v \ge cx^i \quad \text{for all } i \in P. \tag{4}$$

Also by linear programming we know that an optimal solution has been found for the full master problem (2) if and only if the dual solution (y, v) satisfies (4). This may of course be checked through examination of all extreme points x^i. Fortunately, this is not necessary and here comes the major idea behind the decomposition principle. Instead we consider the following linear programming problem, the so-called *subproblem*.

$$\begin{cases} u = \max & (c - yA_1)x \\ \text{s.t.} & A_2 x \le b_2 \\ & x \ge 0. \end{cases} \tag{5}$$

Decomposition Principle of Linear Programming, Table 1

Step 1	Calculate an optimal dual solution (y, v) of the restricted master problem (3).
Step 2	Determine an extreme point by solving the subproblem (5). If (6) is violated expand the index set $\overline{P}$ by including the extreme point and go to Step 1.
Step 3	An optimal solution has been obtained by the solution of the last master problem as $x = \sum_{i \in \overline{P}} x^i \lambda_i$

By assumption an optimal solution exists among the extreme points of X. Let $i^* \in P$ denote the index of an optimal extreme point. Observe that the objective function calculates the maximal value u of $cx^i - yA_1x^i$ among all extreme points x^i in X. Hence by (4) it remains to check if

$$u \leq v. \tag{6}$$

If so, then all constraints of (4) are satisfied and we may stop. Otherwise introduce the elements $(cx^{i^*}, A_1x^{i^*})$ as a new column in the restricted masterproblem (3) and continue by solving it.

The above discussion can be summarized into the *algorithm* in Table 1.

The number of extreme points in X is finite. Hence only a finite number of mutually different columns may be introduced in the restricted master problem. This implies that the algorithm must terminate in a finite number of steps.

The decomposition principle is suited to solve large scale problems. Moreover it has a nice economic interpretation. Consider a central level and a sublevel of a decentralized organization. The central level operates on the first set of constraints $A_1x \leq b_1$ and the sublevel on the second set of constraints $A_2x \leq b_2$. The right hand sides b_1, b_2 may be interpreted as resources for the central level and sublevel, respectively. During the course of the algorithm information is communicated from one level to the other. The central level solves the restricted master problem and as a result marginal prices y on central resources are communicated to the sublevel. The sublevel solves the subproblem in which the objective function incorporates the costs for utilization of the central resources. The sublevel then suggests activities x^i to be incorporated at central level. During the iterations no direct information about the coefficients in the constraints is communicated between the central level and the sublevel. Instead price information is communicated from the central level to the sublevel and the algorithm is the fundamental method among the so-called price-directive procedures.

In most applications the last set of constraints, $A_2x \leq b_2$ have a so-called *block-angular structure*, in which the variables are grouped into independent blocks. This implies that the subproblem separates into multiple independent problems. In an economic context the block-angular structure reflects a division of the sublevel into multiple independent sublevels, each of which communicates directly with the central level.

A counterpart of the present Dantzig–Wolfe decomposition procedure exists in the form of Benders decomposition. In linear programming they are dual in the sense that application of Benders decomposition on the dual program of the original problem (1) is equivalent to the direct application of the present procedure on (1).

See also

- ▶ Generalized Benders Decomposition
- ▶ MINLP: Generalized Cross Decomposition
- ▶ MINLP: Logic-Based Methods
- ▶ Simplicial Decomposition
- ▶ Simplicial Decomposition Algorithms
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Successive Quadratic Programming: Decomposition Methods

References

1. Bazaraa MS, Jarvis JJ, Sherali HD (1990) Linear programming and network flows. Wiley, New York
2. Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programs. Econometrica 29:767–778

Decomposition Techniques for MILP: Lagrangian Relaxation

VISWANATHAN VISWESWARAN
SCA Technologies LLC, Pittsburgh, USA

MSC2000: 90C90, 90C30

Article Outline

Keywords

Decomposition; Lagrangian relaxation; Lagrangian decomposition

A large number of combinatorial optimization problems can be viewed as potentially 'easy' problems to solve that are complicated by a set of side constraints. If the complicating constraints were removed, the resulting problem would have constraints possessing a high degree of structure, for which many efficient algorithms exist. One of the most attractive methods to exploit this property is the Lagrangian relaxation technique, in which the complicating constraints are dualized and then removed from the constraint set. This class of methods, originally proposed by various authors for a variety of problems, and later generalized in [3], has proven highly successful in solving otherwise difficult combinatorial problems. For an excellent introduction to the approach and its applications, see [1,2].

Consider the following problem (P):

$$\max f^{\top} x \tag{1}$$

such that

$$Ax \leq b \tag{2}$$

$$Cx \leq d \tag{3}$$

$$x \in X, \tag{4}$$

where x is an n-vector, b is an m-vector and d is a k-vector, and f, A and C have conformable dimensions. Some or all of the x variables can be integers (i. e. $X \subseteq Z^n$). It is assumed that there is a finite and nonempty set of solutions to the constraints in the problem (2)–(4). Let (LP) represent problem (P) with any integrality constraints in X removed.

The following notation is used in the sequel. For any problem $(\cdot)$, $\mathrm{OS}(\cdot)$ is its optimal set, and $V(\cdot)$ represents its optimal value. For any set S, $\mathrm{Co}(S)$ represents the convex hull of the set.

The *Lagrangian relaxation* (LR_u) of (P) relative to the constraint set (2) and a conformable nonnegative vector u is defined as

$$\begin{cases} \max\limits_{x} & f^{\top} x + u(b - Ax) \\ \text{s.t.} & Cx \leq d \\ & x \in X. \end{cases}$$

The problem

$$(\mathrm{LR}) = \min_{u \geq 0} (\mathrm{LR}_u)$$

is called the *Lagrangian dual* relative to (2). The constraints (2) are referred to as the 'dualized constraints', and u is the corresponding multiplier or dual vector. The constraints should be chosen so that the remaining set $C_x \leq d$ possesses desirable structure. For example, (3) might only specify up per bounds on the variable, or might be a single 'knapsack' constraint of the form $\sum_{i=1}^{n} x_i \leq 1$.

The first point to note about (LR_u) is that it always provides an upper bound for (P), i. e.

$$V(\mathrm{LR}_u) \geq V(P).$$

This can easily be seen from the fact that $u \geq 0$ and $Ax \leq b$ for any solution x which is optimal for (P). In practice, it is desirable to have $V(LR_u)$ as close to $V(P)$ as possible. Moreover, there is already an LP relaxation of (P), obtained by dropping the integrality requirement on x. How does $V(\mathrm{LR}_u)$ relate to $V(\mathrm{LP})$? To answer this question, consider the following relaxation of (P), denoted by (P^*):

$$\begin{cases} \max & f^{\top} x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathrm{Co}\{Cx \leq d\colon\ x \in X\}. \end{cases}$$

It can be shown ([3]) that

$$V(P) \leq V(P^*) = V(\mathrm{LR}) \leq V(\mathrm{LP}).$$

The equality for the optimal values of problems (P^*) and (LR) follows from the fact that they are duals of each other. Moreover, it can be shown that if the multipliers for the constraints obtained from solving the LP relaxation were used, the resulting Lagrangian relaxation provides a bound at least as tight as the bound from (LP). Also, if $\overline{u}$ is an optimal solution for (LR), with $A\overline{x} \leq b$ and $u(A\overline{x} - b) = 0$, then $\overline{x}$ is optimal for (P).

When are the inequalities above strict? This can be shown through the following *integrality property*, again due to [3]: The optimal value of (LR$_u$) is not changed by dropping the integrality condition on the x variables.

If the integrality property (also referred to as the *complementary slackness* property) holds, then

$$V(P) = V(P^*) = V(\text{LR}) = V(\text{LP}).$$

In this case, therefore, Lagrangian relaxation can do no better than the standard LP relaxation for (P). For a large number of practical problems, however, this property does not hold. This fact allows (LR$_u$) to be used in place of (LP) to provide lower bounds in a branch and bound algorithm.

Lagrangian decomposition

A drawback of the Lagrangian relaxation (LR) described above is that only one of the possibly many special structured constraint sets embedded in the problem can be exploited. This results in the loss of structure of all the dualized constraints. One way to avoid this is to use Lagrangian decomposition ([4,5,6,7]).

Introducing a new set of 'copy' constraints $y = x$, problem (P) is equivalent to

$$\max_{x,y} f^\top x$$

such that

$$Ay \leq b \tag{5}$$

$$Cx \leq d \tag{6}$$

$$y = x \tag{7}$$

$$x \in X, \qquad y \in Y, \tag{8}$$

where $X \subseteq Y$. Dualizing the 'copy' constraints (7) results in

$$\max f^\top x + v(y - x)$$

such that

$$Ay \leq b \tag{9}$$

$$Cx \leq d \tag{10}$$

$$x \in X, y \in Y, \tag{11}$$

which can be decomposed to the following problem (LD$_v$):

$$\begin{cases} \max\limits_x & F_1(x) \\ \text{s.t.} & Cx \leq d \\ & x \in X \end{cases} + \begin{cases} \max\limits_y & F_2(y) \\ \text{s.t.} & Ay \leq b \\ & y \in Y \end{cases},$$

where $F_1\ (x) = (f - v)^\top\ x$ and $F_2\ (y) = v^\top\ y$. The Lagrangian decomposition dual (LD) can then be defined to be

$$\min_{v \geq 0} V(\text{LD}_v)$$

If $\overline{u}$ is an optimal solution to (LR), then, with $\overline{v} = \overline{u} \cdot A$, it can be shown that

$$V(\text{LD}_{\overline{v}}) = V(\text{LR}_{\overline{u}}) - \overline{u}(b - A\overline{y})$$

and therefore

$$V(\text{LD}) \leq V(\text{LR}).$$

It is possible to define an integrality property ([5]) such that if either the x- or the y-problem has the property, then V(LD) will be equal to the stronger of the bounds obtained from the two Lagrangian relaxations corresponding to each set of constraints.

Lagrangian decomposition (LD) has several advantages over (LR). Every constraint in the original problem appears in one of the subproblems. It thus avoids having to choose between the various sets of structured constraints. Secondly, as shown above, the bounds from (LD) can be tighter than those from (LR). Furthermore, the bound can be tightened by adding surrogate constraints (for example, a surrogate constraint from $Ax \leq b$ can be added to the x-problem in (LD). Thirdly, analogous to (LR), it can be shown that (LD) is really the dual of a primal problem involving the optimization of the original objective function over the intersection of the convex hulls of the two constraint sets. Finally, empirical results suggest that when using heuristics based on Lagrangian decomposition, any intermediate solutions found in the solution of (LD) lead to better solutions for problem (P) as compared to solutions found by Lagrangian relaxation.

Aggregation Schemes

The main drawback of the Lagrangian decomposition method is that a large number of multipliers (v) are introduced, one for each of the copy variables. The calculation of v can be time consuming at each step. Moreover, the convergence of the scheme can be slowed significantly by the larger number of directions (for the multipliers) to search. In order to avoid this, an alternate approach that has been suggested [8] is to aggregate some or all of the variables using a simplified linear function, and then to dualize the resulting copy constraints. The purpose of the aggregation is to substantially reduce the number of dual variables, while still maintaining the constraint structure as in the standard decomposition.

Let $A \equiv [A_1 \mid A_2]$, and $x \equiv \binom{x^1}{x^2}$, with $x^1 \in \mathbf{R}^{n_1}$ and $x^2 \in \mathbf{R}^{n-n_1}$. Introduce the copy variables $y \equiv \binom{y^1}{y^2} = x$ and the constraints

$$x^1 = y^1 \quad \text{and} \quad A_2 x^2 = g(y^2),$$

where $g(\cdot)$ is the *aggregation function* (for example, $g(y^2) = A_2\, y^2$, or $g(y^2) = y^2$). Then, the problem (P) can be written as

$$\max f^\top x$$

such that

$$A_1 y^1 + g(y^2) \le b \tag{12}$$

$$Cx \le d \tag{13}$$

$$x^1 = y^1 \tag{14}$$

$$A_2 x^2 = g(y^2) \tag{15}$$

$$x \in X, \quad y \in Y, \tag{16}$$

where $X \subseteq Y$. Dualizing the constraints (14) and (15) leads to

$$\begin{cases} \max & f^\top x + w^1(y^1 - x^1) + w^2(g(y^2) - A_2 x^2) \\ \text{s.t.} & A_1 y^1 + g(y^2) \le b \\ & Cx \le d \\ & x \in X, \quad y \in Y, \end{cases}$$

which can be decomposed to the problem (LDA$_w$), given by

$$\begin{cases} \max\limits_x & F_1(x) \\ & Cx \le d \\ & x \in X \end{cases} + \begin{cases} \max\limits_y & F_2(y) \\ & A_1 y^1 + g(y^2) \le b \\ & y \in Y, \end{cases}$$

where $F_1(x) = f^\top x - w^1 x^1 - w^2 A_2 x^2$ and $F_2(y) = w^1 y^1 + w^2 g(y^2)$. The corresponding dual problem (LDA) is then defined by

$$\min_{w \ge 0} \text{LDA}_w$$

It can then be proved that for any optimal solution of (LR) defined by $\overline{u} \in \text{OS(LR)}$, with $\overline{w_1} = \overline{u} A_1$ and $\overline{w^2} = \overline{u}$,

$$V(\text{LDA}_{\overline{w}}) \le V(\text{LR}_{\overline{u}})$$

and therefore

$$V(\text{LDA}) \le V(\text{LR}).$$

This inequality is strict only if the second subproblem in (LDA$_w$) (i. e. the problem of maximizing $F_2(y)$) does not satisfy the integrality (*complementary slackness*) property. Moreover, this inequality holds only for the Lagrangian relaxation with those constraints that have been aggregated. Any other Lagrangian relaxation defined for problem (P) by dualizing other sets of constraints will not necessarily satisfy this inequality.

Similarly, defining $\overline{v} \equiv (\overline{w^1}, \overline{w^2})^\top$, it can also be easily shown that

$$V(\text{LD}_{\overline{v}}) \le V(\text{LDA}_{\overline{w}})$$

and therefore

$$V(\text{LD}) \le V(\text{LDA}).$$

In general, therefore, the bound obtained by aggregating some or all of the variables is stronger than the bound obtained from Lagrangian relaxation but weaker than the bounds from standard Lagrangian decomposition. However, while the standard (LD) introduces n multipliers, (LDA) has $n_1 + m$, which can be considerably less depending on the number of dualized constraints. Moreover, any inherent problem structure is still maintained in (LDA). The aggregate formulation can thus be viewed as a reasonable compromise between tightness of the bounds and speed of solution. As in the case of standard decomposition, (LDA) can be defined by aggregating different subsets of variables. Unfortunately, it is not always apparent a priori which is the best choice for obtaining the tightest bound, and various alternatives may have to be tried in practice.

One possible method of exploiting the potential inequalities in these various bounds is as follows:

1) solve (LR) to obtain $\overline{u}$;
2) if the integrality property does not hold, set $\overline{w^1} = \overline{u}A_1$ and $\overline{w^2} = \overline{u}$, and solve ($\mathrm{LDA}_{\overline{u}}$); this problem is guaranteed to give a tighter bound than (LR);
3) set $\overline{v} \equiv \left(\frac{\overline{w^1}}{\overline{w^2}}\right)$, and solve ($\mathrm{LD}_{\overline{v}}$). Note that if the aggregate function $g(y^2)$ is of the form A_2y, this step will not yield any improvement.

Practical Issues

Because (LR), (LD) and (LDA) can all provide tighter bounds than (LP), any one of the relaxations can be used in place of (LP) to provide upper bounds in a classical branch and bound algorithm to solve (P). Consequently, the choice of the relaxation scheme used, as well as the quality of the bounds obtained, is of considerable importance. These are discussed in some detail below.

Choosing Among Alternate Relaxations

Often, there are several choices for the constraints to be dualized. For example, consider the generalized assignment problem

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

such that

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \ldots, n, \tag{17}$$

$$\sum_{j=1}^{n} a_{ij} x_{ij} \le b_i, \quad i = 1, \ldots, m, x_{ij} = 0 \text{ or } 1 \quad \forall i, j. \tag{18}$$

By dualizing the first set of constraints (17), this problem reduces to m knapsack problems. Conversely, dualizing (18) results in a generalized upper bound (GUB) problem in 0–1 variables. Which of the two relaxations should be used? There are two conflicting factors involved here, namely the tightness of the bounds and the ease of solution of the problem. It would be wise to select a relaxation that yields a problem that is fairly easy to solve, but not so easy that the bounds are very loose. In general, it is difficult to know this a priori. In some instances, however, the test of the integrality property can be useful. For example, in the generalized assignment problem, the integrality property holds for the second relaxation but not the first, suggesting that the second relaxation will yield a tighter bound.

For the case of the Lagrangian decomposition, this issue is not important since all constraints are maintained in one of the subproblems. However, if aggregation is being used, then again, it is in general hard to know a priori which variables to aggregate and which ones to copy. Often, the best solution is to try various alternatives and use the computational results to guide the choice.

Choice of Multipliers

It is clear that for (LR), the best choice for u is an optimal solution to the problem

$$\min_{u \ge 0} V(\mathrm{LR}_u)$$

since this will yield the tightest bound V(LR). Similarly, the best dual vectors v and w for (LD) and (LDA) are those from the optimal solutions to the respective dual problems. Unfortunately, these optimal values for the dual variables cannot be determined a priori, and therefore, an interactive procedure is the only viable approach to improv ing the value of u, v or w. Below, a couple of techniques for updating u for (LR) are discussed, but the methods are just as relevant for updating v and w for (LD) and (LDA).

In general, the function $V(\mathrm{LR}_u)$ is piecewise linear, convex and differentiable at all points except where the Lagrangian problem has multiple optimal solutions. This observation has led to the development of *subgradient techniques* for the determining the u that minimizes $V(\mathrm{LR}_u)$. This method is similar to traditional gradient methods, except that at the nondifferentiable points, it chooses randomly from the set of optimal Lagrange solutions. Given an initial value u^0 (typically $u^0 = 0$), a sequence $\{u^k\}$ is generated by the formula

$$u^{k+1} = \max\{u^k + t^k(Ax^k - b), 0\},$$

where x^k is an optimal solution to (LR_{u^k}) and t^k is a scalar stepsize, generally designed to be a decreasing sequence converging to zero. It is not possible to prove optimality in this method, so usually it is terminated upon reaching a specified number of iterations. Because of its simplicity, the subgradient technique is generally the method of first choice when solving (LR).

An alternate way to update u is to use *dual descent* algorithms, also referred to as *multiplier adjustment* methods. In these methods, the sequence u^k is generated by

$$u^{k+1} = u^k + t^k d^k,$$

where d^k is an *descent direction*, determined from the directional derivative of $V(\mathrm{LR}_{u^k})$ using a finite set of directions. Typically, the direction of steepest descent is chosen, and the stepsize t^k is the one that minimizes $V(\mathrm{LR}_{u^k+td^k})$. Unlike subgradient optimization, this procedure guarantees monotonic bound improvement. Moreover, it may only adjust a few multipliers at each iteration, resulting in improved computational performance. However, for general problems, the set of directions to choose from can be very large, resulting in very poor descent. It is therefore essential to tailor these methods to particular problems to exploit their structure in determining the set of directions.

Applications

Lagrangian relaxation and decomposition have been successfully applied to solve a large number of practical combinatorial problems. These include the generalized assignment problem, capacitated facility location problem, the traveling salesman problem and instances of the general mixed integer programming problem. For each of these problems, the constraints contain well-understood structures such as knapsack, spanning tree and generalized upper bound constraints, thus facilitating the dualization of the other complicating constraints. For a number of problems, these techniques represent the best available solution method. Computational results for these and other problems indicate that the bounds provided by (LR) and (LD) can be extremely sharp. These results have led to (LD) and (LDA) being considered among the best available solution methods for solving these problems.

See also

- ▶ Branch and Price: Integer Programming with Column Generation
- ▶ Integer Linear Complementary Problem
- ▶ Integer Programming
- ▶ Integer Programming: Algebraic Methods
- ▶ Integer Programming: Branch and Bound Methods
- ▶ Integer Programming: Branch and Cut Algorithms
- ▶ Integer Programming: Cutting Plane Algorithms
- ▶ Integer Programming Duality
- ▶ Integer Programming: Lagrangian Relaxation
- ▶ Lagrange, Joseph-Louis
- ▶ Lagrangian Multipliers Methods for Convex Programming
- ▶ LCP: Pardalos–Rosen Mixed Integer Formulation
- ▶ Mixed Integer Classification Problems
- ▶ Multi-objective Integer Linear Programming
- ▶ Multi-objective Mixed Integer Programming
- ▶ Multi-objective Optimization: Lagrange Duality
- ▶ Multiparametric Mixed Integer Linear Programming
- ▶ Parametric Mixed Integer Nonlinear Optimization
- ▶ Set Covering, Packing and Partitioning Problems
- ▶ Simplicial Pivoting Algorithms for Integer Programming
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Time-Dependent Traveling Salesman Problem

References

1. Fisher ML (1981) The Lagrangean relaxation method for solving integer programming problems. Managem Sci, 27(1):1–18
2. Fisher ML (1985) An applications oriented guide to Lagrangean relaxation. Interfaces 15(2):10–21

3. Geoffrion AM (1974) Lagrangian relaxation and its uses in integer programming. Math Program Stud 2:82–114
4. Glover F, Klingman D (1988) Layering strategies for creating exploitable structure in linear and integer programs. Math Program 40:165–182
5. Guignard M, Kim S (1987) Lagrangean decomposition: A model yielding stronger Lagrangean bounds. Math Program 39:215–228
6. Guignard M, Kim S (1987) Lagrangean decomposition for integer programming: Theory and applications. RAIRO Oper Res 21(4):307–323
7. Jörsten K, Näsberg M, Smeds P (1985) Variable splitting–a new Lagrangean relaxation approach to some mathematical programming models. Techn Report Dept Math Linkoping Inst Technol, Sweden MAT-R-85-04
8. Reinoso H, Maculan N (1992) Lagrangean decomposition in integer linear programming: A new scheme. INFOR 30:1–5

De Novo Protein Design Using Flexible Templates

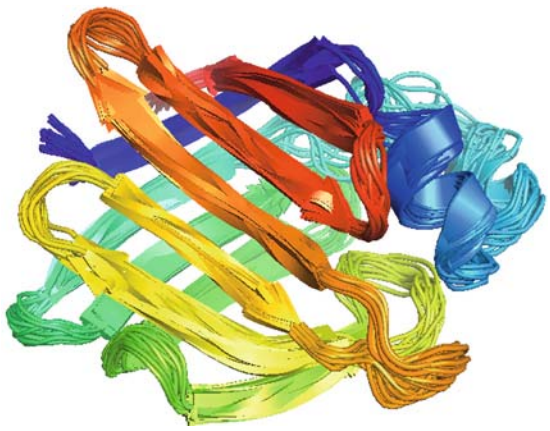
HO KI FUNG, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 92D20, 46N10, 90C10

Article Outline

Introduction

The assumption of a fixed template for de novo peptide and protein design is highly questionable [41], as protein is commonly known to exhibit backbone flexibility, as illustrated by the superposition of NMR structures in Fig. 1. De novo design templates were observed to allow



De Novo Protein Design Using Flexible Templates, Figure 1
Template flexibility as illustrated by the superposition of the 20 NMR structures of apo intestinal fatty acid binding protein (Protein Data Bank code 1AEL)

residues that would not have been permissible had the backbone been fixed [34]. The Mayo group claimed that their ORBIT protein design program was robust against 15% change in the backbone. Nevertheless, they found in a later case study on T4 lysozyme that core repacking to stabilize the fold was difficult to achieve without considering a flexible template [37]. The secondary structures of α-helices and β-sheets actually display twisting and bending in the fold, and Emberly et al. [6,7] applied principal component analysis of database protein structures to quantify the degree and modes of their flexibilities.

In this chapter we classify the various methods of incorporating backbone flexibility into the design template into three main types according to their treatment of the backbone and side-chain conformations. The first type involves considering a set of multiple discrete templates and performing de novo design with discrete rotamers on each of the templates under the fixed-backbone assumption. The second type considers a continuum template by means of algebraic parameterization of the backbone and variation of the parameters to allow for backbone movement during sequence selection. However, it still employs rotamer libraries to simplify the side-chain conformations. Through novel sequence selection formulations [14] and pairwise contact potentials which are discretized over distance bins [35,44,45], the third type considers a continuum design template in which the Cα–Cα distances and dihedral angles assume continuous values between upper and lower bounds observed from the template structures [9], and confirms sequence specificity to the target fold based on these bounded continuous distances and angles via NMR structure refinement methods [16,17] rather than the discrete rotamer approach. For each category we will quote some examples of successes of de novo peptide and protein design.

Flexible Template via Multiple Discrete Templates and Discrete Rotamers

By incorporating protein backbone flexibility via discrete templates and discrete rotamers, de novo protein design frameworks either separate sequence selection and backbone movement explicitly or iterate between sequence space and structure space [3]. Notice that in both cases, the sequence search methods outlined in the previous section are all applicable, as fixed backbones and discrete rotamers are still assumed.

Approaches Which Separate Sequence Selection and Backbone Movement

These approaches consider an ensemble of fixed backbones, searches for sequences for each of them assuming a fixed template, and finally identify the best solutions from all the results. Successes using different kinds of search algorithms include the ones described next.

Successes Using Dead-End Elimination By varying the supersecondary structure parameters, Ross et al. [46] and Su and Mayo [48] generated several sets of perturbed backbones from the native structure and redesigned the core of the $\beta 1$ domain of the streptococcal protein using the DEE algorithm under the fixed-template assumption for each backbone. Confirmed by NMR experiments, six of the seven sequences tested folded into nativelike structure.

Successes Using the Self-Consistent Mean Field Method Kono and Saven [29] applied their self-consistent mean field based protein combinatorial library design strategy to a set of similar backbone structures to obtain new sequences that are robust to distance changes in the template for the immunoglobulin light chain-binding domain of protein L.

Successes of Monte Carlo Methods/Genetic Algorithms The Pande group generated families of 100 fixed templates within 1 Å root-mean-square deviation (rmsd) from the initial backbone using a Monte Carlo method. With these fixed-template ensembles, they performed de novo design, which was based on genetic algorithms, on their Genome@home distributed grid system for 253 naturally occurring proteins. They obtained sequences that exhibited higher diversity than the corresponding natural sequence alignments, as well as good agreement on the sequence entropies of the designed sequences from the same template family [32,33].

In order to incorporate protein flexibility, Kraemer-Pecore et al. [30] executed a Monte Carlo simulation to generate 30 fixed backbones that were within 0.3 Å rmsd of the initial template. A genetic-algorithm-based sequence prediction algorithm [43] which combines filtering and sampling rotamers and energy minimization was then employed for sequence search on each template under the fixed backbone assumption. The work led to the identification of a sequence that folded into the WW domain.

In designing protein conformational switches, Ambroggio and Kuhlman [1,2] also used the Monte Carlo based RosettaDesign to search for sequences for multiple fixed-template structures.

Approaches Which Iterate Between Sequence Space and Structure Space

There are two good examples which belong to this class. The first example is a genetic algorithm/Monte Carlo based framework used by Desjarlais and Handel [5], in which a starting population of backbones is generated by small angle perturbations to the template, rotamers are randomly selected on each backbone, and a genetic algorithm is subsequently used which exchanges not only rotamers but also backbone torsional information in recombination. The framework is ended with a Monte Carlo stage which refines the backbone structures. Using this novel approach, Desjarlais and Handel [5] designed three new core variants of the protein 434 cro. They also compared results on 434 cro and T4 lysozyme with those obtained earlier using fixed-template models and found that they were similar, given that the fixed-template models scan over a much larger rotamer space.

The second one was proposed by Kuhlman et al. [31] and Saunders and Baker [31,47]. Their method starts with a set of initial backbones, searches by a Monte Carlo method for the sequence with the lowest energy for each of them, performs atomic-resolution structure prediction for the sequences to allow shifts in the structure space, and continues until the number of iterations hits a predetermined number. They successfully designed a new sequence for Top 7, a 93-residue α/β protein with a novel fold [31]. They also claimed that the new method better captures sequence variation than approaches that separate sequence selection and backbone movement explicitly.

Flexible Template via Continuum Template and Discrete Rotamers

This method of constituting a continuum template via backbone parameterization and performing sequence search from rotamer libraries was proposed by Harbury et al. [18,19,40]. On the basis of the algebraic parameterization equations developed for coiled-coils by Crick [4], they allowed backbone movement by treating the parameters as variables during sequence search for energy minimization, which was in turn done by the local optimization methods of steepest descent minimization and adopted-basis Newton–Raphson minimization.

Successes

Harbury et al. [18,19,40] adopted this approach to design a family of α-helical bundle proteins with right-handed superhelical twist. The crystal structure of the designed sequences with the optimal specificity was experimentally validated to match the design template.

Flexible Template via Continuum Template and NMR Structure Refinement

Considering discrete rotamers is certainly not the best approach to adopt in de novo design, as about 15% of side-chain conformations are not represented by common rotamer libraries [8]. A recent two-stage de novo design approach proposed by the Floudas group [10,11,26,27] considers a continuum design template without using discrete rotamers for the possible side-chain conformations. The first stage selects a rank-ordered list of low-energy sequences using novel quadratic assignment-like models [13,14] driven by pairwise residue contact potentials, which were developed by the group by solving a linear programming parameter estimation problem, requiring that the native conformations for a large training set of 1250 proteins be ranked energetically more favorably than their high-resolution decoys [35,44,45]. The forcefields developed were found to produce very good *Z* scores in recognizing the native folds for a large test set of proteins [35,44,45]. Rather than being continuous, the dependence of contact potential on distance is discretized into bins. This designed feature serves to make the energy objective function insensitive to a limited degree of backbone movement. For example, in the high resolution C^{α}-C^{α} forcefield [44], if the pair of amino acids selected at two positions *i* and *k*, which are 3.5 Å apart in the template, are Arg and Glu, respectively, their energy contribution to the objective function is Minus 7.77 kcal/mol. Despite small distance variations, this energy value is constant for all Arg–Glu interactions as long as the C^{α} positions of the two residues are 3–4 Å (bin 1) apart. To perform sequence selection based on a flexible template of multiple structures, Fung et al. [14] also developed two novel formulations: a weighted model which considers the distance between any two positions as the weighted average of their distances in all structures, and a binary distance bin model that decides which bin the distance falls into during energy optimization. The latter approach is in a sense similar to the backbone parameterization approach of Harbury et al. [18,19,40] in which there are distance variables associated with the backbone.

The second stage of the approach confirms fold specificity of the sequences generated in the first stage based on a full-atomistic forcefield. The group used to perform the task via ASTRO-FOLD [20,21,22,23,24,25,28,36], a protein structure prediction method via global optimization. Conformational ensembles are generated for each sequence under two sets of conditions. In the first circumstance, the structure is constrained to vary, with some imposed fluctuations, around the template structure. In the second condition, a free folding calculation is performed for which only a limited number of restraints (e. g., disulfide bridges), but not the underlying template structure, are enforced. The relative fold specificity of the sequence, f_{spec}, can be found by summing the statistical weights for those conformers from the free folding simulation that resemble the template structure (denoted as set *temp*), and dividing this sum by the summation of statistical weights for all conformers from the free folding simulation (denoted as set *total*):

$$f_{\text{spec}} = \frac{\sum_{i \in temp} \exp[-\beta E_i]}{\sum_{i \in total} \exp[-\beta E_i]}$$

where $\exp[-\beta E_i]$ is the statistical weight for conformer i.

Note that in this nonrotamer approach, in both the template-constrained and the free folding calculations, all continuous C^{α}–C^{α} and angle values between upper and lower bounds input by the user are considered in sampling the conformers. True backbone flexibility [9] is thus conserved.

Lately the Floudas group developed an approximate fold validation method which is computationally less expensive than ASTRO-FOLD. Through the CYANA 2.1 software for NMR structure refinement [16,17], an ensemble of several hundred conformers is generated for both a new sequence from the first stage and the native sequence. The energies of the conformers are then minimized using TINKER [42], and the fold specificity of the new sequence is calculated using the formula

$$f_{\text{spec}} = \frac{\sum_{i \,\in\, \text{conformers for new sequence}} \exp[-\beta E_i]}{\sum_{i \,\in\, \text{conformers for native sequence}} \exp[-\beta E_i]}$$

based on the assumption that the fold specificity to the flexible template is unity for the native sequence.

Like the fold-validation method via ASTRO-FOLD, all continuous distance and dihedral angle values between their upper and lower bounds, which are input into CYANA on the basis of observations about the template structures, are considered in generating the conformers. This distinguishes the method from the common rotamer approach in which only discrete side-chain conformations are allowed.

Successes

The novel two-stage de novo strategy was applied to (1) the design of new sequences for compstatin, a synthetic 13-residue cyclic peptide that binds to complement protein 3 (C3) and inhibits the activation of the complement system (part of innate immunity) [10,26,27,38,39], (2) the design of a potential peptide-drug candidate derived from the C-terminal sequence of the C3a fragment of C3 [15], and (3) the full sequence of human β-defensin-2, a 41-residue cationic peptide in the immune system [12]. In the case of the compstatin redesign, sequences with 16-fold and 45-fold improvement in specificity over the native sequence were confirmed in experiments [26,27]. For the design of the peptide drug from C3a, the best sequence identified corresponds to 15-fold improvement [15].

References

1. Ambroggio XI, Kuhlman B (2006) Computational design of a single amino acid sequence that can switch between two distinct protein folds. J Am Chem Soc 128:1154–1161
2. Ambroggio XI, Kuhlman B (2006) Design of protein conformational switches. Curr Opin Struct Biol 16:525–530
3. Butterfoss GL, Kuhlman B (2006) Computer-based design of novel protein structures. Annu Rev Biophys Biomol Struct 35:49–65
4. Crick FHC (1953) The fourier transform of a coiled-coil. Acta Crystallogr 6:685–689
5. Desjarlais J, Handel T (1999) Side chain and backbone flexibility in protein core design. J Mol Biol 290:305–318
6. Emberly E, Mukhopadhyay R, Tang C, Wingreen N (2003) Flexibility of α-helices: Results of a statistical analysis of database protein structures. J Mol Biol 327:229–237
7. Emberly E, Mukhopadhyay R, Tang C, Wingreen N (2004) Flexibility of β-sheets: Principal component analysis of database protein structures. Proteins: Struct Funct Genet 55:91–98
8. Filippis VD, Sander C, Vriend G (1994) Predicting local structural-changes that result from point mutations. Prot Eng 7:1203–1208
9. Floudas CA (2005) Research challenges, opportunities and synergism in systems engineering and computational biology. AIChE J 51:1872–1884
10. Floudas CA, Fung HK (2006) Mathematical modeling and optimization methods for de novo protein design. In: Rigoutsos I, Stephanopoulos G (eds) Systems Biology, vol II, Oxford University Press, pp 42–66
11. Floudas CA, Fung HK, Morikis D, Taylor MS, Zhang L (2007) Overcoming the key challenges in de novo protein design: Enhancing computational efficiency and incorporating true backbone flexibility. In: Mondaini R (ed) Modeling of Biosystems: An Interdisciplinary Approach. Springer, Berlin
12. Fung HK, Floudas CA, Morikis D, Taylor MS, Zhang L (2007) Toward full-sequence de novo protein design with flexible templates for human beta-defensin-2. Biophys J (in print)

13. Fung HK, Rao S, Floudas CA, Prokopyev O, Pardalos PM, Rendl F (2005) Computational comparison studies of quadratic assignment like formulations for the in silico sequence selection problem in de novo protein design. J Comb Optim 10:41–60
14. Fung HK, Taylor MS, Floudas CA (2007) Novel formulations for the sequence selection problem in de novo protein design with flexible templates. Optim Methods Softw 22:51–71
15. Fung HK, Taylor MS, Floudas CA, Morikis D, Lambris JD (2007) Redesigning complement 3a based on flexible templates from both x-ray crystallography and molecular dynamics simulation. in preparation
16. Guntert P (2004) Automated nmr structure calculation with cyana. methods mol biol. J Mol Biol 278:353–378
17. Guntert P, Mumenthaler C, Wuthrich K (1997) Torsion angle dynamics for nmr structure calculation with the new program dyana. J Mol Biol 273:283–298
18. Harbury P, Plecs J, Tidor B, Alber T, Kim P (1998) High-resolution protein design with backbone freedom. Science 282:1462–1467
19. Harbury P, Tidor B, Alber T, Kim P (1995) Repacking protein cores with backbone freedom: Structure prediction for coiled coils. Proc Natl Acad Sci USA 92:8408–8412
20. Klepeis JL, Floudas CA (1999) Free energy calculations for peptides via deterministic global optimization. J Chem Phys 110:7491–7512
21. Klepeis JL, Floudas CA (2002) Ab initio prediction of helical segments in polypeptides. J Comput Chem 23:245–266
22. Klepeis J, Floudas CA (2003) Astro-fold: A combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. Biophys J 85:2119–2146
23. Klepeis JL, Floudas CA (2003) Prediction of β-sheet topology and disulfide bridges in polypeptides. J Comput Chem 24:191–208
24. Klepeis JL, Floudas CA (2003) Ab initio tertiary structure prediction of proteins. J Global Optim 25:113–140
25. Klepeis JL, Floudas CA, Morikis D, Lambris J (1999) Predicting peptide structures using nmr data and deterministic global optimization. J Comput Chem 20:1354–1370
26. Klepeis JL, Floudas CA, Morikis D, Tsokos CG, Argyropoulos E, Spruce L, Lambris JD (2003) Integrated structural, computational and experimental approach for lead optimization: Deisgn of compstatin variants with improved activity. J Am Chem Soc 125:8422–8423
27. Klepeis JL, Floudas CA, Morikis D, Tsokos CG, Lambris JD (2004) Design of peptide analogs with improved activity using a novel de novo protein design approach. Ind Eng Chem Res 43:3817–3826
28. Klepeis J, Wei Y, Hecht M, Floudas C (2005) Ab initio prediction of the three-dimensional structure of a de novo designed protein: A double-blind case study. Proteins 58:560–570
29. Kono H, Saven J (2001) Statistical theory of protein combinatorial libraries: Packing interactions, backbone flexibility, and the sequence variability of a main-chain structure. J Mol Biol 306:607–628
30. Kraemer-Pecore C, Lecomte J, Desjarlais J (2003) A de novo redesign of the ww domain. Protein Sci 12:2194–2205
31. Kuhlman B, Dantae G, Ireton G, Verani G, Stoddard B, Baker D (2003) Design of a novel globular protein fold with atomic-level accuracy. Science 302:1364–1368
32. Larson SM, England JL, Desjarlais JR, Pande VS (2002) Thoroughly sampling sequence space: Large-scale protein design of structural ensembles. Protein Sci 11:2804–2813
33. Larson SM, Garg A, Desjarlais JR, Pande VS (2003) Increased detection of structural templates using alignments of designed sequences. Proteins 51:390–396
34. Lim W, Hodel A, Sauer R, Richards F (1994) The crystal structure of a mutant protein with altered but improved hydrophobic core packing. Proc Natl Acad Sci USA 91:423–427
35. Loose C, Klepeis J, Floudas C (2004) A new pairwise folding potential based on improved decoy generation and side chain packing. Proteins: Struct Funct Bioinformatics 54:303–314
36. McAllister S, Mickus B, Klepeis J, Floudas C (2006) Novel approach for alpha-helical topology prediction in globular proteins: Generation of interhelical restraints. Proteins 65:930–952
37. Mooers B, Datta D, Baase W, Zollars E, Mayo S, Matthews B (2003) Repacking the core of t4 lysozyme by automated design. J Mol Biol 332:741–756
38. Morikis D, Floudas CA, Lambris J (2005) Structure-based integrative computational and experimental approach for the optimization of drug design. Lect Notes Comput Sci 3515:680–688
39. Morikis D, Soulika A, Mallik B, Klepeis J, Floudas C, Lambris J (2004) Improvement of the anti-c3 activity of compstatin using rational and combinatorial approaches. Biochem Soc Trans 32:28–32
40. Plecs J, Harbury PB, Kim P, Alber T (2004) Structural test of the parameterized-backbone method for protein design. J Mol Biol 342:289–297
41. Pokala N, Handel T (2001) Review: Protein design-where we were, where we are, where we're going. J Struct Biol 134:269–281
42. Ponder J (1998) TINKER, software tools for molecular design. Department of Biochemistry and Molecular Biophysics, Washington University School of Medicine: St. Louis
43. Raha K, Wollacott A, Italia M, Desjarlais J (2000) Prediction of amino acid sequence from structure. Protein Sci 9:1106–1119
44. Rajgaria R, McAllister SR, Floudas CA (2006) A novel high resolution c^{α}-c^{α} distance dependent force field based on a high quality decoy set. Proteins: Struct, Funct, Bioinformatics 65:726–741

45. Rajgaria R, McAllister SR, Floudas CA (2007) Improving the performance of a high resolution distance dependent force field by including protein side chains. Proteins: Struct Funct Bioinformatics (in print)
46. Ross S, Sarisky C, Su A, Mayo S (2001) Designed protein g core variants fold to native-like structures: Sequence selection by orbit tolerates variation in backbone specification. Protein Sci 10:450–454
47. Saunders CT, Baker D (2005) Recapitulation of protein family divergence using flexible backbone protein design. J Mol Biol 346:631–644
48. Su A, Mayo S (1997) Coupling backbone flexibility and amino acid sequence selection in protein design. Protein Sci 6:1701–1707

De Novo Protein Design Using Rigid Templates

HO KI FUNG, CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Introduction

Computational protein design efforts were first initiated with the premise that the three-dimensional coordinates of the design template or backbone were fixed. This simplification was first proposed in [39], and was appealing because it greatly reduced the combinatorial complexity of the search. Together with consideration of only a limited set of most frequently observed side-chain conformations called rotamers [29,40], the assumption enhanced the efficiency of the initial de novo design efforts, most of which focused on protein cores [5,16,32,41,42], in exploring search spaces. The reason why protein cores were selected instead of the boundary or surface regions was based on the thesis that protein folding is primarily driven by hydrophobic collapse, and thus a good core tends to provide a well-folded and stable structure for the de novo designed protein [10]. The scope of the de novo design encompassed intermediate and surface residues in subsequent years, and obviously the problem became more challenging. In this chapter, we outline the different deterministic and stochastic methods that search for sequences specific to the fixed rigid design template. It should be noted that they all discretize the side-chain conformational space into rotamers for tractability of the search problem. After the introduction of each method we also review examples of successes.

Sequence Search Methods

De novo design algorithms can be classified into two main categories, namely, deterministic and stochastic [8]. The two main methods that fall into the deterministic category are dead-end elimination (DEE) and self-consistent mean field (SCMF), whereas the two major stochastic type frameworks are Monte Carlo and genetic algorithms. Some methods search for low-energy sequences, whereas others assign probability to each of the 20 amino acids for each design position in a sequence in order to maximize the conformational entropy.

Deterministic Methods

The Dead-End Elimination Criteria DEE, which is arguably the most popular rotamer search algorithm, operates on the basis of the systematic elimination of rotamers that cannot be parts of the sequence with the lowest energy. The energy function in DEE is written in the form of the sum of an individual term (rotamer–template) and a pairwise term (rotamer–rotamer):

$$E = \sum_{i=1}^{N} E(i_a) + \sum_{i=1}^{N-1} \sum_{j>i}^{N} E(i_a, j_b) , \qquad (1)$$

where $E(i_a)$ is the rotamer–template energy for rotamer i_a of amino acid i, $E(i_a, j_b)$ is the rotamer–rotamer energy of rotamer i_a and rotatmer j_b of amino acids i and j, respectively, and N is the total number of positions. The original DEE pruning criterion is based on the concept that if the pairwise energy between rotamer i_a and rotamer j_b is higher than that between rotamer i_c and rotamer j_b for all rotamer j_b in a certain rotamer set $\{B\}$, then rotamer i_a cannot be in the global energy minimum conformation and thus can be eliminated. It was proposed in [9] and can be expressed in the following mathematical form:

$$E(i_a) + \sum_{j\neq i}^{N} E(i_a, j_b) \; > \; E(i_c) + \sum_{j\neq i}^{N} E(i_c, j_b) \quad \forall\{B\}. \tag{2}$$

Rotamer i_a can be pruned if the above holds true. Bounds implied by (1) can be utilized to generate the following computationally more tractable inequality [9]:

$$E(i_a) \; + \; \sum_{j\neq i}^{N} \min_b E(i_a, j_b) \\ > \; E(i_c) \; + \; \sum_{j\neq i}^{N} \max_b E(i_c, j_b)\,. \tag{3}$$

The above equations for eliminating rotamers at a single position (or singles) can be extended to eliminating rotamer pairs at two distinct positions (doubles), rotamer triplets at three distinct positions (triples), or above [9,37]. In the case of doubles, the equation becomes

$$\varepsilon(i_a, j_b) \; + \; \sum_{k\neq i,j}^{N} \min_c \varepsilon(i_a, j_b, k_c) \\ > \; \varepsilon(i_{a'}, j_{b'}) \; + \; \sum_{k\neq i,j}^{N} \max_c \varepsilon(i_{a'}, j_{b'}, k_c)\,, \tag{4}$$

where ε is the total energy of rotamer pairs:

$$\varepsilon(i_a, j_b) \; = \; E(i_a) \; + \; E(j_b) \; + \; E(i_a, j_b)\,, \tag{5}$$

$$\varepsilon(i_a, j_b, k_c) \; = \; E(i_a, k_c) \; + \; E(j_b, k_c)\,. \tag{6}$$

It determines a rotamer pair i_a and j_b which always contributes higher energies than rotamer pair $i_{a'}$ and $j_{b'}$ for all possible rotamer combinations. Goldstein [14] improved the original DEE criterion by stating that rotamer i_a can be pruned if the energy contribution is always reduced by an alternative rotamer i_c:

$$E(i_a) - E(i_c) + \sum_{j\neq i}^{N} \min_b [E(i_a, j_b) - E(i_c, j_b)] \; > \; 0\,. \tag{7}$$

This can be generalized to the use of a weighted average of C rotamers i_c to eliminate i_a [14]:

$$E(i_a) - \sum_{c=1,\ldots,C} w_c E(i_c) \; + \; \sum_{j\neq i}^{N} \min_b [E(i_a, j_b) \\ - \sum_{c=1,\ldots,C} w_c E(i_c, j_b)] \; > \; 0\,. \tag{8}$$

Lasters et al. [25] proposed that the most suitable weights w_c can be determined by solving a linear programming problem.

In addition to these criteria proposed by Goldstein [14], Pierce et al. [38] introduced the split DEE, which splits the conformational space into partitions and thus eliminated the dead-ending rotamers more efficiently:

$$E(i_a) \; - \; E(i_c) \\ + \sum_{j,j\neq k\neq i}^{N} \{\min_{a'} [E(i_a, j_{a'}) - E(i_c, j_{a'})]\} \\ + \; [E(i_a, k_{b'}) - E(i_c, k_{b'})] \; > \; 0\,. \tag{9}$$

In general, n splitting positions can be assigned for more efficient but computationally expensive rotamer elimination:

$$E(i_a) \; - \; E(i_c) \\ + \sum_{j,j\neq k_1,\ldots,kn\neq i}^{N} \{\min_{a'} [E(i_a, j_{a'}) - E(i_c, j_{a'})]\} \\ + \sum_{k=k_1,\ldots,kn} [E(i_a, k_{b'}) - E(i_c, k_{b'})] \; > \; 0\,. \tag{10}$$

Looger and Hellinga [27] also introduced the generalized DEE by ranking the energy of rotamer clusters instead of that of individual rotamers and increased the ability of the algorithm to deal with higher levels of combinatorial complexity. Further revisions and improvements on DEE were performed by Wernisch et al. [47] and Gordon et al. [15].

Being deterministic in nature, the different forms of DEE reviewed above all yield the same globally optimal solution upon convergence.

Successes Using Dead-End Elimination: Based on operating the DEE algorithm on a fixed template, the Mayo group devised their optimization of rotamers by an iterative technique (ORBIT) program and applied it to numerous de novo protein designs. Examples are the full-sequence design of the $\beta\beta\alpha$ fold of a zinc finger domain [6], improvement of calmodulin binding affinity [45], full core design of the variable domains of the light and heavy chains of catalytic antibody 48G7 FAB, full core/boundary design, full surface design, and full-sequence design of the $\beta 1$ domain of protein G [15], as well as the redesign of the core of T4 lysozyme [32]. They also adjusted secondary structure parameters to build the "idealized backbone" and used it as a fixed template to design an α/β-barrel protein [33]. The Hellinga group applied DEE with a fixed backbone structure to introduce iron and oxygen binding sites into thioredoxin [2,3], design receptor and sensor proteins with novel ligand-binding functions [28], and confer novel enzymatic properties onto ribose-binding protein [11].

The Self-Consistent Mean-Field Method The SCMF optimization method is an iterative procedure that predicts the values of the elements of a conformational matrix $P(i,a)$ for the probability of a design position i adopting the conformation of rotamer a. Note that $P(i,a)$ sums to unity over all rotamers a for each position i. Koehl and Delarue [19] were among those who introduced such a method for protein design. They started the iteration with an initial guess for the conformational matrix, which assigns equal probability to all rotamers:

$$P(i,a) = \frac{1}{A} \quad a = 1,2,\ldots,A\,. \tag{11}$$

Most importantly, they applied the mean-field potential, $E(i,a)$, which depends on the conformational matrix $P(i,a)$:

$$E(i,a) = U(x_{ia}) + U(x_{ia}, x_0) + \sum_{j=1, j\neq i}^{N} \sum_{b=1}^{B} P(j,b)U(x_{i_a}, x_{j_b})\,, \tag{12}$$

where x_0 corresponds to the coordinates of atoms in the fixed template, and x_{i_a} and x_{j_b} correspond to the coordinates of the atoms of position i assuming the conformation of rotamer a and those of position j assuming the conformation of rotamer b, respectively. The classical Lennard-Jones (12-6) potential can be used to describe potential energy U [19]. The conformational matrix can be subsequently updated using the mean-field potential and the Boltzmann law:

$$P_1(i,a) = \frac{e^{\frac{-E(i,a)}{RT}}}{\sum_{a=1}^{A} e^{\frac{-E(i,a)}{RT}}}\,. \tag{13}$$

The update on $P(i,a)$, namely, $P_1(i,a)$, can then be used to repeat the calculation of the mean-field potential and another update until convergence is attained. Koehl and Delarue [19] set the convergence criterion to be 10^{-4} to define self-consistency. They also proposed the introduction of memory of the previous step to minimize oscillations during convergence:

$$P(i,a) = \lambda P_1(i,a) + (1-\lambda)P(i,a)\,, \tag{14}$$

with the optimal step size λ to be 0.9 [19].

The Saven group [12,24,44,48] extended the SCMF theory and formulated de novo design as an optimization problem maximizing the sequence entropy subject to composition constraints and mean-field energy constraints. In addition to the site probabilities, their method also predicts the number of sequences for a combinatorial library of arbitrary size for the fixed template as a function of energy.

It should be highlighted that though deterministic in nature, the SCMF method does not guarantee convergence to the global optimal solution [26].

Successes Using the Self-Consistent Mean-Field Method Koehl and Delarue [20] applied the SCMF approach to design protein loops. In their optimization procedure, they first selected the loop fragment from a database with the highest site probabilities. Then they placed side chains on the fixed loop backbone from a rotamer library. Kono and Doi [23] also used an energy minimization with an automata network, which bears some resemblance to the SCMF method, to design the cores of the globular proteins of cytochrome b_{562}, triosephosphate isomerase, and barnase. The SCMF method is related to the design of combinatorial libraries of new sequences with good folding properties, which was reviewed in several papers [17,34,35,43].

Stochastic Methods

The fact that de novo design is nondeterministic polynomial-time hard [13,36] means that in the worst case the time required to solve the problem scales non-polynomially with the number of design positions. As the problem complexity exceeds a certain level, deterministic methods may reach their limits and in such instances we may have to resort to stochastic methods, which perform searches for only locally optimal solutions. Monte Carlo methods and genetic algorithms are the two most commonly used types of stochastic methods for de novo protein design.

Monte Carlo Methods Different variants of the Monte Carlo methods have been applied for sequence design. In the classic Monte Carlo method, mutation is performed at a certain position in the sequence and energies of the sequence in the fixed template are calculated before and after the mutation. This usually involves the use of discrete rotamer libraries to simplify the consideration of possible side-chain conformations. The new sequence after mutation is accepted if the energy becomes lower. If the energy is higher, the Metropolis acceptance criterion [30] is used

$$p_{\text{accept}} = \min(1, \exp(-\beta \Delta E)) \quad \beta = \frac{1}{kT}\,, \tag{15}$$

and the sequence is updated if p_{accept} is larger than a random number uniformly distributed between 0 and 1.

In the configurational bias Monte Carlo method, at each step a local energy is used which does not include those positions where a mutation has not been attempted [49]. Cootes et al. [4] reported that the method was more efficient at finding good solutions than the conventional Monte Carlo method, especially for complex systems. Zoz and Savan [49] also devised the mean-field biased Monte Carlo method which biases the sequence search with predetermined site probabilities, which are in turn calculated using SCMF theory. They claimed their new method converges to low-energy sequences faster than classic Monte Carlo and configurational bias Monte Carlo methods.

Successes of Monte Carlo Methods Imposing sequence specificity by keeping the amino acid composition fixed, which reduced significantly the complexity, Koehl and Levitt [21,22] designed new sequences for the fixed backbones of the β1 domain of protein G, λ repressor, and sperm whale myoglobin using the conventional Monte Carlo method. The Baker group also utilized the classic Monte Carlo algorithm in their computational protein design program RosettaDesign. Examples of applications of the program include the redesign of nine globular proteins: the src SH3 domain, λ repressor, U1A, protein L, tenascin, procarboxypeptidase, acylphosphatase, S6, and FKBP12 using fixed templates [7].

Genetic Algorithms Originating in genetics and evolution, genetic algorithms generate a multitude of random amino acid sequences and exchange them for a fixed template. Sequences with low energies form hybrids with other sequences, while those with high energies are eliminated in an iterative process which only terminates when a converged solution is attained [46].

Successes of Genetic Algorithms With fixed backbones, Belda et al. [1] applied genetic algorithms to the design of ligands for prolyl oligopeptidase, p53, and DNA gyrase. In addition, with a cubic lattice and empiricial contact potentials Hohm et al. [18] and Miyazawa and Jernigan [31] also employed evolutionary methods to design short peptides that resemble the antibody epitopes of thrombin and blood coagulation factor VIII with high stability.

References

1. Belda I, Madurga S, Llorà X, Martinell M, Tarragó T, Piqueras MG, Nicolás E, Giralt E (2005) ENPDA: An evolutionary structure-based de novo peptide design algorithm. J Computer-Aided Mol Des 19:585–601
2. Benson D, Wisz M, Hellinga H (1998) The development of new biotechnologies using metalloprotein design. Curr Opin Biotechnol 9:370–376
3. Benson D, Wisz M, Hellinga H (2000) Rational design of nascent metalloenzymes. Proc Natl Acad Sci USA 97:6292–6297
4. Cootes AP, Curmi PMG, Torda AE (2000) Biased monte carlo optimization of protein sequences. J Chem Phys 113:2489–2496
5. Dahiyat B, Mayo S (1996) Protein design automation. Protein Sci 5:895–903
6. Dahiyat B, Mayo S (1997) De novo protein design: Fully automated sequence selection. Science 278:82–87
7. Dantas G, Kuhlman B, Callender D, Wong M, Baker D (2003) A large scale test of computational protein design: Folding

and stability of nine completely redesigned globular proteins. J Mol Biol 332:449–460

8. Desjarlais JR, Clarke ND (1998) Computer search algorithms in protein modification and design. Curr Opin Struct Biol 8:471–475
9. Desmet J, Maeyer MD, Hazes B, Lasters I (1992) The dead-end elimination theorem and its use in side-chain positioning. Nature 356:539–542
10. Dill K (1990) Dominant forces in protein folding. Biochemistry 29:7133–7155
11. Dwyer MA, Looger LL, Hellinga H (2004) Computational design of a biologically active enzyme. Science 304:1967–1971
12. Fu X, Kono H, Saven J (2003) Probabilistic approach to the design of symmetric protein quaternary structures. Protein Eng 16:971–977
13. Fung HK, Rao S, Floudas CA, Prokopyev O, Pardalos PM, Rendl F (2005) Computational comparison studies of quadratic assignment like formulations for the in silico sequence selection problem in de novo protein design. J Comb Optim 10:41–60
14. Goldstein R (1994) Efficient rotamer elimination applied to protein side-chains and related spin glasses. Biophys J 66:1335–1340
15. Gordon B, Hom G, Mayo S, Pierce N (2003) Exact rotamer optimization for protein design. J Comput Chem 24:232–243
16. Handel T, Desjarlais J (1995) De novo design of the hydrophobic cores of proteins. Protein Sci 4:2006–2018
17. Hecht M, Das A, Go A, Bradley L, Wei Y (2004) De novo proteins from designed combinatorial libraries. Protein Sci 13:1711–1723
18. Hohm T, Limbourg P, Hoffmann D (2006) A multiobjective evolutionary method for the design of peptidic mimotopes. J Comput Biol 13:113–125
19. Koehl P, Delarue M (1994) Application of a self-consistent mean field theory to predict protein side-chains conformation and estimate their conformational entropy. J Mol Biol 239:249–275
20. Koehl P, Delarue M (1995) A self consistent mean field approach to simultaneouos gap closure and side-chain positioning in homology modeling. Nat Struct Biol 2:163–170
21. Koehl P, Levitt M (1999) De novo protein design. i. in search of stability and specificity. J Mol Biol 293:1161–1181
22. Koehl P, Levitt M (1999) De novo protein design. ii. plasticity in sequence space. J Mol Biol 293:1183–1193
23. Kono H, Doi J (1994) Energy minimization method using automata network for sequence and side-chain conformation prediction from given backbone geometry. Proteins 19:244–255
24. Kono H, Saven J (2001) Statistical theory of protein combinatorial libraries: Packing interactions, backbone flexibility, and the sequence variability of a main-chain structure. J Mol Biol 306:607–628
25. Lasters I, Maeyer MD, Desmet J (1995) Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. Protein Eng 8:815–822
26. Lee C (1994) Predicting protein mutant energetics by self-consistent ensemble optimization. J Mol Biol 236:918–939
27. Looger L, Hellinga H (2001) Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: Implications for protein design and structural genomics. J Mol Biol 307:429–445
28. Looger L, Dwyer M, Smith J, Hellinga H (2003) Computational design of receptor and sensor proteins with novel functions. Nature 423:185–190
29. Lovell SC, Word JM, Richardson JS, Richardson DC (2000) The penultimate rotamer library. Proteins 40:389–408
30. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:389–408
31. Miyazawa S, Jernigan RL (1996) Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term for simulation and threading. J Mol Biol 256:623–644
32. Mooers B, Datta D, Baase W, Zollars E, Mayo S, Matthews B (2003) Repacking the core of t4 lysozyme by automated design. J Mol Biol 332:741–756
33. Offredi F, Dubail F, Kischel P, Sarinski K, Stern AS, de Weerdt CV, Hoch JC, Prosperi C, François JM, Mayo SL, Martial JA (2003) De novo backbone and sequence design of an idealized α/β-barrel protein: Evidence of stable tertiary structure. J Mol Biol 325:163–174
34. Park S, Stowell XF, Wang W, Yang X, Saven J (2004) Computational protein design and discovery. Annu Rep Prog Chem Sect C 100:195–236
35. Park S, Yang X, Saven J (2004) Advances in computational protein design. Curr Opin Struct Biol 14:487–494
36. Pierce N, Winfree E (2002) Protein design is np-hard. Protein Eng 15:779–782
37. Pierce N, Spriet J, Desmet J, Mayo S (2000) Conformational splitting: A more powerful criterion for dead-end elimination. J Comput Chem 21:999–1009
38. Pierce N, Spriet J, Desmet J, Mayo S (2000) Conformational splitting: A more powerful criterion for dead-end elimination. J Comput Chem 21:999–1009
39. Ponder J, Richards F (1987) Tertiary templates for proteins. J Mol Biol 193:775–791
40. Dunbrack L Jr, Cohen FE (1997) Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci 6:1661–81
41. Richards F, Hellinga H (1994) Optimal sequence selection in proteins of known structure by simulated evolution. Proc Natl Acad Sci USA 91:5803–5807
42. Rosenberg M, Goldblum A (2006) Computational protein design: A novel path to future protein drugs. Curr Pharm Des 12:3973–3997

43. Saven J (2002) Combinatorial protein design. Curr Opin Struct Biol 12:453–458
44. Saven J, Wolynes PG (1997) Statistical mechanics of the combinatorial synthesis and analysis of folding macromolecules. J Phys Chem B 101:8375–8389
45. Shifman J, Mayo S (2002) Modulating calmodulin binding specificity through computational protein design. J Mol Biol 323:417–423
46. Tuffery P, Etchebest C, Hazout S, Lavery R (1991) A new approach to the rapid determination of protein side chain conformations. J Biomol Struct Dyn 8:1267–1289
47. Wernisch L, Hery S, Wodak S (2000) Automatic protein design with all atom force-fields by exact and heuristic optimization. J Mol Biol 301:713–736
48. Zou J, Saven J (2000) Statistical theory of combinatorial libraries of folding proteins: Energetic discrimination of a target structure. J Mol Biol 296:281–294
49. Zou J, Saven J (2003) Using self-consistent fields to bias monte carlo methods with applications to designing and sampling protein sequences. J Chem Phys 118:3843–3854

Derivative-Free Methods for Non-smooth Optimization

ADIL BAGIROV
Centre for Informatics and Applied Optimization,
University of Ballarat,
Ballarat, Australia

MSC2000: 65K05, 90C56

Article Outline

Introduction

Consider the following unconstrained minimization problem:

$$\text{minimize}\, f(x) \text{ subject to}\, x \in \mathbb{R}^n\,, \tag{1}$$

where the objective function f is assumed to be Lipschitz continuous.

Nonsmooth unconstrained optimization problems appear in many applications and in particular in data mining. Over more than four decades different methods have been developed to solve problem (1). We mention among them the bundle method and its different variations (see, for example, [11,12,13,14,17,20]), algorithms based on smoothing techniques [18], and the gradient sampling algorithm [8].

In most of these algorithms at each iteration the computation of at least one subgradient or approximating gradient is required. However, there are many practical problems where the computation of even one subgradient is a difficult task. In such situations derivative-free methods seem to be a better choice since they do not use the explicit computation of subgradients.

Among derivative-free methods, the generalized pattern search methods are well suited for nonsmooth optimization [1,19]. However their convergence are proved under quite restrictive differentiability assumptions. It was shown in [19] that when the objective function f is continuously differentiable in $\mathbb{R}^n$, then the lower limit of the norm of the gradient of the sequence of points generated by the generalized pattern search algorithm goes to zero. The paper [1] provides convergence analysis under less restrictive differentiability assumptions. It was shown that if f is strictly differentiable near the limit of any refining subsequence, then the gradient at that point is zero. However, in many practically important problems this condition is not satisfied, because in such problems the objective functions are not differentiable at local minimizers.

In the paper [15] a derivative-free algorithm for a linearly constrained finite minimax problem was proposed. The original problem was converted into a smooth one using a smoothing technique. This algorithm is globally convergent toward stationary points of the finite minimax problem.

In this paper we describe a derivative-free method based on the notion of a discrete gradient for solving unconstrained nonsmooth optimization problems. Its convergence is proved for a broad class of nonsmooth functions.

Definitions

We use the following notation: $\mathbb{R}^n$ is an n-dimensional space, where the scalar product will be denoted by $\langle x, y\rangle$:

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$$

and $\|\cdot\|$ will denote the associated norm. The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}^1$ will be denoted by ∇f and the closed δ-ball at $x \in \mathbb{R}^n$ by $S_\delta(x)$ (by S_δ if $x = 0$): $S_\delta(x) = \{y \in \mathbb{R}^n : \|x - y\| \le \delta\}, \delta > 0$.

The Clarke Subdifferential

Let f be a function defined on $\mathbb{R}^n$. Function f is called locally Lipschitz continuous if for any bounded subset $X \subset \mathbb{R}^n$ there exists an $L > 0$ such that

$$|f(x) - f(y)| \le L\|x - y\| \forall x, y \in X.$$

We recall that a locally Lipschitz function f is differentiable almost everywhere and that we can define for it a Clarke subdifferential [9] by

$$\partial f(x) = \text{co}\Big\{v \in \mathbb{R}^n : \exists(x^k \in D(f), \\ x^k \to x, k \to +\infty) : v = \lim_{k\to+\infty} \nabla f(x^k)\Big\},$$

where $D(f)$ denotes the set where f is differentiable and co denotes the convex hull of a set. It is shown in [9] that the mapping $\partial f(x)$ is upper semicontinuous and bounded on bounded sets.

The generalized directional derivative of f at x in the direction g is defined as

$$f^0(x, g) = \limsup_{y\to x, \alpha\downarrow 0} \alpha^{-1}[f(y + \alpha g) - f(y)] .$$

If function f is locally Lipschitz continuous, then the generalized directional derivative exists and

$$f^0(x, g) = \max\{\langle v, g\rangle : v \in \partial f(x)\} .$$

f is called a Clarke regular function on $\mathbb{R}^n$ if it is differentiable with respect to any direction $g \in \mathbb{R}^n$ and $f'(x, g) = f^0(x, g)$ for all $x, g \in \mathbb{R}^n$, where $f'(x, g)$ is a derivative of function f at point x with respect to direction g:

$$f'(x, g) = \lim_{\alpha\downarrow 0} \alpha^{-1}[f(x + \alpha g) - f(x)].$$

It is clear that the directional derivative $f'(x, g)$ of the Clarke regular function f is upper semicontinuous with respect to x for all $g \in \mathbb{R}^n$.

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. For point x to be a minimum point of function f on $\mathbb{R}^n$, it is necessary that $0 \in \partial f(x)$.

Semismooth Functions

The function $f : \mathbb{R}^n \to \mathbb{R}^1$ is called semismooth at $x \in \mathbb{R}^n$, if it is locally Lipschitz continuous at x and for every $g \in \mathbb{R}^n$, the limit

$$\lim_{g'\to g, \alpha\downarrow 0, v\in\partial f(x+\alpha g')} \langle v, g\rangle$$

exists. It should be noted that the class of semismooth functions is fairly wide and it contains convex, concave, max- and min-type functions [16]. The semismooth function f is directionally differentiable and

$$f'(x, g) = \lim_{g'\to g, \alpha\downarrow 0, v\in\partial f(x+\alpha g')} \langle v, g\rangle .$$

Quasidifferentiable Functions

A function f is called quasidifferentiable at a point x if it is locally Lipschitz continuous and directionally differentiable at this point and there exist convex, compact sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ such that

$$f'(x, g) = \max_{u\in\underline{\partial} f(x)} \langle u, g\rangle + \min_{v\in\overline{\partial} f(x)} \langle v, g\rangle .$$

The set $\underline{\partial} f(x)$ is called a subdifferential, the set $\overline{\partial} f(x)$ is called a superdifferential, and the pair of sets $[\underline{\partial} f(x), \overline{\partial} f(x)]$ is called a quasidifferential of function f at a point x [10].

Methods

Approximation of Subgradients

We consider a locally Lipschitz continuous function f defined on $\mathbb{R}^n$ and assume that this function is quasidifferentiable. We also assume that both sets $\underline{\partial} f(x)$ and

$\overline{\partial} f(x)$ at any $x \in \mathbb{R}^n$ are polytopes, that is, at a point $x \in \mathbb{R}^n$ there exist sets

$$A = \{a^1, \dots, a^m\}, a^i \in \mathbb{R}^n, i = 1, \dots, m, m \geq 1$$

and

$$B = \{b^1, \dots, b^p\}, b^j \in \mathbb{R}^n, j = 1, \dots, p, p \geq 1$$

such that

$$\underline{\partial} f(x) = \operatorname{co} A, \overline{\partial} f(x) = \operatorname{co} B.$$

This assumption is true, for example, for functions represented as a maximum, minimum, or max-min of a finite number of smooth functions.

We take a direction $g \in \mathbb{R}^n$ such that

$$g = (g_1, \dots, g_n), |g_i| = 1, i = 1, \dots, n$$

and consider the sequence of n vectors $e^j = e^j(\alpha), j = 1, \dots, n$ with $\alpha \in (0, 1]$:

$$\begin{array}{rcl} e^1 & = & (\alpha g_1, 0, \dots, 0), \\ e^2 & = & (\alpha g_1, \alpha^2 g_2, 0, \dots, 0), \\ \dots & = & \dots\dots\dots \\ e^n & = & (\alpha g_1, \alpha^2 g_2, \dots, \alpha^n g_n). \end{array}$$

We introduce the following sets:

$$\underline{R}_0 = A, \overline{R}_0 = B,$$

$$\underline{R}_j = \left\{ v \in \underline{R}_{j-1} : v_j g_j = \max\{w_j g_j : w \in \underline{R}_{j-1}\} \right\},$$

$$\overline{R}_j = \left\{ v \in \overline{R}_{j-1} : v_j g_j = \min\{w_j g_j : w \in \overline{R}_{j-1}\} \right\}.$$

$$j = 1, \dots, n.$$

It is clear that

$$\underline{R}_j \neq \emptyset, \forall j \in \{0, \dots, n\}, \underline{R}_j \subseteq \underline{R}_{j-1}, \forall j \in \{1, \dots, n\}$$

and

$$\overline{R}_j \neq \emptyset, \forall j \in \{0, \dots, n\}, \overline{R}_j \subseteq \overline{R}_{j-1}, \forall j \in \{1, \dots, n\}.$$

Moreover,

$$v_r = w_r \ \forall v, w \in \underline{R}_j, r = 1, \dots, j \tag{2}$$

and

$$v_r = w_r \ \forall v, w \in \overline{R}_j, r = 1, \dots, j. \tag{3}$$

Consider the following two sets:

$$\underline{R}(x, e^j(\alpha)) = \left\{ v \in A : \langle v, e^j \rangle = \max_{u \in A} \langle u, e^j \rangle \right\},$$

$$\overline{R}(x, e^j(\alpha)) = \left\{ w \in B : \langle w, e^j \rangle = \min_{u \in B} \langle u, e^j \rangle \right\}.$$

Proposition 1 *Assume that function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$\underline{R}(x, e^j(\alpha)) \subset \underline{R}_j, \overline{R}(x, e^j(\alpha)) \subset \overline{R}_j, j = 1, \dots, n$$

for all $\alpha \in (0, \alpha_0)$.

Corollary 1 *Assume that function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$\begin{aligned} f'(x, e^j(\alpha)) = & f'(x, e^{j-1}(\alpha)) + v_j \alpha^j g_j + w_j \alpha^j g_j, \\ & \forall v \in \underline{R}_j, \ w \in \overline{R}_j, \ j = 1, \dots, n \end{aligned}$$

for all $\alpha \in (0, \alpha_0]$

Proposition 2 *Assume that function f is quasidifferentiable and its subdifferential and superdifferential are polytopes at a point x. Then the sets $\underline{R}_n$ and $\overline{R}_n$ are singletons.*

Remark 1 In the next subsection we propose an algorithm to approximate subgradients. This algorithm finds a subgradient that can be represented as a sum of elements of the sets $\underline{R}_n$ and $\overline{R}_n$.

Computation of Subgradients

Let $g \in \mathbb{R}^n, |g_i| = 1, i = 1, \dots, n$ be a given vector and $\lambda > 0, \alpha > 0$ be given numbers. We define the following points:

$$x^0 = x, \ x^j = x^0 + \lambda e^j(\alpha), \ j = 1, \dots, n.$$

It is clear that

$$x^j = x^{j-1} + (0, \dots, 0, \lambda \alpha^j g_j, 0, \dots, 0), \ j = 1, \dots, n.$$

Let $v = v(\alpha, \lambda) \in \mathbb{R}^n$ be a vector with the following coordinates:

$$v_j = (\lambda \alpha^j g_j)^{-1} \left[f(x^j) - f(x^{j-1}) \right], j = 1, \dots, n. \tag{4}$$

For any fixed $g \in \mathbb{R}^n, |g_i| = 1, i = 1, \dots, n$ and $\alpha > 0$ we introduce the following set:

$$V(g,\alpha) = \Big\{ w \in \mathbb{R}^n : \exists(\lambda_k \to +0, k \to +\infty), \quad w = \lim_{k\to+\infty} v(\alpha, \lambda_k) \Big\}.$$

Proposition 3 *Assume that f is a quasidifferentiable function and its subdifferential and superdifferential are polytopes at x. Then there exists $\alpha_0 > 0$ such that*

$$V(g,\alpha) \subset \partial f(x)$$

for all $\alpha \in (0, \alpha_0]$.

Remark 2 It follows from Proposition 3 that in order to approximate subgradients of quasidifferentiable functions one can choose a vector $g \in \mathbb{R}^n$ such that $|g_i| = 1, i = 1, \dots, n$, sufficiently small $\alpha > 0, \lambda > 0$, and apply (4) to compute a vector $v(\alpha, \lambda)$. This vector is an approximation to a certain subgradient.

Computation of Subdifferentials and Discrete Gradients

In the previous subsection we demonstrated an algorithm for the computation of subgradients. In this subsection we consider an algorithm for the computation of subdifferentials. This algorithm is based on the notion of a discrete gradient. We start with the definition of the discrete gradient, which was introduced in [2] (for more details, see also [3,4]).

Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$. Let

$$S_1 = \{g \in \mathbb{R}^n : \|g\| = 1\}, G = \{e \in \mathbb{R}^n : e = (e_1, \dots, e_n), |e_j| = 1, j = 1, \dots, n\},$$

$$P = \{z(\lambda) : z(\lambda) \in \mathbb{R}^1, z(\lambda) > 0, \lambda > 0, \lambda^{-1} z(\lambda) \to 0, \lambda \to 0\}.$$

Here S_1 is the unit sphere, G is the set of vertices of the unit hypercube in $\mathbb{R}^n$, and P is the set of univariate positive infinitesimal functions.

We take any $g \in S_1$ and define $|g_i| = \max\{|g_k|, k = 1, \dots, n\}$. We also take any $e = (e_1, \dots, e_n) \in G$, a positive number $\alpha \in (0, 1]$, and define the sequence of n vectors $e^j(\alpha), j = 1, \dots, n$ as in Sect. "Approximation of Subgradients." Then for given $x \in \mathbb{R}^n$ and $z \in P$ we define a sequence of $n + 1$ points as follows:

$$\begin{aligned} x^0 &= x + \lambda g, \\ x^1 &= x^0 + z(\lambda) e^1(\alpha), \\ x^2 &= x^0 + z(\lambda) e^2(\alpha), \\ \dots &= \dots \quad \dots \\ x^n &= x^0 + z(\lambda) e^n(\alpha). \end{aligned}$$

Definition 1 The discrete gradient of function f at point $x \in \mathbb{R}^n$ is the vector $\Gamma^i(x, g, e, z, \lambda, \alpha) = (\Gamma_1^i, \dots, \Gamma_n^i) \in \mathbb{R}^n, g \in S_1$ with the following coordinates:

$$\Gamma_j^i = [z(\lambda)\alpha^j e_j)]^{-1} \left[f(x^j) - f(x^{j-1}) \right], \quad j = 1, \dots, n, j \neq i,$$

$$\Gamma_i^i = (\lambda g_i)^{-1} \left[f(x + \lambda g) - f(x) - \lambda \sum_{j=1, j\neq i}^{n} \Gamma_j^i g_j \right].$$

It follows from the definition that

$$f(x + \lambda g) - f(x) = \lambda \langle \Gamma^i(x, g, e, z, \lambda, \alpha), g \rangle \qquad (5)$$

for all $g \in S_1, e \in G, z \in P, \lambda > 0, \alpha > 0$.

Remark 3 One can see that the discrete gradient is defined with respect to a given direction $g \in S_1$, and in order to compute the discrete gradient $\Gamma^i(x, g, e, z, \lambda, \alpha)$, first we define a sequence of points $x^0, \dots, x^n$ and compute the values of function f at these points; that is, we compute $n + 2$ values of this function including point x. $n - 1$ coordinates of the discrete gradient are defined similarly to those of the vector $v(\alpha, \lambda)$ from the Sect. "Approximation of Subgradients," and the ith coordinate is defined so as to satisfy equality (5), which can be considered as as version of the mean value theorem.

Proposition 4 *Let f be a locally Lipschitz continuous function defined on $\mathbb{R}^n$ and $L > 0$ its Lipschitz constant. Then for any $x \in \mathbb{R}^n, g \in S_1, e \in G, \lambda > 0, z \in P, \alpha > 0$*

$$\|\Gamma^i\| \le C(n)L, C(n) = (n^2 + 2n^{3/2} - 2n^{1/2})^{1/2}.$$

For a given $\alpha > 0$ we define the following set:

$$\begin{aligned} B(x,\alpha) = & \{v \in \mathbb{R}^n : \exists(g \in S_1, e \in G, z_k \in P, \\ & z_k \to +0, \lambda_k \to +0, k \to +\infty), \\ & v = \lim_{k\to+\infty} \Gamma^i(x,g,e,z_k,\lambda_k,\alpha)\}. \end{aligned} \quad (6)$$

Proposition 5 *Assume that f is a semismooth, quasidifferentiable function and its subdifferential and superdifferential are polytopes at a point x. Then there exists $\alpha_0 > 0$ such that*

$$co\, B(x,\alpha) \subset \partial f(x)$$

for all $\alpha \in (0, \alpha_0]$.

Remark 4 Proposition 5 implies that discrete gradients can be applied to approximate subdifferentials of a broad class of semismooth, quasidifferentiable functions.

Remark 5 One can see that the discrete gradient contains three parameters: $\lambda > 0$, $z \in P$, and $\alpha > 0$. $z \in P$ is used to exploit the semismoothness of function f, and it can be chosen sufficiently small. If f is a semismooth quasidifferentiable function and its subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$, then for any $\delta > 0$ there exists $\alpha_0 > 0$ such that $\alpha \in (0, \alpha_0]$ for all $y \in S_\delta(x)$. The most important parameter is $\lambda > 0$. In the sequel we assume that $z \in P$ and $\alpha > 0$ are sufficiently small.

Consider the following set:

$$\begin{aligned} D_0(x,\lambda) = & \operatorname{cl} \operatorname{co} \{v \in \mathbb{R}^n : \exists(g \in S_1, e \in G, z \in P) : \\ & v = \Gamma^i(x,g,e,\lambda,z,\alpha)\}. \end{aligned}$$

Proposition 4 implies that the set $D_0(x,\lambda)$ is compact and it is also convex for any $x \in \mathbb{R}^n$.

Corollary 2 *Let f be a quasidifferentiable semismooth function. Assume that in the equality*

$$f(x+\lambda g) - f(x) = \lambda f'(x,g) + o(\lambda,g), g \in S_1$$

$\lambda^{-1}o(\lambda,g) \to 0$ as $\lambda \to +0$ uniformly with respect to $g \in S_1$. Then for any $\varepsilon > 0$ there exists $\lambda_0 > 0$ such that

$$D_0(x,\lambda) \subset \partial f(x) + S_\varepsilon$$

for all $\lambda \in (0, \lambda_0)$.

Corollary 2 shows that the set $D_0(x,\lambda)$ is an approximation to the subdifferential $\partial f(x)$ for sufficiently small $\lambda > 0$. However, it is true at a given point. To get convergence results for a minimization algorithm based on discrete gradients, we need some relationship between the set $D_0(x,\lambda)$ and $\partial f(x)$ in some neighborhood of a given point x. We will consider functions satisfying the following assumption.

Assumption 1 *Let $x \in \mathbb{R}^n$ be a given point. For any $\varepsilon > 0$ there exist $\delta > 0$ and $\lambda_0 > 0$ such that*

$$D_0(y,\lambda) \subset \partial f(x + \bar{S}_\varepsilon) + S_\varepsilon \quad (7)$$

for all $y \in S_\delta(x)$ and $\lambda \in (0, \lambda_0)$. Here

$$\begin{aligned} \partial f(x+\bar{S}_\varepsilon) = & \bigcup_{y \in \bar{S}_\varepsilon(x)} \partial f(y), \bar{S}_\varepsilon(x) \\ & = \{y \in \mathbb{R}^n : \|x-y\| \le \varepsilon\}. \end{aligned}$$

A Necessary Condition for a Minimum

Consider problem (1), where $f : \mathbb{R}^n \to \mathbb{R}^1$ is an arbitrary function.

Proposition 6 *Let $x^* \in \mathbb{R}^n$ be a local minimizer of function f. Then there exists $\lambda_0 > 0$ such that*

$$0 \in D_0(x,\lambda)$$

for all $\lambda \in (0, \lambda_0)$.

Proposition 7 *Let $0 \notin D_0(x,\lambda)$ for a given $\lambda > 0$ and $v^0 \in \mathbb{R}^n$ be a solution to the following problem:*

$$\textit{minimize} \|v\|^2 \textit{subject to } v \in D_0(x,\lambda).$$

Then the direction $g^0 = -\|v^0\|^{-1}v^0$ is a descent direction.

Proposition 7 shows how the set $D_0(x,\lambda)$ can be used to compute descent directions. However, in many cases the computation of the set $D_0(x,\lambda)$ is not possible. In the next section we propose an algorithm for the computation of descent directions using a few discrete gradients from $D_0(x,\lambda)$.

Computation of Descent Directions

In this subsection we describe an algorithm for the computation of descent directions of the objective function f of Problem (1).

Let $z \in P, \lambda > 0, \alpha \in (0, 1]$, the number $c \in (0, 1)$, and a tolerance $\delta > 0$ be given.

Algorithm 1 An algorithm for the computation of the descent direction.

Step 1. Choose any $g^1 \in S_1, e \in G$; compute $i = \operatorname{argmax}\{|g_j|, j = 1, \dots, n\}$ and a discrete gradient $v^1 = \Gamma^i(x, g^1, e, z, \lambda, \alpha)$. Set $\overline{D}_1(x) = \{v^1\}$ and $k = 1$.

Step 2. Compute the vector $\|w^k\|^2 = \min\{\|w\|^2 : w \in \overline{D}_k(x)\}$. If

$$\|w^k\| \le \delta, \tag{8}$$

then stop. Otherwise go to Step 3.

Step 3. Compute the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4. If

$$f(x + \lambda g^{k+1}) - f(x) \le -c\lambda\|w^k\|, \tag{9}$$

then stop. Otherwise go to Step 5.

Step 5. Compute $i = \operatorname{argmax}\{|g_j^{k+1}| : j = 1, \dots, n\}$ and a discrete gradient

$$v^{k+1} = \Gamma^i(x, g^{k+1}, e, z, \lambda, \alpha),$$

construct the set $\overline{D}_{k+1}(x) = \operatorname{co}\{\overline{D}_k(x) \bigcup \{v^{k+1}\}\}$, set $k = k + 1$, and go to Step 2.

In what follows we provide some explanations of Algorithm 1. In Step 1 we compute the discrete gradient with respect to an initial direction $g^1 \in \mathbb{R}^n$. The distance between the convex hull $\overline{D}_k(x)$ of all computed discrete gradients and the origin is computed in Step 2. This problem is solved using the algorithm from [21]. If this distance is less than the tolerance $\delta > 0$, then we accept point x as an approximate stationary point (Step 2); otherwise we compute another search direction in Step 3. In Step 4 we check whether this direction is a descent direction. If it is, we stop and the descent direction has been computed; otherwise we compute another discrete gradient with respect to this direction in Step 5 and update the set $\overline{D}_k(x)$. At each iteration k we improve the approximation of the subdifferential of function f.

The next proposition shows that Algorithm 1 is terminating.

Proposition 8 *Let f be a locally Lipschitz function defined on $\mathbb{R}^n$. Then, for $\delta \in (0, \bar{C})$, either condition (8) or condition (9) satisfies after m computations of the discrete gradients, where*

$$m \le 2(\log_2(\delta/\bar{C})/\log_2 r + 1), r = 1 - [(1-c)(2\bar{C})^{-1}\delta]^2,$$

$\bar{C} = C(n)L$, and $C(n)$ is a constant from Proposition 4.

Remark 6 Proposition 4 and equality (5) are true for any $\lambda > 0$ and for any locally Lipschitz continuous functions. This means that Algorithm 1 can compute descent directions for any $\lambda > 0$ and for any locally Lipschitz continuous functions in a finite number of iterations. Sufficiently small values of λ give an approximation to the subdifferential, and in this case Algorithm 1 computes local descent directions. However, larger values of λ do not give an approximation to the subdifferential and in this case descent directions computed by Algorithm 1 can be considered global descent directions.

The Discrete Gradient Method

In this section we describe the discrete gradient method. Let sequences $\delta_k > 0, z_k \in P, \lambda_k > 0, \delta_k \to +0, z_k \to +0, \lambda_k \to +0, k \to +\infty$, sufficiently small number $\alpha > 0$, and numbers $c_1 \in (0, 1), c_2 \in (0, c_1]$ be given.

Algorithm 2 Discrete gradient method

Step 1. Choose any starting point $x^0 \in \mathbb{R}^n$ and set $k = 0$.

Step 2. Set $s = 0$ and $x_s^k = x^k$.

Step 3. Apply Algorithm 1 for the computation of the descent direction at $x = x_s^k, \delta = \delta_k, z = z_k, \lambda = \lambda_k, c = c_1$. This algorithm terminates after a finite number of iterations $l > 0$. As a result we get the set $\overline{D}_l(x_s^k)$ and an element v_s^k such that

$$\|v_s^k\|^2 = \min\{\|v\|^2 : v \in \overline{D}_l(x_s^k)\}.$$

Furthermore, either $\|v_s^k\| \le \delta_k$ or for the search direction $g_s^k = -\|v_s^k\|^{-1} v_s^k$

$$f(x_s^k + \lambda_k g_s^k) - f(x_s^k) \le -c_1\lambda_k\|v_s^k\|. \tag{10}$$

Step 4. If

$$\|v_s^k\| \le \delta_k, \tag{11}$$

then set $x^{k+1} = x_s^k, k = k + 1$ and go to Step 2. Otherwise go to Step 5.

Step 5. Construct the following iteration $x^k_{s+1} = x^k_s + \sigma_s g^k_s$, where σ_s is defined as follows:

$$\sigma_s = \operatorname{argmax} \Big\{\sigma \geq 0 : f(x^k_s + \sigma g^k_s) - f(x^k_s) \leq -c_2\sigma\|v^k_s\|\Big\}.$$

Step 6. Set $s = s + 1$ and go to Step 3.

For the point $x^0 \in \mathbb{R}^n$ we consider the set $M(x^0) = \big\{x \in \mathbb{R}^n : f(x) \leq f(x^0)\big\}$.

Proposition 9 *Assume that function f is semismooth quasidifferentiable, its subdifferential and superdifferential are polytopes at any $x \in \mathbb{R}^n$, Assumption 1 is satisfied, and the set $M(x^0)$ is bounded for starting points $x^0 \in \mathbb{R}^n$. Then every accumulation point of $\{x^k\}$ belongs to the set $X^0 = \{x \in \mathbb{R}^n : 0 \in \partial f(x)\}$.*

Remark 7 Since Algorithm 1 can compute descent directions for any values of $\lambda > 0$, we take $\lambda_0 \in (0, 1)$, some $\beta \in (0, 1)$, and update λ_k, $k \geq 1$ as follows:

$$\lambda_k = \beta^k\lambda_0, k \geq 1.$$

Thus in the discrete gradient method we use approximations to subgradients only at the final stage of the method, which guarantees convergence. In most iterations we do not use explicit approximations of subgradients. Therefore it is a derivative-free method.

Remark 8 It follows from (10) and $c_2 \leq c_1$ that always $\sigma_s \geq \lambda_k$ and therefore $\lambda_k > 0$ is a lower bound for σ_s. This leads to the following rule for the computation of σ_s. We define a sequence:

$$\theta_m = m\lambda_k, \ \ m \geq 1,$$

and σ_s is defined as the largest θ_m satisfying the inequality in Step 5.

Applications

There are many problems from applications where the objective and/or constraint functions are not regular. We will consider one of them, the cluster analysis problem, which is an important application area in data mining.

Clustering is also known as the unsupervised classification of patterns; it deals with problems of organizing a collection of patterns into clusters based on similarity. Clustering has many applications in information retrieval, medicine, etc.

In cluster analysis we assume that we have been given a finite set C of points in the n-dimensional space $\mathbb{R}^n$, that is,

$$C = \{c^1, \ldots, c^m\}, \text{ where } c^i \in \mathbb{R}^n, \ i = 1, \ldots, m.$$

We consider here partition clustering, that is, the distribution of the points of set C into a given number q of disjoint subsets C^i, $i = 1, \ldots, q$ with respect to predefined criteria such that:

(1) $C^i \neq \emptyset, i = 1, \ldots, q$;
(2) $C^i \bigcap C^j = \emptyset, i, j = 1, \ldots, q, i \neq j$;
(3) $C = \bigcup_{i=1}^{q} C^i$.

The sets $C^i, i = 1, \ldots, q$ are called clusters. The strict application of these rules is called *hard clustering*, unlike *fuzzy clustering*, where the clusters are allowed to overlap. We assume that no constraints are imposed on the clusters C^i, $i = 1, \ldots, q$, that is, we consider the hard unconstrained clustering problem.

We also assume that each cluster $C^i, i = 1, \ldots, q$ can be identified by its center (or centroid). There are different formulations of clustering as an optimization problem. In [5,6,7] the cluster analysis problem is reduced to the following nonsmooth optimization problem:

$$\begin{aligned} &\text{minimize} f(x^1, \ldots, x^q) \\ &\text{subject to } (x^1, \ldots, x^q) \in \mathbb{R}^{n\times q}, \end{aligned} \tag{12}$$

where

$$f(x^1, \ldots, x^q) = \frac{1}{m}\sum_{i=1}^{m} \min_{s=1,\ldots,q} \|x^s - c^i\|^2. \tag{13}$$

Here $\|\cdot\|$ is the Euclidean norm and $x^s \in \mathbb{R}^n$ stands for the sth cluster center. If $q > 1$, then the objective function (13) in problem (12) is nonconvex and nonsmooth. Moreover, function f is a nonregular function, and the computation of even one subgradient of this function is quite a difficult task. This function can be represented as the difference of two convex functions as follows:

$$f(x) = f_1(x) - f_2(x),$$

where

$$f_1(x) = \frac{1}{m}\sum_{i=1}^{m}\sum_{s=1}^{q} \|x^s - c^i\|^2,$$

$$f_2(x) = \frac{1}{m} \sum_{i=1}^{m} \max_{s=1,\ldots,q} \sum_{k=1,k\neq s}^{q} \|x^k - c^i\|^2.$$

It is clear that function f is quasidifferentiable and its subdifferential and are polytopes at any point.

Thus, the discrete gradient method can be applied to solve clustering problem.

Conclusions

We have discussed a derivative-free discrete gradient method for solving unconstrained nonsmooth optimization problems. This algorithm can be applied to a broad class of optimization problems including problems with nonregular objective functions. It is globally convergent toward stationary points of semismooth, quasidifferentiable functions whose subdifferential and superdifferential are polytopes.

References

1. Audet C, Dennis JE Jr (2003) Analysis of generalized pattern searches. SIAM J Optim 13:889–903
2. Bagirov AM, Gasanov AA (1995) A method of approximating a quasidifferential. J Comput Math Math Phys 35(4):403–409
3. Bagirov AM (1999) Minimization methods for one class of nonsmooth functions and calculation of semi-equilibrium prices. In: Eberhard A et al (eds) Progress in Optimization: Contributions from Australasia. Kluwer, Dordrecht, pp 147–175
4. Bagirov AM (2003) Continuous subdifferential approximations and their applications. J Math Sci 115(5):2567–2609
5. Bagirov AM, Rubinov AM, Soukhoroukova AV, Yearwood J (2003) Supervised and unsupervised data classification via nonsmooth and global optimisation. TOP: Span Oper Res J 11(1):1–93
6. Bagirov AM, Ugon J (2005) An algorithm for minimizing clustering functions. Optim 54(4–5):351–368
7. Bagirov AM, Yearwood J (2006) A new nonsmooth optimisation algorithm for minimum sum-of-squares clustering problems. Eur J Oper Res 170(2):578–596
8. Burke JV, Lewis AS, Overton ML (2005) A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J Optim 15(3):751–779
9. Clarke FH (1983) Optimization and Nonsmooth Analysis. Wiley, New York
10. Demyanov VF, Rubinov AM (1995) Constructive Nonsmooth Analysis. Lang, Frankfurt am Main
11. Hiriart-Urruty JB, Lemarechal C (1993) Convex Analysis and Minimization Algorithms, vols 1 and 2. Springer, Berlin
12. Kiwiel KC (1985) Methods of Descent for Nondifferentiable Optimization. In: Lecture Notes in Mathematics. Springer, Berlin
13. Lemarechal C (1975) An extension of Davidon methods to nondifferentiable problems. In: Balinski ML, Wolfe P (eds) Nondifferentiable Optimization. Mathematical Programming Study, vol 3, North-Holland, Amsterdam, pp 95–109
14. Lemarechal C, Zowe J (1994) A condensed introduction to bundle methods in nonsmooth optimization. In: Spedicato E (ed) Algorithms for Continuous Optimization. Kluwer, Dordrecht, pp 357–482
15. Liuzzi G, Lucidi S, Sciandrone M (2006) A derivative free algorithm for linearly constrained finite minimax problems. SIAM J Optim 16(4):1054–1075
16. Mifflin R (1977) Semismooth and semiconvex functions in constrained optimization. SIAM J Control Optim 15(6):959–972
17. Mifflin R (1977) An algorithm for constrained optimization with semismooth functions. Math Oper Res 2:191–207
18. Polak E, Royset JO (2003) Algorithms for finite and semi-infinite min-max-min problems using adaptive smoothing techniques. J Optim Theory Appl 119(3):421–457
19. Torczon V (1997) On the convergence of pattern search algorithms. SIAM J Optim 7:1–25
20. Wolfe PH (1975) A method of conjugate subgradients of minimizing nondifferentiable convex functions. Math Program Stud 3:145–173
21. Wolfe PH (1976) Finding the nearest point in a polytope. Math Program 11(2):128–149

Derivatives of Markov Processes and Their Simulation

GEORG PFLUG
University Vienna, Vienna, Austria

MSC2000: 90C15, 60J05

Article Outline

Keywords

Derivatives; Stochastic optimization

Introduction

The optimal design of stochastic systems like queueing or inventory systems is a specific stochastic optimization problem.

Let $Y_x(t)$ be an ergodic Markov process with discrete time $t = 1, 2, \ldots$ and values in $\mathbf{R}^m$, depending on a control parameter $x \in \mathbf{R}^d$. Let $H(x, \cdot)$ be some cost function. The problem is to find the control x which minimizes the expectedcosts of the system either under the transient or under the stationary regime:

1) Under the *transient regime*, the process is started at time 0 in a specific starting state y_0 and observed until time T. The optimality problem reads

$$\begin{cases} \min & F(x) = \sum_{t=1}^{T} \mathsf{E}[H(x, Y_x(t)] \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^d . \end{cases} \tag{1}$$

2) Under the *stationary regime* it is assumed that $Y_x(\infty)$ is distributed according to the stationary distribution of the process, which is — by ergodicity — the asymptotic distribution of $Y_x(t)$ as t tends to infinity. The optimality problem reads

$$\begin{cases} \min & F(x) = \mathsf{E}[H(x, Y_x(\infty)] \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^d . \end{cases} \tag{2}$$

As an example, consider the problem of optimally determining the decision limits (x_1, x_2) in an inventory policy (if the inventory at hand has fallen below x_1 order the amount needed to bring the inventory up to x_2). Assuming that the sales are of random size, the inventory system can be modeled as a Markov process depending on control parameters (x_1 and x_2).

The solution method for such an optimization problem is a version of the stochastic quasigradient method (cf. also ► Stochastic quasigradient methods). The solution is stepwise improved by moving it into the direction of an estimate of the negative gradientof the objective function.

The basic problem is therefore to find good estimates for the gradient $\nabla_x F(x)$. This problem is a generalization of the problem of finding derivatives of probability measures (cf. ► Derivatives of probability measures). The general notions of *distributional derivatives* (direct differentiability) and *process derivatives* (inverse differentiability) are applicable here.

Process Derivatives

Suppose that the process $Y_x(\cdot)$ has a representation of the form

$$Y_x(t+1) = K_t(x, Y_x(t), \xi_t),$$

where ξ_t is a sequence of random variables, the distribution of which does not depend on x. If the derivatives $\nabla_x K_t(x, y, \xi)$, $\nabla_y K_t(x, y, \xi)$ and $\nabla_y H(y)$ exist, we get by elementary calculus

$$\nabla_x H(Y_x(t)) = \nabla_y H(Y_x(t)) \times \left[\sum_{i=0}^{t-1} \left(\prod_{j=i+1}^{t-1} \nabla_y K_j(x, Y_x(j), \xi_j) \right) \nabla_x K_i(x, z_x(i), \xi_i) \right], \tag{3}$$

where the order of multiplication in the product is here and in the following from left to right by *descending index*. Formula (3) may be computed recursively, as follows.

Define the $[m \times d]$ (random) matrices N_t by

$$N_t = \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t-1} \nabla_y K_j(x, Y_x(j), \xi_j) \right) \times \nabla_x K_i(x, Y_x(i), \xi_i).$$

This sequence follows the forward recursion

$$N_0 = 0,$$
$$N_{t+1} = \nabla_x K_t(x, Y_x(t), \xi_t) + \nabla_y K_t(x, Y_x(t), \xi_t) \cdot N_t .$$

After having found N_t by this recursion, one may calculate

$$\nabla_x H(Y_x(t)) = \nabla_y H(Y_x(t)) \cdot N_t .$$

This pointwise calculation carries over to the expectation under the standard assumptions of dominated convergence, yielding

$$\nabla_x \mathsf{E}[H(Y_x(t))] = \mathsf{E}[\nabla_y H(Y_x(t)) \cdot N_t].$$

Now, the estimate for the problem in transient regime is

$$\widehat{\nabla_x F}(x) = \sum_{t=1}^{T} H(Y_x(t)) \cdot N_t ,$$

whereas for the stationary regime one uses

$$\widehat{\nabla_x F}(x) = \frac{1}{T-\tau} \sum_{t=\tau+1}^{T} H(Y_x(t)) \cdot N_t,$$

where T is large and τ stands for the warmup-phase of the process, which is skipped for the estimation. Of course, the latter estimate is biased, it bias decreases with increasing T and τ.

Distributional Derivatives

Suppose that the Markov transition has transition density $p_x(y_1 \mid y_0)$, i. e.

$$\mathsf{P}(Y_x(t+1) \in A | Y_x(t) = y_0) = \int_A p_x(y_1|y_0)\, dy_1$$

and starts in state y_0. The expectation of $H(Y_x(t)$ is

$$\mathsf{E}[H(Y_x(t))] = \int \cdots \int H(y_t) \prod_{i=1}^{t} p_x(y_i|y_{i-1})\, dy_t \cdots dy_1 .$$

Introduce the *score function*

$$s_x(y_0, \ldots, y_t) = \sum_{i=1}^{t} \frac{\nabla_x p_x(y_i|y_{i-1})}{p_x(y_i|y_{i-1})} .$$

By the product rule we get the formula

$$\nabla_x \mathsf{E}[H(Y_x(t))] = \int \cdots \int H(y_t) s_x(y_0, \ldots, y_t) \times \prod_{i=1}^{t} p_x(y_i|y_{i-1}) \cdots dy_t \cdots dy_1 .$$

An estimate for $\nabla_x\, \mathsf{E}[H(Y_x(t))]$ is

$$H(Y_x(t)) \cdot W_x(t),$$

where $W_x(t) = s_x(Y_x(0), \ldots, Y_x(t))$ is called the *score function martingale*. As before, the estimate for the problem in transient regime is

$$\widehat{\nabla_x F}(x) = \sum_{t=1}^{T} H(Y_x(t)) \cdot W_x(t),$$

whereas the estimate for the stationary regime is

$$\widehat{\nabla_x F}(x) = \frac{1}{T-\tau} \sum_{t=\tau}^{T} H(Y_x(t)) \cdot W_x(t).$$

It is asymptotically unbiased for $T, \tau \to \infty$ (see [2]).

There is also the a way of attacking directly the derivative of the stationary distribution: Let P_x represent the transition matrix (transition operator) of the Markov process. The stationary distribution π_x satisfies

$$\pi_x = \pi_x \cdot P_x$$

and therefore

$$\nabla_x \pi_x = [\nabla_x \pi_x] \cdot P_x + \pi_x \cdot [\nabla_x P_x],$$

i. e.

$$\nabla_x \pi_x = \pi_x [\nabla_x P_x] S_x , \tag{4}$$

with

$$S_x = \sum_{k=0}^{\infty} (P_x^k - 1 \cdot \pi_x).$$

Here $1 {\cdot} \pi_x$ is the transition with rows being identical to π_x. The operator S_x solves the *Poisson equation*

$$S_x (I - P_x) = I,$$

where I is the identity operator. There is a method, to use equation(4)) as the basis for estimating $\nabla_x\, \mathsf{E}[Y_x(\cdot)]$, see [1, Chapt. 3].

Regenerative Processes

Recall that a set A is a *regenerative set* of the ergodic Markov transition P if

i) $u \to P(u, B)$ is independent of $u \in A$, for all B; and
ii) $\pi(A) > 0$, where π is the unique stationary probability measure pertaining to P.

Suppose that A is a regenerative set for all transitions P_x. The sequence of *regenerative stopping times* of $Y_x(t)$is

$$T_1^{(A)} = \min \{t \colon\; Y_x(t) \in A\} ,$$

$$T_{i+1}^{(A)} = \min \left\{ t > T_i^{(A)} \colon\; Y_x(t) \in A \right\} .$$

These stopping times cut the process into independent pieces. For a process Y_x started in A, the following fundamental equation relates the finite time behavior to the stationary, i. e. long run behavior:

$$\mathsf{E}[H(Y_x(\infty))] = \frac{\mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right]}{\mathsf{E}(T^{(A)})} . \tag{5}$$

The score method for derivative estimation gives

$$\nabla_x \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right] = \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t)) W_x(t)\right]$$

and

$$\nabla_x \mathsf{E}[T^{(A)}] = \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} W_x(t)\right]$$

and — by the quotient rule —

$$\begin{aligned} &\nabla_x \mathsf{E}[H(Y_x(\infty))] \\ &= \frac{\mathsf{E}(T^{(A)}) \cdot \nabla_x \mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right]}{[\mathsf{E}(T^{(A)})]^2} \\ &- \frac{\mathsf{E}\left[\sum_{t=1}^{T^{(A)}} H(Y_x(t))\right] \cdot \nabla_x \mathsf{E}(T^{(A)})}{[\mathsf{E}(T^{(A)})]^2} \end{aligned}$$

(see [2]). For the estimation of $\nabla_x \mathsf{E}[H(Y_x(\infty))]$, all expectations of the right-hand side have to be replaced by estimates.

See also

- ▶ Derivatives of Probability and Integral Functions: General Theory and Examples
- ▶ Derivatives of Probability Measures
- ▶ Discrete Stochastic Optimization
- ▶ Optimization in Operation of Electric and Energy Power Systems

References

1. Pflug GCh (1996) Optimization of stochastic models: the interface between simulation and optimization. Kluwer, Dordrecht
2. Rubinstein RY, Shapiro A (1993) Discrete event systems: Sensitivity and stochastic optimization by the score function method. Wiley, New York

Derivatives of Probability and Integral Functions: General Theory and Examples

S. URYASEV
Department Industrial and Systems Engineering, University Florida, Gainesville, USA

MSC2000: 90C15

Article Outline

Keywords

Probability function; Derivative of an integral; Gradient of an integral; Derivative of a probability function; Gradient of a probability function

Probability functions are commonly used for the analysis of models with uncertainties or variabilities in parameters. For instance, in risk and reliability analysis, performance functions, characterizing the operation of systems, are formulated as probabilities of successful or unsuccessful accomplishment of their missions (core damage probability of a nuclear power plant, probability of successful landing of an aircraft, probability of profitable transactions in a stock market, or percentiles of the risks in public risk assessments). Sensitivity analysis of such performance functions involves evaluating of their derivatives with respect to the parameters. Also, the derivatives of the probability function can be used to solve *stochastic optimization* problems [1].

A probability function can be formally presented as an expectation of a discontinuous indicator function of a set, or as an integral over a domain — depending upon parameters. Nevertheless, differentiability conditions of the probability function do not follow from similar conditions of the expectations of continuous (smooth or convex) functions.

The derivative of the probability function has many equivalent representations. It can be represented as an integral over the surface, an integral over the volume, or a sum of integrals over the volume and over the surface. Also, it can be calculated using weak derivatives of the probability measures or conditional expectations.

The first general result on the differentiability of the probability function was obtained by E. Raik [8]. He represented the gradient of the probability function with one constraint in the form of the surface integral. S. Uryasev [10] extended Raik's formula for probability functions with many constraints. A.I. Kibzun and G.L. Tretyakov [3] extended it to the piecewise smooth constraint and probability density function. Special cases of probability function with normal and gamma distributions were investigated by A. Prékopa [6]. G.Ch. Pflug [5] represented the gradient of probability function in the form of an expectation using weak probability measures.

Uryasev [9] expressed the gradient of the probability function as a volume integral. Also, using a change of variables, K. Marti [4] derived the probability function gradient in the form of the volume integral.

A general analytical formula for the *derivative of probability functions* with many constraints was obtained by Uryasev [10]; it calculates the gradient as an integral over the surface, an integral over the volume, or the sum of integrals over the surface and the volume. Special cases of this formula correspond to the Raik formula [8], the Uryasev formula[9], and the change-of-variables approach [4].

The gradient of the quantile function was obtained in [2].

Notations and Definitions

Let an *integral over the volume*

$$F(x) = \int_{f(x,y)\le 0} p(x,y)\,dy \tag{1}$$

be defined on the Euclidean space $\mathbf{R}^n$, where $f\colon \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^k$ and $p\colon \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$ are some functions. The inequality $f(x, y) \le 0$ in the integral is a system of inequalities

$$f_i(x,y) \le 0, \quad i = 1,\dots,k.$$

Both the kernel function $p(x, y)$ and the function $f(x, y)$ defining the integration set depend upon the parameter x. For example, let

$$F(x) = \mathsf{P}\{f(x,\zeta(\omega)) \le 0\} \tag{2}$$

be a *probability function*, where $\zeta(\omega)$ is a random vector in $\mathbf{R}^m$. The random vector $\zeta(\omega)$ is assumed to have a probability density $p(x, y)$ that depends on a parameter $x \in \mathbf{R}^n$. The probability function can be represented as an *expectation of an indicator function*, which equals one on the integration set, and equals zero outside of it. For example, let

$$\begin{aligned} F(x) &= \mathsf{E}\left[I_{\{f(x,\boldsymbol{\zeta})\le 0\}}g(x,\boldsymbol{\zeta})\right] \\ &= \int_{f(x,y)\le 0} g(x,y)\rho(x,y)\,dy \\ &= \int_{f(x,y)\le 0} p(x,y)\,dy, \end{aligned} \tag{3}$$

where $I_{\{\cdot\}}$ is an indicator function, andthe random vector ζ in $\mathbf{R}^m$ has a probability density $\rho(x, y)$ that depends on a parameter $x \in \mathbf{R}$.

Integral Over the Surface Formula

The following formula calculates the gradient of an integral (1) over the set given by nonlinear inequalities as sum of integral over the volume plus integral over the surface of the integration set. We call this the *integral over the surface formula* because if the density $p(x, y)$ does not depend upon x the gradient of the integral (1) equals an integral over the surface. This formula for the case of one inequality was obtained by Raik [8] and generalized for the case with many inequalities by Uryasev [10].

Let us denote by $\mu(x)$ the integration set

$$\begin{aligned} \mu(x) &= \{y \in \mathbb{R}^m : \ f(x,y) \le 0\} \\ &:= \{y \in \mathbb{R}^m : \ f_l(x,y) \le 0,\ 1 \le l \le k\} \end{aligned}$$

and by $\partial\mu(x)$ the surface of this set $\mu(x)$. Also, let us denote by $\partial_i\mu(x)$ a part of the surface which corresponds to the function $f_i(x, y)$, i. e.,

$$\partial_i\mu(x) = \mu(x) \cap \{y \in \mathbb{R}^m : \ f_i(x,y) = 0\}\,.$$

If the constraint functions are differentiable and the following integral exists, then gradient of integral (1) equals

$$\nabla_x F(x) = \int_{\mu(x)} \nabla_x p(x,y)\,dy - \sum_{i=1}^{k} \int_{\partial_i \mu(x)} \frac{p(x,y)}{\|\nabla_y f_i(x,y)\|} \nabla_x f_i(x,y)\,dS. \quad (4)$$

A potential disadvantage of this formula is that in multidimensional case it is difficult to calculate the integral over the nonlinear surface. Most well known numerical techniques, such as Monte-Carlo algorithms, are applicable to volume integrals. Nevertheless, this formula can be quite useful in various special cases, such as the linear case.

Example 1 (Linear case: Integral over the surface formula [10].)

Let $A(\omega)$, be a random $l \times n$ matrix with the joint density $p(A)$. Suppose that $x \in \mathbf{R}^n$ and $x_j \neq 0, j = 1, \dots, n$. Let us define

$$F(x) = \mathsf{P}\{A(\omega)x \le b,\ A(\omega) \ge 0\}, \quad b = (b_1, \dots, b_l) \in \mathbb{R}^l, \quad x \in \mathbb{R}^n, \quad (5)$$

i. e. $F(x)$ is the probability that the linear constraints $A(\omega)x \le b$, $A(\omega) \ge 0$ are satisfied. The constraint, $A(\omega) \ge 0$, means that all elements $a_{ij}(\omega)$ of the matrix $A(\omega)$ are nonnegative. Let us denote by A_i and A^i the ith row and column of the matrix A

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_l \end{pmatrix} = \left(A^1, \dots, A^n\right);$$

then

$$f(x,A) = \begin{pmatrix} f_1(x,A) \\ \vdots \\ f_k(x,A) \end{pmatrix} = \begin{pmatrix} A_1 x - b_1 \\ \vdots \\ A_l x - b_l \\ -A^1 \\ \vdots \\ -A^n \end{pmatrix},$$

$$k = l + l \times n.$$

The function $F(x)$ equals

$$F(x) = \int_{f(x,A) \le 0} p(A)\,dA. \quad (6)$$

We use formula (4) to calculate the gradient $\nabla_x F(x)$ as an integral over the surface. The function $p(A)$ does not depend upon x and $\nabla_x p(A) = 0$. Formula (4) implies that $\nabla_x F(x)$ equals

$$-\sum_{i=1}^{k} \int_{\partial_i \mu(x)} \frac{p(A)}{\|\nabla_A f_i(x,A)\|} \nabla_x f_i(x,A)\,dS.$$

Since $\nabla_x f_i(x,A) = 0$ for $i = l+1, \dots, k$, then $\nabla_x F(x)$ equals

$$-\sum_{i=1}^{l} \int_{\partial_i \mu(x)} \frac{p(A)}{\|\nabla_A f_i(x,A)\|} \nabla_x f_i(x,A)\,dS = -\sum_{i=1}^{l} \int_{\partial_i \mu(x)} \frac{p(A)}{\|x\|} A_i^\top\,dS = -\|x\|^{-1} \sum_{i=1}^{l} \int_{\substack{Ax \le b, \\ A \ge 0 \\ A_i x = b_i}} p(A) A_i^\top\,dS.$$

Integral Over the Volume Formula

This section presents gradient of the function (1) in the form of volume integral. Let us introduce the following shorthand notations

$$f_{1l}(x,y) = \begin{pmatrix} f_1(x,y) \\ \vdots \\ f_l(x,y) \end{pmatrix}, \quad f(x,y) = f_{1k}(x,y),$$

$$\nabla_y f(x,y) = \begin{pmatrix} \frac{\partial f_1(x,y)}{\partial y_1} & \dots & \frac{\partial f_k(x,y)}{\partial y_1} \\ & \vdots & \\ \frac{\partial f_1(x,y)}{\partial y_m} & \dots & \frac{\partial f_k(x,y)}{\partial y_m} \end{pmatrix}.$$

Divergence for the $n \times m$ matrix H consisting of the elements h_{ji} is denoted by

$$\operatorname{div}_y H = \begin{pmatrix} \sum_{i=1}^m \frac{\partial h_{1i}}{\partial y_i} \\ \vdots \\ \sum_{i=1}^m \frac{\partial h_{ni}}{\partial y_i} \end{pmatrix}.$$

Following [10], the derivative of the function (1) is represented as an integral over the volume

$$\nabla_x F(x) = \int_{\mu(x)} \nabla_x p(x,y)\,dy + \int_{\mu(x)} \operatorname{div}_y \left(p(x,y) H(x,y)\right)\,dy, \quad (7)$$

where a matrix function $H: \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{n\times m}$ satisfies the equation

$$H(x,y)\nabla_y f(x,y) + \nabla_x f(x,y) = 0. \qquad (8)$$

The last system of equations may have many solutions, therefore formula (7) provides a number of equivalent expressions for the gradient. The following section gives analytical solutions of this system of equations. In some cases, this system does not have any solution, and formula (7) is not valid. The following section deals with such cases and provides a general formula where system of equations can be solved only for some of the functions defining the integration set.

Example 2 (Linear case: Integral over the volume formula [10].)

With formula (7), the gradient of the probability function (5) with linear constrains considered in Example 1 can be represented as the integral over the volume. It can be shown that equation (8) does not have a solution in this case. Nevertheless, we can slightly modify the constraints, such that integration set is not changed and equation (8) has a solution. In the vector function $f(x, A)$ we multiply column A^i on x^i if x^i is positive or multiply it on $-x^i$ if x^i is negative. Therefore, we have the following constraint function

$$f(x,A) = \begin{pmatrix} A_1 x - b_1 \\ \vdots \\ A_l x - b_l \\ -(+)x_1 A^1 \\ \vdots \\ -(+)x_n A^n \end{pmatrix}, \qquad (9)$$

where $-(+)$ means that we take an appropriate sign. It can be directly checked that, the matrix $H_l^*(x, A)$

$$H^*(x,A) = \left(h^1(x,A_1),\dots,h^l(x,A_l)\right),$$

$$h^i(x,A_i) = -\begin{pmatrix} a_{i1}x_1^{-1} & & 0 \\ & \ddots & \\ 0 & & a_{in}x_n^{-1} \end{pmatrix}$$

is a solution of system (8). As it will be shown in the next section, this analytical solution follows from the fact that change of the variables $Y^i = x_i A^i$, $i = 1, \dots, n$, eliminates variables x^i, $i = 1, \dots, n$, from the constraints (9).

Since $\nabla_x p(A) = 0$ and $\operatorname{div}_A(p(A)H^*(x,A))$ equals

$$-\begin{pmatrix} x_1^{-1}\left(lp(A) + \sum_{i=1}^l a_{i1}\frac{\partial}{\partial a_{i1}}p(A)\right) \\ \vdots \\ x_n^{-1}\left(lp(A) + \sum_{i=1}^l a_{in}\frac{\partial}{\partial a_{in}}p(A)\right) \end{pmatrix},$$

formula (7) implies that $\partial F(x)/\,\partial x_j\}$ equals

$$-x_j^{-1}\int_{\substack{Ax\le b\\ A\ge 0}}\left(lp(A) + \sum_{i=1}^l a_{ij}\frac{\partial}{\partial a_{ij}}p(A)\right)\,dA.$$

General Formula

Further, we give a general formula [9,10] for the differentiation of integral (1). A gradient of the integral is represented as a sum of integrals taken over a volume and over a surface. This formula is useful when system of equations (8) does not have a solution. We split the set of constraints $K := = \{1, \dots, k\}$ into two subsets K_1 and K_2.Without loss of generality we suppose that

$$K_1 = \{1,\dots,l\}, \quad K_2 = \{l+1,\dots,k\}.$$

The derivative of integral (1) can be represented as the sum of the volume and surface integrals

$$\begin{aligned}\nabla_x F(x) &= \int_{\mu(x)} \nabla_x p(x,y)\,dy \\ &+ \int_{\mu(x)} \operatorname{div}_y\left(p(x,y)H_l(x,y)\right)\,dy \\ &- \sum_{i=l+1}^k \int_{\partial_i\mu(x)} \frac{p(x,y)}{\|\nabla_y f_i(x,y)\|} \\ &\times \left[\nabla_x f_i(+x,y) + H_l(x,y)\nabla_y f_i(x,y)\right]\,dS, \end{aligned} \qquad (10)$$

where the matrix $H_l: \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{n\times m}$ satisfies the equation

$$H_l(x,y)\nabla_y f_{1l}(x,y) + \nabla_x f_{1l}(x,y) = 0. \qquad (11)$$

The last equation can have a lot of solutions and we can choose an arbitrary one, differentiable with respect to the variable y.

The general formula contains as a special cases the integral over the surface formula (4) and integral over the volume formula (7). When the set K_1 is empty, the matrix H_l is absent and the general formula is reduced to the integral over the surface. Also, when the set K_2 is empty we have integral over the volume formula (7). Except these extreme cases, the general formula provides number of intermediate expressions for the gradient in the form of the sum of an integral over the surface and an integral over the volume. Thus, we have a number of equivalent representations of the gradient corresponding to the various sets K_1 and K_2 and solutions of equation (11).

Equation (11) (and equation (8) which is a partial case of equation (11)) can be solved explicitly. Usually, this equation has many solutions. The matrix

$$- \nabla_x f_{1l}(x,y) \times \left(\nabla_y^\top f_{1l}(x,y) \nabla_y f_{1l}(x,y)\right)^{-1} \nabla_y^\top f_{1l}(x,y) \quad (12)$$

is a solution of equation (11). Also, in many cases there is another way to solve equation (11) using change of variables. Suppose that there is a change of variables

$$y = \gamma(x,z)$$

which eliminates vector x from the function $f(x, y)$ defining integration set, i. e., function $f(x, \gamma(x, z))$ does not depend upon the variable x. Denote by $\gamma^{-1}(x, y)$ the inverse function, defined by the equation

$$\gamma^{-1}(x, \gamma(x,z)) = z.$$

Let us show that the following matrix

$$H(x,y) = \nabla_x \gamma(x,z)|_{z=\gamma^{-1}(x,y)} \quad (13)$$

is a solution of (11). Indeed, the gradient of the function $\gamma(x, y(x, z))$ with respect to x equals zero, therefore

$$\begin{aligned} 0 &= \nabla_x f_{1l}(x, \gamma(x,z)) \\ &= \nabla_x \gamma(x,z) \nabla_y f_{1l}(x,y)|_{y=\gamma(x,z)} \\ &\quad + \nabla_x f_{1l}(x,y)|_{y=\gamma(x,z)}, \end{aligned}$$

and function $\nabla_x \gamma(x, z)|_{z=\gamma-1}(x, y)$ is a solution of (11).

Formula (7) with matrix (13) gives the derivative formulas which can be obtained with change of variables in the integration set [4].

Example 3 While investigating the operational strategies for inspected components (see [7]) the following integral was considered

$$F(x) = \int_{\substack{b(y)\le x,\\ y_i \ge \theta,\\ i=1,\dots,m}} p(y)\,dy, \quad (14)$$

where $x \in \mathbf{R}^1$, $y \in \mathbf{R}^m$, p: $\mathbf{R}^m \to \mathbf{R}^1$, $\theta > 0$, $b(y) = \sum_{i=1}^m y_i^\alpha$. In this case

$$f(x,y) = \begin{pmatrix} b(y) - x \\ \theta - y_1 \\ \vdots \\ \theta - y_m \end{pmatrix},$$

and

$$F(x) = \int_{f(x,y)\le 0} p(y)\,dy = \int_{\mu(x)} p(y)\,dy.$$

Let us consider that $l = 1$, i. e. $K_1 = \{1\}$ and $K_2 = \{2, \dots, m+1\}$. The gradient $\nabla_x F(x)$ equals

$$\begin{aligned} &\int_{\mu(x)} \left[\nabla_x p(y) + \mathrm{div}_y \left(p(y) H_1(x,y)\right)\right] dy \\ &- \sum_{i=2}^{m+1} \int_{\partial_i \mu(x)} \frac{p(y)}{\|\nabla_y f_i(x,y)\|} \\ &\times \left[\nabla_x f_i(x,y) + H_1(x,y)\nabla_y f_i(x,y)\right] dS, \end{aligned} \quad (15)$$

Where the matrix $H_1(x, y)$ satisfies (11). In view of

$$\nabla_y f_1(x,y) = \alpha \begin{pmatrix} y_1^{\alpha-1} \\ \vdots \\ y_m^{\alpha-1} \end{pmatrix}, \qquad \nabla_x f_1(x,y) = -1.$$

a solution $H_1^*(x, y)$ of (11) equals

$$\begin{aligned} H_1^*(x,y) = h(y) &:= \left(h_1(y_1), \dots, h_m(y_m)\right) \\ &= \frac{1}{\alpha m}\left(y_1^{1-\alpha}, \dots, y_m^{1-\alpha}\right). \end{aligned} \quad (16)$$

Let us denote

$$\begin{aligned} &(\theta_i|y) = (y_1, \dots, y_{i-1}, \theta, y_{i+1}, \dots, y_m), \\ &y^{-i} = (y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_m), \\ &b(\theta_i|y) = \theta^\alpha + \sum_{\substack{j=1\\ j\ne i}}^m y_j^\alpha. \end{aligned}$$

We denote by $y^{-i} \ge \theta$ the set of inequalities

$$y_j \ge \theta, \quad j = 1, \dots, i-1, \quad i+1, \dots, m.$$

The sets $\partial_i \mu(x)$, $i = 2, \dots, m+1$, have a simple structure

$$\partial_i \mu(x) = \mu(x) \bigcap \{y \in \mathbb{R}^m :\ y_i = \theta\} \\ = \left\{y^{-i} \in \mathbb{R}^{m-1} :\ b(\theta_i|y) \le x,\ y^{-i} \ge 0\right\}.$$

For $i = 2, \dots, m+1$, we have

$$\left(\nabla_y f_i(y)\right)_j = 0, \quad j = 1, \dots, m, \quad j \ne i-1, \tag{17}$$

$$\left(\nabla_y f_i(y)\right)_{i-1} = -1, \quad \left\|\nabla_y f_i(y)\right\| = 1. \tag{18}$$

The function $p(y)$ and the functions $f_i(y)$, $i = 2, \dots, m+1$, do not depend on x, consequently

$$\nabla_x p(y) = 0, \tag{19}$$

$$\nabla_x f_i(y) = 0, \quad i = 2, \dots, m+1. \tag{20}$$

Equations (15)–(20) imply

$$\begin{aligned}\nabla_x F(x) &= \int_{\mu(x)} \operatorname{div}_y \left(p(y)h(y)\right)\, dy \\ &\quad - \sum_{i=2}^{m+1} \int_{\partial_i \mu(x)} \frac{p(y)}{\left\|\nabla_y f_i(y)\right\|} h(y) \nabla_y f_i(y)\, dS \\ &= \int_{\mu(x)} \operatorname{div}_y \left(p(y)h(y)\right)\, dy \\ &\quad + \sum_{i=2}^{m+1} h_{i-1}(\theta) \int_{\partial_i \mu(x)} p(y)\, dS \\ &= \int_{\substack{b(y) \le x, \\ y_i \ge \theta, \\ i=1,\dots,m}} \operatorname{div}_y \left(p(y)h(y)\right)\, dy \\ &\quad + \sum_{i=1}^{m} \frac{\theta^{1-\alpha}}{\alpha m} \int_{\substack{b(\theta_i|y) \le x, \\ y^{-i} \ge \theta}} p(\theta_i|y)\, dy^{-i}.\end{aligned}$$

Since

$$\begin{aligned}&\operatorname{div}_y \left(p(y)h(y)\right) \\ &= h(y) \nabla_y p(y) + p(y) \operatorname{div}_y h(y) \\ &= \frac{1}{\alpha m} \sum_{i=1}^{m} \frac{\partial p(y)}{\partial y_i} y_i^{1-\alpha} + p(y) \frac{1-\alpha}{\alpha m} \sum_{i=1}^{m} y_i^{-\alpha},\end{aligned}$$

we, finally, obtain that the gradient $\nabla_x F(x)$ equals

$$\int_{\substack{b(y) \le x, \\ y_i \ge \theta, \\ i=1,\dots,m}} \sum_{i=1}^{m} \frac{y_i^{-\alpha}}{\alpha m} \left[y_i \frac{\partial p(y)}{\partial y_i} + (1-\alpha) p(y)\right] dy \\ + \frac{\theta^{1-\alpha}}{\alpha m} \sum_{i=1}^{m} \int_{\substack{b(\theta_i|y) \le x, \\ y^{-i} \ge \theta}} p(\theta_i|y)\, dy^{-i}.$$

The formula for $\nabla_x F(x)$ is valid for an arbitrary sufficiently smooth function $p(y)$.

See also

- ▶ Derivatives of Markov Processes and Their Simulation
- ▶ Derivatives of Probability Measures
- ▶ Discrete Stochastic Optimization
- ▶ Optimization in Operation of Electric and Energy Power Systems

References

1. Ermoliev Y, Wets RJ-B (eds) (1988) Numerical techniques for stochastic optimization. Ser Comput Math. Springer, Berlin
2. Kibzun AI, Malyshev VV, Chernov DE (1988) Two approaches to solutions of probabilistic optimization problems. Soviet J Automaton Inform Sci 20(3):20–25
3. Kibzun AI, Tretyakov GL (1996) Onprobability function differentiability. Theory and Control System 2:53–63. (In Russian)
4. Marti K (1996) Differentiation formulas for probability functions: The transformation method. Math Program B 75(2)
5. Pflug GCh (1996) Optimization of stochastic models: the interface between simulation and optimization. Kluwer, Dordrecht
6. Prékopa A (1970) On probabilistic constrained programming, Proc. Princeton Symp. Math. Program. Princeton University Press, Princeton, 113–138
7. Pulkkinen A, Uryasev S (1991) Optimal operational strategies for an inspected component: Solution techniques. Collaborative Paper Internat Inst Appl Systems Anal, Laxenburg, Austria CP-91-13
8. Raik E (1975) The differentiability in theparameter of the probability function and optimization of the probability function via the stochastic pseudogradient method. Eesti NSV Teaduste Akad Toimetised Füüs Mat. 24(1):3–6 (In Russian)
9. Uryasev S (1989) A differentiation formula for integrals over sets given by inclusion. Numer Funct Anal Optim 10(7–8):827–841
10. Uryasev S (1994) Derivatives of probability functions and integrals over sets given by inequalities. J Comput Appl Math 56:197–223
11. Uryasev S (1995) Derivatives of probability functions and some applications. Ann Oper Res 56:287–311

Derivatives of Probability Measures

GEORG PFLUG
University Vienna, Vienna, Austria

MSC2000: 90C15

Article Outline

Keywords

Derivatives; Stochastic optimization

For stochastic optimization problems of the form

$$\begin{cases} \min & F(x) = \int H(x,v)\,d\mu_x(v) \\ \text{s.t.} & x \in S \subseteq \mathbb{R}^d \end{cases} \tag{1}$$

where $H(x, v)$ is a cost function, μ_x a family of probability measures indexed by x and $F(x)$ the objective value function (OVF), the necessary condition $\nabla_x F(x) = 0$ must be expressed in terms of the derivatives of $H(x, \cdot)$ and μ_x w.r.t. x. In particular, concepts of differentiability of probability measures are needed.

Direct Differentiability

Suppose that the family (μ_x) is *dominated*, i. e. there is a Borel measure ν such that the densities

$$g_x(v) = \frac{d\mu_x}{d\nu}(v)$$

exist for all x. Then the differentiability of the measures may be defined by the differentiability of the densities.

Definition 1 The family of densities $(g_x(v))$ is called *strongly $L_1(\nu)$-differentiable* if there is a vector of integrable functions $\nabla_x g_x = (g'_{x,1}, \dots, g'_{x,d})^\top$ such that

$$\int \left| g_{x+h}(v) - g_x(v) - h^\top \cdot \nabla_x g_x(v) \right| \, d\nu(v)$$
$$b = o(\|h\|) \quad \text{as } \|h\| \downarrow 0. \tag{2}$$

The family of densities $(g_x(v))$ is called *weakly $L_1(\nu)$-differentiable* if there is a vector of $L_1(\nu)$ functions $\nabla_x g_x = (g'_{x,1}, \dots, g'_{x,d})^\top$ such that for every bounded measurable function H

$$\int [g_{x+h}(v) - g_x(v) - h^\top \cdot \nabla_x g_x(v)] H(v)\, d\nu(v)$$
$$= o(\|h\|) \quad \text{as } \|h\| \downarrow 0\,. \tag{3}$$

Weak differentiability implies strong differentiability but not vice versa.

There is also a notion of differentiability for families (μ_x), which do not possess densities (see [3]).

If the densities (g_x) are differentiable and $H(x, v)$ is boundedly differentiable in x and bounded and continuous in v, then the gradient of $F(x) = \int H(x, v) g_x(v) d\nu(v)$ is

$$\int \nabla_x H(x,v) g_x(v)\, d\nu(v) + \int H(x,v) \nabla_x g_x(v)\, d\nu(v).$$

Inverse Differentiability

The family (μ_x) is called *process differentiable* if there exists a family of random variables $V_x(\omega)$ — the *process representation* — defined on some probability space $(\Omega, \mathcal{A}, \mathsf{P})$, such that:

a) $V_x(\cdot)$ has distribution μ_x for all x; and
b) $x \longmapsto V_x(\omega)$ is differentiable a.s.

As an example, let μ_x be exponential distributions with densities $g_x(v) = x \exp(-x \cdot u)$. Then $V_x(\omega) = (1/x) U$ for $U \sim$ Uniform $[0, 1]$ is a process representation in the sense of a) and differentiable in the sense of b) with derivative $\nabla_x V_x(\omega) = -(1/x^2) U$.

Process differentiability does not imply and is not implied by weak differentiability. If $G_x(u) = \int_{-\infty}^{u} g_x(v) dv$ is the distribution function, then process differentiability is equivalent to the differentiability of $x \longmapsto G_x^{-1}(u)$, whereas the weak differentiability is connected to the differentiablity of $x \longmapsto G_x(u)$.

If $V_x(\cdot)$ is a process representation of (μ_x), then the objective function

$$F(x) = \int H(x,v)\, d\mu_x(v) = \mathsf{E}[H(x, V_x)]$$

has derivative

$$\nabla_x F(x) = \mathsf{E}[H_x(x, V_x) + H_v(x, V_x) \cdot \nabla_x V_x].$$

where $H_x(x, v) = \nabla_x H(x, v)$ and $H_v(x, v) = \nabla_v H(x, v)$.

Simulation of Derivatives

If the objective function F in (1) is easily calculated, then the stochastic optimization problem reduces to a standard nonlinear deterministic optimization problem. This is however the exception. In the majority of applications, the objective function value has to be approximated either by a numeric integration technique or a Monte-Carlo (MC) estimate. In the same manner, the gradient $\nabla_x F(x)$ may be approximated either by numerical integration or by Monte-Carlo simulation. We discuss here the construction of MC estimates for the gradient $\nabla_x F(x)$. For simplicity, we treat only the univariate case $x \in \mathbf{R}^1$.

We begin with recalling the Monte-Carlo (MC) method for estimating $F(x)$. If $(V_x^{(i)})$ is a sequence of independent identically distributed random variables with distribution function G_x, then the MC estimate

$$\widehat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} H(x, V_x^{(i)})$$

is an unbiased estimate of $F(x)$.

Process Derivatives

If the family (μ_x) has differentiable process representation (V_x), then

$$\widehat{\nabla_x F_n}(x) = \frac{1}{n}\sum_{i=1}^{n}\Big[H_x(x, V_x^{(i)}) + H_v(x, V_x^{(i)}) \cdot \nabla_x V_x^{(i)}\Big] \quad (4)$$

is a MC estimate of $\nabla_x F(x)$. The method of using the process derivative (4) is also called *perturbation analysis* ([1,2]).

Distributional Derivatives

If the densities g_x are differentiable, there are two possibilities to construct estimates. First, one may define the *score function* $s_x(v) = [\nabla_x g_x(v)]\big/g_x(v)$ and construct the score function estimate [4]

$$\widehat{\nabla_x F_n}(x) = \frac{1}{n}\sum_{i=1}^{n}\Big[H_x(x, V_x^{(i)}) + H(x, V_x^{(i)}) s_x(V_x^{(i)})\Big],$$

which is unbiased.

Alternatively, one may write the function $\nabla_x g_x(v)$ in the form

$$\nabla_x g_x(v) = c_x[\dot{g}_x(v) - \ddot{g}_x(v)], \quad (5)$$

where $\dot{g}_x$ and $\ddot{g}_x$ are probability densities w.r.t. v, and c_x is a nonnegative constant. One possibility is to set $\dot{g}_x$ resp. $\ddot{g}_x$ as the appropriately scaled positive, resp. negative, part of $\nabla_x g_x$, but other representations are possible as well. Let now $\dot{V}_x^{(i)}$, resp. $\ddot{V}_x^{(i)}$, be random variables with distributions $\dot{g}_x dv$, resp. $\ddot{g}_x dv$. The *difference estimate* is

$$\widehat{\nabla_x F_n}(x) = \frac{1}{n} \times \sum_{i=1}^{n}\Big\{H_x(x, V_x^{(i)}) + c_x[H(x, \dot{V}_x^{(i)}) - H(x, \ddot{V}_x^{(i)})]\Big\},$$

which is unbiased (see [3]).

Example 2 Assume again that (μ_x) are exponential distributions with expectation x. The probability μ_x has density

$$g_x(y) = x \cdot \exp(-xy).$$

Let V_x be distributed according to μ_x. For simplicity, assume that the cost function H does not depend explicitly on x. We need estimates for $\nabla_x E(H(V_x))$. The three methods are:

1) Score derivative: The score function is

$$\frac{\nabla_x g_x(v)}{g_x(v)} = \frac{1}{x} - v$$

and the score function estimate is

$$\widehat{\nabla_x F}^{(1)} = H\left(V_x)(\frac{1}{x} - V_x\right).$$

2) Difference derivative: There are several representations in the sense of (5). One could use the decomposition of $\nabla_x g_x(\cdot)$ into its positive and negative part (*Jordan–Hahn decomposition*) and get the estimate

$$\widehat{\nabla_x F}^{(2a)} = \frac{1}{x}(H(\dot{V}_x) - H(\ddot{V}_x)),$$

where $\dot{V}_x$ has density

$$xe(1 - xv)e^{-xv} \cdot 1_{v \le \frac{1}{x}}$$

and $\ddot{V}_x$ has density

$$xe(xv-1)e^{-xv}\cdot 1_{v>\frac{1}{x}}$$

and both are independent.
Another possibility is to set

$$\dot{V}_x = -\frac{1}{x}\log U_1,$$
$$\ddot{V}_x = -\frac{1}{x}(\log U_1 + \log U_2),$$

where U_1, U_2 are independent Uniform [0, 1] variates. The final difference estimate is

$$\widehat{\nabla_x F}^{(2b)} = \frac{1}{x}(H(\dot{V}_x) - H(\ddot{V}_x)).$$

3) Process derivative: A process representation of (μ_x) is

$$V_x = -\frac{1}{x}\log(1-U), \quad U \sim \text{Uniform}[0,1].$$

A process derivative of $H(V_x)$ is

$$\widehat{\nabla_x F}_x^{(3)} := H_x(V_x)(-\frac{1}{x}V_x).$$

Notice that in methods 1) and 2) the function H need not to be differentiable and may be an indicator function – as is required in some applications. In method 3), the function H must be differentiable.

Whenever a MC estimate $\widehat{\nabla_x F(x)}$ has been defined, it can be used in a stochastic quasigradient method (SQG; cf. also ► Stochastic quasigradient methods) for optimization

$$X_{s+1} = \text{pr}_S[X_s - \rho_s \widehat{\nabla_x F_n}(X_s)]$$

where pr_S is the projection on the set S and (ρ_s) are the stepsizes. The important feature of such algorithms is the fact that they work with stochastic estimates. In particular, the sample size n per step can be set to 1 and still convergence holds under regularity assumptions. To put it differently, the SQG allows to approach quickly a neighborhood of the solution even with much noise corrupted estimates.

See also

► Derivatives of Markov Processes and Their Simulation
► Derivatives of Probability and Integral Functions: General Theory and Examples
► Discrete Stochastic Optimization
► Optimization in Operation of Electric and Energy Power Systems

References

1. Glasserman P (1991) Gradient estimation via perturbation analysis. Kluwer, Dordrecht
2. Ho YC, Cao X (1983) Perturbation analysis and optimization of queueing networks. J Optim Th Appl 20:559–589
3. Pflug GC (1996) Optimization of stochastic models. Kluwer, Dordrecht
4. Rubinstein RY, Shapiro A (1993) Discrete event systems: Sensitivity analysis and stochastic optimization by the score function method. Wiley, New York

Design Optimization in Computational Fluid Dynamics

DOYLE KNIGHT
Department Mechanical and Aerospace Engineering, Rutgers University, New Brunswick, USA

MSC2000: 90C90

Article Outline

Keywords

Optimization; Computational fluid dynamics

Synonyms

Design Optimization in CFD

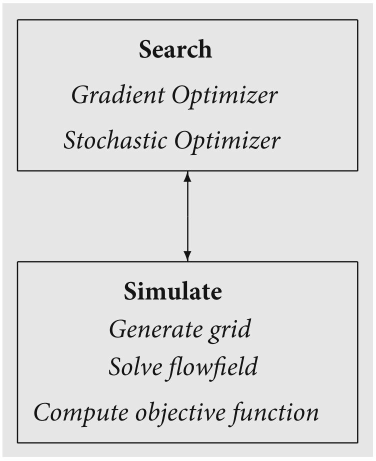
Focus

The article focuses on design optimization using computational fluid dynamics (CFD). *Design* implies the creation of an engineering prototype (e. g., a pump) or engineering process (e. g., particle separator). *Optimization* indicates the selection of a 'best' design. *Computational fluid dynamics* (CFD) represents a family of models of fluid motion implemented on a digital computer. In recent years, efforts have focused on merging elements of these three disciplines to improve design effectiveness and efficiency.

Framework

Consider the design of a prototype or process with n *design variables* $\{x_i\colon i = 1, \ldots, n\}$ denoted by $\mathbf{x}$. It is assumed that n is finite, although infinite-dimensional design spaces also exist (e. g., the shape of a civilian transport aircraft). The domain of $\mathbf{x}$ constitutes the *design space*. A scalar *objective function* $f(\mathbf{x})$ is assumed to be defined for some (or possibly all) points in the design space. This is the simplest design optimization problem. Oftentimes, however, the optimization cannot be easily cast into this form, and other methods (e. g., Pareto optimality) are employed. The purpose of the design optimization is to find the design point $\mathbf{x}^*$ which minimizes f. Note that there is no loss of generality in assuming the objective is to minimize f, since the maximization of an objective function $\widetilde{f}(\mathbf{x})$ is equivalent to the minimization of $f = -\widetilde{f}$.

The design optimization is typically an iterative process involving two principal elements. The first element is the *simulation* which evaluates the objective function by (in the case of computational fluid dynamics) a fluid flow code (*flow solver*). The second element is the *search* which determines the direction for traversing the design space. The search engine is the *optimizer* of which they are several different types as described later. The design optimization process is an iterative procedure involving repetitive simulation and search steps until

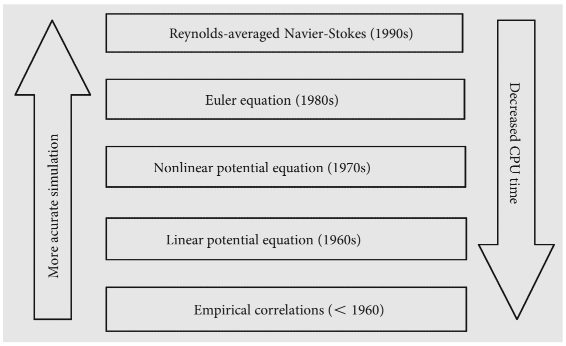


Design Optimization in Computational Fluid Dynamics, Figure 1
Elements of design optimization

a predefined convergence criteria is met. This is illustrated in Fig. 1.

Levels of Simulation

There are five levels of complexity for CFD simulation Fig. 2. *Empirical methods* represent correlations of experimental data and possibly simple one-dimensional analytical models. An example is the NIDA code [15] employed for analysis of two-dimensional and axisymmetric inlets. The code is restricted to a limited family of geometries and flow conditions (e. g., no sideslip). Codes based on the *linear potential* equations (e. g., PANAIR [6]; see also [17]) and *nonlinear potential* equations (e. g., [8]; see also [7]) incorporate increased geometric flexibility while implementing a simplified model of the flow physics (i. e., it is assumed that the shock waves are weak and there is no significant flow separation). Codes employing the *Euler equations* (e. g., [22]) allow for strong shocks and vorticity although neglect viscous effects. *Reynolds-averaged Navier–Stokes codes* (RANS codes) (e. g., GASP [31]) employ a model for the effects of turbulence. The range of execution time between the lowest and highest levels is roughly three orders of magnitude, e. g., on a conventional workstation the NIDA code requires only a few seconds execution time while a 2-dimensional RANS simulation would typically require a few hours.


Reynolds-averaged Navier-Stokes (1990s)
Euler equation (1980s)
Nonlinear potential equation (1970s)
Linear potential equation (1960s)
Empirical correlations (< 1960)
More acurate simulation
Decreased CPU time

Design Optimization in Computational Fluid Dynamics, Figure 2
Levels of CFD simulation

The Stages of Design

There are typically three stages of design: *conceptual*, *preliminary* and *detailed*. As the names suggest, the design specification becomes more precise at successive design stages. Thus, for example, a conceptual design of a civilian transport aircraft may consider a (discrete) design space with the possibility of two, three or four engines, while the preliminary design space assumes a fixed number of engines and considers the details of the engine (e. g., nacelle shapes). It is important to note that the CFD algorithms employed in each of these three stages are likely to be different. Typically, the conceptual design stage employs empirical formulae, while the preliminary design stage may also include simplified CFD codes (e. g., linearized and nonlinear potential methods, and Euler codes), and the detailed design stage may utilize full Reynolds-averaged Navier–Stokes methods. Additionally, experiment is oftentimes essential to verify key features of the design.

Emergence of Automated Design Optimization Using CFD

Although the first numerical simulation of viscous fluid flow was published in 1933 by A. Thom [51], CFD as a discipline emerged with the development of digital mainframe computers in the 1960s. With the principal exception of the work on inverse design methods for airfoils (see, for example, the review [30] and [48]), CFD has mainly been employed in *design analysis* as a cost-effective replacement for some types of experiments. However, CFD can now be employed as part of an *automated design optimization process*. This opportunity has arisen for five reasons. First, the continued rapid improvements in computer performance (e. g., doubling of microprocessor performance every 18 to 24 months [3]) enable routine numerical simulations of increasing sophistication and complexity. Second, improve- ments in the accuracy, efficiency and robustness of CFD algorithms (see, for example, [18]) likewise contribute to the capability for simulation of more complex flows. Third, the development of more accurate turbulence models provides increased confidence in the quality of the flow simulations [16]. Fourth, the development of efficient and robust optimizers enable automated search of design spaces [33]. Finally, the development of sophisticated shell languages (e. g., Perl [43]) provide effective control of pathological events which may occur in an automated design cycle using CFD (e. g., square root of a negative number, failure to converge within a predetermined number of iterations, etc.).

Problem Definition

The general scalar *nonlinear optimization problem* (also known as the *nonlinear programming problem*) is [11,33,52]

$$\text{minimize } f(\mathbf{x}), \tag{1}$$

where $f(\mathbf{x})$ is the scalar *objective function* and $\mathbf{x}$ is the vector of design variables. Typically there are limits on the allowable values of $\mathbf{x}$:

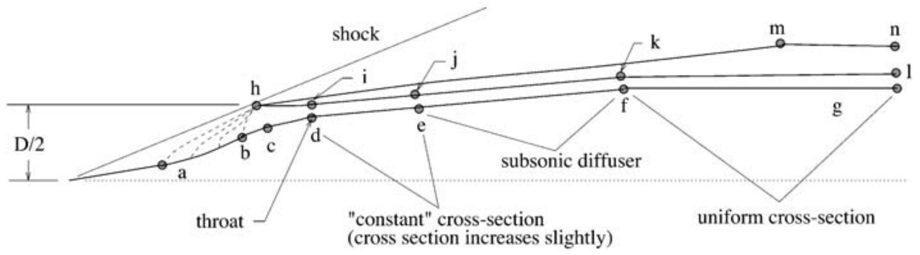
$$\mathbf{a} \le \mathbf{x} \le \mathbf{b}, \tag{2}$$

and m additional *linear* and/or *nonlinear constraints*

$$\begin{cases} c_i(\mathbf{x}) = 0, & i = 1, \dots, m', \\ c_i(\mathbf{x}) \le 0, & i = m' + 1, \dots, m. \end{cases} \tag{3}$$

If f and c_i are linear functions, then the optimization problem is denoted the *linear programming problem*, while if f is quadratic and the c_i are linear, then the optimization problem is denoted the *quadratic programming problem*.

An example of a nonlinear optimization problem using CFD is the design of the shape of an inlet for a supersonic missile. The geometry model of an axisymmetric inlet [53] is shown in Fig. 3.



Design Optimization in Computational Fluid Dynamics, Figure 3
Geometry of high speed inlet

The eight design variables are listed below.

Item	Definition
θ_1	initial cone angle
θ_2	final cone angle
x_d	x-coordinate of throat
r_d	r-coordinate of throat
x_e	x-coordinate of end of 'constant' cross section
θ_3	internal cowl lip angle
H_{ej}	height at end of 'constant' cross section
H_{fk}	height at beginning of 'constant' cross section

There are no general methods for guaranteeing that the *global* minimum of an *arbitrary* objective function $f(\mathbf{x})$ can be found in a finite number of steps [4,11]. Typically, methods focus on determining a *local* minimum with additional (often heuristic) techniques to avoid convergence to a local minimum which is not the global minimum.

A point $\mathbf{x}^*$ is a (strong) *local minimum* [11] if there is a region surrounding $\mathbf{x}^*$ wherein the objective function is defined and $f(\mathbf{x}) > f(\mathbf{x}^*)$ for $\mathbf{x} \neq \mathbf{x}^*$. Provided $f(\mathbf{x})$ is twice continuously differentiable (this is not always true; see, for example, [53]), necessary and sufficient conditions for the existence of a solution to (1) subject to (3) may be obtained [11]. In the one-dimensional case with no constraints the sufficient conditions for a minimum at x^* are

$$g = 0 \text{ and } H > 0 \quad \text{at } x = x^*,$$

where $g = df/dx$ and $H = d^2f / dx^2$. For the multidimensional case with no constraints

$$|g_i| = 0 \quad \text{and} \quad H \text{ is positive definite at } x = x^*, \quad (4)$$

where $g_i = \partial f / \partial x_i$, $|g_i|$ is the norm of the vector g_i, and $H = H_{ij}$ is the *Hessian matrix*

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}.$$

The matrix H is positive definite if all of the eigenvalues of H are positive.

Algorithms for Optimization

The efficacy of an optimization algorithm depends strongly on the nature of the design space. In engineering problems, the design space can manifest pathological characteristics. The objective function f may pos sess multiple local optima [36] arising from physical and/or numerical reasons. Examples of the latter include noise introduced in the objective function by grid refinement between successive flow simulations, and incomplete convergence of the flow simulator. Also, the objective function f and/or its gradient g_i may exhibit near discontinuities for physical reasons. For example, a small change in the the design state $\mathbf{x}$ of a mixed compression supersonic inlet operating at critical conditions can cause the terminal shock to be expelled, leading to a rapid decrease in total pressure recovery [44]. Moreover, the objective function f may not be evaluable at certain points. This may be due to constraints in the flow simulator such as a limited range of applicability for empirical data tables.

A brief description of some different classes of general optimizers is presented. These methods are described for the unconstrained optimization problem for reasons of brevity. See [33] for an overview of optimization algorithms and software packages, and [11] for a comprehensive discussion of the constrained optimization problem. Detailed mathematical exposition of optimization problems is presented in [19].

Gradient Optimizers

If the objective function f can be approximated in the vicinity of a point $\widetilde{\mathbf{x}}$ by a quadratic form, then

$$f \approx \widetilde{f} + \widetilde{g}_i(x_i - \widetilde{x}_i) + \frac{1}{2}(x_i - \widetilde{x}_i)\widetilde{H}_{ij}(x_j - \widetilde{x}_j), \quad (5)$$

where $\widetilde{f}$, $\widetilde{g}$ and $\widetilde{H}_{ij}$ imply evaluation at $\widetilde{\mathbf{x}}$ and the Einstein summation convention is implied. In the relatively simple *method of steepest descent* [40], the quadratic term in (5) is ignored, and a line minimization is performed along the direction of $-g_i$, i. e., a sequence of values of the design variable $\mathbf{x} =^{(\nu)}$, $\nu = 1, \ldots,$ are formed according to

$$\mathbf{x}^{(\nu)} = \widetilde{\mathbf{x}} + \delta\mathbf{x}^{(\nu)}$$

where

$$\delta x_i^{(\nu)} = -\lambda^{(\nu)}\widetilde{g}_i\,|\widetilde{g}_i|^{-1}$$

and $\lambda^{(\nu)}$, $\nu = 1, \ldots,$ are an increasing sequence of displacements. The estimated decrease in the objective function f is $-\lambda^{(\nu)}\,|\widetilde{g}_i|$. The objective function f is evaluated at each iteration ν and the search is terminated when f begins to increase. At this location, the gradient g_i is computed and the procedure is repeated. This method, albeit straightforward to implement, is inefficient for design spaces which are characterized by long, narrow 'valleys' [40].

The *conjugate gradient methods* [40] are more efficient than the method of steepest descent, since they perform a sequence of line minimizations along specific directions in the design space which are mutually orthogonal in the context of the objective function. Consider a line minimization of f along a direction $\mathbf{u} = \{u_i\colon i = 1, \ldots, n\}$. At any point on the line, the gradient of f in the direction of $\mathbf{u}$ is $u_i\widetilde{g}_i$ by definition. At the minimum point $\widetilde{\mathbf{x}}$ in the line search,

$$u_i\widetilde{g}_i = 0$$

by definition. Consider a second line minimization of f along a direction $\mathbf{v}$. From (5) and noting that H_{ij} is symmetric, the change in g_i along the direction $\mathbf{v}$ is $\widetilde{H}_{ij}v_j$. Thus, the condition that the second line minimization also remain a minimization along the first direction $\mathbf{u}$ is

$$u_i\widetilde{H}_{ij}v_j = 0$$

When this condition is satisfied, $\mathbf{u}$ and $\mathbf{v}$ are denoted *conjugate pairs*. Conjugate gradient methods (CGM) generate a sequence of directions $\mathbf{u}, \mathbf{v}, \ldots$ which are mutually conjugate. If f is exactly quadratic, then CGM yield an n-step sequence to the minimum.

Sequential quadratic programming methods employ the Hessian H which may be computed directly when economical or may be approximated from the sequence of gradients g_i generated during the line search (the *quasi-Hessian* [33]). Given the gradient and Hessian, the location x_i^* of the minimum value of f may be found from (5) as

$$\widetilde{H}_{ij}(x_j^* - \widetilde{x}_j) = -\widetilde{g}_i .$$

For the general case where f is not precisely quadratic, a line minimization is typically performed in the direction $(x_i^* - \widetilde{x}_i)$, and the process is repeated.

Variational sensitivity employs the concept of direct differentiation of the optimization function f and governing fluid dynamic equations (in continuous or discrete form) to obtain the gradient g_i, and optimization using a gradient-based method. It is related to the theory of the control of systems governed by partial differential equations [29,39]. For example, the boundary shape (e. g., airfoil surface) is viewed as the (theoretically infinite-dimensional) design space which controls the objective function f. Several different formulations have been developed depending on the stage at which the numerical discretization is performed, and the use of direct or adjoint (costate) equations. Detailed descriptions are provided in [23] and [24]. Additional references include [2,5,20,21,37,38,50].

The following summary follows the presentation in [24] which employs the adjoint formulation. The objective function f is considered to be a function of the flowfield variables w and the physical shape S. The differential change in the objective function is therefore

$$\delta f = \frac{\partial f}{\partial w}\delta w + \frac{\partial f}{\partial S}\delta S. \quad (6)$$

The discretized governing equations of the fluid motion are represented by the vector of equations

$$R(w; S) = 0$$

and therefore

$$\delta R = \frac{\partial R}{\partial w}\delta w + \frac{\partial R}{\partial S}\delta S = 0, \tag{7}$$

where δR is a vector. Assume a vector Lagrange multiplier ψ and combining (6) and (7)

$$\delta f = \left\{ \frac{\partial f}{\partial w} - \psi^{\top} \frac{\partial R}{\partial w} \right\} \delta w + \left\{ \frac{\partial f}{\partial S} - \psi^{\top} \frac{\partial R}{\partial S} \right\} \delta S,$$

where $^{\top}$ indicates vector transpose. If ψ is chosen to satisfy the adjoint (costate) equation

$$\psi^{\top} \frac{\partial R}{\partial w} = \frac{\partial f}{\partial w}, \tag{8}$$

then

$$\delta f = G \delta S,$$

where

$$G = \frac{\partial f}{\partial S} - \psi^{\top} \frac{\partial R}{\partial S}.$$

This yields a straightforward method for optimization using, for example, the method of steepest descent. The increment in the shape is

$$\delta S = -\lambda G,$$

where λ is a positive scalar. The variational sensitivity approach is particularly advantageous when the dimension n of the design space (which defines S) is large, since the gradient of S is obtained from a single flowfield solution (7) plus a single adjoint solution (8) which is comparable to the flowfield solution in cost. Constraints can be implemented by projecting the gradient onto an allowable subspace in which the constraints are satisfied.

Response surface methods employ an approximate representation of the objective function using smooth functions which are typically quadratic polynomials [25]. For example, the objective function may be approximated by

$$f \approx \widehat{f} = \alpha + \sum_{1 \le i \le n} \beta_i x_i + \sum_{1 \le i \le j \le n} \gamma_{ij} x_i x_j$$

where α, β_i and γ_{ij} are coefficients which are determined by fitting $\widehat{f}$ to a discrete set of data using the method of least squares. The minimum of $\widehat{f}$ can then be found by any of the gradient optimizers, with optional recalibration of the coefficients of $\widehat{f}$ as needed. There are many different implementations of the response surface method (see, for example, [12,34] and [46]).

Stochastic Optimizers

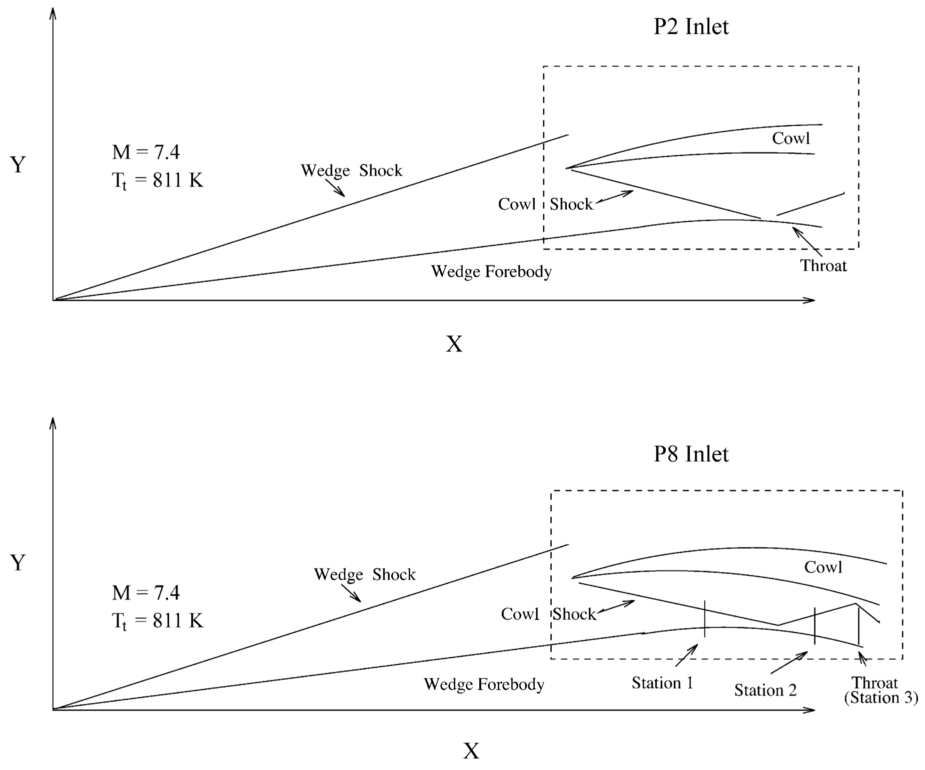
Often the objective function is not well behaved in a portion or all of the design space as discussed above. In such situations, gradient methods can stop without achieving the global optimum (e. g., at an infeasible point, or a local minimum). Stochastic optimizers seek to avoid these problems by incorporating a measure of randomness in the optimization process, albeit oftentimes at a cost of a significant increase in the number of evaluations of the objective function f.

Simulated annealing [26,27,32] mimics the process of crystalization of liquids or annealing of metals by minimizing a function E which is analogous to the energy of a thermodynamic system. Consider a current point (state) in the design space $\widetilde{\mathbf{x}}$ and its associated 'energy' $\widetilde{E}$. A candidate for the next state $\mathbf{x}^*$ is selected by randomly perturbing typically one of the components $\widetilde{x}_j$, $1 \le j \le n$, of, $\widetilde{\mathbf{x}}$ and its energy E^* is evaluated (typically, each component of $\mathbf{x}$ is perturbed in sequence). If $E^* < \widetilde{E}$ then $\widetilde{\mathbf{x}} = \mathbf{x}^*$, i. e., the next state is $\mathbf{x}^*$. If $E^* > \widetilde{E}$ then the probability of selecting $\mathbf{x}^*$ as the next design state is

$$p = \exp\left(-\frac{E^* - \widetilde{E}}{kT} \right),$$

where k is the 'Boltzman constant' (by analogy to statistical mechanics) and T is the 'temperature' which is successively reduced during the optimization according to an assumed schedule [27]. (Of course, only the value of the product kT is important.) The stochastic nature can be implemented by simply calling a random number generator to obtain a value r between zero and one. Then the state $\mathbf{x}^*$ is selected if $r < p$. Therefore, during the sequence of design states, the algorithm permits the selection of a design state with $E > \widetilde{E}$, but the probability of selecting such a state decreases with increasing $E - \widetilde{E}$. This feature tends to enable (but does not guarantee) the optimizer to 'jump out' of a local minimum.

Genetic algorithms (GAs) mimic the process of biological evolution by means of random changes (mutations) in a set of designs denoted the *population* [14]. At each step, the 'least fit' member(s) of the population (i. e., those designs with the highest value of f) are typically removed, and new members are generated by a recombination of some (or all) of the remaining members. There are numerous GA variants. In the approach of [41], an initial population P of designs is generated



Design Optimization in Computational Fluid Dynamics, Figure 4
P2 and P8 inlets

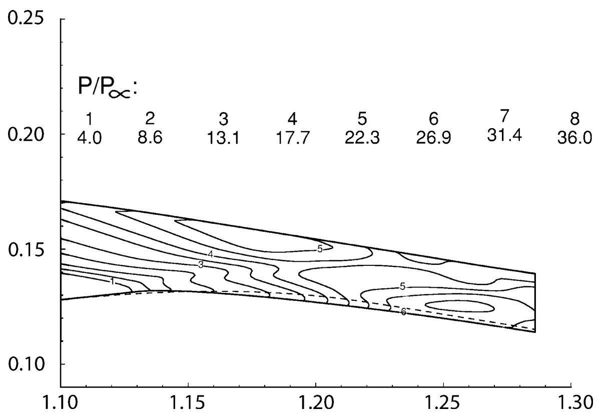
by randomly selecting points $\mathbf{x}_i$, $i = 1, \ldots, p$, satisfying (2). The two best designs (i. e., with the lowest values of f) are joined by a straight line in the design space. A random point $\mathbf{x}'$ is chosen on the line connecting the two best designs. A mutation is performed by randomly selecting a point $\mathbf{x}_{p+1}$ within a specified distance of $\mathbf{x}'$. This new point is added to the population. A member of the population is then removed according to a heuristic criterion, e. g., among the k members with the highest f, remove the member closest to $\mathbf{x}_{p+1}$, thus maintaining a constant number of designs in the population. The removal of the closest member tends to prevent clustering of the population (i. e., maintains diversity). The process is repeated until convergence.

Examples

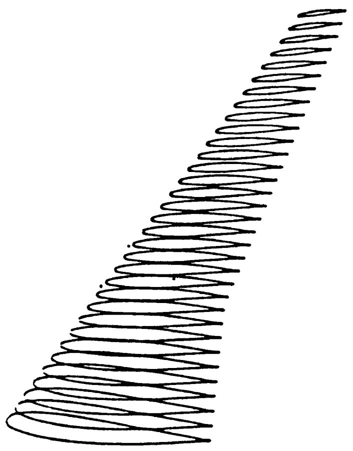
Examples of the above algorithms for optimization using CFD are presented. All of the examples are single discipline involving CFD only. It is emphasized that multidisciplinary optimization (MDO) involving computational fluid dynamics, structural dynamics, electromagnetics, materials and other disciplines is a very active and growing field, and many of the optimization algorithms described herein are appropriate to MDO also. A recent review is presented in [49].

Sequential Quadratic Programming

V. Shukla et al. applied a sequential quadratic programming algorithm CFSQP [28] to the optimal design of two hypersonic inlets (denoted P2 and P8) at Mach 7.4. The geometric model is shown in Fig. 4. The optimization criteria was the minimization of the strength of the shock wave which reflected from the centerbody (lower) surface. This is the same criteria as originally posed in the design of the P2 and P8 inlets [13]. The NPARC flow solver [47] was employed for the P2 optimization, and the GASP flow solver [31] for the P8 optimization.



Design Optimization in Computational Fluid Dynamics, Figure 5
Static pressure contours for optimal P8 inlet (the original centerbody contour is shown by the dotted line)



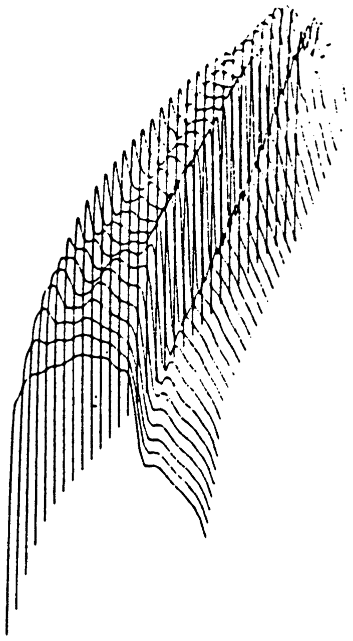
Design Optimization in Computational Fluid Dynamics, Figure 6
Initial shape of wing

The optimization criteria was met for both inlets. In Fig. 5, the static pressure contours for the optimized P8 inlet are shown. The strength of the reflected shock is negligible.

Variational Sensitivity

A. Jameson et al. [24] applied the methodology of variational sensitivity (control theory) to the optimization of a three-dimensional wing section for a subsonic widebody commercial transport. The design objective was to minimize the drag at a given lift coefficient C_L = 0.55 at Mach 0.83 while maintaining a fixed planform. A two stage procedure was implemented. The first stage employed the Euler equations, while the second stage used the full Reynolds-averaged Navier–Stokes equations. In the second stage, the pressure distribution obtained from the Euler optimization is used as the target pressure distribution.

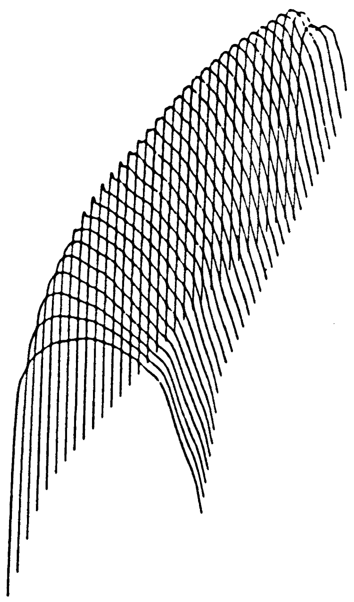
The initial starboard wing shape is shown in Fig. 6 as a sequence of sections in the spanwise direction. The initial pressure distribution on the upper surface, shown as the pressure coefficient c_p plotted with negative values upward, is presented in Fig. 7. A moderately strong shock wave is evident, as indicated by the sharp drop in $-c_p$ at roughly the mid-chord line. After sixty design cycles of the first stage, the drag coefficient was reduced by 15 counts from 0.0196 to 0.0181, and the shock wave eliminated as indicated in the c_p distribution in Fig. 8. A subsequent second stage optimization using the Reynolds–averaged Navier–Stokes equations yielded only slight modifications.



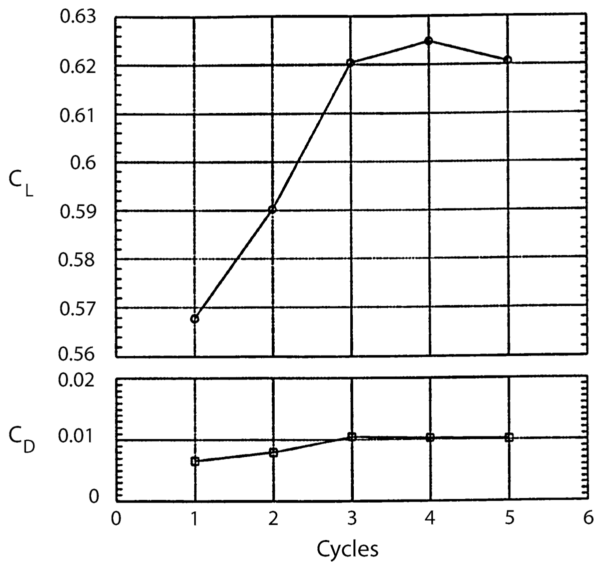
Design Optimization in Computational Fluid Dynamics, Figure 7
Initial surface pressure distribution

Response Surface

R. Narducci et al. [35] applied a response surface method to the optimal design of a two-dimensional



Design Optimization in Computational Fluid Dynamics, Figure 8
Optimized surface pressure distribution



Design Optimization in Computational Fluid Dynamics, Figure 9
Convergence history for transonic airfoil

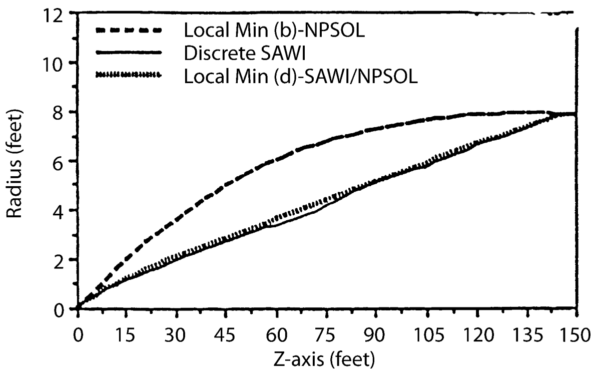
transonic airfoil. The design objective was to maximize the lift coefficient C_L at Mach 0.75 and zero degrees angle of attack, while satisfying the constraints that the drag coefficient $C_D \leq 0.01$ and the thickness ratio $0.075 \leq t \leq 0.15$ where t is the ratio of the maximum airfoil thickness to the airfoil chord. The airfoil surface was represented by a weighted sum of six different shapes which included four known airfoils (a different set of basis functions were employed in [9] for airfoil optimization using a conjugate gradient method). The objective function f was represented by a quadratic polynomial. An inviscid flow solver was employed.

A successful optimization was achieved in five response surface cycles. The history of the convergence of C_L and C_D is shown in Fig. 9. A total of twenty three flow solutions were required for each response surface.

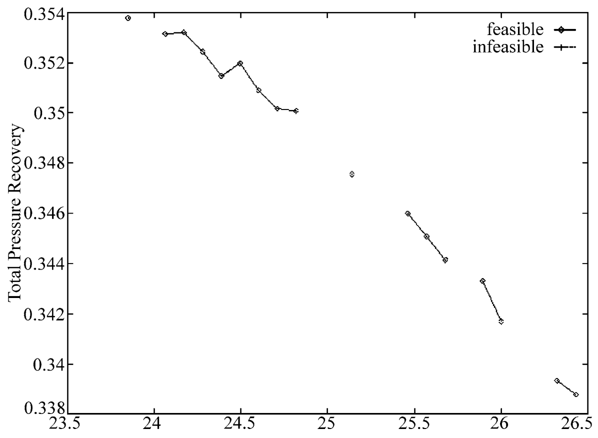
Simulated Annealing

S. Aly et al. [1] applied a modified simulated annealing algorithm to the optimal design of an axisymmetric forebody in supersonic flow. The design objective was to minimize the pressure drag on the forebody of a vehicle at Mach 2.4 and zero angle of attack, subject to constraints on the allowable range of the body radius as a function of axial position. Two different variants of SA were employed, and compared to a gradient optimizer NPSOL [10] which is based on a sequential quadratic programming algorithm. All optimizers employed the same initial design which satisfied the constraints but was otherwise a clearly nonoptimal shape. Optimizations were performed for two different initial shapes. The flow solver was a hybrid finite volume implicit Euler marching method [45].

The first method, denoted simulated annealing with iterative improvement (SAWI), employed SA for the initial phase of the optimization, and then switched to a random search iterative improvement method when close to the optimum. This method achieved from 8% to 31% reduction in the pressure drag, compared to optimal solution obtained NPSOL alone, while requiring fewer number of flowfield simulations (which constitute the principal computational cost). The second method employed SA for the initial phase of the optimization, followed by NPSOL. This approach achieved from 31% to 39% reduction in the pressure drag, compared to the optimal solution obtained by NPSOL alone, while requiring comparable (or less) cputime. The forebody shapes obtained using SA, SA with NPSOL and NPSOL alone are shown in Fig. 10.



Design Optimization in Computational Fluid Dynamics, Figure 10
Forebody shapes obtained using SA, SA with NPSOL and NPSOL. Copyright 1996 AIAA - Reprinted with permission

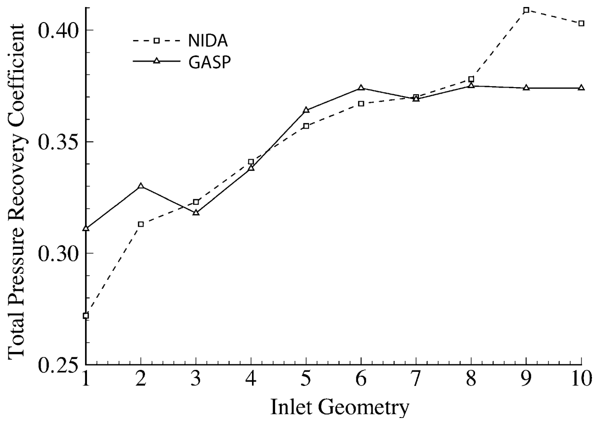


Design Optimization in Computational Fluid Dynamics, Figure 11
Total pressure recovery coefficient versus axial location of throat

Genetic Algorithms

G. Zha et al. applied a modified genetic algorithm (GADO [42]) to the optimal design of an axisymmetric supersonic mixed compression inlet at Mach 4 and 60 kft altitude cruise conditions (see above). The geometric model included eight degrees of freedom (see above), and the optimization criteria was maximization of the inlet total pressure recovery coefficient. The constraints included the requirement for the inlet to start at Mach 2.6, plus additional constraints on the inlet geometry including a minimum cowl thickness and leading edge angle. The constraints were incorporated into the GA using a penalty function. The flow solver was the empirical inlet analysis code NIDA [15]. This code is very efficient, requiring only a few seconds cputime on a workstation, but is limited to 2-dimensional or axisymmetric geometries. Moreover, the design space generated by NIDA (i. e., the total pressure recovery coefficient as a function of the eight degrees of freedom) is nonsmooth with numerous local minima and gaps attributable to the use of empirical data Fig. 11.

The GA achieved a 32% improvement in total pressure recovery coefficient compared to a trial-and-error method [53]. A total of 50 hours on a DEC-2100 workstation was employed. A series of designs generated during the optimization were selected for evaluation by a full Reynolds-averaged Navier–Stokes code (GASP [31]). A close correlation was observed between the predictions of NIDA and GASP Fig. 12.



Design Optimization in Computational Fluid Dynamics, Figure 12
Total pressure recovery coefficient from NIDA and GASP for several different inlet designs

Conclusion

Computational fluid dynamics has emerged as a vital tool in design optimization. The five levels of CFD analysis are utilized in various optimization methodologies. Complex design optimizations have become commonplace. A significant effort is focused on multidisciplinary optimization involving fluid dynamics, solid mechanics, materials and other disciplines.

See also

- ▶ Bilevel Programming: Applications in Engineering
- ▶ Interval Analysis: Application to Chemical Engineering Design Problems
- ▶ Multidisciplinary Design Optimization
- ▶ Multilevel Methods for Optimal Design
- ▶ Optimal Design of Composite Structures
- ▶ Optimal Design in Nonlinear Optics
- ▶ Structural Optimization: History

References

1. Aly S, Ogot M, Pelz R (Sept.–Oct.1996) Stochastic approach to optimal aerodynamic shape design. J Aircraft 33(5):945–961
2. Anderson W, Venkatakrishnan V (1997) Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. AIAA Paper 97–0643, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
3. Berkowitz B (1996) Information age intelligence. Foreign Policy, 103:35–50
4. Boender C, Romeijn H (1995) Stochastic methods. In: Handbook of Global Optimization. Kluwer, Dordrecht, pp 829–869
5. Cabuk H, Modi V (1992) Optimal plane diffusers in laminar flow. J Fluid Mechanics 237:373–393
6. Carmichael R, Erickson L (1981) PAN AIR – A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. AIAA Paper 81–1255, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
7. Caughey D (1982) The computation of transonic potential flows. In: Annual Rev. Fluid Mechanics, 14, pp 261–283
8. Caughey D, Jameson A (Feb. 1979) Numerical calculation of transonic potential flow about wing–body combinations. AIAA J 17(2):175–181
9. Eyi S, Hager J, Lee K (Dec. 1994) Airfoil design optimization using the Navier–Stokes equations. J Optim Th Appl 83(3):447–461
10. Gill P, Murray W, Saunders M, Wright M (1986) User's guide for NPSOL: A FORTRAN package for nonlinear programming. SOL Techn Report Dept Oper Res Stanford Univ 86(2)
11. Gill P, Murray W, Wright M (1981) Practical optimization. Acad. Press, New York
12. Giunta A, Balabanov V, Haim D, Grossman B, Mason W, Watson L (1996) Wing design for a high speed civil transport using a design of experiments methodology. AIAA Paper 96–4001-CP, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
13. Gnos A, Watson E, Seebaugh W, Sanator R, DeCarlo J (Apr. 1973) Investigation of flow fields within large–scale hypersonic inlet models. Techn Note NASA D–7150
14. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA
15. Haas M, Elmquist R, Sobel D (Apr. 1992) NAWC inlet design and analysis (NIDA) code. UTRC Report, R92–970037–1, (United Technologies Res. Center)
16. Haase W, Chaput E, Elsholz E, Leschziner M, Müller U (eds) (1997) ECARP – European computational aerodynamics research project: Validation of CFD codes and assessment of turbulence models. Notes on Numerical Fluid Mechanics. Vieweg, Braunschweig/Wiesbaden
17. Hess J (1990) Panel methods in computational fluid dynamics. In: Annual Rev. Fluid Mechanics, 22, pp 255–274
18. Hirsch C (1988) Numerical computation of internal and external flows, vol I–II. Wiley, New York
19. Horst R, Pardalos PM (eds) (1995) Handbook of global optimization. Kluwer, Dordrecht
20. Ibrahim A, Baysal O (1994) Design optimization using variational methods and CFD. AIAA Paper 94–0093, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
21. Iollo A, Salas M (1995) Contribution to the optimal shape design of two–dimensional internal flows with embedded shocks. ICASE Report 95–20, (NASA Langley Res. Center, Hampton, VA)
22. Jameson A (1982) Steady–state solution of the Euler equations for transonic flow. In: Transonic, Shock and Multidimensional Flows. Acad. Press, New York, pp 37–70
23. Jameson A (1988) Aerodynamic design via control theory. J Sci Comput 3:33–260
24. Jameson A, Pierce N, Martinelli L (1997) Optimum aerodynamic design using the Navier–Stokes equations. AIAA Paper 97–0101, (Amer. Inst. Aeronautics and Astronautics, Reston, VA)
25. Khuri A, Cornell J (1987) Response surfaces: Designs and analyses. M. Dekker, New York
26. Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680
27. van Laarhoven P, Aarts E (1987) Simulated annealing: Theory and Acad. Pressplications. Reidel, London
28. Lawrence AHGC, Zhou J, Tits A (Nov. 1994) User's guide for CFSQP version 2.3: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Techn Report Inst Systems Res Univ Maryland 94–16r1
29. Lions JL (1971) Optimal control of systems governed by partial differential equations. Springer, Berlin (translated from the French)
30. Lores M, Hinson B (1982) Transonic design using computational aerodynamics. In: Progress in Astronautics and Aeronautics, 81. Am Inst Aeronautics and Astronautics, Reston, VA, pp 377–402
31. McGrory W, Slack D, Pressplebaum M, Walters R (1993) GASP version 2.2: The general aerodynamic simulation program. Aerosoft, Blacksburg, VA

32. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953) Equations of state calculations by fast computing machines. J Chem Phys 21:1087–1092
33. Moré J, Wright S (1993) Optimization software guide. SIAM, Philadelphia
34. Myers R, Montgomery D (1995) Response surface methodology: Process and product optimization using design experiments. Wiley, New York
35. Narducci R, Grossman B, Valorani M, Dadone A, Haftka R (1995) Optimization methods for non–smooth or noisy objective functions in fluid dynamic design problems. AIAA Paper 95–1648–CP. Am Inst Aeronautics and Astronautics, Reston, VA
36. Obayashi S, Tsukahara T (1996) Comparison of optimization algorithms for aerodynamic shape design. AIAA Paper 96–2394–CP. Am Inst Aeronautics and Astronautics, Reston, VA
37. Pironneau O (1973) On optimal profiles in Stokes flow. J Fluid Mechanics 59(1):117–128
38. Pironneau O (1974) On optimal design in fluid mechanics. J Fluid Mechanics 64(1):97–110
39. Pironneau O (1984) Optimal shape design for elliptic systems. Springer, Berlin
40. Press W, Flannery B, Teukolsky S, Vetterling W (1986) Numerical recipes. Cambridge Univ. Press, Cambridge
41. Rasheed K, Gelsey A (1996) Adaption of genetic algorithms for continuous design space search. In: Fourth Internat. Conf. Artificial Intelligence in Design: Evolutionary Systems in Design Workshop,
42. Rasheed K, Hirsh H, Gelsey A (1997) A genetic algorithm for continuous design space search. Artif Intell in Eng 11(3):295–305
43. Schwartz R (1993) Learning Perl. O'Reilly, Sebastopol, CA
44. Seddon J, Goldsmith E (eds) (1985) Intake aerodynamics. AIAA Education Ser Amer. Inst. Aeronautics and Astronautics, Reston, VA
45. Siclari M, Del Guidice P (Jan 1990) Hybrid finite volume Approach to Euler solutions for supersonic flows. AIAA J 28(1):66–74
46. Simpson T, Peplinski J, Koch P, Allen J (1997) On the use of statistics in design and the implications for deterministic computer experiments. ASME Paper DETC 97/DTM–3881. Am Soc Mech Engin, New York
47. Sirbaugh J, Smith C, Towne C, Cooper G, Jones R, Power G (Nov. 1994) A users guide to NPARC version 2.0. NASA Lewis Res. Center and Arnold Engin. Developm. Center, Cleveland, OH/Arnold, TN
48. Sobieczky H, Seebass A (1984) Supercritical airfoil and wing design. In: Annual Rev. Fluid Mechanics, 16, pp 337–363
49. Sobieszczanski–Sobieski J, Haftka R (1996) Multidisciplinary aerospace design optimization: Survey of recent developments. AIAA Paper 96–0711. Am Inst Aeronautics and Astronautics, Reston, VA
50. Ta'asan S, Kuruvilla K, Salas M (1992) Aerodyamic design and optimization in one shot. AIAA Paper 92–0025. Am Inst Aeronautics and Astronautics, Reston, VA
51. Thom A (1933) The flow past circular cylinders at low speeds. Proc Royal Soc London A141:651–666
52. Vanderplaats G (1984) Numerical optimization techniques for engineering design: With Applications. McGraw-Hill, New York
53. Zha G, Smith D, Schwabacher M, Rasheed K, Gelsey A, Knight D (Nov.–Dec. 1997) High–performance supersonic missile inlet design using automated optimization. J Aircraft 34(6):697–705

Design of Robust Model-Based Controllers via Parametric Programming

K.I. Kouramas[1], V. Sakizlis[2],
Efstratios N. Pistikopoulos[1]
[1] Centre for Process Systems Engineering, Imperial College London, London, UK
[2] Bechtel Co. Ltd., London, UK

Article Outline

Introduction/Background

Model predictive control (MPC) is very popular for its capacity to deal with multivariable, constraints-model-based control problems for a variety of complex linear or non-linear processes [13]. MPC is based on the receding-time-horizon philosophy where an open-loop, constrained optimal control problem is solved on-line at each sampling time to obtain the optimal control actions. The optimal control problem is solved repetitively at each time when a new measurement or estimate of the state is available, thus establishing an implicit feedback control method [14,15]. The main reasons for the popularity of MPC are its optimal performance, its capability to handle constraints and its inherent robustness due to feedback control properties.

Despite the widely acknowledged capabilities of MPC, there are two main shortcomings that have been a major concern for the industrial and academic communities. The first shortcoming is that MPC implementation is limited to slowly varying processes due to the demanding online computational effort for solving the online optimal control problem. The second is that, despite its inherent robustness due to the implicit feedback, MPC cannot guarantee the satisfaction of constraints and optimal performance in the presence of uncertainties and input disturbances, since usually it relies on nominal models (uncertainty-free models) for the prediction of future states and control actions [14,20,22].

The first shortcoming of MPC can be overcome by employing the so-called parametric MPC (pMPC) or multiparametric MPC (mp-MPC) [4,16,20]. Parametric MPC controllers are based on the well-known *parametric optimization* techniques [9,18] for solving the open-loop optimal control problem offline and obtain the complete map of the optimal control actions as functions of the states. Thus, a feedback control law is obtained offline and the online computational effort is reduced to simple function evaluations of the feedback control. The inevitable presence of uncertainties and disturbances have been ignored by the pMPC community, and only recently has the research started focusing on control problems with uncertainty [2,20]. In traditional MPC the issue of robustness under uncertainty has been dealt with using various methods such as robust model predictive control [3,8], model predictive tubes [6,12] and min-max MPC [21,22]. However, this is still an unexplored area for pMPC, apart from the recent work presented in [2,20].

In this manuscript we discuss the challenges of robust parametric model predictive control (RpMPC) and we present a method for RpMPC for linear, discrete-time dynamic systems with exogenous disturbances (input uncertainty) and a method for RpMPC for systems with model uncertainty. In both cases the uncertainty is described by the realistic scenario where no uncertainty model (stochastic or deterministic) is known but it is assumed that the uncertainty variables satisfy a set of inequalities.

Definitions

Consider the following linear, discrete-time system:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + W\theta_t \\ y_t &= Bx_t + Du_t + F\theta_t \,, \end{aligned} \tag{1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$, $y \in \mathcal{Y} \subset \mathbb{R}^q$ and $\theta \in \Theta \subset \mathbb{R}^w$ are the state, input, output and disturbance (or uncertain) input vectors respectively and A, B, C, D, W and F are matrices of appropriate dimensions. The disturbance input θ is assumed to be bounded in the set $\Theta = \{\theta \in \mathbb{R}^w | \theta_i^L \leq \theta_i \leq \theta_i^U,\ i = 1, \dots, w\}$. This type of uncertainty is used to characterize a broad variety of input disturbances and modeling uncertainties including non-linearities or hidden dynamics [7,11]. This type of uncertainty in general may result in infeasibilities and performance degradation.

Definition 1 The robust controller is defined as the controller that provides a single control sequence that steers the plant into the feasible operating region for a specific range of variations in the uncertain variables.

The general robust parametric MPC (RpMPC) problem is defined as [20]

$$\phi(x_{t|t}) = \min_{u^N \in V^N} \left\{ x_{t+N|t}^T P x_{t+N|t} + \sum_{k=0}^{N-1} \left[y_{t+k|t}^T Q y_{t+k|t} + u_{t+k}^T R u_{t+k} \right] \right\} \tag{2}$$

$$\text{s.t.} \quad x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k} + W\theta_{t+k} \,, \quad k \geq 0 \tag{3}$$

$$y_{t+k|t} = Cx_{t+k|t} + Du_{t+k} + F\theta_{t+k} \,, \quad k \geq 0 \tag{4}$$

$$g(x_{t+k|t}, u_{t+k}) = C_1 x_{t+k|t} + C_2 u_{t+k} + C_3 \leq 0 \,, \quad k = 0, 1, \dots, N-1 \tag{5}$$

$$h(x_{t+N|t}) = D_1 x_{t+N|t} + D_2 \leq 0 \tag{6}$$

$$u_{t+k} = Kx_{t+k|t} \,, \quad k \geq N \tag{7}$$

$$x_{t|t} = x^* \tag{8}$$

where $g : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_g}$ and $h : \mathcal{X} \to \mathbb{R}^{n_h}$ are the path and terminal constraints respectively, x^* is the initial state, $u^N = \{u_t, \dots, u_{t+N-1}\} \in \mathcal{U} \times \dots \times \mathcal{U} = \mathcal{U}^N$ are the predicted future inputs and $\theta^N = \{\theta_t, \dots, \theta_{N-1}\} \in \Theta^N$ are the current and future values of the disturbance.

Formulation

The design of a robust control scheme is obtained by solving a receding horizon constrained optimal control problem where the objective is the deviations expected over the entire uncertainty set, or the nominal value of the output and input deviations. In order to ensure feasibility of (2)–(8) for every possible uncertainty scenario $\theta_{t+k} \in \Theta$, $k = 0, \dots, N-1$, the set of constraints of (2)–(8) is usually augmented with an extra set of feasibility constraints. The type of these constraints, as will be described later, will determine if the RpMPC is an open-loop or closed-loop controller.

Open-Loop Robust Parametric Model Predictive Controller

To define the set of extra feasibility constraints the future state prediction

$$x_{t+k|t} = A^k x^* + \sum_{j=0}^{k-1} (A^j B u_{t+k-1-j} + A^j W \theta_{t+k-1-j}) \tag{9}$$

is substituted into the inequality constraints (5)–(6), which then become

$$\begin{aligned} &\bar{g}_j(x^*, u^N, \theta^N) \le 0\,, \quad j = 1, \dots, J \quad \Leftrightarrow \\ &\sum_{i=1}^{n} \gamma 1_{i,j} x_i^* + \sum_{k=0}^{N-1} \sum_{i=1}^{q} \gamma 2_{i,k,j} u_{t+k,i} \\ &\qquad + \sum_{k=0}^{N-1} \sum_{i=1}^{w} \gamma 3_{i,k,j} \theta + t + k, i + \gamma 4_j \le 0 \end{aligned} \tag{10}$$

where $\gamma 1$, $\gamma 2$, $\gamma 3$ are coefficients that are explicit functions of the elements of matrices A, B, C, D, W, F, C_1, C_2, C_3, D_1, D_2, Q, R, P. The set of feasibility constraints is defined as

$$\begin{aligned} &\psi(x^*, u^N) \le 0 \Leftrightarrow \forall \theta^N \in \Theta^N \left(\forall j = 1, \dots, J \right. \\ &\quad \left. \left[\bar{g}_j(x^*, u^N, \theta^N) \le 0, u^N \in \mathcal{U}^N, x^* \in \mathcal{X} \,.\right] \right) \end{aligned} \tag{11}$$

The constraints $\psi \le 0$ ensure that, given a particular state realization x^*, the single control action u^N satisfies all the constraints for all possible bounded disturbance scenarios over the time horizon. However, this feasibility constraint represents an infinite set of constraints since the inequalities are defined for every possible value of $\theta^N \in \Theta^N$. In order to overcome this problem one has to notice that (11) is equivalent to

$$\begin{aligned} &\max_{\theta^N} \max_{j} \left\{ \bar{g}(x^*, u^N, \theta^N) | \right. \\ &\quad \left. j = 1, \dots, J, u^N \in \mathcal{U}^N, x^* \in \mathcal{X}, \theta^N \in \Theta^N \right\} \le 0 \end{aligned} \tag{12}$$

Adding (12) into (2)–(8) and minimizing the expectation of the objective function (2) over all uncertain realizations θ_{t+k} one obtains the following robust model predictive control problem:

$$\begin{aligned} \phi(x_{t|t}) = \min_{u^N \in V^N} E_{\theta^N \in \Theta^N} &\left\{ x^T_{t+N|t} P x_{t+N|t} + \right. \\ &\left. \sum_{k=0}^{N-1} \left[y^T_{t+k|t} Q y_{t+k|t} + u^T_{t+k} R u_{t+k} \right] \right\} \end{aligned} \tag{13}$$

$$\text{s.t.} \quad (3)\text{–}(8) \quad \text{and} \quad (11) \tag{14}$$

Problem (13)–(14) is a bilevel program that has as constraint a maximization problem, which, as will be shown later, can be solved parametrically and then replaced by a set of linear inequalities of u^N, x^*. The solution to this problem corresponds to a robust control law as it is defined in Definition 1. Problem (13)–(14) is an open-loop robust control formulation in that it obtains the optimal control actions u^N for the worst-case realization of the uncertainty only, as expressed by inequality (12), and does not take into account the information of the past uncertainty values in the future measurements, thus losing the benefit of the prediction property. This implies that the future control actions can be readjusted to compensate for any variation in the past uncertainty realizations, thereby obtaining more "realistic" and less conservative values for the optimal control actions. This problem can be overcome if we consider the following closed-loop formulation of the problem (2)–(8).

Closed-Loop Robust Parametric Model-Based Control

To acquire a closed-loop formulation of the general RpMPC problem, a dynamic programming approach is used to formulate the worst-case closed-loop MPC problem, which requires the solution of a number of embedded optimization problems that in the case of a quadratic objective are non-linear and non-differentiable. Feasibility analysis is used to directly address the problem and a set of constraints is again incorporated in the optimization problem to preserve feasibility and performance for all uncertainty realizations. Future measurements of the state contain information about the past uncertainty values. This implies that the future control actions can be readjusted to compensate for the past disturbance realizations by deriving a closed-loop MPC problem as shown next. The main idea is to introduce constraints into the control optimization problem (2)–(8) that preserve feasibility and performance for all disturbance realizations. These constraints are given as

$$\begin{aligned}
&\psi^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}) \Leftrightarrow \\
&\forall\theta_{t+\ell} \in \Theta\{\exists u_{t+\ell+1} \in \mathcal{U}\{\forall\theta_{t+\ell+1} \\
&\in \Theta\{\exists u_{t+\ell+2} \in \mathcal{U}\ldots\{\forall\theta_{t+N-2} \\
&\in \Theta\{\exists u_{t+N-1} \in \mathcal{U}\{\forall\theta_{t+N-1} \in \Theta \\
&\{\forall j = 1,\ldots,J\left[\bar{g}_j(x^*, [u_{t+k}]_{k=0,\ldots,N-1},\right. \\
&\left.[\theta_{t+k}]_{k=0,\ldots,N-1}) \le 0\right]\}\}\}\ldots\}\}\}\,, \\
&u_{t+k} \in \mathcal{U}\,,\quad k = 0,\ldots,\ell\,,\quad x^* \in \mathcal{X}\,,\quad \theta_{t+k} \in \Theta\,, \\
&\qquad k = 0,\ldots,\ell-1\,,\quad \ell = 0,\ldots,N-1\,. \qquad (15)
\end{aligned}$$

The constraints of (15) are incorporated into (2)–(8) and give rise to a semi-infinite dimensional program that can be posed as a min–max bilevel optimization problem:

$$\phi(x^*) = \min_{u^N \in V^N}\left\{x^T_{t+N|t}Px_{t+N|t} + \sum_{k=0}^{N-1}\left[y^T_{t+k|t}Qy_{t+k|t} + u^T_{t+k}Ru_{t+k}\right]\right\} \qquad (16)$$

$$\text{s.t.} \quad \max_{\theta_{t+N-1,j}} \bar{g}_j(x^*, u^N, \theta^N) \le 0 \qquad (17)$$

$$\vdots$$

$$\max_{\theta_{t+1}}\min_{u_{t+2}}\ldots\max_{\theta_{t+N-2}}\min_{u_{t+N-1}}\max_{\theta_{t+N-1}}\max_{j} \bar{g}_j(x^*, u^N, \theta^N) \le 0 \qquad (18)$$

$$\max_{\theta_t}\min_{u_{t+1}}\max_{\theta_{t+1}}\min_{u_{t+2}}\ldots\max_{\theta_{t+N-2}}\min_{u_{t+N-1}}\max_{\theta_{t+N-1}}\max_{j} \bar{g}_j(x^*, u^N, \theta^N) \le 0 \qquad (19)$$

$$u^N \in \mathcal{U}^N\,,\quad x^* \in \mathcal{X}\,,\quad \theta^N \in \Theta^N\,. \qquad (20)$$

The difference between the above formulation and formulation (13)–(14) is that at every time instant $t+k$ the future control actions $\{u_{t+k+1},\ldots,u_{t+N-1}\}$ are readily adjusted to offset the effect of the past uncertainty $\{\theta_t,\ldots,\theta_{t+k}\}$ to satisfy the constraints. In contrast, in formulation (13)–(14) the control sequence has to ensure constraint satisfaction for all possible disturbance scenarios. The main issue for solving the above optimization problem is how to solve parametrically each of (17)–(19) and replace them with a set of inequalities of u^N, x^* suitable to formulate a multiparametric programming problem. This is shown in the following section.

Methods/Applications

Parametric Solution of the Inner Maximization Problem of the Open-Loop Robust pMPC Problem

An algorithm for solving parametrically the maximization problem of (12), which forms the inner maximization problem of the open-loop RpMPC (13)–(14), comprises the following steps:

Step 1. Solve $G_j(x^*, u^N) = \max_{\theta^N}\{\bar{g}_j(x^*, u^N, \theta^N)|\theta^{N,L} \le \theta^N \le \theta^{N,U}\}, j = 1,\ldots,J$ as a parametric program with respect to θ^N and by recasting the control elements and future states as parameters. The parametric solution can be obtained by following the method in [19], where the critical disturbance points for each maximization are identified as follows:

1. If $\frac{\partial\bar{g}_j}{\partial\theta_{t+k,i}} = \gamma 3_{i,k} > 0 \Rightarrow \theta^{cr}_{t+k,i} = \theta^U_{t+k,i}$, $j = 1,\ldots,J$, then $k = 0,\ldots,N-1$, $i = 1,\ldots,w$;
2. If $\frac{\partial\bar{g}_j}{\partial\theta_{t+k,i}} = \gamma 3_{i,k} < 0 \Rightarrow \theta^{cr}_{t+k,i} = \theta^L_{t+k,i}$, $j = 1,\ldots,J$, then $k = 0,\ldots,N-1$, $i = 1,\ldots,w$.

Substituting $\theta^{cr}_{t+k,i}$ in the constraints $\bar{g} \le 0$ we obtain $G_j(x^*, u^N) = \bar{g}_j(x^*, u^N, \theta^{N,cr})$, where $\theta^{N,cr}$ is the sequence of the critical values of the uncertainty vector θ^{cr}_t over the horizon N.

Step 2. Compare the parametric profiles $G_j(x^*, u^N)$ over the joint space of u^N and x^* and retain the upper bounds. A multiparametric linear program is formulated:

$$\begin{aligned}&\psi(x^*, u^N) = \max_j G_j\\ &\Leftrightarrow \psi(x^*, u^N) = \min_\varepsilon\{\varepsilon | \varepsilon \geq G_j, j = 1, \ldots, J\},\\ &\qquad u^N \in \mathcal{U}^N, \quad x^* \in \mathcal{X}, \qquad (21)\end{aligned}$$

which is equivalent to the comparison procedure of [1].

Step 3. Problem (21) is a multiparametric linear programming problem; hence the solution consists of a set of piece-wise linear expressions for ψ_i in terms of the parameters u^N and x^* and a set of regions Ψ_i, $i = 1, \ldots, \hat{N}_{\text{reg}}$ where these expressions are valid. This statement was proven in [20], sect. 2.2, theorem 2.1, and in [10]. Note that no region Ψ_s exists such that $\psi_s \leq \psi_i$, $\forall\{x^*, u^N\} \in \Psi_s$ and $\forall i \neq s$ since ψ is convex. Thus, inequality (11) can be replaced by the inequalities $\psi_i(x^*, u^N) \leq 0$. In this way problem (13)–(14) can be recast as a single-level stochastic program:

$$\begin{aligned}&\phi(x^*) = \min_{u^N \in \mathcal{U}^N} \{\Phi(x^*, u^N, \theta^{N,n})|\\ &\bar{g}_j(x^*, u^N, \theta^{N,n}) \leq 0, \ j = 1, \ldots, J,\\ &\qquad \psi(x^*, u^N) \leq 0, i = 1, \ldots, \hat{N}_{\text{reg}}\}, \qquad (22)\end{aligned}$$

where Φ is the quadratic objective (13) after substituting (9). The superscript n in $\theta^{N,n}$ denotes the nominal value of θ^N, which is usually zero. An approximate solution to the above stochastic problem can be obtained by discretizing the uncertainty space into a finite set of scenarios $\theta^{N,i}$, $i = 1, \ldots, ns$ with associated objective weights ([20]), thus leading to a multiperiod optimization problem where each period corresponds to a particular uncertainty scenario. By treating the control variables u^N as the optimization variables and the current state x^* as parameters, (22) is recast as multiparametric quadratic program.

Theorem 1 *The solution of (22) is a piece-wise linear control law $u_t(x^*) = \mathcal{A}_c x^* + b_c$ and $CR_c x^* + cr_c$, $c = 1, \ldots, N_c$ is the polyhedral critical region where this control law is valid and guarantees that (5) and (6) are feasible for all $\theta_{t+k} \in \Theta$, $k = 0, \ldots, N-1$.*

The proof of the theorem is straightforward from (21) and [20] and is omitted for brevity's sake. It shows that the solution to (22), and hence (13)–(14), can be obtained as an explicit multiparametric solution [9].

Solution of the Closed-Loop RpMPC Problem

In order to solve the problem (16)–(20), the inner max–min–max problem in (17)–(19) have to be solved parametrically and replaced by simpler linear inequalities, so the resulting problem is a simple multiparametric quadratic program. For simplicity, we only present an algorithm for solving the most difficult problem (19). The same thought process can be performed for the remaining constraints. The algorithm consists of the following steps:

Step 1. Solve

$$\begin{aligned}&G_j^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2})\\ &= \max_{\theta_{t+N-1}} \{\bar{g}_j(x^*, u^N, \theta^N), \theta^{N,L} \leq \theta^N \leq \theta^{N,U}\},\\ &\qquad j = 1, \ldots, J, \qquad (23)\end{aligned}$$

as a multiparametric optimization problem by recasting x^* and u^N as parameters and by following again the method of [19] or [20], sect. 2.2.

Step 2. Compare the parametric profiles $G_j^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2})$ over the joint space of u^N, $[\theta_{t+k}]_{k=0,\ldots,N-2}$ and x^* to retain the upper bounds. For this comparison a multiparametric program is formulated and then solved by following the comparison procedure in [1]:

$$\begin{aligned}&\psi^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2}) = \max_j G_j^{\theta_{t+N-1}}\\ &\Leftrightarrow \psi^{\theta_{t+N-1}}(x^*, u^N, [\theta_{t+k}]_{k=0,\ldots,N-2})\\ &= \min_\varepsilon\{\varepsilon \mid \text{s.t. } G_j^{\theta_{t+N-1}} \leq \varepsilon, \ j = 1, \ldots, J\}. \qquad (24)\end{aligned}$$

The solution of the above optimization consists of a set of linear expressions for $\psi_i^{\theta_{t+N-1}}$ in terms of the parameters x^*, u^N, $[\theta_{t+k}]_{k=0,\ldots,N-2}$ and a set of polyhedral regions $\Psi_i^{\theta_{t+N-1}}$, $i = 1, \ldots, \hat{N}_{\text{reg}}^{\theta_{t+N-1}}$, where these expressions are valid.

Step 3. Set $\ell = N - 1$.

Step 4. Solve the following multiparametric optimization problem over u^ℓ

$$\begin{aligned}&\psi^{u_{t+\ell}}(x^*, u^\ell, \theta^\ell)\\&= \min_{u_{t+\ell}\in\mathcal{U}} \{\psi_i^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}, [\theta_{t+k}]_{k=0,\ldots,\ell-1}),\\&\qquad \text{if } \Psi_i^{\theta_{t+\ell}} \le 0,\ i = 1,\ldots,\hat{N}_{\text{reg}}^{\theta_{t+\ell}}\}\,. \qquad (25)\end{aligned}$$

The above problem can be solved parametrically by following the procedure in [20], appendix A, or [17], chap. 3, sect. 3.2. The solution to (25) is a convex piecewise affine function of $\psi^{u_{t+\ell}}$ in terms of the parameters x^*, u^N, $[\theta_{t+k}]_{k=0,\ldots,N-2}$ that is defined over a set of polyhedral regions $\Psi_i^{u_{t+\ell}}$, $i = 1,\ldots,\hat{N}_{\text{reg}}^{u_{t+\ell}}$.

Step 5. Set $\ell = \ell - 1$ and solve the following maximization problem over $\theta^{\ell-1}$:

$$\begin{aligned}&\psi^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}, [\theta_{t+k}]_{k=0,\ldots,\ell-1})\\&= \max_{\theta_{t+\ell}}\{\psi_i^{\theta_{t+\ell}}(x^*, [u_{t+k}]_{k=0,\ldots,\ell}, [\theta_{t+k}]_{k=0,\ldots,\ell-1}),\\&\qquad \text{if } \Psi_i^{u_{t+\ell+1}} \le 0,\ i = 1,\ldots,\hat{N}_{\text{reg}}^{u_{t+\ell+1}}\}\,. \qquad (26)\end{aligned}$$

Since the function on the left-hand side of the above equality is a convex piecewise affine function, its maximization with respect to $[\theta_{t+k}]_{k=0,\ldots,\ell-1}$ reduces to the method of [19] followed by a comparison procedure as described in step 2.

Step 6. If $\ell > 0$, then go to step 4, else terminate the procedure and store the affine functions $\psi_i^{\theta_t}$, $i = 1,\ldots,\hat{N}_{\text{reg}}^{\theta_t}$.

Step 7. The expressions $\psi_i^{\theta_t}(u_t, x^*)$ are the max–min–max constraint (19). Similarly, the remaining max–min–max constraints are replaced by the set of inequalities

$$\begin{aligned}&\psi_i^{\theta_{t+1}}(x^*, [u_t^T, u_{t+1}^T]^T, \theta_t) \le 0\,,\\&\ldots\,,\\&\psi_i^{\theta_{t+N-2}}(x^*, [u_t^T, u_{t+1}^T, \ldots, u_{t+N-2}^T]^T,\\&\qquad\qquad [\theta_t^T, \theta_{t+1}^T, \ldots, \theta_{t+N-3}^T]^T) \le 0\,,\\&\psi_i^{\theta_{t+N-1}}(x^*, [u_t^T, u_{t+1}^T, \ldots, u_{t+N-1}^T]^T,\\&\qquad\qquad [\theta_t^T, \theta_{t+1}^T, \ldots, \theta_{t+N-2}^T]^T) \le 0\,.\end{aligned}$$

Substituting the inequalities in step 7 into the max–min–max constraints of (16)–(20) we obtain the following stochastic multparametric program:

$$\begin{aligned}\phi(x^*) = &\min_{u^N\in\mathcal{U}^N} E_{\theta^N\in\Theta^N}\left\{\Phi(x^*, u^N, \theta^{N,n})\right\}\\ \text{s.t. } &\bar{g}_j(x^*, u^N, \theta^{N,n}) \le 0\\&\psi_i^{\theta_t}(x^*, u_t) \le 0,\ i = 1,\ldots,\hat{N}_{\text{reg}}^{\theta_0}\\&\psi_i^{\theta_{t+1}}(x^*, [u_t^T, u_{t+1}^T]^T, \theta_t^n),\ i = 1,\ldots,\hat{N}_{\text{reg}}^{\theta_1}\\&\vdots\\&\psi_i^{\theta_{t+N-2}}(x^*, [u_{t+k}]_{k=0,\ldots,N-2}, [\theta_{t+k}^n]_{k=0,\ldots,N-3}),\\&\qquad\qquad i = 1,\ldots,\hat{N}_{\text{reg}}^{\theta_{t+N-2}}\\&\psi_i^{\theta_{t+N-1}}(x^*, [u_{t+k}]_{k=0,\ldots,N-1}, [\theta_{t+k}^n]_{k=0,\ldots,N-2}),\\&\qquad\qquad i = 1,\ldots,\hat{N}_{\text{reg}}^{\theta_{t+N-1}}\\&x_{t|t} = x^*,\ j = 1,\ldots,J\,,\end{aligned} \qquad (27)$$

where Φ is again the quadratic objective function in (16). By discretizing the expectation of the value function to a set of discrete uncertainty scenarios and by treating the current state x^* as parameter and the control actions as optimization variables and the problem is recast as a parametric quadratic program. The solution is a complete map of the control variables in terms of the current state. The results for the closed-loop RpMPC controller are summarized in the following theorem.

Theorem 2 *The solution of (27) is obtained as a linear piecewise control law $u_t(x^*) = \mathcal{A}_c x^* + b_c$ and a set of polyhedral regions $\mathcal{CR}_c = \{x^* \in \mathcal{X} | CR_c x^* + cr_c \le 0\}$ in the state space for which system (1) satisfies constraints (5)–(6) for all $\theta^N \in \Theta^N$.*

Cases

A special case of the RpMPC problem (2)–(8) arises when the system matrices in the first equation in (1) are uncertain in that their entries are unknown but bounded within specific bounds. For simplicity we will consider the simpler case where $W, F = 0$ and the entries a_{ij} and b_{ij} of matrices A and B are not known but satisfy

$$\begin{aligned}&a_{ij} = \bar{a}_{ij} + \delta a_{ij},\ b_{i\ell} = \bar{b}_{i\ell} + \delta b_{i\ell}\\&\delta a_{ij} \in \mathcal{A}_{ij} = \{\delta a_{ij} \in \mathbb{R} | -\varepsilon|\bar{a}_{ij}| \le \delta a_{ij} \le \varepsilon|\bar{a}_{ij}|\}\,,\end{aligned} \qquad (28)$$

$$\delta b_{i\ell} \in \mathcal{B}_{i\ell} = \{\delta b_{i\ell} \in \mathbb{R} | -\varepsilon|\bar{b}_{i\ell}| \leq \delta b_{i\ell} \leq \varepsilon|\bar{b}_{i\ell}|\} , \quad (29)$$

where $\bar{a}_{ij}$, $\bar{b}_{i\ell}$ are the nominal values of the entries of A, B respectively and δa_{ij}, $\delta b_{i\ell}$ denote the uncertainty in the matrix entries, which is assumed to be bounded as in (28)–(29). The general RpMPC formulation (2)–(8) must be redefined to include the introduced model uncertainty by adding the extra constraints

$$\begin{aligned} &a_{ij} = \bar{a}_{ij} + \delta a_{ij}, b_{i\ell} = \bar{b}_{i\ell} + \delta b_{i\ell} \\ &\forall \delta a_{ij} \in \mathcal{A}_{ij}, \forall \delta b_{i\ell} \in \mathcal{B}_{i\ell} , \end{aligned} \quad (30)$$

The new formulation of the RpMPC (2)–(8) and (30) gives rise to a semi-infinite dimensional problem with a rather high computational complexity.

Definition 2 A feasible solution u^N for problem (2)–(8) and (30), for a given initial state x^*, is called a *robust* or *reliable* solution.

Obviously, a robust solution for a given x^* is a control sequence u^N (future prediction vector) for which constraints (5)–(6) are satisfied for all admissible values of the uncertainty. Since it is difficult to solve this MPC formulation by the known parametric optimization methods, the problem must be reformulated in a multi-parametric quadratic programming (mpQP) form. Our objective in this section is to obtain such a form by considering the worst-case values of the uncertainty, i. e. those values of the uncertain parameters for which the linear inequalities of (5)–(6) are critically satisfied. Usually, the objective function (2) is formulated to penalize the nominal system behavior; thus one must substitute $x_{t+k|t} = \bar{A}^k x^* + \sum_{j=0}^{k-1} \bar{A}^j \bar{B} u_{t+k-1-j}$ in (2). In this way the objective function is a quadratic function of u^N and x^*. Finally, the uncertain evolution of the system $x_{t+k|t} = A^k x^* + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j}$ is replaced in the constraints (5)–(6) to formulate a set of linear inequalities. Thus the following formulation of the RpMPC is obtained:

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2}(u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2}(x^*)^T Y x^* \right\} , \quad (31)$$

$$\begin{aligned} \text{s.t. } & C_{1i}^T A^k x^* + \sum_{j=0}^{k-1} C_{1i}^T A^j B u_{t+k-1-j} \\ & + C_{2i}^T u_{t+k} + C_{3i} \leq 0 , \\ & k = 1, \ldots, N-1 , \quad i = 1, \ldots, n_g , \end{aligned} \quad (32)$$

$$D_{1\ell}^T A^N x^* + \sum_{j=0}^{N-1} D_{1\ell}^T A^j B u_{t+k-1-j} + D_{2\ell} \leq 0 , \quad \ell = 1, \ldots, n_h , \quad (33)$$

$$\forall \delta a_{ij} \in \mathcal{A}_{ij} , \quad \forall \delta b_{i\ell} \in \mathcal{B}_{i\ell} , \quad i, j = 1, \ldots, n , \quad \ell = 1, \ldots, m . \quad (34)$$

It is evident that the new formulation of the RpMPC problem (31)–(34) is also a semi-infinite dimensional problem. This formulation can be further simplified if one considers that for any uncertain matrices A and B, the entries of the matrices A^k and $A^k B$ for all $k \geq 0$ are given respectively by [17]

$$a_{i\ell}^k = \bar{a}_{i\ell}^k + \delta a_{i\ell}^k, -\epsilon|\delta a_{i\ell,\min}^k| \leq \delta a_{i\ell}^k \leq \epsilon|\delta a_{i\ell,\max}^k| , \quad (35)$$

$$\begin{aligned} ab_{i\ell}^k &= \bar{ab}_{i\ell}^k + \delta ab_{i\ell}^k, \\ &-\epsilon|\delta ab_{i\ell,\min}^k| \leq \delta ab_{i\ell}^k \leq \epsilon|\delta ab_{i\ell,\max}^k| . \end{aligned} \quad (36)$$

The analysis on (35)–(36) follows from [17], chap. 3, and is omitted for brevity's sake.

Robust Counterpart (RC) Problem

Using the basic properties of matrix multiplication and (35)–(36), problem (31)–(34) reformulates into

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2}(u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2}(x^*)^T Y x^* \right\} , \quad (37)$$

$$\begin{aligned} &\sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} C_{1iq} ab_{q\ell}^j u_{t+k-1-j,\ell} \\ &+ \sum_{\ell} C_{2i\ell} u_{t+k,\ell} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} C_{1iq} a_{q\ell}^k x_\ell^* + C_{3i} \leq 0 , \end{aligned} \quad (38)$$

$$k = 1, \ldots, N-1,\ i = 1, \ldots, n_g$$

$$\sum_{j=1}^{N-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} D_{1iq} ab^j_{q\ell} u_{t+k-1-j,\ell} + \sum_{q=1}^{n} \sum_{\ell=1}^{n} D_{1iq} a^k_{q\ell} x^*_\ell + D_{2i} \le 0 \quad (39)$$

$$i = 1, \ldots, n_h \forall \delta a_{ij} \in \mathcal{A}_{ij},\quad \forall \delta b_{i\ell} \in \mathcal{B}_{i\ell}, i, j = 1, \ldots, n, \ell = 1, \ldots, m\,. \quad (40)$$

This is a robust multiparametric QP problem (robust mp-QP) where the coefficients of the linear inequalities in the constraints are uncertain, the vector u^N is the optimization variable and the initial states x^* are the parameters. A similar robust LP problem was studied in [5] where the coefficients of the linear constraints are uncertain, similar to (35)–(36); however, no multiparametric programming problems were considered.

In a similar fashion to the analysis in [5] we construct the *robust counterpart* of the robust mp-QP problem (37)–(40):

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2}(u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2}(x^*)^T Y x^* \right\}, \quad (41)$$

s.t.

$$\sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} C_{1iq} \bar{ab}^j_{q\ell} u_{t+k-1-j,\ell} + \sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} \epsilon \max\{|C_{1iq}||\delta ab^k_{q\ell,\min}|, |C_{1iq}||\delta a^k_{q\ell,\max}|\}|u_{t+k-1-j,\ell}| + \sum_{\ell} C_{2i\ell} u_{t+k,\ell}$$
$$+ \sum_{q=1}^{n} \sum_{\ell=1}^{n} C_{1iq} \bar{a}^k_{q\ell} x^*_\ell + \sum_{q=1}^{n} \sum_{\ell=1}^{n} \epsilon \max\{|C_{1iq}||\delta a^k_{q\ell,\min}|, |C_{1iq}||\delta a^k_{q\ell,\max}|\}|x^*_\ell| + C_{3i} \le 0$$
$$k = 1, \ldots, N-1\,,\quad i = 1, \ldots, n_g\,, \quad (42)$$

$$\sum_{j=1}^{N-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} D_{1iq} \bar{ab}^j_{q\ell} u_{t+k-1-j,\ell} + \sum_{j=1}^{N-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} \max\{|D_{1iq}||\delta ab^k_{q\ell,\min}|, |D_{1iq}||\delta ab^k_{q\ell,\max}|\}|u_{t+k-1-j,\ell}|$$
$$+ \sum_{q=1}^{n} \sum_{\ell=1}^{n} D_{1iq} \bar{a}^k_{q\ell} x^*_\ell + \sum_{q=1}^{n} \sum_{\ell=1}^{n} \max\{|D_{1iq}||\delta a^k_{q\ell,\min}|, |D_{1iq}||\delta a^k_{q\ell,\max}|\}|x^*_\ell| + D_{2i} \le 0$$
$$i = 1, \ldots, n_h\,, \quad (43)$$

$$u^N \in \mathcal{U}^N\,,\quad x^* \in \mathcal{X}\,. \quad (44)$$

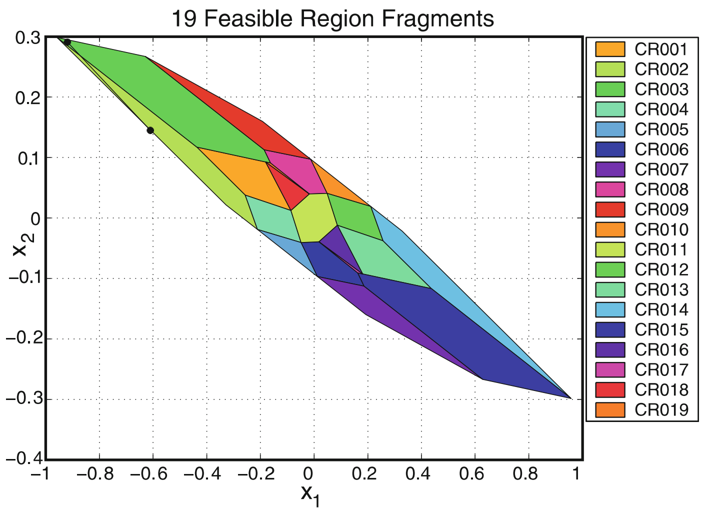
In this way the initial semi-infinite dimensional problem (37)–(40) becomes the above multiparametric non-linear program (mp-NLP). However, the parametric solution of this mp-NLP problem is still very difficult.

Interval Robust Counterpart Problem

The interval robust counterpart (IRC) problem can then be formulated as follows:

$$\phi(x^*) = \min_{u^N \in U^N} \left\{ \frac{1}{2}(u^N)^T H u^N + x^{*T} F u^N + \frac{1}{2}(x^*)^T Y x^* \right\}, \quad (45)$$

$$\text{s.t.} \quad \sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} C_{1iq} \bar{ab}^j_{q\ell} u_{t+k-1-j,\ell} + \sum_{j=1}^{k-1} \sum_{q=1}^{n} \sum_{\ell=1}^{m} \epsilon \max\{|C_{1iq}||\delta ab^k_{q\ell,\min}|, |C_{1iq}||\delta a^k_{q\ell,\max}|\} z_{t+k-1-j,\ell} + \sum_{\ell} C_{2i\ell} u_{t+k,\ell}$$
$$+ \sum_{q=1}^{n} \sum_{\ell=1}^{n} C_{1iq} \bar{a}^k_{q\ell} x^*_\ell + \sum_{q=1}^{n} \sum_{\ell=1}^{n} \epsilon \max\{|C_{1iq}||\delta a^k_{q\ell,\min}|, |C_{1iq}||\delta a^k_{q\ell,\max}|\} w_\ell + C_{3i} \le 0$$
$$k = 1, \ldots, N-1\,,\quad i = 1, \ldots, n_g\,, \quad (46)$$



Design of Robust Model-Based Controllers via Parametric Programming, Figure 1
Critical regions for the nominal parametric MPC and state trajectory

$$\sum_{j=1}^{N-1}\sum_{q=1}^{n}\sum_{\ell=1}^{m} D_{1iq}\bar{ab}^{j}_{q\ell}u_{t+k-1-j,\ell}$$
$$+\sum_{j=1}^{N-1}\sum_{q=1}^{n}\sum_{\ell=1}^{m}\max\{|D_{1iq}||\delta ab^{k}_{q\ell,\min}|,$$
$$|D_{1iq}||\delta ab^{k}_{q\ell,\max}|\}z_{t+k-1-j,\ell}$$
$$+\sum_{q=1}^{n}\sum_{\ell=1}^{n}D_{1iq}\bar{a}^{k}_{q\ell}x^{*}_{\ell}+\sum_{q=1}^{n}\sum_{\ell=1}^{n}\max\{|D_{1iq}||\delta a^{k}_{q\ell,\min}|,$$
$$|D_{1iq}||\delta a^{k}_{q\ell,\max}|\}w_{\ell}+D_{2i}\le 0$$
$$i=1,\dots,n_h\,, \qquad (47)$$

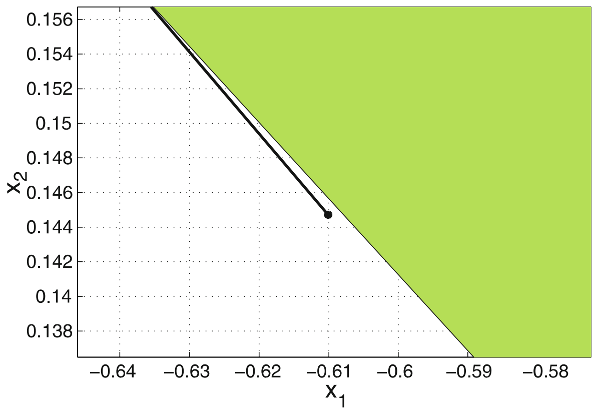
$$-z_{t+k-1-j,\ell}\le u_{t+k-1-j,\ell}\le z_{t+k-1-j,\ell}\,, \qquad (48)$$

$$-w_{\ell}\le x^{*}_{\ell}\le w_{\ell}\,, \qquad (49)$$

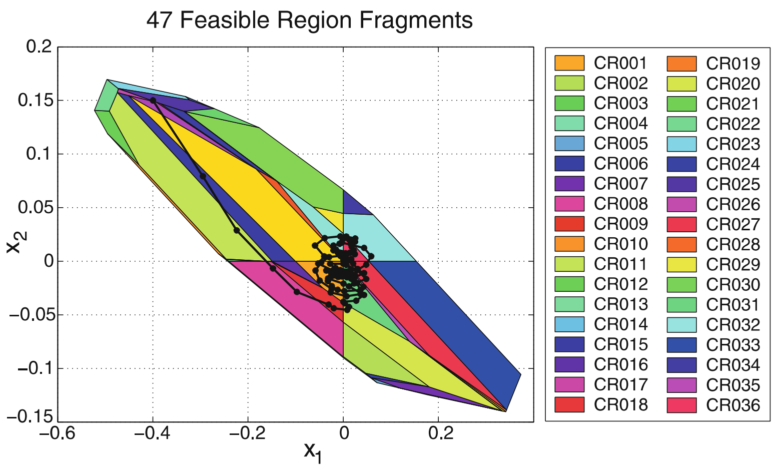
$$u^{N}\in\mathcal{U}^{N}\,,\quad x^{*}\in\mathcal{X}\,, \qquad (50)$$

where the non-linear inequalities (42)–(43) have been replaced by four new linear inequalities. Two new variables have been introduced to replace the absolute values of the $u_{t+k-1-j,\ell}$ and x^{*}_{ℓ}, thus leading to the relaxed IRC problem.

The IRC is a mpQP problem with a quadratic index and linear inequalities, where the optimization variables now are the vectors $u_{t+k-1-j}$, $z_{t+k-1-j}$ and w and the parameters are the states x^{*}. The IRC problem can be solved with the known parametric optimization methods [4,9,16] since the objective function is strictly convex by assumption. The optimal control inputs u^{N}, optimization variables z and w and hence the optimal control u_t can then be obtained as explicit functions $u^{N}(x^{*})$, $z(x^{*})$ and $w(x^{*})$ of the initial state x^{*}. Furthermore, the control input u_t is obtained as the explicit, optimal control



Design of Robust Model-Based Controllers via Parametric Programming, Figure 2
Magnification of Fig. 1 around the state trajectory at the second time instant



Design of Robust Model-Based Controllers via Parametric Programming, Figure 3
Critical regions for the nominal parametric MPC and state trajectory

law [9] $u_t(x^*) = \mathcal{A}_c x^* + b_c$ which is valid in the polyhedral region $CR_c = \{x^* \in \mathcal{X} | CR_c x^* + cr \leq 0\}$, $c = 1, \ldots, N_c$, where N_c is the number of critical regions obtained from the parametric programming algorithm.

The general RpMPC problem obtained from the case where the dynamic system (1) pertains to model uncertainties have now been transformed into the IRC problem and can be solved as a mp-QP problem. It is obvious that a feasible solution for the IRC problem is also a feasible solution for the RC and hence the initial RpMPC problem (2)–(8) and (30). Hence:

Lemma 1 *If u^N is a feasible solution for the IRC problem, then it is also a feasible solution for the RC problem, and hence it is a robust solution for the initial RpMPC problem (2)–(8), (30).*

Example 2 Consider a two-dimensional, discrete-time linear system (1) where $W = F = 0$ and

$$A = \begin{bmatrix} 0.7326 + \delta a & -0.0861 \\ 0.1722 & 0.0064 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0609 + \delta b \\ 0.0064 \end{bmatrix}, \tag{51}$$

where the entries a_{11} and b_1 of the A and B matrices are uncertain, where δa and δb are bounded as in (28)–(29) with $\epsilon = 10\%$ and the nominal values are $\bar{a}_{11} = 0.7326$ and $\bar{b}_1 = 0.0609$. The state and control constraints are $-3 \leq [0 \;\; 1.4142]^T x \leq 3$, $-2 \leq u \leq 2$, and the terminal constraint is

$$\begin{bmatrix} 0.070251 & 1 \\ -0.070251 & -1 \\ 0.21863 & 1 \\ -0.21863 & -1 \end{bmatrix} x \leq \begin{bmatrix} 0.02743 \\ 0.02743 \\ 0.022154 \\ 0.022154 \end{bmatrix}. \tag{52}$$

Moreover,

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} R = 0.01, \quad P = \begin{bmatrix} 1.8588 & 1.2899 \\ 1.2899 & 6.7864 \end{bmatrix}. \tag{53}$$

Initially, the MPC problem (2)–(8) is formulated and solved only for the nominal values of A and B, thus solving a multiparametric quadratic programming problem as described in [4,16]. Then the IRC problem is formulated as in (45)–(50) by using POP software [9]. The resulting regions for both cases are shown in Figs. 1 and 3 respectively. A simulation of the state trajectories of the nominal and the uncertain system are shown in Figs. 1 and 3 respectively. In these simulations the uncertain parameters δa and δb were simulated as a sequence of random numbers that take their values on the upper or lower bounds of δa, δb i. e. a time-varying uncertainty. It is clear from Fig. 1 (and Fig. 2, which displays the magnified area around the state trajectory at the second time instant) that the nominal solution to problem (2)–(8) cannot guarantee robustness in the presence of the uncertainty and the nominal system trajectory results in constraint violation. On the other hand, the controller obtained with the method discussed here manages to retain the trajectory in the set of feasible initial states (obtained by the critical regions of the parametric solution) and drives the trajectory close to the origin. One should notice that the space of feasible initial states (Fig. 3) given by the critical regions of the parametric solution is smaller than the one given in the nominal system's case (Fig. 1).

Conclusions

In this chapter two robust parametric MPC problems were analyzed. In the first problem two methods for robust parametric MPC are discussed, an open-loop and a closed-loop method, for treating robustness issues arising from the presence of input disturbances/uncertainties. In the second problem, a robust parametric MPC procedure was discussed for the control of dynamic systems with uncertainty in the system matrices by employing robust parametric optimization methods.

References

1. Acevedo J, Pistikopoulos EN (1999) An algorithm for multiparametric mixed-integer linear programming problems. Oper Res Lett 24:139–148
2. Bemporad A, Borelli F, Morari M (2003) Min–max control of constrained uncertain discrete-time linear systems. IEEE Trans Autom Contr 48(9):1600–1606
3. Bemporad A, Morari M (1999) Robust model predictive control: a survey Robustness in indentification and control. Springer, Berlin, pp 207–226
4. Bemporad A, Morari M, Dua V, Pistikopoulos EN (2002) The explicit linear quadratic regulator for constrained systems. Automatica 38:3–20
5. Ben-Tal A, Nemirovski A (2000) Robust solutions of linear programming problems contaminated with uncertain data. Math Program 88:411–424
6. Bertsekas DP, Rhodes IB (1971) On the minimax reachability of target sets and target tubes. Automatica 7:233–247
7. Camacho E, Bordons C (1999) Model Predictive Control. Springer, Berlin
8. Chisci L, Rossiter JA, Zappa G (2001) Systems with persistent disturbances: predictive control with restricted constraints. Automatica 37:1019–1028
9. Dua V, Bozinis NA, Pistikopoulos EN (2002) A multiparametric programming approach for mixed integer and quadratic engineering problems. Comput Chem Eng 26:715–733
10. Fiacco A (1983) Introduction to sensitivity and stability analysis in nonlinear programming. Academic, New York
11. Kothare MV, Balakrishnan V, Morari M (1996) Robust constrained model predictive control using linear matrix inequalities. Automatica 32(10):1361–1379
12. Langson W, Chryssochoos I, Raković SV, Mayne DQ (2004) Robust model predictive control using tubes. Automatica 40:125–133
13. Lee J, Cooley B (1997) Recent advances in model predictive control and other related areas. In: Carnahan B, Kantor J, Garcia C (eds) Proceedings of chemical process control – V: Assesment and new directions for research, vol 93 of AIChE Symposium Series No. 316, AIChE and CACHE, pp 201–216
14. Mayne D, Rawlings J, Rao C, Scokaert PO (2000) Constrained model predictive control: stability and optimality. Automatica 36:789–814
15. Morari M, Lee J (1999) Model predictive control: past, present and future. Comput Chem Eng 23:667–682
16. Pistikopoulos EN, Dua V, Bozinis NA, Bemporad A, Morari M (2002) On-line optimization via off-line parametric optimization tools. Automatica 26:175–185
17. Pistikopoulos EN, Georgiadis M, Dua V (eds) (2007) Multiparametric Model-Based Control. In: Process Systems Engineering, vol 2. Wiley-VCH, Weinheim
18. Pistikopoulos EN, Georgiadis M, Dua V (eds) (2007) Multiparametric Programming. In: Process Systems Engineering, vol 1. Wiley-VCH, Weinheim
19. Pistikopoulos EN, Grossmann I (1988) Optimal retrofit design for improving process flexibility in linear systems. Comput Chem Eng 12(7):719–731
20. Sakizlis V, Kakalis NMP, Dua V, Perkins JD, Pistikopoulos EN (2004) Design of robust model-based controllers via parametric programming. Automatica 40:189–201
21. Scokaert P, Mayne D (1998) Min–max feedback model predictive control for constrained linear systems. IEEE Trans Autom Contr 43(8):1136–1142
22. Wang YJ, Rawlings JB (2004) A new robust model predictive control method I: theory and computation. J Process Control 14:231–247

Determining the Optimal Number of Clusters

MENG PIAO TAN,
CHRISTODOULOS A. FLOUDAS
Department of Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 90C26, 91C20, 68T20, 68W10, 90C11, 92-08, 92C05, 92D10

Article Outline

Introduction

Clustering is probably the most important unsupervised learning problem and involves finding coherent structures within a collection of unlabeled data. As such it gives rise to data groupings so that the patterns are similar within each group and remote between different groups. Besides having been extensively applied in areas such as image processing and pattern recognition, clustering also sees rich applications in biology, market research, social network analysis, and geology. For instance, in marketing and finance, cluster analysis is used to segment and determine target markets, position new products, and identify clients in a banking database having a heavy real estate asset base. In libraries, clustering is used to aid in book ordering and in insurance, clustering helps to identify groups of motor insurance policy holders with high average claim costs. Given its broad utility, it is unsurprising that a substantial number of clustering methods and approaches have been proposed.

On the other hand, fewer solutions to systematically evaluate the quality or validity of clusters have been presented [1]. Indeed, the prediction of the optimal number of groupings for any clustering algorithm remains a fundamental problem in unsupervised classification. To address this issue, numerous cluster indices have been proposed to assess the quality and the results of cluster analysis. These criteria may then be used to compare the adequacy of clustering algorithms and different dissimilarity measures, or to choose the optimal number of clusters. Some of these measures are introduced in the following section.

Methods

Dunn's Validity Index

This technique [2,5] is based on the idea of identifying the cluster sets that are compact and well separated. For any partition of clusters, where c_i represent the ith cluster of such a partition, Dunn's validation index, D, can be calculated as

$$D = \min_{1\le i\le n}\left\{\min_{\substack{1\le j\le n\\ i\ne j}}\left\{\frac{d(c_1,c_j)}{\max_{1\le k\le n}} d'(c_k)\right\}\right\}.$$

Here, $d(c_\mathrm{i},c_\mathrm{j})$ is the distance between clusters c_i, and c_j (intercluster distance), $d'(c_\mathrm{k})$ is the intracluster distance of cluster c_k, and n is the number of clusters. The goal of this measure is to maximize the intercluster distances and minimize the intracluster distances. Therefore, the number of cluster that maximizes D is taken as the optimal number of clusters to be used.

Davies–Bouldin Validity Index

This index [4] is a function of the ratio of the sum of within-cluster scatter to between-cluster separation:

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{i\ne j}\left\{\frac{S_n(Q_i)+S_n(Q_j)}{S(Q_i,Q_j)}\right\}.$$

In this expression, DB is the Davies–Bouldin index, n is the number of clusters, S_n is the average distance of all objects from the cluster to their cluster center, and $S(Q_iQ_j)$ is the distance between cluster centers. Hence, the ratio is small if the clusters are compact and far from each other. Consequently, the Davies–Bouldin index will have a small value for a good clustering.

The silhouette validation technique [22] calculates the silhouette width for each sample, the average silhouette width for each cluster, and the overall average silhouette width for a total data set. With use of this approach each cluster can be represented by a so-called silhouette, which is based on the comparison of its tightness and separation. The average silhouette width can be applied for the evaluation of clustering validity and can also be used to decide how good are the number of selected clusters. To construct the silhouettes S(i) the following formula is used:

$$S(i) = \frac{(b(i) - a(i))}{\max\{a(i), b(i)\}} .$$

Here, $a(i)$ is the average dissimilarity of the ith object to all other objects in the same cluster and $b(i)$ is the minimum average dissimilarity of the ith object to all objects in the other clusters.

It follows from the formula that s(i) lies between -1 and 1. If the silhouette value is close to 1, it means that sample is "well clustered" and has been assigned to a very appropriate cluster. If the silhouette value is close to 0, it means that that sample could be assigned to another "closest" cluster as well, and the sample lies equally far away from both clusters. If the silhouette value is close to -1, it means that sample is "misclassified" and is merely somewhere in-between the clusters. The overall average silhouette width for the entire plot is simply the average of the $S(i)$ for all objects in the whole dataset and the largest overall average silhouette indicates the best clustering (number of clusters). Therefore, the number of clusters with the maximum overall average silhouette width is taken as the optimal number of the clusters.

Measure of Krzanowski and Lai

This index is based on the decrease of the within-cluster sum of squares (WSS) [15] and is given by

$$\mathrm{KL}(k) = \left|\frac{\mathrm{DIFF}(k)}{\mathrm{DIFF}(k+1)}\right|, \quad \text{where}$$
$$\mathrm{DIFF}(k) = (k-1)^{\frac{2}{p}}\,\mathrm{WSS}(k-1) - k^{\frac{2}{p}}\,\mathrm{WSS}(k) .$$

Assuming that g is the ideal cluster number for a given dataset, and k is a particular number of clusters, then WSS(k) is assumed to decrease rapidly for $k \leq g$ and decreases only slightly for $k > g$. Thus, it is expected that KL(k) will be maximized for the optimal number of clusters.

Measure of Calinski and Harabasz

This method [3] assesses the quality of k clusters via the index

$$\mathrm{CH}(k) = \frac{\mathrm{BSS}(k-1)/(k-1)}{\mathrm{WSS}(k)/(n-k)} .$$

Here, WSS(k) and BSS(k) are the WSS and the between-cluster sums of squares, for a dataset of n members. The measure seeks to choose clusters that are well isolated from one another and coherent, but at the same time keep the number of clusters as small as possible, thus maximizing the criterion at the optimal cluster number. Incidentally, a separate study comparing 28 validation criteria [18] found this measure to perform the best.

In addition, some other measures to determine the optimal number of clusters are (i) the C index [10], (ii) the Goodman–Kruskal index [8]), (iii) the isolation index [19], (iv) the Jaccard index [11], and (v) the Rand index [20].

Applications

As can be seen, while it is relatively easy to propose indices of cluster validity, it is difficult to incorporate these measures into clustering algorithms and to appoint suitable thresholds on which to define key decision values [9,12]. Most clustering algorithms do not contain built-in screening functions to determine the optimal number of clusters. This implies that for a given clustering algorithm, the most typical means of determining the optimal cluster number is to repeat the clustering numerous times, each with a different number of groupings, and hope to catch a maximum or minimum turning point for the cluster validity index in play.

Nonetheless, there have been attempts to incorporate measures of cluster validity into clustering algorithms. One such method [21] introduces a validity index:

$$\mathit{Validity} = \frac{\mathit{Intra-Cluster}}{\mathit{Inter-Cluster}} .$$

Since it is desirable for the intracluster distance and the intercluster distance to be minimized and maximized, respectively, the above validity measure should be as small as possible. Using the K-means algorithm, Ray and Turi [21] proposed running the process for two up to a predetermined maximum number of clusters. At each stage, the cluster with the maximum variance is split into two and clustering is repeated with these updated centers, until the desired turning point for the validity measure is observed. Another approach [16] is based on simulated annealing, which was originally formulated to simulate a collection of atoms in equilibrium at a given temperature [14,17]. It assumes two given parameters *D*, which is the cutoff cluster diameter, and *P*, a *P*-value statistic, as well as *p(d)*, the distribution function of the Euclidean distances between the members in a dataset. Then, the upper boundary for the fraction of incorrect vector pairs is given by

$$f(D, K = 1) = \int_D^\infty p(x)\,\mathrm{d}x\ .$$

On the other hand, it is possible to define a lower boundary for *f(D,K)* with a preassigned *P*-value cutoff. The clustering algorithm then sequentially increases the cluster number until the two indicators converge.

A Novel Clustering Approach with Optimal Cluster Determination

See also the article on "Gene Clustering: A Novel Optimization-Based Approach".

Recently, we proposed a novel clustering approach [23,24] that expeditiously contains a method to predict the optimal cluster number. The clustering seeks to minimize the Euclidean distances between the data and the assigned cluster centers as

$$\underset{w_{ij},z_{jk}}{\mathrm{MIN}} \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}\left(a_{ik}-z_{jk}\right)^2\ .$$

To make the nonlinear problem tractable, we apply a variant of the generalized benders decomposition algorithm [6,7], the global optimum search. The global optimum search decomposes the problem into a primal problem and the master problem. The former solves the continuous variables while fixing the integer variables and provides an upper-bound solution, while the latter finds the integer variables and the associated Lagrange multipliers while fixing the continuous variables and provides a lower-bound solution. The two sequences are iteratively updated until they converge at an optimal solution in a finite number of steps.

In determining the optimal cluster number, we note that the optimal number of clusters occurs when the intercluster distance is maximized and the intracluster distance is minimized. We adapt the novel work of Jung et al. [13] in defining a clustering balance, which has been shown to have a minimum value when intracluster similarity is maximized and intercluster similarity is minimized. This provides a measure of how optimal is a certain number of clusters used for a particular clustering algorithm. Given *n* data points, each having *k* feature points, *j* clusters, and a binary decision variable for cluster membership w_{ij}, we introduce the following:

$$\text{Global center, } z_k^o = \frac{1}{n}\sum_{i=1}^{n} a_{ik}, \quad \forall k\,,$$

Intracluster error sum,

$$\Lambda = \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}\left\|a_{ik}-z_{jk}\right\|_2^2\,,$$

$$\text{Intercluster error sum, } \Gamma = \sum_{j=1}^{c}\sum_{k=1}^{s}\left\|z_{jk}-z_k^o\right\|_2^2\,.$$

Jung et al. [13] next proposed a clustering balance parameter, which is the α-weighted sum of the two error sums:

$$\text{Clustering balance,} \quad \varepsilon = \alpha\Lambda + (1-\alpha)\,\Gamma\ .$$

We note here that the right α ratio is 0.5. There are two ways to come to this conclusion. We note that the factor α should balance the contributive weights of the two error sums to the clustering balance. At extreme cluster numbers, that is, the largest and smallest numbers possible, the sum of the intracluster and intercluster error sums at both cluster numbers should be balanced. In the minimal case, all the data points can be placed into a single cluster, in which case the intercluster error sum is zero and the intracluster error sum can be calculated with ease. In the maximal case, each data point forms its own cluster, in which case the intracluster error sum is zero and the intercluster error sum can be easily found. Obviously the intracluster error sum in the minimal case and the intercluster error sum in the maximal case are equal, suggesting that the most appropriate weighting factor to use is in fact 0.5. The second approach uses a clustering gain parameter proposed by Jung et al. [13]. This gain parameter is the difference between the decreased intercluster error sum γ_j compared with the value at the initial stage and the increased intracluster error sum λ_j compared with the value at the initial stage, and is given by

$$\gamma_{jk} = \sum_{i=1}^{n} w_{ij}\left\|a_{ik}-z_k^o\right\|_2^2 - \left\|z_{jk}-z_k^o\right\|_2^2\,, \quad \forall j,\forall k\,,$$

$$\lambda_{jk} = \sum_{i=1}^{n} w_{ij}\left\|a_{ik}-z_{jk}\right\|_2^2, \ \forall j,\forall k\,,$$

$$Gain,\ \Delta_{jk} = \sum_{i=1}^{n} w_{ij} \left\| a_{ik} - z_k^o \right\|_2^2 - \left\| z_{jk} - z_k^o \right\|_2^2 - \sum_{i=1}^{n} w_{ij} \left\| a_{ij} - z_{jk} \right\|_2^2,\ \forall \mathrm{j}, \forall \mathrm{k}\,.$$

With the identities

$$\sum_{i=1}^{n} w_{ij} a_{ik} = n_j z_{jk},\ \forall j,\ \forall k\,,$$
$$\sum_{i=1}^{n} w_{ij} = n_j,\ \forall j\,,$$

where $\mathrm{n_j}$ denotes the number of data points in cluster *j*, the gain can be simplified to

$$\Delta_{\mathrm{jk}} = \left(n_j - 1\right) \left\| z_k^o - z_{jk} \right\|_2^2, \forall \mathrm{j}, \forall \mathrm{k}\,,$$
$$\Delta = \sum_{\mathrm{j}=1}^{\mathrm{c}} \sum_{k=1}^{s} \left(n_j - 1\right) \left\| z_k^o - z_{jk} \right\|_2^2 .$$

Jung et al. [13] showed the clustering gain to have a maximum value at the optimal number of clusters, and demonstrated that the sum total of the clustering gain and balance parameters is a constant. As can be seen from the following derivation, this is only possible if the α ratio is 0.5:

Sum of clustering balance and clustering gain, Ω

$$\begin{aligned}
&= \varepsilon + \Delta \\
&= \Lambda + \Gamma + \Delta \\
&= \left[\sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_{jk} \right\|_2^2 \right] \\
&\quad + \left[\sum_{j=1}^{c} \left\| z_{jk} - z_k^o \right\|_2^2 \right] + \ldots \\
&\quad \left[\begin{array}{l} \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_k^o \right\|_2^2 - \sum_{j=1}^{c} \left\| z_{jk} - z_k^o \right\|_2^2 \\ - \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_{jk} \right\|_2^2 \end{array} \right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} w_{ij} \left\| a_{ik} - z_k^o \right\|_2^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{s} \left\| a_{ik} - z_k^o \right\|_2^2 ,
\end{aligned}$$

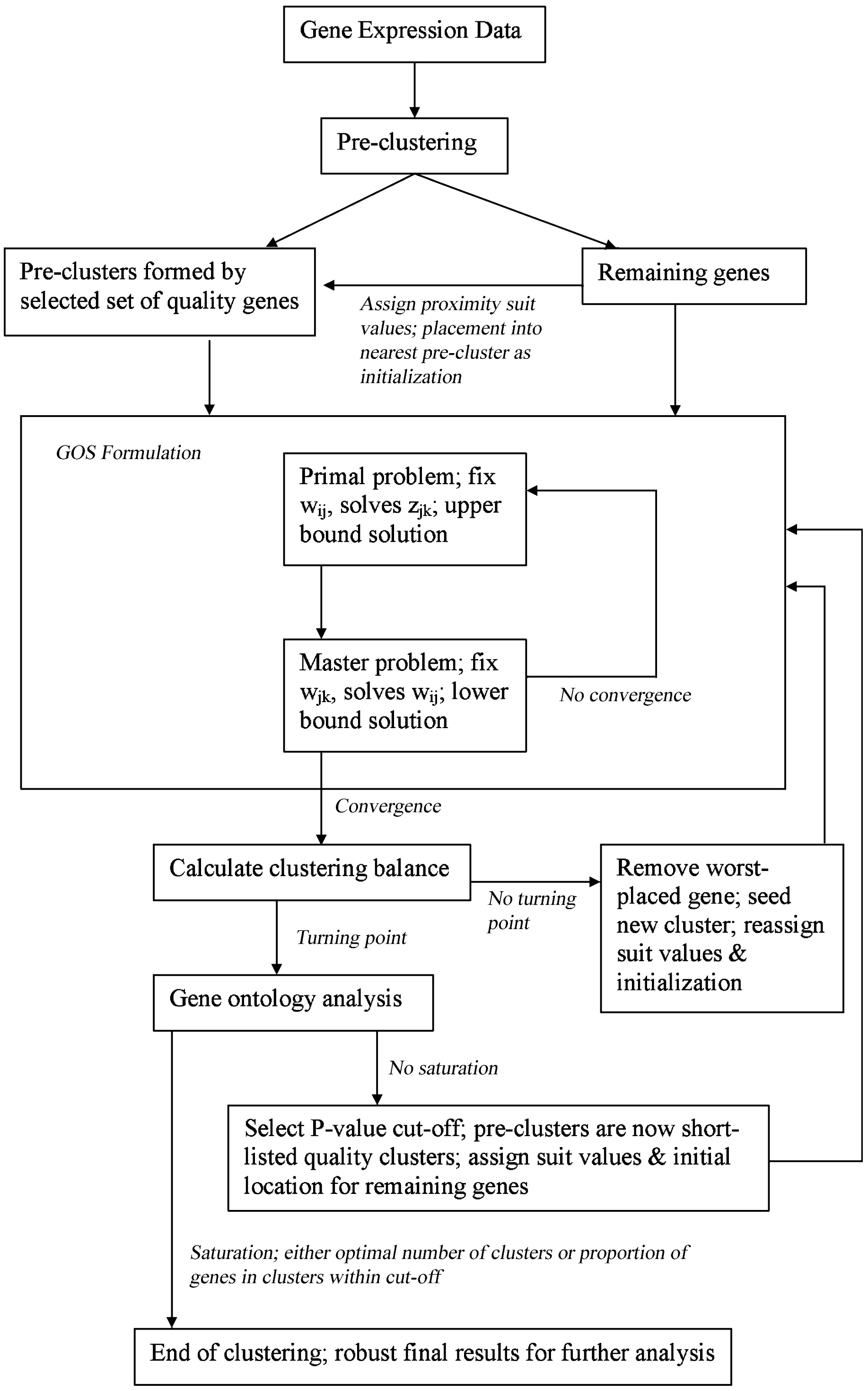
which is a constant for any given dataset.

Extension for Biological Coherence Refinement

Today, the advent of DNA microarray technology has made possible the large-scale monitoring of genomic behavior. In working with gene expression data, it is often useful to utilize external validation in evaluating clusters of gene expression data. Besides assessing the biological meaning of a cluster through the functional annotations of its constituent genes using gene ontology resources, other indications of strong biological coherence [25] are (i) the proportion of genes that reside in clusters with good P-value scores, (ii) cluster correlation, since closely related genes are expected to exhibit very similar patterns of expression, and (iii) cluster specificity, which is the proportion of genes within a cluster that annotates for the same function. A novel extension of the previously described work [25] allows not just for the determination of the optimal cluster number within the framework of a robust yet intuitive clustering method, but also for an iterative refinement of biological validation for the clusters. The algorithm is as follows.

Gene Preclustering We precluster the original data by proximity studies to reduce the computational demands by (i) identifying genes with very similar responses and (ii) removing outliers deemed to be insignificant to the clustering process. To provide just adequate discriminatory characteristics, preclustering can be done by reducing the expression vectors into a set of representative variables $\{+, o, -\}$, or by pregrouping genes that are close to one another by correlation or some other distance function.

Iterative Clustering We let the initial clusters be defined by the genes preclustered previously, and find the distance between each of the remaining genes and these initial clusters and as a good initialization point place these genes into the nearest cluster. For each gene, we allow its suitability in a limited number of clusters on the basis of the proximity study. In the primal problem of the global optimum search algorithm, we solve for $\mathrm{z_{jk}}$. These, together with the Lagrange multipliers, are used in the master problem to solve for $\mathrm{w_{ij}}$. The primal problem gives an upper-bound solution and the master problem gives a lower bound. The optimal solution is obtained when both bounds converge. Then, the worst-


Gene Expression Data
Pre-clustering
Pre-clusters formed by selected set of quality genes
Remaining genes
Assign proximity suit values; placement into nearest pre-cluster as initialization
GOS Formulation
Primal problem; fix w_{ij}, solves z_{jk}; upper bound solution
Master problem; fix w_{jk}, solves w_{ij}; lower bound solution
No convergence
Convergence
Calculate clustering balance
No turning point
Remove worst-placed gene; seed new cluster; reassign suit values & initialization
Turning point
Gene ontology analysis
No saturation
Select P-value cut-off; pre-clusters are now short-listed quality clusters; assign suit values & initial location for remaining genes
Saturation; either optimal number of clusters or proportion of genes in clusters within cut-off
End of clustering; robust final results for further analysis

Determining the Optimal Number of Clusters, Figure 1
Iterative clustering procedure. *GOS* global optimum search

placed gene is removed and used as a seed for a new cluster. This gene has already been subjected to a membership search, so there is no reason for it to belong to any of the older clusters. The primal and master problems are iterated and the number of clusters builds up gradually until the optimal number is attained.

Iterative Extension Indication of strong biological coherence is characterized by good P values based on gene ontology resources and the proportion of genes that reside in such clusters. As an extension, we would like to mine for the maximal amount of relevant information from the gene expression data and sieve out the least relevant data. This is important because information such as biological function annotation drawn from the cluster content is often used in the further study of coregulated gene members, common reading frames, and gene regulatory networks. From the clustered genes, we impose a coherence floor, based on some or all of the possible performance factors such as functional annotation, cluster specificity, and correlation, to demarcate genes that have already been well clustered. We then iterate to offer the poorly placed genes an opportunity to either find relevant membership in one of the strongly coherent clusters, or regroup amongst themselves to form quality clusters. Through this process, a saturation point will be reached eventually whereby the optimal number of clusters becomes constant as the proportion of genes distributed within clusters of high biological coherence levels off. Figure 1 shows a schematic of the entire clustering algorithm.

References

1. Azuaje F (2002) A Cluster Validity Framework for Genome Expression Data. Bioformatics 18:319–320
2. Bezdek JC, Pal NR (1998) Some New Indexed of Cluster Validity. IEEE Trans Syst Man Cybern 28:301–315
3. Calinski RB, Harabasz J (1974) A Dendrite Method for Cluster Analysis. Commun Stat 3:1–27
4. Davis DL, Bouldin DW (1979) A Cluster Separation Measure. IEEE Trans Pattern Anal Machine Intell 1(4):224–227
5. Dunn JC (1974) Well Separated Clusters and Optimal Fuzzy Partitions. J Cyber 4:95–104
6. Floudas CA (1995) Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, New York
7. Floudas CA, Aggarwal A, Ciric AR (1989) Global Optimum Search for Non Convex NLP and MINLP Problems. Comp Chem Eng 13(10):1117–1132
8. Goodman L, Kruskal W (1954) Measures of Associations for Cross-Validations. J Am Stat Assoc 49:732–764
9. Halkidi M, Batistakis Y, Vazirgiannis M (2002) Cluster Validity Methods: Part 1. SIGMOD Record 31(2):40–45
10. Hubert L, Schultz J (1976) Quadratic Assignment as a General Data-Analysis Strategy. Brit J Math Stat Psych 29: 190–241
11. Jaccard P (1912) The Distribution of Flora in the Alpine Zone. New Phytol 11:37–50
12. Jain AK, Dubes RC (1988) Algorithms for Clustering Data. Prentice-Hall Advanced Reference Series, Prentice-Hall, New Jersey
13. Jung Y, Park H, Du D, Drake BL (2003) A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering. J Global Optim 25:91–111
14. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science 220(4598):671–680
15. Krzanowski WJ, Lai YT (1985) A Criterion for Determining the Number of Groups in a Data Set using Sum of Squares Clustering. Biometrics 44:23–44
16. Lukashin AV, Fuchs R (2001) Analysis of Temporal Gene Expression Profiles: Clustering by Simulated Annealing and Determining the Optimal Number of Clusters. Bioinformatics 17(5):405–414
17. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller EJ (1953) Equations of State Calculations by Fast Computing Machines. J Chem Phys 21:1087
18. Milligan GW, Cooper MC (1985) An Examination of Procedures for Determining the Number of Clusters in a Data Set. Psychometrika 50:159–179
19. Pauwels EJ, Fregerix G (1999) Finding Salient Regions in Images: Non-parametric Clustering for Image Segmentation and Grouping. Comput Vis Image Underst 75:73–85
20. Rand WM (1971) Objective Criteria for the Evaluation of Clustering Methods. J Am Stat Assoc 66(336):l846–850
21. Ray S, Turi R (1999) Determination of Number of Clusters in K-Means Clustering and Application in Color Image Segmentation. In: Proceed 4th Int Conf Advances in Pattern Recognition and Digital Techniques, 137–143
22. Rousseeuw PJ (1987) Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. J Comp Appl Math 20:53–65
23. Tan MP, Broach JR, Floudas CA (2007) A Novel Clustering Approach and Prediction of Optimal Number of Clusters: Global Optimum Search with Enhanced Positioning. J Global Optim 39:323–346
24. Tan MP, Broach JR, Floudas CA (2008) Evaluation of Normalization and Pre-Clustering Issues in a Novel Clustering Approach: Global Optimum Search with Enhanced Positioning. J Bioinf Comput Biol 5(4):895–913
25. Tan MP, Broach JR, Floudas CA (2008) Microarray Data Mining: A Novel Optimization-Based Iterative Clustering Approach to Uncover Biologically Coherent Structures (submitted for publication)

Deterministic and Probabilistic Optimization Models for Data Classification

YA-JU FAN, W. ART CHAOVALITWONGSE
Department of Industrial and Systems Engineering, Rutgers University, Piscataway, USA

MSC2000: 65K05, 55R15, 55R35, 90C11

Article Outline

A classification problem is concerned with categorizing a data point (entity) into one of G ($G \geq 2$) mutually exclusive groups based upon m (positive integer) specific measurable features of the entity. A classification rule is typically constructed from a sample of entities, where the group classifications are known or labeled (training or supervised learning). Then it can be used to classify new unlabeled entities. Many classification methods are based on distance measures. A common approach is to find a hyperplane to classify two groups ($G = G_1 \bigcup G_2$). The hyperplane can be represented in a form of $A\omega = \gamma$, where A denotes an $n \times m$ input data matrix, n is the total number of input data points, and m is the total number of data features/attributes. The classification rule is then made by the weight vector ω to map data points onto a hyperplane, and the scalar γ, which are best selected by solving a mathematical programming model. The goal is to have entities of Group 1 (G_1) lie on one side of the hyperplane and entities of Group 2 (G_2) lie on the other side. Support Vector Machines (SVM) is the most studied hyperplane construction method. The SVM concept is to construct a hyperplane that minimizes the upper bound on the out-of-sample error. The critical step of SVM is to transform (or map) data points on to a high dimensional space, known as *kernel transformation*, and classify data points by a separating plane [9]. Subsequently, the hybrid linear programming discriminant model is proposed by [12,13,20]. The hybrid model does not depend on data transformation, where the objective is to find a plane that minimizes violations and maximizes satisfactions of the classified groups. Glover [19] proposed a mixed integer programming (MIP) formulation for the hybrid model by adding binary variables for misclassified entities. Other MIP formulations that are subsequently developed include [1,15,16]. Recently, a new technique that use multiple hyperplanes for classification has been proposed by [17]. This technique constructs a piecewise-linear model that gives convex separating planes. Subsequently, Better, Glover and Samorani [6] proposed multi-hyperplane formulations that generate multiple linear hyperplanes simultaneously with the consequence of forming a binary decision tree.

In classification, the selection of data's features/attributes is also very critical. Many mathematical programming methods have been proposed for selecting well represented features/attributes. Bennett and Mangasarian [5,23] gives a feature selection formulation such that the model not only separates entities into two groups, but also tries to suppress nonsignificant features. In a more recent study, Chaovalitwongse et al. (2006) proposed Support Feature Machine (SFM) formulations can be used to find a set of features that gives the highest classification performance [10].

Baysian decision method has also been widely studied in classification. However, there are only few studies incorporating the Baysian model with mathematical programming approaches. Among those studies, Asparouhov and Danchev [4] formulates a MIP model with binary variables, which are conformed with the Bayesian decision theory. In the case of multi-group classification, Anderson [2] developed a mathematical formulation that incorporates the population densities and prior probabilities of training data. This model yields classification rules for multi-groups with a reject option, (a set having entities that does not belong to any group) [22].

Deterministic Optimization Models

Support Vector Machines

Support Vector Machines (SVM) is aimed at finding a hyperplane that separates the labeled input data into two groups, G_1 and G_2. Then the optimal plane can be used for classifying new data point. The hyperplane can be mathematically expressed by $A\omega = \gamma$, where $\omega \in \Re^m$ is an m-dimensional vector of real numbers, m is total number of attributes/features used to represent a data entity, and $\gamma \in \Re^n$ is a scalar vector. All elements from G_1 and G_2 will be separated by this hyperplane under the assumption that the sets G_1 and G_2 are separable. Define margin as the minimum distance from the plane to elements in a group, G_1 or G_2. The objective function of SVM is to find a separating hyperplane with the largest margin. The data set G_1 can be represented by the matrix $A_i \in \Re^{k\times m}$, $i \in G_1$ and the set G_2 can be represented by the matrix $A_j \in \Re^{(n-k)\times m}$, $j \in G_2$, where k are number of data points (entities) of group G_1. Two open half spaces defined by the hyperplane are $\{A_i\omega < \gamma\}$ and $\{A_j\omega > \gamma\}$. One contains elements of G_1 and the other contains elements of G_2. Therefore, a linear programming (LP) problem can be formulated to determine the optimal values of vectors ω and γ. To construct valid inequalities for linear programming, we rescale the variables (ω, γ), by dividing them by the positive value $\min\limits_{i\in G_1, j\in G_2} \{A_i\omega - \gamma, -A_j\omega + \gamma\}$. Let e denote a vector of ones, and the resulting inequalities become

$$A_i\omega \geq e\gamma + e, \quad A_j\omega \leq e\gamma - e\,. \tag{1}$$

The performance of the SVM relies heavily on the kernel transformation, the data mapping to a high dimension. SVM can also incorporates nonlinear mapping $\phi(\cdot)$. If the new dimension is sufficiently high enough, the data from two classes can always be separated by a hyperplane [9,11]. Examples of SVM kernel functions include linear, polynomial, radial basis function (RBF) and sigmoid. Recently, Shimodaira et al. [24] has proposed the Dynamic Time-Alignment Kernel for time series data.

Robust LP for SVM

It is important to note that the above LP model assumes that A_i and A_j are perfectly separable, which is usually not a case in practice. In other words, it is possible that the inequalities in Eq. (1) provide no solution as the data are not perfectly separable. Bennett and Mangasarian [5] proposed an improved formulation that minimizes an average misclassifications given by

$$\begin{aligned} &\min_{\omega,\gamma,y,z} \frac{e^T y}{m} + \frac{e^T z}{k} \\ &\text{s.t. } A_i\omega - e\gamma - e \geq y, \\ &\quad -A_j\omega + e\gamma - e \geq z, \\ &\quad y \geq 0, z \geq 0\,. \end{aligned}$$

It is easy to see that the variables y and z are, in fact, the vectors representing the violations of inequalities in Eq. (2) and minimizing the objective function would lead to the minimum average violation.

Feature Selection with SVM

We note that an extension of the robust LP formulation can be used for feature selection [5,23]. A new term is added in the objective function in the robust LP model to suppress the components of ω. This would try to eliminate all unnecessary features. Let v denote the absolute value of the weight vector ω, log is the base of the natural logarithm, and $\lambda \in (0, 1)$. The mathematical program with a concave objective function and linear constraints for feature selection is given by

$$\begin{aligned} \min_{\omega,\gamma,y,z,v} \quad & (1-\lambda)\left(\tfrac{e^T y}{m} + \tfrac{e^T z}{k}\right) + \lambda e^T(e - \log^{-\alpha v}) \\ \text{s.t.} \quad & A_i\omega - e\gamma - e \geq y, \\ & -A_j\omega + e\gamma - e \geq z, \\ & y \geq 0, z \geq 0, \\ & -v \leq \omega \leq v\,. \end{aligned}$$

Note that when $\lambda = 0$, the model gives a plane that separates A_i and A_j without considering feature suppression. On the other hand, when $\lambda > 0$, the objective not only tries to separate A_i and A_j, but also tries to eliminate as many of ω components as possible. Specifically, for each v_i $(i = 1, \ldots, n)$, we minimize an exponential smoothing of the step function $(1 - \log^{-\alpha v_i})$. This step function enables the deletion of irrelevant components of ω. There also exists a finitely-terminating algorithm that solves this problem using successive linear programming [8].

Hybrid LP Discriminant Model

A hybrid LP discriminant model is proposed in [12,13,20]. This model is guaranteed to give the optimal solution regardless of the nature of the data. Therefore, the solution is invariant to transformations. This model is improved to overcome many shortcomings of contemporary linear discriminant formulations, which are reviewed and discussed in [21]. Recall that m is the number of attributes, all data points in G_1 are represented by an $k \times m$ matrix $A_i,\ i \in G_1$, and all data points in G_1 are represented by an $(n-k) \times m$ matrix $A_j,\ j \in G_2$. For the simplicity of mathematical representation, the membership in G_1 or G_2 can be represented by $i \in G_1$ or $i \in G_2$, respectively. This model will give a hyperplane of the form $A^T\omega = \gamma$, where the model seeks for the optimal weight vector ω, and a scalar γ, where data points of Group 1 lie on one side of the hyperplane and data points of Group 2 lie on the other side (i. e., $A_i\omega < \gamma, i \in G_1$ and $A_i\omega > \gamma, i \in G_2$). Let y_i and z_i represent external and internal deviation variables referring to the point violations and satisfactions of the classification rule. More specifically, they are the magnitudes of the data points lying outside or inside their targeted half spaces. The objective is to minimize violations and maximize the satisfactions of the classified groups. Thus, in the objective function, variable h_i's discourage external deviations and variable k_i's encourage internal deviations. Then $h_i \leq k_i$ for $i = 0$ and $i \in G$, must be satisfied. The hybrid model is given by

$$\begin{aligned}
\min\ & h_0 y_0 + \sum_{i\in G} h_i y_i - k_0 z_0 - \sum_{i \in G} k_i z_i \\
\text{s.t. } & \mathbf{A}_i\omega - y_0 - y_i + z_0 + z_i = \gamma, \quad i \in G_1 \\
& \mathbf{A}_i\omega + y_0 + y_i - z_0 - z_i = \gamma, \quad i \in G_2 \\
& z_0 + \sum_i z_i = 1, \quad i \in G \qquad (2) \\
& y_0, z_0 \geq 0 \\
& y_i, z_i \geq 0, \quad i \in G \\
& \omega, \gamma \qquad \text{unrestricted.}
\end{aligned}$$

We note that Eq. (2) is a normalization constraint that is necessary for avoiding a trivial solution where all $\omega_j = 0$ and $\gamma = 0$. Glover [18] identifies more normalization methods to conquer the problem with null weighting.

MIP Discriminant Model

There are several related mixed integer formulations in the literature [1,15,16]. In general, due to the computational requirements, these standard MIP formulations can only be applied to classification problems with a relatively small number of observations. Glover [19] proposed a compact mathematical program for discriminant model, which is a variant of the above-mentioned hybrid LP model. This objective of this model is to minimize the number of misclassified entities. The MIP discriminant model is given by

$$\begin{aligned}
\min\ & \sum_{i\in G} z_i \\
\text{s.t. } & A_i x - M z_i + \beta_i = b, \quad i \in G_1 \\
& A_i x + M z_i - \beta_i = b, \quad i \in G_2 \\
& \beta_i \geq 0, \quad i \in G \\
& z_i \in \{0,1\}, \quad i \in G \\
& x, b \quad \text{unrestricted,}
\end{aligned}$$

where β_i are slack variables, and M is a large constant chosen so that when $z_i = 1$, $A_i x \leq b + M z_i$ will be redundant for $i \in G_1$ and $A_i x \geq b - M z_i$ will be redundant for $i \in G_2$. This model can incorporate a normalization constraint, $(-n_2 \sum_{i \in G_1} A_i + n_1 \sum_{i\in G_2} A_i)x = 1$, where n_1 and n_2 are the number of entities in G_1 and G_2, respectively.

Multi-hyperplane Classification

Multi-hyperplane formulations, given by Better et al. [6], generate multiple linear hyperplanes simultaneously with the consequence of forming a decision tree. The hyperplanes are generated from an extension of the Discriminant Model proposed by Glover [18]. Instead of using kernel transformation that projects data into a high dimensional space to improve the performance of SVM, the multi-hyperplane approach approximates a nonlinear separation by constructing multiple hyperplanes. Let $d = 0$ when we are at a root node of a binary tree, where none of the classifications have been done. Let $d = D$ when the tree has two leaf nodes corresponding to the final separation step. In order to explain the model, we define the following terms.

- Successive Perfect Separation (SPS) is a procedure that forces all elements of Group 1 (G_1) and Group 2 (G_2) to lie on one side of the hyperplane at each node for any depth $d \in \{0, \ldots, D-1\}$. SPS is a special use of a variant based on a proposal of Glover [18].

- SPS decision tree is a tree that results from the two-group classification iteratively applying the SPS procedure. The root node ($d = 0$) contains all the entities in the data set, and at $d = D$ the two leaf nodes correspond to the final separation step.

For a given maximum depth D, an initial multi-hyperplane model considers each possible SPS tree type of depth d, for $d = 0, \ldots, D-1$. A root node is viewed as a "problem" node where all data points from both groups need to be separated. A leaf node, on the other hand, is considered to be a "decision" node where data points are classified into two groups. Define slicing variables sl_i for $i \in \{1, \ldots, D-1\}$. There are total of $D-1$ slicing variables needed for a tree having maximum depth D. Specifically, at depth $d = 1$, $sl_1 = 0$ if the "left" node constitutes a leaf node while the "right" node constitutes a root (or problem) node. Without loss of generality, we herein consider $D = 3$ for the initial multi-hyperplane model. The mathematical model for multi-hyperplane SVM can be formally defined as follows.

Let M and ε denote large and small positive constants, respectively, and G denote a set of the union of entities in G_1 and G_2. Suppose there are n entities in the training data set. Define a binary variable $z_i^* = 0$ if object i is correctly classified by the "tree", otherwise $z_i^* = 1$. Define a binary variable and $z_{hi}^* = 0$ if object i is correctly classified by "hyperplane h", otherwise $z_{hi}^* = 1$. The multi-hyperplane SVM model also includes traditional hyperplane constraints for each depth d of the tree and the normalization constraint, which is similar to the mixed integer programming model in [18]. Then, ε is added to prevent data points from lying on the hyperplane. Tree-type constraints are included to identify the optimal tree structure for the data set, which will be in part of the optimal classification rule. Binary variables y_i are used for tree types (0,1) and (1,0) to activate or deactivate either-or constraints. The SPS decision tree formulation for the depth $D = 3$ is given by

$$\min \sum_{i=1}^{n} z_i^*$$

$$\text{s.t.} A_i x_d - M z_{di} + \beta_i = b_d - \varepsilon \qquad i \in G_1,\ d = 1,2,3 \tag{3}$$

$$A_i x_d + M z_{di} - \beta_i = b_d + \varepsilon \qquad i \in G_2,\ d = 1,2,3 \tag{4}$$

$$M(sl_1 + sl_2) + z_i^* \geq z_{1i} + z_{2i} + z_{3i} - 2 \quad i \in G_1 \tag{5}$$

$$M(sl_1 + sl_2) + M z_i^* \geq z_{1i} + z_{2i} + z_{3i} \quad i \in G_2 \tag{6}$$

$$M(2 - sl_1 - sl_2) + M z_i^* \geq z_{1i} + z_{2i} + z_{3i} \quad i \in G_1 \tag{7}$$

$$M(2 - sl_1 - sl_2) + z_i^* \geq z_{1i} + z_{2i} + z_{3i} - 2 \quad i \in G_2 \tag{8}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} - M y_i \qquad i \in G_1 \tag{9}$$

$$M(1 + sl_1 - sl_2) + M z_i^* \geq z_{2i} + z_{3i} - M[1 - y_i] \quad i \in G_1 \tag{10}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} \qquad i \in G_2 \tag{11}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{2i} + z_{3i} - 1 \quad i \in G_2 \tag{12}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} \qquad i \in G_1 \tag{13}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{2i} + z_{3i} - 1 \qquad i \in G_1 \tag{14}$$

$$M(1 + sl_1 - sl_2) + z_i^* \geq z_{1i} - M y_i \qquad i \in G_2 \tag{15}$$

$$M(1 + sl_1 - sl_2) + M z_i^* \geq z_{2i} + z_{3i} - M[1 - y_i] \quad i \in G_2 \tag{16}$$

$$\sum_{j=1}^{n}\sum_{d=1}^{3} x_{jd} = 1 \tag{17}$$

$$z_i^* \in \{0,1\},\ z_{di} \in \{0,1\},\ y_i \in \{0,1\},$$
$$i \in G, d = 1,2,3$$
$$sl_k \in \{0,1\}\ k = 1,2,\ x, b \text{ unrestricted},$$

where the constraints in Eqs. (3)-(4) are the hyperplane constraints, Eqs. (5)-(6) are the constraints for tree type (0,0), Eqs. (7)-(8) are the constraints for tree type (1,1), Eqs. (9)-(12) are the constraints for tree type (0,1), Eqs. (13)-(16) are the constraints for tree type (1,0), and Eq. (17) is the normalization constraint. This small model with $D = 3$ performs well for small depths and has computational limitations. The reader should refer to [6] for a greater detail of an improved and generalized structure model for all types of SPS trees.

Support Feature Machines

Support Feature Machines (SFM) proposed in [10] is a mathematical programming technique used to identify a set of features that gives the highest performance in classification using the nearest neighbor rule. SFM can be formally defined as follows. Assume there are n data points, each with m features, we define the decision variables $x_j \in \{0,1\}$ $(j = 1,\dots,m)$ indicating if feature j is selected by SFM and $y_i \in \{0,1\}$ $(i = 1,\dots,n)$ indicating if sample i can be correctly classified by SFM. There are two versions of SFM, *voting* and *averaging*. Each version uses different weight matrices, which are provided by user's classification rule.

The objective function of voting SFM is to maximize the total correct classification as in Eq. (18). There are two sets of constraints used to ensure that the training samples are classified based on the voting nearest neighbor rule as in Eqs. (19)-(20). There is a set of logical constraints in Eq. (21) used to ensure that at least one feature is used in the voting nearest neighbor rule. The mixed-integer program for voting SFM is given by:

$$\max \sum_{i=1}^{n} y_i \tag{18}$$

$$\text{s.t.} \sum_{j=1}^{m} a_{ij}x_j - \sum_{j=1}^{m} \frac{x_j}{2} \le My_i$$
$$\text{for } i = 1,\dots,n \tag{19}$$

$$\sum_{j=1}^{m} \frac{x_j}{2} - \sum_{j=1}^{m} a_{ij}x_j + \epsilon \le M(1-y_i)$$
$$\text{for } i = 1,\dots,n \tag{20}$$

$$\sum_{j=1}^{m} x_j \ge 1 \tag{21}$$

$$x \in \{0,1\}^m,\quad y \in \{0,1\}^n,$$

where $a_{ij} = 1$ if the nearest neighbor rule correctly classified sample i at electrode j, 0 otherwise, n is total number of training samples, m is total number of features, $M = m/2$, and ϵ is a small positive number used to break a tie during the voting $(0 < \epsilon < 1/2)$.

The objective function of averaging SFM is to maximize the total correct classification as in Eq. (22). There are two sets of constraints used to ensure that the training samples are classified based on the distance averaging nearest neighbor rule as in Eqs. (23)-(24). There is a set of logical constraints in Eq. (25) used to ensure that at least one feature is used in the distance averaging nearest neighbor rule. The mixed-integer program for averaging SFM is given by:

$$\max \sum_{i=1}^{n} y_i \tag{22}$$

$$\text{s.t.} \sum_{j=1}^{m} \bar{d}_{ij}x_j - \sum_{j=1}^{m} d_{ij}x_j \le M_{1i}y_i$$
$$\text{for } i = 1,\dots,n \tag{23}$$

$$\sum_{j=1}^{m} d_{ij}x_j - \sum_{j=1}^{m} \bar{d}_{ij}x_j \le M_{2i}(1-y_i)$$
$$\text{for } i = 1,\dots,n \tag{24}$$

$$\sum_{j=1}^{m} x_j \ge 1 \tag{25}$$

$$x \in \{0,1\}^m,\quad y \in \{0,1\}^n,$$

where d_{ij} is the average statistical distance between sample i and all other samples from the same class at feature j (*intra-class distance*), $\bar{d}_{ij}$ is the average statistical distance between sample i and all other samples from different class at feature j (*inter-class distance*), $M_{1i} = \sum_{j=1}^{m} \overline{d_{ij}}$, and $M_{2i} = \sum_{j=1}^{m} d_{ij}$.

Probabilistic Optimization Models

The deterministic classification models in the previous section make a strong assumption that the data are separable. In the case that the data may not be well separated, using the deterministic models may lead to a high misclassification rate. The classification models that incorporate probabilities may be a better option for such noisy data. When the population densities and prior probabilities are known, there are probabilistic models that consider constrained rules with a reject option [2] as well as a Baysian-based model [4].

Bayesian-Based Mathematical Program

The Baysian-based mathematical program that are conformed with the Bayesian decision theoretic approach is proposed by Asparouhov and Danchev [4]. The model can be formally defined as follows. Denote $c \in \Re$ as a cut-off value, $x \in B^m$ as a vector of m binary values, and $\omega \in \Re^m$ is a decision variable having m-dimensional vector of real numbers. A preprocessing needs to be performed so that if $x^T\omega \leq c$, the entity x belong to class 1; otherwise it belongs to class 2. Suppose we have a set of n data points, n_1 data points are in G_1 and n_2 data points are in G_2, $(n = n_1 + n_2)$. Let s be a non empty multinomial cell. Denote n_{is} as the number of design set observation from the class i, where $i = 1, 2$, falling in this cell s. There are 2^m number of multinomial cells. Each cell is unique and all observations that belongs to it have exactly the same values of the m binary variables. Denote M as a sufficiently large positive real number, and ϵ as a small positive number. In addition to having a geometric interpretation, this formulation is inspired from Bayesian decision theoretic approach and having prior probabilities, $n_i/n > \forall i$, incorporated. Experimental studies in [4] suggest this Baysian-based model can give better performance than other contemporary linear discriminant models. The Baysian-based classification formulation is given by

$$\begin{aligned}
\min_{\omega, z_s, c} \ & \sum_s (|n_{1s} - n_{2s}|\, z_s + \min(n_{1s}, n_{2s})) \\
\text{s.t. } & \mathbf{x}_s^T\omega - Mz_s \leq c \text{ if } n_{1s} \geq n_{2s} \\
& \mathbf{x}_s^T\omega + Mz_s \geq c + \varepsilon \text{ if } n_{1s} < n_{2s} \\
& n_{1s} + n_{2s} \neq 0 \\
& z_s \in \{0,1\},\ \omega \in \Re^m,\ c \in \Re\,.
\end{aligned}$$

Probabilistic Models for Classification

An optimization model proposed by Anderson [2] incorporates population densities, prior probabilities from all groups, and misclassification probabilities. This method is aimed to find a partition $\{R_0, R_1, \ldots, R_G\}$ of Re^m where m is the number of features. This method naturally forms a multi-group classification. The objective is to maximize the probability of correct allocation subject to constraints on the misclassification probabilities. The mathematical model can be formally defined as follows. Let f_h, $h = 1, \ldots, G$, denote the group conditional density functions. Let π_g denote the prior probability that a randomly selected entity is from group g, $g = 1, \ldots, G$, and α_{hg}, $h \neq g$, are constants between 0 and 1. The probabilistic classification model is then given by

$$\begin{aligned}
\min \ & \sum_{g=1}^{G} \pi_g \int_{R_g} f_g(w)\,\mathrm{d}w \\
\text{s.t. } & \int_{R_g} f_h(w)\,\mathrm{d}w \leq \alpha_{hg} \\
& \text{for } h, g = 1, \ldots, G,\ h \neq g\,.
\end{aligned}$$

The optimal rule that can be used as a classification method is given by

$$R_g = \left\{ x \in \Re^k : L_g(x) = \max_{h \in 0,1,\ldots,G} L_h(x) \right\}\,, \quad (26)$$

where $g = 0, \ldots, G$, $L_0(x) = 0$, and $L_h(x) = \pi_h f_h(x) - \sum_{i=1, i\neq h}^{G} \lambda_{ih} f_i(x)$, for $h = 1, \ldots, G$. In general, there exist nonnegative constants $\lambda_{ih}, i, h \in 1, \ldots, G, i \neq h$, such that this optimal rule holds. The procedure for deriving a discriminant rule is composed of two stages. The first stage is to compute $\hat{f}_h$, which are estimated density functions f_h, and $\hat{\pi}_h$, which are estimated prior probabilities π_h, for $h = 1, \ldots, G$. There are many methods proposed for density estimation. The second stage is to estimate the optimal $\lambda'_{jh}s$, given the estimates $\hat{f}'_h s$ and $\hat{\pi}'_h s$. For estimating the $\lambda'_{jh}s$, there is a MIP approach proposed in [14], and a LP approach proposed in [22].

The MIP approach uses binary variables to record whether each entity was allocated to each region. This approach measures the probabilities of correct classification and misclassification for any candidate set of

$\lambda'_{ih}s$, which are calculated as the proportion of training samples that fall into each of the regions. The objective is to maximize a linear combination of variables representing correct allocation. The proportions of training samples misclassified were incorporated in constraints on misclassification probabilities. On the other hand, the LP approach does not have binary variables to incorporate proportions of misclassified training data points, and to provide a mechanism for modeling a priori bounds on misclassification probabilities. Instead, the LP approach provides a mechanism for estimating $\lambda'_{ih}s$ that balances the minimization of misclassifications and the maximization of correct classifications. This can be demonstrated as follows. Redefine the function $L_h, h = 1, \ldots, G$, as

$$L_h(x) = \pi_h p_h(x) - \sum_{i=1, i \neq h}^{G} \lambda_{ih} p_i(x) , \tag{27}$$

where $p_i(x) = f_i(x)/\sum_{t=1}^{G} f_t(x)$. This is analogous to the definition of original p_i in Eq. (26) since R_g can be expressed as $R_g = \{x \in \Re^k : L_g(x) \leq L_h(x),\ h = 0, \ldots, G\}$, if and only if, $\left(\left(1/\sum_{t=1}^{G}\right) f_t(x)\right) L_g(x) \leq \left(\left(1/\sum_{t=1}^{G}\right) f_t(x)\right) L_h(x)$. Note that this new definition of L_h is just an assumption. In addition, we also assume that we have a training sample of n data points whose group classifications are known. There are n_g data points in group g and $\sum_{g=1}^{G} n_g = n$. For notational convenience, let $\gamma = 1, \ldots, G$ and $N_g = 1, \ldots, n_g$. Each data point x has k attributes, denoted as $x^{gj} \in \Re^k$ for $g = 1, \ldots, G$ and $j = 1, \ldots, n_g$.

MIP Formulation for Anderson's Model

In order to find the optimal estimation of the second stage for solving Anderson's formula in Eq. (27), after the estimates $\hat{f}'_h s$ and $\hat{\pi}'_h s$ are given, the optimal $\lambda'_{jh} s$ is the final goal. For estimating the $\lambda'_{jh} s$, Gallagher et al. [14] proposed a MIP formulation. Same notation used in Anderson's formula in last section is applied here. The model ensures that the proportion of training data points and total data points n_g of group g in region R_h is less than or equal to a pre-specified percentage, $\alpha_{hg} > (0 < \alpha_{hg} < 1)$, for $h, g \in \gamma$ and $h \neq g$. The original formulation of the approach is a nonlinear MIP model given by

$$\min_{L_{hgj}, y_{gj}, \lambda_{ih}, u_{gj}} \sum_{g \in \gamma} \sum_{j \in N_g} u_{ggj}$$

s.t.

$$L_{hgj} = \pi_h \hat{p}_h(x^{gj}) - \sum_{i \in \gamma \setminus \{h\}} \lambda_{ih} \hat{p}_i(x^{gj}) \tag{28}$$

$$\text{for} \quad h, g \in \gamma, j \in N_g$$

$$y_{gj} = \max\left\{0, L_{hgj} : h = 1, \ldots, G\right\} \tag{29}$$

$$\text{for} \quad g \in \gamma, j \in N_g$$

$$y_{gj} - L_{ggj} \leq M\left(1 - u_{ggj}\right) \tag{30}$$

$$\text{for} \quad g \in \gamma, j \in N_g$$

$$y_{gj} - L_{hgj} \geq \varepsilon(1 - u_{hgj}) \tag{31}$$

$$\text{for} \quad h, g \in \gamma, h \neq g, j \in N_g$$

$$\sum_{j \in N_g} u_{hgj} \leq \lfloor \alpha_{hg} n_g \rfloor \tag{32}$$

$$\text{for} \quad h, g \in \gamma, h \neq g$$

$$-\infty < L_{hgj} < \infty \quad \text{for} \quad h, g \in \gamma, j \in N_g$$

$$y_{gj} \geq 0 \quad \text{for} \quad g \in \gamma, j \in N_g$$

$$\lambda_{ih} \geq 0 \quad \text{for} \quad i \in N_g, h \in \gamma$$

$$u_{gj} \in \{0, 1\} \quad \text{for} \quad g \in \gamma, j \in N_g$$

The above nonlinear mixed integer programming model can be transformed to an equivalent linear mixed integer model. The transformation is made by replacing the constraint in Eq. (29) with the following constraints:

$$\begin{aligned}
&y_{gj} \geq L_{hgj} \quad h, g \in \gamma,\ j \in N_g \\
&\tilde{y}_{hgj} - L_{hgj} \leq M(1 - v_{ghj}) \quad h, g \in \gamma,\ j \in N_g \\
&\tilde{y}_{hgj} \leq \pi_h \hat{p}_h(x^{gj}) v_{hgj} \quad h, g \in \gamma,\ j \in N_g \\
&\sum_{h \in G} v_{hgj} \leq 1 \quad g \in \gamma,\ j \in N_g \\
&\sum_{h \in G} \tilde{y}_{hgj} = y_{gj} \quad g \in \gamma,\ j \in N_g ,
\end{aligned}$$

where $\tilde{y}_{hgj} \geq 0$ and $v_{ghj} \in \{0, 1\}$, for $h, g \in \gamma,\ j \in N_g$. The constraints in Eq. (28) define the decision variable L_{hgj} as a function value of L_h at x^{gj}. The variable y_{gj} in Eq. (29) gives a result that x^{gj} lies in region R_h, if and only if, $y_{gj} = L_{hgj}$. The binary variable u_{hgj} is used to indicate whether or not x^{gj} lies in region R_h. The constraints in Eq. (30) together with the objective function ensure that $u_{ggj} = 1$, if and only if, the jth entity from group g is correctly allocated to group g. The constraints in Eqs. (31)-(32) ensure that at most $\lfloor \alpha_{hg} n_g \rfloor$ data points of group g are allocated to group h, $h \neq g$. Note that the condition of indicator variables, $u_{hgj} = 0$, $h \neq g$, implies that $x^{gj} \notin R_h$ by Eq. (31), but the converse need not hold. As a result, the number of misclassifications may be overcounted. To force the converse hold, (that is $u_{hgj} = 1$, if and only if, $x^{gj} \in R_h, \forall h, g \in \gamma$), one can include the following constraints: $y_{gj} - L_{hgj} \leq M\left(1 - u_{hgj}\right)$ for $h, g \in \gamma, j \in N_g$. However, the addition of such constraints substantially increases the solution times and the actual amount of overcounting is minimal. M and ε are large and small positive constants, respectively. Since this MIP formulation is very difficult to solve, especially it involves $2GN$ binary variables. There is a preprocessing strategy suggested in [14] by aggregating variables and constraints. Special branching strategies for solving the MIP model is also suggested in [14]. Those strategies include branching on the smallest indexed fractional-valued binary variable, branching on the most infeasible fractional-valued binary variable, pseudo reduced-cost branching schemes, and strong branching [3,7].

LP Formulation for Anderson's Model

In order to estimate the $\lambda'_{jh}s$ for solving Anderson's formula in the second stage in Eq. (27), Lee et al. [22] proposed the Linear Programming (LP) model that minimizes a penalty function in order to allocate each training entity to its correct group or to the reserved-judgment region. Note that same notation used in the MIP approach and Anderson's formula is consistent here. The method is given by

$$\min_{L_{hgj},\omega_{gj},y_{gj},\lambda_{ih}} \sum_{g\in\gamma} \sum_{j\in N_g} \left(c_1\omega_{gj} + c_2 y_{gj}\right)$$

s.t.

$$L_{hgj} = \pi_h \hat{p}_h(x^{gj}) - \sum_{i\in\gamma\setminus\{h\}} \lambda_{ih}\hat{p}_i(x^{gj}) \qquad (33)$$

$$\text{for} \quad h, g \in \gamma, j \in N_g$$

$$L_{ggj} - L_{hgj} + \omega_{gj} \geq 0 \qquad (34)$$

$$\text{for} \quad h, g \in \gamma, h \neq g, j \in N_g$$

$$L_{ggj} + \omega_{gj} \geq 0 \qquad (35)$$

$$\text{for} \quad g \in \gamma, j \in N_g$$

$$-L_{hgj} + y_{gj} \geq 0 \qquad (36)$$

$$\text{for} \quad h, g \in \gamma, j \in N_g$$

$$-\infty < L_{hgj} < \infty \quad \text{for} \quad h, g \in \gamma, j \in N_g$$

$$\omega_{gj} \geq 0 \quad \text{for} \quad g \in \gamma, j \in N_g$$

$$y_{gj} \geq 0 \quad \text{for} \quad g \in \gamma, j \in N_g$$

$$\lambda_{ih} \geq 0 \quad \text{for} \quad i \in N_g, h \in \gamma\,.$$

The constraints in Eq. (33) define the decision variable L_{hgj} as a function value of L_h for x^{gj}. If the optimal solution yields $\omega_{gj} = 0$, for some (g, j) pair, the constraints in Eqs. (34)-(35) imply that $L_{ggj} = \max\{0, L_{hgj} : h \in \gamma\}$. Thus, when $\omega_{gj} = 0$, it means that the jth entity from group g is correctly classified. If $y_{gj} = 0$ is the case for some (g, j) pair, then the constraints in Eq. (36) implies that $L_{ggj} = \max\{0, L_{hgj} : h \in \gamma\} = 0$. Hence, the jth entity from group g is placed in the reserved-judgment region. If both ω_{gj} and y_{gj} are positive, the jth entity from group g is misclassified. The optimization solver is attempting either to correctly classify training data points ($\omega_{gj} = 0$), or to place them in the reserved-judgment region ($y_{gj} = 0$). The optimizer's emphasis can be realized by varying the weights c_1 and c_2. It is possible for both ω_{gj} and y_{gj} to be zero. One should decide how to interpret in such situation. Recall the optimal rule in Eq. (26), which constrains that if x belongs to the reserved judgment region ($h = 0$) then it gives the function value $L_0(x) = 0$.

References

1. Abad PL, Banks WJ (1993) New LP based heuristics for the classification problem. Eur J Oper Res 67:88–100
2. Anderson JA (1969) Constrained discrimination between k populations. J Royal Stat Soc Series B 31:123–139
3. Applegate D, Bixby RE, Chvatal V, Cook W (1994) The traveling salesman problem. Technical Report, Dept Comput Appl Math, Rice University, Houston, TX
4. Asparouhov O, Danchev S (1997) Discrimination and classification in the presence of binary variables. Biocybern Biomed Eng 17(1–2):25–39
5. Bennett KP, Mangasarian OL (1992) Robust linear programming discrimination of two linearly inseparable sets. Optim Meth Soft 1:23–34
6. Better M, Glover F, Samorani M (2006) Multi-Hyperplane Formulations for Classification and Discrimination Analysis. submitted for the Student Paper Award of the Decision Analysis Society, working paper, University of Colorado, Boulder, Colorado
7. Bixby RE, Cook W, Cox A, Lee EK (1995) Computational experience with parallel mixed integer programming in a distributed environment. Research Monograph CRPC-TR95554, Center for Research on Parallel Computation, Rice University, Houston, TX
8. Bradley PS, Mangasarian OL, Street WN (1998) Feature selection via mathematical programming. INFORMS J Comput 10:209–217
9. Burges C (1998) Tutorial on Support Vector Machines for Pattern Recognition. Data Min Know Discov 2:121–167
10. Chaovalitwongse WA, Fan YJ, Sachdeo RC (2006) Novel Optimization Models for Multidimensional Time Series Classification: Application to the Identification of Abnormal Brain Activity. Submitted to Oper Res
11. Duda RO, Hart PE, Stork DG Pattern Classification, 2nd edn. Wiley-Interscience, New York
12. Freed E, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Eur J Oper Res 7(1):44–60
13. Freed E, Glover F (1986) Resolving certain difficulties and improving the classification power of the LP discriminant analysis procedure. Decis Sci 17:589–595
14. Gallagher RJ, Lee EK, Patterson DA (1997) Constrained discriminant analysis via 0/1 mixed integer programming. Annals Oper Res 74:65–88
15. Glen JJ (1999) Integer programming methods for normalization and variable selection in mathematical programming discriminant analysis models. J Oper Res Soc 50: 1043–1053
16. Glen JJ (2003) An iterative mixed integer programming method for classification accuracy maximizing discriminant analysis. Comput Oper Res 30:181–198
17. Glen JJ (2005) Mathematical programmming models for piecewise-linear discriminant analysis. J Oper Res Soc 56:331–341
18. Glover F (1990) Improved Linear Programming Models for Discriminant Analysis. Decis Sci 21(4):771–785
19. Glover F (1993) Improved Linear and Integer Programming Models for Discriminant Analysis. Creative and Innovative Approaches to the Science of Management. RGK Foundation Press, Austin, pp 187–215
20. Glover F, Keene S, Duea B (1988) A new class of models for the discriminant problem. Decis Sci 19:269–280
21. Joachimsthaler EA, Stam A (1990) Mathematical Programming Approaches for the Classification Problem in Two-Group Discriminant Analysis. Multi Behav Res 25(4):427–454
22. Lee EK, Gallagher RJ, Patterson DA (2003) A Linear Programming Approach to Discriminant Analysis with a Reserved-Judgment Region. INFORMS J Comput 15(1): 23–41
23. Mangasarian OL (1997) Mathematical Programming in Data Mining. Data Min Know Discov 1:183–201
24. Shimodaira H, Noma KI, Nakai M, Sagayama S (2001) Dynamic Time-Alignment Kernel in Support Vector Machine. Adv Neural Inf Process Sys 14(2):921–928

Differential Equations and Global Optimization

MAURIZIO BRUGLIERI[1], PIERLUIGI MAPONI[1], MARIA CRISTINA RECCHIONI[2], FRANCESCO ZIRILLI[3]

[1] Dip. Mat. e Fisica, University Camerino, Camerino, Italy

[2] Ist. Mat. e Stat., University Ancona, Ancona, Italy

[3] Dip. Mat. G. Castelnuovo, University Roma La Sapienza, Roma, Italy

MSC2000: 60G35, 65K05

Article Outline

Keywords

Global optimization; Stochastic differential equation

Let $\mathbf{N}$ be the set of natural numbers. Let $N \in \mathbf{N}$ and $\mathbf{R}^N$ be the N-dimensional real Euclidean space, let $\mathbf{C}^N$ be the N-dimensional complex Euclidean space, $\mathbf{R}$, $\mathbf{C}$ are used in place of $\mathbf{R}^1$, $\mathbf{C}^1$ respectively. Let $\mathbf{R}^+ = \{x \in \mathbf{R}: x \geq 0\}$. Let $x \in \mathbf{R}$; then $|x|$ denotes the absolute value of x. Let $\mathbf{x}, \mathbf{y} \in \mathbf{R}^N$; then $(\mathbf{x}, \mathbf{y})$ denotes the scalar product in $\mathbf{R}^N$ and $\| \mathbf{x} \|$ denotes the euclidean norm in $\mathbf{R}^N$. Let $\mathbf{S}^N = \{\mathbf{x} \in \mathbf{R}^{N+1}: \| \mathbf{x} \| = 1\}$.

Let $D \subseteq \mathbf{R}^N$ be a connected domain, let $F: D \to \mathbf{R}$ be a given function. The following problem is considered:

$$\min_{\mathbf{x} \in D} F(\mathbf{x}). \tag{1}$$

When $D = \mathbf{R}^N$ problem (1) is called the *global unconstrained optimization problem*. When $D \subset \mathbf{R}^N$ it is called the *global constrained optimization problem*.

Without loss of generality one considers only the minimization problem, that is, problem (1), since the maximization problem can be easily reduced to a minimization problem.

To solve problem (1) means to find a point $\mathbf{x}^* \in D$ such that $F(\mathbf{x}^*) \leq F(\mathbf{x})$, $\forall \mathbf{x} \in D$.

A large number of problems with great theoretical and practical interest can be formulated as global optimization problems, that is, as problem (1).

In this article the global optimization problems are studied only from the point of view of numerical optimization and in particular of numerical methods based on differential equations. Many other fruitful points of view are possible to study that include the set of global minimizers of F on D or in general the set of critical points of F on D depending on the hypotheses made on F and D.

A method to solve problem (1) in the sense of numerical optimization is usually an iterative scheme that from a given initial guess $\mathbf{x}^0 \in D$ is able to compute a sequence $\{\mathbf{x}^n \in D: n \in \mathbf{N}\}$ such that $\mathbf{x}^n \to \mathbf{x}^*$ when $n \to \infty$.

Problem (1) can be easily solved in some special cases, that is, when the function F and the domain D have special forms, for example one can recall the following two important cases:

- *linear programming problem*: F linear function, D convex polyhedron, i. e., $D \subset \mathbf{R}^N$ is defined implicitly by means of equalities and inequalities between linear functions;
- *convex programming problem*: F convex function, $D \subset \mathbf{R}^N$ convex region.

One notes that the linear programming problem can be considered as a special case of the convex programming problem. For both cases effective methods to solve problem (1) are known, e. g., for the linear programming problem the simplex method, see [7], and for the convex programming problem the Newton method coupled with some strategy to treat the constraints that define D, for example active set strategy, see [9].

In general, problem (1) is a difficult one since the property of being a global minimizer is not a local property. That is, a global minimizer $\mathbf{x}^*$ cannot be recognized from local properties of the function F at $\mathbf{x}^*$, such as the value of F and its derivatives at $\mathbf{x}^*$. Numerical algorithms to recognize global properties are unusual and in general computationally expensive.

For example, let $D = \mathbf{R}$, $m, \alpha \in \mathbf{R}$, $\delta > 0$, one considers the following two functions:

$$F_1(x) = -\frac{1}{1+x^2},$$
$$F_2(x) = -\frac{1}{1+x^2} + m \begin{cases} e^{\frac{1}{(x-\alpha)^2-\delta^2}}\, e^{\frac{1}{\delta^2}}, & x \in (\alpha-\delta, \alpha+\delta), \\ 0, & x \notin (\alpha-\delta, \alpha+\delta). \end{cases}$$

Function F_1 has in $x = 0$ the unique local minimizer which is also the global minimizer, i. e. $x^* = 0$. Let $m < -1$, $0 < \delta < |\alpha|$; then function F_2 has several critical points including two local minimizers, one is $x = 0$ and the other is $x = x_2 \in (\alpha - \delta, \alpha + \delta)$. Moreover the global minimizer of F_2 is $x = x^* = x_2$. One notes that F_1, F_2 are smooth functions and that they coincide, for every $x \in \mathbf{R} \setminus (\alpha - \delta, \alpha + \delta)$ where $\delta > 0$ is arbitrary.

Let $D = \mathbf{R}^N$, let $F: D \to \mathbf{R}$ be a continuously differentiable function, let ∇F be the gradient of F, let $\mathbf{x} \in D$ be such that $(\nabla F)(\mathbf{x}) \neq 0$ then the vector $-(\nabla F)(\mathbf{x})$ gives the direction of steepest descent for the function F at the point $\mathbf{x}$. One can consider the following system of differential equations:

$$\frac{d\mathbf{x}}{dt}(t) = -(\nabla F)(\mathbf{x}(t)), \quad t > 0, \tag{2}$$

$$\mathbf{x}(0) = \mathbf{x}^0. \tag{3}$$

Under some hypotheses on F, the solution of problem (2), (3) is a trajectory in $\mathbf{R}^N$ starting from $\mathbf{x}^0$ and ending in the critical point $\mathbf{x}^*_{\text{loc}}(\mathbf{x}^0)$ of F whose attraction region contains $\mathbf{x}^0$. Using a numerical integration scheme for (2), (3) one can obtain a numerical optimization method, for example choosing the Euler integration scheme with variable stepsize from (2), (3) one obtains the so-called *steepest descent algorithm*. Let $\xi^k \in \mathbf{R}^N$ be the approximation of $\mathbf{x}(t_k)$, $k \in \mathbf{N}$, where $t_0 = 0$, $0 < t_k < t_{k+1} < +\infty$, $k = 1, 2, \dots$, and $t_k \to +\infty$ when $k \to \infty$, obtained with a numerical optimization method coming from (2), (3). Suppose $\{\boldsymbol{\xi}^k\ k \in \mathbf{N}\}$ is a sufficiently good approximation of the solution $\mathbf{x}(t)$, $t > 0$ of (2), (3) one has $\lim_{k\to\infty} \boldsymbol{\xi}^k = \lim_{t\to+\infty} \mathbf{x}(t) = \mathbf{x}^*_{\text{loc}}(\mathbf{x}^0)$, thus the numerical optimization methods obtained from (2), (3) compute critical points that depend on the initial guess $\mathbf{x}^0$. So that these critical points usually are not global minimizers of F.

One can consider numerical optimization methods due to other differential equations instead of (2), that is differential equations taking in account higher order derivatives of F or of $\mathbf{x}(t)$. However the minimizers computed with these numerical optimization methods depend only on local properties of the function F, thus in general they will not be global minimizers of F. So that methods based on ordinary differential equations are inadequate to deal with problem (1).

In this article it is described how to use stochastic differential equations to avoid this difficulty. In fact one wants to destabilize the trajectories generated by problem (2), (3) using a stochastic perturbation in order to be able to reach global minimizers. This must be an appropriate perturbation, that is the corresponding perturbed trajectories must be able to leave the attraction region of a local minimizer of F to go in an attraction region of another minimizer of F obtaining as $t \to +\infty$ the solution of problem (1). This is done by adding a stochastic term, i. e., a Brownian motion on the right-hand side of equation (2). Moreover this stochastic term takes into account the domain D, when $D \subset \mathbf{R}^N$. This is done introducing the solution of the Skorokhod reflection problem.

In the second section one gives some mathematical background about stochastic differential equations that is necessary to state the results of the third and fourth sections. In the third section, the unconstrained version of problem (1) is treated, i. e., $D = \mathbf{R}^N$. In the fourth section, the constrained version of problem (1) is treated, i. e., $D \subset \mathbf{R}^N$. In both these sections one gives methods, convergence analysis and discussion when possible of a relevant software library. In the last section one gives some information about new application areas of global optimization such as graph theory and game theory.

Mathematical Background

Let $\Omega \subseteq \mathbf{R}$, Σ be a σ-field of subsets of Ω and P be a probability measure on Σ. The triple $(\Omega, \Sigma, \mathrm{P})$ is called a *probability measure space*, see [5] for a detailed introduction to probability theory. Let $\Omega' \subseteq \mathbf{R}$, Υ be a topology of subsets of Ω'. Then $X : \Omega \to \Omega'$ is a random variable if $\{X \in A\} \in \Sigma$ for every $A \in \Upsilon$.

The *distribution function* $G_X : \mathbf{R} \to [0, 1]$ of X is defined by $G_X(x) = \mathrm{P}\{X \le x\}$, $x \in \mathbf{R}$ and one denotes with g_X its density. The expected value or the mean value of X is defined as follows:

$$m(X) = \int_{\mathbb{R}} x G_X(\,dx) = \int_{\mathbb{R}} x g_X(x)\,dx \tag{4}$$

and the variance of X is given by:

$$v(X) = m((X - m(X))^2). \tag{5}$$

For example, a random variable X has discrete distribution, or is concentrated on $x_1, \dots, x_n$, when $g_X(x) = \sum_{i=1}^n p_i \delta(x - x_i)$, where $p_i > 0$, $x_i \in \Omega'$, $i = 1, \dots, n$, $\sum_{i=1}^n p_i = 1$ and δ is the Dirac delta. Given $m \in \mathbf{R}$, $v > 0$ a random variable has normal distribution when

$$g_X(x) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(x-m)^2}{2v}},$$

one notes that $m(X) = m$ and $v(X) = v$.

A *stochastic process* is a family of random variables depending on a parameter t, that is, $\{X(t)\colon \Omega \to \Omega', t \ge 0\}$. A *Brownian motion* is a stochastic process $\{w(t)\colon t \ge 0\}$ having the following properties:

- $\mathrm{P}\{w(0) = 0\} = 1$;
- for every choice of t_i, $i = 1, \dots, k$, $0 \le t_i < t_{i+1} < +\infty$, $i = 1, \dots, k-1$, the increments $w(t_{i+1}) - w(t_i)$, $i = 1, \dots, k-1$, are independent and normally distributed random variables with mean value equal to zero and variance equal to $t_{i+1} - t_i$.

An *N-dimensional Brownian motion* is a N-dimensional process

$$\{\mathbf{w}(t) = (w_1(t), \dots, w_N(t))\colon\ t \ge 0\}$$

where its components $\{w_i(t)\colon \Omega \to \Omega', t \ge 0\}$, $i = 1, \dots, N$, are independent Brownian motions. The Brownian motion is a good mathematical model to describe phenomena that are the superposition of a large number of chaotic elementary independent events. The most famous example of Brownian motion is the motion of pollen grains immersed in a fluid, the grains have a chaotic perpetual motion due to the collisions with the molecules of the fluid, see [15, p. 39].

Let $\Pi = \Omega' \times \cdots \times \Omega' \subset \mathbf{R}^N$, where $\times$ denotes the Cartesian product of sets. Let Υ' be a topology of subsets of Π. Let s, t, be such that $0 \le s \le t$, let $\mathbf{x} \in \Pi$, $A \in \Upsilon'$, then the transition distribution function of a N-dimensional stochastic process $\{\mathbf{X}(t) : t \ge 0\}$ is defined as follows:

$$T(s, \mathbf{x}, t, A) = \mathsf{P}\{\mathbf{X}(t) \in A \text{ and } \mathbf{X}(s) = \mathbf{x}\}. \tag{6}$$

When T can be written as:

$$T(s, \mathbf{x}, t, A) = \int_A p(s, \mathbf{x}, t, \mathbf{y})\, d\mathbf{y} \tag{7}$$

for every $0 \le s \le t$, $\mathbf{x} \in \Pi$, $A \in \Upsilon'$ then the function p is called the *transition probability density* of the process $\{\mathbf{X}(t)\colon t \ge 0\}$.

Finally, if there exists a density distribution function π that depends only on $\mathbf{x} \in \Pi$ such that:

$$\pi(\mathbf{x}) = \lim_{t \to +\infty} p(s, \mathbf{u}, t, \mathbf{x}), \tag{8}$$

then π is called the *steady-state distribution density* of the process $\{\mathbf{X}(t)\colon t \ge 0\}$.

One considers the following stochastic differential equation:

$$d\mathbf{Z}(t) = \boldsymbol{\alpha}(\mathbf{Z}(t), t)\, dt + \beta(\mathbf{Z}(t), t)\, d\mathbf{w}(t), \quad t > 0, \tag{9}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{10}$$

where $\mathbf{w}$ is the N-dimensional Brownian motion, α is the drift coefficient and β is the diffusion coefficient, see [8, p. 98] or [8, p. 196] for a detailed discussion. One notes that $d\mathbf{w}$ cannot be considered as a differential in the elementary sense and must be understood as a stochastic differential, see [8, p. 59]. Under regularity assumptions on α and β there exists a unique solution $\{\mathbf{Z}(t)\colon \mathrm{t} > 0\}$ of (9), (10), see [8, p. 98].

When $\boldsymbol{\alpha}$ is minus the gradient of a potential function equation (9) is called the *Smoluchowski-Kramers equation*. The Smoluchowski-Kramers equation is a singular limit of the Langevin equation. The Langevin equation expresses Newton principle for a particle subject to a random force field, see [15, p. 40].

Let $\mathrm{div}_{\mathbf{y}}$ be the divergence operator with respect to the variables $\mathbf{y}$, $\Delta_{\mathbf{y}}$ be the Laplace operator with respect to the variables $\mathbf{y}$ and $L_{\beta}, \alpha(\cdot) = \mathrm{div}_{\mathbf{y}}(\cdot\alpha) - (1/2)\Delta_{\mathbf{y}}(\cdot\beta^2)$. Under regularity assumptions on α and β, the transition probability density $p(s, \mathbf{x}, t, \mathbf{y})$, $0 \le s < t$, $\mathbf{x}, \mathbf{y} \in \mathbf{R}^N$, associated to the solution $\{\mathbf{Z}(t)\colon t \ge 0\}$ of problem (9), (10) exists and satisfies the Fokker–Planck equation, (see 8, p. 149]) that is, given $\mathbf{x} \in \mathbf{R}^N$, $s \ge 0$ one has:

$$\frac{\partial p}{\partial t} + L_{\beta,\boldsymbol{\alpha}}(p) = 0, \quad \mathbf{y} \in \mathbb{R}^N, \ t > s, \tag{11}$$

$$\lim_{t \to s, t > s} p(s, \mathbf{x}, t, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}), \quad \mathbf{y} \in \mathbb{R}^N. \tag{12}$$

For the treatment of the constrained global optimization, that is, problem (1) with $D \subset \mathbf{R}^N$, a stochastic process depending on the domain D must be considered. Let $\nu(\mathbf{x}) \subset \mathbf{S}^{\mathrm{N}-1}$ be the set-valued function that gives the outward unit normals of the boundary ∂D of D at the point $\mathbf{x} \in \partial D$. One notes that when $\mathbf{x}$ is a regular point of ∂D, $\nu(\mathbf{x})$ is a singleton. Let $\eta\colon [0, T] \to \mathbf{R}^N$, with possibly $[0, T] = \mathbf{R}^+$, let $|\eta|(t)$ be the total variation of η in the interval $[0, \mathrm{t}]$, where $t < T$. The *Skorokhod problem* is defined as follows: let ϕ, ψ, $\eta\colon [0, T] \to \mathbf{R}^N$, then the triple (ϕ, ψ, η) satisfies the Skorokhod problem, on $[0, T]$ with respect to D, if $|\eta|(T) < +\infty$, $\phi(0) = \psi(0)$ and for $t \in [0, T]$ the following relations hold:

$$\boldsymbol{\phi}(t) = \boldsymbol{\psi}(t) + \boldsymbol{\eta}(t), \tag{13}$$

$$\boldsymbol{\phi}(t) \in D, \tag{14}$$

$$|\boldsymbol{\eta}|(t) = \int_0^t \chi_{\{r \in \mathbb{R}:\ \boldsymbol{\phi}(r) \in \partial D\}}(s)\, d|\boldsymbol{\eta}|(s), \tag{15}$$

$$\boldsymbol{\eta}(t) = -\int_0^t \boldsymbol{\gamma}(s)\, d|\boldsymbol{\eta}|(s), \tag{16}$$

where χ_S is the characteristic function of the set S and $\boldsymbol{\gamma}(s) \in \nu(\phi(s))$, when $s \in [0, T]$ and $\boldsymbol{\phi}(s) \in \partial D$ and $\gamma(s) = \mathbf{O}$ elsewhere. Viewing $\psi(t)$, $t \in [0, T]$, as the trajectory of a point $\mathbf{A} \in \mathbf{R}^N$, one has that at time zero $\mathbf{A}$ is inside D, since $\psi(0) \in D$. Moreover the trajectory of $\mathbf{A}$ is reflected from the boundary of D and the reflected trajectory can be viewed as $\phi(t)$, $t \in [0, T]$. That is, ϕ is equal to ψ until $\mathbf{A} \in D$, when $\mathbf{A}$ goes out of D it is brought back on ∂D in the normal direction to ∂D. One notes that the function $\boldsymbol{\eta}$ gives the reflection rule with respect to the boundary of D of the function ψ. In [16] it is proved that under suitable assumptions on D and F there exists a unique solution of the Skorokhod problem.

One considers the following stochastic differential equation with reflection term, that is:

$$d\mathbf{Z}(t) = \boldsymbol{\alpha}(\mathbf{Z}(t), t)\,dt + \beta(\mathbf{Z}(t), t)\,d\mathbf{w}(t) + d\boldsymbol{\eta}(t), \quad t > 0\,, \tag{17}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{18}$$

where

$$(\mathbf{Z}, \mathbf{Z} - \boldsymbol{\eta}, \boldsymbol{\eta}) \tag{19}$$

is the solution of the Skorokhod problem. One notes that relations (14), (19) imply that the solution of (17), (18) verifies $\mathbf{Z}(t) \in D$, $t > 0$. In [16] it is proved that under some hypotheses there exists a unique solution $\{\mathbf{Z}(t)\colon t \geq 0\}$ of (17), (18), (19) for every $\mathbf{x}^0 \in D$.

Global Unconstrained Optimization

Given problem (1) with $D = \mathbf{R}^N$ one considers the following stochastic differential equation:

$$d\mathbf{Z}(t) = -(\nabla F)(\mathbf{Z}(t))\,dt + \sigma(t)d\mathbf{w}(t), \quad t > 0, \tag{20}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{21}$$

where $\{\mathbf{w}(t)\colon t \geq 0\}$ is the N-dimensional Brownian motion and $\sigma(t)$ is a suitable decreasing function that guarantees the convergence of the stochastic process $\{\mathbf{Z}(t) : t \geq 0\}$ to a random variable with density concentrated on the global minimizers of F. Under some assumptions on F, the transition probability density $p(0, \mathbf{x}^0, t, \mathbf{x})$, $\mathbf{x}^0$, $\mathbf{x} \in \mathbf{R}^N$, $t > 0$, of the process $\{\mathbf{Z}(t) : t \geq 0\}$ exists and verifies equations (11), (12); moreover, when $\sigma \equiv \epsilon$, $\epsilon > 0$, for the steady-state distribution density $\pi_\epsilon(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^N$, the following equation holds:

$$L_{\epsilon,-\nabla F}(\pi_\epsilon) = 0, \quad \mathbf{x} \in \mathbb{R}^N, \tag{22}$$

one has:

$$\pi_\epsilon(\mathbf{x}) = C_\epsilon e^{-\frac{2F(\mathbf{x})}{\epsilon^2}}, \quad \mathbf{x} \in \mathbb{R}^N, \quad \epsilon > 0, \tag{23}$$

where:

$$C_\epsilon = \left(\int_{\mathbb{R}^N} e^{-\frac{2F(\mathbf{y})}{\epsilon^2}}\,d\mathbf{y}\right)^{-1}, \quad \epsilon > 0. \tag{24}$$

One assumes $C_\epsilon < +\infty$ for $\epsilon > 0$. Moreover, one has:

$$p(0, \mathbf{x}^0, t, \mathbf{x}) = \pi_\epsilon(\mathbf{x}) + e^{-\frac{2F(\mathbf{x}^0)}{\epsilon^2}} \sum_{n=1}^{\infty} \pi_\epsilon^n(\mathbf{x})\pi_\epsilon^n(\mathbf{x}^0)e^{\lambda_\epsilon^n t}, \tag{25}$$

where π_ϵ^n is the eigenfunction of $L_{\epsilon,-\nabla F}$ corresponding to the eigenvalue λ_ϵ^n, $n = 1, 2, \ldots$, and $0 = \lambda_\epsilon^0 > \lambda_\epsilon^1 > \cdots$. One notes that the eigenfunctions π_ϵ^n, $n = 1, 2 \ldots$, are appropriately normalized and π_ϵ is the eigenfunction of $L_{\epsilon,-\nabla F}$ corresponding to the eigenvalue $\lambda_\epsilon^0 = 0$. Consider $N = 1$, the function F smooth and with three extrema in $x^-, x^0, x^+ \in \mathbf{R}$ such that $x^- < x^0 < x^+$. Moreover, F increases in (x^-, x^0) and in $(x^+, +\infty)$ and decreases in $(-\infty, x^-)$ and in (x^0, x^+). Let:

$$\begin{cases} c^- = \dfrac{d^2F}{dx^2}(x^-), \\ c^0 = \dfrac{d^2F}{dx^2}(x^0), \\ c^+ = \dfrac{d^2F}{dx^2}(x^+). \end{cases} \tag{26}$$

One assumes $c^\pm$, c^0 to be nonzero. In [1] it is shown that when $F(x^-) < F(x^+)$, one has $\pi_\epsilon(x) \to \delta(x - x^-)$ as $\epsilon \to 0$ while when $F(x^-) = F(x^+)$ one has $\pi_\epsilon(x) \to \gamma\delta(x - x^-) + (1-\gamma)\delta(x - x^+)$, where $\gamma = [1 + \sqrt{\frac{c^-}{c^+}}]^{-1}$ as $\epsilon \to 0$, and the limits are taken in distribution sense. That is in [1] it is shown that the steady-state distribution density tends to Dirac deltas concentrated on the global minimizers of F when $\epsilon \to 0$.

In [12] it is shown that:

$$\lambda_\epsilon^1 \approx -\frac{\sqrt{c^+c^0}}{2\pi} e^{-\frac{2}{\epsilon^2}\delta F} \quad \text{as} \quad \epsilon \to 0, \tag{27}$$

where $\delta F = \max\{F(x^0) - F(x^-), F(x^0) - F(x^+)\}$. Formula (25) shows that p converges to π_ϵ when $t \to +\infty$, but the rate of convergence becomes slow when ϵ is small. Replacing ϵ with $\sigma(t)$ a slowly decreasing function such that $\sigma(t) \to 0$ when $t \to +\infty$, using elementary adiabatic perturbation theory one can expect that the condition:

$$\int_0^{+\infty} e^{-\frac{2}{\sigma^2(t)}\delta F}\,\mathrm{d}t = +\infty \tag{28}$$

guarantees that $\{\mathbf{Z}(t) : t > 0\}$ is a solution of (20), (21) when $t \to +\infty$ converges to a random variable concentrated on the global minimizers of F.

In [6] the following result is proved:

Theorem 1 (***convergence theorem***) *Let $F : \boldsymbol{R}^N \to \boldsymbol{R}$ be a twice continuously differentiable function satisfying the following properties:*

$$\min_{\mathbf{x}\in\mathbb{R}^N} F(\mathbf{x}) = 0, \tag{29}$$

$$\lim_{\|\mathbf{x}\|\to+\infty} F(\mathbf{x}) = \lim_{\|\mathbf{x}\|\to+\infty} \|(\nabla F)(\mathbf{x})\| = +\infty, \tag{30}$$

$$\lim_{\|\mathbf{x}\|\to+\infty} \|(\nabla F)(\mathbf{x})\|^2 - (\Delta F)(\mathbf{x}) > -\infty, \tag{31}$$

let $\sigma(t) = \sqrt{(c)/(\log t)}$ for $t \to +\infty$, where $c > c_F > 0$ and c_F is a constant depending on the function F. Then the transition probability density p of the process $\{\mathbf{Z}(t) : t \geq 0\}$, solution of (20), (21), converges weakly to a stationary distribution π, that is:

$$p(0, \mathbf{x}, t, \cdot) \to \pi \quad \text{when } t \to +\infty. \tag{32}$$

Moreover the distribution π is the weak limit of π_ϵ, given by (23), as $\epsilon \to 0$.

One notes that (20), (21) is obtained perturbing the trajectories given by the steepest descent equation for F with the Brownian motion and σ is a factor that controls the amplitude of this perturbation. The fact that $\sigma(t) \to 0$ when $t \to +\infty$ makes possible the stabilization of the perturbed trajectories at the minimizers of F. With the assumptions of the convergence theorem it is possible to conclude that π is concentrated on the global minimizers of F, so that the random variable $\mathbf{Z}(t) = (Z_1(t), \dots, Z_N(t))$ 'converges' to $\mathbf{x}^*$, solution of problem (1), as $t \to +\infty$. That is, when $\mathbf{x}^*$ is the unique global minimizer of F, then $\mathrm{P}\{Z_i(t) = x_i^*\} \to 1$ when $t \to +\infty$ for $i = 1, \dots,$ N.

The stochastic differential equation (20) can be integrated numerically to obtain an algorithm for the solution of problem (1). Let $t_0 = 0$, $t_k = \sum_{l=0}^{k-1} h_l$, where $h_l > 0$, $l = 0, 1, \dots$, are such that $t_k \to +\infty$ when $k \to \infty$ then using the Euler method one has:

$$\boldsymbol{\zeta}^0 = \mathbf{x}^0, \tag{33}$$

$$\boldsymbol{\zeta}^{k+1} = \boldsymbol{\zeta}^k - h_k(\nabla F)(\boldsymbol{\zeta}^k) + \sigma(t_k)(\mathbf{w}(t_k + h_k) - \mathbf{w}(t_k)), \tag{34}$$

where $k = 0, 1, \dots$ and $\zeta^k \in \mathbf{R}^N$ is the approximation of $\mathbf{Z}(t_k)$, $k = 1, 2, \dots$, see [2,3].

In (34) due to the presence of the stochastic term, one can substitute the gradient of F with a kind of 'stochastic gradient' of F in order to save computational work, see [2,3] for details.

One notes that the sequence $\{\zeta^k : k \in \mathbf{N}\}$ depends on the particular realization of the Brownian motion $\{\mathbf{w}(t_k) : k = 0, 1, \dots\}$. That is, solving several times problem (20), (21), by means of (33), (34), the solutions obtained are not necessarily the same. However, the convergence theorem states that 'all' the solutions $\{\zeta^k : k \in \mathbf{N}\}$ obtained by (33), (34) tend to $\mathbf{x}^*$ as $k \to +\infty$.

So that in the numerical algorithm derived from (20), (21) using (33), (34) one can approximate by means of n_T independent realizations (i. e., trajectories) of the stochastic process $\{\mathbf{Z}(t) : t \geq 0\}$, solution of (20), (21). A possible strategy for a numerical algorithm is the following: after an 'observation period' the various trajectories are compared, one of them is discarded and is not considered any more, another one is branched. The new set of trajectories are computed throughout the next observation period. The following stopping conditions are used:

- uniform stop: the final values of the function F at the end of the various trajectories are numerically equal;
- maximum trial duration: a maximum number of observation periods has been reached.

One notes that the algorithms based on the discretization of the stochastic differential equations have sound mathematical basis, that is for a wide class of functions F some convergence results such as the convergence theorem given above are available. These algorithms usually have a slow convergence rate, this can be seen from the kind of function σ which is required in the convergence theorem. This implies that the algorithms based on stochastic differential equations have an high computational cost, so that their use is usually restricted to low-dimensional problems. However these algorithms can be parallelized with a significant computational advantage, for example in the algorithm described above each trajectory can be computed independently from the others until the end of an observation period. One notes that the algorithms derived from (20), (21) are in some sense similar to the simulated annealing algorithm (cf. also ► Simulated annealing methods in protein folding) introduced in combinatorial optimization in [11].

Global Constrained Optimization

Given problem (1) with $D \subset \mathbf{R}^N$ the following stochastic differential equation with reflection term is considered:

$$d\mathbf{Z}(t) = -(\nabla F)(\mathbf{Z}(t))\,dt + \sigma(t)\,d\mathbf{w}(t) + d\boldsymbol{\eta}(t), \quad t > 0, \tag{35}$$

$$\mathbf{Z}(0) = \mathbf{x}^0, \tag{36}$$

where $\mathbf{x}^0 \in D$, $\{\mathbf{w}(t) : t \geq 0\}$ is the N-dimensional Brownian motion, $\sigma(t)$ is a suitable decreasing function that guarantees the convergence of the stochastic process $\{\mathbf{Z}(t) : t > 0\}$ to a random variable with density concentrated on the global minimizers of F on D when $t \to +\infty$ and $\eta(t)$ is a suitable function to assure $\mathbf{Z}(t) \in D$, $t > 0$, that is, $(\mathbf{Z}, \mathbf{Z} - \eta, \eta)$ is the solution of the Skorokhod problem in $\mathbf{R}^+$ respect to D.

Let int(D) be the set of the interior points of D. One assumes that D is the closure of int(D). Let $p(0, \mathbf{x}^0, \mathrm{t}, \mathbf{x})$, $\mathbf{x}^0, \mathbf{x} \in \mathrm{int}(D)$, $t > 0$, be the transition probability density of the process $\{\mathbf{Z}(t) \colon t > 0\}$, solution of (35), (36), when $\sigma \equiv \epsilon$, $\epsilon > 0$. Then p satisfies the Fokker–Planck equation:

$$\frac{\partial p}{\partial t} + L_{\epsilon,-\nabla F}(p) = 0, \quad \mathbf{x} \in \mathrm{int}(D), \tag{37}$$

$$\lim_{t \to 0^+} p(0, \mathbf{x}^0, t, \mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}^0), \quad \mathbf{x} \in \mathrm{int}(D), \tag{38}$$

$$\left(\frac{\epsilon^2}{2} \nabla_{\mathbf{x}} p + p \nabla F, \mathbf{n}(\mathbf{x}) \right) = 0, \quad \mathbf{x} \in \partial D, \quad t > 0, \tag{39}$$

where $L_{\epsilon,-\nabla F}$ is defined in (11) (12) and $\mathbf{n}(\mathbf{x}) \in \nu(\mathbf{x})$ is the outward unit normal to ∂D in $\mathbf{x} \in \partial D$. One notes that boundary condition (39) assures that $\mathrm{P}\{\mathbf{Z}(t) \in D\} = 1$ for every $t > 0$. This boundary condition follows from the requirement that $(\mathbf{Z}, \mathbf{Z} - \eta, \eta)$ is the solution of the Skorokhod problem.

One assumes the following properties of F and D:

- $F : D \to \mathbf{R}$ is twice continuously differentiable;
- $D \subset \mathbf{R}^N$ is a bounded convex domain such that exists p satisfying (37), (38), (39) and exists the steady-state distribution density π of the process solution of (35), (36);
- let π_ϵ be the steady-state distribution density of the process solution of (35), (36) when $\sigma \equiv \epsilon$, $\epsilon > 0$, that is:

$$\pi_\epsilon(\mathbf{x}) = C_\epsilon e^{-\frac{2F(\mathbf{x})}{\epsilon^2}}, \quad \mathbf{x} \in D, \tag{40}$$

$$C_\epsilon = \left(\int_D e^{-\frac{2F(\mathbf{y})}{\epsilon^2}}\,d\mathbf{y} \right)^{-1} \tag{41}$$

and π is the weak limit of π_ϵ as $\epsilon \to 0$.

In analogy with the unconstrained case one can conjecture that when $D \subset \mathbf{R}^N$ and $F : D$ rarr; $\mathbf{R}$ satisfy the properties listed above and when $\sigma(t) = \sqrt{(c)/(\log t)}$ for $t \to +\infty$, where $c > c_F > 0$ and c_F is a constant depending on F, then the transition probability density $p(0, \mathbf{x}^0, \mathrm{t}, \mathbf{y})$, $\mathbf{x}^0, \mathbf{x} \in D$, $t > 0$ of the process $\{\mathbf{Z}(t) \colon t \geq 0\}$, solution of (35), (36) converges to a steady-state distribution density π when $t \to +\infty$ and π is the distribution density obtained as weak limit of π_ϵ when $\epsilon \to 0$. That is, the process $\{\mathbf{Z}(\mathrm{t}) \colon t \geq 0\}$ converges in law to a random variable concentrated at the points $\mathbf{x}^* \in D$ that solve problem (1).

A numerical algorithm to solve problem (1), with $D \subset \mathbf{R}^N$, can be obtained using a numerical method to integrate problem (35), (36). This is done integrating numerically problem (20), (21) and 'adding' the constraints given by D. In the numerical algorithm the trajectories can be computed using formulas (33), (34) when the trajectories are in D, when a trajectory violates the constraints, it is brought back on ∂D putting to zero its normal component with respect to the violated constraints. Finally the stopping conditions are the same ones considered in the previous section.

Analogously to the unconstrained problem, the algorithms based on the stochastic differential equations for the constrained case have slow convergence rate. However these algorithms have a high rate of parallelism.

Miscellaneous Results

In this section are shown two mathematical problems that are somewhat unusual as optimization problems.

Clique Problem

Let $I = \{1, \dots, N\} \subset \mathbf{N}$ be a finite set, let $I * I$ be the set of unordered pairs of elements of I. Let $E \subseteq I * I$. Then a *graph* G is a pair $G = (I, E)$, where I is the set of the nodes of G and E is the set of the edges of G, i. e. $\{i, j\} \in E$ implies that G has an edge joining nodes $i, j \in I$. A graph $G = (I, E)$ is said to be *complete* or to be a *clique* when $E = I * I$. A graph $G' = (I', E')$ is a subgraph of $G = (I, E)$ when $I' \subseteq I$ and $E' \subseteq E \cap (I' * I')$.

The *maximum clique problem* can be defined as follows: Given $G = (I, E)$, find the largest subgraph G' of G which is complete. Let $k(G)$ be the number of nodes of the graph G'.

Several algorithms exist to obtain a numerical solution of the maximum clique problem see, for example, [14] where the branch and bound algorithm is described.

One considers here the maximum clique problem as a continuous optimization problem. The adjacency matrix A of the graph $G = (I, E)$ is a square matrix of order equal to the number of nodes of G and its generic entry $A_{i,j}$, at row i and at column j, is defined equal to 1 if $\{i, j\} \in E$ and is equal to 0 otherwise. Then in [13] it is shown that:

$$1 - \frac{1}{k(G)} = \max_{\mathbf{x} \in S} \mathbf{x}^t A \mathbf{x}, \tag{42}$$

where

$$S = \left\{ \mathbf{x} = (x_1, \dots, x_N)^t \in \mathbb{R}^N : \sum_{i=1}^{N} x_i = 1,\ x_i \geq 0,\ i = 1, \dots, N \right\} .$$

One notes that many maximizers of (42) can exist, however there exists always a maximizer $\mathbf{x}^* = (x_1^*, \dots, x_N^*)^t$ of problem (42) such that for $i = 1, \dots, N$ one has $x_i^* = 1/k(G)$ if $i \in G'$ and $x_i^* = 0$ if $i \notin G'$. That is the maximum clique problem is reduced to a continuous global optimization problem that can be treated with the algorithms described above. Several other problems in graph theory can be reformulated as continuous optimization problems.

Quasivariational Inequalities

Let $X \subset \mathbf{R}^N$ be a nonempty set, let $\Omega(\mathbf{x}) \subset X$, $\mathbf{x} \in X$, be a set-valued function and let $\mathbf{F} : \mathbf{R}^N \to \mathbf{R}^N$. The *quasivariational inequality* problem, is defined as follows: Find a vector $\mathbf{x}^* \in \Omega(\mathbf{x}^*)$ such that:

$$\left(\mathbf{F}(\mathbf{x}^*), \mathbf{y} - \mathbf{x}^* \right) \geq 0, \quad \forall \mathbf{y} \in \Omega(\mathbf{x}^*), \tag{43}$$

see [4] for a detailed introduction to quasivariational inequalities. This problem can be reduced to the search of a fixed-point of a function defined implicitly by a variational inequality.

The quasivariational inequalities have many applications such as for example the study of the generalized Nash equilibrium points of an N-player noncooperative game. See [10] for a detailed discussion on N-player noncooperative games.

See also

- ▶ αBB Algorithm
- ▶ Continuous Global Optimization: Applications
- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ DIRECT Global Optimization Algorithm
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Topology of Global Optimization

References

1. Aluffi-Pentini F, Parisi V, Zirilli F (1985) Global optimization and stochastic differential equations. J Optim Th Appl, 47:1–17
2. Aluffi-Pentini F, Parisi V, Zirilli F (1988) A global optimization algorithm using stochastic differential equations. ACM Trans Math Software, 14:345–365
3. Aluffi-Pentini F, Parisi V, Zirilli F (1988) SIGMA - A stochastic integration global minimization algorithm. ACM Trans Math Softw 14:366–380
4. Baiocchi C, Capelo A (1984) Variational and quasi-variational inequalities: Application to Free-boundary problems. Wiley, New York
5. Billingsley P (1995) Probability and measure. Wiley, New York
6. Chiang TS, Hwang CR, Sheu SJ (1987) Diffusion for global optimization in Rn. SIAM J Control Optim 25:737–753
7. Dantzing GB (1963) Linear programming and extensions. Princeton Univ Press, Princeton
8. Friedman A (1975) Stochastic differential equations and applications, vol 1. Acad Press, New York

9. Gill PE, Murray W, Wright MH (1981) Practical optimization. Acad Press, New York
10. Harker P, Pang J (1990) Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. Math Program 48:161–220
11. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
12. Matkowsky BJ, Schuss Z (1981) Eigenvalues of the Fokker-Planck operator and the equilibrium for the diffusions in potential fields. SIAM J Appl Math 40:242–254
13. Motzkin TS, Straus EG (1964) Maxima for graphs and a new proof of a theorem of Turán. Notices Amer Math Soc 11:533–540
14. Pardalos PM, Rodgers GP (1990) Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing 45:131–144
15. Schuss Z (1980) Theory and applications of stochastic differential equations. Wiley, New York
16. Tanaka H (1979) Stochastic differential equations with reflecting boundary conditions in convex regions. Hiroshima Math J 9:163–177

Dini and Hadamard Derivatives in Optimization

VLADIMIR F. DEMYANOV
St. Petersburg State University, St. Petersburg, Russia

MSC2000: 90Cxx, 65K05

Article Outline

Keywords

Dini directional derivatives; Hadamard directional derivatives; Necessary and sufficient optimality conditions; Bouligand cone; Composition theorem; Lipschitz function; Quasidifferentiable function; Concave function; Convex function; First order approximation of a function; Nondifferentiable optimization; Nonsmooth analysis; Numerical methods; Maximizer; Maximum function; Minimum function; Minimizer; Steepest ascent direction; Steepest descent direction; Unconstrained optimum

Directional Derivatives

Let f be a function defined on some open set $X \subset \mathbf{R}^n$ and taking its values in $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. The set $\operatorname{dom} f = \{x \in X : |f(x)| < +\infty\}$ is called the *effective set* (or *domain*) of the function f. Take $x \in \operatorname{dom} f$, $g \in \mathbf{R}^n$. Put

$$f_{\mathrm{D}}^{\uparrow}(x, g) := \limsup_{\alpha \downarrow 0} \frac{1}{\alpha}\left[f(x + \alpha g) - f(x)\right], \tag{1}$$

$$f_{\mathrm{D}}^{\downarrow}(x, g) := \liminf_{\alpha \downarrow 0} \frac{1}{\alpha}\left[f(x + \alpha g) - f(x)\right]. \tag{2}$$

Here $\alpha \downarrow 0$ means that $\alpha \to +0$.

The quantity $f_{\mathrm{D}}^{\uparrow}(x, g)$ (respectively, $f_{\mathrm{D}}^{\downarrow}(x, g)$) is called the *Dini upper* (respectively, *lower*) derivative of the function f at the point x in the direction g.

The limit

$$f'(x, g) = f_{\mathrm{D}}'(x, g) := \lim_{\alpha \downarrow 0} \frac{1}{\alpha}\left[f(x + \alpha g) - f(x)\right], \tag{3}$$

is called the *Dini derivative* of f at the point x in the direction g. If the limit in (3) exists, then $f_{\mathrm{D}}^{\uparrow}(x, g) = f_{\mathrm{D}}^{\downarrow}(x, g) = f'(x, g)$.

The quantity

$$f_{\mathrm{H}}^{\uparrow}(x, g) := \limsup_{[\alpha, g'] \to [+0, g]} \frac{1}{\alpha}\left[f(x + \alpha g') - f(x)\right] \tag{4}$$

(respectively,

$$f_{\mathrm{H}}^{\downarrow}(x, g) := \liminf_{[\alpha, g'] \to [+0, g]} \frac{1}{\alpha}\left[f(x + \alpha g') - f(x)\right]), \tag{5}$$

is called the *Hadamard upper* (respectively, *lower*) derivative of the function f at the point x in the direction g.

The limit

$$f_{\mathrm{H}}'(x, g) := \lim_{[\alpha, g'] \to [+0, g]} \frac{1}{\alpha}\left[f(x + \alpha g') - f(x)\right] \tag{6}$$

is called the *Hadamard derivative* of f at x in the direction g.

If the limit in (6) exists, then $f_{\mathrm{H}}^{\uparrow}(x, g) = f_{\mathrm{H}}^{\downarrow}(x, g)$.

Note that the limits in (1), (2), (4) and (5) always exist but are not necessarily finite.

Remark 1 In the one-dimensional case ($\mathbf{R}^n = \mathbf{R}$) the Hadamard directional derivatives coincide with the corresponding Dini directional derivatives:

$$f_{\mathrm{H}}^{\uparrow}(x,g) = f_{\mathrm{D}}^{\uparrow}(x,g),$$
$$f_{\mathrm{H}}^{\downarrow}(x,g) = f_{\mathrm{D}}^{\downarrow}(x,g),$$
$$f'_{\mathrm{H}}(x,g) = f'_{\mathrm{D}}(x,g).$$

If the limit in (3) exists and is finite, then the function f is called *differentiable* (or *Dini differentiable*) at x in the direction g. The function f is called *Dini directionally differentiable* (Dini d.d.) at the point x if it is Dini differentiable at x for every $g \in \mathbf{R}^n$. Analogously, if the limit in (6) exists and is finite, the function f is called *Hadamard differentiable* at x in the direction g. The function f is called *Hadamard directionally differentiable* (Hadamard d.d.) at the point x if it is Hadamard differentiable at x for every $g \in \mathbf{R}^n$.

If the limit in (6) exists and is finite, then the limit in (3) also exists and $f'_H(x,g) = f'(x,g)$. The converse is not necessarily true.

All these derivatives are *positively homogeneous* (of degree one) functions of direction:

$$f_Q^*(x,\lambda g) = \lambda f_Q^*(x,g), \quad \forall \lambda \geq 0. \tag{7}$$

(Here $*$ is either $\uparrow$, or $\downarrow$, and Q is either D, or H.)

A function f defined on an open set X is called *Dini uniformly directionally differentiable* at a point $x \in X$ if it is directionally differentiable at x and for every $\varepsilon > 0$ there exists a real number $\alpha_0 > 0$ such that

$$\frac{1}{\alpha}\left[f(x+\alpha g) - f(x) - \alpha f'(x,g)\right] < \varepsilon,$$
$$\forall \alpha \in (0,\alpha_0), \quad \forall g \in S,$$

where $S = \{g \in \mathbf{R}^n \colon \|g\| = 1\}$ is the unit sphere.

Proposition 2 (see [2, Thm. I.3.2]) *A function f is Hadamard d.d. at a point $x \in X$ if and only if it is Dini uniformly differentiably at x and its directional derivative $f'(x,g)$ is continuous as a function of direction.*

Remark 3 If f is locally Lipschitz and Dini directionally differentiable at $x \in X$, then it is Hadamard d.d. at x, too.

For Dini and Hadamard derivatives (see (3) and (6)) there exists a calculus:

Proposition 4 *Let functions f_1 and f_2 be Dini (Hadamard) directionally differentiable at a point $x \in X$. Then their sum, difference, product and quotient (if $f_2(x) \neq 0$) are also Dini (Hadamard) d.d. at this point and the following formulas hold:*

$$(f_1 \pm f_2)'_Q(x,g) = f'_{1Q}(x,g) \pm f'_{2Q}(x,g), \tag{8}$$

$$(f_1 f_2)'_Q(x,g) = f_1(x)f'_{2Q}(x,g) + f_2(x)f'_{1Q}(x,g), \tag{9}$$

$$\left(\frac{f_1}{f_2}\right)'_Q(x,g) = -\frac{1}{(f_2(x))^2}\bigl(f_1(x)f'_{2Q}(x,g) - f_2(x)f'_{1Q}(x,g)\bigr). \tag{10}$$

Here Q is either D, or H.

These formulas follow from the classical theorems of differential calculus.

Proposition 5 *Let*

$$\varphi(x) = \max_{i \in 1:N} f_i(x), \tag{11}$$

where the functions f_i are defined and continuous on an open set $X \subset \mathbf{R}^n$ and Dini (Hadamard) d.d. at a point $x \in X$ in a direction g. Then the function φ is also Dini (Hadamard) d.d. at x and

$$\varphi'_Q(x,g) = \max_{i \in R(x)} f'_i(x,g), \tag{12}$$

where $R(x) = \{i \in 1:N : f_i(x) = \varphi(x)\}$ (see [2, Cor. I.3.2]).

If φ is defined by

$$\varphi(x) = \max_{y \in Y} f_i(x,y),$$

where Y is some set, then under some additional conditions a formula, analogous to (12), also holds (see [2, Chap. I, Sec. 3]).

A theorem on the differentiability of a composition can also be stated.

Unfortunately, formulas similar to (8)–(10) and (12) are not valid for Dini (Hadamard) upper and lower derivatives.

The Dini and Hadamard upper and lower directional derivatives are widely used in nonsmooth analysis and nondifferentiable optimization. For example, the following *mean value theorem* holds.

Proposition 6 (see [2, Thm. I.3.1]) *Let f be defined and continuous on the interval $\{y: y = x + \alpha g, \alpha \in [0, \alpha_0], \alpha_0 > 0\}$. Put*

$$m = \inf_{\alpha \in [0,\alpha_0]} f_D^{\downarrow}(x + \alpha g, g),$$
$$M = \sup_{\alpha \in [0,\alpha_0]} f_D^{\uparrow}(x + \alpha g, g).$$

Then [1]

$$m\alpha_0 \le f(x + \alpha_0 g) - f(x) \le M\alpha_0 .$$

The following first order approximations may be constructed via the Dini and Hadamard derivatives.

Proposition 7 *Let f be defined on an open set $X \subset \mathbf{R}^n$, and Dini d.d. at a point $x \in X$. Then*

$$f(x + \Delta) = f(x) + f_D'(x, \Delta) + o_D(x, \Delta). \tag{13}$$

If f is Hadamard d.d. at x, then

$$f(x + \Delta) = f(x) + f_H'(x, \Delta) + o_H(x, \Delta). \tag{14}$$

Let f be defined on an open set $X \subset \mathbf{R}^n$ and finite at $x \in X$. Then

$$f(x + \Delta) = f(x) + f_D^{\uparrow}(x, \Delta) + \overline{o}_D(x, \Delta), \tag{15}$$

$$f(x + \Delta) = f(x) + f_D^{\downarrow}(x, \Delta) + \underline{o}_D(x, \Delta), \tag{16}$$

$$f(x + \Delta) = f(x) + f_H^{\uparrow}(x, \Delta) + \overline{o}_H(x, \Delta), \tag{17}$$

$$f(x + \Delta) = f(x) + f_H^{\downarrow}(x, \Delta) + \underline{o}_H(x, \Delta), \tag{18}$$

where

$$\frac{o_D(x, \alpha\Delta)}{\alpha} \overset{\alpha\downarrow 0}{\to} 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{19}$$

$$\frac{o_H(x, \alpha\Delta)}{\|\Delta\|} \overset{\|\Delta\|\to 0}{\to} 0, \tag{20}$$

$$\limsup_{\alpha\downarrow 0} \frac{\overline{o}_D(x, \alpha\Delta)}{\alpha} = 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{21}$$

$$\liminf_{\alpha\downarrow 0} \frac{\underline{o}_D(x, \alpha\Delta)}{\alpha} = 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{22}$$

$$\limsup_{[\alpha, \Delta'] \to [+0, \Delta]} \frac{\overline{o}_H(x, \alpha\Delta')}{\alpha} \ge 0, \quad \forall \Delta \in \mathbb{R}^n, \tag{23}$$

$$\liminf_{[\alpha, \Delta'] \to [+0, \Delta]} \frac{\underline{o}_H(x, \alpha\Delta')}{\alpha} \le 0, \quad \forall \Delta \in \mathbb{R}^n. \tag{24}$$

First Order Necessary and Sufficient Conditions for an Unconstrained Optimum

Let a function f be defined on an open set $X \subset \mathbf{R}^n$, Ω be a subset of X. A point $x^* \in \Omega$ is called a *local minimum point* (*local minimizer*) of the function f on the set Ω if there exists $\delta > 0$ such that

$$f(x) \ge f(x^*), \quad \forall x \in \Omega \cap B_\delta(x^*),$$

where $B_\delta(x^*) = \{x \in \mathbf{R}^n : \| x - x^* \| \le \delta\}$. If $\delta = +\infty$, then the point x^* is called a *global minimum point* (*global minimizer*) of f on Ω. A point $x^* \in \Omega$ is called a *strict local minimum point* (*strict local minimizer*) of f on Ω if there exists $\delta > 0$ such that

$$f(x) > f(x^*), \quad \forall x \in \Omega \cap B_\delta(x^*), \quad x \ne x^*.$$

Analogously one can define local, global and strict *local maximum points* (*maximizers*) of f on Ω.

It may happen that the set of local (global, strict local) minimizers (maximizers) is empty.

If $\Omega = X$ then the problem of finding a minimum or a maximum of f on X is called an *unconstrained optimization problem.*

Proposition 8 *Let a function f be Dini (Hadamard) directionally differentiable on X. For a point $x^* \in \operatorname{dom} f$ to be a local or global minimizer of f on X it is necessary that*

$$f_D'(x^*, g) \ge 0, \quad \forall g \in \mathbb{R}^n, \tag{25}$$

$$\left(f_H'(x^*, g) \ge 0 \quad \forall g \in \mathbb{R}^n \right). \tag{26}$$

If f is Hadamard d.d. at x^* and

$$f_H'(x^*, g) > 0, \quad \forall g \in \mathbb{R}^n, \quad g \ne 0_n, \tag{27}$$

then x^* is a strict local minimizer of f.

Here $0_n = (0, \dots, 0)$ is the zero element of $\mathbf{R}^n$.

Proposition 9 *Let f be Dini (Hadamard) d.d. on X. For a point $x^{**} \in \operatorname{dom} f$ to be a local or global maximizer of f on X it is necessary that*

$$f_D'(x^{**}, g) \le 0, \quad \forall g \in \mathbb{R}^n, \tag{28}$$

$$(f_H'(x^{**}, g) \le 0, \quad \forall g \in \mathbb{R}^n). \tag{29}$$

If f is Hadamard d.d. at x^{**} and

$$f'_H(x^{**}, g) < 0, \quad \forall g \in \mathbb{R}^n, \quad g \neq 0_n, \tag{30}$$

then x^{**} is a strict local maximizer of f.

Note that (26) implies (25), and (29) implies (28). In the smooth case $f'_H(x, g) = (f'(x), g)$ ($f'(x)$ being the gradient of f at x) and the conditions (27) and (30) are impossible. It means that the sufficient conditions (27) and (30) are essentially nonsmooth.

Proposition 10 *Let f be defined on an open set on X ⊂ $\mathbf{R}^n$. For a point $x^* \in$ dom f (i.e., $|f(x)| < +\infty$) to be a local or global minimizer of f on X it is necessary that*

$$f^{\downarrow}_D(x^*, g) \geq 0, \quad \forall g \in \mathbb{R}^n, \tag{31}$$

$$f^{\downarrow}_H(x^*, g) \geq 0, \quad \forall g \in \mathbb{R}^n. \tag{32}$$

If

$$f^{\downarrow}_H(x^*, g) > 0, \quad \forall g \in \mathbb{R}^n, \quad g \neq 0_n, \tag{33}$$

then x^* is a strict local minimizer of f.

Note that (32) implies (31) but (31) does not necessarily imply (32).

Proposition *Let f be defined on an open set on X ⊂ $\mathbf{R}^n$. For a point $x^{**} \in$ dom f to be a local or global maximizer of f on X it is necessary that*

$$f^{\uparrow}_D(x^{**}, g) \leq 0, \quad \forall g \in \mathbb{R}^n \tag{34}$$

and

$$f^{\uparrow}_H(x^{**}, g) \leq 0, \quad \forall g \in \mathbb{R}^n. \tag{35}$$

If

$$f^{\uparrow}_H(x^{**}, g) < 0, \quad \forall g \in \mathbb{R}^n, \quad g \neq 0_n, \tag{36}$$

then x^{**} is a strict local maximizer of f.

The condition (35) implies (34) but (34) does not necessarily imply (35).

Remark 12 Observe that the conditions for a minimum are different from the conditions for a maximum.

A point x^* satisfying the conditions (25) or (31) is called a *Dini* inf-*stationary point* of f, while a point x^* satisfying (26) or (32) is called an *Hadamard* inf-*stationary point*. A point x^{**} satisfying the conditions (28) or (34) is called a *Dini* sup-*stationary point* of f, while a point x^{**} satisfying (28) or (35) is called an *Hadamard* sup-*stationary point*.

Remark 13 Note that the function f is not assumed to be continuous or even finite-valued.

Let $x_0 \in \operatorname{dom} f$ and assume that the condition (31) does not hold, i. e. x_0 is not a Dini inf-stationary point. If $g_0 \in \mathbf{R}^n$, $\|g_0\| = 1$,

$$f^{\downarrow}_D(x_0, g_0) = \inf_{\|g\|=1} f^{\downarrow}_D(x_0, g),$$

then g_0 is called a *Dini steepest descent direction* of f at x_0 ($\|g\|$ is the Euclidean norm).

If (32) does not hold and if $g_0 \in \mathbf{R}^n$, $\|g_0\| = 1$,

$$f^{\downarrow}_H(x_0, g_0) = \inf_{\|g\|=1} f^{\downarrow}_H(x_0, g),$$

then g_0 is called an *Hadamard steepest descent direction* of f at x_0.

Analogously if x_0 is not a Dini sup-stationary point and if $g^0 \in \mathbf{R}^n$, $\|g^0\| = 1$,

$$f^{\uparrow}_D(x_0, g^0) = \sup_{\|g\|=1} f^{\uparrow}_D(x_0, g),$$

then g^0 is called a *Dini steepest ascent direction* of f at x_0.

If x_0 is not an Hadamard sup-stationary point of f (i. e. (35) does not hold) and if $g^0 \in \mathbf{R}^n$, $\| g^0 \| = 1$,

$$f^{\uparrow}_H(x_0, g^0) = \sup_{\|g\|=1} f^{\uparrow}_H(x_0, g),$$

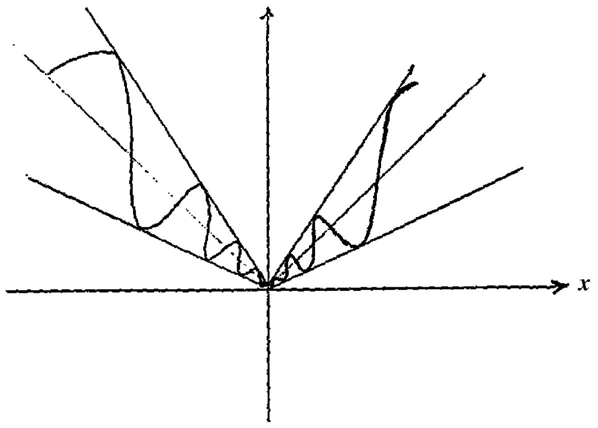
then g^0 is called an *Hadamard steepest ascent direction* of f at x_0.

Of course it is possible that there exist many steepest descent or/and steepest ascent directions of f at x_0.

It may also happen that some direction is a direction of steepest ascent and, at the same time, a direction of steepest ascent as well (which is impossible in the smooth case).

Example 14 Let $X = \mathbf{R}$,

$$f(x) = \begin{cases} |x| + \frac{1}{2} x \sin \frac{1}{x}, & x \neq 0, \\ 0, & x = 0. \end{cases}$$



Dini and Hadamard Derivatives in Optimization, Figure 1

Take $x_0 = 0$. It is clear that (see Fig. 1):

$$f_D^{\uparrow}(x_0, g) = |g| + \frac{1}{2}|g| = \frac{3}{2}|g|,$$
$$f_D^{\downarrow}(x_0, g) = |g| - \frac{1}{2}|g| = \frac{1}{2}|g|.$$

As $X = \mathbf{R}$, the Hadamard derivatives coincide with the Dini ones (see Remark 1).

$$f_D^{\downarrow}(x_0, g) > 0, \quad \forall g \neq 0,$$

we may conclude (see (32)) that x_0 is a strict local minimizer (in fact it is a global minimizer but our theory does not allow us to claim this).

Note that $f_D^{\uparrow}$ and $f_D^{\downarrow}$ are positively homogeneous (see (7)), therefore it is sufficient to consider (in $\mathbf{R}$) only two directions: $g_1 = 1$ and $g_2 = -1$.

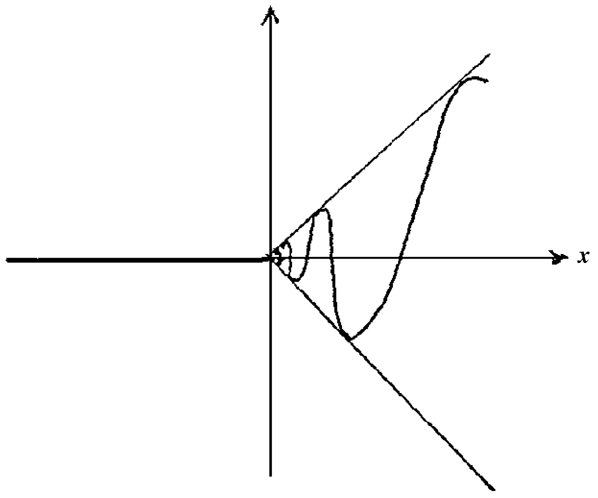
Example 15 Let $X = \mathbf{R}$, $x_0 = 0$,

$$f(x) = \begin{cases} x \sin \frac{1}{x}, & x > 0, \\ 0, & x \leq 0. \end{cases}$$

It is clear that (see Fig. 2) that

$$f_D^{\uparrow}(x_0, g) = \begin{cases} |g|, & g > 0, \\ 0, & g \leq 0, \end{cases}$$
$$f_D^{\downarrow}(x_0, g) = \begin{cases} -|g|, & g > 0, \\ 0, & g \leq 0. \end{cases}$$

Neither the condition (25) nor the condition (31) holds, therefore we conclude that x_0 is neither a local



Dini and Hadamard Derivatives in Optimization, Figure 2

minimizer nor a local maximizer. Since

$$\max_{\|g\|=1} f_D^{\uparrow}(x_0, g)$$
$$= \max\{f_D^{\uparrow}(x_0, +1), f_D^{\uparrow}(x_0, -1)\}$$
$$= \max\{1, 0\} = f_D^{\uparrow}(x_0, +1) = +1,$$

then $g_1 = +1$ is a steepest ascent direction.

Since

$$\min_{\|g\|=1} f_D^{\downarrow}(x_0, g)$$
$$= \min\{f_D^{\downarrow}(x_0, +1), f_D^{\downarrow}(x_0, -1)\}$$
$$= \min\{-1, 0\} = f_D^{\downarrow}(x_0, +1) = -1,$$

then $g_1 = +1$ is a steepest descent direction as well.

Conditions for a Constrained Optimum

Let a function f be defined on an open set $X \subset \mathbf{R}^n$, Ω be a subset of X. Let $x \in \Omega$, $|f(x)| < +\infty$, $g \in \mathbf{R}^n$. The limit

$$f_D^{\uparrow}(x, g; \Omega) = \limsup_{\substack{\alpha \downarrow 0 \\ x + \alpha g \in \Omega}} \frac{f(x + \alpha g) - f(x)}{\alpha} \tag{37}$$

is called the *Dini conditional upper derivative* of the function f at the point x in the direction g with respect to Ω. If no sequence $\{\alpha_k\}$ exists such that $\alpha_k \downarrow 0$, $x + \alpha_k g \in \Omega$ for all k, then, by definition, we set $f_D^{\uparrow}(x, g; \Omega) = -\infty$.

The limit

$$f_{\mathrm{D}}^{\downarrow}(x,g;\Omega)=\liminf_{\substack{\alpha\downarrow 0\\ x+\alpha g\in\Omega}}\frac{f(x+\alpha g)-f(x)}{\alpha} \tag{38}$$

is called the *Dini conditional lower derivative* of the function f at the point x in the direction g with respect to Ω. If no sequence $\{\alpha_k\}$ exists such that $\alpha_k\downarrow 0$, $x+\alpha_k g\in\Omega$ for all k, then, by definition, we set $f_{\mathrm{D}}^{\downarrow}(x,g;\Omega)=+\infty$.

The limit

$$f_{\mathrm{H}}^{\uparrow}(x,g;\Omega)=\limsup_{\substack{[\alpha,g']\to[+0,g]\\ x+\alpha g'\in\Omega}}\frac{f(x+\alpha g')-f(x)}{\alpha} \tag{39}$$

is called the *Hadamard conditional upper derivative* of the function f at the point x in the direction g with respect to Ω. If no sequences $\{\alpha_k\}$, $\{g_k\}$ exist such that $[\alpha_k, g_k]\to[+0,g]$, $x+\alpha_k g_k\in\Omega$ for all k, then, by definition, we set $f_{\mathrm{H}}^{\uparrow}(x,g;\Omega)=-\infty$.

The limit

$$f_{\mathrm{H}}^{\downarrow}(x,g;\Omega)=\liminf_{\substack{[\alpha,g']\to[+0,g]\\ x+\alpha g'\in\Omega}}\frac{f(x+\alpha g')-f(x)}{\alpha} \tag{40}$$

is called the *Hadamard conditional lower derivative* of f at x in the direction g with respect to Ω. If no sequences $\{\alpha_k\}$, $\{g_k\}$ exist such that $[\alpha_k, g_k]\to[+0,g]$, $x+\alpha_k g_k\in\Omega$ for all k, then, by definition, we set $f_{\mathrm{H}}^{\downarrow}(x,g;\Omega)=+\infty$.

Proposition 16 (see [1]) *For a point $x^*\in\Omega$ and such that $|f(x^*)|<\infty$ to be a local or global minimizer of f on Ω it is necessary that*

$$f_{D}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in\mathbb{R}^n, \tag{41}$$

$$f_{H}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in\mathbb{R}^n. \tag{42}$$

Furthermore, if

$$f_{H}^{\downarrow}(x^*,g;\Omega)>0,\quad \forall g\in\mathbb{R}^n,\quad g\neq 0_n, \tag{43}$$

then x^* is a strict local minimizer of f on Ω.

A point $x^*\in\Omega$ satisfying (41) ((42)) is called a *Dini* (*Hadamard*) inf-*stationary point* of f on Ω.

Proposition 17 *For a point $x^{**}\in\Omega$ and such that $|f(x^{**})|<\infty$ to be a local or global minimizer of f on Ω it is necessary that*

$$f_{D}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in\mathbb{R}^n, \tag{44}$$

$$f_{H}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in\mathbb{R}^n\,. \tag{45}$$

If

$$f_{H}^{\uparrow}(x^{**},g;\Omega)<0,\quad \forall g\in\mathbb{R}^n,\quad g\neq 0, \tag{46}$$

then x^{**} is a strict local maximizer of f on Ω.

A point $x^{**}\in\Omega$ satisfying (44) ((45)) is called a *Dini* (*Hadamard*) sup-*stationary point* of f on Ω.

The condition (41) is equivalent to

$$f_{\mathrm{D}}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in K(x^*,\Omega), \tag{47}$$

where

$$K(x^*,\Omega)=\left\{g\in\mathbb{R}^n:\ \exists\alpha_k:\ \begin{array}{c}\alpha_k\downarrow 0,\\ x^*+\alpha_k g\in\Omega,\\ \forall k\end{array}\right\}. \tag{48}$$

Analogously, the condition (44) is equivalent to

$$f_{\mathrm{D}}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in K(x^{**},\Omega). \tag{49}$$

The condition (42) is equivalent to

$$f_{\mathrm{H}}^{\downarrow}(x^*,g;\Omega)\geq 0,\quad \forall g\in \Gamma(x^*,\Omega), \tag{50}$$

where

$$\Gamma(x^*,\Omega)=\left\{g\in\mathbb{R}^n:\ \exists\{[\alpha_k,g_k]\}:\ \begin{array}{c}[\alpha_k,g_k]\to[+0,g],\\ x^*+\alpha_k g_k\in\Omega,\\ \forall k\end{array}\right\}. \tag{51}$$

Analogously, the condition (45) is equivalent to

$$f_{\mathrm{H}}^{\uparrow}(x^{**},g;\Omega)\leq 0,\quad \forall g\in \Gamma(x^{**},\Omega). \tag{52}$$

Note that the cones $K(x^*,\Omega)$ and $K(x^{**},\Omega)$ are not necessarily closed, while the cones $\Gamma(x^*,\Omega)$ and $\Gamma(x^{**},\Omega)$ are the Bouligand cones to Ω at x^* and x^{**}, respectively, and therefore always closed.

Now it is possible to define conditional steepest ascent and descent directions.

Remark 18 It is also possible (see [3, p. 156]) to define the Dini and Hadamard conditional directional derivatives as follows:

$$f_{\mathrm{D}}'(x,g;\Omega)=\lim_{\substack{\alpha\downarrow 0\\ x+\alpha g\in\Omega}}\frac{f(x+\alpha g)-f(x)}{\alpha}, \tag{53}$$

$$f'_{\mathrm{H}}(x, g; \Omega) = \lim_{\substack{[\alpha, g'] \to [+0, g] \\ x + \alpha g' \in \Omega}} \frac{f(x + \alpha g') - f(x)}{\alpha}. \quad (54)$$

A function f is called *Dini* (*Hadamard*) *conditionally differentiable* at x in a direction g if the limit in (53) ((54)) exists and is finite.

Remark 19 The conditional directional derivatives defined by (37)–(40) essentially depend on the set Ω.

In some cases it is possible to 'separate' the function f and the set Ω in the necessary conditions (47), (49), (50) and (52). For example, if f is Lipschitz and directionally differentiable at x, then

$$\begin{aligned} & f_{\mathrm{D}}^{\uparrow}(x, g; \Omega) = f_{\mathrm{D}}^{\downarrow}(x, g; \Omega) \\ & = f_{\mathrm{H}}^{\uparrow}(x, g; \Omega) = f_{\mathrm{H}}^{\downarrow}(x, g; \Omega) \\ & = f'(x, g) \quad \forall g \in K(x, \Omega). \end{aligned}$$

In this case the derivatives at the left-hand sides of (47), (49) and (50), (52) should be replaced by $f'(x^*, g)$ or $f'(x^{**}, g)$ respectively.

Note that if $g \in \Gamma(x, g)$ but $g \notin K(x,g)$ then $f_{\mathrm{D}}^{\uparrow}(x, g; \Omega)$ and $f_{\mathrm{D}}^{\downarrow}(x, g; \Omega)$ are not finite, by definition, while

$$f_{\mathrm{D}}^{\uparrow}(x, g; \Omega) = f_{\mathrm{D}}^{\downarrow}(x, g; \Omega) = f'(x, g).$$

Remark 20 The necessary optimality conditions for unconstrained and constrained optimization problems described above can be used to construct numerical methods for finding corresponding (inf- or sup-stationary) points.

For special classes of functions (e. g., convex, concave, max-type, minmax-type, quasidifferentiable functions), the derivative (3) has a more 'constructive' form and therefore the conditions (25)–(36) and (41)–(46) take also more 'constructive' forms (see, e. g., [2]).

Remark 22 The limits in (4), (5), (6), (39) and (40) are taken if

$$[\alpha, g'] \to [+0, g]. \quad (55)$$

Sometimes, in the literature instead of this relation one can see two relations

$$\alpha \to +0, \quad g' \to g. \quad (56)$$

It was demonstrated in [4] that the limits resulting from (55) and (56) do not necessarily coincide. This warning should be taken into account.

See also

- ▶ Global Optimization: Envelope Representation
- ▶ Nondifferentiable Optimization
- ▶ Nondifferentiable Optimization: Cutting Plane Methods
- ▶ Nondifferentiable Optimization: Minimax Problems
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Nondifferentiable Optimization: Parametric Programming
- ▶ Nondifferentiable Optimization: Relaxation Methods
- ▶ Nondifferentiable Optimization: Subgradient Optimization Methods

References

1. Demyanov VF, Di Pillo G, Facchinei F (1997) Exact penalization via Dini and Hadamard conditional derivatives. Optim Methods Softw 9:19–36
2. Demyanov VF, Rubinov AM (1995) Constructive nonsmooth analysis. P Lang, Frankfurt am Main
3. Dieudonne J (1969) Foundations of modern analysis. Acad Press, New York
4. Giannessi F (1995) A common understanding or a common misunderstanding. Numer Funct Anal Optim 16(9–10):1359–1363

Directed Tree Networks

THOMAS ERLEBACH[1], KLAUS JANSEN[2], CHRISTOS KAKLAMANIS[3], GIUSEPPE PERSIANO[4]

[1] Department of Computer Science, University of Leicester, Leicester, UK

[2] Institut für Informatik, Univerisität Kiel, Kiel, Germany

[3] RA CTI, University of Patras, Patras, Greece

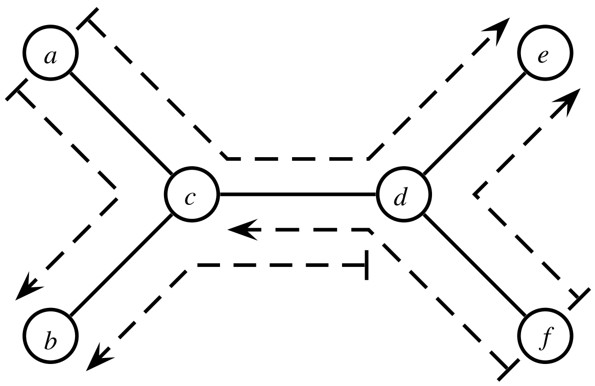
[4] Dipartimento di Informatica ed Appl., Università di Salerno, Fisciano, Italy

MSC2000: 05C85

Article Outline

Keywords and Phrases

Optical networks; Path coloring; Bipartite edge coloring; Approximation algorithms



Directed Tree Networks, Figure 1
Example path coloring instance

Technological developments in the field of optical communication networks using *wavelength-division multiplexing* have triggered intensive research in an optimization problem concerning the assignment of colors to paths in a directed tree. Here, the term *directed tree* refers to the graph obtained from an undirected tree by replacing each undirected edge by two directed edges with opposite directions. This *path coloring problem* was first studied by M. Mihail, C. Kaklamanis and S. Rao [19]. An instance of it is given by a directed tree $T = (V, E)$ and a set $P = \{p_1, \ldots, p_t\}$ of directed simple (i. e., not visiting any vertex twice) paths in T, where each path is specified by an ordered pair of vertices (start vertex and end vertex). The task is to assign colors to the given paths such that paths receive different colors if they share a directed edge. The goal is to minimize the number of different colors used. For given $T = (V, E)$ and P, let $L(e)$ denote the load on directed edge $e \in E$, i. e., the number of paths containing e. Obviously, the maximum load $L = \max_{e \in E} L(e)$ is a lower bound on the number of colors in an optimal coloring. Consider Fig. 1 for an example of a tree with six vertices and paths from a to e, from f to e, from f to c, from d to b, and from a to b. A possible valid coloring is to assign these paths the colors 1, 2, 1, 2, and 3, respectively. The maximum load of the paths is 2, because 2 paths use the edge (d, c). It is not possible to color these paths with 2 colors, because the *conflict graph* of the paths (a graph with a vertex for each path and an edge between vertices if the corresponding paths share an edge) is a cycle of length 5. Hence, the coloring with three colors is an optimal coloring.

The path coloring problem models the *assignment of wavelengths* to directed connection requests in *all-optical networks* with *tree topology*. In such networks data is transmitted in optical form via laser beams [13]. Two adjacent nodes of the network are connected by a pair of optical fiber links, one for each direction. When wavelength-division multiplexing is used, multiple signals can be transmitted over the same link if they use different wavelengths, and the nodes are capable of switching an incoming signal onto any outgoing link depending on the wavelength of the signal. However, the wavelength of a signal cannot be changed, and every connection uses the same wavelength on the whole transmitter-receiver path. If two signals using the same wavelength are transmitted over the same directed link, the data is lost due to interference. The number of available wavelengths is called *optical bandwidth*, and it is a scarce resource. Therefore, one is interested in minimizing the number of wavelengths necessary to route a given set of requests. This optimization problem corresponds to the path coloring problem defined above: paths correspond to connection requests, and colors correspond to wavelengths.

Complexity Results

Whereas the path coloring problem can be solved in linear time in *chain networks* as it is equivalent to *interval graph* coloring, it is *NP*-hard in directed tree networks. More precisely, it is *NP*-complete to decide whether a set of paths in a directed tree of arbitrary degree can be colored using at most 3 colors [8,9], and it is *NP*-

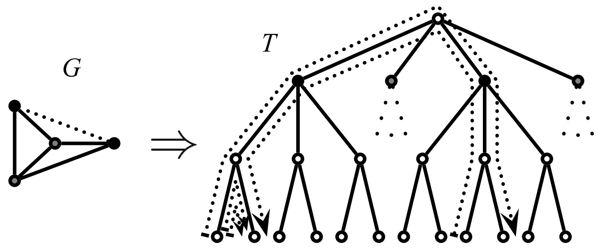
complete to decide whether a set of paths in a directed binary tree can be colored using at most k colors if k is part of the input [7,9,17]. The respective *reductions* will be outlined in the following. It should be remarked that the special case in which both the maximum degree of the tree as well as the maximum load of the paths are bounded by a constant can be solved optimally in polynomial time [7,9].

Reduction from Edge Coloring

It is an *NP*-complete problem to decide whether the edges of a given 3-regular undirected graph G can be colored with three colors such that edges receive different colors if they are incident to the same vertex [14]. Let G be a 3-regular undirected graph with n vertices and m edges. A directed tree T with $n' = 10n + 1$ vertices and a set P of $t = 4m$ paths in T can be constructed in polynomial time such that the paths in T can be colored using three colors if and only if the edges of G can be colored with three colors. T consists of a root r, one child c_v of the root for every vertex v of G, three children c_v^1, c_v^2 and c_v^3 of every c_v, and two children $c_v^{i,1}$ and $c_v^{i,2}$ of every c_v^i. For each edge $e = \{v, w\}$ of G, four paths in T are created: one from $c_v^{i,1}$ to $c_w^{j,2}$ and one from $c_w^{j,1}$ to $c_v^{i,2}$, called real paths, and two copies of the path from $c_v^{i,1}$ to $c_v^{i,2}$, called blockers. Here i and j are chosen such that the subtree rooted at c_v^i resp. c_w^j is not used by any paths other than the paths created for this particular edge e. Figure 2 shows an example of a 3-regular graph G, the constructed tree T (two of the four subtrees are represented by dotted triangles), and the paths created for the edge between the black vertices of G.

If the paths in T are to be colored with three colors, the blockers ensure that the two real paths corresponding to e receive the same color and, therefore, this color cannot be used by any other real path corresponding to an edge incident to v or w. If there exists a 3-coloring of the paths in T, a 3-coloring of the edges of G can be obtained by assigning each edge the color of its corresponding real paths. On the other hand, if there exists a 3-coloring of the edges of G, a 3-coloring of the paths in T can be obtained by assigning the real paths corresponding to edge e the same color as e and coloring the blockers with the remaining two colors. Hence, a solution to the path coloring problem in T would also solve the *edge coloring* problem in G.

Since it has just been proved *NP*-complete to decide whether paths in a directed tree can be colored with three colors, it follows that there cannot be an approximation algorithm for path coloring with absolute *approximation ratio* $< 4/3$ unless $P = NP$.



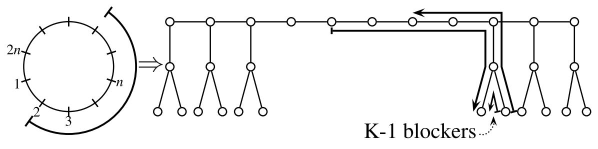
Directed Tree Networks, Figure 2
Reduction from edge coloring

Reduction from Arc Coloring

The *NP*-complete *arc coloring* problem [12] is to decide for a given set of n arcs $A_1, \ldots, A_n$ on a circle and a given integer k whether the arcs can be colored with k colors such that arcs receive different colors if they intersect. Without loss of generality, assume that each arc is specified by a pair (a_i, b_i) with $a_i \neq b_i$ and $1 \leq a_i, b_i \leq 2n$. The span of arc A_i is $sp(A_i) = \{a_i + 1, a_i + 2, \ldots, b_i\}$ if $a_i < b_i$ and $sp(A_i) = \{a_i + 1, \ldots, 2n, 1, 2, \ldots, b_i\}$ if $a_i > b_i$. Two arcs A_i and A_j intersect iff $sp(A_i) \cap sp(A_j) \neq \emptyset$. Note that one can view the arc coloring problem as a path coloring problem on a cycle.

If a number is contained in the span of more than k arcs, then the arcs can surely not be colored with k colors and the answer to this instance of arc coloring is *no*. Otherwise, one can assume that every number i, $1 \leq i \leq 2n$ is contained in the span of exactly k arcs; if this were not the case, one could simply add arcs of the form $(i, i + 1)$ until the condition holds, without changing the answer of the coloring problem.

Now consider a chain of n vertices $v_1, v_2, \ldots, v_n$. Imagine the chain drawn from left to right, with v_1 the start vertex at its left end. The directed edges from left to right followed by the directed edges from right to left make up a cycle of length $2n$. The given circular arcs can be translated into directed paths on this cycle such that two paths share a directed edge iff the corre-



Directed Tree Networks, Figure 3
Reduction from arc coloring

sponding arcs intersect, but these paths do not yet constitute a valid path coloring problem because some of the paths are not simple: an arc $(1, 2n)$ would correspond to a path running from v_1 to v_n and back to v_1, for example. Nevertheless, it is possible to obtain a valid instance of the path coloring problem by splitting paths that are not simple into two or three simple paths and by using blockers to make sure that the paths derived from one non-simple path must receive the same color in any valid k-coloring.

For this purpose, extend the chain by adding k vertices on both ends, resulting in a chain of length $n + 2k$. Connect each of the newly added vertices to a distinct subtree consisting of a new vertex with two leaf children. The resulting network is a binary tree T. If a path arrives at vertex v_n coming from the left (i. e., from v_{n-1}) and "turns around" to revisit v_{n-1}, divide the path into two: one coming from the left, passing through v_n and ending at the left leaf of one of the subtrees added on the right side of the chain; the other one starting at the right leaf of that subtree, passing through v_n and continuing left. In addition, add $k - 1$ blockers in that subtree, i. e., paths from the right leaf to the left leaf. Observe that there are no more than k paths containing v_n as an inner vertex, and a different subtree can be chosen for each of these paths. A symmetric splitting procedure is applied to the paths that contain v_1 as an inner vertex, i. e., the paths that arrive at v_1 coming from the right (i. e., from v_2) and "turn around" to revisit v_2. This way, all non-simple paths are split into two or three simple paths, and a number of blockers are added.

The resulting set of paths in T can be colored with k colors if and only if the original arc coloring instance is a *yes*-instance. The blockers ensure that all paths corresponding to the same arc receive the same color in any k-coloring. Hence, a k-coloring of the paths can be used to obtain a k-coloring of the arcs by assigning each arc the color of its corresponding paths. Also, a k-coloring of the arcs can be turned into a k-coloring of the paths by assigning all paths corresponding to an arc the same color as the arc and by coloring the blockers with the remaining $k - 1$ colors. This shows that the decision version of the path coloring problem is *NP*-complete already for binary trees.

Approximation Algorithms

Since the path coloring problem in directed tree networks is *NP*-hard, one is interested in polynomial-time *approximation algorithms* with provable *performance guarantee*. All such approximation algorithms that have been developed so far belong to the class of *greedy algorithms*. A greedy algorithm picks a start vertex s in the tree T and assigns colors to the paths touching (starting at, ending at, or passing through) s first. Then it visits the remaining vertices of the tree in some order that ensures that the current vertex is adjacent to a previously visited vertex; for example, a *depth-first search* can be used to obtain such an order. When the algorithm processes vertex v, it assigns colors to all paths touching v without changing the color of paths that have been colored at a previous vertex. Each such step is referred to as *coloring extension*. Furthermore, the only information about the paths touching the current vertex that the algorithm considers is which edges incident to the current vertex they use. To emphasize this latter property, greedy algorithms are sometimes referred to as *local greedy algorithms*.

Whereas all greedy algorithms follow this general strategy, individual variants differ with respect to the solution to the coloring extension substep. The best known algorithm was presented by T. Erlebach, K. Jansen, C. Kaklamanis, and P. Persiano in [11,16] (see also [10]). It colors a set of paths with maximum load L in a directed tree network of arbitrary degree with at most $\lceil 5L/3 \rceil$ colors. In the next section this will be shown to be best possible in the class of greedy algorithms.

For the sake of clarity, assume that the load on all edges is exactly L and that L is divisible by 3. The algorithm maintains two *invariants*: (a) the number of colors used is at most $5L/3$, and (b) for each pair of directed edges with opposite directions the number of colors used to color paths going through either of these edges is at most $4L/3$. First, the algorithm picks a leaf s

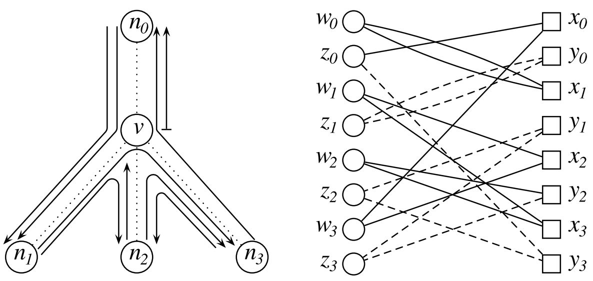
of T as the start vertex and colors all paths starting or ending at s using at most L colors. Therefore, the invariants are satisfied initially. It remains to show that they still hold after a coloring extension step if they were satisfied at the beginning of this step.

Reduction to Constrained Bipartite Edge Coloring

The coloring extension problem at a current vertex v is reduced to a *constrained edge coloring* problem in a *bipartite graph* G_v with left vertex set V_1 and right vertex set V_2. This reduction was introduced by M. Mihail, C. Kaklamanis and S. Rao in [19]. Let $n_0, n_1, \dots, n_k$ be the neighbors of v in T, and let n_0 be the unique neighbor that was processed before v. For every neighbor n_i of v the graph G_v contains four vertices: vertices w_i and z_i in V_1, and vertices x_i and y_i in V_2. Vertex w_i is said to be opposite x_i, and z_i is opposite y_i. A pair of opposite vertices is called a *line* of G_v. A line *sees* a color if it appears on an edge incident to a vertex of that line. For every path touching v there is one edge in G_v: an edge (w_i, x_j) for each path coming from n_i, passing through v and going to n_j; an edge (w_i, y_i) for each path coming from n_i and ending at v; and an edge (z_i, x_i) for each path starting at v and going to n_i.

It is easy to see that coloring the paths touching v is equivalent to coloring the edges of G_v. Note that the vertices w_i and x_i have degree L in G_v, while the other vertices may have smaller degree. If this is the case, the algorithm adds dummy edges (shown dashed in Fig. 4) in order to make the graph L-regular.

As the paths that contain the edges (n_0, v) or (v, n_0) have been colored at a previous vertex, the edges incident to w_0 and x_0 are already colored with at most $4L/3$ colors by invariant (b). These edges are called *color-forced* edges. A color that appears on exactly one color-forced edge is a *single color*. A color that appears on two color-forced edges is a *double color*. Since there are at most $4L/3$ colors on $2L$ color-forced edges, there must be at least $2L/3$ double colors. Furthermore, one can assume that there are exactly $2L/3$ double colors and $2L/3$ single colors, because if there are too many double colors then it is possible to split an appropriate number of double colors into two single colors for the duration of the current coloring extension step. In order to maintain invariant (a), the algorithm must color the uncolored edges of G_v using at most $L/3$ new colors (colors not used on the color-forced edges). Invariant (b) is satisfied by ensuring that no line of G_v sees more than $4L/3$ colors.



Directed Tree Networks, Figure 4
Construction of the bipartite graph

Partition Into Matchings

G_v is an L-regular bipartite graph and its edges can thus be partitioned into L *perfect matchings* efficiently. Each *matching* is classified according to the colors on its two color-forced edges: SS-matchings contain two single colors, ST-matchings contain one single color and one double color, PP-matchings contain the same (preserved) double color on both color-forced edges, and TT-matchings contain two different double colors. Next, the L matchings are grouped into chains and cycles: a chain of length $\ell \geq 2$ is a sequence of ℓ matchings $M_1, \dots, M_\ell$ such that M_1 and M_ℓ are ST-matchings, $M_2, \dots, M_{\ell-1}$ are TT-matchings, and two consecutive matchings share a double color; a cycle of length $\ell \geq 2$ is a sequence of ℓ TT-matchings such that consecutive matchings as well as the first and the last matching share a double color. Obviously, the set of L matchings is in this way entirely partitioned into SS-matchings, chains, cycles, and PP-matchings. In addition, if a chain or cycle contains parallel color-forced edges, then the algorithm exchanges these edges in the respective matchings, thus dividing the original chain or cycle into a shorter sequence of the same type and an extra cycle.

Now the algorithm chooses *triplets*, i. e., groups of three matchings, and colors the uncolored edges of each triplet using at most one new color and at most four *active* colors. The active colors are selected among the colors on color-forced edges of that triplet, and a color is active in at most one triplet. The algorithm ensures that a line that sees the new color does not see one of the active colors of that triplet. This implies that no line of G_v sees more than $4L/3$ colors altogether, as required to maintain invariant (b).

Coloring of Triplets

The rules for choosing triplets ensure that each triplet contains two color-forced edges with single colors and four color-forced edges with double colors. Furthermore, most triplets are chosen such that one double color appears twice, and this double color as well as the two single colors can be reused without considering conflicts outside the triplet. V. Kumar and E.J. Schwabe proved in [18] that such triplets can be colored as required using three active colors and one new color. This coloring procedure can be sketched as follows. Partition the edges of the triplet into a matching on all vertices except w_0 and x_0 and a *gadget*, i. e., a subgraph in which w_0 and x_0 have degree 3 while all other vertices have degree 2. A gadget consists of a number of cycles of even length not containing w_0 or x_0 and either three disjoint paths from w_0 to x_0 or one path from w_0 to x_0, one path from w_0 to w_0, and one path from x_0 to x_0. A careful case analysis shows that the triplet can be colored by reusing the single colors and the double color to color the gadget and using a new color for the matching. If a partitioning into gadget and matching does not exist, the triplet contains a PP-matching and can be colored using the double color of the PP-matching for the uncolored edges of the PP-matching and a single color and a new color for the uncolored edges of the cycle cover consisting of the other two matchings.

In the following, the terms *even sequence* and *odd sequence* refer to sequences of TT-matchings of even resp. odd length such that consecutive matchings share a double color. Note that an even sequence can be grouped into triplets by combining two consecutive matchings of the sequence with an SS-matching as long as SS-matchings are available and combining each remaining TT-matching with a chain of length 2. There are always enough SS-matchings or chains of length 2 because the ratio between color-forced edges with double colors and color-forced edges with single colors is 2 : 1 in G_v initially and remains the same after extracting triplets. Similarly, an odd sequence can be grouped into triplets if there is at least one chain of length 2, which can be used to form a triplet with the first matching of the sequence, leaving an even sequence behind.

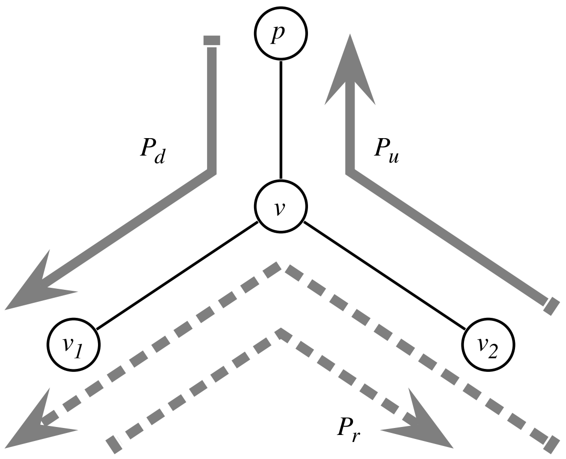
Selection of Triplets

Now the rules for selecting triplets are as follows. From chains of odd length, combine the first two matchings and the last matching to form a triplet. The remainder of the chain (if non-empty) is an even sequence and can be handled as described above. Cycles of even length are even sequences and can be handled the same way. As long as there is a chain of length 2 left, chains of even length ≥ 4 and odd cycles can be handled, too. Pairs of PP-matchings can be combined with an SS-matching, single PP-matchings can be combined with chains of length 2. If there are two chains of even length ≥ 4, combine the first two matchings of one chain with the last matching of the other and the last two matchings of the first chain with the first matching of the other, leaving two even sequences behind. So far, all triplets contained a double color twice and could be colored as outlined above. What remains is a number of cycles of odd length, at most one chain of even length, at most one PP-matching, and some SS-matchings. To deal with these, it is necessary to form some triplets that contain four distinct double colors. However, it is possible to ensure that the set of color-forced edges of G_v (inside and outside the triplet) colored with one of these double colors does not contain parallel edges; T. Erlebach, K. Jansen, C. Kaklamanis and P. Persiano showed in [11] that such a triplet can be colored as required using its single colors, two of its double colors, and one new color.

In the end, the entire graph G_v has been partitioned into triplets, and each triplet has been colored using at most one new color and such that a line that sees a new color in a triplet does not see one of the active colors of that triplet. Hence, invariants (a) and (b) hold at the end of the coloring extension step, and once the coloring extension step has been performed for all vertices of T all paths have received one of $\lceil 5L/3 \rceil$ colors. Since the number OPT of colors necessary in an optimal coloring is at least L, this implies that the algorithm uses at most $\lceil 5OPT/3 \rceil$ colors to color the paths. From the lower bound in the next section it will be clear that the algorithm (and any other greedy algorithm) is not better than $5OPT/3$ in the worst case.

Note that greedy algorithms are well-suited for practical distributed implementation in optical networks: one node of the network initiates the wavelength assignment by assigning wavelengths to all connections going through that node; then it transfers control to its neighbors who can extend the assignment independently and in parallel, transferring control to their neighbors in turn once they are done.

It should be mentioned that simpler variants of greedy algorithms are known that are restricted to *binary trees* and color a given set of paths with load L using $\lceil 5L/3 \rceil$ colors. These algorithms do not make use of the reduction to constrained bipartite edge coloring [6,15].



Directed Tree Networks, Figure 5
Lower bound for greedy algorithms

Lower Bounds

Two kinds of *lower bounds* have been investigated for path coloring in directed tree networks. First, one wants to determine the best *worst-case performance guarantee* achievable by any greedy algorithm. Second, it is interesting to know how many colors are required even in an optimal coloring for a given set of paths with load L in the worst case.

Lower Bound for Greedy Algorithms

For a given local greedy algorithm A and positive integer L, an *adversary* can construct an instance of path coloring in a directed binary tree network such that A uses at least $\lfloor 5L/3 \rfloor$ colors while an optimal solution uses only L colors [15]. The construction proceeds inductively. As A considers only the edges incident to a vertex v when it colors the paths touching v, the adversary can determine how these paths should continue and introduce new paths not touching v depending on the coloring A produces at vertex v.

Assume that there are $\alpha_i L/2$ paths going through each of the directed edges between vertex v and its parent, and that these paths have been colored with $\alpha_i L$ different colors. Initially, this assumption can be satisfied for $\alpha_0 = 1$ by introducing L paths in either direction on the link between the start vertex picked by algorithm A and one of its neighbors and letting appropriately chosen $L/2$ of these paths start resp. end at that neighbor. Denote the set of paths coming down from the parent by P_d and let them continue to (pass through) the left child v_1 of v. Denote the set of paths going up to the parent by P_u and let them pass through the right child v_2 of v. Introduce a set P_ℓ of $(1 - \alpha_i/2)L$ paths coming from v_2 and going left to v_1, and a set P_r of L paths coming from v_1 and going right to v_2.

Algorithm A must use $(1 - \alpha_i/2)L$ new colors to color the paths in P_ℓ. No matter which colors it chooses for the paths in P_r, it will use at least $(1 + \alpha_i/4)L$ different colors on the connection between v and v_1 or on the connection between v and v_2. The best it can do with respect to minimizing the number of colors appearing between v and v_1 and between v and v_2 is to color $(1 - \alpha_i/2)L$ paths of P_r with colors used for P_ℓ, $\alpha_i L/4$ paths of P_r with colors used for P_d, and $\alpha_i L/4$ paths of P_r with colors used for P_u. In that case, it uses $(1 + \alpha_i/4)L$ colors on each of the downward connections of v. Any other assignment uses more colors on one of the downward connections.

If the algorithm uses at least $(1 + \alpha_i/4)L$ different colors for paths on, say, the connection between v and v_1, let $(1 + \alpha_i/4)L/2$ of the downward paths and equally many of the upward paths extend to the left child of v_1, such that all of these paths use different colors, and let the remaining paths terminate or begin at v_1. Now the inductive assumption holds for the left child of v_1 with $\alpha_{i+1} = 1 + \alpha_i/4$. Hence, the number of colors on a pair of directed edges can be increased as long as $\alpha_i < 4/3$. When $\alpha_i = 4/3$, $4L/3$ colors are used for the paths touching v and its parent, and algorithm A must use $L/3$ new colors to color the paths in P_ℓ, using $5L/3$ colors altogether.

The previous calculations have assumed that all occurring terms like $(1+\alpha_i/4)L/2$ are integers. If one takes the possibility of non-integral values into account and carries out the respective calculations for all cases, one can show that, for every L, every greedy algorithm can be forced to use $\lfloor 5L/3 \rfloor$ colors on a set of paths with maximum load L [15].

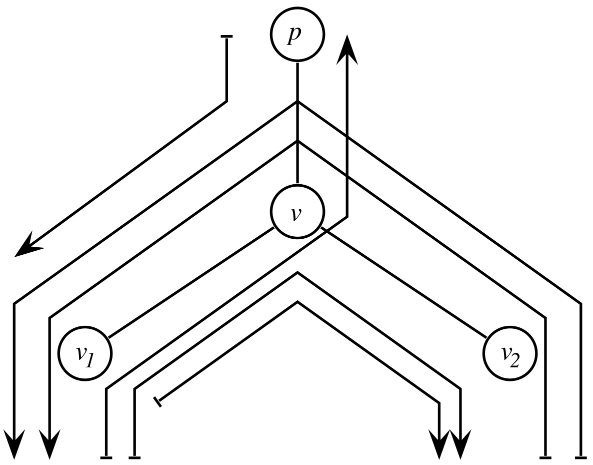
Furthermore, it is not difficult to show that the paths resulting from this worst-case construction for greedy algorithms can be colored optimally using only L colors. Hence, this yields also a lower bound of $\lfloor 5OPT/3 \rfloor$ colors for any greedy algorithm.

Lower Bounds for Optimal Colorings

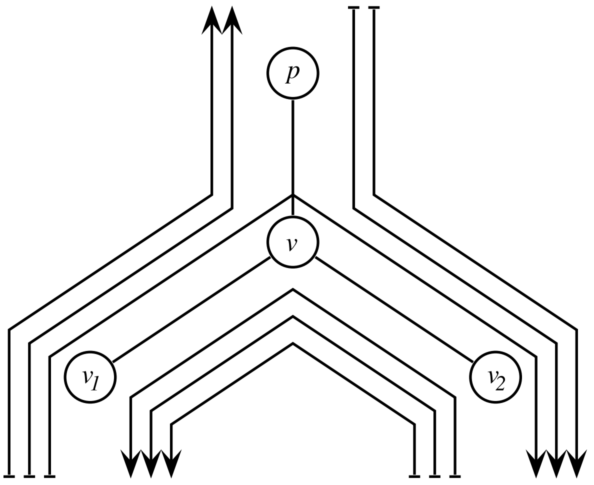
The instance of path coloring depicted in Fig. 1 consists of 5 paths in a binary tree with maximum load $L = 2$ such that even an optimal coloring requires 3 colors. Consider the instances of path coloring obtained from this instance by replacing each path by ℓ identical copies. Such an instance consists of 5ℓ paths with maximum load $L = 2\ell$, and an optimal coloring requires at least $\lceil 5\ell/2 \rceil = \lceil 5L/4 \rceil$ colors because no more than two of the given paths can be assigned the same color. Furthermore, $\lceil 5\ell/2 \rceil$ colors are also sufficient to color these instances: for example, if ℓ is even, use colors $1, \ldots, \ell$ for paths from a to e, colors $\ell+1, \ldots, 2\ell$ for paths from f to e, colors $1, \ldots, \ell/2$ and $2\ell + 1, \ldots, 5\ell/2$ for paths from f to c, colors $\ell/2 + 1, \ldots, 3\ell/2$ for paths from d to b, and colors $3\ell/2 + 1, \ldots, 5\ell/2$ for paths from a to b. Hence, for every even L there is a set of paths in a binary tree with load L such that an optimal coloring requires $\lceil 5L/4 \rceil$ colors [4,18].

While the path coloring instance with $L = 2$ and $OPT = 3$ could be specified easily, K. Jansen used a more involved construction to obtain an instance with $L = 3$ and $OPT = 5$ [15]. It makes use of three components as building blocks. Each component consists of a vertex v with its parent and two children and a specification of the usage of edges incident to v by paths touching v.

The root component ensures that at least 3 colors are used either on the left downward connection (extending below v_1) or on the right downward connection (extending below v_2). Each child of the root component is connected to a type A component, i. e., the child is identified with the parent vertex of a type A component and the corresponding paths are identified as well.

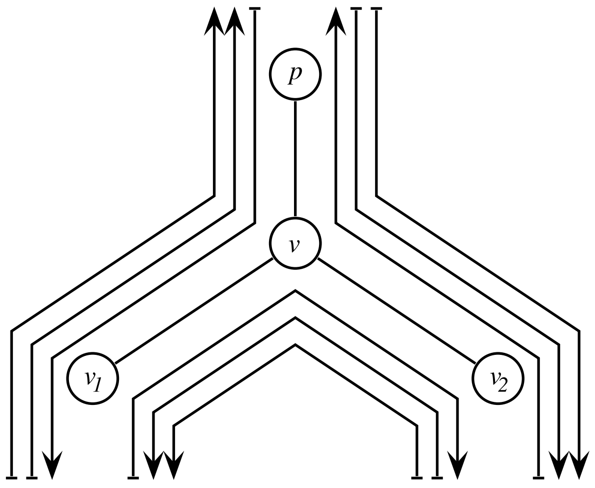


Directed Tree Networks, Figure 6
Root component



Directed Tree Networks, Figure 7
Type A component

Type A components have the property that, if the paths touching v and its parent are colored with 3 colors, at least 4 colors must be used either for the paths touching v and v_1 or for those touching v and v_2. (If the paths touching v and its parent are colored with 4 colors, the remaining paths of the type A component require even 5 colors.) Hence, there is at least one child in one of the two type A components below the root component such that the paths touching this child and its parent are colored with four colors.



Directed Tree Networks, Figure 8
Type B component

The final component used is of type B. It has the property that, if the paths touching v and its parent are colored with 4 colors, at least 4 colors must be used either for the paths touching v and v_1 or for those touching v and v_2. For certain arrangements of colors on the paths touching v and its parent, 5 colors are necessary. It is possible to arrange a number of type B components in a binary tree such that for any combination of four colors on paths entering the tree of type B components at its root, 5 colors are necessary to complete the coloring. Hence, if one attaches a copy of this tree of type B components to each of the children of a type A component, it is ensured that at least one of the trees will be entered by paths with four colors and consequently 5 colors are necessary to color all paths. Since the load on every directed edge is at most 3, this gives a worst-case example for path coloring in binary trees with $L = 3$ and $OPT = 5$.

Randomized Algorithms

In [1,2], V. Auletta, I. Caragiannis, C. Kaklamanis and G. Persiano presented a class of *randomized algorithms* for path coloring in directed tree networks. They gave a randomized algorithm that, with high probability, uses at most $7/5L + o(L)$ colors for coloring any set of paths of maximum load L on binary trees of height $o(L^{1/3})$. The analysis of the algorithm uses tail inequalities for hypergeometrical probability distributions such as *Azuma's inequality*. Moreover, they proved that no randomized greedy algorithm can achieve, with high probability, a performance ratio better than 3/2 for trees of height $\Omega(L)$ and better than $1.293 - o(1)$ for trees of constant height.

These results have been improved in [5] by I. Caragiannis, A. Ferreira, C. Kaklamanis, S. Pérennes, and H. Rivano, who gave a randomized approximation algorithm for bounded-degree trees that has approximation ratio $1.61 + o(1)$. The algorithm first computes in polynomial time an optimal solution for the fractional path coloring problem and then applies *randomized rounding* to obtain an integral solution.

Related Topics

A number of further results related to the path coloring problem in directed tree networks or in networks with different topology are known. The number of colors required for sets of paths that have a special form have been investigated, e. g., *one-to-all instances*, *all-to-all instances*, *permutations*, and *k-relations*. A survey of many of these results can be found in [4]. The undirected version of the path coloring problem has been studied by P. Raghavan and E. Upfal in [20]; here, the network is represented by an undirected graph and paths must receive different colors if they share an undirected edge. Approximation results for directed and undirected path coloring problems in *ring networks*, *mesh networks*, and arbitrary networks (all of these are *NP*-hard no matter whether the paths are fixed or can be chosen by the algorithm [7]) have been derived.

An on-line variant of path coloring was studied by Y. Bartal and S. Leonardo in [3]. Here, the algorithm is given connection requests one by one and must determine a path connecting the corresponding vertices and a color for this path without any knowledge of future requests. The worst-case ratio between the number of colors used by the *on-line algorithm* and that used by an optimal off-line algorithm with complete advance knowledge is the *competitive ratio*. In [3] on-line algorithms with competitive ratio $O(\log n)$ are presented for trees, trees of rings, and meshes with n vertices.

References

1. Auletta V, Caragiannis I, Kaklamanis C, Persiano P (2000) Randomized Path Coloring on Binary Trees. In: Jansen K,

Khuller S (eds) Approximation Algorithms for Combinatorial Optimization (APPROX 2000), LNCS, vol 1913. Springer, Berlin, pp 60–71

2. Auletta V, Caragiannis I, Kaklamanis C, Persiano P (2002) Randomized Path Coloring on Binary Trees. Theoret Comput Sci 289:355–399
3. Bartal Y, Leonardi S (1997) On-Line routing in all-optical networks. Proceedings of the 24th International Colloquium on Automata, Languages and Programming ICALP 97, LNCS, vol 1256. Springer, Berlin, pp 516–526
4. Beauquier B, Bermond J-C, Gargano L, Hell P, Perennes S, Vaccaro U (1997) Graph problems arising from wavelength-routing in all-optical networks. Proceedings of IPPS 97, Second Workshop on Optics and Computer Science (WOCS), Geneva
5. Caragiannis I, Ferreira A, Kaklamanis C, Pérennes S, Rivano H (2001) Fractional Path Coloring with Applications to WDM Networks. Proceedings of the 28th International Colloquium on Automata, Languages and Programming ICALP 01, LNCS, vol 2076. Springer, Berlin, pp 732–743
6. Caragiannis I, Kaklamanis C, Persiano P (1997) Bounds on optical bandwidth allocation on directed fiber tree topologies. Proceedings of IPPS 97, Second Workshop on Optics and Computer Science (WOCS), Geneva
7. Erlebach T, Jansen K (1997) Call scheduling in trees, rings and meshes. Proceedings of the 30th Hawaii International Conference on System Sciences HICSS-30, vol 1, IEEE Computer Society Press, Maui, pp 221–222
8. Erlebach T, Jansen K (1997) Scheduling of virtual connections in fast networks. Proceedings of the 4th Parallel Systems and Algorithms Workshop PASA 96, World Scientific Publishing, Jülich, pp 13–32
9. Erlebach T, Jansen K (2001) The complexity of path coloring and call scheduling. Theoret Comput Sci 255(1–2): 33–50
10. Erlebach T, Jansen K, Kaklamanis C, Mihail M, Persiano P (1999) Optimal Wavelength Routing on Directed Fiber Trees. Theoret Comput Sci 221(1–2):119–137
11. Erlebach T, Jansen K, Kaklamanis C, Persiano P (1998) An optimal greedy algorithm for wavelength allocation in directed tree networks. Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location. DIMACS Series Disc Math Theoret Comput Sci AMS 40:117–129
12. Garey MR, Johnson DS, Miller GL, Papadimitriou CH (1980) The complexity of coloring circular arcs and chords. SIAM J Algebraic Discrete Methods 1(2):216–227
13. Green PE (1991) The future of fiber-optic computer networks. IEEE Comput 24(9):78–87
14. Holyer I (1981) The NP-completeness of edge-coloring. SIAM J Comput 10(4):718–720
15. Jansen K (1997) Approximation results for wavelength routing in directed trees. Proceedings of IPPS 97, Second Workshop on Optics and Computer Science (WOCS), Geneva
16. Kaklamanis C, Persiano P, Erlebach T, Jansen K (1997) Constrained bipartite edge coloring with applications to wavelength routing. Proceedings of the 24th International Colloquium on Automata, Languages and Programming ICALP 97, LNCS, vol 1256, Bologna. Springer, Berlin, pp 493–504
17. Kumar SR, Panigrahy R, Russel A, Sundaram R (1997) A note on optical routing on trees. Inf. Process. Lett 62:295–300
18. Kumar V, Schwabe EJ (1997) Improved access to optical bandwidth in trees. Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms SODA 97, New Orleans. pp 437–444
19. Mihail M, Kaklamanis C, Rao S (1995) Efficient access to optical bandwidth. Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp 548–557
20. Raghavan P, Upfal E (1994) Efficient routing in all-optical networks. Proceedings of the 26th Annual ACM Symposium on Theory of Computing STOC 94, ACM SIGACT, Monteal. ACM Press, New York, pp 134–143

Direct Global Optimization Algorithm

DONALD R. JONES
General Motors Corp., Warren, USA

MSC2000: 65K05

Article Outline

Keywords

Global optimization; Black-box optimization; Nonsmooth optimization; Constraints; Deterministic optimization

For a *black-box global optimization* algorithm to be truly global, some effort must be allocated to global search, that is, search done primarily to ensure that potentially good parts of the space are not overlooked. On the other hand, to be efficient, some effort must also be placed on local search near the current best solution. Most algorithms either move progressively from global to local search (e. g., simulated annealing) or combine a fundamentally global method with a fundamentally local method (e. g., multistart, tunneling).

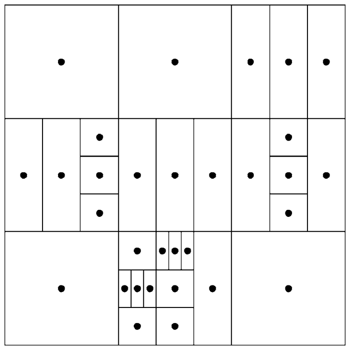
DIRECT introduces a new approach: in each iteration several search points are computed using all possible weights on local versus global search (how this is done will be made clear shortly). This approach eliminates the need for *'tuning parameters'* that set the balance between local and global search, resulting in an algorithm that is robust and easy-to-use.

DIRECT is especially valuable for *engineering optimization* problems. In these problems, the objective and constraint functions are often computed using time-consuming computer simulations, so there is a need to be efficient in the use of function evaluations. The problems may contain both continuous and integer variables, and the functions may be nonlinear, nonsmooth, and multimodal. While many algorithms address these problem features individually, DIRECT is one of the few that addresses them collectively. However, the versatility of DIRECT comes at a cost: the algorithm suffers from a curse of dimensionality that limits it to low-dimensional problems (say, no more than 20 variables).

The general problem solved by DIRECT can be written as follows:

$$\begin{cases} \min & f(x_1,\dots,x_n) \\ \text{s.t.} & g_1(x_1,\dots,x_n) \le 0, \\ & \quad\vdots \\ & g_m(x_1,\dots,x_n) \le 0, \\ & \ell_i \le x_i \le u_i, \\ & x_i \in I \text{ integer}. \end{cases}$$

To prove convergence, we must assume that the objective and constraint functions are continuous in the neighborhood of the optimum, but the functions can otherwise be nonlinear, nondifferentiable, nonconvex, and multimodal. While DIRECT does not explicitly handle equality constraints, problems with equalities can often be rewritten as problems with *inequality constraints* (either by replacing the equality with an inequality that becomes binding in the solution, or by using the equalities to eliminate variables). The set I in the above problem is the set of variables that are restricted to *integer* values. DIRECT works best when the integer variables describe an ordered quantity, such as the number of teeth on a gear. It is less effective when the integer variables are categorical.



Direct Global Optimization Algorithm, Figure 1

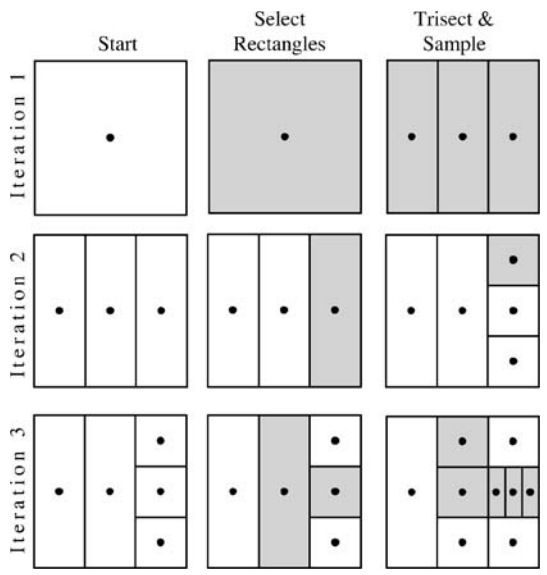
In what follows, we begin by describing how DIRECT works when there are no inequality and integer constraints. This basic version corresponds, with minor differences, to the originally published algorithm [2]. After describing the basic version, we then introduce extensions to handle inequality and integer constraints (this article is the first publication to document these extensions). We conclude with a step-by-step description of the algorithm.

The bounds on the variables limit the search to an n-dimensional hyper-rectangle. DIRECT proceeds by *partitioning* this rectangle into smaller rectangles, each of which has a 'sampled point' at its center, that is, a point where the functions have been evaluated. An example of such a partition for $n = 2$ is shown in Fig. 1.

We have drawn the rectangle as a square because later, whenever we measure distances or lengths, we will weight each dimension so that the original range $(u_i - \ell_i)$ has a weighted distance of one. Drawing the hyper-rectangle as a hyper-cube allows us to visualize relative lengths as they will be used in the algorithm.

Figure 2 shows the first three iterations of DIRECT on a hypothetical two-variable problem. At the start of each iteration, the space is partitioned into rectangles. DIRECT then selects one or more of these rectangles for further search using a technique described later. Finally, each selected rectangle is *trisected* along one of its long sides, after which the center points of the outer thirds are sampled. In this way, we sample two new points in the rectangle and maintain the property that every sampled point is at the center of a rectangle (this property would not be preserved if the rectangle were bisected).

At the beginning of iteration 1, there is only one rectangle (the entire space). The process of selecting



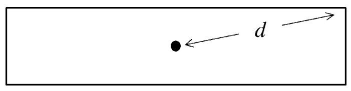
Direct Global Optimization Algorithm, Figure 2

rectangles is therefore trivial, and this rectangle is trisected as shown. At the start of iteration 2, the selection process is no longer trivial because there are three rectangles. In the example, we select just one rectangle, which is then trisected and sampled. At the start of iteration 3, there are 5 rectangles; in this example, two of them are selected and trisected.

The key step in the algorithm is the selection of rectangles, since this determines how search effort is allocated across the space. The trisection process and other details are less important, and we will defer discussion of them until later.

To motivate how DIRECT selects of rectangles, let us begin by considering the extremes of pure global search and pure local search. A pure global search strategy would select one of the biggest rectangles in each iteration. If this were done, all the rectangles would become small at about the same rate. In fact, if we always trisected one of the biggest rectangles, then after 3^{kn} function evaluations every rectangle would be a cube with side length 3^{-k}, and the sampled points would form a uniform grid. By looking everywhere, this pure global strategy avoids overlooking good parts of the space.

A pure local strategy, on the other hand, would sample the rectangle whose center point has the best objective function value. This strategy is likely to find good



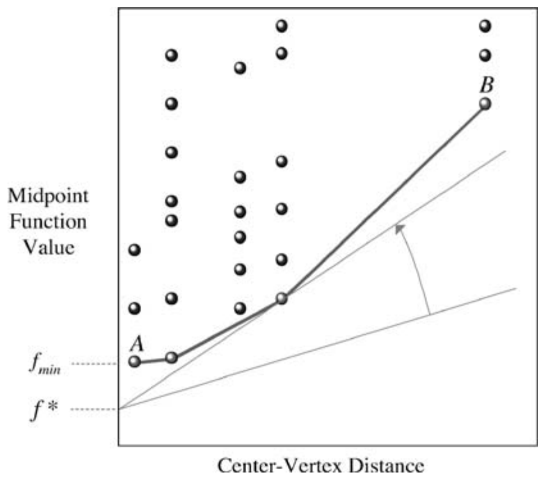
Direct Global Optimization Algorithm, Figure 3

solutions quickly, but it could overlook the rectangle that contains the global optimum (this would happen if the rectangle containing the global optimum had a poor objective function value at the center).

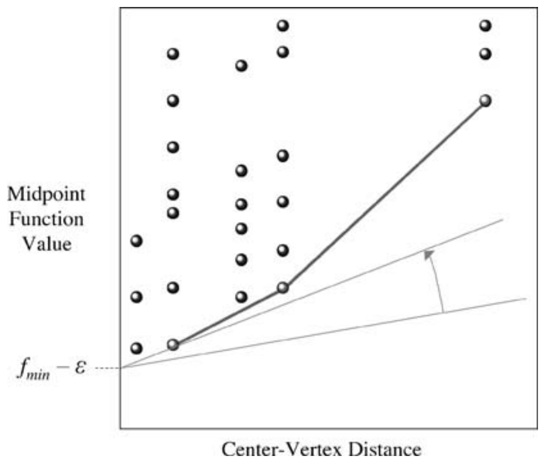
To select just one 'best' rectangle, we would have to introduce a tuning parameter that controlled the local/global balance. Unfortunately, the algorithm would then be extremely sensitive to this parameter, since the proper setting would depend on the (unknown) difficulty of the problem at hand.

DIRECT avoids tuning parameters by rejecting the idea of selecting just one rectangle. Instead, several rectangles are selected using all possible relative weightings of local versus global search. The idea of using all possible weightings may seem impractical, but with the help of a simple diagram this idea can actually be made quite intuitive. For this diagram, we will need a way to measure of the size of a rectangle. We will measure size using the distance between the center point and the vertices, as shown in Fig. 3.

With this measure of rectangle size, we can now turn our attention to Fig. 4 which shows how rectangles are selected. In the figure, each rectangle in the partition is represented by a dot. The horizontal coordinate of a dot is the size of the rectangle, measured by the center-vertex distance. The vertical coordinate is the function value at the midpoint of the rectangle. The dot labeled *A* represents the rectangle with the lowest function value, and so this would be the rectangle selected by a pure local strategy. Similarly, the dot labeled *B* represents one of the biggest rectangles, and so it would be selected by a pure global strategy. DIRECT selects not only these two extremes but also all the rectangles on the lower-right convex hull of the cloud of dots (the dots connected by the line). These rectangles represent 'efficient trade-offs' between local versus global search, in the sense that each of them is best for some relative weighting of midpoint function value and center-vertex distance. (We will explain the other lines in Fig. 4. shortly.)



Direct Global Optimization Algorithm, Figure 4



Direct Global Optimization Algorithm, Figure 5

One might think that the idea illustrated in Fig. 4 would extend naturally to the constrained case; that is, we would simply select any rectangle that was best for some weighting of objective value, center-vertex distance, and constraint values. Unfortunately, this does not work because it leads to excessive sampling in the infeasible region. However, as we explain next, there is an alternative way of thinking about the lower-right convex hull that does extend to the constrained case.

For the sake of the exposition, let us suppose for the moment that we know the optimal function value f^*. For the function to reach f^* within rectangle r, it would have to undergo a rate of change of at least $(f_r - f^*)/d_r$, where f_r is the function value at the midpoint of rectangle r and d_r is the center-vertex distance. This follows because the function value at the center is f_r and the maximum distance over which the function can fall to f^* is the center-vertex distance d_r. Intuitively, it seems 'more reasonable' to assume that the function will undergo a gradual change than to assume it will make a steep descent to f^*. Therefore, if only we knew the value f^*, a reasonable criterion for selecting a rectangle would be to choose the one that minimizes $(f_r - f^*)/d_r$.

Figure 4 shows a graphical way to find the rectangle that minimizes $(f_r - f^*)/d_r$. Along the vertical axis we show the current best function value, $f_{\min}$, as well as the supposed global minimum f^*. Now suppose we anchor a line at the point $(0, f^*)$ and slowly swing it upwards. When we first encounter a dot, the slope of the line will be precisely the ratio $(f_r - f^*)/d_r$, where r is the index of the rectangle corresponding to the encountered dot. Moreover, since this is the first dot touched by the line, rectangle r must be the rectangle that minimizes $(f_r - f^*)/d_r$.

Of course, in general we will not know the value of f^*. But we do know that, whatever f^* is, it satisfies $f^* \leq f_{\min}$. So imagine that we repeat the line-sweep exercise in Fig. 4 for all values of f^* ranging from $f_{\min}$ to $-\infty$. How many rectangles could be selected? Well, with a little thought, it should be clear that the set of dots that can be selected via these line sweeps is precisely the lower-right convex hull of the dots.

This alternative approach to deriving the lower-right convex hull suggests a small but important modification to the selection rule. In particular, to prevent DIRECT from wasting function evaluations in pursuit of very small improvements, we will insist that the value of f^* satisfy $f^* \leq f_{\min} - \epsilon$. That is, we are only interested in selecting rectangles where it is reasonable that we can find a 'significantly better' solution. A natural value of ϵ would be the desired accuracy of the solution. In our implementation, we have set $\epsilon = \max(10^{-4}|f_{\min}|, 10^{-8})$.

As shown in Fig. 5, the implication of this modification is that some of the smaller rectangles on the lower-right convex hull may be skipped. In fact, the smallest rectangle that will be selected is the one chosen when $f^* = f_{\min} - \epsilon$.

The version of DIRECT described so far corresponds closely to the originally published version [2].

The only difference is that, in the original version, a selected rectangle was trisected not just on a single long side, but rather on all long sides. This approach eliminated the need to arbitrarily select a single long side when there were more than one and, as a result, it added an element of robustness to the algorithm. Experience has since shown, however, that the robustness benefit is small and that trisecting on a single long side (as here) accelerates convergence in higher dimensions.

Let us now consider how the rectangle selection procedure can be extended to handle inequality constraints. The key to handling constraints in DIRECT is to work with an *auxiliary function* that combines information on the objective and constraint functions in a special manner. To express this auxiliary function, we will need some additional notation. Let g_{rj} denote the value of constraint j at the midpoint of rectangle r. In addition, let $c_1, \dots, c_m$ be positive weighting coefficients for the inequality constraints (we will discuss how these coefficients are computed later). Finally, for the sake of the exposition, let us again suppose that we know the optimal function value f^*. The auxiliary function, evaluated at the center of rectangle r, is then as follows:

$$\max(f_r - f^*, 0) + \sum_{j=1}^{m} c_j \max(g_{rj}, 0)$$

The first term of the auxiliary function exacts a penalty for any deviation of the function value f_r above the global minimum value f^*. Note that, in a constrained problem, it is possible for f_r to be less than f^* by violating the constraints; due to the maximum operator, the auxiliary function gives no credit for values of f_r below f^*. The second term in the auxiliary function is a sum of weighted constraint violations. Clearly, the lowest possible value of the auxiliary function is zero and occurs only at the global minimum. At any other point, the auxiliary function is positive either due to suboptimality or infeasibility.

This auxiliary function is not a *penalty function* in the standard sense. A standard penalty function would be a weighted sum of the objective function and constraint violations; it would not include the value f^* since this value is generally unknown. Moreover, in the standard approach, it is critical that the penalty coefficients be sufficiently large to prevent the penalty function from being minimized in the infeasible region. This is not true for our auxiliary function: as long as f^* is the optimal function value, the auxiliary function is minimized at the global optimum for *any* positive constraint coefficients.

For the global minimum to occur in rectangle r, the auxiliary function must fall to zero starting from its (positive) value at the center point. Moreover, the maximum distance over which this change can occur is the center-vertex distance d_r. Thus, to reach the global minimum in rectangle r, the auxiliary function must undergo a minimum rate of change, denoted $h_r(f^*)$, given by

$$h_r(f^*) = \frac{\max(f_r - f^*, 0) + \sum_{j=1}^{m} c_j \max(g_{rj}, 0)}{d_r} .$$

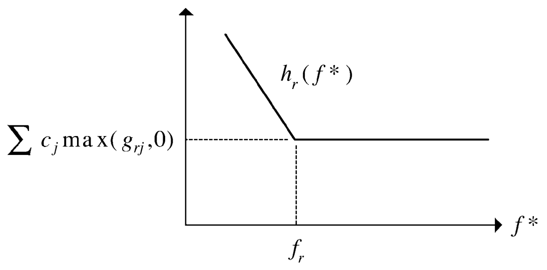
Since it is more reasonable to expect gradual changes than abrupt ones, a reasonable way to select a rectangle would be to select rectangle that minimizes the rate of change $h_r(f^*)$. Of course, this is impractical because we generally will not know the value f^*. Nevertheless, it is possible to select the set of rectangles that minimize $h_r(f^*)$ for *some* $f^* \leq f_{\min} - \epsilon$. This is how we select rectangles with constraints—assuming a feasible point has been found so that $f_{\min}$ is well-defined (we will show how this is implemented shortly). If no feasible point has been found, we simply select the rectangle that minimizes

$$\frac{\sum_{j=1}^{m} c_j \max(g_{rj}, 0)}{d_r} .$$

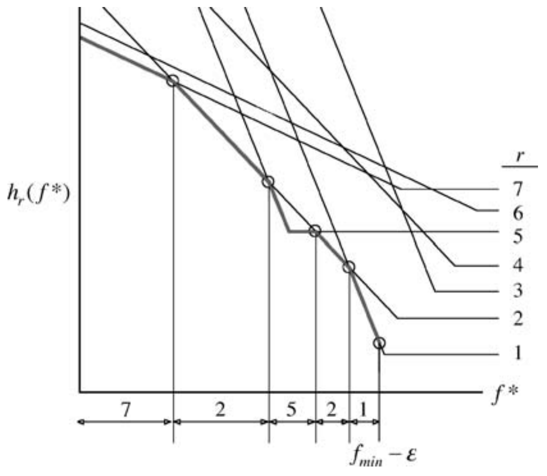
That is, we select the rectangle where the weighted constraint violations can be brought to zero with the least rate of change.

To implement this selection rule, it is again helpful to draw a diagram. This new selection diagram is based on plotting the rate-of-change function $h_r(f^*)$ as a function of f^*. Figure 6 illustrates this function. For values of $f^* \geq f_r$, the first term in the numerator of $h_r(f^*)$ is zero, and so $h_r(f^*)$ is constant. As f^* falls below f_r, however, the $h_r(f^*)$ increases, because we now exact a penalty for f_r being above the supposed global minimum f^*. The slope of $h_r(f^*)$ function to the left of f_r is $-1/d_r$.

Figure 7 superimposes, in one diagram, the rate-of-change functions for a hypothetical set of seven rectangles. For a particular value of f^*, we can visually find the rectangle that minimizes $h_r(f^*)$ by starting at the point



Direct Global Optimization Algorithm, Figure 6



Direct Global Optimization Algorithm, Figure 7

$(f^*, 0)$ along the horizontal axis and moving vertically until we first encounter a curve. What we want, however, is the set of *all* rectangles that can be selected in this way using *any* $f^* \leq f_{\min} - \epsilon$. This set can be found as follows (see Fig. 7). We start with $f^* = f_{\min} - \epsilon$ and move upwards until we first encounter a curve for some rectangle. We note this rectangle and follow its curve to the left until it intersects the curve for another rectangle (these intersections are circled in Figure 7). When this happens, we note this other rectangle and follow its curve to the left. We continue in this way until we find a curve that is never intersected by another one. This procedure will identify all the $h_r(f^*)$ functions that participate in the lower envelope of the curves to the left of $f_{\min} - \epsilon$. The set of rectangles found in this way is the set selected by DIRECT.

Along the horizontal axis in Fig. 7, we identify ranges of f^* values for which different rectangles have the lowest value of $h_r(f^*)$. As we scan from $f_{\min} - \epsilon$ to the left, the rectangles that participate in the lower envelope are 1, 2, 5, 2, and 7. This example illustrates that it is possible to encounter a curve more than once (here rectangle 2), and care must be taken not to double count such rectangles. It is also possible for some curves to coincide along the lower envelope, and so be 'tied' for the least rate of change (this does not happen in Fig. 7). In such cases, we select all the tied rectangles.

Tracing the lower envelope in Fig. 7 is not computationally intense. To see this, note that each selected rectangle corresponds to a curve on the lower envelope, and for each such curve the work we must do is to find the intersection with the next curve along the lower envelope. Finding this next intersection requires computing the intersection of the current curve with all the other curves. It follows that the work required for each selected rectangle (and hence for every two sampled points) is only on the order of the total number of rectangles in the partition.

The tracing of the lower envelope can also be accelerated by some pre-processing. In particular, it is possible to quickly identify rectangles whose curves lie completely above other curves. For example, in Fig. 7, curve 3 lies above curve 1, and curve 4 lies above curve 2. These curves cannot possibly participate in the lower envelope, and so they can be deleted from consideration before the lower envelope is traced.

It remains to explain how the constraint coefficients $c_1, \ldots, c_m$ are computed, as well as a few other details about trisection and the handling of integer variables. We will cover these details in turn, and then bring everything together into a step-by-step description of the algorithm.

To understand how we compute the constraint coefficient c_j, suppose for the moment that we knew the average rate of change of the objective function, denoted a_0, and the average rate of change of constraint j, denoted a_j. Furthermore, suppose that at the center of a rectangle we have $g_j > 0$. At the average rate of change of constraint j, we would have to move a distance equal to g_j/a_j to get rid of the constraint violation. If during this motion the objective function got worse at its average rate of change, it would get worse by a_0 times the distance, or $a_0(g_j/a_j) = (a_0/a_j)\, g_j$. Thus we see that the ratio a_0/a_j provides a way of converting units of constraint violation into potential increases in the objective function. For this reason, we will set $c_j = a_0/a_j$.

The average rates of change are estimated in a very straightforward manner. We maintain a variable s_0 for the sum of observed rates of change of the objective function. Similarly we maintain variables $s_1, \ldots, s_m$ for the sum of observed rates of change for each of the m constraints. All of these variables are initialized to zero at the start of the algorithm and updated each time a rectangle is trisected. Let x^{mid} denote the midpoint of the parent rectangle and let x^{left} and x^{right} denote the midpoints of the left and right child rectangles after trisection. The variables are updated as follows:

$$s_0 = s_0 + \sum_{\text{child}=\text{left}}^{\text{right}} \frac{\left|f(x^{\text{child}}) - f(x^{\text{mid}})\right|}{\left\|x^{\text{child}} - x^{\text{mid}}\right\|}$$

$$s_j = s_j + \sum_{\text{child}=\text{left}}^{\text{right}} \frac{\left|g_j(x^{\text{child}}) - g_j(x^{\text{mid}})\right|}{\left\|x^{\text{child}} - x^{\text{mid}}\right\|}.$$

Now the average rates of change are $a_0 = s_0/N$ and $a_j = s_j/N$, where N is the number of rates of change accumulated into the sums. It follows that

$$\frac{a_0}{a_j} = \frac{\frac{s_0}{N}}{\frac{s_j}{N}} = \frac{s_0}{s_j}.$$

We may therefore compute c_j using

$$c_j = \frac{s_0}{\max(s_j, 10^{-30})},$$

where we use the maximum operator in the denominator to prevent division by zero.

So far we have said that we will always trisect a rectangle along one of its long sides. However, as shown in Fig. 2, several sides may be tied for longest, and so we need some way to break these ties. Our tie breaking mechanism is as follows. We maintain counters t_i ($i = 1, \ldots, n$) for how many times we have split along dimension i over the course of the entire search. These counters are initialized to zero at the beginning of the algorithm, and counter t_i is incremented each time a rectangle is trisected along dimension i. If we select a rectangle that has several sides tied for being longest, we break the tie in favor of the side with the lowest t_i value. If several long sides are also tied for the lowest t_i value, we break the tie arbitrarily in favor of the lowest-indexed dimension. This tie breaking strategy has the effect of equalizing the number of times we split on the different dimensions.

Let us now turn to the calculation of the center-vertex distance. Recall that we measure distance using a weighted metric that assigns a length of one to the initial range of each variable ($u_i - \ell_i$). Each time a rectangle is split, the length of that side is then reduced by a factor of 1/3. Now consider a rectangle that has been trisected T times. Let $j = \text{mod}(T, n)$, so that we may write $T = kn + j$ where $k = (T - j)/n$. After the first kn trisections, all of the n sides will have been trisected k times and will therefore have length 3^{-k}. The remaining j trisections will make j of the sides have length $3^{-(k+1)}$, leaving $n - j$ sides with length 3^{-k}. Simple algebra then shows that the distance d from the center to the vertices is given by

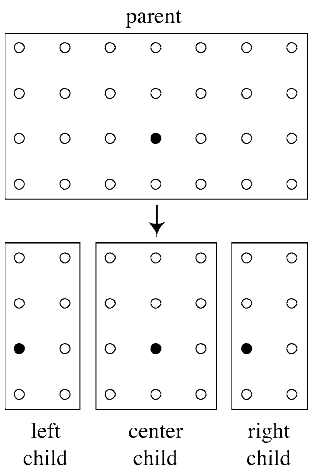
$$d = \frac{3^{-k}}{2}\left(\frac{j}{9} + n - j\right)^{0.5}.$$

(This is not obvious, but can be easily verified.)

The handling of integer variables is amazingly simple, involving only minor changes to the trisection routine and to the way the midpoint of a rectangle is defined. For example, consider an integer variable with range [1, 8]. We could not define the midpoint to be 4.5 because this is not an integer. Instead, we will use the following procedure. Suppose the range of a rectangle along an integer dimension is $[a, b]$, with both a and b being integers. We will define the 'midpoint' as $\lfloor (a + b)/2 \rfloor$, that is, it is the floor of algebraic average (the *floor* of z, denoted $\lfloor z \rfloor$, is the greatest integer less than or equal to z).

To trisect along the integer dimension, we first compute $\Delta = \lfloor (b - a + 1)/3 \rfloor$. If $\Delta \geq 1$, then after trisection the left child will have the range $[a, a + \Delta - 1]$, the center child will have the range $[a + \Delta, b - \Delta]$, and the right child will have range $[b - \Delta + 1, b]$. If $\Delta = 0$, then the integer side must have a range of two (i. e., $b = a + 1$). In this case, the center child will have the range $[a, a]$ the right child will have the range $[b, b]$, and there will be no left child. This procedure maintains the property that the midpoint of the parent rectangle always becomes the midpoint of the center child. As an example, Fig. 8 shows how a rectangle would be trisected when there are two integer dimensions. In the figure, the circles represent possible integer combinations, and the filled circles represent the midpoints.

Integer variables introduce three other complications. The first, which may be seen in Fig. 8, is that the



Direct Global Optimization Algorithm, Figure 8

sampled point may not be in the true geometric center of the rectangle. As a result, the center-vertex distance will not be unique but will vary from vertex to vertex. We ignore this detail and simply use the formula given above for the continuous case, which only depends upon the number of times a rectangle has been trisected.

The second complication concerns how we define a 'long' side. In the continuous case, the length of a side is directly related to the number of times it has been trisected along that dimension. Specifically, if a rectangle has been split k times along some side, then the side length will be 3^{-k} (recall that we measure distance relative to the original range of each variable). In the continuous case, therefore, the set of long sides is the same as the set of sides that have been split upon the least. When there are integers, however, the side lengths will no longer be multiples of 1/3. To keep things simple, however, we ignore this and continue to define a 'long' side as one that has been split upon the least. However, if an integer side has been split so many times that its side length is zero (i. e., the range contains a single integer), then this side will not be considered long.

The third and final complication is that, if *all* the variables are integer, then it is possible for a rectangle to be reduced to a single point. If this happens, the rectangle would be fathomed; hence, it should be ignored in the rectangle selection process in all subsequent iterations.

DIRECT stops when it reaches a user-defined limit on function evaluations. It would be preferable, of course, to stop when we have achieved some desired accuracy in the solution. However, for black-box problems where we only assume continuity, better stopping rules are hard to develop.

As for convergence, it is easy to show that, as f^* moves to $-\infty$, DIRECT will select one of the largest rectangles. Because we always select one of the largest rectangles, and because we always subdivide on a long side, every rectangle will eventually become very small and the sampled points will be dense in the space. Since we also assume the functions are continuous in the neighborhood of the optimum, this insures that we will get within any positive tolerance of the optimum after a sufficiently large number of iterations.

Although we have now described all the elements of DIRECT, our discussion has covered several pages, and so it will be helpful to bring everything together in a step-by-step description of the algorithm.

1) Initialization.
   Sample the center point of the entire space. If the center is feasible, set $x_{\min}$ equal to the center point and $f_{\min}$ equal to the objective function value at this point. Set $s_j = 0$ for $j = 0, \ldots, m$; $t_i = 0$ for $i = 1, \ldots, n$; and neval = 1 (function evaluation counter). Set maxeval equal to the limit on the number of function evaluations (stopping criterion).
2) Select rectangles.
   Compute the c_j values using the current values of s_0 and s_j, $j = 1, \ldots, m$. If a feasible point has *not* been found, select the rectangle that minimizes the rate of change required to bring the weighted constraint violations to zero. On the other hand, if a feasible point has been found, identify the set of rectangles that participate in the lower envelope of the $h_r(f^*)$ functions for some $f^* \leq f_{\min} - \epsilon$. A good value for ϵ is $\epsilon = \max(10^{-4} |f_{\min}|, 10^{-8})$. Let S be the set of selected rectangles.
3) Choose any rectangle $r \in S$.
4) Trisect and sample rectangle r.
   Choose a splitting dimension by identifying the set of long sides of rectangle r and then choosing the long side with the smallest t_i value. If more than one side is tied for the lowest t_i value, choose the one with the lowest-dimensional index. Let i be the resulting splitting dimension. Note that a 'long side'

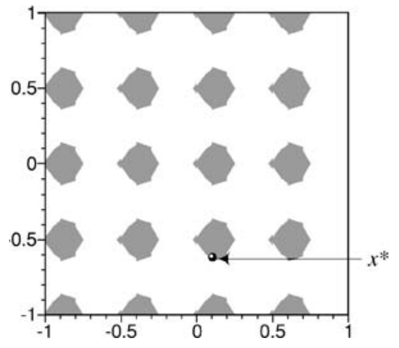
is defined as a side that has been split upon the least and, if integer, has a positive range. Trisect rectangle r along dimension i and increment t_i by one. Sample the midpoint of the left third, increment neval by one, and update $x_{\min}$ and $f_{\min}$. If neval = maxeval, go to Step 7. Otherwise, sample the midpoint of the right third, increment *neval* by one, and update $x_{\min}$ and $f_{\min}$ (note that there might not be a right child when trisecting on an integer variable). Update the s_j $j = 0, \dots, m$. If all n variables are integer, check whether a child rectangle has been reduced to a single point and, if so, delete it from further consideration. Go to Step 5.

5) Update S.
   Set $S = S - \{r\}$. If S is not empty, go to Step 3. Otherwise go to Step 6.
6) Iterate.
   Report the results of this iteration, and then go to Step 2.
7) Terminate.
   The search is complete. Report $x_{\min}$ and $f_{\min}$ and stop.

The results of DIRECT are slightly sensitive to the order in which the selected rectangles are trisected and sampled because this order affects the t_i values and, hence, the choice of splitting dimensions for other selected rectangles. In our current implementation, we select the rectangles in Step 3 in the same order that they are found as we scan the lower envelope in Fig. 7 from $f^* = f_{\min} - \epsilon$ towards $f^* = -\infty$.

On the first iteration, all the s_j will be zero in Step 2 and, hence, all the c_j will be zero when computed using $c_j = s_0/\max(s_j, 10^{-30})$. Thus, in the beginning the constants c_j will not be very meaningful. This is not important, however, because on the first iteration there is only one rectangle eligible for selection (the entire space), and so the selection process is trivial. As the iterations proceed, the s_j will be based on more observations, leading to more meaningful c_j constants and better rectangle selections.

When there are no inequality constraints, the above step-by-step procedure reduces to the basic version of DIRECT described earlier. To see this, note that, when there are no constraints, every point is feasible and so $f_r \geq f_{\min} \geq f^*$ for all rectangles r. This fact, combined with the lack of any constraint violations, means that the $h_r(f^*)$ function given earlier reduces to $(f_r - f^*)/d_r$,



Direct Global Optimization Algorithm, Figure 9

which is precisely the rate-of-change function we minimized in the unconstrained version. Thus, in the unconstrained case, tracing the lower envelope in Fig. 7 identifies the same rectangles as tracing the lower-right convex hull in Fig. 5.

We will illustrate DIRECT on the following two-dimensional test function:

$$\begin{cases} \min & f(x_1, x_2) \\ \text{s.t.} & g(x_1, x_2) \leq 0 \\ & -1 \leq x_1, x_2 \leq +1, \end{cases}$$

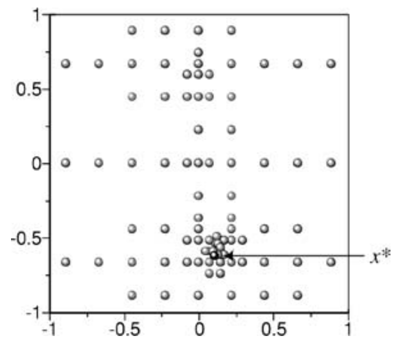
where

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2,$$

$$g(x_1, x_2) = -\sin(4\pi x_1) + 2\sin^2(2\pi x_2).$$

We call this problem the *Gomez #3 problem* since it was listed as the third test problem in an article by S. Gomez and A. Levy [1]. The global minimum of the Gomez #3 problem occurs at the point (0.109, − 0.623) where the function value is − 0.9711. The problem is difficult because the feasible region consists of many disconnected, approximately circular parts, giving rise to many local optima (see Fig. 9).

For this test function, DIRECT gets within 1% of the optimum after 89 function evaluations and within 0.01% after 513 function evaluations. The first 89 sampled points are shown in Fig. 10. For comparison, the tunneling algorithm of Gomez and Levy [1] converged using an average of 1053 objective function evaluations and 1873 constraint evaluations (averaged over 20 random starting points). One reason DIRECT converges



Direct Global Optimization Algorithm, Figure 10

quickly is that it searches both globally and locally during each iteration; as a result, as soon as the global part of the algorithm finds the basin of convergence of the optimum, the local part of the algorithm automatically exploits it.

In this example, DIRECT quickly gets close to the optimum but takes longer to achieve a high degree of accuracy. This suggests that the best performance would be obtained by combining DIRECT with a good local optimizer. The simplest way to do this is to run DIRECT for a predetermined number of function evaluations and then use the resulting solution as a starting point for a local optimization. While straightforward, this approach is highly sensitive to the number of function evaluations used in the global phase with DIRECT. If one uses too few function evaluations, DIRECT might not discover the basin of convergence of the global minimum.

To ensure that the global optimum is eventually found, we must somehow return to the global phase after we have performed a local search. One way of doing this is as follows. We start the local optimizer the very first time a feasible point is found (or perhaps after a minimum initial phase of 50–100 evaluations). After the local finishes, we return to DIRECT. However, DIRECT does not proceed the same as it would have without the local optimizer. Instead, the search will be more global, because the local optimizer will have reduced the value of $f_{\min}$ (which affects rectangle selection). DIRECT will now be looking for a point that improves upon the local solution—in effect, it will be looking for the basin of convergence of a better minimum. If DIRECT finds such an improving point, then we run a local search from this point and again return to DIRECT. This process continues until we reach a predetermined limit on the total number of function evaluations (for both DIRECT and the local optimizer). Used in this way, DIRECT becomes an intelligent routine for selecting starting points for the local optimizer.

While DIRECT works well on the Gomez #3 problem and on test functions reported in [2], the algorithm is not without its disadvantages. For example, DIRECT's use of a space-partitioning approach requires the user to have relatively tight lower and upper bounds on all the variables. DIRECT will perform miserably if one specifies wide bounds such as $[-10^{30}, +10^{30}]$. The space-partitioning approach also limits the algorithm to low-dimensional problems (say, less than 20). While integer variables are handled, they must be ordered, such as the number of gear teeth, since only then can we expect the function value at a rectangle's midpoint to be indicative of what the function is like in the rest of the rectangle. Another limitation is that equality constraints are not handled. Finally, the stopping criterion—a limit on function evaluations—is weak.

The advantages of DIRECT, however, are considerable. The algorithm can handle nonsmooth, nonlinear, multimodal, and even discontinuous functions (as long as the discontinuity is not close to the global optimum). The algorithm works well in the presence of noise, since a small amount of noise usually has little impact on the set of selected rectangles until late in the search. Computational overhead is low, and the algorithm can exploit parallel processing because it generates several new points per iteration. Based on the comparisons in [2], the algorithm also appears to be efficient in terms of the number of function evaluations required to get close to the global minimum. But the most important advantage of DIRECT stems from its unique approach to balancing local and global search—the simple idea of not sampling just one point per iteration, but rather sampling several points using all possible weightings of local versus global search. This approach leads to an algorithm with no tuning parameters, making the algorithm easy-to-use and robust.

See also

► αBB Algorithm
► Continuous Global Optimization: Applications

- ▶ Continuous Global Optimization: Models, Algorithms and Software
- ▶ Differential Equations and Global Optimization
- ▶ Global Optimization Based on Statistical Models
- ▶ Global Optimization in Binary Star Astronomy
- ▶ Global Optimization Methods for Systems of Nonlinear Equations
- ▶ Global Optimization Using Space Filling
- ▶ Topology of Global Optimization

References

1. Gomez S, Levy A (1982) The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions. In: Lecture Notes Math, vol 909. Springer, Berlin, 34–47
2. Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Th Appl 73(1):157–181

Direct Search Luus–Jaakola Optimization Procedure

LJ Optimization Procedure

REIN LUUS
Department Chemical Engineering,
University Toronto, Toronto, Canada

MSC2000: 93-XX

Article Outline

Keywords

Optimization; Direct search; Nonseparable problem optimization; Sensitivity; Global optimization

Direct search optimization procedures are attractive because of the ease with which they can be used. Optimization procedures where auxiliary functions such as gradients are not calculated are desirable for problems where discontinuous functions are encountered, or where numerous constraints make the calculation and use of gradients very difficult in searching for the global optimum. The reliability of getting to the vicinity of the global optimum is an additional feature that makes the use of direct search optimization an attractive means of optimization.

The need for an efficient and easy to use optimization procedure was illustrated in [1], in attempting to obtain the best weighting factors in a Liapunov function used for time suboptimal control of a linear gas absorber. Although at that time the best optimization procedure for that problem was the hill-climbing procedure due to H.H. Rosenbrock [35], the method encountered difficulties in establishing the global optimum. In the 1970s a large number of direct search optimization procedures were introduced. One such method is due to R. Luus and T.H.I. Jaakola [29], which has been called in the literature by numerous authors as the *LJ optimization procedure*. The method is based on using a number of randomly chosen test points over some region and contracting the region after every iteration, always starting the iteration with the best point found from the previous iteration as the center of the region. The ease of programming and the ease with which inequality constraints can be handled make this direct search procedure attractive.

Optimization Problem

We consider the problem of minimizing the *performance index* or *cost function*

$$I = f(x_1, \dots, x_n) \qquad (1)$$

subject to p inequality constraints

$$g_j(x_1, \dots, x_n) \geq 0, \quad j = 1, \dots, p, \qquad (2)$$

through the appropriate choice of $x_1, \dots, x_n$. The direct search optimization procedure suggested in [29] involves only three steps:

- Choose a number of points in the n-dimensional space through the equation

  $$\mathbf{x} = \mathbf{x}^* + \mathbf{D}\mathbf{r}, \qquad (3)$$

  where $\mathbf{D}$ is a diagonal matrix with diagonal elements chosen at random between -1 and $+1$, and $\mathbf{r}$ is the region size vector.

- Check the feasibility of each point with respect to (2), and for each feasible point evaluate the performance index I in (1).
- At the end of each iteration, $\mathbf{x}^*$ is replaced by the best feasible value of $\mathbf{x}$ obtained in Step 2, and the region size vector $\mathbf{r}$ is reduced in size by

  $$\mathbf{r}^{(j+1)} = \gamma \mathbf{r}^{(j)}, \tag{4}$$

  where γ is a contraction factor such as 0.95. This procedure is continued for a number of iterations and the results are examined.

If adequate convergence is not obtained, then the procedure can be repeated by carrying out another pass, using the information obtained from the previous pass. This optimization procedure enabled several difficult optimization problems to be solved in the original paper [29], and provided a means to solve a wide variety of problems in optimal control, such as time suboptimal control [9,8,7] and gave good approximation to optimal control by providing a means of obtaining the elements for the feedback gain matrix. The LJ optimization procedure was found very useful for stabilizing systems through shifting of poles [30] and testing stabilizability of linear systems [13]. Research was done to improve the likelihood of getting the global optimum for nonunimodal systems [37], but even without any modification, the reliability of the LJ procedure was found to be very good [38], even for the difficult bifunctional catalyst blend problem [26]. Therefore, the LJ optimization procedure could be used effectively for optimization of complex systems such as heat exchanger networks [17], a transformer design problem [36], design of structural columns in such a way that the amount of material would be minimized [3], and problems dealing with metallurgical processes [34]. The simplicity of the method was illustrated by the computer program given in its entirety in reference [17].

When the variables are restricted to be integers, special procedures may be necessary [12], since we cannot simply search on integer values to get the global optimum. Thus the scope of problems where LJ optimization procedure has been successfully applied is quite wide. In parameter estimation, N. Kalogerakis and Luus [6] found that by LJ optimization reliable estimates could be obtained for parameters in very few iterations, so that these estimates could then be used as starting values for quadratically convergent Gauss–Newton method, without having to worry about nonconvergence. In model reduction the LJ method has been found useful to match the reduced system's Nyquist plot to that of the original system [15], or used directly in time domain [40]. LJ optimization procedure has also been used successfully in model reduction in sampled-data systems [39] and is illustrated with several examples in [23].

Handling Equality Constraints

Suppose that in addition to the inequality constraints in (2), we also have m equality constraints

$$h_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m, \tag{5}$$

where these equality constraints are 'difficult' in the sense that they can not be used to solve for some particular variable.

Although a two-pass method to deal with equality constraints [10] was effective to solve optimization problems involving recycle streams [11], the general approach for handling *equality constraints* with LJ optimization procedure was not solved satisfactorily, until it was shown [4] that *penalty functions* can be used very effectively in direct search optimization. The work was extended in [33], and now it appears that the use of a quadratic penalty function incorporating a shifting term is the best way of dealing with difficult equality constraints [19]. We consider the *augmented performance index*

$$J = I + \theta \sum_{i=1}^{m} (h_i - s_i)^2, \tag{6}$$

where a shifting term s_i is introduced for each equality constraint. To solve the optimization problem, LJ optimization procedure is used in a multipass fashion, where at the beginning of each pass consisting of a number of iterations, the region sizes are restored to a fraction of the sizes used at the beginning of the previous pass. The shifting terms s_i are updated after every pass simply by adjusting the values at the beginning of pass $(q+1)$ based on the deviation of the left-hand side in (5) from zero; i. e.,

$$s_i^{(q+1)} = s_i^{(q)} - h_i, \quad i = 1, \dots, m, \tag{7}$$

where q is the pass number. Upon optimization the product $-2\,\theta\, s_i$ gives the Lagrange multiplier associated with the ith equality constraint, yielding useful sensitivity information. Full details are given in [19]. This approach to dealing with equality constraints was used in [31] in using *iterative dynamic programming* (IDP) to solve an optimal control problem where the final state was specified, and in [28] where the volume of the fed-batch reactor was specified at the final time.

Another approach to deal with equality constraints is to solve the algebraic equations at each iteration by grouping the equations [21]. For several optimization problems this approach yielded very rapid convergence to the global optimum [22].

Use of LJ Optimization Procedure for High-Dimensional Problems

In [2] it was found that LJ optimization can be used quite effectively to solve optimal control problems, where the system is divided into a number of time stages. This approach was used in [27] to solve a very difficult optimal control problem involving the determination of the optimum drug delivery schedule to minimize the tumor size at the end of 12 weeks. The problem was broken into 84 time stages, each consisting of a single day. In spite of the *state constraints* and *discontinuous functions*, this 84-dimensional optimization problem was solved successfully on a personal computer in reasonable computation time. Especially now that the personal computers are much faster, such a problem is considerably easier to solve. To solve high-dimensional problems a multipass method was used for LJ optimization where after a pass, the region would be restored to a value smaller than used at the beginning of the previous pass and the procedure was repeated. In the case of the *cancer chemotherapy* problem, the problem required a number of runs for successful solution [27].

Determination of Region Size

One of the problems that was outstanding for the LJ optimization procedure was how to choose the region size vector $\mathbf{r}$ effectively at the beginning of the iterations, especially when a multipass procedure was used. This problem was recently solved in [20], by suggesting that the initial region size be determined by the extent of the variation of the variable during the previous pass. With the use of reliable values for the region size at the beginning of each pass in a multipass run, the computational effort is decreased quite substantially. For example, when we consider the *nonseparable optimization problem* introduced in [32], where we have a system described by three difference equations:

$$x_1(k+1) = \frac{x_1(k)}{1 + 0.01u_1(k)(3 + u_2(k))},$$

$$x_2(k+1) = \frac{x_2(k) + u_1(k)x_1(k+1)}{1 + u_1(k)(1 + u_2(k))},$$

$$x_3(k+1) = \frac{x_3(k)}{1 + 0.01u_2(k)(1 + u_3(k))},$$

with the initial condition

$$\mathbf{x}(0) = [2 \quad 5 \quad 7]^{\top}.$$

The control variables are constrained by

$$0 \le u_1(k) \le 4,$$
$$0 \le u_2(k) \le 4,$$
$$0 \le u_3(k) \le 0.5.$$

The performance index to be minimized is

$$\begin{aligned} I &= x_1^2(P) + x_2^2(P) + x_3^2(P) \\ &+ \left[\left(\sum_{k=1}^{P} x_1^2(k-1) + x_2^2(k-1) + 2u_3^2(k-1) \right) \right. \\ &\cdot \left. \left(\sum_{k=1}^{P} x_3^2(k-1) + 2u_1^2(k-1) + 2u_2^2(k-1) \right) \right]^{\frac{1}{2}} \end{aligned}$$

where P is the number of stages. When P is taken as 100, then we have a 300 variable optimization problem, because at each stage there are three control variables to be determined. Without the use of a reliable way of determining the region sizes over which to take the control variables, the problem is very difficult, but with the method suggested in [20] the problem was solved quite readily by the LJ optimization procedure by using 100 random points per iteration and 60 passes, each consisting of 201 iterations, to yield $I = 258.3393$. Although the computational requirements appear enormous, the actual computation time was less than 20 minutes on a Pentium-120 personal computer [20], which corresponds to less than one minute on the Pentium4/2.4 GHz personal computer. This value of the performance index is very close to the value $I = 258.3392$ obtained by use of iterative dynamic programming [18]. To solve this problem, IDP is much more efficient in spite of the nonseparability of the problem, because in IDP the problem is solved as a 3 variable problem over 100 stages, rather than a 300 variable optimization problem. Therefore, the LJ procedure is useful in checking the optimal control policy obtained by some other method. Here, the control policies obtained by IDP and LJ optimization procedure are almost identical, where a sudden change at around stage 70 occurs in the control variables u_1 and u_2. Therefore, LJ optimization procedure is ideally suited for checking results obtained by other methods, especially when the optimal control policy differs from what is expected, as is the case with this particular example.

Recently it was shown that the convergence of the LJ optimization procedure in the vicinity of the optimum can be improved substantially by incorporating a simple line search to choose the best center point for a subsequent pass [24]. For a typical model reduction problem, to reach the global optimum the computation time was reduced by a factor of four when the line search was incorporated. Due to its simplicity, the LJ optimization procedure can be programmed very easily. Computational experience with numerous optimization problems has shown that the method has high reliability of obtaining the *global optimum*, so the LJ optimization procedure provides a very good means of obtaining the optimum for very complex problems.

See also

► Interval Analysis: Unconstrained and Constrained Optimization

References

1. Bennett HW, Luus R (1971) Application of numerical hill-climbing in control of systems via Liapunov's direct method. Canad J Chem Eng 49:685–690
2. Bojkov B, Hansel R, Luus R (1993) Application of direct search optimization to optimal control problems. Hungarian J Industr Chem 21:177–185
3. Dinkoff B, Levine M, Luus R (1979) Optimum linear tapering in the design of columns. Trans ASME 46:956–958
4. Hartig F, Keil FJ, Luus R (1995) Comparison of optimization methods for a fed-batch reactor. Hungarian J Industr Chem 23:141–148
5. Jaakola THI, Luus R (1974) A note on the application of nonlinear programming to chemical-process optimization. Oper Res 22:415–417
6. Kalogerakis N, Luus R (1982) Increasing the size of region of convergence for parameter estimation. Proc. 1982 Amer. Control Conf., 358–362
7. Luus R (1974) Optimal control by direct search on feedback gain matrix. Chem Eng Sci 29:1013–1017
8. Luus R (1974) A practical approach to time-optimal control of nonlinear systems. Industr Eng Chem Process Des Developm 13:405–408
9. Luus R (1974) Time-optimal control of linear systems. Canad J Chem Eng 52:98–102
10. Luus R (1974) Two-pass method for handling difficult equality constraints in optimization. AIChE J 20:608–610
11. Luus R (1975) Optimization of multistage recycle systems by direct search. Canad J Chem Eng 53:217–220
12. Luus R (1975) Optimization of system reliability by a new nonlinear integer programming procedure. IEEE Trans Reliabil 24:14–16
13. Luus R (1975) Solution of output feedback stabilization and related problems by stochastic optimization. IEEE Trans Autom Control 20:820–821
14. Luus R (1976) A discussion on optimization of an alkylation process. Internat J Numer Methods in Eng 10:1187–1190
15. Luus R (1980) Optimization in model reduction. Internat J Control 32:741–747
16. Luus R (1990) Optimal control by dynamic programming using systematic reduction in grid size. Internat J Control 19:995–1013
17. Luus R (1993) Optimization of heat exchanger networks. Industr Eng Chem Res 32:2633–2635
18. Luus R (1996) Application of iterative dynamic programming to optimal control of nonseparable problems. Hungarian J Industr Chem 25:293–297
19. Luus R (1996) Handling difficult equality constraints in direct search optimization. Hungarian J Industr Chem 24: 285–290
20. Luus R (1998) Determination of the region sizes for LJ optimization procedure. Hungarian J Industr Chem 26:281–286

21. Luus R (1999) Effective solution procedure for systems of nonlinear algebraic equations. Hungarian J Industr Chem 27:307–310
22. Luus R (2000) Handling difficult equality constraints in direct search optimization. Part 2. Hungarian J Industr Chem 28:211–215
23. Luus R (2000) Iterative dynamic programming. Chapman and Hall/CRC, London, pp 44–66
24. Luus R (2007) Use of line search in the Luus–Jaakola optmization procedure. Proc IASTED Internat Conf on Computational Intelligence. Banff, Alberta, Canada, July 2-4, 2007, pp 128–135
25. Luus R, Brenek P (1989) Incorporation of gradient into random search optimization. Chem Eng Techn 12:309–318
26. Luus R, Dittrich J, Keil FJ (1992) Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. Canad J Chem Eng 70:780–785
27. Luus R, Hartig F, Keil FJ (1995) Optimal drug scheduling of cancer chemotherapy by direct search optimization. Hungarian J Industr Chem 23:55–58
28. Luus R, Hennessy D (1999) Optimization of fed-batch reactors by the Luus–Jaakola optimization procedure. Industr Eng Chem Res 38
29. Luus R, Jaakola THI (1973) Optimization by direct search and systematic reduction in the size of search region. AIChE J 19:760–766
30. Luus R, Mutharasan R (1974) Stabilization of linear system behaviour by pole shifting. Internat J Control 20:395–405
31. Luus R, Storey C (1997) Optimal control of final state constrained systems. Proc. IASTED Intl. Conf. Modelling, Simulation and Optimization, pp 245–249
32. Luus R, Tassone V (1992) Optimal control of nonseparable problems by iterative dynamic programming. Proc. 42nd Canad. Chemical Engin. Conf., Toronto, Canada, pp 81–82
33. Luus R, Wyrwicz R (1996) Use of penalty functions in direct search optimization. Hungarian J Industr Chem 24:273–278
34. Papangelakis VG, Luus R (1993) Reactor optimization in the pressure oxidation process. Proc. Internat. Symp. Modelling, Simulation and Control of Metall. Processes, 159–171
35. Rosenbrock HH (1960) An automatic way of finding the greatest or least value of a function. Computer J 3:175–184
36. Spaans R, Luus R (1992) Importance of search-domain reduction in random optimization. JOTA 75:635–638
37. Wang BC, Luus R (1977) Optimization of non-unimodal systems. Internat J Numer Methods in Eng 11:1235–1250
38. Wang BC, Luus R (1978) Reliability of optimization procedures for obtaining global optimum. AIChE J 19:619–626
39. Yang SM, Luus R (1983) A note on model reduction of digital control systems. Internat J Control 37:437–439
40. Yang SM, Luus R (1983) Optimization in linear system reduction. Electronics Lett 19:635–637

Discontinuous Optimization

MARCEL MONGEAU
Laboratoire MIP, University Paul Sabatier,
Toulouse, France

MSC2000: 90Cxx

Article Outline

Keywords

Discontinuous optimization; Nondifferentiable optimization; Piecewise linear programming; Active set methods; Exact penalty method

Continuous optimization refers to optimization involving objective functions whose domain of definition is a continuum, as opposed to a set of discrete points in combinatorial (or discrete) optimization. Discontinuous optimization is the special case of continuous optimization in which the objective function, although defined over a continuum (let us suppose over $\mathbf{R}^n$), is not necessarily a continuous function.

We define the discontinuous optimization problem as:

$$\begin{cases} \inf & \widetilde{f}(x) \\ \text{s.t.} & f_i(x) = 0, \quad i \in E, \\ & f_i(x) \geq 0, \quad i \in I, \end{cases} \tag{1}$$

where the index sets E and I are finite and disjoint and $\widetilde{f}$ and f_i, $i \in E \cup I$ are a collection of (possibly discontinuous) piecewise differentiable functions that map $\mathbf{R}^n$ to $\mathbf{R}$. A *piecewise differentiable function* $f\colon \mathbf{R}^n \to \mathbf{R}$ is a function whose derivative is defined everywhere except over a subset of a finite number of sets, called *ridges*, of the form $\{x \in \mathbf{R}^n\colon r(x) = 0\}$, where r is a differentiable function, and these ridges partition the domain into subdomains over each of which f is differentiable. By abuse of language, we shall call $r(x)$ a ridge of f.

Without loss of generality, we can restrict our attention to the *unconstrained optimization problem*: $\inf_x f(x)$, where f is a (possibly discontinuous) piecewise differentiable function. Indeed, in order to solve problem (1), one can consider the unconstrained l_1 *exact penalty function*

$$f_\gamma(x) := \gamma \widetilde{f}(x) + \sum_{i\in E} |f_i(x)| - \sum_{i\in I} \min[0, f_i(x)]$$

for a succession of decreasing positive values of the *penalty parameter* γ (f_γ is clearly a piecewise differentiable function). Notice however that using the l_1 penalty function (and dealing with the decrease of a penalty parameter) is only one approach to handling the constrained problem and may not be the best way.

Given a (possibly discontinuous) piecewise differentiable function f defined over $\mathbf{R}^n$ and the finite set $\{r_i(x)\}_{i\in\mathcal{R}}$ of its ridges, we define a *cell* of f to be a nonempty set $C \subseteq \mathbf{R}^n$ such that for all $x, y \in C$ we have $\operatorname{sign}(r_i(x)) = \operatorname{sign}(r_i(y)) \neq 0$, for all $i \in \mathcal{R}$, where the function sign is either 1, -1 or 0, according to whether its argument is positive, negative or zero. Thus, f is differentiable over a cell.

Considering the optimization of functions which are nonsmooth and even discontinuous is motivated by applications in VLSI and floor-planning problems, plant layout, batch production, switching regression, discharge allocation for hydro-electric generating stations, fixed-charge problems, for example (see [4, Introd.] for references). Note that most of these problems can alternatively be modeled within the context of mixed integer programming, a field straddling combinatorial optimization and continuous optimization.

The inescapable nonconvexity nature of discontinuous functions gives rise to the existence of several local optima in discontinuous optimization problems. We do not address here the difficult issue of global optimization. We are concerned with finding a *local* infimum of the above optimization problem. An algorithm looking for local optima can however be used as an adjunct to some heuristic or global optimization method for discontinuous optimization problems but the inherent combinatorial nature of such an approach is often ultimately dominant. More importantly, it provides a framework allowing the optimizer to deal directly with the nonsmoothnesses and discontinuities involved, and thereby, improve solutions found by heuristic methods, when this is possible.

Leaving aside the heuristic methods (which many people facing practical discontinuous optimization problems rely upon in order to solve mixed integer programming formulation of discontinuous optimization problems), previous work on discontinuous optimization includes smoothing algorithms. The *smoothing algorithms* express discontinuities by means of a step function, and then they approximate the step function by a function which is not only continuous but moreover smooth, so that the resulting problem can be solved by a gradient technique (cf. also ► Conjugate–gradient methods). Both I.I. Imo and D.J. Leech [7] and I. Zang [9] developed methods in which the objective function is replaced only in the neighborhood of the discontinuities. Two drawbacks of these methods are the potential numerical instability when we want this neighborhood to be small, and the cost of evaluating the smoothed functions. In many instances the discontinuities of the first derivative are exactly the regions of interest and smoothing has the effect of making such regions less discernible.

Another approach, which deals explicitly with the discontinuities within the framework of continuous optimization, is the following *active set method* (introduced in [4]). Recall the following definitions relevant to active set methods: the *null space* of M, denoted by $\mathcal{N}(M)$, is defined by

$$\mathcal{N}(M) \equiv \left\{x \in \mathbb{R}^n : \ Mx = \vec{0}\right\}.$$

We say that a ridge r is *active* at $\widehat{x}$ if $r(\widehat{x}) = 0$. Let $\mathcal{A}(\widehat{x}) \subseteq \mathcal{R}$ be the (finite) index set of the ridges that are active at the current point $\widehat{x}$, and let $A(\widehat{x})$ be the *matrix of activities*, having as columns the gradients of the ridges that are active at $\widehat{x}$. In the case of linear ridges, $r_i(x) := a_i^\top x - b_i$, a direction $d \in \mathcal{N}(A^\top(\widehat{x}))$ is said to *preserve* each activity $i \in \mathcal{A}(\widehat{x})$ since for each $i \in \mathcal{A}(\widehat{x})$ we have $r_i(\widehat{x} + \alpha d) = r_i(\widehat{x}) = 0$.

If $\mathcal{A}(\widehat{x}) \neq \emptyset$, then $\nabla f(\widehat{x})$ is not necessarily defined. This is because we cannot talk about the gradient of the function at $\widehat{x}$ since there is no vector $g \in \mathbf{R}^n$ such that $g^\intercal d$ is the first order change of f in direction d, for any $d \in \mathbf{R}^n$. Thus, we cannot use, as in the smooth situation, the negative gradient direction as a descent direction. We term any $(n \times 1)$-vector $g_{\hat{x}}$ such that

$$f'(\widehat{x}; d) = g_{\hat{x}}^\top d, \quad \text{for all } d \in \mathcal{N}(A^\top),$$

a *restricted gradient* of f at $\widehat{x}$, because it is the gradient of the restriction of f to the space $\mathcal{N}(A^\top(\widehat{x}))$.

Let us first consider the continuous piecewise linear case. We assume that the ridges of f are given, and also we assume that the restriction of f to any cell is known. Hence, we are assuming that more information on the structure of the objective function is available than, for example, in a bundle method [8], which assumes that only one element of the subdifferential is known at any point.

It is shown in [4] that, under some nondegeneracy assumptions (e. g. the gradients of the ridges which are active at $\overline{x}$ are linearly independent), any continuous piecewise linear function f can be *decomposed* in a neighborhood of $\widehat{x}$ into a smooth function and a sum of continuous functions having a single ridge as follows:

$$f(x) = f(\widehat{x}) + g_{\hat{x}}^\top (x - \widehat{x}) + \sum_{i \in \mathcal{A}(\hat{x})} \nu_{\hat{x}}^i \min(0, a_i^\top (x - \widehat{x})) ,$$

for some scalars $\{\nu_{\hat{x}}^i\}_{i \in \mathcal{A}(\hat{x})}$, and some vector $g_{\hat{x}} \in \mathbb{R}^n$. We term $g_{\hat{x}}$ *the* restricted gradient of f at $\widehat{x}$. Note that if m ridges of f are active at $\widehat{x}$, it means that there are 2^m cells in any small neighborhood of $\widehat{x}$. The vector $g_{\hat{x}}$ and the m scalars $\{\nu_{\hat{x}}^i\}_{i \in \mathcal{A}(\widehat{x})}$, together with the m gradients of the activities, $\{a_i\}_{i \in \mathcal{A}(\widehat{x})}$, thus completely characterize the behavior of f over the 2^m cells in the neighborhood of $\widehat{x}$!

With such a decomposition at any point of $\mathbf{R}^n$, an algorithm for finding a local minimum of a continuous piecewise linear function f is readily obtained, as long as we assume no degeneracy at any iterate and at any breakpoint encountered in the line search (we shall discuss later the degenerate situation):

```
1      Choose any $x^1 \in \mathbb{R}^n$ and set $k \leftarrow 1$.
BEGIN  REPEAT
2      Identify the activities, $\mathcal{A}(x^k)$, and compute $d^k \equiv -P(g_{x^k})$, the projection of
       the restricted gradient onto the space orthogonal to the gradients of the activities.
       IF $d^k = \overrightarrow{0}$ ($x^k$ is a dead point; compute a single-dropping descent direction or
       establish optimality), THEN
3      Compute $\{u_i\}_{i \in \mathcal{A}(x^k)}$, the coefficients of $\{a_i\}_{i \in \mathcal{A}(x^k)}$ in the linear combination of
       $g_{x^k}$ in terms of the columns of $\mathcal{A}(x^k)$.
4      IF $u_i < 0$ or $u_i > -\nu_{x^k}^i$, for some $i \in \mathcal{A}(x^k)$ (violated optimality condition), THEN
5      (Drop activity $i$)
       Redefine $d^k = P_{-i}(a_i)$, if the violated inequality found corresponds to $u_i \geq 0$;
       otherwise $d^k = -P_{-i}(a_i)$ if it is $u_i \leq -\nu_{x^k}^i$, where $P_{-i}$ is the orthogonal
       projector onto the space orthogonal to the gradients of all the activities but activity $i$.
       ELSE stop: $x^k$ is a local minimum of $f$.
       ENDIF ENDIF
6      (Line search)
       Determine the step size $\alpha^k$ by solving $\min_{\alpha>0} f(x^k + \alpha d^k)$. This line search
       can be done from $x^k$, moving from one break-point of $f$ to the next, in the direction $d^k$,
       until either we establish unboundedness of the objective function or the value of $f$ starts increasing.
7      Update $x^{k+1} = x^k + \alpha^k d^k$, $k \leftarrow k + 1$.
END    REPEAT
```

Continuous piecewise linear minimization algorithm

Remark that in step 6, the directional derivative of the objective function in the direction d^k can easily be updated from one breakpoint to the other in terms of the scalar $\nu_{\overline{x}}^i$, where i is the index of the ridge crossed at breakpoint $\overline{x}$.

Let us now consider the case where f is still piecewise linear but with possibly discontinuities across some ridges. We term such ridges: *faults*, and $\mathcal{F}(\widehat{x})$ denotes the faults that are active at $\widehat{x}$.

Note first that a (local) minimum does not always exist in the discontinuous case. Consider for example the following univariate function, having $x = 0$ as a fault:

$$f(x) = \begin{cases} x+1 & \text{if } x \geq 0, \\ -x & \text{otherwise.} \end{cases}$$

Hence, we rather look for a *local infimum*. In order to find such a local infimum of a function f having some faults, we shall simply generalize the algorithm for the continuous problem by implicitly considering any discontinuity or *jump* across a fault i in f as the limiting case of a continuous situation. Since we are looking for a local infimum, without loss of generality we shall henceforth only consider functions f such that

$$f(x) = \liminf_{\overline{x} \to x} f(\overline{x}),$$

in other words, we consider the lower semicontinuous envelope of f.

The algorithm for the discontinuous case is essentially the same as in the continuous case except that we consider dropping an active fault from a dead point, x, only if we do so along a direction d such that

$$\lim_{\delta \to 0^+} f(x + \delta d) = f(x)$$

(i. e. as $\delta > 0$ is small, the value of f does not jump up from x to $x + \delta d$). Thus, virtually only step 4 must be adapted from the continuous problem algorithm in order to solve the discontinuous case. To make more carefully the intuitive concept of directions jumping up or down, we define the set of *soaring directions* from a point $\widehat{x}$ to be:

$$S(\widehat{x}) := \left\{ d \in \mathbb{R}^n : \begin{array}{c} \exists \epsilon > 0, \overline{\delta} > 0 : \\ \forall\, 0 < \delta < \overline{\delta} : \\ f(\widehat{x} + \delta d) - f(\widehat{x}) > \epsilon \end{array} \right\}.$$

If we define, for a nondegenerate point $\widehat{x}$,

$$S^+(\widehat{x}) := \left\{ i \in \mathcal{A}(\widehat{x}) : \begin{array}{c} \text{if } d^{i^+} \in \mathcal{N}(A_{-i}^\top) \\ \text{and } a_i^\top d^{i^+} > 0 \\ \text{then } d^{i^+} \in S(\widehat{x}) \end{array} \right\},$$

$$S^-(\widehat{x}) := \left\{ i \in \mathcal{A}(\widehat{x}) : \begin{array}{c} \text{if } d^{i^-} \in \mathcal{N}(A_{-i}^\top) \\ \text{and } a_i^\top d^{i^-} < 0 \\ \text{then } d^{i^-} \in S(\widehat{x}) \end{array} \right\},$$

then the set of soaring single-dropping directions from $\widehat{x}$ are simply the directions dropping an activity $i \in S^+(\widehat{x})$ positively and the directions dropping an $i \in S^-(\widehat{x})$ negatively (we say that activity i is *dropped positively* (*negatively*) if all current activities, except for the ith, are preserved and if, moreover, $a_i^\top d$ is positive (negative)). A fault can now be defined more rigorously: a *positive* (*negative*) fault of f at a point $\widehat{x}$ is a ridge $i \in \mathcal{R}$ such that for any neighborhood, $B(\widehat{x})$, of $\widehat{x}$, there exists a nondegenerate point $x' \in B(\widehat{x})$ with $i \in \mathcal{S}^+(x')$ (with $i \in \mathcal{S}^-(x')$). The set of all positive (negative) faults at $\widehat{x}$ is denoted by $\mathcal{F}^+(\widehat{x})$ $\mathcal{F}^-(\widehat{x})$). The *set of faults* of f at a point $\widehat{x}$ is denoted by

$$\mathcal{F}(\widehat{x}) := \mathcal{F}^+(\widehat{x}) \cup \mathcal{F}^-(\widehat{x}).$$

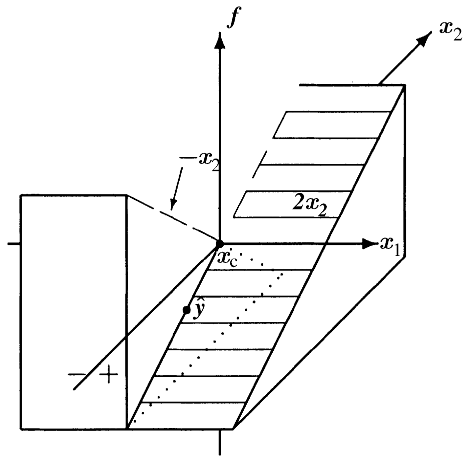
We modify the continuous problem algorithm in such a way that, at a nondegenerate dead point, x^k, we do not need to verify the optimality conditions corresponding to soaring single-dropping directions ($u_i \geq 0, i \in \mathcal{S}^+ (x^k)$ and $u_i \leq -\nu^i$, $i \in \mathcal{S}^- (x^k)$), so that we never consider such single-dropping directions in order to establish whether x^k is optimal. This is reasonable since we are looking for a local minimum. The line-search step (step 6) is modified similarly: when we encounter a breakpoint $\overline{x}$ on a fault along a direction $d \in S(\overline{x})$ (jump up), we stop; while if d is such that $-d \in S(\overline{x})$, (jump down), we carry on to the next breakpoint, and update properly the directional derivative along d.

Note that one has to be careful at a 'contact' point $x_c \in \mathbf{R}$ (defined below). At x_c, contrary to at other points of a fault, we can drop activity i *both* positively and negatively.

The function $f\colon \mathbf{R}^2 \to \mathbf{R}$, given by

$$f(x) = \begin{cases} 2x_2 & \text{if } x_1 > 0 \text{ or} \\ & (x_1 = 0 \text{ and } x_2 \leq 0), \\ -x_2 & \text{otherwise,} \end{cases} \tag{2}$$

illustrates well the situation. Figure 1 shows the graph of f in a neighborhood of $x_c := (0, 0)^\top$ (the dotted lines are simply lines that could be seen if the hatched surface were transparent). The point x_c is a contact point with respect to the fault $x_1 = 0$.



Formally, we define $x_c \in \mathbf{R}^n$ to be a *contact point* of f with respect to $i \in \mathcal{A}(x_c)$, when $i \in \mathcal{F}(x_c)$ such that either

1) $i \in \mathcal{F}^+(x_c) \cap \mathcal{F}^-(x_c)$, or
2) there exist $\sigma^+, \sigma^- \in 3^{|\mathcal{R}|}$ such that $\sigma_i^+ = 1, \sigma_i^- = -1$ and

$$\lim_{\substack{x \to x_c,\\ \sigma(x)=\sigma^+}} f(x) = \lim_{\substack{x \to x_c,\\ \sigma(x)=\sigma^-}} f(x)$$

(continuity when crossing ridge i, which is a fault, at x_c), where $\sigma(x)$ is the vector whose kth component is $\operatorname{sign}(r_k(x))$.

Note that the fault $x_1 = 0$ and the point $x_c = (0, 0)^\top$ satisfy both conditions 1) and 2) in the above definition of a contact point for the function f defined by (2). They however satisfy only condition 1) for the function f: $\mathbf{R}^2 \to \mathbf{R}$, defined by

$$f(x) = \begin{cases} 1 & \text{if } x_1 \geq 0 \text{ and } x_2 \geq 0, \\ 2 & \text{if } x_1 \leq 0 \text{ and } x_2 \leq 0 \\ & \text{ and } x \neq (0,0)^\top, \\ 3 & \text{if } x_1 > 0 \text{ and } x_2 < 0, \\ 4 & \text{otherwise,} \end{cases}$$

with faults $x_1 = 0$ and $x_2 = 0$. For the function f: $\mathbf{R}^2 \to \mathbf{R}$ given by

$$f(x) = \begin{cases} -x_2 & \text{if } x_1 > 0 \text{ and } x_2 < 0, \\ 0 & \text{otherwise,} \end{cases}$$

with fault $x_1 = 0$, we satisfy only condition 2) ($\mathcal{F}^-(x_c)$ is empty).

An algorithm similar to the one introduced in the continuous case, but which does not consider soaring single-dropping directions, will encounter no difficulty with the discontinuity in f at any noncontact point (e. g. for (2), at any point other than x_c). Let us assume, without loss of generality, that at the kth iterate, x^k, $\mathcal{F}(x^k) = \mathcal{F}^-(x^k)$. The only step of the continuous algorithm which need to be modified is (assuming moreover that all points encountered in the algorithm are noncontact points):

```
4 | IF $u_i < 0$ for some $i \in \mathcal{A}(x^k)$, or $u_i > -v^i_{x^k}$ for
  | some $i \in \mathcal{A}(x^k)\backslash\mathcal{F}(x^k)$ (violated optimality
  | condition), THEN
```

The paper [4] describes techniques (including perturbation) to cope with problems that occur in certain cases where the hypothesis of nondegeneracy is not satisfied at points encountered in the course of the algorithm. One cannot however extend this algorithm to deal with *dead-point iterates* (i. e. not encountered as breakpoint along the line search) without considering carefully the combinatorial nature of the problem of degeneracy. Nevertheless, no difficulties were encountered in the computational experiments reported in [4], although serious problems can still arise at certain singular points (contact points and dead-point iterates, at which the objective function is not decomposable). Indeed, in the discontinuous case, there is no straightforward extension of this approach to the cases where the algorithm encounters a contact point. In the continuous case, the behavior of f over two juxtaposed cells are linked. At contact points however, there is coincidence of the values of restrictions of f to subdomains not otherwise linked to each other.

Let us now discuss the extension to the nonlinear case. An advantage of the active set approach for the continuous piecewise linear optimization problem, over, for example, the simplex-format algorithm of R. Fourer [6], is that it generalizes it not only to the discontinuous situation but also to the nonseparable and certain (decomposable) nonconvex cases. Above all, the active set approach is readily extendable to the nonlinear case, by adapting conventional techniques for nonlinear programming, as was done above with the projected gradient method for the (possibly discontinuous) piecewise linear case. The definition of decomposition must first be generalized so that it expresses the first order behavior of a piecewise differentiable function in the neighborhood of a point. The piecewise linear algorithm described above used descent directions attempting to decrease the smooth part of the function while maintaining the value of its nonsmooth part (when preserving the current activities). A first order algorithm for the nonlinear case could obtain these two objectives *up to first order changes*, as in the approach of A.R. Conn and T. Pietrzykowski to nonlinear optimization, via an l_1 exact penalty function [5]. In order to develop a second order algorithm, assuming now that f is (possibly discontinuous) *piecewise twice-differentiable* (i. e. twice differentiable everywhere except over a finite number of ridges), one must first extend the definition of first order decomposition to that of *second order decomposition*. One could then consider extending the strategies used by T.F. Coleman and Conn [2] on the exact penalty function approach to nonlinear programming (although the l_1 exact penalty function involves only first order types of nondifferentiabilities – ridges). The main idea is to attempt to find a direction which minimizes the change in f (up to second order terms) subject to preserving the activities (up to second order terms). Specifically, second order conditions must be derived (which are the first order conditions plus a condition on the 'definiteness' of the *reduced Hessian* of the *twice-differentiable part* of f (in the second order decomposition of f)). An analog of the Newton step (or of a modification of the Newton method; cf. also ► Gauss-Newton method: Least squares, relation to Newton's method) using a nonorthogonal projection [3] is then taken (or a single-dropping direction is used). An algorithm following these lines would be expected to possess global convergence properties (regardless of starting point) and a fast (2-step superlinear) asymptotic convergence rate as in [1].

See also

► Nondifferentiable Optimization

References

1. Coleman TF, Conn AR (1982) Nonlinear programming via an exact penalty function: Asymptotic analysis. Math Program 24:123–136
2. Coleman TF, Conn AR (1982) Nonlinear programming via an exact penalty function: Global analysis. Math Program 24:137–161
3. Conn AR (1976) Projection matrices – A fundamental concept in optimization. In: Vogt WG, Mickle MH (eds) 7th Annual Conf. in Modelling and Simulation, Ed. April 26–27, pp 599–605
4. Conn AR, Mongeau M (1998) Discontinuous piecewise linear optimization. Math Program 80(3):315–380
5. Conn AR, Pietrzykowski T (Apr. 1977) A penalty function method converging directly to a constrained optimum. SIAM J Numer Anal 14(2):348–375
6. Fourer R (1985) A simplex algorithm for piecewise-linear programming I: Derivation and proof. Math Program, 33:204–233
7. Imo II, Leech DJ (1984) Discontinuous optimization in batch production using SUMT. Internat J Production Res 22(2):313–321
8. Lemaréchal C (1978) Bundle methods in nonsmooth optimization. In: Lemaréechal C, Mifflin R (eds) Proc. IIASA Workshop, Nonsmooth Optimization, March 28–April 8, 1977, vol 3, Pergamon, Oxford, pp 79–102
9. Zang I (1981) Discontinuous optimization by smoothing. Math Oper Res 6(1):140–152

Discretely Distributed Stochastic Programs: Descent Directions and Efficient Points

KURT MARTI
Federal Armed Forces, University Munich,
Neubiberg, Germany

MSC2000: 90C15, 90C29

Article Outline

Keywords

Stochastic program; Stochastic optimization; Mean value function; Uncertainty; Efficient point; Efficient solution; Admissible solution; Pareto optimal solution; Partially monotonous; Partial monotonicity; Robustness; Stochastic matrix; Markov kernel; Descent direction; Necessary optimality condition without using (sub)gradients parametric representations; Scenario analysis

Introduction

Many problems in stochastic optimization, as for instance optimal stochastic structural design problems, stochastic control problems, problems of scenario analysis, etc., can be described [3,5] by mean value minimization problems of the type

$$\begin{cases} \min & F(x) \\ \text{s.t.} & x \in D, \end{cases} \tag{1}$$

where the objective function $F = F_u$ is the *mean value function*, defined by

$$F(x) = \mathsf{E}u(A(\omega)x - b(\omega)), \quad x \in \mathbb{R}. \tag{2}$$

Here, $(A(\omega), b(\omega))$ is a random $m \times (n+1)$ matrix, E denotes the expectation operator, D is a convex subset of R^n and $u : \mathbf{R}^m \to \mathbf{R}$ designates a convex loss function measuring the loss arising from the deviation $z = A(\omega)x - b(\omega)$ between the output $A(\omega)\,x$ of the stochastic linear system $x \to A(\omega)\,x$ and the random target $b(\omega)$.

Solving (1), (2), the loss function u should be exactly known. However, in practice mostly there is some uncertainty about the appropriate selection of u, for instance due to difficulties in assigning appropriate penalty costs to the deviation $z = A(\omega)x - b(\omega)$ between the output $A(\omega)x$ and the target $b(\omega)$. We suppose that $u \in U$, where U is a given set of convex loss functions containing the true, but unknown loss function u_0. A possible way out in this situation of uncertainty about u is either to construct (feasible) descent directions h of F at a (iteration) point x being valid for a large class U of loss functions, or to provide the decision maker with a certain set $E = E_{D,U}$ $(\subset D)$ of *efficient points* or solutions, being substitutes for an optimal solution x^* of (1), (2); hence, this set $E_{D,U}$ or at least its closed hull $\overline{E_{D,U}}$ should contain an optimal solution $x^* = x_u^*$ of (1), (2) for each $u \in U$. An important class $U = C_m^J$ of loss functions u is the set of partially monotonous increasing convex loss functions on $\mathbf{R}^m$ defined asfollows:

Definition 1 Let J be a given subset of $\{1, \ldots, m\}$. For $J = \emptyset$ we put $C_m^{\emptyset} = C_m$, where C_m is the set of all convex functions u on $\mathbf{R}^m$. If $J \neq \emptyset$, then $\mathcal{C}_m^J$ denotes the set of all convex functions $u : \mathbf{R}^m \to \mathbf{R}$ having the following property:

$$z_I \leq w_I,\ z_{II} \leq w_{II} \implies u(z) \leq u(w). \tag{3}$$

Here, $z_I \in \mathbf{R}^{|J|}$, $z_{II} \in \mathbf{R}^{m-|J|}$ is the partition of any $z \in \mathbf{R}^m$ into the subvectors $z_I = (z_i)_{i \in J}$, $z_{II} = (z_i)_{i \notin J}$. Moreover, $z_I \leq w_I$ means that $z_j \leq w_j$ for all $j \in J$.

Remark 2 In many cases one has loss functions $u \in C_m^J$ with one of the following additional strict partial monotonicity property:

$$\begin{cases} z_I \leq w_I, \\ z_{II} = w_{II}, \\ z_i < w_i \text{ for some } i \in J \end{cases} \implies u(z) < u(w), \tag{4}$$

$$z_I < w_I,\ z_{II} = w_{II} \implies u(z) < u(w), \tag{5}$$

where $z_I < w_I$ means that $z_j < w_j$ for all $j \in J$.

For a given set U of convex loss functions u containing the true, but unknown loss function u_0, a first definition of efficient solutions can be given as follows:

Definition 3 A point $x \in D$ is called a *nondominated*, *admissible* or *Pareto optimal solution* of (1), (2) if there is no vector $\widetilde{x} \in D, \widetilde{x} \neq x$, such that

$$F_u(\tilde{x}) \leq F_u(x) \quad \text{for all } u \in U, \tag{6}$$

$$F_{\tilde{u}}(\tilde{x}) < F_u(x) \quad \text{for some } \tilde{u} \in U, \tag{7}$$

where $F_u(x) := \mathsf{E}u(A(\omega)x - b(\omega))$. Let $E_{D,u}^0$ denote the set of all nondominated solutions of (1), (2).

Discretely Distributed Stochastic Programs

In the following we consider the construction of descent directions and efficient solutions for (1), (2) in the case that $(A(\omega), b(\omega))$ has a discrete distribution

$$\mathsf{P}_{(A(\cdot),b(\cdot))} = \sum_{i=1}^{r} \alpha_i \epsilon_{(A^i,b^i)}, \tag{8}$$

where $r > 1$ is an integer, $\alpha_i > 0$, $i = 1, \dots, r$, $\sum_{i=1}^{r}\alpha_i = 1$, and $\epsilon_{(A^i,b^i)}$ denotes the one-point measure in the given $m \times (n+1)$ matrix (A^i, b^i), $i = 1, \dots, r$.

Example: Scenario Analysis

Given a certain planning problem, in *scenario analysis* [1,2,6,7,8] the future evolution or development of the system to be considered is anticipated or explored by means of a (usually small) number r (e. g., $r = 3, 4, 5, 6$) of so-called scenarios $s^1, \dots, s^r$. Scenarios s^i, $i = 1, \dots, r$, are plausible alternative models of the future development given by 'extreme points' of a certain set of basic or key variables. An individual scenario or a certain mixture of the scenarios $s^1, \dots, s^r$ is assumed then to be revealed in the considered future time period. We assume now that the planningproblem can be described mathematically by the optimization problem

$$\begin{cases} \min & c'x \\ \text{s.t.} & Tx = (\leq)\, h, \quad x \in D. \end{cases} \tag{9}$$

Here, D is a given convex subset of $\mathbf{R}^n$, and the data (c, T, h) are given by $(c, T, h) = (c^i, T^i, h^i)$ for scenario s^i, $i = 1, \dots, r$, where c^i is an n-vector, T^i an $m \times n$ matrix and h^i an m-vector. Having written here the scenarios $s^1, \dots, s^r$ by means of (9) and the data (c^i, T^i, h^i), $i = 1, \dots, r$, and facing therefore the subproblems

$$\begin{cases} \min & c^{i\prime}x \\ \text{s.t.} & T^i x = (\leq)\, h^i, \quad x \in D, \end{cases} \tag{10}$$

for $i = 1, \dots, r$, the decision maker has then to select an appropriate decision $x \in D$. Since one is unable in general to predict with certainty which scenario s^i will occur, scenario analysts are looking for decisions x^0 which are 'robust' with respect to the different scenarios or 'scenario-independent', cf. [6,7,8]. Obviously, this robustness concept is closely related to the idea of detecting 'similarities' within the family of optimal solutions $x^*(s^i)$, $i = 1, \dots, r$, of the individual subproblems $(10)_{(i)}$, $i = 1, \dots, r$. Let $\alpha_1, \dots, \alpha_r$ with $\alpha_i > 0$, $i = 1, \dots, r$, $\sum_{i=1}^{r}\alpha_i = 1$, be (subjective) probabilities for the occurrence of $s^1, \dots, s^r$, or weights reflecting the relative importance of $s^1, \dots, s^r$. Considering loss functions $u \in C_m^J$ for evaluating the violations $z^i = T^i\, x - h^i$ of the constraint $T^i x = h^i$, $T^i x \leq h^i$, resp., in $(10)_{(\text{i})}$, a class of robust or scenario-independent decisions are obviously the efficient solutions of

$$\min_{x \in D} \sum_{i=1}^{r} \alpha_i \left(c^{i\prime} x + u(T^i x - h^i)\right), \tag{11}$$

which is a discretely distributed stochastic optimization problem of the type (1), (2).

A System of Linear Relations for the Construction of Descent Directions

Fundamental for the computation of the set $E_{D,\,U}$ of efficient solutions of (1), (2) is the following construction method for descent directions of the objective function F of (1), (2), cf. [3,4]. We suppose that the true, but unknown loss function u in (1) is, see Definition 1, an element of $\mathcal{C}_m^J$ for some known index set $J \subset \{1, \dots, m\}$. We recall that for any vector $z \in \mathbf{R}^m$ the subvectors z_I, z_{II} are defined by $z_I = (z_i)_{i \in J}$, $z_{II} = (z_i)_{i \notin J}$;see (3). Of course, if $J = \emptyset$, then $z = z_{II}$ and z_I does not exist. For any $m \times (n+1)$ matrix (A, b), let (A_I, b_I), (A_{II}, b_{II}), resp., denote the submatrices of (A, b) having the rows (A_i, b_i) with $i \in J$, $i \in \{1, \dots, \mathrm{m}\} \setminus J$, respectively.

Given an n-vector x (e. g., the tth iteration point of an algorithm for solving (1), (2)), we consider, in extension of [3, system (3.1)–(3.4b)], the following system of linear relationsfor the unknowns (y, Π), where $y \in \mathbf{R}^n$ and $\Pi = (\pi_{ij})$ is an auxiliary $r \times r$ matrix:

$$\sum_{j=1}^{r} \pi_{ij} = 1, \qquad \pi_{ij} \geq 0, \quad i, j = 1, \dots, r, \tag{12}$$

$$\alpha_j = \sum_{i=1}^{r} \alpha_i \pi_{ij}, \quad j = 1, \dots, r, \tag{13}$$

$$A_I^j y - b_I^j \leq \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j} (A_I^i x - b_I^i), \qquad j = 1, \dots, r, \tag{14}$$

$$A_{II}^j y - b_{II}^j = \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j} (A_{II}^i x - b_{II}^i), \qquad j = 1, \dots, r. \tag{15}$$

The transition probability measure

$$K^j = \sum_{i=1}^{r} \beta_{ij} \epsilon_{z^i}, \quad \beta_{ij} = \frac{\alpha_i \pi_{ij}}{\alpha_j}, \quad z^i = A^i x - b^i, \tag{16}$$

is not a one-point measure for at least one j, $1 \leq j \leq r$.

There exists at least one j, $1 \leq j \leq r$, such that for all $i = 1, \dots, r$

$$K^j \text{ is not a one-point measure and } \pi_{ij} > 0. \tag{17}$$

$$\text{At least one inequality in (14) holds with } < . \tag{18}$$

The constraint $x \in D$ in (1) can be handled by adding the condition

$$y \in D. \tag{19}$$

Remark 4

a) By ϵ_z we denote the one-point measure in a point $z \in \mathbf{R}^m$.

b) According to (12), Π is a stochastic matrix. System (12)–(15) has always the trivial solution $(y, \Pi) = (x, \mathrm{Id})$, where Id is the $r \times r$ identity matrix.

c) If (y, Π) solves (12)–(15), then

$$\overline{A}_I y \leq \overline{B}_I x, \quad \overline{A}_{II} y = \overline{B}_{II} x, \tag{20}$$

where $\overline{A}_I = \mathsf{E} A_I(\omega)$, $\overline{A}_{II} = \mathsf{E} A_{II}(\omega)$.

d) If $\mathsf{P}_{A_{II}(\cdot)y - b_{II}(\cdot)}$ denotes the probability distribution of the random $(m - |J|)$-vector $A_{II}(\omega)x - b_{II}(\omega)$, then (12), (13) and (15) mean that the distributions $\mathsf{P}_{A_{II}(\cdot)y - bII(\cdot)}$ and $\mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)}$ corresponding to y, x, resp., are related by

$$\begin{aligned}\mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)} &= K\mathsf{P}_{A_{II}(\cdot)y - b_{II}(\cdot)} \\ &= \int K(w, \cdot)\mathsf{P}_{A_{II}(\cdot)y - b_{II}(\cdot)}\,(dw),\end{aligned} \tag{21}$$

where $K(w, \cdot)$ is the *Markov kernel* defined by

$$K(w^j, \cdot) := K^j = \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j} \epsilon_{z^i}, \tag{22}$$

with $w^j = A^j y - b^j$, $z^i = A^i x - b^i$, $i, j = 1, \dots, r$. Since $\int zK(w, dz) = w$, the Markov kernel K is also calleda *dilatation*.

e) If n-vectors x, y are related by (21), (22), then for every convex subset $B \subset \mathbf{R}^m - |J|$ we have that

$$\mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)}(B) = 1 \implies \mathsf{P}_{A_{II}(\cdot)x - b_{II}(\cdot)}(B) = 1;$$

hence, the distribution of $A_{II}(\cdot)\, y - b_{II}(\cdot)$ is concentrated to the convex hull of the support of $\mathcal{P}_{A_{II}}(\cdot)x - b_{II}(\cdot)$.

f) If $J = \emptyset$, then (14) vanishes and (15) reads

$$A^j y - b^j = \sum_{i=1}^{r} \frac{\alpha_i \pi_{ij}}{\alpha_j}(A^i x - b^i), \qquad j = 1, \dots, r. \tag{23}$$

In the special case

$$(A_I^j, b_I^j) = (\overline{A}_I, \overline{b}_I) \quad \text{for all } j = 1, \dots, r, \tag{24}$$

i. e., if $(A_I(\omega), b_I(\omega))$ is constant with probability one, then (14) is reduced, cf. (20), to

$$\overline{A}_I y \leq \overline{A}_I x. \tag{25}$$

The meaning of (12)–(15) and the additional conditions (16)–(18) for the basic mean value minimization problem (1), (2) with objective function F is summarized in the next result.

Theorem 5 *Let J be any fixed subset of $\{1, \dots, m\}$.*

a) If (y, Π) is a solution of (12)–(15), then $F(y) \leq F(x)$ for every $u \in \mathcal{C}_m^J$. For $J = \emptyset$ also the converse holds: If there is a vector y such that $F(y) \leq F(x)$ for all $u \in \mathcal{C}_m$ $(\mathcal{C}_m^{\emptyset})$, then there exists an $r \times r$ matrix Π such that (y, Π) satisfies (12), (13) and (23).

b) If (y, Π) is a solution of (12)–(15) and (16), then $F(y) < F(x)$ for every $u \in \mathcal{C}_m^J$ which is strictly convex on $\operatorname{conv}\{z^i : 1 \leq i \leq r\}$.

c) If (y, Π) is a solution of (12)–(15) and (17), then $F(y) < F(x)$ for every $u \in \mathcal{C}_m^J$ which is not affine-linear on $\operatorname{conv}\{z^i : 1 \leq i \leq r\}$.

d) If (y, Π) fulfills (12)–(15) and (18), then $F(y) < F(x)$ for every $u \in \mathcal{C}_m^J$ satisfying (4).

Proof If x and (y, Π) are related by (12)–(15), then $F(y) \leq \sum_{j=1}^r \alpha_j u(\sum_{i=1}^r \beta_{ij} z^i)$ for every $u \in \mathcal{C}_m^J$.

If x, (y, Π) are related by (12)–(15) and (18), then $F(y) < \sum_{j=1}^r \alpha_j u(\sum_{i=1}^r \beta_{ij} z^i)$ for every $u \in \mathcal{C}_m^J$ fulfilling (4). The rest can then be shown as in [3, Thm. 2.2].

A simple, but important consequence of the above theorem is stated next:

Corollary 6 *For given $x \in \mathbf{R}^n$ or $x \in D$ let (y, Π) be any solution of (12)–(15) such that $y \neq x$, $y \in D \setminus \{x\}$, respectively.*

a) Then $h = y - x$ is a descent direction, a feasible descent direction, resp., of F at x for every $u \in \mathcal{C}_m^J$ such that F is not constant on the line segment $\overline{xy}$ joining x and y.

b) *If (y, II) fulfills also (16), (17), (18), resp., then $h = y - x$ is a (feasible) descent direction of F at x for every $u \in \mathcal{C}_m^J$ which is strictly convex on conv $\{z^i : 1 \leq i \leq r\}$, is not affine-linear on conv $\{z^i : 1 \leq i \leq r\}$, fulfills (4), respectively.*

Efficient Solutions of (1), (2)

In the following we suppose that the unknown loss function u is an element of $\mathcal{C}_m^J$, where J is a given subset of $\{1, \dots, m\}$. For a given point $x \in D$, the descent direction-finding procedure described in the previous section only can fail completely if for each solution (y, II) of (12)–(15) with $x \in D$ we have that

$$A^j y = A^j x \quad \text{for each } j = 1, \dots, r. \tag{26}$$

Indeed, in this case we either have $y = x$, or, for arbitrary loss functions u, the objective function F of (1), (2) is constant on the whole line through the points x, y. This observation suggests the following basic efficiency concept.

Definition 7 A point $x \in D$ is called a $(\mathcal{C}_m^J)$-*efficient point* or a $(\mathcal{C}_m^J)$-*efficient solution* of (1), (2) if and only if for each solution (y, II) of (12)–(15) with $y \in D$ we have that $A^j y = A^j x$ for each $j = 1, \dots, r$, i. e., $A(\omega)x = A(\omega)y$ with probability 1. Let $E_{D,J}$ denote the set of all efficient points of (1), (2).

For deriving parametric representations of $E_{D,J}$, we need the following definitions and lemmas.

For a given n-vector x and $z^i = A^i x - b^i$, $i = 1, \dots, r$, let $S = S_x \subset \{1, \dots, r\}$ with $|S| = s$ be an index set such that $\{z^i : 1 \leq i \leq r\} = \{z^i : i \in S\}$, where $z_i \neq z_j$ for $i, j \in S$, $i \neq j$. Defining for $i \in S$, $j = 1, \dots, r$, the quantities

$$\widetilde{\alpha_i} := \sum_{z^t = z^i} \alpha_t,$$
$$\tau_{ij} := \frac{1}{\widetilde{\alpha_i}} \sum_{z^t = z^i} \alpha_t \pi_{tj}, \quad \widetilde{\beta_{ij}} := \frac{\widetilde{\alpha_i} \tau_{ij}}{\alpha_j}, \tag{27}$$

we find that relations (12)–(15) can also be represented by

$$\sum_{j=1}^{r} \tau_{ij} = 1, \quad \tau_{ij} \geq 0, \quad j = 1, \dots, r, \quad i \in S, \tag{28}$$

$$\alpha_j = \sum_{i \in S} \widetilde{\alpha}_i \tau_{ij}, \quad j = 1, \dots, r, \tag{29}$$

$$A_I^j y - b_I^j \leq \sum_{i \in S} \widetilde{\beta}_{ij} z_I^i, \quad j = 1, \dots, r, \tag{30}$$

$$A_{II}^j y - b_{II}^j = \sum_{i \in S} \widetilde{\beta}_{ij} z_{II}^i, \quad j = 1, \dots, r. \tag{31}$$

For the next lemma we still need the $s \times r$ matrix $T^0 = (\tau_{ij}^0)$ defined by

$$\tau_{ij}^0 = \begin{cases} 0 & \text{if } z^i \neq z^j, \\ \frac{\alpha_j}{\widetilde{\alpha_i}} & \text{if } z^j = z^i, \end{cases} \quad \text{for } i \in S, \ j = 1, \dots, r. \tag{32}$$

Lemma 8 *Let (y, II) be a solution of (12)–(15), and let $T = T(II) = (\tau_{ij})$ be the $s \times r$ matrix having the elements τ_{ij} given by (27). If (26) holds, then $T(II) = T^0$ and (14) holds with '='.*

Lemma 8 implies the following important property of efficient solutions:

Corollary 9 *Let $x \in D$ be an efficient solution of (1), (2). If (y, II) is any solution of (21)–(22) with $y \in D$, then $T(II) = T^0$ and (14) holds with '='.*

For $J = \emptyset$ we obtain the set $E_D := E_{D,\emptyset}$ of all C_m-efficient solutions of (1), (2). This set is studied in [3]. An important relationship between E_D and E_D, J for any $J \subset \{1, \dots, m\}$ is given next:

Lemma 10 *$E_{D,J} \subset E_D$ for every $J \subset \{1, \dots, m\}$.*

Comparison of Definitions 7 and 3

Comparing the efficient solutions according to Definition 7 and the nondominated solutions according to Definition 3, first for $J = \emptyset$, i. e., $U = C_m$, we find the-following correspondence:

Theorem 11 $E_{D,\emptyset} = E_{D,C_m}^{(0)}$.

The next corollary follows immediately from the above theorem and Lemma 10.

Corollary 12 *$E_{D,J} \subset E_{D,\emptyset} = E_{D,C_m}^{(0)}$ for $J \subset \{1, \dots, m\}$.*

Considering now $U = C_m^J$ we have this inclusion:

Theorem 13 *$E_{D,J} \supset E_D, C_m^{J\,(0)}$ for $J \subset \{1, \dots, m\}$.*

The following inclusion follows from Corollary 12 and Theorem 13.

Corollary 14 *$E_{D,C_m^J}^{(0)} \subset E_{D,J} \subset E_{D,C_m}^{(0)}$ for $J \subset \{1, \dots, m\}$.*

A converse statement to Theorem 13 can be obtained for (24):

Theorem 15 *If $(A_I(\omega), b_I(\omega)) = (\overline{A}_I, \overline{b}_I)$ with probability 1, then $E_{D,J} = E^{(0)}_{D,C^J_m}$ for each $J \subset \{1, \dots, m\}$.*

Further Characterization of $E_{D,J}$

The C^J_m-efficiency of a point $x \in D$ can also be described in the following way.

Theorem 16 *A point $x \in D$ is (C^J_m)-efficient if and only if for every solution (y, II) of (12)-(15) we have that $A^j y = A^j x$ for all $j = 1, \dots, r$, or $h = y - x$ is not a feasible direction for D at x.*

Necessary Optimality Conditions Without Using (Sub)Gradients

If $x \in D$ is efficient, then, cf. Theorem 16, the descent direction-finding method described in in the previous Section fails at x. Since especially in any optimal solution x^* of (1), (2) no feasible descent direction may exist, efficient points are candidates for optimal solutions:

Theorem 17 *Suppose that for every $x \in D$ and every solution (y, II) of (12)-(15) with $y \in D$ the objective function F of (1), (2) with a loss function $u \in C^J_m$ is constant on the line segment xy if and only if $A^j y = A^j x$ for every $j = 1, \dots, r$. If x^* is an optimal solution of (1), (2), then $x^* \in E_{D,J}$.*

Remark 18 The assumption in Theorem 17 concerning F is fulfilled, e. g., if $u \in C^J_m$ is strictly convex on the convex hull conv $\{(A^j y - b^j)(A^j x - b^j): x, y \in D, 1 \le j \le r\}$ generated by the line segments $(A^j y - b^j)(A^j x - b^j)$ joining $(A^j y - b^j)$ and $(A^j x - b^j)$.

If the assumption in Theorem 17 concerning F does not hold, then it may happen that F is constant on a certain line segment xy though $A^j y \neq A^j x$ for at least one index j, $1 \le j \le r$. Hence, Theorem 17 can not be applied then directly. However, in this case the following modification of Theorem 17 holds true.

Theorem 19 *Let u be an arbitrary loss function from C^J_m for some $J \subset \{1, \dots, m\}$. If D is a compact convex subset of $\mathbf{R}^n$, then there exists at least one optimal solution x^* of (1), (2) lying in the closure $\overline{E}_{D,J}$ of the set $E_{D,J}$ of efficient solutions of (1), (2).*

Parametric Representation of $E_{D,J}$

Suppose that $u \in C^J_m$ for some index set $J \subset \{1, \dots, m\}$. For solving the descent direction-generating relations (12)–(15) and (16)–(18), resp., we may use, see Theorem 5 and Corollary 6, the quadratic program, cf. [3,4],

$$\begin{cases} \min & \eta'(\overline{A}_I y - \overline{A}_I x) + \sum_{j=1}^{r} \alpha_j \sum_{i \in S} \widetilde{\beta}^2_{ij} \\ \text{s.t.} & \sum_{j=1}^{r} \tau_{ij} = 1, \quad \tau_{ij} \ge 0, \\ & j = 1, \dots, r, \quad i \in S, \\ & \alpha_j = \sum_{i \in S} \widetilde{\alpha}_i \tau_{ij}, \quad j = 1, \dots, r, \\ & A^j_I y - b^j_I \le \sum_{i \in S} \widetilde{\beta}_{ij} z^i_I, \quad j = 1, \dots, r, \\ & A^j_{II} y - b^j_{II} = \sum_{i \in S} \widetilde{\beta}_{ij} z^i_{II}, \quad j = 1, \dots, r, \\ & y \in D, \end{cases} \tag{33}$$

where $\eta = (\eta_l)$ is a $|J|$-vector having fixed positive components η_l, $l \in J$. Efficient solutions of (1), (2) can be characterized as follows:

Lemma 20 *A vector $x \in D$ is an efficient solution of (1), (2) if and only if (33) has an optimal solution (y^*, T^*) such that $A^j y^* = A^j x$ for all $j = 1, \dots, r$.*

Remark 21 According to Lemma 8 we have then also that $T^* = T^0$ and (14) holds with '='.

We suppose now that the feasible domain D of (1), (2) is given by

$$D = \{x \in \mathbb{R}^n : \ g_k(x) \le 0, \ k = 1, \dots, \kappa\}. \tag{34}$$

Here, $g_1, \dots, g_\kappa$ are differentiable, convex functions. Moreover, we suppose that (33) has a feasible solution (y, T) such that for each nonaffine linear function g_k

$$g_k(y) < 0. \tag{35}$$

No constraint qualifications are needed in the important special case $D = \{x \in \mathbf{R}^n : G_x \le g\}$, where (G, g) is a given$\kappa \times (n + 1)$ matrix.

By means of the Kuhn-Tucker conditions of (33), the following *parametric representation* of $E_{D,J}$ can be derived [3,4]:

Theorem 22 *Let D be given by (34), and assume that the constraint qualification (35) holds for every $x \in D$. An n-vector x is an efficient solution of (1), (2) if and* *only if x satisfies the linear relations*

$$\lambda_j - \lambda_i - \left(\frac{\widehat{\gamma}_j}{\alpha_j} - \frac{\widehat{\gamma}_i}{\alpha_i}\right)' z^i = 2\left(\frac{1}{\alpha_i} - \frac{1}{\alpha_j}\right), \quad \text{if } z^i = z^j, \tag{36}$$

$$\lambda_j - \lambda_i - \left(\frac{\widehat{\gamma}_j}{\alpha_j} - \frac{\widehat{\gamma}_i}{\alpha_i}\right)' z^i \geq \frac{2}{\alpha_i}, \quad \text{if } z^i \neq z^j, \tag{37}$$

where $\lambda_1, \ldots, \lambda_r$ are arbitrary real parameters, and the parameter m-vectors $\gamma_1, \ldots, \gamma_r$ and further parameter vectors $\rho \in \boldsymbol{R}^\kappa$, $y \in \boldsymbol{R}^n$ are selected such that

$$\sum_{j=1}^{r} A^{j'} \widehat{\gamma}_j + \sum_{k=1}^{\kappa} \rho_k \nabla g_k(y) = 0, \tag{38}$$

$$\gamma_{jI} \geq 0, \quad j = 1, \ldots, r, \tag{39}$$

$$g_k(x) \leq 0, \quad k = 1, \ldots, \kappa, \tag{40}$$

$$g_k(y) \leq 0, \quad \rho_k g_k(y) = 0, \quad \rho_k \geq 0, \quad k = 1, \ldots, \kappa, \tag{41}$$

$$A^j y = A^j x, \quad j = 1, \ldots, r, \tag{42}$$

and the vectors $\widehat{\gamma}_j$ are defined by $\widehat{\gamma}_j = \binom{\alpha_j \eta + \gamma_{jI}}{\gamma_{jII}}$, j = 1, …, r.

See also

- ▶ Approximation of Extremum Problems with Probability Functionals
- ▶ Approximation of Multivariate Probability Integrals
- ▶ Extremum Problems with Probability Functions: Kernel Type Solution Methods
- ▶ General Moment Optimization Problems
- ▶ Logconcave Measures, Logconvexity
- ▶ Logconcavity of Discrete Distributions
- ▶ L-shaped Method for Two-Stage Stochastic Programs with Recourse
- ▶ Multistage Stochastic Programming: Barycentric Approximation
- ▶ Preprocessing in Stochastic Programming
- ▶ Probabilistic Constrained Linear Programming: Duality Theory
- ▶ Probabilistic Constrained Problems: Convexity Theory
- ▶ Simple Recourse Problem: Dual Method
- ▶ Simple Recourse Problem: Primal Method
- ▶ Stabilization of Cutting Plane Algorithms for Stochastic Linear Programming Problems
- ▶ Static Stochastic Programming Models
- ▶ Static Stochastic Programming Models: Conditional Expectations
- ▶ Stochastic Integer Programming: Continuity, Stability, Rates of Convergence
- ▶ Stochastic Integer Programs
- ▶ Stochastic Linear Programming: Decomposition and Cutting Planes
- ▶ Stochastic Linear Programs with Recourse and Arbitrary Multivariate Distributions
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Stochastic Programming: Minimax Approach
- ▶ Stochastic Programming Models: Random Objective
- ▶ Stochastic Programming: Nonanticipativity and Lagrange Multipliers
- ▶ Stochastic Programming with Simple Integer Recourse
- ▶ Stochastic Programs with Recourse: Upper Bounds
- ▶ Stochastic Quasigradient Methods in Minimax Problems
- ▶ Stochastic Vehicle Routing Problems
- ▶ Two-Stage Stochastic Programming: Quasigradient Method
- ▶ Two-Stage Stochastic Programs with Recourse

References

1. Carroll JM (ed) (1995) Scenario-based design. WileyNew York , New York
2. Chandler J, Cockle P (1982) Techniques of scenario planning. McGraw-Hill, New York
3. Marti K (1988) Descent directions and efficient solutions in discretely distributed stochastic programs. Lecture Notes Economics and Math Systems, vol 299. Springer, Berlin
4. Marti K (1992) Computation of efficient solutions of discretely distributed stochastic optimization problems. ZOR 36:259–294
5. Marti K (1996) Stochastic optimization methodsin engineering. In: Dolezal J, Fidler J (eds) System Modelling and Optimization. Chapman and Hall, London, pp 75–87
6. Reibnitz Uvon (1988) Scenario techniques. McGraw-Hill, New York
7. Steinsiek E, Knauer P (1981) Szenarien als Instrument der Umweltplanung. Angewandte Systemanalyse 2:10–19
8. Zentner RD (1975) Scenarios: A new tool for corporate planners. Chem and Eng News, Internat Ed 53:22–34

Discrete Stochastic Optimization

GEORG PFLUG
University Vienna, Vienna, Austria

MSC2000: 90C15, 90C27

Article Outline

Keywords

Derivatives; Stochastic optimization

A *stochastic combinatorial optimization* problem is of the form

$$\begin{cases} \min & F(x) = \int H(x,v)\, d\mu_x(v) \\ \text{s.t.} & x \in S \end{cases} \tag{1}$$

where $S = \{x_1, \dots, x_N\}$ is a finite discrete feasible set. If the value of the objective function F is easily obtainable, the problem is just a deterministic combinatorial optimization problem. In most applications however, the value of the objective function F has to be approximated by numerical integration or Monte-Carlo simulation.

Some problems of type (1) exhibit a special structure, which can be exploited for solution methods, like the *stochastic linear optimization problems*, where S are the integer points of a convex polyhedron and H is piecewise linear (see ▶ Stochastic integer programming: Continuity, stability, rates of convergence). In this contribution, we discuss problems with an arbitrary and unstructured feasible set S.

An example is the stochastic single machine tardiness problem (SSMTP): The optimal sequence of m jobs, which are processed on a single machine has to be found. Each job has a random processing time, which is distributed according to the distribution function G_i, $i = 1, \dots, m$ (independent of all others), and a fixed due date d_i. The feasible set S is the set of all $m!$ permutations π of $\{1, \dots, m\}$. If π is the solution found, we process job $\pi(1)$ as the first, $\pi(2)$ as the second and so on. Let $c_i(u)$ be the costs for job i being late u time units ($c_i(u) = 0$ for $u \le 0$). The SSMTP is

$$\begin{cases} \min & \sum_{i=1}^{m} \mathsf{E}[c_i(V_{\pi(1)} + \cdots + V_{\pi(i)} - d_{\pi(i)})] \\ \text{s.t.} & \pi \in S \end{cases} \tag{2}$$

where V_i are random variables distributed independently according to G_i. The analytic calculation of the objective function (OF) in (2) involves multiple integrals (the convolution of up to m distribution functions). A simple way of approximating the OF is by Monte-Carlo simulation. Let $V_i^{(1)}, \dots, V_i^{(n)}$ be independent random (pseudorandom) variables, each with distribution G_i. The true expectation $F(\pi) = \mathsf{E}[c_i(V_{\pi(1)} + \cdots + V_{\pi(\mathrm{i})} - d\pi(\mathrm{i}))]$ is approximated by the estimate

$$\widehat{F}_n(\pi) = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} [c_i(V_{\pi(1)}^{(j)} + \cdots + V_{\pi(i)}^{(j)} - d_{\pi(i)})].$$

In principle, all exact (branch and bound) and heuristic (evolutionary algorithms, tabu search, ant systems, random search, simulated annealing, genetic algorithms) methods for combinatorial optimization may be applied to stochastic combinatorial optimization — just that the exact values $F(x)$ have to be replaced by stochastic estimates $\widehat{F}_n(x)$, which are based on sample size n.

The main difficulty in stochastic combinatorial optimization is the fact that even if $F(x_1) \le F(x_2) - \delta$, it may happen with positive probability that $\widehat{F}_n(x_1) > \widehat{F}_n(x_2)$, that is we may wrongly conclude that x_2 is better than x_1. The probability of this error decreases to zero with sample size n increasing to infinity. A compromise between the quality of the solution and the costs of very large samples has to be found in stochastic optimization.

If the random distribution μ_x in (1) does not depend on x, *common random numbers* may be taken. To be more precise, let $V^{(1)}, \dots, V^{(n)}$ be a sample from μ and let

$$\widehat{F}_n(x_i) = \frac{1}{n} \sum_{j=1}^{n} H(x_i, V^{(j)}).$$

The estimates $\widehat{F}$ are now correlated, and the probability that $\widehat{F}_n(x_1) > \widehat{F}_n(x_2)$ although $\widehat{F}_n(x_1) \le \widehat{F}_n(x_2) - \delta$

is typically much smaller for such a choice than with samples taken independently for each x_j (see [6]).

If the estimates $\widehat{F}$ are difficult to get (e. g. they need real observation or expensive simulation) allocation rules decide, which estimate or which set of estimates has to be taken next. These rules try to exclude quickly subsets of the feasible set, which – with high statistical evidence – do not contain optimal solutions. The effort is then concentrated on the (shrinking) set of not yet excluded points.Allocation rules may be based on *subset selection* (see [3]) or *ordinal optimization* (see [5]). There is also a connection to experimental design, in particular to *sequential experimental design*: In experimental design one has to choose the next point(s) for sampling, which – based on the information gathered so far – will give the best additional information which we need to solve the underlying estimation or optimization problem (for experimental design literature see [1] and the references therein).

For large sets S, which have graph-neighborhood or partition structures, 'stochastic' variants of neighbor search or branch and bound methods may be used. In particular, stochastic simulated annealing and stochastic branch and bound have been studied in literature.

Stochastic Simulated Annealing

This is a variant of ordinary simulated annealing (cf. ► Simulated annealing): The Metropolis rule for the acceptance probability is calculated on the basis of the current stochastic estimates of the objective function, i. e. the new state x_j is preferred to the current state x_i with probability

$$\min\left(\exp\left(-\frac{\widehat{F}_n(x_j)-\widehat{F}_n(x_i)}{k_B T}\right),1\right)$$

where k_B is the Boltzmann constant and T is the temperature. The estimates $\widehat{F}$ are improved in each step by taking additional observations, i. e. increasing the sample size n. For an analysis of this algorithm see [4].

Stochastic Branch and Bound

For the implementation of a stochastic branch and bound method (cf. also ► Integer programming: Branch and bound methods), an estimate of a lower bound function is needed. Recall that a function $\mathbf{F}$, defined on the subsets of S, is called a *lower bound function* if

$$\inf\{F(x)\colon\ x\in T\}\geq \mathbf{F}(T)$$

for all $T\subseteq S$.

In stochastic branch and bound an estimate $\widehat{\mathbf{F}}$ of $\mathbf{F}$ can be found for instance by sampling $\widehat{F}_n(x_i)$ for each x_i in T with $\mathsf{E}(\widehat{F}_n(x_i)) = F(x_i)$ and setting

$$\widehat{\mathbf{F}}(T)=\inf\left\{\widehat{F}_n(x_i)\colon\ x_i\in T\right\}.$$

The bound-step of the branch and bound method is replaced by a statistical test, whether the lower bound estimate of a branch is significantly larger than the estimate of an intermediate solution. After each step, all estimates are improved by taking additional observations. For details see [2] and [7].

In all these algorithms, common random numbers may decrease the variance.

See also

► Derivatives of Markov Processes and Their Simulation
► Derivatives of Probability and Integral Functions: General Theory and Examples
► Derivatives of Probability Measures
► Optimization in Operation of Electric and Energy Power Systems

References

1. Chernoff H (1989) Sequential analysis and optimal design. SIAM, Philadelphia
2. Ermoliev YM, Norkin VI, Ruszczyński A (1998) On optimal allocation of indivisibles under uncertainty. Oper Res 46:381–395
3. Futschik A, Pflug GCh (1997) Optimal allocation of simulation experiments in discrete stochastic optimization and approximative algorithms. Europ J Oper Res 101:245–260
4. Gutjahr W, Pflug G (1996) Simulated annealing for noisy cost functions. J Global Optim 8:1–13
5. Ho YC, Sreenivas RS, Vakili P (1992) Ordinal optimization of DEDS. J Discret Event Dynamical Systems 2:61–88
6. Kleywegt AJ, Shapiro A (1999) The sample average approximation method for stochastic discrete optimization. Georgia Inst Technol, Atlanta, GA
7. Norkin VI, Pflug GCh, Ruszczyński A (1998) A branch and bound method for stochastic global optimization. Math Program 83:425–450

Disease Diagnosis: Optimization-Based Methods

EVA K. LEE, TSUNG-LIN WU
Center for Operations Research in Medicine and HealthCare,
School of Industrial and Systems Engineering,
Georgia Institute of Technology,
Atlanta, USA

MSC2000: 90C05, 90C06, 90C10, 90C11, 90C20, 90C30, 90C90, 90-08, 65K05

Article Outline

Abstract

In this chapter, we present classification models based on mathematical programming approaches. We first provide an overview of various mathematical programming approaches, including linear programming, mixed integer programming, nonlinear programming, and support vector machines. Next, we present our effort of novel optimization-based classification models that are general purpose and suitable for developing predictive rules for large heterogeneous biological and medical data sets. Our predictive model simultaneously incorporates (1) the ability to classify any number of distinct groups; (2) the ability to incorporate heterogeneous types of attributes as input; (3) a high-dimensional data transformation that eliminates noise and errors in biological data; (4) the ability to incorporate constraints to limit the rate of misclassification, and a reserved-judgment region that provides a safeguard against overtraining (which tends to lead to high misclassification rates from the resulting predictive rule); and (5) successive multistage classification capability to handle data points placed in the reserved-judgment region. To illustrate the power and flexibility of the classification model and solution engine, and its multigroup prediction capability, application of the predictive model to a broad class of biological and medical problems is described. Applications include the differential diagnosis of the type of erythemato-squamous diseases; predicting presence/absence of heart disease; genomic analysis and prediction of aberrant CpG island methylation in human cancer; discriminant analysis of motility and morphology data in human lung carcinoma; prediction of ultrasonic cell disruption for drug delivery; identification of tumor shape and volume in treatment of sarcoma; multistage discriminant analysis of biomarkers for prediction of early atherosclerois; fingerprinting of native and angiogenic microvascular networks for early diagnosis of diabetes, aging, macular degeneracy, and tumor metastasis; prediction of protein localization sites; and pattern recognition of satellite images in classification of soil types. In all these applications, the predictive model yields correct classification rates ranging from 80 to 100%. This provides motivation for pursuing its use as a medical diagnostic, monitoring and decision-making tool.

Introduction

Classification is a fundamental machine learning task whereby rules are developed for the allocation of independent observations to groups. Classic examples of applications include medical diagnosis – the allocation of patients to disease classes on the basis of symptoms and laboratory tests – and credit screening – the acceptance or rejection of credit applications on the basis of applicant data. Data are collected concerning observations with known group membership. These *training data* are used to develop rules for the classification of future observations with unknown group membership.

In this introduction, we briefly describe some terminologies related to classification, and provide a brief description of the organization of this chapter.

Pattern Recognition, Discriminant Analysis, and Statistical Pattern Classification

Cognitive science is the science of learning, knowing, and reasoning. *Pattern recognition* is a broad field within *cognitive science*, which is concerned with the process of recognizing, identifying, and categorizing input information. These areas intersect with computer science, particularly in the closely related areas of *artificial intelligence*, *machine learning*, and *statistical pattern recognition*. Artificial intelligence is associated with constructing machines and systems that reflect human abilities in cognition. Machine learning refers to how these machines and systems replicate the learning process, which is often achieved by seeking and discovering patterns in data, or statistical pattern recognition.

Discriminant analysis is the process of discriminating between categories or populations. Associated with discriminant analysis as a statistical tool are the tasks of determining the features that best discriminate between populations, and the process of classifying new objects on the basis of these features. The former is often called *feature selection* and the latter is referred to as *statistical pattern classification*. This work will be largely concerned with the development of a viable statistical pattern classifier.

As with many computationally intensive tasks, recent advances in computing power have led to a sharp increase in the interest and application of discriminant analysis techniques. The reader is referred to Duda et al. [25] for an introduction to various techniques for pattern classification, and to Zopounidis and Doumpos [121] for examples of applications of pattern classification.

Supervised Learning, Training, and Cross-Validation

An *entity* or *observation* is essentially a data point as commonly understood in statistics. In the framework of statistical pattern classification, an entity is a set of quantitative measurements (or qualitative measurements expressed quantitatively) of *attributes* for a particular object. As an example, in medical diagnosis an entity could be the various blood chemistry levels of a patient. With each entity is associated one or more *groups* (or *populations*, *classes*, *categories*) to which it belongs. Continuing with the medical diagnosis example, the groups could be the various classes of heart disease. Statistical classification seeks to determine rules for associating entities with the groups to which they belong. Ideally, these associations align with the associations that human reasoning would produce on the basis of information gathered on objects and their apparent categories.

Supervised learning is the process of developing classification rules based on entities for which the classification is already known. Note that the process implies that the populations are already well defined. *Unsupervised learning* is the process of discovering patterns from unlabeled entities and thereby discovering and describing the underlying populations. Models derived using supervised learning can be used for both functions of discriminant analysis – feature selection and classification. The model that we consider is a method for supervised learning, so we assume that populations are previously defined.

The set of entities with known classification that is used to develop classification rules is the *training set*. The training set may be partitioned so that some entities are withheld during the model-development process, also known as the *training* of the model. The withheld entities form a *test set* that is used to determine the validity of the model, a process known as *cross-validation*. Entities from the test set are subjected to the rules of classification to measure the performance of the rules on entities with unknown group membership.

Validation of classification models is often performed using *m*-fold cross-validation where the data with known classification are partitioned into *m folds* (subsets) of approximately equal size. The classification model is trained *m* times, with the *m*th fold withheld during each run for testing. The performance of the model is evaluated by the classification accuracy on the *m* test folds, and can be represented using a *classification matrix* or *confusion matrix*.

The classification matrix is a square matrix with the number of rows and columns equal to the number of groups. The *ij*th entry of the classification matrix contains the number or proportion of test entities from group *i* that were classified by the model as belonging to group *j*. Therefore, the number or proportion of correctly classified entities is contained in the diagonal elements of the classification matrix, and the number or proportion of misclassified entities is in the off-diagonal entries.

Bayesian Inference and Classification

The popularity of *Bayesian inference* has risen drastically over the past several decades, perhaps in part due to its suitability for statistical learning. The reader is referred to O'Hagan [92] for a thorough treatment of Bayesian inference. Bayesian inference is usually contrasted against *classical inference*, though in practice they often imply the same methodology.

The Bayesian method relies on a *subjective* view of probability, as opposed to the *frequentist* view upon which classical inference is based [92]. A subjective probability describes a degree of belief in a *proposition* held by the investigator based on some information. A frequency probability describes the likelihood of an *event* given an infinite number of trials.

In Bayesian statistics, inferences are based on the *posterior distribution*. The posterior distribution is the product of the *prior probability* and the *likelihood function*. The prior probability distribution represents the initial degree of belief in a proposition, often before empirical data are considered. The likelihood function describes the likelihood that the behavior is exhibited, given that the proposition is true. The posterior distribution describes the likelihood that the proposition is true, given the observed behavior.

Suppose we have a proposition or random variable θ about which we would like to make inferences, and data x. Application of Bayes's theorem gives

$$\mathrm{d}F(\theta|x) = \frac{\mathrm{d}F(\theta)\,\mathrm{d}F(x|\theta)}{\mathrm{d}F(x)}\,.$$

Here, F denotes the (cumulative) distribution function. For ease of conceptualization, assume that F is differentiable, then $\mathrm{d}F = f$, and the above equality can be rewritten as

$$f(\theta|x) = \frac{f(\theta)f(x|\theta)}{f(x)}\,.$$

For classification, a prior probability function $\pi(g)$ describes the likelihood that an entity is allocated to group g regardless of its exhibited feature values x. A group density function $f(x|g)$ describes the likelihood that an entity exhibits certain measurable attribute values, given that it belongs to population g. The posterior distribution for a group $P(g|x)$ is given by the product of the prior probability and group density function, normalized over the groups to obtain a unit probability over all groups. The observation x is allocated to group h if $h = \arg\max_{g\in\mathcal{G}} P(g|x) = \arg\max_{g\in\mathcal{G}} \frac{\pi(g)f(x|g)}{\sum_{j\in\mathcal{G}}\pi(j)f(x|j)}$, where $\mathcal{G}$ denotes the set of groups.

Discriminant Functions

Most classification methods can be described in terms of *discriminant functions*. A discriminant function takes as input an observation and returns information about the classification of the observation. For data from a set of groups $\mathcal{G}$, an observation x is assigned to group h if $h = \arg\max_{g\in\mathcal{G}} l_g(x)$, where the functions l_g are the discriminant functions. Classification methods restrict the form of the discriminant functions, and training data are used to determine the values of the parameters that define the functions.

The optimal classifier in the Bayesian framework can be described in terms of discriminant functions. Let $\pi_g = \pi(g)$ be the prior probability that an observation is allocated to group g and let $f_g(x) = f(x|g)$ be the likelihood that data x are drawn from population g. If we wish to minimize the probability of misclassification given x, then the optimal allocation for an entity is to the group $h = \arg\max_{g\in\mathcal{G}} P(g|x) = \arg\max_{g\in\mathcal{G}} \frac{\pi_g f_g(x)}{\sum_{j\in\mathcal{G}}\pi_j f_j(x)}$. Under the Bayesian framework,

$$P(g|x) = \frac{\pi_g f(x|g)}{f(x)} = \frac{\pi_g f(x|g)}{\sum\limits_{j\in\mathcal{G}} \pi_j f(x|j)}\,.$$

The discriminant functions can be $l_g(x) = P(g|x)$ for $g \in \mathcal{G}$. The same classification rule is given by $l_g(x) = \pi_g f(x|g)$ and $l_g(x) = \log f(x|g) + \log \pi_g$. The problem then becomes finding the form of the prior functions and likelihood functions that match the data.

If the data are multivariate normal with equal covariance matrices ($f(x|g) \sim N(\mu_g, \Sigma)$), then a linear discriminant function (LDF) is optimal:

$$\begin{aligned} l_g(x) &= \log f(x|g) + \log \pi_g \\ &= -1/2(x-\mu_g)^T \Sigma^{-1}(x-\mu_g) - 1/2 \log |\Sigma_g| \\ &\quad - d/2 \log 2\pi + \log \pi_g \\ &= w_g^T x + w_{g0} , \end{aligned}$$

where d is the number of attributes, $w_g = \Sigma^{-1}\mu_g$, and $w_{g0} = -1/2\mu_g^T \Sigma^{-1}\mu_g + \log \pi_g + x^T \Sigma^{-1} x - d/2 \log 2\pi$. Note that the last two terms of w_{g0} are constant for all g and need not be calculated. When there are two groups ($\mathcal{G} = \{1, 2\}$) and the priors are equal ($\pi_1 = \pi_2$), the discriminant rule is equivalent to Fisher's linear discriminant rule [30]. Fisher's rule can also be derived, as it was by Fisher, by choosing w so that $(w^T\mu_1 - w^T\mu_2)^2/(w^T\Sigma w)$ is maximized.

These LDFs and quadratic discriminant functions (QDFs) are often applied to data sets that are not multivariate normal or continuous (see pp. 234–235 in [98]) by using approximations for the means and covariances. Regardless, these models are *parametric* in that they incorporate assumptions about the distribution of the data. Fisher's LDF is *nonparametric* because no assumptions are made about the underlying distribution of the data. Thus, for a special case, a parametric and a nonparametric model coincide to produce the same discriminant rule. The LDF derived above is also called the *homoscedastic model*, and the QDF is called the *heteroscedastic model*. The exact form of discriminant functions in the Bayesian framework can be derived for other distributions [25].

Some classification methods are essentially methods for finding coefficients for LDFs. In other words, they seek coefficients w_g and constants w_{g0} such that $l_g(x) = w_g x + w_{g0}$, $g \in \mathcal{G}$ is an optimal set of discriminant functions. The criteria for optimality are different for different methods. LDFs project the data onto a linear subspace and then discriminate between entities in that subspace. For example, Fisher's LDF projects two-group data on an optimal line, and discriminates on that line. A good linear subspace may not exist for data with overlapping distributions between groups and therefore the data will not be classified accurately using these methods. The hyperplanes defined by the discriminant functions form boundaries between the group regions. A large portion of the literature concerning the use of mathematical programming models for classification describes methods for finding coefficients of LDFs [121].

Other classification methods seek to determine parameters to establish QDFs. The general form of a QDF is $l_g(x) = x^T W_g x + w_g^T x + w_{g0}$. The boundaries defining the group regions can assume any hyperquadric form, as can the Bayes decision rules for arbitrary multivariate normal distributions [25].

In this paper, we survey the development and advances of classification models via the mathematical programming techniques, and summarize our experience in classification models applied to prediction in biological and medical applications. The rest of this chapter is organized as follows. Section "Mathematical Programming Approaches" first provides a detailed overview of the development and advances of mathematical programming based classification models, including linear programming (LP), mixed integer programming (MIP), nonlinear programming, and support vector machine (SVM) approaches. In Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology", we describe our effort in developing optimization-based multigroup multistage discriminant analysis predictive models for classification. The use of the predictive models for various biological and medical problems is presented. Section "Progress and Challenges" provides several tables to summarize the progress of mathematical programming based classification models and their characteristics. This is followed by a brief description of other classification methods in Sect. "Other Methods", and by a summary and concluding remarks in Sect. "Summary and Conclusion".

Mathematical Programming Approaches

Mathematical programming methods for statistical pattern classification emerged in the 1960s, gained popularity in the 1980s, and have grown drastically since. Most of the mathematical programming approaches are nonparametric, which has been cited as an advantage when analyzing contaminated data sets over methods that require assumptions about the distribution of the data [107]. Most of the literature about mathematical programming methods is concerned with either using mathematical programming to determine the coefficients of LDFs or *support vector machines* (SVMs).

The following notation will be used. The subscripts i, j, and k are used for the observation, attribute, and group, respectively. Let x_{ij} be the value of attribute j of observation i. Let m be the number of attributes, K be the number of groups, G_k represent the set of data from group k, M be a big positive number, and ϵ be a small positive number. The abbreviation "urs" is used in reference to a variable to denote "unrestricted in sign."

Linear Programming Classification Models

The use of linear programs to determine the coefficients of LDFs has been widely studied [31,46,50,74]. The methods determine the coefficients for different objectives, including minimizing the sum of the distances to the separating hyperplane, minimizing the maximum distance of an observation to the hyperplane, and minimizing other measures of badness of fit or maximizing measures of goodness of fit.

Two-Group Classification One of the earliest LP classification models was proposed by Mangasarian [74] to construct a hyperplane to separate two groups of data. Separation by a nonlinear surface using LP was also proposed when the surface parameters appear linearly. Two sets of points may be inseparable by one hyperplane or surface through a single-step LP approach, but they can be strictly separated by more planes or surfaces via a multistep LP approach [75]. In [75] real problems with up to 117 data points, ten attributes, and three groups were solved. The three-group separation was achieved by separating group 1 from groups 2 and 3, and then group 2 from group 3.

Studies of LP models for the discriminant problem in the early 1980s were carried out by Hand [47], Freed and Glover [31,32], and Bajgier and Hill [5]. Three LP models for the two-group classification problem, including minimizing the sum of deviations (MSD), minimizing the maximum deviation (MMD), and minimizing the sum of interior distances (MSID) were proposed. Freed and Glover [33] provided computational studies of these models where the test conditions involved normal and nonnormal populations.

MSD:

$$\begin{aligned}
\text{Minimize} \quad & \sum_i d_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le 0 \quad \forall i \in G_1\,, \\
& w_0 + \sum_j x_{ij} w_j + d_i \ge 0 \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d_i \ge 0 \quad \forall i\,.
\end{aligned}$$

MMD:

$$\begin{aligned}
\text{Minimize} \quad & d \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d \le 0 \quad \forall i \in G_1\,, \\
& w_0 + \sum_j x_{ij} w_j + d \ge 0 \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d \ge 0\,.
\end{aligned}$$

MSID:

$$\begin{aligned}
\text{Minimize} \quad & pd - \sum_i e_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d + e_i \le 0 \quad \forall i \in G_1\,, \\
& w_0 + \sum_j x_{ij} w_j + d - e_i \ge 0 \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d \ge 0\,, \\
& e_i \ge 0 \quad \forall i\,,
\end{aligned}$$

where p is a weight constant.

The objective function of the MSD model is the L_1-norm distance, while the objective function of MMD is the L_∞-norm distance. They are special cases of L_p-norm classification [50,108].

In some models the constant term of the hyperplane is a fixed number instead of a decision variable. The model minimize the sum of deviations with constant cutoff score MSD^0 shown below is an example where the cutoff score b replaces w_0 in the formulation. The same replacement could be used in other formulations.

MSD^0:

$$\begin{aligned}
\text{Minimize}\quad & \sum_i d_i \\
\text{subject to}\quad & \sum_j x_{ij} w_j - d_i \le b \quad \forall i \in G_1\,, \\
& \sum_j x_{ij} w_j + d_i \ge b \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d_i \ge 0 \quad \forall i\,.
\end{aligned}$$

A gap can be introduced between the two regions determined by the separating hyperplane to prevent degenerate solutions. Take MSD as an example; the separation constraints become

$$w_0 + \sum_j x_{ij} w_j - d_i \le -\epsilon \quad \forall i \in G_1\,,$$

$$w_0 + \sum_j x_{ij} w_j + d_i \ge \epsilon \quad \forall i \in G_2\,.$$

The small number ϵ can be normalized to 1.

Besides introducing a gap, another normalization approach is to include constraints such as $\sum_{j=0}^{m} w_j = 1$ or $\sum_{j=1}^{m} w_j = 1$ in the LP models to avoid unbounded or trivial solutions.

Specifically, Glover et al. [45] gave the hybrid model, as follows.

Hybrid model:

$$\begin{aligned}
\text{Minimize}\quad & pd + \sum_i p_i d_i - qe - \sum_i q_i e_i \\
\text{subject to}\quad & w_0 + \sum_j x_{ij} w_j - d - d_i + e + e_i = 0 \\
& \quad \forall i \in G_1\,, \\
& w_0 + \sum_j x_{ij} w_j + d + d_i - e - e_i = 0 \\
& \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d, e \ge 0\,, \\
& d_i, e_i \ge 0 \quad \forall i\,,
\end{aligned}$$

where p, p_i, q, q_i are the costs for different deviations. Including different combinations of deviation terms in the objective function then leads to variant models.

Joachimsthaler and Stam [50] reviewed and summarized LP formulations applied to two-group classification problems in discriminant analysis, including MSD, MMD, MSID, and MIP models, and the hybrid model. They summarized the performance of the LP methods together with the traditional classification methods such as Fisher's LDF [30], Smith's QDF [106], and a logistic discriminant method. In their review, MSD sometimes but not uniformly improves classification accuracy, compared with traditional methods. On the other hand, MMD is found to be inferior to MSD. Erenguc and Koehler [27] presented a unified survey of LP models and their experimental results, in which the LP models include several versions of MSD, MMD, MSID, and hybrid models. Rubin [99] provided experimental results comparing these LP models with Fisher's LDF and Smith's QDF. He concluded that QDF performs best when the data follow normal distributions and that QDF could be the benchmark when seeking situations for advantageous LP methods. In summary, the above mentioned review papers [27,50,99] describe previous work on LP classification models and their comparison with traditional methods. However, it is difficult to make definitive statements about the conditions under which one LP model is superior to others, as stated in [107].

Stam and Ungar [110] introduced the software package RAGNU, a utility program in conjunction with the LINDO optimization software, for solving two-group classification problems using LP-based methods. LP formulations such as MSD, MMD, MSID, hybrid models, and their variants are contained in the package.

There are some difficulties in LP-based formulations, in that some models could result in unbounded, trivial, or unacceptable solutions [34,87], but possible remedies are proposed. Koehler [51,52,53] and Xiao [114,115] characterized the conditions of unacceptable solutions in two-group LP discriminant models, including MSD, MMD, MISD, the hybrid model, and their variants. Glover [44] proposed the normalization constraint $\sum_{j=1}^{m}(-|G_2| \sum_{i \in G_1} x_{ij} + |G_1| \sum_{i \in G_2} x_{ij}) w_j = 1$, which is more effective and reliable. Rubin [100] examined the separation failure for two-group models and suggested applying the models twice, reversing the group designations the second time. Xiao and Feng [116] proposed a regularization method to avoid multiple solutions in LP discriminant analysis by adding the term $\epsilon \sum_{j=1}^{m} w_j^2$ in the objective functions.

Bennett and Mangasarian [9] proposed the following model which minimizes the average of the deviations, which is called robust LP (RLP):

$$\begin{aligned}
\text{Minimize} \quad & \frac{1}{|G_1|}\sum_{i\in G_1} d_i + \frac{1}{|G_2|}\sum_{i\in G_2} d_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij}w_j - d_i \le -1 \quad \forall i \in G_1\,, \\
& w_0 + \sum_j x_{ij}w_j + d_i \ge 1 \quad \forall i \in G_2\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d_i \ge 0 \quad \forall i\,.
\end{aligned}$$

It is shown that this model gives the null solution $w_1 = \cdots = w_m = 0$ if and only if $\frac{1}{|G_1|}\sum_{i\in G_1} x_{ij} = \frac{1}{|G_2|}\sum_{i\in G_2} x_{ij}$ for all j, in which case the solution $w_1 = \cdots = w_m = 0$ is guaranteed to be not unique. Data of different diseases have been tested by the proposed classification methods, as in most of Mangasarian's papers.

Mangasarian et al. [86] described two applications of LP models in the field of breast cancer research, one in diagnosis and the other in prognosis. The first application is to discriminate benign from malignant breast lumps, while the second one is to predict when breast cancer is likely to recur. Both of them work successfully in clinical practice. The RLP model [9] together with the multisurface method tree algorithm [8] is used in the diagnostic system.

Duarte Silva and Stam [104] included the second-order (i. e., quadratic and cross-product) terms of the attribute values in the LP-based models such as MSD and hybrid models and compared them with linear models, Fisher's LDF, and Smith's QDF. The results of the simulation experiments show that the methods which include second-order terms perform much better than first-order methods, given that the data substantially violate the multivariate normality assumption. Wanarat and Pavur [113] investigated the effect of the inclusion of the second-order terms in the MSD, MIP, and hybrid models when the sample size is small to moderate. However, the simulation study shows that second-order terms may not always improve the performance of a first-order LP model even with data configurations that are more appropriately classified by Smith's QDF. Another result of the simulation study is that inclusion of the cross-product terms may hurt the model's accuracy, while omission of these terms causes the model to be not invariant with respect to a nonsingular transformation of the data.

Pavur [94] studied the effect of the position of the contaminated normal data in the two-group classification problem. The methods for comparison in that study included MSD, minimizing the number of misclassifications (MM; (described in the "Mixed Integer Programming Classification Models" section), Fisher's LDF, Smith's QDF, and nearest -neighbor models. The nontraditional methods such as LP models have potential for outperforming the standard parametric procedures when nonnormality is present, but this study shows that no one model is consistently superior in all cases.

Asparoukhov and Stam [3] proposed LP and MIP models to solve the two-group classification problem where the attributes are binary. In this case the training data can be partitioned into multinomial cells, allowing for a substantial reduction in the number of variables and constraints. The proposed models not only have the usual geometric interpretation, but also possess a strong probabilistic foundation. Let s be the index of the cells, n_{1s}, n_{2s} be the number of data points in cell s from groups 1 and 2, respectively, and $(b_{s1}, \ldots, b_{sm})$ be the binary digits representing cell s. The model shown below is the LP model of minimizing the sum of deviations for two-group classification with binary attributes.

Cell conventional MSD:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{s:\, n_{1s}+n_{2s}>0} (n_{1s}d_{1s} + n_{2s}d_{2s}) \\
\text{subject to} \quad & w_0 + \sum_j b_{sj}w_j - d_{1s} \le 0 \quad \forall s : n_{1s} > 0\,, \\
& w_0 + \sum_j b_{sj}w_j + d_{2s} > 0 \quad \forall s : n_{2s} > 0\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& d_{1s}, d_{2s} \ge 0 \quad \forall s\,.
\end{aligned}$$

Binary attributes are usually found in medical diagnoses data. In this study three real data sets of disease discrimination were tested: developing postoperative pulmonary embolism or not, having dissecting aneurysm or other diseases, and suffering from posttraumatic epilepsy or not. In these data sets the MIP model for binary attributes (BMIP), which will be described later, performs better than other LP models or traditional methods.

Multigroup Classification Freed and Glover [32] extended the LP classification models from two-group to multigroup problems. One formulation which uses a single discriminant function is given below:

$$\begin{aligned}
\text{Minimize}\quad & \sum_{k=1}^{K-1} c_k\alpha_k \\
\text{subject to}\quad & \sum_j x_{ij}w_j \le U_k \quad \forall i \in G_k\ \forall k\,, \\
& \sum_j x_{ij}w_j \ge L_k \quad \forall i \in G_k\ \forall k\,, \\
& U_k + \epsilon \le L_{k+1} + \alpha_k \\
& \quad \forall k = 1, \ldots, K-1\,,\ w_j \text{ urs} \quad \forall j\,, \\
& U_k, L_k \text{ urs} \quad \forall k\,, \\
& \alpha_k \text{ urs} \quad \forall k = 1, \ldots, K-1\,,
\end{aligned}$$

where the number ϵ could be normalized to be 1, and c_k is the misclassification cost. However, single-function classification is not as flexible and general as multiple-function classification. Another extension from the two-group case to the multigroup case in [32] is to solve two-group LP models for all pairs of groups and determine classification rules based on these solutions. However, in some cases the group assignment is not clear and the resulting classification scheme may be suboptimal [107].

For the multigroup discrimination problem, Bennett and Mangasarian [10] defined the piecewise-linear separability of data from K groups as the following: The data from K groups are piecewise-linear-separable if and only if there exist $(w_0^k, w_1^k, \ldots, w_m^k) \in R^{m+1}$, $k = 1, \ldots, K$, such that $w_0^h + \sum_j x_{ij}w_j^h \ge w_0^k + \sum_j x_{ij}w_j^k + 1$, $\forall i \in G_h\ \forall h, k \ne h$. The following LP will generate a piecewise-linear separation for the K groups if one exists, otherwise it will generate an error-minimizing separation:

$$\begin{aligned}
\text{Minimize}\quad & \sum_h \sum_{k\ne h} \frac{1}{|G_h|} \sum_{i\in G_h} d_i^{hk} \\
\text{subject to}\quad & d_i^{hk} \ge -(w_0^h + \sum_j x_{ij}w_j^h) \\
& \qquad + (w_0^k + \sum_j x_{ij}w_j^k) + 1 \\
& \qquad \forall i \in G_h\ \forall h, k \ne h\,, \\
& w_j^k \text{ urs} \quad \forall j, k\,, \\
& d_i^{hk} \ge 0 \quad \forall i \in G_h\ \forall h,\ k \ne h\,.
\end{aligned}$$

The method was tested in three data sets. It performs pretty well in two of the data sets which are totally (or almost totally) piecewise-linear separable. The classification result is not good in the third data set, which is inherently more difficult. However, combining the multisurface method tree algorithm [8] results in an improvement in performance.

Gochet et al. [46] introduced an LP model for the general multigroup classification problem. The method separates the data with several hyperplanes by sequentially solving LPs. The vectors w^k, $k = 1, \ldots, K$, are estimated for the classification decision rule. The rule is to classify an observation i into group s, where $s = \arg\max_k\{w_0^k + \sum_j x_{ij}w_j^k\}$.

Suppose observation i is from group h. Denote the goodness of fit for observation i with respect to group k as

$$\begin{aligned}
& G_{hk}^i(w^h, w^k) \\
& \quad = \Big[\big(w_0^h + \sum_j x_{ij}w_j^h\big) - \big(w_0^k + \sum_j x_{ij}w_j^k\big)\Big]^+\,, \\
& \quad \text{where } [a]^+ = \max\{0, a\}\,.
\end{aligned}$$

Likewise, denote the badness of fit for observation i with respect to group k as

$$\begin{aligned}
& B_{hk}^i(w^h, w^k) \\
& \quad = \Big[\big(w_0^h + \sum_j x_{ij}w_h^h\big) - \big(w_0^k + \sum_j x_{ij}w_j^k\big)\Big]^-\,, \\
& \quad \text{where } [a]^- = -\min\{0, a\}\,.
\end{aligned}$$

The total goodness of fit and total badness of fit are then defined as

$$G(w) = G(w^1, \ldots, w^K) = \sum_h \sum_{k\ne h} \sum_{i\in G_h} G_{hk}^i(w^h, w^k)\,,$$

$$B(w) = B(w^1, \ldots, w^K) = \sum_h \sum_{k\ne h} \sum_{i\in G_h} B_{hk}^i(w^h, w^k)\,.$$

The LP is to minimize the total badness of fit, subject to a normalization equation, in which $q > 0$:

$$\begin{aligned}
&\text{Minimize} && B(w)\,,\\
&\text{subject to} && G(w) - B(w) = q\,,\\
& && w \text{ urs}\,.
\end{aligned}$$

Expanding $G(w)$ and $B(w)$ and substituting $G^i_{hk}(w^h, w^k)$ and $B^i_{hk}(w^h, w^k)$ by γ^i_{hk} and β^i_{hk} respectively, the LP becomes

$$\begin{aligned}
&\text{Minimize} && \sum_h \sum_{k \neq h} \sum_{i \in G_h} \beta^i_{hk}\\
&\text{subject to} && \Big(w^h_0 + \sum_j x_{ij} w^h_j\Big) - \Big(w^k_0 + \sum_j x_{ij} w^k_j\Big)\\
& && \quad = \gamma^i_{hk} - \beta^i_{hk} \quad \forall i \in G_h \ \forall h, k \neq h\,,\\
& && \sum_h \sum_{k \neq h} \sum_{i \in G_h} (\gamma^i_{hk} - \beta^i_{hk}) = q\,,\\
& && w^k_j \text{ urs} \quad \forall j, k\,,\\
& && \gamma^i_{hk}, \beta^i_{hk} \geq 0 \quad \forall i \in G_h \ \forall h, k \neq h\,.
\end{aligned}$$

The classification results for two real data sets show that this model can compete with Fisher's LDF and the nonparametric k-nearest-neighbor method.

The LP-based models for classification problems highlighted above are all nonparametric models. In Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology", we describe LP-based and MIP-based classification models that utilize a parametric multigroup discriminant analysis approach [39,40,60,63]. These latter models have been employed successfully in various multigroup disease diagnosis and biological/medical prediction problems [16,28,29,56,57,59,60,64,65].

Mixed Integer Programming Classification Models

While LP offers a polynomial-time computational guarantee, MIP allows more flexibility in (among other things) modeling misclassified observations and/or misclassification costs.

Two-Group Classification In the two-group classification problem, binary variables can be used in the formulation to track and minimize the exact number of misclassifications. Such an objective function is also considered as the L_0-norm criterion [107].

MM:

$$\begin{aligned}
&\text{Minimize} && \sum_i z_i\\
&\text{subject to} && w_0 + \sum_j x_{ij} w_j \leq M z_i \quad \forall i \in G_1\,,\\
& && w_0 + \sum_j x_{ij} w_j \geq -M z_i \quad \forall i \in G_2\,,\\
& && w_j \text{ urs} \quad \forall j\,,\\
& && z_i \in \{0,1\} \quad \forall i\,.
\end{aligned}$$

The vector w is required to be a nonzero vector to prevent the trivial solution.

In the MIP formulation the objective function could include the deviation terms, such as those in the hybrid models, as well as the number of misclassifications [5]; or it could represent expected cost of misclassification [1,6,101,105]. In particular, there are some variant versions of the basic model.

Stam and Joachimsthaler [109] studied the classification performance of MM and compared it with that of MSD, Fisher's LDF, and Smith's QDF. In some cases the MM model performs better, but in some cases it does not. MIP formulations are in the review studies of Joachimsthaler and Stam [50] and Erenguc and Koehler [27], and are contained in the software developed by Stam and Ungar [110]. Computational experiments show that the MIP model performs better when the group overlap is higher [50,109], although it is still not easy to reach general conclusions [107].

Since the MIP model is $\mathcal{NP}$-hard, exact algorithms and heuristics are proposed to solve it efficiently. Koehler and Erenguc [54] developed a procedure to solve MM in which the condition of nonzero w is replaced by the requirement of at least one violation of the constraints $w_0 + \sum_j x_{ij} w_j \leq 0$ for $i \in G_1$ or $w_0 + \sum_j x_{ij} w_j \geq 0$ for $i \in G_2$. Banks and Abad [6] solved the MIP of minimizing the expected cost of misclassification by an LP-based algorithm. Abad and Banks [1] developed three heuristic procedures for the problem of minimizing the expected cost of misclassification. They also included the interaction terms of the attributes in the data and applied the heuristics [7]. Duarte Silva and Stam [105] introduced the divide and conquer algorithm for the classification problem of minimizing the misclassification cost by solving MIP and LP subproblems. Rubin [101] solved the same problem by using a decomposition approach, and tested this procedure on some data sets, including two breast cancer data sets. Yanev and Balev [119] proposed exact and heuristic algorithms for solving MM, which are based on some specific properties of the vertices of a polyhedral set neatly connected with the model.

For the two-group classification problem where the attributes are binary, Asparoukhov and Stam [3] proposed LP and MIP models which partition the data into multinomial cells and result in fewer variables and constraints. Let s be the index of the cells, n_{1s}, n_{2s} be the number of data points in cell s from groups 1 and 2, respectively, and $(b_{s1}, \dots, b_{sm})$ be the binary digits representing cell s. Below is the BMIP, which performs best in the three real data sets in [3]:
BMIP

$$\begin{aligned}
\text{Minimize}\quad & \sum_{s:\, n_{1s}+n_{2s}>0} \{|n_{1s}-n_{2s}|z_s + \min(n_{1s}, n_{2s})\} \\
\text{subject to}\quad & w_0 + \sum_j b_{sj}w_j \le Mz_s \ \forall s: n_{1s} \ge n_{2s}; \\
& \qquad\qquad\qquad\qquad n_{1s} > 0\,, \\
& w_0 + \sum_j b_{sj}w_j > -Mz_s \ \forall s: n_{1s} < n_{2s}\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& z_s \in \{0,1\} \quad \forall s:\ n_{1s}+n_{2s} > 0\,.
\end{aligned}$$

Pavur et al. [96] included different secondary goals in model MM and compared their misclassification rates. A new secondary goal was proposed, which maximizes the difference between the means of the discriminant scores of the two groups. In this model the term $-\delta$ is added to the minimization objective function as a secondary goal with a constant multiplier, while the constraint $\sum_j \bar{x}_j^{(2)} w_j - \sum_j \bar{x}_j^{(1)} w_j \ge \delta$ is included, where $\bar{x}_j^{(k)} = 1/|G_k| \sum_{i\in G_k} x_{ij}\ \forall j$, for $k = 1, 2$. The results of simulation study show that an MIP model with the proposed secondary goal has better performance than the other models studied.

Glen [42] proposed integer programming (IP) techniques for normalization in the two-group discriminant analysis models. One technique is to add the constraint $\sum_{j=1}^m |w_j| = 1$. In the proposed model, w_j for $j = 1, \dots, m$ is represented by $w_j = w_j^+ - w_j^-$, where $w_j^+, w_j^- \ge 0$, and binary variables δ_j and γ_j are defined such that $\delta_j = 1 \Leftrightarrow w_j^+ \ge \epsilon$ and $\gamma_j = 1 \Leftrightarrow w_j^- \ge \epsilon$. The IP normalization technique is applied to MSD and MMD, and the MSD version is presented below.

MSD – with IP normalization:

$$\begin{aligned}
\text{Minimize}\quad & \sum_i d_i \\
\text{subject to}\quad & w_0 + \sum_{j=1}^m x_{ij}(w_j^+ - w_j^-) - d_i \le 0 \\
& \quad \forall i \in G_1\,, \\
& w_0 + \sum_{j=1}^m x_{ij}(w_j^+ - w_j^-) + d_i \ge 0 \\
& \quad \forall i \in G_2\,, \\
& \sum_{j=1}^m (w_j^+ + w_j^-) = 1\,, \\
& w_j^+ - \epsilon\delta_j \ge 0 \quad \forall j = 1, \dots, m\,, \\
& w_j^+ - \delta_j \le 0 \quad \forall j = 1, \dots, m\,, \\
& w_j^- - \epsilon\gamma_j \ge 0 \quad \forall j = 1, \dots, m\,, \\
& w_j^- - \gamma_j \le 0 \quad \forall j = 1, \dots, m\,, \\
& \delta_j + \gamma_j \le 1 \quad \forall j = 1, \dots, m\,, \\
& w_0 \text{ urs}\,, \\
& w_j^+, w_j^- \ge 0 \quad \forall j = 1, \dots, m\,, \\
& d_i \ge 0 \quad \forall i\,, \\
& \delta_j, \gamma_j \in \{0,1\} \quad \forall j = 1, \dots, m\,.
\end{aligned}$$

The variable coefficients of the discriminant function generated by the models are invariant under origin shifts. The proposed models were validated using two data sets from [45,87]. The models were also extended for attribute selection by adding the constraint $\sum_{j=1}^m (\delta_j + \gamma_j) = p$, which allows only a constant number, p, of attributes to be used for classification.

Glen [43] developed MIP models which determine the thresholds for forming dichotomous variables as well as the discriminant function coefficients, w_j. For each continuous attribute to be formed as a dichotomous attribute, the model finds the threshold among possible thresholds while determining the separating hyperplane and optimizing the objective function such as minimizing the sum of deviations or minimizing the number of misclassifications. Computational results of a real data set and some simulated data sets show that the MSD model with dichotomous categorical variable formation can improve classification performance. The reason for the potential of this technique is that the LDF generated is a nonlinear function of the original variables.

Multigroup Classification Gehrlein [41] proposed MIP formulations of minimizing the total number of misclassifications in the multigroup classification problem. He gave both a single-function classification scheme and a multiple-function classification scheme, as follows.

General single-function classification (GSFC) – minimizing the number of misclassifications:

$$\begin{aligned}
\text{Minimize}\quad & \sum_i z_i \\
\text{subject to}\quad & w_0 + \sum_j x_{ij} w_j - M z_i \le U_k \quad \forall i \in G_k\,, \\
& w_0 + \sum_j x_{ij} w_j + M z_i \ge L_k \quad \forall i \in G_k\,, \\
& U_k - L_k \ge \delta' \quad \forall k\,, \\
& \left.\begin{array}{l} L_g - U_k + M y_{gk} \ge \delta \\ L_k - U_g + M y_{kg} \ge \delta \\ y_{gk} + y_{kg} = 1 \end{array}\right\} \forall g, k,\ g \ne k\,, \\
& w_j \text{ urs} \quad \forall j\,, \\
& U_k, L_k \text{ urs} \quad \forall k\,, \\
& z_i \in \{0,1\} \quad \forall i\,, \\
& y_{gk} \in \{0,1\} \quad \forall g, k,\ g \ne k\,,
\end{aligned}$$

where U_k, L_k denote the upper and lower endpoints of the interval assigned to group k, and $y_{gk} = 1$ if the interval associated with group g precedes that with group k and $y_{gk} = 0$ otherwise. The constant δ' is the minimum width of an interval of a group and the constant δ is the minimum gap between adjacent intervals.

General multiple-function classification (GMFC) – minimizing the number of misclassifications:

$$\begin{aligned}
\text{Minimize}\quad & \sum_i z_i \\
\text{subject to}\quad & w_0^h + \sum_j x_{ij} w_j^h - w_0^k \\
& \quad - \sum_j x_{ij} w_j^k + M z_i \ge \epsilon \\
& \quad \forall i \in G_h,\ \forall h, k \ne h\,, \\
& w_j^k \text{ urs} \quad \forall j, k\,, \\
& z_i \in \{0,1\} \quad \forall i\,.
\end{aligned}$$

Both models work successfully on the iris data set provided by Fisher [30].

Pavur [93] solved the multigroup classification problem by sequentially solving the GSFC in one dimension each time. LDFs were generated by successively solving the GSFC with the added constraints that all linear discriminants are uncorrelated to each other for the total data set. This procedure could be repeated for the number of dimensions that is believed to be enough. According to the simulation results, this procedure substantially improves the GSFC model and sometimes outperforms GMFC, Fisher's LDF, or Smith's QDF.

To solve the three-group classification problem more efficiently, Loucopoulos and Pavur [71] made a slight modification to the GSFC and proposed the model MIP3G, which also minimizes the number of misclassifications. Compared with GSFC, MIP3G is also a single-function classification model, but it reduces the possible group orderings from six to three in the formulation and thus becomes more efficient. Loucopoulos and Pavur [72] reported the results of a simulation experiment on the performance of GMFC, MIG3G, Fisher's LDF, and Smith's QDF for a three-group classification problem with small training samples. Second-order terms were also considered in the experiment. Simulation results show that GMFC and MIP3G can outperform the parametric procedures in some nonnormal data sets and that the inclusion of second-order terms can improve the performance of MIP3G in some data sets. Pavur and Loucopoulos [95] investigated the effect of the gap size in the MIP3G model for the three-group classification problem. A simulation study illustrates that for fairly separable data, or data with small sample sizes, a non-zero-gap model can improve the performance. A possible reason for this result is that the zero-gap model may be overfitting the data.

Gallagher et al. [39,40,63] and Lee [59,60] proposed MIP models, both heuristic and exact, as a computational approach to solving the constrained discriminant method described by Anderson [2]. These models are described in detail in Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology".

Nonlinear Programming Classification Models

Nonlinear programming approaches are natural extensions for some of the LP-based models. Thus far, nonlinear programming approaches have been developed for two-group classification.

Stam and Joachimsthaler [108] proposed a class of nonlinear programming methods to solve the two-group classification problem under the L_p-norm objective criterion. This is an extension of MSD and MMD, for which the objectives are the L_1-norm and L_∞-norm, respectively.

Minimize the general L_p-norm distance:

$$\begin{aligned}
\text{Minimize} \quad & \Big(\sum_i d_i^p\Big)^{1/p} \\
\text{subject to} \quad & \sum_j x_{ij} w_j - d_i \le b \quad \forall i \in G_1 , \\
& \sum_j x_{ij} w_j + d_i \ge b \quad \forall i \in G_2 , \\
& w_j \ \text{urs} \quad \forall j , \\
& d_i \ge 0 \quad \forall i .
\end{aligned}$$

The simulation results show that, in addition to the L_1-norm and the L_∞-norm, it is worth the effort to compute other L_p-norm objectives. Restricting the analysis to $1 \le p \le 3$, plus $p = \infty$, is recommended. This method was reviewed by Joachimsthaler and Stam [50] and Erenguc and Koehler [27].

Mangasarian et al. [85] proposed a nonconvex model for the two-group classification problem:

$$\begin{aligned}
\text{Minimize} \quad & d^1 + d^2 \\
\text{subject to} \quad & \sum_j x_{ij} w_j - d^1 \le 0 \quad \forall i \in G_1 , \\
& \sum_j x_{ij} w_j + d^2 \ge 0 \quad \forall i \in G_2 , \\
& \max_{j=1,\dots,m} |w_j| = 1 , \\
& w_j \ \text{urs} \quad \forall j , \\
& d^1, d^2 \ \text{urs} ,
\end{aligned}$$

This model can be solved in polynomial-time by solving $2m$ linear programs, which generate a sequence of parallel planes, resulting in a piecewise-linear nonconvex discriminant function. The model works successfully in clinical practice for the diagnosis of breast cancer.

Further, Mangasarian [76] also formulated the problem of minimizing the number of misclassifications as a linear program with equilibrium constraints (LPEC) instead of the MIP model MM described previously:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i \in G_1 \cup G_2} z_i \\
\text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le -1 \quad \forall i \in G_1 , \\
& z_i \Big(w_0 + \sum_j x_{ij} w_j - d_i + 1\Big) = 0 \\
& \qquad \forall i \in G_1 , \\
& w_0 + \sum_j x_{ij} w_j + d_i \ge 1 \quad \forall i \in G_2 , \\
& z_i \Big(w_0 + \sum_j x_{ij} w_j + d_i - 1\Big) = 0 \\
& \qquad \forall i \in G_2 , \\
& d_i (1 - z_i) = 0 \quad \forall i \in G_1 \cup G_2 , \\
& 0 \le z_i \le 1 \quad \forall i \in G_1 \cup G_2 , \\
& d_i \ge 0 \quad \forall i \in G_1 \cup G_2 , \\
& w_j \ \text{urs} \quad \forall j .
\end{aligned}$$

The general LPEC can be converted to an exact penalty problem with a quadratic objective and linear constraints. A stepless Frank–Wolfe-type algorithm is proposed for the penalty problem, terminating at a stationary point or a global solution. This method is called the parametric misclassification minimization (PMM) procedure, and numerical testing is included in [77].

To illustrate the next model, we first define the step function $s\colon R \to \{0,1\}$ as

$$s(u) = \begin{cases} 1 & \text{if } u > 0\,, \\ 0 & \text{if } u \le 0\,. \end{cases}$$

The problem of minimizing the number of misclassifications is equivalent to

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in G_1 \cup G_2} s(d_i) \\ \text{subject to} \quad & w_0 + \sum_j x_{ij} w_j - d_i \le -1 \quad \forall i \in G_1\,, \\ & w_0 + \sum_j x_{ij} w_j + d_i \ge 1 \quad \forall i \in G_2\,, \\ & d_i \ge 0 \quad \forall i \in G_1 \cup G_2\,, \\ & w_j \text{ urs} \quad \forall j\,. \end{aligned}$$

Mangasarian [77] proposed a simple concave approximation of the step function for nonnegative variables: $t(u,\alpha) = 1 - e^{-\alpha u}$, where $\alpha > 0, u \ge 0$. Let $\alpha > 0$ and approximate $s(d_i)$ by $t(d_i,\alpha)$. The problem then reduces to minimizing a smooth concave function bounded below on a nonempty polyhedron, which has a minimum at a vertex of the feasible region. A finite successive linearization algorithm (SLA) was proposed, terminating at a stationary point or a global solution. Numerical tests of SLA were done and compared with the PMM procedure described above. The results show that the much simpler SLA obtains a separation that is almost as good as PMM in considerably less computing time.

Chen and Mangasarian [21] proposed an algorithm on a defined hybrid misclassification minimization problem, which is more computationally tractable than the $\mathcal{NP}$-hard misclassification minimization problem. The basic idea of the hybrid approach is to obtain iteratively w_0 and $(w_1, \ldots, w_m)$ of the separating hyperplane:

1. For a fixed w_0, solve RLP [9] to determine $(w_1, \ldots, w_m)$.
2. For this $(w_1, \ldots, w_m)$, solve the one-dimensional misclassification minimization problem to determine w_0.

Comparison of the hybrid method is made with the RLP method and the PMM procedure. The hybrid method performs better in the testing sets of the tenfold cross-validation and is much faster than PMM.

Mangasarian [78] proposed the model of minimizing the sum of arbitrary-norm distances of misclassified points to the separating hyperplane. For a general norm $||\cdot||$ on R^m, the dual norm $||\cdot||'$ on R^m is defined as $||x||' = \max_{||y||=1} x^T y$. Define $[a]^+ = \max\{0, a\}$ and let $w = (w_1, \ldots, w_m)$. The formulation can then be written as

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in G_1} \Big[w_0 + \sum_j x_{ij} w_j \Big]^+ \\ & \quad + \sum_{i \in G_2} \Big[-w_0 - \sum_j x_{ij} w_j \Big]^+ \\ \text{subject to} \quad & ||w||' = 1\,, \\ & w_0, w \text{ urs}\,. \end{aligned}$$

The problem is to minimize a convex function on a unit sphere. A decision problem related to this minimization problem is shown to be $\mathcal{NP}$-complete, except for $p = 1$. For a general p-norm, the minimization problem can be transformed via an exact penalty formulation to minimizing the sum of a convex function and a bilinear function on a convex set.

Support Vector Machine

A support vector machine (SVM) is a type of mathematical programming approach [112]. It has been widely studied, and has become popular in many application fields in recent years. The introductory description of SVMs given here is summarized from the tutorial by Burges [20]. In order to maintain consistency with SVM studies in published literature, the notation used below is slightly different from the notation used to describe the mathematical programming methods in earlier sections.

In the two-group separable case, the objective function is to maximize the margin of a separating hyperplane, $2/||w||$, which is equivalent to minimizing $||w||^2$:

$$\begin{aligned}
\text{Minimize} \quad & w^T w\,, \\
\text{subject to} \quad & x_i^T w + b \geq +1 \quad \text{for } y_i = +1\,, \\
& x_i^T w + b \leq -1 \quad \text{for } y_i = -1\,, \\
& w, b \text{ urs}\,,
\end{aligned}$$

where $x_i \in R^m$ represents the values of attributes of observation i and $y_i \in \{-1, 1\}$ represents the group of observation i.

This problem can be solved by solving its Wolfe dual problem:

$$\begin{aligned}
\text{Maximize} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j\,, \\
\text{subject to} \quad & \sum_i \alpha_i y_i = 0\,, \\
& \alpha_i \geq 0 \quad \forall i\,.
\end{aligned}$$

Here, α_i is the Lagrange multiplier for the training point i, and the points with $\alpha_i > 0$ are called the support vectors (analogous to the support of a hyperplane, and thus the introduction of the name "support vector"). The primal solution w is given by $w = \sum_i \alpha_i y_i x_i$. b can be computed by solving $y_i(w^T x_i + b) - 1 = 0$ for any i with $\alpha_i > 0$.

For the nonseparable case, slack variables ξ_i are introduced to handle the errors. Let C be the penalty for the errors. The problem becomes

$$\begin{aligned}
\text{Minimize} \quad & \frac{1}{2} w^T w + C\Big(\sum_i \xi_i\Big)^k\,, \\
\text{subject to} \quad & x_i^T w + b \geq +1 - \xi_i \quad \text{for } y_i = +1\,, \\
& x_i^T w + b \leq -1 + \xi_i \quad \text{for } y_i = -1\,, \\
& w, b \text{ urs}\,, \\
& \xi_i \geq 0 \quad \forall i\,.
\end{aligned}$$

When k is chosen to be 1, neither the ξ_i nor their Lagrange multipliers appear in the Wolfe dual problem:

$$\begin{aligned}
\text{Maximize} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j\,, \\
\text{subject to} \quad & \sum_i \alpha_i y_i = 0\,, \\
& 0 \leq \alpha_i \leq C \quad \forall i\,.
\end{aligned}$$

The data points can be separated nonlinearly by mapping the data into some higher-dimensional space and applying linear SVM to the mapped data. Instead of knowing explicitly the mapping Φ, SVM needs only the dot products of two transformed data points $\Phi(x_i) \cdot \Phi(x_j)$. The kernel function K is introduced such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. Replacing $x_i^T x_j$ by $K(x_i, x_j)$ in the above problem, the separation becomes nonlinear, while the problem to be solved remains a quadratic program. In testing a new data point x after training, the sign of the function $f(x)$ is computed to determine the group of x:

$$f(x) = \sum_{i=1}^{N_s} \alpha_i y_i \Phi(s_i) \cdot \Phi(x) + b = \sum_{i=1}^{N_s} \alpha_i y_i K(s_i, x) + b\,,$$

where the s_i are the support vectors and N_s is the number of support vectors. Again the explicit form of $\Phi(x)$ is avoided.

Mangasarian provided a general mathematical programming framework for SVM, called generalized SVM or GSVM [79,83]. Special cases can be derived from GSVM, including the standard SVM.

Many SVM-type methods have been developed by Mangasarian and others to solve huge classification problems more efficiently. These methods include successive overrelaxation for SVM [82], proximal SVM [36,38], smooth SVM [68], reduced SVM [67], Lagrangian SVM [84], incremental SVMs [37], and other methods [13,81]. Mangasarian [80] summarized some of the developments. Examples of applications of SVM include breast cancer studies [69,70] and genome research [73].

Hsu and Lin [49] compared different methods for multigroup classification using SVMs. Three methods studied were based on several binary classifiers: one against one, one against all, and directed acyclic graph (DAG) SVM. The other two methods studied are methods with decomposition implementation. The experimental results show that the one-against-one and DAG methods are more suitable for practical use than the other methods. Lee et al. [66] proposed a generic approach to multigroup problems with some theoretical properties, and the proposed method was well applied to microarray data for cancer classification and satellite radiance profiles for cloud classification.

Gallagher et al. [39,40,63] offered the first discrete SVM for multigroup classification with reserved judgement. The approach has been successfully applied to a diverse variety of biological and medical applications (see Sect. "Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology").

Mixed Integer Programming Based Multigroup Classification Models and Applications to Medicine and Biology

Commonly used methods for classification, such as LDFs, decision trees, mathematical programming approaches, SVMs, and artificial neural networks, can be viewed as attempts at approximating a *Bayes optimal rule* for classification; that is, a rule that maximizes (minimizes) the total probability of correct classification (misclassification). Even if a Bayes optimal rule is known, intergroup misclassification rates may be higher than desired. For example, in a population that is mostly healthy, a Bayes optimal rule for medical diagnosis might misdiagnose sick patients as healthy in order to maximize total probability of correct diagnosis. As a remedy, a constrained discriminant rule that limits the misclassification rate is appealing.

Assuming that the group density functions and prior probabilities are known, Anderson [2] showed that an optimal rule for the problem of maximizing the probability of correct classification subject to constraints on the misclassification probabilities must be of a specific form when discriminating among multiple groups with a simplified model. The formulae in Anderson's result depend on a set of parameters satisfying a complex relationship between the density functions, the prior probabilities, and the bounds on the misclassification probabilities. Establishing a viable mathematical model to describe Anderson's result, and finding values for these parameters that yield an optimal rule are challenging tasks. The first computational models utilizing Anderson's formulae were proposed in [39,40].

Discrete Support Vector Machine Predictive Models

As part of the work carried out at Georgia Institute of Technology's Center for Operations Research in Medicine, we have developed a general-purpose discriminant analysis modeling framework and computational engine that are applicable to a wide variety of applications, including biological, biomedical, and logistics problems. Utilizing the technology of large-scale discrete optimization and SVMs, we have developed novel classification models that simultaneously include the following features: (1) the ability to classify any number of distinct groups; (2) the ability to incorporate heterogeneous types of attributes as input; (3) a high-dimensional data transformation that eliminates noise and errors in biological data; (4) constraints to limit the rate of misclassification, and a reserved-judgment region that provides a safeguard against overtraining (which tends to lead to high misclassification rates from the resulting predictive rule); and (5) successive multistage classification capability to handle data points placed in the reserved-judgment region. Studies involving tumor volume identification, ultrasonic cell disruption in drug delivery, lung tumor cell motility analysis, CpG island aberrant methylation in human cancer, predicting early atherosclerosis using biomarkers, and fingerprinting native and angiogenic microvascular networks using functional perfusion data indicate that our approach is adaptable and can produce effective and reliable predictive rules for various biomedical and biobehavior phenomena [16,28,29,56,57,59,60,64,65].

Based on the description in [39,40,59,60,63], we summarize below some of the classification models we have developed.

Modeling of Reserved-Judgment Region for General Groups When the population densities and prior probabilities are known, the constrained rules with a reject option (reserved judgment), based on Anderson's results, call for finding a partition $\{R_0, \ldots, R_G\}$ of $\mathbb{R}^k$ that maximizes the probability of correct allocation subject to constraints on the misclassification probabilities; i. e.,

$$\text{Maximize} \quad \sum_{g=1}^{G} \pi_g \int_{R_g} f_g(w)\,dw \tag{1}$$

$$\text{subject to} \quad \int_{R_g} f_h(w)dw \le \alpha_{hg},\ h, g = 1, \ldots, G,\quad h \ne g\,, \tag{2}$$

where $f_h, h \in \{1, \ldots, G\}$, are the group conditional density functions, π_g denotes the prior probability that a randomly selected entity is from group $g, g \in \{1, \ldots, G\}$, and $\alpha_{hg}, h \ne g$, are constants between 0 and 1. Under quite general assumptions, it was shown that there exist unique (up to a set of measure zero) nonnegative constants $\lambda_{ih}, i, h \in \{1, \ldots, G\}, i \neq h$, such that the optimal rule is given by

$$R_g = \{x \in \mathbb{R}^k \colon L_g(x) = \max_{h \in \{0,1,\ldots,G\}} L_h(x)\} , \quad g = 0, \ldots, G , \tag{3}$$

where

$$L_0(x) = 0 , \tag{4}$$

$$L_h(x) = \pi_h f_h(x) - \sum_{i=1, i \neq h}^{G} \lambda_{ih} f_i(x), h = 1, \ldots, G . \tag{5}$$

For $G = 2$ the optimal solution can be modeled rather straightforwardly. However, finding optimal λ_{ih}'s for the general case, $G \geq 3$, is a difficult problem, with the difficulty increasing as G increases. Our model offers an avenue for modeling and finding the optimal solution in the general case. It is the first such model to be computationally viable [39,40].

Before proceeding, we note that R_g can be written as $R_g = \{x \in \mathbb{R}^k \colon L_g(x) \geq L_h(x)$ for all $h = 0, \ldots, G\}$. So, since $L_g(x) \geq L_h(x)$ if, and only if, $(1/\sum_{t=1}^{G} f_t(x)) L_g(x) \geq (1/\sum_{t=1}^{G} f_t(x)) L_h(x)$, the functions $L_h, h = 1, \ldots, G$, can be redefined as

$$L_h(x) = \pi_h p_h(x) - \sum_{i=1, i \neq h}^{G} \lambda_{ih} p_i(x), h = 1, \ldots, G , \tag{6}$$

where $p_i(x) = f_i(x)/\sum_{t=1}^{G} f_t(x)$. We assume that L_h is defined as in (6) in our model.

Mixed Integer Programming Formulations Assume that we are given a training sample of N entities whose group classifications are known; say, n_g entities are in group g, where $\sum_{g=1}^{G} n_g = N$. Let the k-dimensional vectors x^{gj}, $g = 1, \ldots, G, j = 1, \ldots, n_g$, contain the measurements on k available characteristics of the entities. Our procedure for deriving a discriminant rule proceeds in two stages. The first stage is to use the training sample to compute estimates, $\hat{f}_h$, either parametrically or nonparametrically, of the density functions f_h [89] and estimates, $\hat{\pi}_h$, of the prior probabilities $\pi_h, h = 1, \ldots, G$. The second stage is to determine the optimal λ_{ih}'s given these estimates. This stage requires being able to estimate the probabilities of correct classification and misclassification for any candidate set of λ_{ih}'s. One could, in theory, substitute the estimated densities and prior probabilities into (5), and directly use the resulting regions R_g in the integral expressions given in (1) and (2). This would involve, even in simple cases such as normally distributed groups, the numerical evaluation of k-dimensional integrals at each step of a search for the optimal λ_{ih}'s. Therefore, we have designed an alternative approach. After substituting the $\hat{f}_h$'s and $\hat{\pi}_h$'s into (5), we simply calculate the proportion of training sample points which fall in each of the regions $R_1, \ldots, R_G$. The MIP models discussed below attempt to maximize the proportion of training sample points correctly classified while satisfying constraints on the proportions of training sample points misclassified. This approach has two advantages. First, it avoids having to evaluate the potentially difficult integrals in (1) and (2). Second, it is nonparametric in controlling the training sample misclassification probabilities. That is, even if the densities are poorly estimated (by assuming, for example, normal densities for non-normal data), the constraints are still satisfied for the training sample. Better estimates of the densities may allow a higher correct classification rate to be achieved, but the constraints will be satisfied even if poor estimates are used. Unlike most SVM models that minimize the sum of errors, our objective is driven by the number of correct classifications, and will not be biased by the distance of the entities from the supporting hyperplane.

A word of caution is in order. In traditional unconstrained discriminant analysis, the true probability of correct classification of a given discriminant rule tends to be smaller than the rate of correct classification for the training sample from which it was derived. One would expect to observe such an effect for the method described herein as well. In addition, one would expect to observe an analogous effect with regard to constraints on misclassification probabilities – the true probabilities are likely to be greater than any limits imposed on the proportions of training sample misclassifications. Hence, the α_{hg} parameters should be carefully chosen for the application in hand.

Our first model is a nonlinear 0/1 MIP model with the nonlinearity appearing in the constraints. Model 1 maximizes the number of correct classifications of the given N training entities. Similarly, the constraints on the misclassification probabilities are modeled by ensuring that the number of group g training entities in region R_h is less than or equal to a prespecified percentage, $\alpha_{hg}(0 < \alpha_{hg} < 1)$, of the total number, n_g, of group g entities, $h, g \in \{1, \ldots, G\}, h \neq g$.

For notational convenience, let $\mathbf{G} = \{1, \ldots, G\}$ and $\mathbf{N}_g = \{1, \ldots, n_g\}$, for $g \in \mathbf{G}$. Also, analogous to the definition of p_i, define $\hat{p}_i$ by $\hat{p}_i = \hat{f}_i(x)/\sum_{t=1}^{G} \hat{f}_t(x)$. In our model, we use binary indicator variables to denote the group classification of entities. Mathematically, let u_{hgj} be a binary variable indicating whether or not x^{gj} lies in region R_h; i. e., whether or not the jth entity from group g is allocated to group h. Then model 1 can be written as follows.

Discriminant analysis MIP (DAMIP):

$$\text{Maximize} \quad \sum_{g \in G} \sum_{j \in N_g} u_{ggj}$$

$$\text{subject to} \quad L_{hgj} = \hat{\pi}_h \hat{p}_h(x^{gj}) - \sum_{i \in G \backslash h} \lambda_{ih} \hat{p}_i(x^{gj}), \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{7}$$

$$y_{gj} = \max\{0, L_{hgj} : h = 1, \ldots, G\}, \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{8}$$

$$y_{gj} - L_{ggj} \leq M(1 - u_{ggj}), \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{9}$$

$$y_{gj} - L_{hgj} \geq \varepsilon(1 - u_{hgj}), \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, h \neq g, \tag{10}$$

$$\sum_{j \in N_g} u_{hgj} \leq \lfloor \alpha_{hg} n_g \rfloor, \quad h, g \in \mathbf{G}, \; h \neq g, \tag{11}$$

$$-\infty < L_{hgj} < \infty, y_{gj} \geq 0, \lambda_{ih} \geq 0, u_{hgj} \in \{0, 1\}.$$

Constraint (7) defines the variable L_{hgj} as the value of the function L_h evaluated at x^{gj}. Therefore, the continuous variable y_{gj}, defined in constraint (8), represents $\max\{L_h(x^{gj}) : h = 0, \ldots, G\}$; and consequently, x^{gj} lies in region R_h if, and only if, $y_{gj} = L_{hgj}$. The binary variable u_{hgj} is used to indicate whether or not x^{gj} lies in region R_h; i. e., whether or not the jth entity from group g is allocated to group h. In particular, constraint (9), together with the objective, forces u_{ggj} to be 1 if, and only if, the jth entity from group g is correctly allocated to group g; and constraints (10) and (11) ensure that at most $\lfloor \alpha_{hg} n_g \rfloor$ (i. e., the greatest integer less than or equal to $\alpha_{hg} n_g$) group g entities are allocated to group $h, h \neq g$. One caveat regarding the indicator variables u_{hgj} is that although the condition $u_{hgj} = 0, h \neq g$, implies (by constraint (10)) that $x^{gj} \notin R_h$, the converse need not hold. As a consequence, the number of misclassifications may be overcounted. However, in our preliminary numerical study we found that the actual amount of overcounting is minimal. One could force the converse (thus, $u_{hgj} = 1$ if and only if $x^{gj} \in R_h$) by adding constraints $y_{gj} - L_{hgj} \leq M(1 - u_{hgj})$, for example. Finally, we note that the parameters M and ϵ are extraneous to the discriminant analysis problem itself, but are needed in the model to control the indicator variables u_{hgj}. The intention is for M and ϵ to be, respectively, large and small positive constants.

Model Variations We explore different variations in the model to grasp the quality of the solution and the associated computational effort.

A first variation involves transforming model 1 to an equivalent linear mixed integer model. In particular, model 2 replaces the N constraints defined in (8) with the following system of $3GN + 2N$ constraints:

$$y_{gj} \geq L_{hgj}, \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{12}$$

$$\tilde{y}_{hgj} - L_{hgj} \leq M(1 - v_{hgj}), \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{13}$$

$$\tilde{y}_{hgj} \leq \hat{\pi}_h \hat{p}_h(x^{gj}) v_{hgj}, \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{14}$$

$$\sum_{h \in G} v_{hgj} \leq 1, \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{15}$$

$$\sum_{h \in G} \tilde{y}_{hgj} = y_{gj}, \quad g \in \mathbf{G}, j \in \mathbf{N}_g, \tag{16}$$

where $\tilde{y}_{hgj} \geq 0$ and $v_{hgj} \in \{0, 1\}, h, g \in \mathbf{G}, j \in \mathbf{N}_g$. These constraints, together with the nonnegativity of y_{gj} force $y_{gj} = \max\{0, L_{hgj} : h = 1, \ldots, G\}$.

The second variation involves transforming model 1 to a heuristic linear MIP model. This is done by replacing the nonlinear constraint (8) with $y_{gj} \geq L_{hgj}, h, g \in \mathbf{G}, j \in \mathbf{N}_g$, and including penalty terms in the objective function. In particular, model 3 has the objective

$$\text{Maximize} \sum_{g\in G}\sum_{j\in N_g} \beta u_{ggj} - \sum_{g\in G}\sum_{j\in N_g} \gamma y_{gj} ,$$

where β and γ are positive constants. This model is heuristic in that there is nothing to force $y_{gj} = \max\{0, L_{hgj} : h = 1, \ldots, G\}$. However, since in addition to trying to force as many u_{ggj}'s to 1 as possible, the objective in model 3 also tries to make the y_{gj}'s as small as possible, and the optimizer tends to drive y_{gj} towards $\max\{0, L_{hgj} : h = 1, \ldots, G\}$. We remark that β and γ could be stratified by group (i. e., introduce possibly distinct $\beta_g, \gamma_g, g \in \mathbf{G}$) to model the relative importance of certain groups to be correctly classified.

A reasonable modification to models 1, 2, and 3 involves relaxing the constraints specified by (11). Rather than placing restrictions on the number of type g training entities classified into group h, for all $h, g \in \mathbf{G}, h \neq g$, one could simply place an upper bound on the *total* number of misclassified training entities. In this case, the $G(G-1)$ constraints specified by (11) would be replaced by the single constraint

$$\sum_{g\in G}\sum_{h\in G\setminus\{g\}}\sum_{j\in N_g} u_{hgj} \le \lfloor \alpha N \rfloor , \tag{17}$$

where α is a constant between 0 and 1. We will refer to models 1, 2, and 3 modified in this way as models 1T, 2T, and 3T, respectively. Of course, other modifications are also possible. For instance, one could place restrictions on the total number of type g points misclassified for each $g \in \mathbf{G}$. Thus, in place of the constraints specified in (17), one would include the constraints $\sum_{h\in G\setminus\{g\}} \sum_{j\in N_g} u_{hgj} \le \lfloor \alpha_g N \rfloor$, $g \in \mathbf{G}$, where $0 < \alpha_g < 1$.

We also explore a heuristic linear model of model 1. In particular, consider the linear program (DALP):

$$\text{Maximize} \quad \sum_{g\in G}\sum_{j\in N_g} (c_1 w_{gj} + c_2 y_{gj}) \tag{18}$$

$$\text{subject to} \quad L_{hgj} = \pi_h \hat{p}_h(x^{gj}) - \sum_{i\in G\setminus h} \lambda_{ih} \hat{p}_i(x^{gj}) , \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g , \tag{19}$$

$$L_{ggj} - L_{hgj} + w_{gj} \ge 0, \quad h, g \in \mathbf{G}, h \neq g, j \in \mathbf{N}_g, \tag{20}$$

$$L_{ggj} + w_{gj} \ge 0, \quad g \in \mathbf{G}, j \in \mathbf{N}_g , \tag{21}$$

$$-L_{hgj} + y_{gj} \ge 0, \quad h, g \in \mathbf{G}, j \in \mathbf{N}_g , \tag{22}$$

$$-\infty < L_{hgj} < \infty, w_{gj}, y_{gj}, \lambda_{ih} \ge 0 .$$

Constraint (19) defines the variable L_{hgj} as the value of the function L_h evaluated at x^{gj}. As the optimization solver searches through the set of feasible solutions, the λ_{ih} variables will vary, causing the L_{hgj} variables to assume different values. Constraints (20), (21), and (22) link the objective-function variables with the L_{hgj} variables in such a way that correct classification of training entities and allocation of training entities into the reserved-judgment region are captured by the objective-function variables. In particular, if the optimization solver drives w_{gj} to zero for some g,j pair, then constraints (20) and (21) imply that $L_{ggj} = \max\{0, L_{hgj} : h \in \mathbf{G}\}$. Hence, the jth entity from group g is correctly classified. If, on the other hand, the optimal solution yields $y_{gj} = 0$ for some g,j pair, then constraint (22) implies that $\max\{0, L_{hgj} : h \in \mathbf{G}\} = 0$. Thus, the jth entity from group g is placed in the reserved-judgment region. (Of course, it is possible for both w_{gj} and y_{gj} to be zero. One should decide prior to solving the linear program how to interpret the classification in such cases.) If both w_{gj} and y_{gj} are positive, the jth entity from group g is misclassified.

The optimal solution yields a set of λ_{ih}'s that best allocates the training entities (i. e., "best" in terms of minimizing the penalty objective function). The optimal λ_{ih}'s can then be used to define the functions L_h, $h \in G$, which in turn can be used to classify a new entity with feature vector $x \in \mathbb{R}^k$ by simply computing the index at which $\max\{L_h(x) : h \in \{0, 1, \ldots, G\}\}$ is achieved.

Note that model DALP places no a priori bound on the number of misclassified training entities. However, since the objective is to minimize a weighted combination of the variables w_{gj} and y_{gj}, the optimizer will attempt to drive these variables to zero. Thus, the optimizer is, in essence, attempting either to correctly classify training entities ($w_{gj} = 0$), or to place them in the reserved-judgment region ($y_{gj} = 0$). By varying the weights c_1 and c_2, one has a means of controlling the optimizer's emphasis for correctly classifying training entities versus placing them in the reserved-

Disease Diagnosis: Optimization-Based Methods, Table 1
Model size

Model	Type	Constraints	Total variables	0/1 Variables
1	Nonlinear MIP	$2GN + N + G(G-1)$	$2GN + N + G(G-1)$	GN
2	Linear MIP	$5GN + 2N + G(G-1)$	$4GN + N + G(G-1)$	$2GN$
3	Linear MIP	$3GN + G(G-1)$	$2GN + N + G(G-1)$	GN
1T	Nonlinear MIP	$2GN + N + 1$	$2GN + N + G(G-1)$	GN
2T	Linear MIP	$5GN + 2N + 1$	$4GN + N + G(G-1)$	$2GN$
3T	Linear MIP	$3GN + 1$	$2GN + N + G(G-1)$	GN
DALP	Linear program	$3GN$	$NG + N + G(G-1)$	0

judgment region. If $c_2/c_1 < 1$, the optimizer will tend to place a greater emphasis on driving the w_{gj} variables to zero than driving the y_{gj} variables to zero (conversely, if $c_2/c_1 > 1$). Hence, when $c_2/c_1 < 1$, one should expect to get relatively more entities correctly classified, fewer placed in the reserved-judgment region, and more misclassified, than when $c_2/c_1 > 1$. An extreme case is when $c_2 = 0$. In this case, there is no emphasis on driving y_{gj} to zero (the reserved-judgment region is thus ignored), and the full emphasis of the optimizer is to drive w_{gj} to zero.

Table 1 summarizes the number of constraints, the total number of variables, and the number of 0/1 variables in each of the discrete SVM models, and in the heuristic LP model (DALP). Clearly, even for moderately sized discriminant analysis problems, the MIP instances are relatively large. Also, note that model 2 is larger than model 3, in terms of both the number of constraints and the number of variables. However, it is important to keep in mind that the difficulty of solving an MIP problem cannot, in general, be predicted solely by its size; problem structure has a direct and substantial bearing on the effort required to find optimal solutions. The LP relaxation of these MIP models poses computational challenges as commercial LP solvers return (optimal) LP solutions that are infeasible, owing to the equality constraints, and the use of big M and small ϵ in the formulation.

It is interesting to note that the set of feasible solutions for model 2 is "tighter" than that for model 3. In particular, if F_i denotes the set of feasible solutions of model i, then

$$F_1 = \{(L, \lambda, u, y)\colon \text{ there exists } \tilde{y}, v \text{ such that } (L, \lambda, u, y, \tilde{y}, v) \in F_2\} \subsetneq F_3 \,. \quad (23)$$

The novelties of the classification models developed herein include the following: (1) they are suitable for discriminant analysis given any number of groups, (2) they accept heterogeneous types of attributes as input, (3) they use a parametric approach to reduce high-dimensional attribute spaces, and (4) they allow constraints on the number of misclassifications, and utilize a reserved judgment to facilitate the reduction of misclassifications. The lattermost point opens the possibility of performing multistage analysis.

Clearly, the advantage of an LP model over an MIP model is that the associated problem instances are computationally much easier to solve. However, the most important criterion in judging a method for obtaining discriminant rules is how the rules perform in correctly classifying new unseen entities. Once the rule has been developed, applying it to a new entity to determine its group is trivial. Extensive computational experiments have been performed to gauge the qualities of solutions of different models [17,19,40,59,60,63].

Validation of Model and Computational Effort We performed tenfold cross-validation, and designed simulation and comparison studies on our models. The results reported in [40,63] demonstrate that our approach works well when applied to both simulated data and data sets from the machine learning database repository [91]. In particular, our methods compare favorably and at times superior to other mathematical programming methods, including the GSFC model by Gehrlein [41], and the LP model by Gochet et al. [46], as well as Fisher's LDF, artificial neural networks, quadratic discriminant analysis, tree classification, and other SVMs, on real biological and medical data.

Classification Results for Real-World Biological and Medical Applications

The main objective in discriminant analysis is to derive rules that can be used to classify entities into groups. Computationally, the challenge lies in the effort expended to develop such a rule. Once the rule has been developed, applying it to a new entity to determine its group is trivial. Feasible solutions obtained from our classification models correspond to predictive rules. Empirical results [40,63] indicate that the resulting classification model instances are computationally very challenging, and even intractable by competitive commercial MIP solvers. However, the resulting predictive rules prove to be very promising, offering correct classification rates on new unknown data ranging from 80 to 100% for various types of biological/medical problems. Our results indicate that the general-purpose classification framework that we have designed has the potential to be a very powerful predictive method for clinical settings.

The choice of MIP as the underlying modeling and optimization technology for our SVM classification model is guided by the desire to simultaneously incorporate a variety of important and desirable properties of predictive models within a general framework. MIP itself allows for incorporation of continuous and discrete variables, and linear and nonlinear constraints, providing a flexible and powerful modeling environment.

Our mathematical modeling and computational algorithm design shows great promise as the resulting predictive rules are able to produce higher rates of correct classification for new biological data (with unknown group status) compared with existing classification methods. This is partly due to the transformation of raw data via the set of constraints in (7). While most mathematical programming approaches directly determine the hyperplanes of separation using raw data, our approach transforms the raw data via a probabilistic model, before the determination of the supporting hyperplanes. Further, the separation is driven by maximizing the sum of binary variables (representing correct classification or not of entities), instead of maximizing the margins between groups, or minimizing a sum of errors (representing distances of entities from hyperplanes), as in other SVMs. The combination of these two strategies offers better classification capability. Noise in the transformed data is not as profound as in raw data. And the magnitudes of the errors do not skew the determination of the separating hyperplanes, as all entities have *equal* importance when correct classification is being counted.

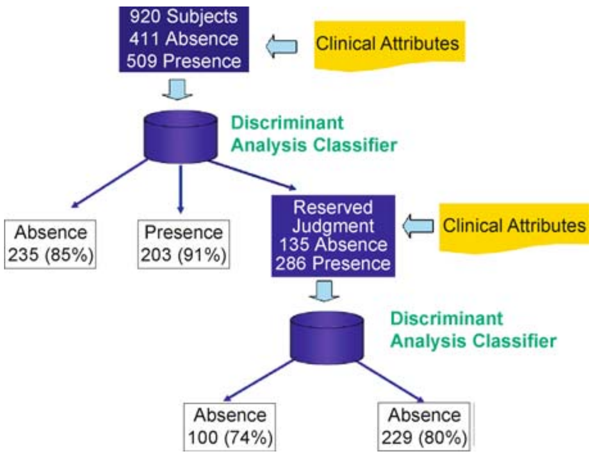
To highlight the broad applicability of our approach, below we briefly summarize the application of our predictive models and solution algorithms to ten different biological problems. Each of the projects was carried out in close partnership with experimental biologists and/or clinicians. Applications to finance and other industry applications are described elsewhere [17,40,63].

Determining the Type of Erythemato-Squamous Disease The differential diagnosis of erythemato-squamous diseases is an important problem in dermatology [60]. They all share the clinical features of erythema and scaling, with very little differences. The six groups are psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris. Usually a biopsy is necessary for the diagnosis but unfortunately these diseases share many histopathological features as well. Another difficulty for the differential diagnosis is that a disease may show the features of another disease at the beginning stage and may have the characteristic features at the following stages [91].

The six groups consisted of 366 subjects (112, 61, 72, 49, 52, and 20 respectively) with 34 clinical attributes. Patients were first evaluated clinically with 12 features. Afterwards, skin samples were taken for the evaluation of 22 histopathological features. The values of the histopathological features were determined by an analysis of the samples under a microscope. The 34 attributes include (1) clinical attributes (erythema, scaling, definite borders, itching, koebner phenomenon, polygonal papules, follicular papules, oral mucosal involvement, knee and elbow involvement, scalp involvement, family history, age) and (2) histopathological attributes (melanin incontinence, eosinophils in the infiltrate, polymorphonuclear leukocyte infiltrate, fibrosis of the papillary dermis, exocytosis, acanthosis, hyperkeratosis, parakeratosis, clubbing of the rete ridges, elongation of the rete ridges, thinning of the suprapapillary epidermis, spongiform pustule, Munro microabscess, focal hypergranulosis, disappearance of the granular layer, vacuolization and damage of basal layer, spongiosis, sawtooth appearance of retes, follicular horn plug, perifollicular parakeratosis, inflammatory monoluclear infiltrate, band-like infiltrate).

Our multigroup classification model selected 27 discriminatory attributes, and successfully classified the patients into six groups, each with an unbiased correct classification of greater than 93% (with 100% correct rate for groups 1, 3, 5, and 6) with an average overall accuracy of 98%. Using 250 subjects to develop the rule, and testing the remaining 116 patients, we obtained a prediction accuracy of 91%.

Predicting Presence/Absence of Heart Disease The four databases concerning heart disease diagnosis were collected by Dr. Andras Janosi of the Hungarian Institute of Cardiology, Budapest; Dr. William Steinbrunn of University Hospital, Zurich; Dr. Matthias Pfisterer of University Hospital, Basel; and Dr. Robert Detrano of V.A. Medical Center, Long Beach, and Cleveland Clinic Foundation [60]. Each database contains the same 76 attributes. The "goal" field refers to the presence of heart disease in the patient. The classification attempts to distinguish presence (values 1, 2, 3, 4, involving a total of 509 subjects) from absence (value 0, involving 411 subjects) [91]. The attributes include demographics, physiocardiovascular conditions, traditional risk factors, family history, personal lifestyle, and cardiovascular exercise measurements. This data set has posed some challenges to past analysis via various classification approaches, resulting in less than 80% correct classification. Applying our classification model without reserved judgment, we obtained 79 and 85% correct classification for each group respectively. To determine the usefulness of multistage analysis, we applied two-stage classification. In the first stage, 14 attributes were selected as discriminatory. One hundred and thirty-five group absence subjects were placed into the reserved-judgment region, with 85% of the remaining being classified as group absence correctly; while 286 group presence subjects were placed into the reserved-judgment region, and 91% of the remaining were classified correctly into the group presence. In the second stage, 11 attributes were selected with 100 and 229 classified into group absence and presence respectively. Combining the two stages, we obtained a correct classification of 82 and 85%, respectively, for diagnosis of absence or presence of heart disease. Figure 1 illustrates the two-stage classification.



Disease Diagnosis: Optimization-Based Methods, Figure 1
A tree diagram for two-stage classification and prediction of heart disease

Predicting Aberrant CpG Island Methylation in Human Cancer More details of this work can be found in [28,29]. Epigenetic silencing associated with aberrant methylation of promoter-region CpG islands is one mechanism leading to loss of tumor suppressor function in human cancer. Profiling of CpG island methylation indicates that some genes are more frequently methylated than others, and that each tumor type is associated with a unique set of methylated genes. However, little is known about why certain genes succumb to this aberrant event. To address this question, we used restriction landmark genome scanning (RLGS) to analyze the susceptibility of 1749 unselected CpG islands to de novo methylation driven by overexpression of DNMT1. We found that whereas the overall incidence of CpG island methylation was increased in cells overexpressing DNMT1, not all loci were equally affected. The majority of CpG islands (69.9%) were resistant to de novo methylation, regardless of DNMT1 overexpression. In contrast, we identified a subset of methylation-prone CpG islands (3.8%) that were consistently hypermethylated in multiple DNMT1 overexpressing clones. Methylation-prone and methylation-resistant CpG islands were not significantly different with respect to size, C+G content, CpG frequency, chromosomal location, or gene association or promoter association. To discriminate methylation-prone from methylation-resistant CpG islands, we developed a novel DNA pattern recognition model and algorithm [61], and coupled our predictive model described herein with the patterns found. We were able to derive a classification function based on the frequency of seven novel sequence patterns that was capable of discriminating methylation-prone from methylation-resistant CpG islands with 90% correctness upon cross-validation, and 85% accuracy when tested against blind CpG islands unknown to us regarding the methylation status. The data indicate that CpG islands differ in their intrinsic susceptibility to de novo methylation, and suggest that the propensity for a CpG island to become aberrantly methylated can be predicted on the basis of its sequence context.

The significance of this research is twofold. First, the identification of sequence patterns/attributes that distinguish methylation-prone CpG islands will lead to a better understanding of the basic mechanisms underlying aberrant CpG island methylation. Because genes that are silenced by methylation are otherwise structurally sound, the potential for reactivating these genes by blocking or reversing the methylation process represents an exciting new molecular target for chemotherapeutic intervention. A better understanding of the factors that contribute to aberrant methylation, including the identification of sequence elements that may act to target aberrant methylation, will be an important step in achieving this long-term goal. Secondly, the classification of the more than 29,000 known (but as yet unclassified) CpG islands in human chromosomes will provide an important resource for the identification of novel gene targets for further study as potential molecular markers that could impact on both cancer prevention and treatment. Extensive RLGS fingerprint information (and thus potential training sets of methylated CpG islands) already exists for a number of human tumor types, including breast, brain, lung, leukemias, hepatocellular carcinomas, and primitive neuroectodermal tumor [23,24,35,102]. Thus, the methods and tools developed are directly applicable to CpG island methylation data derived from human tumors. Moreover, new microarray-based techniques capable of "profiling" more than 7000 CpG islands have been developed and applied to human breast cancers [15,117,118]. We are uniquely poised to take advantage of the tumor CpG island methylation profile information that will likely be generated using these techniques over the next several years. Thus, our general-predictive modeling framework has the potential to lead to improved diagnosis and prognosis and treatment planning for cancer patients.

Discriminant Analysis of Cell Motility and Morphology Data in Human Lung Carcinoma Refer to [16] for more details of this work. This study focuses on the differential effects of extracellular matrix proteins on the motility and morphology of human lung epidermoid carcinoma cells. The behavior of carcinoma cells is contrasted with that of normal L-132 cells, resulting in a method for the prediction of metastatic potential. Data collected from time-lapsed videomicroscopy were used to simultaneously produce quantitative measures of motility and morphology. The data were subsequently analyzed using our discriminant analysis model and algorithm to discover relationships between motility, morphology, and substratum. Our discriminant analysis tools enabled the consideration of many more cell attributes than is customary in cell motility studies. The observations correlate with behaviors seen in vivo and suggest specific roles for the extracellular matrix proteins and their integrin receptors in metastasis. Cell translocation in vitro has been associated with malignancy, as has an elongated phenotype [120] and a rounded phenotype [97]. Our study suggests that extracellular matrix proteins contribute in different ways to the malignancy of cancer cells, and that multiple malignant phenotypes exist.

Ultrasound-Assisted Cell Disruption for Drug Delivery Reference [57] discusses this in detail. Although biological effects of ultrasound must be avoided for safe diagnostic applications, ultrasound's ability to disrupt cell membranes has attracted interest as a method to facilitate drug and gene delivery. This preliminary study seeks to develop rules for predicting the degree of cell membrane disruption based on specified ultrasound parameters and measured acoustic signals. Too much ultrasound destroys cells, while cell membranes will not open up for absorption of macromolecules when too little ultrasound is applied. The key is to increase cell permeability to allow absorption of macromolecules, and to apply ultrasound transiently to disrupt viable cells so as to enable exogenous material to enter without cell damage. Thus our task is to uncover a "predictive rule" of ultrasound-mediated disruption of red blood cells using acoustic spectrums and measurements of cell permeability recorded in experiments.

Our predictive model and solver for generating prediction rules were applied to data obtained from a sequence of experiments on bovine red blood cells. For each experiment, the attributes consisted of four ultrasound parameters, acoustic measurements at 400 frequencies, and a measure of cell membrane disruption. To avoid overtraining, various feature combinations of the 404 predictor variables were selected when developing the classification rule. The results indicate that the variable combination consisting of ultrasound exposure time and acoustic signals measured at the driving frequency and its higher harmonics yields the best rule, and our method compares favorably with classification tree and other ad hoc approaches, with a correct classification rate of 80% upon cross-validation and 85% when classifying new unknown entities. Our methods used for deriving the prediction rules are broadly applicable, and could be used to develop prediction rules in other scenarios involving different cell types or tissues. These rules and the methods used to derive them could be used for real-time feedback about ultrasound's biological effects. For example, it could assist clinicians during a drug delivery process, or could be imported into an implantable device inside the body for automatic drug delivery and monitoring.

Identification of Tumor Shape and Volume in Treatment of Sarcoma Reference [56] includes the detailed analysis. This project involves the determination of tumor shape for adjuvant brachytherapy treatment of sarcoma, based on catheter images taken after surgery. In this application, the entities are overlapping consecutive triplets of catheter markings, each of which is used for determining the shape of the tumor contour. The triplets are to be classified into one of two groups: group 1 (triplets for which the middle catheter marking should be bypassed) and group 2 (triplets for which the middle marking should not be bypassed). To develop and validate a classification rule, we used clinical data collected from 15 soft-tissue sarcoma patients. Cumulatively, this comprised 620 triplets of catheter markings. By careful (and tedious) clinical analysis of the geometry of these triplets, 65 were determined to belong to group 1, the "bypass" group, and 555 were determined to belong to group 2, the "do-not-bypass" group.

A set of measurements associated with each triplet was then determined. The choice of what attributes to measure to best distinguish triplets as belonging to group 1 or group 2 is nontrivial. The attributes involved the distance between each pair of markings, angles, and the curvature formed by the three triplet markings. On the basis of the attributes selected, our predictive model was used to develop a classification rule. The resulting rule provides 98% correct classification on cross-validation, and was capable of correctly determining/predicting 95% of the shape of the tumor with new patients' data. We remark that the current clinical procedure requires manual outline based on markers in films of the tumor volume. This study was the first to use automatic construction of tumor shape for sarcoma adjuvant brachytherapy [56,62].

Discriminant Analysis of Biomarkers for Prediction of Early Atherosclerosis More detail on this work can be found in [65]. Oxidative stress is an important etiologic factor in the pathogenesis of vascular disease. Oxidative stress results from an imbalance between injurious oxidant and protective antioxidant events, of which the former predominate [88,103]. This results in the modification of proteins and DNA, alteration in gene expression, promotion of inflammation, and deterioration in endothelial function in the vessel wall, all processes that ultimately trigger or exacerbate the atherosclerotic process [22,111]. It was hypothesized that novel biomarkers of oxidative stress would predict early atherosclerosis in a relatively healthy nonsmoking population free from cardiovascular disease. One hundred and twenty-seven healthy nonsmokers, without known clinical atherosclerosis had carotid intima media thickness (IMT) measured using ultrasound. Plasma oxidative stress was estimated by measuring plasma lipid hydroperoxides using the determination of reactive oxygen metabolites (d-ROMs) test. Clinical measurements include traditional risk factors, including age, sex, low-density lipoprotein (LDL), high-density lipoprotein (HDL), triglycerides, cholesterol, body-mass index (BMI), hypertension, diabetes mellitus, smoking history, family history of coronary artery disease, Framingham risk score, and high-sensitivity C-reactive protein.

For this prediction, the patients were first clustered into two groups: (group 1, IMT $\geq$ 0.68; group 2, IMT < 0.68). On the basis of this separator, 30 patients belonged to group 1, and 97 belonged to group 2. Through each iteration, the classification method trains and learns from the input training set and returns the most discriminatory patterns among the 14 clinical measurements; ultimately resulting in the development of a prediction rule based on observed values of these discriminatory patterns among the patient data. Using all 127 patients as a training set, the predictive model identified age, sex, BMI, HDL cholesterol, family history of coronary artery disease under 60, high-sensitivity C-reactive protein, and d-ROM as discriminatory attributes that together provide unbiased correct classification of 90 and 93%, respectively, for group 1 (IMT $\geq$ 0.68) and group 2 (IMT < 0.68) patients. To further test the power of the classification method for correctly predicting the IMT status of new/unseen patients, we randomly selected a smaller patient training set of size 90. The predictive rule from this training set yielded 80 and 89% correct rates for predicting the remaining 37 patients as group 1 and group 2 patients, respectively. The importance of d-ROM as a discriminatory predictor for IMT status was confirmed during the machine learning process. This biomarker was selected in every iteration as the "machine" learned and was trained to develop a predictive rule to correctly classify patients in the training set. We also performed predictive analysis using Framingham risk score and d-ROM; in this case the unbiased correct classification rates (for the 127 individuals) for groups 1 and 2 were 77 and 84%, respectively. This is the first study to illustrate that this measure of oxidative stress can be effectively used along with traditional risk factors to generate a predictive rule that can potentially serve as an inexpensive clinical diagnostic tool for prediction of early atherosclerosis.

Fingerprinting Native and Angiogenic Microvascular Networks Through Pattern Recognition and Discriminant Analysis of Functional Perfusion Data The analysis and findings are described in [64]. The cardiovascular system provides oxygen and nutrients to the entire body. Pathological conditions that impair normal microvascular perfusion can result in tissue ischemia, with potentially serious clinical effects. Conversely, development of new vascular structures fuels the progression of cancer, macular degeneration, and atherosclerosis. Fluorescence microangiography offers superb imaging of the functional perfusion of new and existent microvasculature, but quantitative analysis of the complex capillary patterns is challenging. We developed an automated pattern-recognition algorithm to systematically analyze the microvascular networks, and then applied our classification model described herein to generate a predictive rule. The pattern-recognition algorithm identifies the complex vascular branching patterns, and the predictive rule demonstrates, respectively, 100 and 91% correct classification for perturbed (diseased) and normal tissue perfusion. We confirmed that transplantation of normal bone marrow to mice in which genetic deficiency resulted in impaired angiogenesis eliminated predicted differences and restored normal-tissue perfusion patterns (with 100% correctness). The pattern-recognition and classification method offers an elegant solution for the automated fingerprinting of microvascular networks that could contribute to better understanding of angiogenic mechanisms and be utilized to diagnose and monitor microvascular deficiencies. Such information would be valuable for early detection and monitoring of functional abnormalities before they produce obvious and lasting effects, which may include improper perfusion of tissue, or support of tumor development.

The algorithm can be used to discriminate between the angiogenic response in a native healthy specimen compared with groups with impairment due to age or chemical or other genetic deficiency. Similarly, it can be applied to analyze angiogenic responses as a result of various treatments. This will serve two important goals. First, the identification of discriminatory patterns/attributes that distinguish angiogenesis status will lead to a better understanding of the basic mechanisms underlying this process. Because therapeutic control of angiogenesis could influence physiological and pathological processes such as wound and tissue repairing, cancer progression and metastasis, or macular degeneration, the ability to understand it under different conditions will offer new insight into developing novel therapeutic interventions, monitoring and treatment, especially in aging, and heart disease. Thus, our study and the results form the foundation of a valuable diagnostic tool for changes in the functionality of the microvasculature and for discovery of drugs that alter the angiogenic response. The methods can be applied to tumor diagnosis, monitoring, and prognosis. In particular, it will be possible to derive microangiographic fingerprints to acquire specific microvascular patterns associated with early stages of tumor development. Such "angioprinting" could become an extremely helpful early diagnostic modality, especially for easily accessible tumors such as skin cancer.

Prediction of Protein Localization Sites The protein localization database consists of eight groups with a total of 336 instances (143, 77, 52, 35, 20, 5, 2, and 2, respectively) with seven attributes [91]. The eight groups are eight localization sites of protein, including cytoplasm (cp), inner membrane without signal sequence (im), perisplasm (pp), inner membrane, uncleavable signal sequence (imU), outer membrane (om), outer membrane lipoprotein (omL), inner membrane lipoprotein (imL), inner membrane, and cleavable signal sequence (imS). However, the last four groups were taken out of our classification experiment since the population sizes are too small to ensure significance.

The seven attributes include McGeoch's method for signal sequence recognition (mcg), von Heijne's method for signal sequence recognition (gvh), von Heijne's signal peptidase II consensus sequence score (lip), presence of charge on N-terminus of predicted lipoproteins (chg), score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins (aac), score of the ALOM membrane spanning region prediction program (alm1), and score of the ALOM program after excluding putative cleavable signal regions from the sequence (alm2).

In the classification we use four groups, 307 instances, with seven attributes. Our classification model selected the discriminatory patterns mcg, gvh, alm1, and alm2 to form the predictive rule with unbiased correct classification rates of 89%, compared with 81% by other classification models [48].

Pattern Recognition in Satellite Images for Determining Types of Soil The satellite database consists of the multispectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multispectral values. In the sample database, the class of a pixel is coded as a number. There are six groups with 4435 samples in the training data set and 2000 samples in the testing data set; and each sample entity has 36 attributes describing the spectral bands of the image [91].

The original Landsat Multi-Spectral Scanner (MSS) image data for this database were generated from data purchased from NASA by the Australian Centre for Remote Sensing. The Landsat satellite data are one of the many sources of information available for a scene. The interpretation of a scene by integrating spatial data of diverse types and resolutions including multispectral and radar data, maps indicating topography, land use, etc. is expected to assume significant importance with the onset of an era characterized by integrative approaches to remote sensing (for example, NASA's Earth Observing System commencing this decade).

One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infrared. Each pixel is an 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80 m $\times$ 80 m. Each image contains 2340×3380 such pixels.

The database is a (tiny) subarea of a scene, consisting of 82×100 pixels. Each line of data corresponds to a 3×3 square neighborhood of pixels completely contained within the 82×100 subarea. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the nine pixels in the 3×3 neighborhood and a number indicating the classification label of the central pixel. The number is a code for the following six groups: red soil, cotton crop, gray soil, damp gray soil, soil with vegetation stubble, and very damp gray soil. Running our classification model, we selected 17 discriminatory attributes to form the classification rule, producing an unbiased prediction with 85% accuracy.

Further Advances

Brooks and Lee [17,18] devised other variations of the basic DAMIP model. They also showed that

DAMIP is strongly universally consistent (in some sense) with very good rates of convergence from Vapnik and Chervonenkis theory. A polynomial-time algorithm for discriminating between two populations with the DAMIP model was developed, and DAMIP was shown to be $\mathcal{NP}$-complete for a general number of groups. The proof demonstrating $\mathcal{NP}$-completeness employs results used in generating edges of the conflict graph [4,11,12,55]. Exploiting the necessary and sufficient conditions that identify edges in the conflict graph is the central contribution to the improvement in solution performance over industry-standard software. The conflict graph is the basis for various valid inequalities, a branching scheme, and for conditions under which integer variables are fixed for all solutions. Additional solution methods are identified which include a heuristic for finding solutions at nodes in the branch-and-bound tree, upper bounds for model parameters, and necessary conditions for edges in the conflict hypergraph [26,58]. Further, we have concluded that DAMIP is a computationally feasible, consistent, stable, robust, and accurate classifier.

Progress and Challenges

We summarize in Table 2 the mathematical programming techniques used in classification problems as reviewed in this chapter.

As noted by current research efforts, multigroup classification remains $\mathcal{NP}$-complete and much work is needed to design effective models as well as to derive novel and efficient computational algorithms to solve these multigroup instances.

Other Methods

While most classification methods can be described in terms of discriminant functions, some methods are not trained in the paradigm of determining coefficients or parameters for functions of a predefined form. These methods include *classification and regression trees*, *nearest-neighbor* methods, and *neural networks*.

Classification and regression trees [14] are nonparametric approaches to prediction. Classification trees seek to develop classification rules based on successive binary partitions of observations based on attribute values. Regression trees also employ rules consisting of binary partitions, but are used to predict continuous responses.

The rules generated by classification trees are easily viewable by plotting them in a treelike structure, from which the name arises. A test entity may be classified using rules in a tree plot by first comparing the entity's data with the root node of the tree. If the root node condition is satisfied by the data for a particular entity, the left branch is followed to another node; otherwise, the right branch is followed to another node. The data from the observation are compared with conditions at subsequent nodes until a leaf node is reached.

Nearest-neighbor methods begin by establishing a set of labeled prototype observations. The nearest-neighbor classification rule assigns test entities to groups according to the group membership of the nearest prototype. Different measures of distance may be used. The k-nearest-neighbor rule assigns entities to groups according to the group membership of the k nearest prototypes.

Neural networks are classification models that can also be interpreted in terms of discriminant functions, though they are used in a way that does not require finding an analytic form for the functions [25]. Neural networks are trained by considering one observation at a time, modifying the classification procedure slightly with each iteration.

Summary and Conclusion

In this chapter, we presented an overview of mathematical programming based classification models, and analyzed their development and advances in recent years. Many mathematical programming methods are geared toward two-group analysis only, and their performance is often compared with Fisher's LDF or Smith's QDF. It has been noted that these methods can be used for multiple group analysis by finding $G(G-1)/2$ discriminants for each pair of groups ("one against one") or by finding G discriminants for each group versus the remaining data ("one against all"), but these approaches can lead to ambiguous classification rules [25].

Mathematical programming methods developed for multiple group analysis have been described [10,32,39,40,41,46,59,60,63,93]. Multiple group formulations for SVMs have been proposed and tested [17,36,40,49,59,60,66], but are still considered computationally in-

Disease Diagnosis: Optimization-Based Methods, Table 2
Progress in mathematical programming-based classification models

Mathematical programming methods	References
Linear programming	
Two-group classification	
Separate data by hyperplanes	[74,75]
Minimizing the sum of deviations, minimizing the maximum deviation, and minimizing the sum of interior distances	[5,31,32,33,47,99]
Hybrid model	[45,99]
Review	[27,50,107]
Software	[110]
Issues about normalization	[34,44,51,52,53,87, 100,114,115,116]
Robust linear programming	[9,86]
Inclusion of second-order terms	[104,113]
Effect of the position of outliers	[94]
Binary attributes	[3]
Multigroup classification	
Single function classification	[32]
Multiple function classification	[10,46]
Classification with reserved-judgment region using linear programming	[39,40,60,63]
Mixed integer programming	
Two-group classification	
Minimizing the number of misclassifications	[1,5,6,7,54,101,105, 109,119]
Review	[27,50,107]
Software	[110]
Secondary goals	[96]
Binary attributes	[3]
Normalization and attribute selection	[42]
Dichotomous categorical variable formation	[43]
Multigroup classification	
Multigroup classification	[41,93]
Three-group classification	[71,72,95]
Classification with reserved-judgment region using mixed integer programming	[17,39,40,59,60]
Nonlinear programming	
Two-group classification	
L_p-norm criterion	[108]
Review	[27,50,107]
Piecewise-linear nonconvex discriminant function	[85]
Minimizing the number of misclassifications	[21,76,77]
Minimizing the sum of arbitrary-norm distances	[78]
Support vector machine	
Introduction and tutorial	[20,112]
Generalized support vector machine	[79,83]
Methods for huge-size problems	[13,36,37,38,67,68, 80,81,82,84]
Multigroup support vector machine	[17,38,39,40,49,59, 60,63,66]

tensive [49]. The "one-against-one" and "one-against-all" methods with SVMs have been successfully applied [49,90].

We also discussed a class of multigroup general-purpose predictive models that we have developed based on the technology of large-scale optimization and SVMs [17,19,39,40,59,60,63]. Our models seek to maximize the correct classification rate while constraining the number of misclassifications in each group. The models incorporate the following features: (1) the ability to classify any number of distinct groups; (2) allow incorporation of heterogeneous types of attributes as input; (3) a high-dimensional data transformation that eliminates noise and errors in biological data; (4) constrain the misclassification in each group and a reserved-judgment region that provides a safeguard against overtraining (which tends to lead to high misclassification rates from the resulting predictive rule); and (5) successive multistage classification capability to handle data points placed in the reserved-judgment region. The performance and predictive power of the classification models is validated through a broad class of biological and medical applications.

Classification models are critical to medical advances as they can be used in genomic, cell, molecular, and system-level analyses to assist in early prediction, diagnosis and detection of disease, as well as for intervention and monitoring. As shown in the CpG island study for human cancer, such prediction and diagnosis opens up novel therapeutic sites for early intervention. The ultrasound application illustrates its application to a novel drug delivery mechanism, assisting clinicians during a drug delivery process, or in devising devices that can be implanted into the body for automated drug delivery and monitoring. The lung cancer cell motility study offers an understanding of how cancer cells behave in different protein media, thus assisting in the identification of potential gene therapy and target treatment. Prediction of the shape of a cancer tumor bed provides a personalized treatment design, replacing manual estimates by sophisticated computer predictive models. Prediction of early atherosclerosis through inexpensive biomarker measurements and traditional risk factors can serve as a potential clinical diagnostic tool for routine physical and health maintenance, alerting physicians and patients to the need for early intervention to prevent serious vascular disease. Fingerprinting of microvascular networks opens up the possibility for early diagnosis of perturbed systems in the body that may trigger disease (e. g., genetic deficiency, diabetes, aging, obesity, macular degeneracy, tumor formation), identification of target sites for treatment, and monitoring prognosis and success of treatment. Determining the type of erythemato-squamous disease and the presence/absence of heart disease helps clinicians to correctly diagnose and effectively treat patients. Thus, classification models serve as a basis for predictive medicine where the desire is to diagnose early and provide personalized target intervention. This has the potential to reduce healthcare costs, improve success of treatment, and improve quality of life of patients.

References

1. Abad PL, Banks WJ (1993) New LP based heuristics for the classification problem. Eur J Oper Res 67:88–100
2. Anderson JA (1969) Constrained discrimination between *k* populations. J Roy Statist Soc Ser B (Methodological) 31(1):123–139
3. Asparoukhov OK, Stam A (1997) Mathematical programming formulations for two-group classification with binary variables. Ann Oper Res 74:89–112
4. Atamturk A (1998) Conflict graphs and flow models for mixed-integer linear optimization problems. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta
5. Bajgier SM, Hill AV (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. Decis Sci 13:604–618
6. Banks WJ, Abad PL (1991) An efficient optimal solution algorithm for the classification problem. Decis Sci 22:1008–1023
7. Banks WJ, Abad PL (1994) On the performance of linear programming heuristics applied on a quadratic transformation in the classification problem. Eur J Oper Res 74:23–28
8. Bennett KP (1992) Decision tree construction via linear programming. In: Evans M (ed) Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference, pp 97–101
9. Bennett KP, Mangasarian OL (1992) Robust linear programming discrimination of two linearly inseparable sets. Optim Methods Softw 1:23–34
10. Bennett KP, Mangasarian OL (1994) Multicategory discrimination via linear programming. Optim Methods Softw 3:27–39
11. Bixby RE, Lee EK (1998) Solving a truck dispatching scheduling problem using branch-and-cut. Oper Res 46:355–367
12. Borndörfer R (1997) Aspects of set packing, partitioning and covering. PhD thesis, Technischen Universität Berlin, Berlin
13. Bradley PS, Mangasarian OL (2000) Massive data discrimination via linear support vector machines. Optim Methods Softw 13(1):1–10
14. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and Regression Trees. Wadsworth and Brooks/Cole Advanced Books and Software, Pacific Grove
15. Brock GJ, Huang TH, Chen CM, Johnson KJ (2001) A novel technique for the identification of CpG islands exhibiting altered methylation patterns (ICEAMP). Nucleic Acids Res 29:e123
16. Brooks JP, Wright A, Zhu C, Lee EK (2007) Discriminant analysis of motility and morphology data from human lung carcinoma cells placed on purified extracellular matrix proteins. Ann Biomed Eng, Submitted
17. Brooks JP, Lee EK (2006) Solving a mixed-integer programming formulation of a multi-category constrained discrimination model. In: Proceedings of the 2006 INFORMS Workshop on Artificial Intelligence and Data Mining, Pittsburgh
18. Brooks JP, Lee EK (2007) Analysis of the consistency of a mixed integer programming-based multi-category constrained discriminant model. Submitted
19. Brooks JP, Lee EK (2007) Mixed integer programming constrained discrimination model for credit screening. In: Proceedings of the 2007 Spring Simulation Multiconference, Business and Industry Symposium, pp 1–6, Norfolk, VA, March ACM Digital Library

20. Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Mining Knowl Discov 2: 121–167
21. Chen C, Mangasarian OL (1996) Hybrid misclassification minimization. Adv Comput Math 5:127–136
22. Chevion M, Berenshtein E, Stadtman ER (2000) Human studies related to protein oxidation: protein carbonyl content as a marker of damage. Free Radical Res 33(Suppl):S99–S108
23. Costello JF, Fruhwald MC, Smiraglia DJ, Rush LJ, Robertson GP, Gao X, Wright FA, Feramisco JD, Peltomaki P, Lang JC, Schuller DE, Yu L, Bloomfield CD, Caligiuri MA, Yates A, Nishikawa R, Su HH, Petrelli NJ, Zhang X, O'Dorisio MS, Held WA, Cavenee WK, Plass C (2000) Aberrant CpG-island methylation has non-random and tumour-type-specific patterns. Nat Genet 24:132–138
24. Costello JF, Plass C, Cavenee WK (2000) Aberrant methylation of genes in low-grade astrocytomas. Brain Tumor Pathol 17:49–56
25. Duda RO, Hart PE, Stork DG (2001) Pattern Classification. Wiley, New York
26. Easton T, Hooker K, Lee EK (2003) Facets of the independent set plytope. Math Program Ser B 98:177–199
27. Erenguc SS, Koehler GJ (1990) Survey of mathematical programming models and experimental results for linear discriminant analysis. Managerial Decis Econ 11:215–225
28. Feltus FA, Lee EK, Costello JF, Plass C, Vertino PM (2003) Predicting aberrant CpG island methylation. Proc Natl Acad Sci USA 100:12253–12258
29. Feltus FA, Lee EK, Costello JF, Plass C, Vertino PM (2006) DNA signatures associated with CpG island methylation states. Genomics 87:572–579
30. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugenics 7:179–188
31. Freed N, Glover F (1981) A linear programming approach to the discriminant problem. Decis Sci 12:68–74
32. Freed N, Glover F (1981) Simple but powerful goal programming models for discriminant problems. Eur J Oper Res 7:44–60
33. Freed N, Glover F (1986) Evaluating alternative linear programming models to solve the two-group discriminant problem. Decis Sci 17:151–162
34. Freed N, Glover F (1986) Resolving certain difficulties and improving the classification power of LP discriminant analysis formulations. Decis Sci 17:589–595
35. Fruhwald MC, O'Dorisio MS, Rush LJ, Reiter JL, Smiraglia DJ, Wenger G, Costello JF, White PS, Krahe R, Brodeur GM, Plass C (2000) Gene amplification in NETs/medulloblastomas: mapping of a novel amplified gene within the MYCN amplicon. J Med Genet 37:501–509
36. Fung GM, Mangasarian OL (2001) Proximal support vector machine classifiers. In Proceedings KDD-2001, San Francisco
37. Fung GM, Mangasarian OL (2002) Incremental support vector machine classification. In: Grossman R, Mannila H, Motwani R (eds) Proceedings of the Second SIAM International Conference on Data Mining. SIAM, Philadelphia, pp 247–260
38. Fung GM, Mangasarian OL (2005) Multicategory proximal support vector machine classifiers. Mach Learn 59:77–97
39. Gallagher RJ, Lee EK, Patterson DA (1996) An optimization model for constrained discriminant analysis and numerical experiments with iris, thyroid, and heart disease datasets. In: Proceedings of the 1996 American Medical Informatics Association
40. Gallagher RJ, Lee EK, Patterson DA (1997) Constrained discriminant analysis via 0/1 mixed integer programming. Ann Oper Res 74:65–88
41. Gehrlein WV (1986) General mathematical programming formulations for the statistical classification problem. Oper Res Lett 5(6):299–304
42. Glen JJ (1999) Integer programming methods for normalisation and variable selection in mathematical programming discriminant analysis models. J Oper Res Soc 50:1043–1053
43. Glen JJ (2004) Dichotomous categorical variable formation in mathematical programming discriminant analysis models. Naval Res Logist 51:575–596
44. Glover F (1990) Improved linear programming models for discriminant analysis. Decis Sci 21:771–785
45. Glover F, Keene S, Duea B (1988) A new class of models for the discriminant problem. Decis Sci 19:269–280
46. Gochet W, Stam A, Srinivasan V, Chen S (1997) Multigroup discriminant analysis using linear programming. Oper Res 45(2):213–225
47. Hand DJ (1981) Discrimination and classification. Wiley, New York
48. Horton P, Nakai K (1996) A probablistic classification system for predicting the cellular localization sites of proteins. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, St. Louis, USA, pp 109–115
49. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Networks 13(2):415–425
50. Joachimsthaler EA, Stam A (1990) Mathematical programming approaches for the classification problem in two-group discriminant analysis. Multivariate Behavior Res 25(4):427–454
51. Koehler GJ (1989) Characterization of unacceptable solutions in LP discriminant analysis. Decis Sci 20:239–257
52. Koehler GJ (1989) Unacceptable solutions and the hybrid discriminant model. Decis Sci 20:844–848
53. Koehler GJ (1994) A response to Xiao's "necessary and sufficient conditions of unacceptable solutions in LP discriminant analysls": Something is amiss. Decis Sci 25: 331–333
54. Koehler GJ, Erenguc SS (1990) Minimizing misclassifications in linear discriminant analysis. Decis Sci 21: 63–85

55. Lee EK (1993) Solving a truck dispatching scheduling problem using branch-and-cut. PhD thesis, Computational and Applied Mathematics, Rice University, Houston
56. Lee EK, Fung AYC, Brooks JP, Zaider M (2002) Automated planning volume definition in soft-tissue sarcoma adjuvant brachytherapy. Biol Phys Med 47:1891–1910
57. Lee EK, Gallagher RJ, Campbell AM, Prausnitz MR (2004) Prediction of ultrasound-mediated disruption of cell membranes using machine learning techniques and statistial analysis of acoustic spectra. IEEE Trans Biomed Eng 51:1–9
58. Lee EK, Maheshwary S (2006) Conflict hypergraphs in integer programming. Technical report, Georgia Institute of Technology, submitted
59. Lee EK (2007) Optimization-based predictive models in medicine and biology. Optimization in Medicine. Springer Netherlands. Springer Series in Optimization and Its Application 12:127–151
60. Lee EK (2007) Large-scale optimization-based classification models in medicine and biology. Ann Biomed Eng Syst Biol Bioinformat 35(6):1095–1109
61. Lee EK, Easton T, Gupta K (2006) Novel evolutionary models and applications to sequence alignment problems. Ann Oper Res Oper Res Medic – Comput Optim Medic Life Sci 148:167–187
62. Lee EK, Fung AYC, Zaider M (2001) Automated planning volume contouring in soft-tissue sarcoma adjuvant brachytherapy treatment. Int J Radiat Oncol Biol Phys 51:391
63. Lee EK, Gallagher RJ, Patterson DA (2003) A linear programming approach to discriminant analysis with a reserved-judgment region. INFORMS J Comput 15(1):23–41
64. Lee EK, Jagannathan S, Johnson C, Galis ZS (2006) Fingerprinting native and angiogenic microvascular networks through pattern recognition and discriminant analysis of functional perfusion data. submitted
65. Lee EK, Wu TL, Ashfaq S, Jones DP, Rhodes SD, Weintrau WS, Hopper CH, Vaccarino V, Harrison DG, Quyyumi AA (2007) Prediction of early atherosclerosis in healthy adults via novel markers of oxidative stress and d-ROMs. Working paper
66. Lee Y, Lin Y, Wahba G (2004) Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. J Am Stat Assoc 99:67–81
67. Lee YJ, Mangasarian OL (2001) RSVM: Reduced support vector machines. In: Proceedings of the SIAM International Conference on Data Mining, Chicago, April 5–7
68. Lee YJ, Mangasarian OL (2001) SSVM: A smooth support vector machine for classification. Comput Optim Appl 20(1):5–22
69. Lee YJ, Mangasarian OL, Wolberg WH (2000) Breast cancer survival and chemotherapy: A support vector machine analysis. In: DIMACS Series in Discrete Mathematical and Theoretical Computer Science, vol 55. American Mathematical Society, Providence, pp 1–10
70. Lee YJ, Mangasarian OL, Wolberg WH (2003) Survival-time classification of breast cancer patients. Comput Optim Appl 25:151–166
71. Loucopoulos C, Pavur R (1997) Computational characteristics of a new mathematical programming model for the three-group discriminant problem. Comput Oper Res 24(2):179–191
72. Loucopoulos C, Pavur R (1997) Experimental evaluation of the classificatory performance of mathematical programming approaches to the three-group discriminant problem: The case of small samples. Ann Oper Res 74:191–209
73. Luedi PP, Hartemink AJ, Jirtle RL (2005) Genome-wide prediction of imprinted murine genes. Genome Res 15:875–884
74. Mangasarian OL (1965) Linear and nonlinear separation of patterns by linear programming. Oper Res 13:444–452
75. Mangasarian OL (1968) Multi-surface method of pattern separation. IEEE Trans Inform Theory 14(6):801–807
76. Mangasarian OL (1994) Misclassification minimization. J Global Optim 5:309–323
77. Mangasarian OL (1996) Machine learning via polyhedral concave minimization. In: Fischer H, Riedmueller B, Schaeffler S (eds) Applied Mathematics and Parallel computing – Festschrift for Klaus Ritter. Physica-Verlag, Heidelberg, pp 175–188
78. Mangasarian OL (1999) Arbitrary-norm separating plane. Oper Res Lett 24:15–23
79. Mangasarian OL (2000) Generalized support vector machines. In: Smola AJ, Bartlett P, Schökopf B, Schuurmans D (eds) Advances in Large Margin Classifiers. MIT Press, Cambridge, pp 135–146
80. Mangasarian OL (2003) Data mining via support vector machines. In: Sachs EW, Tichatschke R (eds) System Modeling and Optimization XX. Kluwer, Boston, pp 91–112
81. Mangasarian OL (2005) Support vector machine classification via parameterless robust linear programming. Optim Methods Softw 20:115–125
82. Mangasarian OL, Musicant DR (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Networks 10:1032–1037
83. Mangasarian OL, Musicant DR (2001) Data discrimination via nonlinear generalized support vector machines. In: Ferris MC, Mangasarian OL, Pang JS (eds) Complementarity: Applications, Algorithms and Extensions. Kluwer, Boston, pp 233–251
84. Mangasarian OL, Musicant DR (2001) Lagrangian support vector machines. J Mach Learn Res 1:161–177
85. Mangasarian OL, Setiono R, Wolberg WH (1990) Pattern recognition via linear programming: Theory and application to medical diagnosis. In: Coleman TF, Li Y (eds) Large-Scale Numerical Optimization. SIAM, Philadelphia, pp 22–31

86. Mangasarian OL, Street WN, Wolberg WH (1995) Breast cancer diagnosis and prognosis via linear programming. Oper Res 43(4):570–577
87. Markowski EP, Markowski CA (1985) Some difficulties and improvements in applying linear programming formulations to the discriminant problem. Decis Sci 16:237–247
88. McCord JM (2000) The evolution of free radicals and oxidative stress. Am J Med 108:652–659
89. McLachlan GJ (1992) Discriminant analysis and statistical pattern recognition. Wiley, New York
90. Müller KR, Mika S, Rätsch G, Tsuda K, Schölkopf B (2001) An introduction to kernel-based learning algorithms. IEEE Trans Neural Networks 12(2):181–201
91. Murphy PM, Aha DW (1994) UCI Repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html. Department of Information and Computer Science, University of California, Irvine
92. O'Hagan A (1994) Kendall's Advanced Theory of Statistics: Bayesian Inference, vol 2B. Halsted Press, New York
93. Pavur R (1997) Dimensionality representation of linear discriminant function space for the multiple-group problem: An MIP approach. Ann Oper Res 74:37–50
94. Pavur R (2002) A comparative study of the effect of the position of outliers on classical and nontraditional approaches to the two-group classification problem. Eur J Oper Res 136:603–615
95. Pavur R, Loucopoulos C (2001) Evaluating the effect of gap size in a single function mathematical programming model for the three-group classification problem. J Oper Res Soc 52:896–904
96. Pavur R, Wanarat P, Loucopoulos C (1997) Examination of the classificatory performance of MIP models with secondary goals for the two-group discriminant problem. Ann Oper Res 74:173–189
97. Raz A, Ben-Zéev A (1987) Cell contact and architecture of malignant cells and their relationship to metastasis. Cancer Metastasis Rev 6:3–21
98. Rencher AC (1998) Multivariate Statistical Inference and Application. Wiley, New York
99. Rubin PA (1990) A comparison of linear programming and parametric approaches to the two-group discriminant problem. Decis Sci 21:373–386
100. Rubin PA (1991) Separation failure in linear programming discriminant models. Decis Sci 22:519–535
101. Rubin PA (1997) Solving mixed integer classification problems by decomposition. Ann Oper Res 74:51–64
102. Rush LJ, Dai Z, Smiraglia DJ, Gao X, Wright FA, Fruhwald M, Costello JF, Held WA, Yu L, Krahe R, Kolitz JE, Bloomfield CD, Caligiuri MA, Plass C (2001) Novel methylation targets in de novo acute myeloid leukemia with prevalence of chromosome 11 loci. Blood 97:3226–3233
103. Sies H (1985) Oxidative stress: introductory comments. In: Sies H (ed) Oxidative Stress. Academic Press, London, pp 1–8
104. Duarte Silva AP, Stam A (1994) Second order mathematical programming formulations for discriminant analysis. Eur J Oper Res 72:4–22
105. Duarte Silva AP, Stam A (1997) A mixed integer programming algorithm for minimizing the training sample misclassification cost in two-group classification. Ann Oper Res 74:129–157
106. Smith CAB (1947) Some examples of discrimination. Ann Eugenics 13:272–282
107. Stam A (1997) Nontraditional approaches to statistical classification: Some perspectives on l_p-norm methods. Ann Oper Res 74:1–36
108. Stam A, Joachimsthaler EA (1989) Solving the classification problem in discriminant analysis via linear and nonlinear programming methods. Decis Sci 20:285–293
109. Stam A, Joachimsthaler EA (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. Eur J Oper Res 46:113–122
110. Stam A, Ungar DR (1995) RAGNU: A microcomputer package for two-group mathematical programming-based nonparametric classification. Eur J Oper Res 86:374–388
111. Tahara S, Matsuo M, Kaneko T (2001) Age-related changes in oxidative damage to lipids and DNA in rat skin. Mechan Ageing Develop 122:415–426
112. Vapnik V (1995) The Nature of Statistical Learning Theory. Springer, New York
113. Wanarat P, Pavur R (1996) Examining the effect of second-order terms in mathematical programming approaches to the classification problem. Eur J Oper Res 93: 582–601
114. Xiao B (1993) Necessary and sufficient conditions of unacceptable solutions in LP discriminant analysis. Decis Sci 24:699–712
115. Xiao B (1994) Decision power and solutions of LP discriminant models: Rejoinder. Decis Sci 25:335–336
116. Xiao B, Feng Y (1997) Alternative discriminant vectors in LP models and a regularization method. Ann Oper Res 74:113–127
117. Yan PS, Chen CM, Shi H, Rahmatpanah F, Wei SH, Caldwell CW, Huang TH (2001) Dissecting complex epigenetic alterations in breast cancer using CpG island microarrays. Cancer Res 61:8375–8380
118. Yan PS, Perry MR, Laux DE, Asare AL, Caldwell CW, Huang TH (2000) CpG island arrays: an application toward deciphering epigenetic signatures of breast cancer. Clin Cancer Res 6:1432–1438
119. Yanev N, Balev S (1999) A combinatorial approach to the classification problem. Eur J Oper Res 115:339–350
120. Zimmermann A, Keller HU (1987) Locomotion of tumor cells as an element of invasion and metastasis. Biomed Pharmacotherapy 41:337–344
121. Zopounidis C, Doumpos M (2002) Multicriteria classification and sorting methods: A literature review. Eur J Oper Res 138:229–246

Disjunctive Programming

DP

HANIF D. SHERALI
Virginia Polytechnic Institute and State University,
Blacksburg, USA

MSC2000: 90C09, 90C10, 90C11

Article Outline

Keywords

Disjunctive programming; Polyhedral annexation; Facial disjunctive program; Cutting planes; Nondominated cuts; Facet; Valid inequalities; Reformulation-linearization technique; Lift-and-project; Tight relaxations; Model reformulation; Convex hull; Mixed integer 0–1 programs; Polynomial programs; Nonconvex programs

Disjunctive programming (DP) problems can be stated in the form

$$\text{(DP)} \qquad \text{Minimize}\,\{f(x)\colon\; x \in X, x \in \cup_{h\in H} S_h\}\,,$$

where $f\colon \mathbf{R}^n \to \mathbf{R}$ is a lower semicontinuous function, X is a closed convex subset of the nonnegative orthant of $\mathbf{R}^n$, and H is an index set for the collection of nonempty polyhedra

$$S_h = \left\{x\colon\; A^h x \geq b^h, x \geq 0\right\}, \qquad h \in H. \tag{1}$$

The name for this class of problems arises from the feature that the constraints in (1) include the *disjunction* that at least one of the (linear) sets of constraints defining S_h, for $h \in H$, must be satisfied. Problems including other logical conditions such as conjunctions, negations, and implications can be cast in the framework of this problem. Problem (DP) subsumes the classes of 0–1 *mixed integer problems*, the generalized lattice point problem, the cardinality constrained linear program, the extreme point optimization problem, the linear complementarity problem, among numerous others, and finds application in several related problems such as orthogonal production scheduling, scheduling on identical machines, multistage assignment, location-allocation problems, load balancing problems, the segregated storage problem, the fixed-charge problem, project/portfolio selection problems, goal programming problems, and many other game theory and decision theory problems (see [35] for a detailed discussion of such problems and applications).

The theory and algorithms for disjunctive programming problems are mainly supported by the fundamental *disjunctive cut principle*. The forward part of this result due to E. Balas [4,5] states that for any nonnegative surrogate multiplier vectors λ^h, $h \in H$, the inequality

$$\sup_{h\in H}\{\lambda^h A^h\} x \geq \inf_{h\in H}\{\lambda^h b^h\} \tag{2}$$

is valid for (or is implied by) the disjunction $x \in \cup_{h\in H} S_h$, where the $\sup\{\cdot\}$ and $\inf\{\cdot\}$ are taken componentwise in (2). More importantly, the converse part of this result due to R.G. Jeroslow [16] states that for any given valid inequality $\pi x \geq \pi_0$ for the disjunction $x \in \cup_{h\in H} S_h$, there exist nonnegative surrogate multipliers λ^h, $h \in H$, such that the disjunctive cut (2) implies this given valid inequality, or uniformly dominates it, over the nonnegative orthant. This disjunctive cut principle also arises from the setting of convexity cuts and polyhedral annexation methods as propounded by F. Glover [11,12], and it subsumes as well as can improve upon many types of classical cutting planes such as Gomory's mixed integer cuts, intersection cuts, and reverse outer polar cuts for 0–1 programs (see [4,5,11,12,35]). H.P. Williams [39] provides some additional insights into disjunctive formulations.

The generation of particular types of 'deep cuts' to delete a given solution (say, the origin, without loss of generality) based on the criteria of maximizing the Euclidean distance or the rectilinear distance between the origin and the nonnegative region feasible to the cutting plane, or maximizing the surplus in the cut with respect to the origin subject to suitable normalization constraints has also been explored in [34,37]. The intent behind such cutting plane methods is to generate *nondominated valid inequalities* that are supports (and hopefully, facets) of the closure convex hull of solutions feasible to the disjunction. H.D. Sherali and C.M. Shetty [35,37] discuss how different alternate formulations of the disjunctive statement can influence the strength of the cut derived therefrom, and demonstrate how a sequence of augmented formulations can be used to sequentially tighten a given valid inequality. This process turns out to be precisely the Glover polyhedral annexation scheme in [12]. In contrast with this sequence dependent *'lifting' procedure*, Sherali and Shetty [37] propose a 'simultaneous lifting' variant of this approach. Other types of disjunctive cutting planes for special problems include the cuts of [4,5,10,11,12,20], and [32] for linear knapsack, multiple choice and combinatorial disjunctions, [29] for linear complementarity problems, and the facet cuts of [25] based on the convex hull of certain types of disjunctions.

Balas [3] also provides an algebraic characterization for the closure *convex hull* of a union of polyhedra. This characterization is particularly useful in the study of the important class of facial disjunctive programs, that subsumes mixed integer 0–1 problems and linear complementarity problems, for example. A *facial disjunctive program* (FDP) can be stated as follows.

$$\text{(FDP)} \qquad \text{Minimize}\, \{cx\colon\ x \in X \cap Y\}\,,$$

where X is a nonempty polytope in $\mathbf{R}^n$, and where Y is a conjunction of some $\widehat{h}$ disjunctions given in the so-called *conjunctive normal form* (conjunction of disjunctions)

$$Y = \cap_{h\in H}\left[\cup_{i\in Q_h}\left\{x\colon\ a_i^h x \geq b_i^h\right\}\right]. \tag{3}$$

Here, $H = \{1,\dots,\widehat{h}\}$ and for each $h \in H$ we have specified a disjunction that requires at least one of the inequalities $a_i^h x \geq b_i^h$, for $i \in Q_h$, to be satisfied. The terminology 'facial' conveys the feature that $X \cap \{x\colon a_i^h x \geq b_i^h\}$ defines a face of X for each $i \in Q_h$, $h \in H$. For example, in the context of 0–1 mixed integer problems, the set X represents the linear programming relaxation of the problem, and for each binary variable x_h, $h \in H$, the corresponding disjunction in (3) states that $x_h \leq 0$ or $x_h \geq 1$ should hold true (where $0 \leq x_h \leq 1$ is included within X). Balas [3] shows that for facial disjunctive programs, the convex hull of feasible solutions can be constructed inductively by starting with $K_0 = X$ and then determining

$$K_h = \text{conv}\left[\cup_{i\in Q_h}\left(K_{h-1}\cap\left\{x\colon\ a_i^h x \geq b_i^h\right\}\right)\right] \quad \text{for} \quad h = 1,\dots,\widehat{h}\,, \tag{4}$$

where $K_{\widehat{h}}$ produces $\text{conv}(X \cap Y)$. Based on this, a hierarchy of relaxations $K_0,\dots,K_{\widehat{h}}$ is generated for (FDP) that spans the spectrum from the linear programming to the convex hull representation [6]. Each member in this hierarchy can also be viewed as being obtained by representing the feasible region of the original problem as the intersection of the union of certain polyhedra, and then taking a hull-relaxation of this representation. Here, for a set $D = \cap_j D_j$, where each D_j is the union of certain polyhedra, the hull-relaxation of D [3] is defined as $h - \text{rel}(D) = \cap_j \text{conv}(D_j) \supseteq \text{conv}(D)$.

In the context of 0–1 mixed integer problems (MIP), Sherali and W.P. Adams [27,28] develop a *reformulation-linearization technique* (RLT) for generating a hierarchy of such relaxations, introducing the notion of multiplying constraints using factors composed of x_h and $(1 - x_h)$, $h \in H$, to reformulate the problem, followed by a variable substitution to linearize the resulting problem. Approaches based on such constraint product and linearization strategies were used by these authors earlier in the context of several special applications [1,2,26]. Later, L. Lovász and A. Schrijver [17] independently used more general constraint factors to generate a similar hierarchy for 0–1 problems. The foregoing RLT construct can be specialized to derive K_h defined by (4) for 0–1 MIPs, where in this case,

$$K_h \equiv \text{conv}\left[(K_{h-1}\cap\{x\colon\ x_h \leq 0\}) \cup (K_{h-1}\cap\{x\colon\ x_h \geq 1\})\right]$$

can be obtained by multiplying the (implicitly defined) constraints of K_{h-1} by x_h and $(1 - x_h)$ and then linearizing the resulting problem. This RLT approach is used in [8] in the *'lift-and-project' hierarchy* of relaxations. However, the RLT process of [27,28] generates tighter relaxations at each level which can be viewed as hull relaxations produced by the intersection of the convex hull of the union of certain specially constructed polyhedra. No direct realization of (4) can produce these relaxations. For a survey on RLT approaches and for further enhancements, see [29,30].

In the context of general facial disjunctive programs, Jeroslow [15] presented a cutting plane algorithm that generates suitable facetial inequalities at each stage of the procedure such that an overall finite convergence is guaranteed via (4). This is accomplished by showing that in the worst case, the hierarchy $K_0, \ldots, K_{\widehat{h}}$ would be generated. The lift-and-project algorithm of [8] employs this cutting plane procedure based on the foregoing hierarchy of relaxations. Balas [7] also addresses an enhanced procedure that considers two variables at a time to define the disjunctions. The RLT process is used to construct partial convex hulls, and the resulting relaxations are embedded in a branch and cut algorithm.

Furthermore, for general facial disjunctive programs, Sherali and Shetty [36] present another finitely convergent cutting plane algorithm. At each step, this procedure searches for *extreme faces* of X relative to the cuts generated thus far (these are faces that do not contain any feasible points lying in a lower-dimensional face of X, see [18]), and based on the dimension of this extreme face and its feasibility to Y, either a disjunctive face cut or a disjunctive intersection cut is generated. This procedure was specialized for *bilinear programming* problems in [33] to derive a first nonenumerative finitely convergent algorithm for this class of problems.

Other disjunctive cutting plane algorithms include the Sherali–Sen procedures [31] for solving the general class of *extreme point mathematical programs*, the Baptiste–LePape procedures [9], and the Pinto–Grossmann procedures [21] for solving certain scheduling problems having disjunctive logic constraints. S. Sen and Sherali [24] also discuss issues related to designing convergent cutting plane algorithms, and present examples to show nonconvergence of certain iterative disjunctive cutting plane methods. Sensitivity and stability issues related to feasible and optimal sets of disjunctive programs have been addressed in [14]; [13] deals with the problem of solving algebraic systems of disjunctive equations. For other applications of disjunctive methods to process systems engineering, and to logic programming, see [19,23,38].

See also

- ► MINLP: Branch and Bound Global Optimization Algorithm
- ► MINLP: Branch and Bound Methods
- ► MINLP: Global Optimization with αBB
- ► MINLP: Logic-Based Methods
- ► Reformulation-linearization Methods for Global Optimization

References

1. Adams WP, Sherali HD (1986) A tight linearization and an algorithm for zero-one quadratic programming problems. Managem Sci 32(10):1274–1290
2. Adams WP, Sherali HD (1990) Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 38(2):217–226
3. Balas E (1974) Disjunctive programming: Properties of the convex hull of feasible points. Managem Sci Res Report GSIA Carnegie-Mellon Univ 348, no. July
4. Balas E (1974) Intersection cuts from disjunctive constraints. Managem Sci Res Report Carnegie-Mellon Univ 330, no. Feb
5. Balas E (1975) Disjunctive programming: Cutting planes from logical conditions. In: Mangasarian OL, Meyer RR, Robinson SM (eds) Nonlinear Programming. Acad. Press, New York
6. Balas E (1985) Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J Alg Discrete Meth 6:466–485
7. Balas E (1997) A modified lift-and-project procedure. Math Program 79(1–3):19–32
8. Balas E, Ceria S, Cornuejols G (1993) A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math Program 58:295–324
9. Baptiste P, Lepape C (1996) Disjunctive constraints for manufacturing scheduling: Principles and extensions. Internat J Comput Integrated Manufacturing 9(4):306–310
10. Glover F (1973) Convexity cuts and cut search. Oper Res 21:123–134
11. Glover F (1974) Polyhedral convexity cuts and negative edge extensions. Z Oper Res 18:181–186
12. Glover F (1975) Polyhedral annexation in mixed integer and combinatorial programming. Math Program 8:161–188 (See also MSRS Report 73-9, Univ. Colorado, August, 1973).
13. Grossmann IE, Turkay M (1996) Solution of algebraic systems of disjunctive equations. Comput Chem Eng 20, Suppl.:S339–S344
14. Helbig S (1994) Stability in disjunctive optimization II. Continuity of the feasible and optimal set. Optim 31(1):63–93
15. Jeroslow RG (1977) A cutting plane game and its algorithms. Discussion Paper Center Oper Res and Econometrics Univ Catholique de Louvain 7724, no. June
16. Jeroslow RG (1977) Cutting plane theory: Disjunctive methods. Ann Discret Math 1:293–330

17. Lovasz L, Schrijver A (1991) Cones of matrices and set functions and 0-1 optimization. SIAM J Optim 1:166–190
18. Majthay A, Whinston A (1974) Quasi-concave minimization subject to linear constraints. Discret Math 9:35–59
19. Mcaloon K, Tretkoff C (1997) Logic, modeling, and programming. Ann Oper Res 71:335–372
20. Owen G (1973) Cutting planes for programs with disjunctive constraints. Optim Theory Appl 11:49–55
21. Pinto JM, Grossmann IE (1997) A logic based approach to scheduling problems with resource constraints. Comput Chem Eng 21(8):801–818
22. Ramarao B, Shetty CM (1984) Application of disjunctive programming to the linear complementarity problem. Naval Res Logist Quart 31:589–600
23. Sakama C, Seki H (1997) Partial deduction in disjunctive logic programming. J Logic Programming 32(3):229–245
24. Sen S, Sherali HD (1985) On the convergence of cutting plane algorithms for a class of nonconvex mathematical programs. Math Program 31(1):42–56
25. Sen S, Sherali HD (1986) Facet inequalities from simple disjunctions in cutting plane theory. Math Program 34(1): 72–83
26. Sherali HD, Adams WP (1984) A decomposition algorithm for a discrete location-allocation problem. Oper Res 32(878–900
27. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J Discret Math 3(3):411–430
28. Sherali H, Adams WP (1994) A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. Discrete Appl Math 52:83–106 Manuscript, Virginia Polytechnic Inst. State Univ., 1989.
29. Sherali HD, Adams WP (1996) Computational advances using the reformulation-linearization technique (RLT) to solve discrete and continuous nonconvex problems. OPTIMA 49:1–6
30. Sherali HD, Adams WP, Driscoll P (1998) Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. Oper Res 46(3):396–405
31. Sherali HD, Sen S (1985) A disjunctive cutting plane algorithm for the extreme point mathematical programming problem. Opsearch (Theory) 22(2):83–94
32. Sherali HD, Sen S (1985) On generating cutting planes from combinatorial disjunctions. Oper Res 33(4):928–933
33. Sherali HD, Shetty CM (1980) A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts. Math Program 19:14–31
34. Sherali HD, Shetty CM (1980) On the generation of deep disjunctive cutting planes. Naval Res Logist Quart 27(3):453–475
35. Sherali HD, Shetty CM (1980) Optimization with disjunctive constraints. Lecture Notes Economics and Math Systems, vol 181. Springer, Berlin
36. Sherali HD, Shetty CM (1982) A finitely convergent procedure for facial disjunctive programs. Discrete Appl Math 4:135–148
37. Sherali HD, Shetty CM (1983) Nondominated cuts for disjunctive programs and polyhedral annexation methods. Opsearch (Theory) 20(3):129–144
38. Vecchietti A, Grossmann IE (1997) LOGMIP: A disjunctive 0-1 nonlinear optimizer for process systems models. Comput Chem Eng 21, Suppl.:S427–S432
39. Williams HP (1994) An alternative explanation of disjunctive formulations. Europ J Oper Res 72(1):200–203

Distance Dependent Protein Force Field via Linear Optimization

R. Rajgaria, S. R. McAllister,
Christodoulos A. Floudas
Department of Chemical Engineering,
Princeton University, Princeton, USA

Article Outline

Abstract

Protein force fields play an important role in protein structure prediction. Knowledge based force fields use the database information to derive the interaction energy between different residues or atoms of a protein. These simplified force fields require less computational effort and are relatively easy to use. A C^{α}–C^{α} distance dependent high resolution force field has been developed using a set of high quality (low rmsd) decoys. A linear programming based formulation was used in which non-native “decoy” conformers are forced to take a higher energy compared to the corresponding native structure. This force field was tested on an independent test set and was found to excel on all the metrics that are widely used to measure the effectiveness of a force field.

Keywords

Force field; Potential model; High resolution decoys; Protein structure prediction; Linear optimization; Protein design potential

Introduction

Predicting the structure of a protein from its amino acid sequence is one of the biggest and yet most fundamental problems in computational structural biology. Anfinsen's hypothesis [1] is one of the main approaches used to solve this problem, which says that for a given physiological set of conditions the native structure of a protein corresponds to the global Gibbs free energy minimum. Thus, one needs a force field to calculate the energy of different conformers and pick the one with the lowest energy.

Physics-based force fields consider various types of interactions (for example, van der Waals interactions, hydrogen bonding, electrostatic interactions etc.) occurring at the atomic level of a protein to calculate the energy of a conformer. CHARMM [19], AMBER [5], ECEPP [20], ECEPP/3 [21] and GROMOS [24] are a few examples of the physics-based force fields. On the other hand, knowledge-based force fields use information from databases. Researchers have used the Boltzmann distribution [4,7,26,], optimization based techniques [17,27] and many other approaches [6,12,13,14,15,16,18,23,25] to calculate these parameters. A recent review on such potentials can be found in Floudas et al. [8].

This work presents a novel C^α–C^α distance dependent high resolution force field that has been generated using linear optimization based framework [22]. The emphasis is on the high resolution, which would enable us to differentiate between native and non-native structures that are very similar to each other (rmsd < 2 Å). The force field is called high resolution because it has been trained on a large set of high resolution decoys (small rmsd with respect to the native) and it intends to effectively distinguish high resolution decoys structures from the native structure.

The basic framework used in this work is similar to the one developed by Loose et al. [17]. However, it has been improved and applied to a diverse and enhanced (both in terms of quantity and quantity) set of high resolution decoys. The new proposed model has resulted in remarkable improvements over the LKF potential. These high resolution decoys were generated using torsion angle dynamics in combination with restricted variations of the hydrophobic core within the native structure. This decoy set highly improves the quality of training and testing. The force field developed in this paper was tested by comparing the energy of the native fold to the energies of decoy structures for proteins separate from those used to train the model. Other leading force fields were also tested on this high quality decoy set and the results were compared with the results of our high resolution potential. The comparison is presented in the Results section.

Theory and Modeling

In this model, amino acids are represented by the location of its C^α atom on the amino acid backbone. The conformation of a protein is represented by a coordinate vector, X, which includes the location of the C^α of each amino acid. The native conformation is denoted as X_n, while the set $i = 1, \ldots, N$ is used to denote the decoy conformations X_i. Non-native decoys are generated for each of $p = 1, \ldots, P$ proteins and the energy of the native fold for each protein is forced to be lower than those of the decoy conformations (Anfinsen's hypothesis). This constraint is shown in the following equation:

$$\begin{aligned} &E(X_{p,i}) - E(X_{p,n}) > \varepsilon \\ &\quad p = 1, \ldots, P \quad i = 1, \ldots, N \end{aligned} \tag{1}$$

Equation (1) requires the native conformer to be always lower in energy than its decoy. A small positive parameter ε is used to avoid the trivial solution in which all energies are set to zero. An additional constraint (Eq. 2) is used to produce a nontrivial solution by constraining the sum of the differences in energies between decoy and native folds to be greater than a positive constant [28]. For the model presented in this paper, the values of ε and Γ were set to 0.01 and 1000, respectively.

$$\sum_p \sum_i [E(X_{p,i}) - E(X_{p,n})] > \Gamma \tag{2}$$

The energy of each conformation is taken as the arithmetic sum of pairwise interactions corresponding to each amino acid combination at a particular "contact" distance. A contact exists when the C^α carbons of two

Distance Dependent Protein Force Field via Linear Optimization, Table 1
Distance dependent bin definition [17]

Bin ID	C^{α} Distance [Å]
1	3–4
2	4–5
3	5–5.5
4	5.5–6
5	6–6.5
6	6.5–7
7	7–8
8	8–9

amino acids are within 9 Å of each other. The energy of each interaction is a function of the C^{α}–C^{α} distances and the identity of the interacting amino acids. To formulate the model, the energy of an interaction between a pair of amino acids, IC, within a distance bin, ID, was defined as $\theta^{\text{IC,ID}}$. The eight distance bins defined for the formulation are shown in Table 1. The energy for any fold X, of decoy i, for a protein p, is given by Eq. (3).

$$E(X_{p,i}) = \sum_{\text{IC}} \sum_{\text{ID}} N_{p,i,\text{IC},\text{ID}} \theta_{\text{IC},\text{ID}} \qquad (3)$$

In this equation, $N_{p,i,\text{IC},\text{ID}}$ is the number of interactions between an amino acid pair IC, at a C^{α}–C^{α} distance ID. The set IC ranges from 1 to 210 to account for the 210 unique combinations of the 20 naturally occurring amino acids. These bin definitions yield a total of 1680 interaction parameters to be determined by this model. To determine these parameters, a linear programming formulation is used in which the energy of a native protein is compared with a large number of its decoys. The violations, in which a non-native fold has a lower energy than the natural conformation, are minimized by optimizing with respect to these interaction parameters.

Equation (1) can be rewritten in terms of $N_{p,i,\text{IC},\text{ID}}$ as Eq. (4), where the slack parameters, S_p, are positive variables (Eq. 5) that represent the difference between the energies of the decoys and the native conformation of a given protein.

$$\sum_{\text{IC}} \sum_{\text{ID}} [N_{p,i,\text{IC},\text{ID}} - N_{p,n,\text{IC},\text{ID}}] \theta_{\text{IC},\text{ID}} + S_p \geq \varepsilon$$
$$p = 1, \ldots, P \quad i = 1, \ldots, N \qquad (4)$$

$$S_p \geq 0 \quad p = 1, \ldots, P \qquad (5)$$

$$\min_{\theta(\text{IC},\text{ID})} \sum_p S_p \qquad (6)$$

The objective function for this formulation is to minimize the sum of the slack variables, S_p, written in the form of Eq. (6). The relative magnitude of $\theta_{\text{IC,ID}}$ is meaningless because if all $\theta_{\text{IC,ID}}$ parameters are multiplied by a common factor then Eqs. (4) and (5) are still valid. In this formulation, $\theta_{\text{IC,ID}}$ values were bound between −25 and 25.

Physical Constraints

The above mentioned equations constitute the basic constraints needed to solve this model. However, this set does not guarantee a physically realistic solution. It is possible to come up with a set of parameters that can satisfy Eqs. (2,3,4,5,6) but would not reflect the actual interaction occurring between amino acids in a real system. To prohibit these unrealistic cases, another set of constraints based on the physical properties of the amino acids was imposed. Statistical results presented in Bahar and Jernigan [2] were also incorporated through the introduction of hydrophilic and hydrophobic constraints. The details of these physical constraints are given elsewhere [22].

Database Selection and Decoy Generation

The protein database selection is critical to force field training. This set should adequately represent the PDB set [3]. At the same time, it should not be too large, as the training becomes difficult with an increase in the size of the training set. Zhang and Skolinck [29] developed a set of 1,489 nonhomologous single domain proteins. High resolution decoys were generated for these proteins and used for training and testing purposes. High quality decoy generation was based on the hypothesis that high-quality decoy structures should preserve information about the distances within the hydrophobic core of the native structure of each protein. For each of the proteins in the database, a number of distance constraints are introduced based on the hydrophobic-hydrophobic distances within the native structure. Using a set of proximity parameters, a large number of decoy structures are generated using DYANA [9]. The rmsd distribution of decoy structures can be found elsewhere [22].

Training and Test Set

Of the 1400 proteins used for decoy generation, 1250 were randomly selected for training and the rest were used for testing purposes. For every protein in the set, 500–1600 decoys were generated depending on the fraction of secondary structure present in the native structure of the protein. These decoys were sorted based on their C^{α} rmsd to the native structure and then 500 decoys were randomly selected to represent the whole rmsd range. This creates a training set of $500 \times 1250 = 625,000$ decoys. However, because of computer memory limitations, it is not possible to include all of these decoys at the same time for training. An iterative scheme, "Rank and Drop", was employed to overcome the memory problem while effectively using all the high quality structures. In this scheme, a subset of decoys is used to generate a force field. This force field is then used to rank all the decoys and a set of most challenging decoys (based on their energy value) is selected for the next round of force field generation. This process of force field generation and decoy ranking is repeated until there is no improvement in the ranking of the decoys [22]. This force field model was solved using the GAMS modeling language coupled with the CPLEX linear programming package [11].

It is equally important to test a force field on a difficult and rigorous testing set to confirm its effectiveness. The test set was comprised of 150 randomly selected proteins (41–200 amino acids in length). For each of the 150 test proteins, 500 high resolution decoys were generated using the same technique that was used to generate training decoys. The minimum C^{α} based rmsds for these non-native structures were in the range of 0–2 Å. This HR force field was also tested on another set of medium resolution decoys [17]. This set has 200 decoys for 151 proteins. The minimum RMSD of the decoys of this set ranged from 3–16 Å. This set, along with the high resolution decoy set, spans the practical range of possible protein structures that one might encounter during protein structure prediction.

Results and Discussion

A linear optimization problem was solved using information from 625,000 decoy structures and the values of all the energy parameters were obtained. The ability to distinguish between the native structure and native-like conformers is the most significant test for any force field. The HR force field was tested on 500 decoys of the 150 test proteins. In this testing, the relative position, or rank, of the native conformation among its decoys was calculated. An ideal force field should be able to assign rank 1 to the native structures of all the test proteins. Other force fields like LKF [17], TE13 [27], and HL [10] were also tested on this set of high resolution decoys. All these force fields are fundamentally different from each other in their methods of energy estimation. Comparing the results obtained with these force fields aims to assess the fundamental utility of the HR force field. The comparison of the energy rankings obtained using different force fields is presented in Table 2. From this table it is evident that the HR force field is the most effective in identifying the native structures by rank. The HR force field correctly identified the native folds of 113 proteins out of a set of 150 proteins, which compares favorably to a maximum of 92 (out of 148) by the TE13 force field.

Distance Dependent Protein Force Field via Linear Optimization, Table 2
Testing force fields on 150 proteins of the high resolution decoy set. TE13 force field was only tested on 148 cases

FF-Name	Average Rank	No of Firsts	Average rmsd
HR	1.87	113 (75.33%)	0.451
LKF	39.45	17 (11.33%)	1.721
TE13	19.94	92 (62.16%)	0.813
HL	44.93	70 (46.67%)	1.092

Another analysis was carried out to evaluate the discrimination ability of these potentials. In this evaluation, all the decoys of the test set were ranked using these potentials. For each test protein, the C^{α} rmsd of the rank 1 conformer was calculated with respect to the native structure of that protein. The C^{α} rmsd would be zero for the cases in which a force field selects the native structure as rank 1. However, it will not be zero for all other cases in which a non-native conformer is assigned the top rank. The average of these rmsds represents the spatial separation of the decoys with respect to the native structure. The average rmsd value obtained for each of the force fields is shown in Table 2. It can be seen that the average C^{α} rmsd value is least for the HR force field. The average C^{α} rmsd value for the HR force field is 0.451 Å, which is much less compared to 1.721 Å by the LKF, and 0.813 Å by TE13 force field. This means that the structures predicted by the HR force fields have the least spatial deviation from their corresponding native structures.

The HR force field was also tested on the test set published by Loose et al. [17] and was found to do better than other force fields. The comparison results for this test can be found elsewhere [22]. The effectiveness of the HR force field is further reinforced by its success on the medium resolution decoy test set. On the test set of 110 medium resolution decoys, it was capable of correctly identifying 78.2 % of the native structures, significantly more than other force fields.

Conclusions

The HR force field was developed using an optimization based linear programming formulation, in which the model is trained using a diverse set of high quality decoys. Physically observed interactions between certain amino acids were written in the form of mathematical constraints and included in the formulation.

The decoys were generated based on the premise that high quality decoy structures should preserve information about the distance within the hydrophobic core of the native structure of each protein. The set of interaction energy parameters obtained after solving the model were found to be of very good discriminatory capacity. This force field performed well on a set of independent, non-homologous high resolution decoys. This force field can become a powerful tool for fold recognition and de novo protein design.

References

1. Anfinsen CB (1973) Principles that govern the folding of protein chains. Science 181:223–230
2. Bahar I, Jernigan RL (1997) Inter-residue potential in globular proteins and the dominance of highly specific hydrophillic interactions at close separation. J Molec Biol 266:195–214
3. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 28:235–242
4. Bryant SH, Lawrence CE (1991) The frequency of ion-pair substructures in proteins is quantitaively related to electrostatic potential, a statistical model for nonbonded interactions. Proteins: Structure, Function, Bioinformatics 9:108–119
5. Cornell WD, Cieplak P, Bayly CI, Gould IR, Merz Jr KM, Ferguson DM, Spellmeyer DC, Fox T, Caldwell JW, Kollman PA (1995) A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J Am Chem Soc 117:5179–5197
6. DeBolt S, Skolnick J (1996) Evaluation of atomic level mean force potentials via inverse folding and inverse refinement of proteins structures: atomic burial position and pairwise non-bonded interactions. Protein Eng 9:637–655
7. Finkelstein AV, Badretdinov AY, Gutin AM (1995) Why do proteins architectures have Boltzmann-like statistics? Proteins: Structure, Function, Bioinformatics 23:142–150
8. Floudas CA, Fung HK, McAllister SR, Mönnigmann M, Rajgaria R (2006) Advances in protein structure prediction and de novo protein design: A review. Chem Eng Sci 61:966–988
9. Güntert P, Mumenthaler C, Wüthrich K (1997) Torsion angle dynamics for NMR structure calculation with the new program DYANA. J Molec Biol 273:283–298
10. Hinds DA, Levitt M (1994) Exploring conformational space with a simple lattice model for protein structure. J Molec Biol 243:668–682
11. ILOG CPLEX (2003) User's Manual 9.0
12. Jernigan RL, Bahar I (1996) Structure-derived potentials and protein simulations. Curr Opin Struct Biol 6:195–209
13. Liwo A, Kazmierkiewicz R, Czaplewski C, Groth M, Oldziej S, Wawak RJ, Rackovsky S, Pincus MR, Scheraga HA (1998) A united-residue force field for off-lattice protein structure simulations. III. Origin of backbone hydrogen bonding cooperativity in united residue potential. J Comput Chem 19:259–276
14. Liwo A, Odziej S, Czaplewski C, Kozlowska U, Scheraga HA (2004) Parametrization of backbone-electrostatic and multibody contributions to the UNRES force field for protein-structure prediction from ab initio energy surfaces of model systems. J Phys Chem B 108:9421–9438
15. Liwo A, Oldziej S, Pincus MR, Wawak RJ, Rackovsky S, Scheraga HA (1997) A united-residue force field for off-lattice protein structure simulations. I. Functional forms and parameters of long-range side-chain interaction potentials from protein crystal data. J Comput Chem 18:849–873
16. Liwo A, Pincus MR, Wawak RJ, Rackovsky S, Oldziej S, Scheraga HA (1997) A united-residue force field for off-lattice protein structure simulations. II. Parameterization of short-range interactions and determination of weights of energy terms by z-score optimization. J Comput Chem 18:874–887
17. Loose C, Klepeis JL, Floudas CA (2004) A new pairwise folding potential based on improved decoy generation and side-chain packing. Proteins: Structure, Function, Bioinformatics 54:303–314
18. Lu H, Skolnick J (2001) A distance-dependent knowledge-based potential for improved protein structure selection. Proteins: Structure, Function, Bioinformatics 44:223–232
19. MacKerell Jr AD, Bashford D, Bellott M, Dunbrack Jr RL, Evanseck JD, Field MJ, Fischer S, Gao J, Guo H, Ha S, Joseph-McCarthy D, Kuchnir L, Kuczera K, Lau FTK, Mattos C, Michnick S, Ngo T, Nguyen DT, Prodhom B, Reiher III WE, Roux

B, Schlenkrich M, Smith JC, Stote R, Straub J, Watanabe M, Wiórkiewicz-Kuczera J, Yin D, Karplus M (1998) All-atom empirical potential for molecular modeling and dynamics studies of proteins. J Phys Chem B 102:3586–3616
20. Momany FA, McGuire RF, Burgess AW, Scheraga HA (1975) Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. J Phys Chem 79:2361–2381
21. Némethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA (1992) Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. J Phys Chem 96:6472–6484
22. Rajgaria R, McAllister SR, Floudas CA (2006) Development of a novel high resolution C^{α}–C^{α} distance dependent force field using a high quality decoy set. Proteins: Structure, Function, Bioinformatics 65:726–741
23. Samudrala R, Moult J (1998) An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction. J Molec Biol 275:895–916
24. Scott WRP, Hunenberger PH, Trioni IG, Mark AE, Billeter SR, Fennen J, Torda AE, Huber T, Kruger P, VanGunsteren WF (1997) The GROMOS biomolecular simulation program package. J Phys Chem A 103:3596–3607
25. Subramaniam S, Tcheng DK, Fenton J (1996) A knowledge-based method for protein structure refinement and prediction. In: States D, Agarwal P, Gaasterland T, Hunter L, Smith R (eds) Proceedings of the 4th International Conference on Intelligent Systems in Molecular Biology. AAAI Press, Boston, pp 218–229
26. Tanaka S, Scheraga HA (1976) Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. Macromolecules 9:945–950
27. Tobi D, Elber R (2000) Distance-dependent, pair potential for protein folding: Results from linear optimization. Proteins: Structure, Function, Bioinformatics 41:40–46
28. Tobi D, Shafran G, Linial N, Elber R (2000) On the design and analysis of protein folding potentials. Proteins: Structure, Function, Bioinformatics 40:71–85
29. Zhang Y, Skolnick J (2004) Automated structure prediction of weakly homologous proteins on a genomic scale. Proc Natl Acad Sci USA 101:7594–7599

Domination Analysis in Combinatorial Optimization

GREGORY GUTIN
Department of Computer Science, Royal Holloway, University of London, Egham, UK

MSC2000: 90C27, 90C59, 68Q25, 68W40, 68R10

Article Outline

Keywords and Phrases

Domination analysis; Domination number; Domination ratio

Introduction

Exact algorithms allow one to find optimal solutions to NP-hard combinatorial optimization (CO) problems. Many research papers report on solving large instances of some NP-hard problems (see, e. g., [25,27]). The running time of exact algorithms is often very high for large instances (many hours or even days), and very large instances remain beyond the capabilities of exact algorithms. Even for instances of moderate size, if we wish to remain within seconds or minutes rather than hours or days of running time, only heuristics can be used. Certainly, with heuristics, we are not guaranteed to find optimum, but good heuristics normally produce near-optimal solutions. This is enough in most applications since very often the data and/or mathematical model are not exact anyway.

Research on CO heuristics has produced a large variety of heuristics especially for well-known CO problems. Thus, we need to choose the best ones among them. In most of the literature, heuristics are compared in computational experiments. While experimental analysis is of definite importance, it cannot cover all possible families of instances of the CO problem at hand and, in particular, it practically never covers the hardest instances.

Approximation Analysis [3] is a frequently used tool for theoretical evaluation of CO heuristics. Let $\mathcal{H}$ be a heuristic for a combinatorial minimization problem P and let $\mathcal{I}_n$ be the set of instances of P of size n. In approximation analysis, we use the performance ratio $r_H(n) = \max\{f(I)/f^*(I) : I \in \mathcal{I}_n\}$, where $f(I)(f^*(I))$ is the value of the heuristic (optimal) solution of I. Unfortunately, for many CO problems, estimates for $r_{\mathcal{H}}(n)$ are not constants and provide only a vague picture of the quality of heuristics. Moreover, even constant performance ratio does not guarantee that the heuristic often outputs good-quality solutions, see, e. g., the discussion of the DMST heuristic below.

Domination Analysis (DA) (for surveys, see [22,24]) provides an alternative and a complement to approximation analysis. In DA, we are interested in the domination number or domination ratio of heuristics. *Domination number* (*ratio*) of a heuristic H for a combinatorial optimization problem P is the maximum number (fraction) of all solutions that are not better than the solution found by H for any instance of P of size n. In many cases, DA is very useful. For example, we will see later that the greedy algorithm has domination number 1 for many CO problems. In other words, the greedy algorithm, in the worst case, produces the unique worst possible solution. This is in line with latest computational experiments with the greedy algorithm, see, e. g., [25], where the authors came to the conclusion that the greedy algorithm 'might be said to self-destruct' and that it should not be used even as 'a general-purpose starting tour generator'.

The *Asymmetric Traveling Salesman Problem (ATSP)* is the problem of computing a minimum weight tour (Hamilton cycle) passing through every vertex in a weighted complete digraph K_n^* on n vertices. The *Symmetric TSP (STSP)* is the same problem, but on a complete undirected graph. When a certain fact holds for both ATSP and STSP, we will simply speak of *TSP*. Sometimes, the maximizing version of TSP is of interest, we denote it by *Max TSP*.

APX is the class of CO problems that admit polynomial time approximation algorithms with a constant performance ratio [3]. It is well known that while Max TSP belongs to APX, TSP does not. This is at odds with the simple fact that a 'good' approximation algorithm for Max TSP can be easily transformed into an algorithm for TSP. Thus, it seems that both Max TSP and TSP should be in the same class of CO problems. The above asymmetry was already viewed as a drawback of performance ratio already in the 1970's, see, e. g. [11,28,33]. Notice that from the DA point view Max TSP and TSP are equivalent problems.

Zemel [33] was the first to characterize measures of quality of approximate solutions (of binary integer programming problems) that satisfy a few basic and natural properties: the measure becomes smaller for better solutions, it equals 0 for optimal solutions and it is the same for corresponding solutions of equivalent instances. While the performance ratio and even the relative error (see [3]) do not satisfy the last property, the parameter *1-r*, where *r* is the domination ratio, does satisfy all of the properties.

Local Search (LS) is one of the most successful approaches in constructing heuristics for CO problems. Recently, several researchers investigated LS with Very Large Scale Neighborhoods (see, e. g., [1,12,24]). The hypothesis behind this approach is that the larger the neighborhood the better quality solution are expected to be found [1]. However, some computational experiments do not support this hypothesis; sometimes an LS with small neighborhoods proves to be superior to that with large neighborhoods. This means that some other parameters are responsible for the relative power of neighborhoods. Theoretical and experimental results on TSP indicate that one such parameter may well be the domination number of the corresponding LS.

In our view, it is advantageous to have bounds for both performance ratio and domination number (or, domination ratio) of a heuristic whenever it is possible. Roughly speaking this will enable us to see a 2D rather than 1D picture. For example, consider the double minimum spanning tree heuristic (DMST) for the Metric STSP (i. e., STSP with triangle inequality). DMST starts from constructing a minimum weight spanning tree T in the complete graph of the STSP, doubles every edge in T, finds a closed Euler trail E in the 'double' T, and cancels any repetition of vertices in E to obtain a TSP tour H. It is well-known and easy to prove that the weight of H is at most twice the weight of the optimal tour. Thus, the performance ratio for DMST is bounded by 2. However, Punnen, Margot and Kabadi [29] proved that the domination number of DMST is 1. Interestingly, in practice DMST often performs much worse than the well-known 2-Opt LS heuristic. For 2-Opt LS we cannot give any constant approximation guarantee, but the heuristic is of very large domination number [29].

The above example indicates that it makes sense to use DA to rank heuristics for the CO problem under consideration. If the domination number of a heuristic $\mathcal{H}$ is larger than the domination of a heuristic $\mathcal{H}'$ (for all or 'almost all' sizes n), we may say that $\mathcal{H}$ is better than $\mathcal{H}'$ in the worst case (from the DA point of view). Berend, Skiena and Twitto [10] used DA to rank some well-known heuristics for the Vertex Cover problem (and, thus, the Independent Set and Clique problems). The three problems and the heuristics will be defined in the corresponding subsection of the Cases section. Ben-Arieh et al. [7] studied three heuristics for the *Generalized TSP*: the vertices of the complete digraph are partitioned into subsets and the goal is to find a minimum weight cycle containing exactly one vertex from each subset. In the computational experiment in [7] one of the heuristics was clearly inferior to the other two. The best two behaved very similarly. Nevertheless, the authors of [7] managed to 'separate' the two heuristics by showing that one of the heuristics was of much larger domination number.

One might wonder whether a heuristic $\mathcal{A}$, which is significantly better that another heuristic $\mathcal{B}$ from the DA point of view, is better that $\mathcal{B}$ in computational experiments. In particular, whether the ATSP greedy algorithm, which is of domination number 1, is worse, in computational experiments, than any ATSP heuristic of domination number at least $(n-2)!$? Generally speaking the answer to this natural question is negative. This is because computational experiments and DA indicate different aspects of quality of heuristics. Nevertheless, it seems that many heuristics of very small domination number such as the ATSP greedy algorithm perform poorly also in computational experiments and, thus, cannot be recommended to be widely used in computational practice.

The rest of the entry is organized as follows. We give additional terminology and notation in the section Definitions. In the section Methods, we describe two powerful methods in DA. In the section Cases, we consider DA results for some well-known CO problems.

Definitions

Let $\mathcal{P}$ be a CO problem and let $\mathcal{H}$ be a heuristic for $\mathcal{P}$. The *domination number* $\text{domn}(\mathcal{H},\mathcal{I})$ of $\mathcal{H}$ *for an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are not better than the solution s produced by $\mathcal{H}$ including s itself. For example, consider an instance $\mathcal{T}$ of the STSP on 5 vertices. Suppose that the weights of tours in $\mathcal{T}$ are 2,5,5,6,6,9,9,11,11,12,12,15 (every instance of STSP on 5 vertices has 12 tours) and suppose that the greedy algorithm computes the tour T of weight 6. Then $\text{domn}(\texttt{greedy},\mathcal{T}) = 9$. In general, if $\text{domn}(\mathcal{H},\mathcal{I})$ equals the number of solutions in $\mathcal{I}$, then $\mathcal{H}$ finds an optimal solution for $\mathcal{I}$. If $\text{domn}(\mathcal{H},\mathcal{I}) = 1$, then the solution found by $\mathcal{H}$ for $\mathcal{I}$ is the unique worst possible one.

The *domination number* $\text{domn}(\mathcal{H},n)$ of $\mathcal{H}$ is the minimum of $\text{domn}(\mathcal{H},\mathcal{I})$ over all instances $\mathcal{I}$ of size n. Since the ATSP on n vertices has $(n-1)!$ tours, an algorithm for the ATSP with domination number $(n-1)!$ is exact. The domination number of an exact algorithm for the STSP is $(n-1)!/2$. If an ATSP heuristic $\mathcal{A}$ has domination number equal 1, then there is an assignment of weights to the arcs of each complete digraph K_n^*, $n \geq 2$, such that $\mathcal{A}$ finds the unique worst possible tour in K_n^*.

While studying TSP we normally consider only feasible solutions (tours), for several other problems some authors take into consideration also infeasible solutions [10]. One example is the *Maximum Independent Set* problem, where given a graph G, the aim is to find an independent set in G of maximum cardinality. Every non-empty set of vertices is considered to be a solution by Berend, Skiena and Twitto [10]. To avoid dealing with infeasible solutions (and, thus, reserving the term 'solution' only for feasible solutions) we also use the notion of the *blackball number* introduced in [10]. The *blackball number* $\text{bbn}(\mathcal{H},\mathcal{I})$ of $\mathcal{H}$ *for a an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are better than the solution produced by $\mathcal{H}$. The *blackball number* $\text{bbn}(\mathcal{H},n)$ of $\mathcal{H}$ is the maximum of $\text{domn}(\mathcal{H},\mathcal{I})$ over all instances $\mathcal{I}$ of size n.

When the number of solutions depends not only on the size of the instance of the CO problem at hand (for example, the number of independent sets of vertices in a graph G on n vertices depends on the structure of G), the domination ratio of an algorithm $\mathcal{A}$ is of interest: the *domination ratio* of $\mathcal{A}$, $\mathrm{domr}(\mathcal{A}, n)$, is the minimum of $\mathrm{domn}(\mathcal{A}, \mathcal{I})/\mathrm{sol}(\mathcal{I})$, where $\mathrm{sol}(\mathcal{I})$ is the number of solutions of $\mathcal{I}$, taken over all instances $\mathcal{I}$ of size n. Clearly, domination ratio belongs to the interval $(0, 1]$ and exact algorithms are of domination ratio 1.

Methods

Currently, there are two powerful methods in DA. One is used to prove that the heuristic under consideration is of domination number 1. For this method to be useful, the heuristic has to be a greedy-type algorithm for a CO problem on independence systems. We describe the method and its applications in the subsection Greedy-Type Algorithms. The other method is used prove that the heuristic under consideration is of very large domination number. For many problems this follows from the fact that the heuristic always finds a solution that is not worse than the average solution. This method is described in the subsection Better-Than-Average Heuristics.

Greedy-Type Algorithms

The main practical message of this subsection is that one should be careful while using the classical greedy algorithm and its variations in combinatorial optimization (CO): there are many instances of CO problems for which such algorithms will produce the unique worst possible solution. Moreover, this is true for several well-known optimization problems and the corresponding instances are not exotic, in a sense. This means that not always the paradigm of greedy optimization provides any meaningful optimization at all.

An *independence system* is a pair consisting of a finite set E and a family $\mathcal{F}$ of subsets (called *independent sets*) of E such that (I1) and (I2) are satisfied.

(I1) the empty set is in $\mathcal{F}$;
(I2) If $X \in \mathcal{F}$ and Y is a subset of X, then $Y \in \mathcal{F}$.

All maximal sets of $\mathcal{F}$ are called *bases*. An independence system is *uniform* if all its bases are of the same cardinality.

Many combinatorial optimization problems can be formulated as follows. We are given an independence system $(E,\mathcal{F})$, a set $W \subseteq \mathbb{Z}_+$ and a weight function w that assigns a weight $w(e) \in W$ to every element of E ($\mathbb{Z}_+$ is the set of non-negative integers). The weight $w(S)$ of $S \in \mathcal{F}$ is defined as the sum of the weights of the elements of S. It is required to find a base $B \in \mathcal{F}$ of minimum weight. We will consider only such problems and call them the *(E,F,W)-optimization problems*.

If $S \in \mathcal{F}$, then let $I(S) = \{x : S \cup \{x\} \in \mathcal{F}\} - S$. This means that $I(S)$ consists of those elements from E-S, which can be added to S, in order to have an independent set of size $|S| + 1$. Note that by (I2) $I(S) \neq \emptyset$ for every independent set S which is not a base.

The greedy algorithm tries to construct a minimum weight base as follows: it starts from an empty set X, and at every step it takes the current set X and adds to it a minimum weight element $e \in I(X)$, the algorithm stops when a base is built. We assume that the greedy algorithm may choose any element among equally weighted elements in $I(X)$. Thus, when we say that the greedy algorithm *may construct* a base B, we mean that B is built provided the appropriate choices between elements of the same weight are made.

An *ordered partitioning* of an ordered set $Z = \{z_1, z_2, \dots, z_k\}$ is a collection of subsets $A_1, A_2, \dots, A_q$ of Z such that if $z_r \in A_i$ and $z_s \in A_j$ where $1 \le i < j \le q$ then $r < s$. Some of the sets A_i may be empty and $\cup_{i=1}^{q} A_i = Z$.

The following theorem by Bang-Jensen, Gutin and Yeo [6] characterizes all uniform independence systems $(E,\mathcal{F})$ for which there is an assignment of weights to the elements of E such that the greedy algorithm solving the $(E, \mathcal{F}, \{1, 2, \dots, r\})$-optimization problem may construct the unique worst possible solution.

Theorem 1 *Let $(E,\mathcal{F})$ be a uniform independence system and let $r \ge 2$ be a natural number. There exists a weight assignment $w : E \to \{1, 2, \dots, r\}$ such that the greedy algorithm may produce the unique worst possible base if and only if $\mathcal{F}$ contains some base B with the property that for some ordering $x_1, \dots, x_k$ of the elements of B and some ordered partitioning $A_1, A_2, \dots, A_r$ of $x_1, \dots, x_k$ the following holds for every base $B' \neq B$ of $\mathcal{F}$:*

$$\sum_{j=0}^{r-1} |I(A_{0,j}) \cap B'| < \sum_{j=1}^{r} j \cdot |A_j| \,, \qquad (1)$$

where $A_{0,j} = A_0 \cup \dots \cup A_j$ and $A_0 = \emptyset$.

The special case $r = 2$ has an 'easier' characterization also proved in [6].

Theorem 2 *Let $(E,\mathcal{F})$ be a uniform independence system. For every choice of distinct natural numbers a,b there exists a weight function $w\colon E \to \{a, b\}$ such that the greedy algorithm may produce the unique worst base if and only if $\mathcal{F}$ contains a base $B = \{x_1, x_2, \ldots, x_k\}$ such that for some $1 \le i < k$ the following holds:*

(a) *If B' is a base such that $\{x_1, \ldots, x_i\} \subseteq B'$ then $B' = B$.*
(b) *If B' is a base such that $\{x_{i+1}, \ldots, x_k\} \subseteq B'$ then $B' = B$.*

Using Theorem 1, the authors of [6] proved the following two corollaries.

Corollary 3 *Consider STSP as an $(E,\mathcal{H},W)$-optimization problem. Let $n \ge 3$.*

(a) *If $n \ge 4$ and $|W| \le \lfloor \frac{n-1}{2} \rfloor$, then the greedy algorithm never produces the unique worst possible base (i. e., tour).*
(b) *If $n \ge 3$, $r \ge n - 1$ and $W = \{1, 2, \ldots, r\}$, then there exists a weight function $w\colon E \to \{1, 2, \ldots, r\}$ such that the greedy algorithm may produce the unique worst possible base (i. e., tour).*

Corollary 4 *Consider ATSP as an $(E,\mathcal{H},W)$-optimization problems. Let $n \ge 3$.*

(a) *If $|W| \le \lfloor \frac{n-1}{2} \rfloor$, then the greedy algorithm never produces the unique worst possible base (i. e., tour).*
(b) *For every $r \ge \lceil \frac{n+1}{2} \rceil$ there exists a weight function $w\colon E(K_n^*) \to \{1, 2, \ldots, r\}$ such that the greedy algorithm may produce the unique worst possible base (i. e., tour).*

Let $\mathcal{F}$ be the sets of those subsets X of $E(K_{2n})$ which induce a bipartite graph with at most n vertices in each partite set. Then $(E(K_{2n}), \mathcal{F})$ is a uniform independence system and the bases of $(E(K_{2n}), \mathcal{F})$ correspond to copies of the complete balanced bipartite graph $K_{n,n}$ in K_{2n}. The $(E(K_{2n}), \mathcal{F}, \mathbb{Z}_+)$-optimization problem is called the *Minimum Bisection Problem*. Theorem 2 implies the following:

Corollary 5 [6] *Let $n \ge 4$. The greedy algorithm for the $(E(K_{2n}), \mathcal{F}, W)$-optimization problem may produce the unique worst solution even if $|W| = 2$.*

For $W = \mathbb{Z}_+$, the following sufficient condition can often be used:

Theorem 6 [21] *Let $(E,\mathcal{F})$ be an independence system which has a base $B' = \{x_1, x_2, \ldots, x_k\}$ such that the following holds for every base $B \in \mathcal{F}$, $B \neq B'$,*

$$\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| < k(k+1)/2\,.$$

Then the greedy algorithm for the $(E, \mathcal{F}, \mathbb{Z}_+)$-optimization problem may produce the unique worst solution.

Gutin, Yeo and Zverovich [23] considered the well-known *nearest neighbor (NN)* TSP heuristic: the tour starts at any vertex x of the complete directed or undirected graph; we repeat the following loop until all vertices have been included in the tour: add to the tour a vertex (among vertices not yet in the tour) closest to the vertex last added to the tour. It was proved in [23] that the domination number of NN is 1 for any $n \ge 3$.

Bendall and Margot [8] studied greedy-type algorithms for many CO problems. Greedy-type algorithms were introduced in [18]. They include NN and were defined as follows. A *greedy-type* algorithm $\mathcal{H}$ is similar to the greedy algorithm: start with the partial solution $X = \emptyset$; and then repeatedly add to X an element of minimum weight in $I_{\mathcal{H}}(X)$ (ties are broken arbitrarily) until X is a base of $\mathcal{F}$, where $I_{\mathcal{H}}(X)$ is a subset of $I(X)$ that does not depend on the cost function c, but only on the independence system $(E,\mathcal{F})$ and the set X. Moreover, $I_{\mathcal{H}}(X)$ is non-empty if $I(X) \neq \emptyset$, a condition that guarantees that $\mathcal{H}$ always outputs a base. Bendall and Margot [8] obtained complicated sufficient conditions for an independent system $(E,\mathcal{F})$ that ensure that every greedy-type algorithm is of domination number 1 for the $(E, \mathcal{F}, \mathbb{Z}_+)$-optimization problem.

The conditions imply that every greedy-type algorithm is of domination number 1 for the following classical CO problems [8]: (1) The Minimum Bisection Problem; (2) The *k-Clique Problem*: find a set of k vertices in a complete graph so that the sum of the weights of the edges between them is minimum; (3) ATSP; (4) STSP; (5) The *MinMax Matching Subgraph Problem*:

find a maximal (with respect to inclusion) matching so that the sum of the weights of the edges in the matching is minimum; (6) The *Assignment Problem*: find a perfect matching in a weighted complete bipartite graph so that the sum of the weights of the edges in the matching is minimum.

Better-Than-Average Heuristics

The idea of this method is to show that a heuristic is of very large domination number if it always produces a solution that is not worse than the average solution. The first such result was proved by Rublineckii [31] for the STSP.

Theorem 7 *Every STSP heuristic that always produces a tour not worse than the average tour (of the instance) is of domination number at least* $(n-2)!$ *when n is odd and* $(n-2)!/2$ *when n is even.*

The following similar theorem was proved by Sarvanov [32] for *n* odd and Gutin and Yeo [20] for *n* even.

Theorem 8 *Every ATSP heuristic that always produces a tour not worse than the average tour (of the instance) is of domination number at least* $(n-2)!$ *for each* $n \neq 6$.

The two theorems has been used to prove that a wide variety of TSP heuristics have domination number at least $\Omega((n-2)!)$. We discuss two families of such heuristics.

Consider an instance of the ATSP (STSP). Order the vertices $x_1, x_2, \ldots, x_n$ of K_n^* (K_n) using some rule. The *generic vertex insertion algorithm* proceeds as follows. Start with the cycle $C_2 = x_1x_2x_1$. Construct the cycle C_j from C_{j-1} ($j = 3, 4, 5, \ldots, n$), by inserting the vertex x_j into $C_{j\text{-}1}$ at the optimum place. This means that for each arc $e = xy$ which lies on the cycle $C_{j\text{-}1}$ we compute $w(xx_j) + w(x_jy) - w(xy)$, and insert x_j into the arc $e = xy$, which obtains the minimum such value. Here *w(uv)* denotes the weight of an arc *uv*. E.M. Lifshitz (see [31]) was the first to prove that the generic vertex insertion algorithm always produces a tour not worse than the average tour. Thus, we have the following:

Corollary 9 *The generic vertex insertion algorithm has domination number at least* $(n-2)!$ $(n \neq 6)$.

In TSP local search (LS) heuristics, a neighborhood *N(T)* is assigned to every tour *T*; *N(T)* is a set of tours in some sense close to *T*. The *best improvement* LS proceeds as follows. We start from a tour T_0. In the *i*'th iteration ($i \geq 1$), we search in the neighborhood $N(T_{i-1})$ for the best tour T_i. If the weights of $T_{i\text{-}1}$ and T_i do not coincide, we carry out the next iteration. Otherwise, we output T_i.

The *k-Opt*, $k \geq 2$, neighborhood of a tour *T* consists of all tours that can be obtained by replacing a collection of *k* edges (arcs) by a collection of *k* edges (arcs). It is easy to see that one iteration of the best improvement *k*-Opt LS can be completed in time $O(n^k)$. Rublineckii [31] showed that every local optimum for the best improvement 2-Opt or 3-Opt LS for STSP is of weight at least the average weight of a tour and, thus, by Theorem 7 is of domination number at least $(n-2)!/2$ when *n* is even and $(n-2)!$ when *n* is odd. Observe that this result is of restricted interest since to reach a *k*-Opt local optimum one may need exponential time (cf. Section 3 in [26]). However, Punnen, Margot and Kabadi [29] managed to prove that, in polynomial time, the best improvement 2-Opt and 3-Opt LS's for STSP produce a tour of weight at least the average weight of a tour. Thus, we have the following:

Corollary 10 *For the STSP the best improvement 2-Opt LS produces a tour, which is not worse than at least* $\Omega((n-2)!)$ *other tours, in at most* $O(n^3 \log n)$ *iterations.*

Corollary 10 is also valid for the best improvement 3-Opt LS and some other LS heuristics for TSP, see [24,29]. In the next section, we will give further examples of better-than-average heuristics for problems other than TSP.

Cases

Traveling Salesman Problem

In the previous sections, we discussed several TSP heuristics. However, there are many more TSP heuristics and, in this subsection, we consider some of them. In the next subsection, some general upper bounds are given on the domination number of TSP heuristics.

Gutin, Yeo and Zverovich [23] considered the *repeated nearest neighbor (RNN)* heuristic, which is the following variation of the NN heuristic: construct *n* tours by starting NN from each vertex of the complete (di)graph and choose the best tour among the *n* tours. The authors of [23] proved that for the ATSP, RNN always produces a tour, which is not worse than at least $n/2-1$ other tours, but for some instance it finds a tour, which is not worse than at most *n-2* other tours, $n \geq 4$. We also show that, for some instance of the STSP on $n \geq 4$ vertices, RNN produces a tour not worse than at most 2^{n-3} tours.

Another ATSP heuristic, max-regret-fc (fc abbreviates First Coordinate), was first introduced by Ghosh et al. [13]. Extensive computational experiments in [13] demonstrated a clear superiority of max-regret-fc over the greedy algorithm and several other construction heuristics from [14]. Therefore, the result of Theorem 11 obtained by Gutin, Goldengorin and Huang [15] was somewhat unexpected.

Let K_n^* be a complete digraph with vertices $V = \{1, 2, \dots, n\}$. The weight of an arc *(i,j)* is denoted by w_{ij}. Let Q be a collection of disjoint paths in K_n^*. An arc *a=(i,j)* is a *feasible addition* to Q if *Q+a* is either a collection of disjoint paths or a tour in K_n^*. Consider the following two ATSP heuristics: max-regret-fc and max-regret.

The heuristic *max-regret-fc* proceeds as follows. Set $W = T = \emptyset$. While $V \neq W$ do the following: For each $i \in V \setminus W$, compute two lightest arcs *(i,j)* and *(i,k)* that are feasible additions to T, and compute the difference $\Delta_i = |w_{ij} - w_{ik}|$. For $i \in V \setminus W$ with maximum Δ_i choose the lightest arc *(i,j)*, which is a feasible addition to T and add *(i,j)* to M and i to W.

The heuristic *max-regret* proceeds as follows. Set $W^+ = W^- = T = \emptyset$. While $V \neq W^+$ do the following: For each $i \in V \setminus W^+$, compute two lightest arcs *(i,j)* and *(i,k)* that are feasible additions to T, and compute the difference $\Delta_i^+ = |w_{ij} - w_{ik}|$; for each $i \in V \setminus W^-$, compute two lightest arcs *(j,i)* and *(k,i)* that are feasible additions to T, and compute the difference $\Delta_i^- = |w_{ji} - w_{ki}|$. Compute $i' \in V \setminus W^+$ with maximum $\Delta_{i'}^+$ and $i'' \in V \setminus W^-$ with maximum $\Delta_{i''}^-$. If $\Delta_{i'}^+ \geq \Delta_{i''}^-$ choose the lightest arc (i', j'), which is a feasible addition to T and add (i', j') to M, i' to W^+ and j' to W^-. Otherwise, choose the lightest arc (j'', i''), which is a feasible addition to T and add (j'', i'') to M, i'' to W^- and j'' to W^+.

Notice that in max-regret-fc, if $|V \setminus W| = 1$ we set $\Delta_i = 0$. A similar remark applies to max-regret.

Theorem 11 *The domination number of both max-regret-fc and max-regret equals 1 for each* $n \geq 2$.

Upper Bounds for Domination Numbers of ATSP Heuristics

It is realistic to assume that any ATSP algorithm spends at least one unit of time on every arc of K_n^* that it considers. We use this assumption in this subsection.

Theorem 12 [17] *Let $\mathcal{A}$ be an ATSP heuristic of running time t(n). Then the domination number of $\mathcal{A}$ does not exceed* $\max_{1 \leq n' \leq n} (t(n)/n')^{n'}$.

Corollary 13 [17] *Let $\mathcal{A}$ be an ATSP heuristic of complexity t(n). Then the domination number of $\mathcal{A}$ does not exceed* $\max\{e^{t(n)/e}, (t(n)/n)^n\}$, *where e is the basis of natural logarithms.*

The next assertion follows directly from the proof of Corollary 13.

Corollary 14 [17] *Let $\mathcal{A}$ be an ATSP heuristic of complexity t(n). For $t(n) \geq en$, the domination number of $\mathcal{A}$ does not exceed* $(t(n)/n)^n$.

We finish this subsection with a result from [17] that improves (and somewhat clarifies) Theorem 20 in [29].

Theorem 15 *Unless P = NP, there is no polynomial time ATSP algorithm of domination number at least* $(n-1)! - \lfloor n - n^{\alpha} \rfloor!$ *for any constant* $\alpha < 1$.

Multidimensional Assignment Problem (MAP)

In case of s dimensions, MAP is abbreviated by s-AP and defined as follows. Let $X_1 = X_2 = \cdots = X_s = \{1, 2, \dots, n\}$. We will consider only vectors that belong to the Cartesian product $X = X_1 \times X_2 \times \cdots \times X_s$. Each vector e is assigned a weight *w(e)*. For a vector e, e_j denotes its jth coordinate, i. e., $e_j \in X_j$. A *partial assignment* is a collection $e^1, e^2, \dots, e^t$ of $t \leq n$ vectors such that $e_j^i \neq e_j^k$ for each $i \neq k$ and $j \in \{1, 2, \dots, s\}$. An *assignment* is a partial assignment with n vectors. The *weight* of a partial assignment $A = \{e^1, e^2, \dots, e^t\}$ is $w(A) = \sum_{i=1}^{t} w(e^i)$. The objective is to find an assignment of minimum weight. Notice that s-AP has $(n!)^{s-1}$ solutions (assignments).

s-AP can be considered as the $(X, \mathcal{F}, \mathbb{Z}_+)$-optimization problem. ($\mathcal{F}$ consists of partial assignments including the empty one.) This allows us to define the greedy algorithm for s-AP and to conclude from Theorem 6 that the greedy algorithm is of domination number 1 (for every fixed $s \geq 3$).

In the subsection Traveling Salesman Problem, we considered the *max-regret-fc and max-regret heuristics*. In fact, max-regret was first introduced for 3-AP by Balas and Saltzman [4]. (See [15] for detailed description of the s-AP max-regret-fc and max-regret heuristics for each $s \geq 2$.) In computational experiments, Balas and Saltzman [4] compared the greedy algorithm with max-regret and concluded that max-regret is superior to the greedy algorithm with respect to the quality of solutions. However, after conducting wider computational experiments, Robertson [30] came to a different conclusion: the greedy algorithm and max-regret are of similar quality for 3-AP. Gutin, Goldengorin and Huang [15] share the conclusion of Robertson: both max-regret and max-regret-fc are of domination number 1 (similarly to the greedy algorithm) for s-AP for each $s \geq 3$. Moreover, there exists a family of s-AP instances for which all three heuristics will find the unique worst assignment [15] (for each $s \geq 3$).

Similarly to TSP, we may obtain MAP heuristics of factorial domination number if we consider better-than-average heuristics. This follows from the next theorem:

Theorem 16 [15] *Let $\mathcal{H}$ be a heuristic that for each instance of s-AP constructs an assignment of weight at most the average weight of an assignment. Then the domination number of $\mathcal{H}$ is at least $((n-1)!)^{s-1}$.*

Balas and Saltzman [4] introduced a *3-Opt heuristic* for 3-AP which is similar to the 3-Opt TSP heuristic. The *3-Opt neighborhood* of an assignment $A = \{e^1, e^2, \ldots, e^n\}$ is the set of all assignments that can be obtained from A by replacing a triple of vectors with another triple of vectors. The 3-Opt is a local search heuristic that uses the 3-Opt neighborhood. It is proved in [15] that an assignment, that is the best in its 3-Opt neighborhood, is at least as good as the average assignment. This implies that 3-Opt is of domination number at least $((n-1)!)^2$. We cannot guarantee that 3-Opt local search will stop after polynomial number of iterations. Moreover, 3-Opt is only for 3-AP. Thus, the following heuristic introduced and studied in [15] is of interest.

Recursive Opt Matching (ROM) proceeds as follows. Compute a new weight $\bar{w}(i,j) = w(X_{ij})/n^{s-2}$, where X_{ij} is the set of all vectors with last two coordinates equal i and j, respectively. Solving the 2-AP with the new weights to optimality, find an optimal assignment $\{(i, \pi_s(i)) : i = 1, 2, \ldots, n\}$, where π_s is a permutation on X_s. While $s \neq 1$, introduce *(s-1)*-AP with weights given as follows: $w'(f^i) = w(f^i, \pi_s(i))$ for each vector $f^i \in X'$, where $X'n = X_1 \times X_2 \times \cdots \times X_{s-1}$, with last coordinate equal i and apply ROM recursively. As a result we have obtained permutations $\pi_s, \pi_{s-1}, \ldots, \pi_2$. The output is the assignment $\{(i, \pi_2(i), \ldots, \pi_s(\pi_{s-1}(\ldots(\pi_2(i)))\ldots)) : i = 1, 2, \ldots, n\}$.

Clearly, ROM is of running time $O(n^3)$ for every fixed $s \geq 3$. Using Theorem 16, it is proved in [15] that ROM is of domination number at least $((n-1)!)^{s-1}$.

Minimum Partition and Multiprocessor Scheduling Problems

In this subsection, N always denotes the set $\{1, 2, \ldots, n\}$ and each $i \in N$ is assigned a positive integral weight $\sigma(i)$. $\mathcal{A} = (A_1, A_2, \ldots, A_p)$ is a *p-partition* of N if each $A_i \subseteq N$, $A_i \cap A_j = \emptyset$ for each $i \neq j$ and the union of all sets in $\mathcal{A}$ equals N. For a subset A of N, $\sigma(A) = \sum_{i \in A} \sigma(i)$. The *Minimum Multiprocessor Scheduling Problem* (MMS) [3] can be stated as follows. We are given a triple (N, σ, p), where p is an integer, $p \geq 2$. We are required to find a p-partition C of N that minimizes $\sigma(\mathcal{A}) = \max_{1 \leq i \leq p} \sigma(A_i)$ over all p-partitions $\mathcal{A} = (A_1, A_2, \ldots, A_p)$ of N.

Clearly, if $p \geq n$, then MMS becomes trivial. Thus, in what follows, $p < n$. The size s of MMS is $\Theta(n + \sum_{i=1}^{n} \log \sigma(i))$. Consider the following heuristic $\mathcal{H}$ for MMS. If $s \geq p^n$, then we simply solve the problem optimally. This takes $O(s^2)$ time, as there are at most $O(s)$ solutions, and each one can be evaluated and compared to the current best in $O(s)$ time. If $s < p^n$, then we sort the elements of the sequence $\sigma(1), \sigma(2), \ldots, \sigma(n)$. For simplicity of notation, assume that $\sigma(1) \geq \sigma(2) \geq \cdots \geq \sigma(n)$. Compute $r = \lceil \log n / \log p \rceil$ and solve MMS for $(\{1, 2, \ldots, r\}, \sigma, p)$ to optimality. Suppose we have obtained a p-partition $\mathcal{A}$ of $\{1, 2, \ldots, r\}$. Now for i from *r+1* to n add i to the set A_j of the current p-partition $\mathcal{A}$ with smallest $\sigma(A_j)$. The following result was proved by Gutin, Jensen and Yeo [16].

Theorem 17 *The heuristic $\mathcal{H}$ runs in time $O(s^2 \log s)$ and* $\lim_{s \to \infty} \operatorname{domr}(\mathcal{H}, s) = 1$.

The *Minimum Partition Problem (MP)* is MMS with *p=2*. Alon, Gutin and Krivelevich [2] proved Theorem 17 for MP with s replaced by n.

Max Cut Problem

The *Max Cut (MC)* is the following problem: given a weighted graph *G=(V,E)*, find a bipartition (a *cut*) *(X,Y)* of *V* such that the sum of weights of the edges with one end vertex in *X* and the other in *Y*, called the *weight of the cut (X,Y)*, is maximum. For this problem, there are some better-than-average heuristics. The simplest is probably the following greedy-like heuristic *C*; order the vertices arbitrarily and put each vertex in its turn either in *X* or in *Y* in order to maximize in each step the total weight of crossing edges.

Using an advanced probabilistic approach Alon, Gutin and Krivelevich [2] proved that the heuristic *C* is of domination ratio larger than 0.025. For the unweighted MC (all weights are equal), a better quality algorithm can be designed as described in [2]. Its domination ratio is at least $1/3 - o(1)$.

Constraint Satisfaction Problems

Let r be a fixed positive integer, and let $\mathcal{F} = \{f_1, f_2, \dots, f_m\}$ be a collection of Boolean functions, each involving at most r of the n variables, and each having a positive weight $w(f_i)$. The *Max-r-Constraint Satisfaction Problem* (or *Max-r-CSP*, for short), is the problem of finding a truth assignment to the variables so as to maximize the total weight of the functions satisfied. Note that this includes, as a special case, the Max Cut problem. Another interesting special case is the Max-*r*-SAT problem, in which each of the Boolean functions f_i is a clause of at most *r* literals.

Alon, Gutin and Krivelevich [2] proved the following:

Theorem 18 *For each fixed integer $r \geq 1$ there exists a linear time algorithm for the Max-r-CSP problem, whose domination ratio exceeds* $\frac{1}{2^{4/3} \cdot 2^{6r}}$.

Vertex Cover, Independent Set and Clique Problems

A *clique* in a graph *G* is a set of vertices in *G* such that every pair of vertices in the set are connected by an edge. The *Maximum Clique Problem* (*MCl*) is the problem of finding a clique of maximum cardinality in a graph. A *vertex cover* in a graph *G* is a set *S* of vertices in *G* such that every edge is incident to a vertex in *S*. The *Minimum Vertex Cover Problem* (*MVC*) is the problem of finding a minimum cardinality vertex cover. An *independent set* in a graph is a set *S* of vertices such that no edge joins two vertices in *S*. The *Maximum Independent Set Problem* (*MIS*) is the problem of finding a minimum cardinality independent set in a graph. It is easy to see that the number of cliques and independent sets in a graph depends on its structure, and not only on the number of vertices. The same holds for vertex covers.

Notice that if *C* is a vertex cover of a graph *G*, then $V(G) \setminus C$ is an independent set in *G*; if *Q* is a clique in *G*, then *Q* is an independent set in the complement of *G*. These well-known facts imply that if there is a heuristic for one of the problem of domination ratio at least *r(n)*, all other problems admit a heuristic of domination ratio at least *r(n)*.

MCl, MIS and MVC are somewhat different from the previous problems we have considered. Firstly, the number of feasible solutions, for an input of size *n*, depends on the actual input, and not just its size. The second difference is that the three problems do not admit polynomial-time heuristics of domination ratio at least $1/p(n)$ for any polynomial *p(n)* in *n* unless P=NP. This was proved by Gutin, Vainshtein and Yeo [19].

Because of the first difference, it is better to compare heuristics for the problems using the blackball number rather than domination number. Since a heuristic for MVC can be easy transformed into a heuristic for the other two problems, we restrict ourselves only to MVC heuristics.

The *incremental deletion heuristic* starts with an arbitrary permutation π of vertices of *G* and an initial solution *S=V(G)*. We consider each vertex of *G* in turn (according to π), deleting it from *S* if the resulting subset remains a (feasible) solution. A seemingly better heuristic for MVC is obtained by ordering the vertices by degree (lower degrees first), and then applying the incremental deletion heuristic. We call it the *increasing-degree deletion heuristic*. The well-known *maximal matching heuristic* constructs a maximal matching *M* and outputs both end-vertices of all edges in *M* as a solution. Berend, Skiena and Twitto [10] proved that the incremental deletion heuristic (increasing-degree deletion heuristic, maximal matching heuristic) is of blackball number $2^{n-1} - n$ (of blackball number larger than $2 - \epsilon)^n$ for each $\epsilon > 0$, of blackball number approximately 1.839^n). Clearly, the maximal matching heuristic is the best among the three heuristics from the DA point of view.

Quadratic Assignment Problem

The *Quadratic Assignment Problem* (*QAP*) can be formulated as follows. We are given two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of integers. Our aim is to find a permutation π of $\{1, 2, \dots, n\}$ that minimizes the sum

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \,.$$

Gutin and Yeo [23] described a better-than-average heuristic for QAP and proved that the heuristic is of domination number at least $n!/\beta^n$ for each $\beta > 1$. Moreover, the domination number of the heuristic is at least $(n-2)!$ for every prime power n. These results were obtained using a group-theoretical approach.

See also

► Traveling Salesman Problem

References

1. Ahuja RK, Ergun Ö, Orlin JB, Punnen AP (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123:75–102
2. Alon N, Gutin G, Krivelevich M (2004) Algorithms with large domination ratio. J Algorithms 50:118–131
3. Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M (1999) Complexity and Approximation. Springer, Berlin
4. Balas E, Saltzman MJ (1991) An algorithm for the three-index assignment problem. Oper Res 39:150–161
5. Bang-Jensen J, Gutin G (2000) Digraphs: Theory, Algorithms and Applications. Springer, London
6. Bang-Jensen J, Gutin G, Yeo A (2004) When the greedy algorithm fails. Discret Optim 1:121–127
7. Ben-Arieh D, Gutin G, Penn M, Yeo A, Zverovitch A (2003) Transformations of Generalized ATSP into ATSP: experimental and theoretical study. Oper Res Lett 31:357–365
8. Bendall G, Margot F (2006) Greedy Type Resistance of Combinatorial Problems. Discret Optim 3:288–298
9. Berge C (1958) The Theory of Graphs. Methuen, London
10. Berend B, Skiena SS, Twitto Y, Combinatorial dominance guarantees for heuristic algorithms. ACM Trans Algorithm (to appear)
11. Cornuejols G, Fisher ML, Nemhauser GL (1977) Location of bank accounts to optimize float; an analytic study of exact and approximate algorithms. Manag Sci 23:789–810
12. Deineko VG, Woeginger GJ (2000) A study of exponential neighbourhoods for the traveling salesman problem and the quadratic assignment problem. Math Prog Ser A 87:519–542
13. Ghosh D, Goldengorin B, Gutin G, Jäger G (2007) Tolerance-based greedy algorithms for the traveling salesman problem. Commun DQM 41:521–538
14. Glover F, Gutin G, Yeo A, Zverovich A (2001) Construction heuristics for the asymmetric TSP. Eur J Oper Res 129: 555–568
15. Gutin G, Goldengorin B, Huang J (2006) Worst Case Analysis of Max-Regret, Greedy and Other Heuristics for Multidimensional Assignment and Traveling Salesman Problems. Lect Notes Comput Sci 4368:214–225
16. Gutin G, Jensen T, Yeo A (2006) Domination analysis for minimum multiprocessor scheduling. Discret Appl Math 154:2613–2619
17. Gutin G, Koller A, Yeo A (2006) Note on Upper Bounds for TSP Domination Number. Algorithm Oper Res 1:52–54
18. Gutin G, Vainshtein A, Yeo A (2002) When greedy-type algorithms fail. Unpublished manuscript
19. Gutin G, Vainshtein A, Yeo A (2003) Domination Analysis of Combinatorial Optimization Problems. Discret Appl Math 129:513–520
20. Gutin G, Yeo A (2002) Polynomial approximation algorithms for the TSP and the QAP with a factorial domination number. Discret Appl Math 119:107–116
21. Gutin G, Yeo A (2002) Anti-matroids. Oper Res Lett 30: 97–99
22. Gutin G, Yeo A (2005) Domination Analysis of Combinatorial Optimization Algorithms and Problems. In: Golumbic M, Hartman I (eds) Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications. Springer, New York, pp 152–176
23. Gutin G, Yeo A, Zverovitch A (2002) Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discret Appl Math 117:81–86
24. Gutin G, Yeo A, Zverovitch A (2002) Exponential Neighborhoods and Domination Analysis for the TSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 223–256
25. Johnson DS, Gutin G, McGeoch LA, Yeo A, Zhang X, Zverovitch A (2002) Experimental Analysis of Heuristics for ATSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 445–487
26. Johnson DS, McGeoch LA (1997) The traveling salesman problem: A case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local Search in Combinatorial Optimization. Wiley, Chichester, pp 251–310
27. Johnson DS, McGeoch LA (2002) Experimental Analysis of Heuristics for STSP. In: Gutin G, Punnen AP (eds) The Traveling Salesman Problem and its Variations. Kluwer, Dordrecht, pp 369–443
28. Kise H, Ibaraki T, Mine H (1979) Performance analysis of six approximation algorithms for the one-machine maximum

lateness scheduling problem with ready times. J Oper Res Soc Japan 22:205–223
29. Punnen AP, Margot F, Kabadi SN (2003) TSP heuristics: domination analysis and complexity. Algorithmica 35:111–127
30. Robertson AJ (2001) A set of greedy randomized adaptive local search procedure implementations for the multidimentional assignment problem. Comput Optim Appl 19:145–164
31. Rublineckii VI (1973) Estimates of the accuracy of procedures in the Traveling Salesman Problem. Numer Math Comput Tech 4:18–23 (in Russian)
32. Sarvanov VI (1976) The mean value of the functional of the assignment problem. Vestsi Akad Navuk BSSR, Ser Fiz-Mat Navuk 2:111–114 (in Russian)
33. Zemel E (1981) Measuring the quality of approximate solutions to zero-one programming problems. Math Oper Res 6:319–332

Duality Gaps in Nonconvex Optimization

PANOS PARPAS, BERÇ RUSTEM
Department of Computing, Imperial College, London, UK

MSC2000: 90B50, 78M50

Article Outline

Abstract

Duality gaps in optimization problems arise because of the nonconvexities involved in the objective function or constraints. The Lagrangian dual of a nonconvex optimization problem can also be viewed as a two-person zero-sum game. From this viewpoint, the occurrence of duality gaps originates from the order in which the two players select their strategies. Therefore, duality theory can be analyzed as a zero-sum game where the order of play generates an asymmetry. One can conjecture that this asymmetry can be eliminated by allowing one of the players to select strategies from a larger space than that of the finite-dimensional Euclidean space. Once the asymmetry is removed, then there is zero duality gap. The aim of this article is to review two methods by which this process can be carried out. The first is based on randomization of the primal problem. The second extends the space from which the dual variables can be selected. Duality gaps are important in mathematical programming and some of the results reviewed here are more than 50 years old, but only recently methods have been discovered to take advantage of them. The theory is elegant and helps appreciate the game-theoretic origins of the dual problem and the role of Lagrange multipliers.

Background

We discuss how duality gaps arise, and how they can be eliminated in nonconvex optimization problems. A standard optimization problem is stated as follows:

$$\begin{aligned} \min \quad & f(x)\,, \\ & g(x) \le 0\,, \\ & x \in X\,, \end{aligned} \tag{1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ and $g\colon \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be smooth and nonconvex. The feasible region of (1) is denoted by $\mathcal{F}$, and it is assumed to be nonempty and compact. X is some compact convex set.

In order to understand the origins of duality in mathematical programming, consider devising a strategy to determine whether a point, say y, is the globally optimal solution of (1). Such a strategy can be concocted as follows: if $f(y)$ is the global solution of (1) then the following system of inequalities

$$\begin{aligned} & f(x) < f(y)\,, \\ & g(x) \le 0\,, \\ & x \in X \end{aligned} \tag{2}$$

will not have a solution. We can reformulate (2) in a slightly more convenient framework. Indeed, suppose that there exist m positive scalars λ_i, $i = 1, \dots, m$, such that

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) < f(y) \tag{3}$$

has no solution. Then (2) does not have a solution either. The left-hand side of (3) is called the Lagrangian function associated with (1). It is clear from the discussion above that the Lagrangian can be used to answer questions about the optimal solutions of (1). The usefulness of the dual function emanates from the following duality observation: Let f^* be the optimal objective function value of (1), and let $L\colon \mathbb{R}^m \to \mathbb{R}$ be defined as follows:

$$L(\lambda) = \inf_{x \in X} L(x, \lambda)\,.$$

Then it is easy to prove that:

$$\sup_{\lambda \geq 0} L(\lambda) \leq f^*\,. \tag{4}$$

This result is known as the weak duality theorem, and it is valid with a quite general set of assumptions. The strong duality theorem asserts that if f and g are convex, $f^* > -\infty$, and the interior of $\mathcal{F}$ is not empty, then

$$\sup_{\lambda \geq 0} L(\lambda) = f^*\,.$$

Proofs of the weak and strong duality theorems can be found in [1,10].

Game Theory Interpretation

There is an interesting relationship between (1) and the following optimization problem:

$$\sup_{\lambda \geq 0} \inf_{x \in X} \; f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)\,. \tag{5}$$

We refer to (1) as the primal problem, while (5) is referred to as the Lagrangian dual. The λ's that appear in (5) are called the Lagrange multipliers (or dual variables).

It is interesting to note that (1) can equivalently be restated as follows:

$$\inf_{x \in X} \sup_{\lambda \geq 0} \; f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)\,. \tag{6}$$

The relationship between (6) (or (1)) and (5) can be analyzed as a two-person zero-sum game. In this game player A chooses the x variables, and player B chooses the λ variables. If player A chooses x', and player B chooses λ', then player A pays $L(x',\lambda')$ to player B. Naturally, player A wishes to minimize this quantity, while player B attempts to maximize it.

In game theory equilibria play an important role. An equilibrium, in the present context means a point from which no player will gain by a unilateral change of strategy. For the game outlined above an equilibrium point (x^*, λ^*) must satisfy

$$\begin{aligned} L(x, \lambda^*) &\geq L(x^*, \lambda^*) \\ &\geq L(x^*, \lambda) \quad \forall x \in X, \quad \forall \lambda \in \mathbb{R}^m_+\,. \end{aligned} \tag{7}$$

A point satisfying the preceding equation is also known as a saddle point of L. To see that (7) is an equilibrium point we argue as follows: Given that player A wishes to minimize the amount paid to player B, then it is obvious that if player B chooses λ^* and player A selects anything other than x^*, player A will be worse off. Similarly, if player A chooses x^*, then if player B chooses anything other than λ^*, then player B will be worse off.

By the strong duality theorem, we know that the game has an equilibrium point under convexity assumptions. For the general case, insight can be obtained by interpreting (5) and (6) as two different games. A saddle point will exist if the optimal values of the two games are equal.

Our next task is to interpret (5) and (6) as games. Indeed consider the following situation: Player A chooses a strategy first, and then player B chooses a strategy. Thus, player B already knows the strategy that player A has chosen. As a result player B will have an advantage. Player A will argue as follows: "If I choose x, then player B will choose $\sup_{\lambda \geq 0} L(x, \lambda)$, therefore I had better choose the strategy that will minimize my losses." In other words player A will choose the optimal strategy given by solving (6).

Now consider the same game, but with the order of play reversed, i. e., player B chooses first, and player A second. Then applying the rules of rational behavior (as above), we see that player B will select the λ that solves (5).

Consequently, duality gaps originate from the order in which the two players select their strategies. In the next section we see how this asymmetry can be eliminated by allowing one of the players to select strategies from a larger space than that of the finite-dimensional

Euclidean space. Once the asymmetry is eliminated, then there is zero duality gap.

Methods

As argued above, the player that chooses first is disadvantaged, since the other player can adjust. In this section we discuss two methods in which this asymmetry in the order of play can be eliminated. Both methods were proposed early in the history of mathematical programming. The first method proceeds by randomization (increasing the powers of player A). It is difficult to say who suggested this strategy first. Since the origins of the idea emanate from mixed strategies in game theory, one could argue that the idea was first suggested by Borel in the 1920s [14]. A modern proof can be found in [2]. The second method allows player B to select the dual variables from a larger space. This idea seems to have been suggested by Everett [3], and then by Gould [6]. A review can be found in [11]. Algorithms that attempt to reduce the duality gap appeared in [4,5,7,8,9,12].

Randomization

Assume that player A chooses first, then the game can be described by

$$P^* = \inf_{x\in X} \sup_{\lambda\geq 0} L(x,\lambda)\,,$$

and in general by

$$P^* \geq D^* = \sup_{\lambda\geq 0} \inf_{x\in X} L(x,\lambda)\,.$$

Player A has a handicap since player B will choose a strategy knowing what player A will do. In order to avoid having a duality gap, we consider giving more flexibility to player A. We thus allow player A to choose strategies from $\mathcal{M}(X)$, where $\mathcal{M}(X)$ denotes the space of probability measures on $\mathcal{B}$ (the σ-field generated by X). Player A will therefore choose a strategy by solving

$$\begin{aligned} P^* = \inf_{\mu\in\mathcal{M}(X)} & \int_X f(x)\,\mathrm{d}\mu(x) \\ & \int_X g(x)\,\mathrm{d}\mu(x) \leq 0 \\ & \int_X \mathrm{d}\mu(x) = 1\,. \end{aligned} \tag{8}$$

Equivalently:

$$\begin{aligned} P^* = \inf_{\mu\in\mathcal{M}(X)} \sup_{\lambda\geq 0} & \int_X f(x)\,\mathrm{d}\mu(x) \\ & + \sum_{i=1}^m \lambda_i \int_X g(x)\,\mathrm{d}\mu(x) + \lambda_0 \left(\int_X \mathrm{d}\mu(x) - 1\right)\,. \end{aligned}$$

The dual of (8) is given by

$$\begin{aligned} D^* = \sup_{\lambda\geq 0} \inf_{\mu\in\mathcal{M}(X)} & \int_X f(x)\,\mathrm{d}\mu(x) \\ & + \sum_{i=1}^m \lambda_i \int_X g(x)\,\mathrm{d}\mu(x) + \lambda_0 \left(\int_X \mathrm{d}\mu(x) - 1\right)\,. \end{aligned}$$

Then it can be shown that $P^* = D^*$. The proof is beyond the scope of this article; it can be found in [2].

Functional Lagrange Multipliers

We now consider the case where player B chooses first. From the previous section, it follows that player B will choose a strategy according to:

$$D^* = \sup_{\lambda\geq 0} \inf_{x\in X} L(x,\lambda)\,. \tag{9}$$

We have already pointed out that the following holds:

$$D^* \leq \inf_{x\in X} \sup_{\lambda\geq 0} L(x,\lambda)\,.$$

In order for the preceding equation to hold as an equality, without any convexity assumptions, we consider increasing the space of available strategies of B. This was suggested in [3,6]. The exposition here is based on [11]. Let $\mathcal{H}$ denote all the feasible right-hand sides for (1):

$$\mathcal{H} = \{b\in\mathbb{R}^m \mid \exists x\in X\colon g(x)\leq b\}\,.$$

Let $\mathcal{D}$ denote the following set of functions:

$$\mathcal{D} = \{z\colon \mathbb{R}^m\to\mathbb{R} \mid z(d_1)\leq z(d_2),\ \text{if}\, d_1\leq d_2,\ \forall d_1,d_2\in\mathcal{H}\}\,.$$

The following dual can be defined using the concepts above:

$$\begin{aligned} D^* = \sup_z\, & z(0) \\ & z(g(x))\leq f(x) \quad \forall x\in X \quad z\in\mathcal{D}\,. \end{aligned} \tag{10}$$

The dual in (10) is different from the type of duals that we have been discussing in this article. If, however, we assume that $c + \mathcal{D} \subset \mathcal{D}$, then it was shown in [11] that (10) is equivalent to the following:

$$D^* = \sup_{z \in \mathcal{D}} \inf_{x \in X} f(x) + z(g(x))$$

dual problem. A proof that the duality gap between (10) and (1) is zero can be found in [11].

Conclusions

We have discussed Lagrangian duality, and the existence of duality gaps from a game-theoretic viewpoint. We have discussed two ways in which duality gaps can be eliminated. The first is randomization and the second is the use of functional Lagrange multipliers. Unfortunately none of the two methods are immediately applicable to real-world problems. However, for certain classes of problems the functional Lagrange multiplier approach can be useful. It was shown in [13] that if the original problem involves the optimization of polynomial functions, and if the Lagrange multipliers are allowed to be themselves polynomials then there will be no duality gap. Unlike the general case discussed in this article, polynomial Lagrange multipliers can be manipulated numerically. This approach can potentially help develop efficient algorithms for a large class of problems.

References

1. Bertsekas DP (1999) Nonlinear Programming, 2nd edn. Athena Scientific, Belmont
2. Ermoliev Y, Gaivoronski A, Nedeva C (1985) Stochastic optimization problems with incomplete information on distribution functions. SIAM J Control Optim 23(5):697–716
3. Everett H (1963) Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Oper Res 11:399–417
4. Floudas CA, Visweswaran V (1990) A global optimization algorithm (gop) for certain classes of nonconvex nlps: I. theory. Comput Chem Eng 14(12):697–716
5. Floudas CA, Visweswaran V (1993) A primal-relaxed dual global optimization approach. J Optim Theory Appl 78(2): 187–225
6. Gould FJ (1972) Nonlinear duality theorems. Cahiers Centre Études Recherche Opér 14:196–212
7. Liu WB, Floudas CA (1993) A remark on the GOP algorithm for global optimization. J Global Optim 3(4):519–521
8. Liu WB, Floudas CA (1996) A generalized primal-relaxed dual approach for global optimization. J Optim Theory Appl 90(2):417–434
9. Rajasekaran S, Pardalos PM, Reif JH, Rolim J (eds) (2001) Combinatorial Optimization. In: Handbook of randomized computing, vol I, II, vol 9. Kluwer, Dordrecht
10. Rockafellar RT (1997) Convex analysis. Princeton Landmarks in Mathematics. Princeton Univ. Press, Princeton
11. Tind J, Wolsey LA (1981) An elementary survey of general duality theory in mathematical programming. Math Programm 21(3):241–261
12. Visweswaran V, Floudas CA (1990) A global optimization algorithm (gop) for certain classes of nonconvex nlps: II. applications of theory and test problems. Comput Chem Eng 14(12):1417–1434
13. Waki H, Kim S, Kojima M, Muramatsu M (2006) Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. SIAM J Optim 17(1):218–242
14. Weintraub ER (ed) (1992) Toward a history of game theory. Duke University Press, Durham, Annual supplement to History of Political Economy, vol 24

Duality in Optimal Control with First Order Differential Equations

SABINE PICKENHAIN
Brandenburg Technical University Cottbus,
Cottbus, Germany

MSC2000: 49K05, 49K10, 49K15, 49K20

Article Outline

Keywords

Control problems; First order partial differential equations; Duality theory; Necessary and sufficient optimality conditions

Consider the following optimal control problem with first order ordinary or partial differential equations:

$$\text{(P)} \quad \min J(x,u) = \int_{\Omega} r(t, x(t), u(t))\, dt$$

subject to functions $(x, u) \in W_p^{1,n}(\Omega) \times L_p^{\nu}(\Omega)$, fulfilling

- the *state equations*

$$x^i_{t_\alpha}(t) = g^i_\alpha(t, x(t), u(t)) \quad \text{a.e. on } \Omega,$$
$$(\alpha = 1, \dots, m; \quad i = 1, \dots, n);$$

- the *control restrictions*

$$u(t) \in U \subseteq \mathbb{R}^\nu \quad \text{a.e. on } \Omega;$$

- the *state constraints*

$$x(t) \in \overline{G(t)} \subseteq \mathbb{R}^n \quad \text{on } \overline{\Omega};$$

- and the *boundary conditions*

$$x(s) = \varphi(s) \quad \text{on } \partial\Omega.$$

The data of problem (P) satisfies the following hypothesis:

H1) For $m = 1$ we have $1 \le p \le \infty$, for $m \ge 2$ we have $m < p < \infty$.

H2) The sets Ω and

$$X := \{(t, \xi) \in \mathbb{R}^m \times \mathbb{R}^n :\ t \in \Omega,\ \xi \in G(t)\}$$

are strongly Lipschitz domains in the sense of C.B. Morrey and S. Hildebrandt [6]; the set U is closed.

H3) The functions r, r_ξ, g^i_α, $(g^i_\alpha)_\xi$, φ are continuous with respect to all arguments.

H4) The set of all admissible pairs (x, u), denoted by $\mathcal{Z}$, is nonempty.

The characterization of optimal solutions of special variational problems by dual or complementary problems has been well known in physics for a long time, e. g.,

- in elasticity theory, the principle of the minimum of potential energy (Dirichlet's principle) and the principle of tension (Castigliano's principle) are dual or complementary to each other.
- in the theory of electrostatic fields, the principle of the minimum of potential energy and the Thomson (Lord Kelvin) principle are dual problems.

A first systematic approach to duality for special problems in calculus of variations was given by K. Friedrichs ([4], 1928). In the 1950s and 1960s, this concept was extended by W. Fenchel [3], J.-J. Moreau, R.T. Rockafellar [19,20] and I. Ekeland and R. Temam [2] to larger classes of variational and control problems. Basing on Legendre transformation (or Fenchel conjugation), it was proved to be a suitable tool to handle convex problems.

Nonconvex problems (P) require an extended concept of duality. The construction of R. Klötzler, given in 1977 [7], can be regarded as a further development of Hamilton–Jacobi field theory.

Construction of a Dual Problem

In a very general setting, a problem (D) of maximization of an (extended real-valued) functional L over an arbitrary set $S \neq \emptyset$ is said to be a *dual problem* to (P) if the *weak duality relation*

$$\sup(\text{D}) \le \inf(\text{P})$$

is satisfied.

The different notions of duality given in the introduction can be embedded into the following construction scheme:

1 The set of admissible pairs $(x, u) = z \in \mathcal{Z}$ is represented by the intersection of two suitable nonempty sets $\mathcal{Z}_0$ and $\mathcal{Z}_1$.

2 For an (extended real-valued) functional $\Phi : \mathcal{Z}_0 \times S_0 \to \overline{\mathbf{R}}$ the *equivalence relation*

$$\inf_{z \in \mathcal{Z}} J(z) = \inf_{z \in \mathcal{Z}_0} \sup_{S \in S_0} \Phi(z, S),$$

holds.

3 Assuming $L_0(S) := \inf_{z \in Z_0} \Phi(z, S)$, each problem

$$\text{(D)} \qquad \begin{cases} \max & L(S) \\ \text{s.t.} & S \in S_1 \subseteq S_0 \end{cases}$$

is a (weak) dual problem to (P) if $L(S) \le L_0(S)$ for all $S \in S_1$.

The proof of the *weak duality relation* results from the well-known inequality

$$\inf_{z\in\mathcal{Z}_0}\sup_{S\in S_0}\Phi(z,S)\geq\sup_{S\in S_0}\inf_{z\in\mathcal{Z}_0}\Phi(z,S).$$

Fenchel–Rockafellar Duality

In accordance with [2], we transform (P) into a general variational problem:

$$(\mathrm{V})\quad\begin{cases}\min & \int_\Omega l(t,x(t),x_{t_\alpha}(t))\,dt\\ \text{s.t.} & x\in\mathcal{X}\end{cases}$$

where $l\colon\ \mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^{nm}\to\overline{\mathbb{R}}$ is given by

$$l(t,\xi,w):=\begin{cases}\inf\{r(t,\xi,v):\\ \quad v\in U\text{ with}\\ \quad w=g(t,\xi,v)\}, & (t,\xi)\in\overline{X},\\ \infty & \text{else.}\end{cases}$$

and $\mathcal{X}=\{x\in W_p^{1,n}(\Omega):x(s)=\varphi(s)\text{ on }\partial\Omega\}$.

Then (P) is called *convex* if (V) is convex in the sense of [2, p. 113]. In this case both problems are equivalent [15]. The Fenchel-dual problem is obtained by the following settings in the above construction scheme:

1) $\mathcal{Z}_0=\{z=(x,u)\in W_p^{1,n}(\Omega)\times L_p^{1,\nu}(\Omega):$
$u(t)\in U$ a.e. on Ω,
$x(t)\in\overline{G(t)}$ on Ω,
$x(s)=\varphi(s)$ on $\partial\Omega\}$,
$\mathcal{Z}_1=\{z=(x,u)\in W_p^{1,n}(\Omega)\times L_p^{1,\nu}(\Omega):$
$x_{t_\alpha}^i(t)=g_\alpha^i(t,x(t),u(t))$ a.e. on Ω,
$\alpha=1,\dots,m;\ i=1,\dots,n\}$.

2) $\mathcal{S}_0=L_q^{n(1+m)}(\Omega)\ (p^{-1}+q^{-1}=1)$, Φ is the classical *Lagrange functional*,

$$\Phi(z,S)=J(z)+\sum_{i,\alpha}\left\langle x_{t_\alpha}^i-g(\cdot,x,u),\ y_i^\alpha\right\rangle,$$

where $\langle\cdot,\cdot\rangle$ is the bilinear canonical pairing over $L_p(\Omega)\times L_p^*(\Omega)$, [15]. By use of the *Hamiltonian* of (P),

$$\mathcal{H}\colon\ \mathbb{R}^1\times\mathbb{R}^n\times\mathbb{R}^{nm}\to\overline{\mathbb{R}},$$
$$\mathcal{H}(t,\xi,\eta)=\sup\{H(t,\xi,v,\eta)\colon\ v\in U\}$$

with

$$H(t,\xi,v,\eta)=-r(t,\xi,v)+\sum_{i,\alpha}\eta_i^\alpha g_\alpha^i(t,\xi,v)$$

it can be formulated as follows [15]:

$$(\mathrm{D_R})_q\quad\begin{cases}\max\\ \quad\left[-\int_\Omega\left(\sup_{\xi\in G(t)}[\mathcal{H}(t,\xi,-y(t))\right.\right.\\ \quad\left.-\,y_0(t)^\top\xi]\right)dt\\ \quad-\sup_{\xi\in X}\left[\int_\Omega\left(y_0(t)^\top\zeta(t)\right.\right.\\ \quad\left.\left.\left.+\sum_{i,\alpha}y_i^\alpha(t)\zeta_{t_\alpha}^i(t)\right)dt\right]\right]\\ \text{s.t.}\quad(y_0,y)\in L_q^{n(1+m)}(\Omega).\end{cases}$$

Duality in the Sense of Klötzler

The duality in the sense of Klötzler is realized by the following settings in the general construction scheme [14]:

1) $\mathcal{Z}_0$ and $\mathcal{Z}_1$ are chosen as before.
2) $S_0=W_q^{1,n}(\overline{X})$, and Φ is an extended Lagrange functional,

$$\Phi(z,S)=J(z)+\sum_{i,\alpha}x_{t_\alpha}^i-g_\alpha^i(\cdot,x,u),S_{\xi_\alpha}^i(\cdot,x),$$

where $\langle\cdot,\cdot\rangle$ is again the bilinear canonical pairing over $L_p(\Omega)\times L_p^*(\Omega)$.

By use of Gauss' theorem, the dual problem reads as follows [8]:

$$(\mathrm{D_K})_q\quad\begin{cases}\max & \int_{\partial\Omega}S(s,\zeta(s))\mathfrak{n}(s)\,do(s)\\ \text{s.t.} & S\in S_1,\end{cases}$$

where

$$S_1:=\left\{S\in S_0\colon\ \begin{array}{c}\sum_{\alpha=1}^m S_{t_\alpha}^\alpha(t,\xi)\\ +\mathcal{H}(t,\xi,S(t,\xi))\leq 0\\ \text{a.e. on }\overline{X}\end{array}\right\},$$

$\mathfrak{n}(\cdot)$ is the exterior unit normal vector to $\partial\Omega$.

In this way we can characterize minimizers of (P) in terms of solutions of the *Hamilton–Jacobi inequality* or of the *Hamilton–Jacobi equation*. Since classical solutions of the latter equation may fail to exist, on the one hand techniques were developed to construct generalized solutions of this equation (viscosity solutions [11], generalized solutions involving the Clarke generalized gradient [1] or lower Dini derivatives [24]). On the other hand, optimization techniques for parametric problems in finite-dimensional spaces are used to minimize the defect in the Hamilton–Jacobi inequality and to get sufficient conditions for (local) optimality [16,17,27,28,29].

Bidual Problems, Generalized Flows, Relaxed Controls

Duality allows to associate the bidual problem with a *flow problem* or a *relaxed control problem.* $(\mathrm{D_K})_q$, interpreted as an infinite linear programming problem, [13], has a dual problem again, which can be identified as a generalized flow problem in the sense of L.C. Young [26]: Assuming compactness of the control set U, one obtains the bidual problem $(\mathrm{D_K})_q^* = (\mathrm{F})$:

$$(\mathrm{F}) \quad \begin{cases} \min & \int_D r(t,\xi,v)\, d\nu(t,\xi,v) \\ \text{s.t.} & \nu \in \mathfrak{N}_D , \end{cases}$$

with

$$\mathfrak{N}_D := \Big\{ \nu \in \mathcal{R}_D : \int_{\partial\Omega} \sum_\alpha \psi^\alpha(s,\zeta(s)) \mathrm{n}_\alpha(s)\, do(s) = \int_D \Big(\psi^\alpha_{t_\alpha}(t,\xi) + \sum_{i,\alpha} \psi^\alpha_{\xi_i}(t,\xi) g^i_\alpha(t,\xi,v) \Big) d\nu(t,\xi,v),\ \forall \psi \in C^{1,m}(\overline{X}) \Big\} ,$$

where $\mathcal{R}_D$ is the set of all nonnegative Radon measures on $D := \overline{X} \times U$. $\mathfrak{N}_D$ contains special measures

$$d\nu(t,\xi,v) = d\delta_{x(t)}\, d\mu_t(v)\, dt,$$

with $\mu = \{\mu_t : t \in \Omega\} \in \mathfrak{M}_U$, $\mathfrak{M}_U$ is a *regular family of probability measures*, concentrated on U, [5], and δ_ξ are Dirac measures concentrated on the point $\xi \in \overline{G(t)}$.

Thus the relaxed control problem

$$\overline{(\mathrm{P})} \quad \begin{cases} \min & \int_\Omega \int_U r(t,x(t),v)\, d\mu_t(v)\, dt \\ \text{s.t.} & (x,\mu) \in W_p^{1,n}(\Omega) \times \mathfrak{M}_U , \end{cases}$$

satisfying $x(t) \in \overline{G(t)}$ and fulfilling the following variational equation for all $\psi \in C^{1,m}(\overline{X})$,

$$\int_{\partial\Omega} \sum_\alpha \psi^\alpha(s,\zeta(s)) \mathrm{n}_\alpha(s)\, do(s) = \int_\Omega \Big[\sum_\alpha \psi^\alpha_{t_\alpha}(t,x(t)) + \sum_{i,\alpha} \psi^\alpha_{\xi_i}(t,x(t)) \int_U g^i_\alpha(t,x(t),v)\, d\mu_t(v) \Big] dt$$

has the embedding (F), and

$$\sup (\mathrm{D_K})_\infty = \inf (\mathrm{F}) \le \inf \overline{(\mathrm{P})}$$

holds, [13].

Strong Duality Results

The property of strong duality between (P) and (D) is defined by the equation

$$\sup (\mathrm{D}) = \inf (\mathrm{P}),$$

and this common value is called *settle-value* of Φ.

Case A

Control problems with single integrals and ordinary differential equations, $\Omega = [0, T]$.

For convex problems (P),

$$\sup (\mathrm{D_R})_q = \min (\mathrm{P})$$

holds, [2]. Moreover, in $(\mathrm{D_R})_q$ the optimal solution (y_0^*, y^*) exists and fulfills

$$y^* \in W_p^{1,n}(\Omega)$$

and

$$\frac{d}{dt} y^*(t) = y_0^*(t) \text{ a.e. on } (0,T),$$

[15]. Both dual problems, $(\mathrm{D_R})_q$ and $(\mathrm{D_K})_q$, coincide, if in $(\mathrm{D_K})_q$ a linear setting

$$S(t,\xi) = a(t) + \xi^\top y(t)$$

is chosen, [14].

For nonconvex control problems with compact U is was shown by different techniques, that (P) as well as

$\overline{(\mathrm{P})}$ possess a same dual problem, [12,22,23], and strong duality

$$\sup\,(\mathrm{D_K})_\infty = \min\,\overline{(\mathrm{P})}$$

holds. The variational equation appearing in $\overline{(\mathrm{P})}$ is in this case equivalent to the *generalized state equations*

$$\frac{d}{dt}x^i(t) = \int_U g^i(t, x(t), v)\, d\mu_t(v) \text{ a.e. on } (0, T)$$

with the *boundary conditions*

$$x(s) = \varphi(s) \quad \text{for } s = 0,\ s = T.$$

The question of existence of a solution of the dual problem $(\mathrm{D_K})_\infty$ was discussed in [21].

Case B

Control problems with multiple integrals and first order partial differential equations.

As before, for convex problems

$$\sup\,(\mathrm{D_R})_q = \min\,(\mathrm{P})$$

holds. The equivalence of $(\mathrm{D_R})_q$ and $(\mathrm{D_K})_q$ is lost in general. Results concerning strong duality between (D) and $\overline{(\mathrm{P})}$ in the nonconvex case are largely missing.

Sufficient Optimality Conditions

First- and second order sufficient optimality conditions for global minimizers can be derived by means of duality. In the general concept, $(x^*, u^*) \in \mathcal{Z}$ is a global minimizer of (P) if

$$J(x^*, u^*) = \inf_{z\in Z_0} \sup_{S\in S_0} \Phi(z, S)$$
$$= \max_{S\in S_0} \inf_{z\in Z_0} \Phi(z, S)$$

and it exists an $S^* \in \mathcal{S}_1$ with

$$L_0(S^*) = \max_{S\in S_1} L(S) = \max_{S\in S_0} L_0(S).$$

Following the concept of Klötzler, these equations are satisfied if and only if for $S^* \in W^{1,n}_\infty(\overline{X})$ the following conditions are fulfilled:

a) the *Hamilton–Jacobi inequality*

$$\Lambda(t, \xi) := \sum_\alpha S^{*\alpha}_{t_\alpha}(t, \xi)$$
$$+ \mathcal{H}(t, \xi, S^*_\xi(t, \xi)) \le 0 \text{ on } \overline{X};$$

b) the *Hamilton–Jacobi equation*

$$\sum_\alpha S^{*\alpha}_{t_\alpha}(t, x^*(t))$$
$$+ \mathcal{H}(t, x^*(t), S^*_\xi(t, x^*(t))) = 0 \text{ on } \overline{\Omega};$$

c) the *maximum condition*

$$\mathcal{H}(t, x^*(t), S^*_\xi(t, x^*(t)))$$
$$= \mathcal{H}(t, x^*(t), u^*(t), S^*_\xi(t, x^*(t))) \text{ a.e. on } \Omega.$$

From conditions a) and b) follows that $x^*(t)$ must be a global minimizer of the parametric optimization problem

$$(\mathrm{P})_t \quad \begin{cases} \max & \Lambda(t, \xi) \\ \text{s.t.} & \xi \in \overline{G(t)} \end{cases}$$

with parameter $t \in \overline{\Omega}$. For this last problem $(\mathrm{P})_t$ first- and second order sufficient optimality conditions can be derived with the quadratic setting

$$S^{*\alpha}(t, \xi) = a^\alpha(t) + y^{\alpha\top}(t)(\xi - x(t))$$
$$+ \frac{1}{2}(\xi - x^*(t))Q^\alpha(t)(\xi - x^*(t))$$

in the dual problem $(\mathrm{D_K})_\infty$, where $y^\alpha \in W^{1,n}_\infty(\Omega)$ and $Q^\alpha \in W^{1,nn}_\infty(\Omega)$ symmetric.

The ideas, mentioned above, can be used for identifying *strong local minimizers* of (P) too. In this case $\overline{X}$ is to be replaced by

$$\overline{X}_\varepsilon = \overline{X}$$
$$\cap \left\{(t, \xi) \in \mathbb{R}^{n+1}\colon\ \|\xi - x(t)\| < \varepsilon,\ t \in \overline{\Omega}\right\},$$

[16,17,27,28,29]. The second order condition for $(\mathrm{P})_t$ yields a definiteness condition for a Riccati-type expression which generalizes the known theory of conjugated points in the calculus of variations in one independent variable.

Duality and Maximum Principle

Case A

Control problems with single integrals, $\Omega = [0, T]$. For convex problems (P) it can be shown that the *Pontryagin maximum principle* is not only a necessary but also a sufficient optimality condition. In this case the canonical variables in the Maximum principle solve at the same time the dual problem $(D_K)_\infty$, [15].

Case B

Control problems with multiple integrals, $\Omega \subseteq \mathbf{R}^m$, $m \geq 2$. For convex problems (P) or relaxed problems $\overline{(P)}$ a maximum principle was proved in the beginning of the 1990s, [10,18,25]. It turns out, that the canonical variables in this principle are not necessarily functions but contents or measures from $L^*_\infty(\Omega)$ or $C^*(\Omega)$. A corresponding duality theory with dual variables in these measure spaces was developed by Klötzler [9] and strong duality was shown. As before, in the convex case the canonical variables of the maximum principle solve the dual problem.

See also

- ▶ Control Vector Iteration
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Optimal Control of a Flexible Arm
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-Infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Clarke FH, Vinter RB (1983) Local optimality conditions and Lipschitz solutions to the Hamilton–Jacobi equation. SIAM J Control Optim 21:856–870
2. Ekeland I, Temam R (1976) Convex analysis and variational problems. North-Holland, Amsterdam
3. Fenchel W (1953) Convex cones, sets and functions. Lecture Notes Dept Math. Princeton Univ. Press, Princeton
4. Friedrichs K (1929) Ein Verfahren der Variationsrechnung, das Minimum eines Integrals als das Maximum eines anderen Ausdrucks darzustellen. Göttinger Nachrichten 13–20
5. Gamkrelidze RV (1978) Principles of optimal control theory. Plenum, New York
6. Hildebrandt S (1962) Über die Identität der Sobolewschen und Calkin-Morreyschen Räume. Math Ann 148:226–237
7. Klötzler R (1977) On a general conception of duality in optimal control. Proc Equadiff 4:189–196
8. Klötzler R (1980) Starke Dualität in der Steuerungstheorie. Math Nachrichten 95:253–263
9. Klötzler R (1995) Optimal transportation flows. J Anal Appl 14:391–401
10. Klötzler R, Pickenhain S (1993) Pontryagin's maximum principle for multidimensional control problems. Internat Ser Numer Math, vol 111. Birkhäuser, Basel pp 21–30
11. Lions P-L (1982) Generalized solutions of Hamilton-Jacobi equations. Res Notes Math, vol 69. Pitman, Boston, MA
12. Pickenhain S (1982) Starke Dualität bei verallgemeinerten Steuerungsproblemen. Z Anal Anwend 1(4):15–24
13. Pickenhain S (1987) Dualität bei Steuerungsproblemen mehrfacher Integrale. Z Anal Anwend 6(2):159–174
14. Pickenhain S (1988) Zum Vergleich verschiedener Dualitätsbegriffe aus der Sicht der Steuerungstheorie. Z Anal Anwend 7(3):277–285
15. Pickenhain S (1992) Beiträge zur Theorie mehrdimensionaler Steuerungsprobleme. Habilitationsschrift Univ. Leipzig
16. Pickenhain S (1992) Sufficiency conditions for weak local minima in multidimensional optimal control problems with mixed control-state restrictions. Z Anal Anwend 11(4):559–568
17. Pickenhain S, Tammer K (1991) Sufficient conditions for local optimality in multidimensional control problems with state restrictions. Z Anal Anwend 10(3):397–405
18. Pickenhain S, Wagner M (1999) Critical points in relaxed deposit problems. In: Ioffe A, Reich S, Shafrir I (eds) Calculus of Variations and Optimal Control, Technion '98. Pitman Res Notes Math Chapman and Hall/CRC, London/Boca Raton, FL, pp 217–236
19. Rockafellar RT (1970) Conjugate convex functions in optimal control and the calculus of variations. J Math Anal Appl 32:174–222
20. Rockafellar RT (1974) Conjugate duality and optimization. Regional Conf Ser, vol 16. SIAM, Philadelphia
21. Vinter RB (1983) Weakest conditions for existence of Lipschitz continous Krotov functions in optimal control theory. SIAM J Control Optim 21(2):215–234
22. Vinter RB, Levis RM (1978) The equivalence of the strong and weak formulation for certain problems in optimal control. SIAM J Control Optim 16(4):546–570

23. Vinter RB, Levis RM (1978) A nessesary and sufficient condition for optimality of dynamic programming type, making no a priori assumptions on the control. SIAM J Control Optim 16(4):571–583
24. Vinter RB, Wolenski P (1990) Hamilton-Jacobi theory for problems with data measurable in time. SIAM J Control Optim 28:1404–1419
25. Wagner M (1999) Pontryagin's maximum principle for Dieudonné-Rashevsky type problems involving Lipschitz functions. Optim:165–184 46
26. Young LC (1968) Calculus of variations and optimal control theory. W.B. Saunders, Philadelphia
27. Zeidan V (1983) Sufficient conditions for the generalized problem of Bolza. Trans Amer Math Soc 275:561–586
28. Zeidan V (1984) Extendend Jacobi sufficiency criterion for optimal control. SIAM J Control Optim 22:294–301
29. Zeidan V (1984) First- and second-order sufficient conditions for optimal control and the Calculus of Variations. Appl Math Optim 11:209–226

Duality for Semidefinite Programming

HENRY WOLKOWICZ
Department Combinatorics and Optimization,
University Waterloo, Waterloo, Canada

MSC2000: 90C30

Article Outline

Keywords

Semidefinite programming; Convex programming; Lagrangian duality; Strong and weak duality; Constraint qualification; Complementarity

Synonyms

SDP duality

Basic Properties

Consider the primal *semidefinite program*

$$\text{SDP} \quad \mu^* := \begin{cases} \min & C \bullet X \\ \text{s.t.} & \mathcal{A}X = b \\ & X \succeq 0, \end{cases}$$

where $C \bullet X = \operatorname{trace} CX$ denotes the inner product of the symmetric matrices C, X; $X \succeq Y$ denotes that the symmetric matrix $X - Y$ is positive semidefinite; and $\mathcal{A}$: $\mathcal{S}^n \to \mathbf{R}^m$ is a linear operator on the space of symmetric matrices, with adjoint $\mathcal{A}^*$. Equivalently, the linear constraint can be written using symmetric matrices A_i, $i = 1, \dots, m$, as

$$\mathcal{A}_i \bullet X = b_i \quad \text{for all } i = 1, \dots, m;$$

while the adjoint operation on $y \in \mathbf{R}^m$ is

$$\mathcal{A}^* y := \sum_{i=1}^{m} y_i A_i .$$

The *Lagrangian* function is

$$\mathcal{L}(X, y) := C \bullet X + y^\top (b - \mathcal{A}X).$$

The primal problem is equivalent to

$$\mu^* = \min_{X \succeq 0} \max_{y} \mathcal{L}(X, y) = C \bullet X + y^\top (b - \mathcal{A}X).$$

The equivalence can be seen by using the *hidden constraint* in the outer minimization problem $b - \mathcal{A}\, X = 0$, i. e. if this constraint does not hold then the inner maximum value is $+\infty$.

By interchanging the maximum and minimum and rewriting the order of terms in the Lagrangian, we get the dual problem and *weak duality*:

$$\mu^* \geq \nu^* := \max_{y} \min_{X \succeq 0} b^\top y + (C - \mathcal{A}^* y) \bullet X.$$

Using the hidden constraint in the outer maximization problem $C - \mathcal{A}^* y \succeq 0$, this becomes equivalent to

$$\text{(D)} \quad \nu^* = \begin{cases} \max & b^\top y \\ \text{s.t.} & \mathcal{A}^* y \preceq C. \end{cases}$$

The dual pair SDP and (D) look very much like a dual pair of linear programs (denoted LP) where the adjoint operator replaces the transpose and positive semidefiniteness of matrix variables replaces nonnegativity of vector variables. In fact, duality theory for SDP has a lot of similarities with that of LP: *weak duality* $\mu^* \geq \nu^*$ follows from the interchange of maximum and minimum; from this we get that unboundedness of SDP (respectively, (D)) implies infeasibility of (D) (respectively, (D)).

Weak duality illustrates one of the powerful uses of the dual program, i. e. it provides lower bounds on the optimal value of the primal program.

Other formulations of SDP provide similar duals. In fact, SDP is a special case of cone programming. Let K, L be two *convex cones*, i. e. K (and L) satisfy: the Minkowski sum $K + K \subset K$ and $\alpha\ K \subset K$ for all $\alpha \in$ **R**. Define the primal cone program as

$$(\mathrm{PC}) \quad \nu^* = \begin{cases} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}X \succeq_K b \\ & X \succeq_L 0, \end{cases}$$

where $X \succeq_L Y$ denotes $X - Y \in L$ (and similarly for K), and $\langle\cdot,\cdot\rangle$ denotes the appropriate inner product. Then the above min-max argument yields the dual cone program

$$(\mathrm{DC}) \quad \nu^* = \begin{cases} \max & \langle b, y\rangle \\ \text{s.t.} & \mathcal{A}^* y \preceq_{L^+} C \\ & y \succeq_{K^+} 0, \end{cases}$$

where $\cdot^+$ denotes taking the *polar cone*.

It is an interesting exercise to see that this elegant dual formulation works for linear programs that have mixtures of inequality and equality constraints with mixtures of free and nonnegative variables.

Strong Duality

However, unlike linear programming, *strong duality* for SDP needs a constraint qualification, e. g. strict primal feasibility (called *Slater's condition*),

$$\text{there exists a } \widehat{X} \succ 0 \text{ with } \mathcal{A}\widehat{X} = b.$$

This constraint qualification implies strong duality holds, i. e. that $\mu^* = \nu^*$ and ν^* is attained. Conversely,

$$\mathcal{A}^* y \prec_{L^+} C$$

also implies that $\mu^* = \nu^*$ but with μ^* attained. If Slater's condition does not hold, then a duality gap $\mu^* > \nu^*$ can exist, and/or the dual (or primal) optimal value may not be attained, see e. g. [10].

Example 1 If the dual is

$$(\mathrm{D}) \quad \nu^* = \begin{cases} \sup & x_2 \\ \text{s.t.} & \begin{pmatrix} x_2 & 0 & 0 \\ 0 & x_1 & x_2 \\ 0 & x_2 & 0 \end{pmatrix} \preceq \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \end{cases}$$

then the primal is

$$(\mathrm{P}) \quad \mu^* = \begin{cases} \inf & U_{11} \\ \text{s.t.} & U_{22} = 0 \\ & U_{11} + 2U_{23} = 1 \\ & U \succeq 0 \end{cases}$$

and we have the duality gap

$$\nu^* = 0 < \mu^* = 1.$$

If strong duality holds, we get the following primal-dual characterization of optimality for the dual pair X, y, with $X \succeq 0$:

- $\mathcal{A}\ X = b$ (primal feasibility);
- $\mathcal{A}^* y \preceq C$ (dual feasibility);
- $X\ (\mathcal{A}^* y - C) = 0$ (complementary slackness).

These optimality conditions provide the basis for:

i) the primal simplex method (maintain primal feasibility and complementary slackness while striving for dual feasibility);
ii) the dual simplex method (maintain dual feasibility and complementary slackness while striving for primal feasibility); and
iii) the interior point methods (maintain primal and dual feasibility while striving for complementary slackness).

Unlike the LP case, there are currently no efficient algorithms for primal or dual simplex methods for SDP; however, interior point methods have proven to be very successful. Thus we see the importance of duality for both theoretical and algorithmic purposes.

Strict Complementarity

Another example of the difference between LP and SDP arises in the complementary slackness conditions. If an optimal pair X, y exist, then in the LP case there also exists an optimal pair that satisfies strict complementarity, i. e.

$$X + (C - \mathcal{A}^* y) \succ 0,$$

where in the LP case this is a sum of nonnegative diagonal matrices, see [4,5]. However, in the SDP case, there may not exist such a strict complementary optimal pair, though the existence is generic, see [8].

Closing the Duality Gap

Both the strict complementarity and strict feasiblity, or Slater's constraint qualification, are generic, see [8]. But there are classes of problems where strong duality fails, e. g. relaxations that arise from hard combinatorial problems, e. g. [13].

One can regularize semidefinite programs and guarantee that Slater's constraint qualification holds, e. g. [2,3,12]. This involves finding the minimal face of the semidefinite cone that contains the feasible set, i. e. the so-called *minimal cone*. A numerical procedure for regularization is presented in [3]. However, this process is not computationally tractable. An equivalent approach is the extended Lagrange–Slater dual program of M. Ramana [9,10]. This provides a means of writing down a regularized program that is of polynomial size. Thus strong duality can be attained theoretically using the above techniques and exploiting the structure of specific problems. However, lack of regularity (Slater's condition) is an indication of an ill-posed problem. Thus, the question of whether regularization can be done computationally for general problems is still an open question, see e. g. [7].

Extensions

The SDPs considered above have all contained linear objectives and constraints. There is no reason to restrict SDPs to this special case. Duality for general cone programs with possible nonlinear objectives and constraints is considered in [2,3,11]. Applications for quadratic objectives SDP appear in, e. g., [1,6].

See also

- ▶ Interior Point Methods for Semidefinite Programming
- ▶ Semidefinite Programming and Determinant Maximization
- ▶ Semidefinite Programming: Optimality Conditions and Stability
- ▶ Semidefinite Programming and Structural Optimization
- ▶ Semi-infinite Programming, Semidefinite Programming and Perfect Duality
- ▶ Solving Large Scale and Sparse Semidefinite Programs

References

1. Alfakih A, Khandani A, Wolkowicz H (1997) Solving Euclidean distance matrix completion problems via semidefinite programming. Techn Report CORR Univ Waterloo 98-9, in Computational Optim and Appl
2. Borwein JM, Wolkowicz H (1981) Characterizations of optimality for the abstract convex program with finite dimensional range. J Austral Math Soc (Ser A) 30:390–411
3. Borwein JM, Wolkowicz H (1981) Regularizing the abstract convex program. J Math Anal Appl 83:495–530
4. Goldman AJ, Tucker AW (1956) Polyhedral convex cones. Linear equalities and related systems. In: Ann of Math Stud, vol 38. Princeton Univ. Press, Princeton, pp 19–40
5. Goldman AJ, Tucker AW (1956) Theory of linear programming. Linear inequalities and related systems. In: Ann of math Stud, vol 38. Princeton Univ. Press, Princeton, pp 53–97
6. Johnson CR, Kroschel B, Wolkowicz H (1998) An interior-point method for approximate positive semidefinite completions. Comput Optim Appl 9(2):175–190
7. Klerk Ede (1997) Interior point methods for semidefinite programming. PhD Thesis Delft Univ
8. Pataki G, Tuncel L (1997) On the generic properties of convex optimization problems in conic form. Techn Report CORR Dept Combinatorics and Optim Waterloo, Ont 97-16
9. Ramana MV (1993) An algorithmic analysis of multiquadratic and semidefinite programming problems. PhD Thesis Johns Hopkins Univ, Baltimore, Md
10. Ramana M, Tuncel L, Wolkowicz H (1997) Strong duality for semidefinite programming. SIAM J Optim 7(3):641–662
11. Shapiro A (1997) First and second order analysis of nonlinear semidefinite programs. Math Program 77:301–320
12. Wolkowicz H (1981) Some applications of optimization in matrix theory. Linear Alg Appl 40:101–118
13. Zhao Q, Karisch SE, Rendl F, Wolkowicz H (1998) Semidefinite programming relaxations for the quadratic assignment problem. J Combin Optim 2:71–109

Duality Theory: Biduality in Nonconvex Optimization

DAVID YANG GAO
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 49-XX, 90-XX, 93-XX

Article Outline

Keywords

Duality; Nonconvex optimization; d.c. programming; SuperLagrangian; Biduality; Clarke duality; Hamiltonian system

It is known that in convex optimization, the Lagrangian associated with a constrained problem is usually a saddle function, which leads to the classical *saddle Lagrange duality* (i. e. the *monoduality*) theory. In nonconvex optimization, a so-called superLagrangian was introduced in [1], which leads to a nice biduality theory in convex *Hamiltonian systems* and in the so-called *d.c. programming*.

SuperLagrangian Duality

Definition 1 Let $L(x, y^*)$ be an arbitrary given real-valued function on $\mathcal{X} \times \mathcal{Y}^*$.

A function $L\colon \mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ is said to be a *supercritical function* (or a ∂^+*-function*) on $\mathcal{X} \times \mathcal{Y}^*$ if it is concave in each of its arguments.

A function $L\colon \mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ is said to be a *subcritical function* (or a ∂^-*-function*) on $\mathcal{X} \times \mathcal{Y}^*$ if $-L$ is a supercritical function on $\mathcal{X} \times \mathcal{Y}^*$.

A point $(\bar{x}, \bar{y}^*)$ is said to be a *supercritical point* (or a ∂^+*-critical point*) of L on $\mathcal{X} \times \mathcal{Y}^*$ if

$$L(\bar{x}, y^*) \le L(\bar{x}, \bar{y}^*) \ge L(x, \bar{y}^*) \qquad (1)$$

holds for all $(x, y^*) \in \mathcal{X} \times \mathcal{Y}^*$.

A point $(\bar{x}, \bar{y}^*)$ is said to be a *subcritical point* (or a ∂^-*-critical point*) of L on $\mathcal{X} \times \mathcal{Y}^*$ if

$$L(\bar{x}, y^*) \ge L(\bar{x}, \bar{y}^*) \le L(x, \bar{y}^*) \qquad (2)$$

holds for all $(x, y^*) \in \mathcal{X}\times\mathcal{Y}^*$.

Clearly, a point $(\bar{x}, \bar{y}^*)$ is a subcritical point of L on $\mathcal{X} \times \mathcal{Y}^*$ if and only if it is a supercritical point of $-L$ on $\mathcal{X} \times \mathcal{Y}^*$. A supercritical function $L(x, y^*)$ is called the *superLagrangian* if it is a *Lagrange form* associated with a constrained optimization problem. $L(x, y^*)$ is called the *subLagrangian* if $-L(x, y^*)$ is a superLagrangian.

For example, the quadratic function

$$L(x, y) = axy - \frac{1}{2}bx^2 - \frac{1}{2}cy^2, \quad b, c > 0,$$

is concave for each x and y, and hence is a supercritical point function on $\mathbf{R} \times \mathbf{R}$. But $L(x, y)$ is not concave on the vector (x, y) since the Hessian matrix of L

$$D^2L(x, y) = \begin{pmatrix} -b & a \\ a & -c \end{pmatrix}$$

is not necessarily to be negative-definite for any $a \in \mathbf{R}$ and $b, c > 0$. L is a subcritical function if $b, c < 0$. But L may not be convex on (x, y) for the same reason.

Since L is a subLagrangian if and only if $-L$ is a superLagrangian, here we only consider the duality theory for the superLagrangian.

Theorem 2 (*Supercritical point*) *Let $L(x, y^*)$ be an arbitrary given function, partially Gâteaux differentiable on an open subset $\mathcal{X}_a \times \mathcal{Y}^*_a \subset \mathcal{X} \times \mathcal{Y}^*$. If $(\bar{x}, \bar{y}^*) \in \mathcal{X}_a \times \mathcal{Y}^*_a$ is either a supercritical or subcritical point of L, then $(\bar{x}, \bar{y}^*)$ is a critical point of L on $\mathcal{X}_a \times \mathcal{Y}^*_a$.*

Any critical point of a Gâteaux differentiable superLagrangian is a supercritical point. However, if $(\bar{x}, \bar{y}^*)$ is a supercritical point of L, it does not follows that L is a superLagrangian. In the d.c. programming or variational analysis of convex Hamiltonian systems, the following statements are of important theoretical value.

S1) Under certain necessary and sufficient conditions we have

$$\inf_{x \in \mathcal{X}_a} \sup_{y^* \in \mathcal{Y}^*_a} L(x, y^*) = \inf_{y^* \in \mathcal{Y}^*_a} \sup_{x \in \mathcal{X}_a} L(x, y^*). \quad (3)$$

A statement of this type is called a *superminimax theorem* and the pair $(\bar{x}, \bar{y}^*)$ is called a *superminimax point* of L on $\mathcal{X}_a \times \mathcal{Y}^*_a$.

S2) Under certain conditions, a pair $(\bar{x}, \bar{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ exists such that

$$L(x, \bar{y}^*) \le L(\bar{x}, \bar{y}^*) \ge L(\bar{x}, y^*) \quad (4)$$

holds for all $(x, y^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$. A statement of this type is called a *supercritical point theorem.*

By the fact that the maxima of $L(x, y^*)$ can be taken in either order on $\mathcal{X}_a \times \mathcal{Y}_a^*$, the equality

$$\sup_{x \in \mathcal{X}_a} \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*) = \sup_{y^* \in \mathcal{Y}_a^*} \sup_{x \in \mathcal{X}_a} L(x, y^*) \quad (5)$$

always holds. A pair $(\bar{x}, \bar{y}^*)$ which maximizes L on $\mathcal{X}_a \times \mathcal{Y}_a^*$ is called a *supermaximum point* of L on $\mathcal{X}_a \times \mathcal{Y}_a^*$.

For a given superLagrangian L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$, we let $\mathcal{X}_k \subseteq \mathcal{X}_a$ and $\mathcal{Y}_s^* \subseteq \mathcal{Y}_a^*$ be such that

$$\sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*) < +\infty, \quad \forall x \in \mathcal{X}_k,$$

$$\sup_{x \in \mathcal{X}_a} L(x, y^*) < +\infty, \quad \forall y^* \in \mathcal{Y}_s^*.$$

Theorem 3 (***superLagrangian duality***) *Let the Lagrangian L:* $\mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ *be an arbitrary given function. If there exists either a supermaximum point* $(\bar{x}, \bar{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ *such that*

$$L(\bar{x}, \bar{y}^*) = \max_{x \in \mathcal{X}_a} \max_{y^* \in \mathcal{Y}_a^*} L(x, y^*) = \max_{y^* \in \mathcal{Y}_a^*} \max_{x \in \mathcal{X}_a} L(x, y^*), \quad (6)$$

or a superminimax point $(\bar{x}, \bar{y}^*) \in \mathcal{X}_a \times \mathcal{Y}_a^*$ *such that*

$$L(\bar{x}, \bar{y}^*) = \min_{x \in \mathcal{X}_a} \max_{y^* \in \mathcal{Y}_a^*} L(x, y^*) = \min_{y^* \in \mathcal{Y}_a^*} \max_{x \in \mathcal{X}_a} L(x, y^*), \quad (7)$$

then $(\bar{x}, \bar{y}^*)$ *is a supercritical point of L on* $\mathcal{X}_a \times \mathcal{Y}_a^*$.

Conversely, if L is partially Gâteaux differentiable on an open subset $\mathcal{X}_a \times \mathcal{Y}_a^* \subset \mathcal{X} \times \mathcal{Y}^*$, *and* $(\bar{x}, \bar{y}^*)$ *is a supercritical point of L on* $\in \mathcal{X}_a \times \mathcal{Y}_a^*$, *then either the supermaximum theorem in the form*

$$L(\bar{x}, \bar{y}^*) = \max_{x \in \mathcal{X}_k} \max_{p \in \mathcal{Y}_a^*} L(x, y^*) = \max_{y^* \in \mathcal{Y}_s^*} \max_{x \in \mathcal{X}_a} L(x, y^*), \quad (8)$$

holds, or the superminimax theorem in the form

$$L(\bar{x}, \bar{y}^*) = \min_{x \in \mathcal{X}_k} \max_{y^* \in \mathcal{Y}_a^*} L(x, y^*) = \min_{y^* \in \mathcal{Y}_s^*} \max_{x \in \mathcal{X}_a} L(x, y^*) \quad (9)$$

holds.

This superLagrangian duality theorem shows a very important fact in Hamiltonian systems, i. e. the critical points of the Lagrangian L either maximize or minimaximize L on $\mathcal{X}_k \times \mathcal{Y}_s^*$ in either order.

Nonconvex Primal and Dual Problems

Let L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be an arbitrary given supercritical function. For any fixed $x \in \mathcal{X}_a$, let

$$P(x) = \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*). \quad (10)$$

Clearly, the function $P(x)$ need not be either convex or concave. Let $\mathcal{X}_k \subset \mathcal{X}_a$ be the *primal feasible set* such that P: $\mathcal{X}_k \to \mathbf{R}$ is finite and Gâteaux differentiable. Then for a nonconvex function P, two primal problems can be proposed as:

$$(\mathcal{P}_{\inf}): \quad P(x) \to \min, \quad \forall u \in \mathcal{X}_k, \quad (11)$$

$$(\mathcal{P}_{\sup}): \quad P(x) \to \max, \quad \forall u \in \mathcal{X}_k. \quad (12)$$

The problems $(\mathcal{P}_{\inf})$ and $(\mathcal{P}_{\sup})$ are realisable if the primal feasible set $\mathcal{X}_k$ is not empty.

Dually, for any fixed $y^* \in \mathcal{Y}_a^*$, let

$$P^d(y^*) = \sup_{x \in \mathcal{X}_a} L(x, y^*) \quad (13)$$

with the *dual feasible set* $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$ such that P^d: $\mathcal{Y}_s^* \to \mathbf{R}$ is finite and Gâteaux differentiable. The two nonconvex dual problems are:

$$(\mathcal{P}_{\inf}^d): \quad P^d(y^*) \to \min, \quad \forall y^* \in \mathcal{Y}_s^*, \quad (14)$$

$$(\mathcal{P}_{\sup}^d): \quad P^d(y^*) \to \max, \quad \forall y^* \in \mathcal{Y}_s^*. \quad (15)$$

These two dual problems are realisable if the dual feasible set $\mathcal{Y}_s^*$ is not empty.

Theorem 4 (***Biduality theorem***) *Let L:* $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ *be a given arbitrary function such that P and* P^d *are well-defined by (10) and (13) on the open subsets* $\mathcal{X}_k$ *and* $\mathcal{Y}_s^*$, *respectively.*

1) *If* $(\bar{x}, \bar{y}^*)$ *is a supercritical point of* L *on* $\mathcal{X}_k \times \mathcal{Y}_s^*$, *then* $DP(\bar{x}) = 0$, $DP^d(\bar{y}^*) = 0$, *and*

$$P(\bar{x}) = L(\bar{x}, \bar{y}^*) = P^d(\bar{y}^*). \quad (16)$$

2) *If* $(\bar{x}, \bar{y}^*)$ *is a supercritical point of* L *on* $\mathcal{X}_k \times \mathcal{Y}_s^*$, *then* $\bar{x}$ *is a minimizer of* P *on* $\mathcal{X}_k$ *if and only if* $\bar{y}^*$ *is a minimizer of* P^d *on* $\mathcal{Y}_s^*$, *i. e. the double-min duality*

$$P(\bar{x}) = \inf_{x \in \mathcal{X}_k} P(x) \Leftrightarrow \inf_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = P^d(\bar{y}^*) \quad (17)$$

holds.

3) *If* $(\bar{x}, \bar{y}^*)$ *is a supercritical point of* L *on* $\mathcal{X}_k \times \mathcal{Y}_s^*$, *then* $\bar{x}$ *is a maximizer of* P *on* $\mathcal{X}_k$ *if and only if* $\bar{y}^*$ *is a maximizer of* P^d *on* $\mathcal{Y}_s^*$, *i. e. the double-max duality*

$$P(\bar{x}) = \sup_{x \in \mathcal{X}_k} P(x) \Leftrightarrow \sup_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = P^d(\bar{y}^*) \quad (18)$$

holds.

D.C. Programming and Hamiltonian

In d.c. programming, the primal function P: $\mathcal{X}_k \to \mathbf{R}$ can be written as

$$P(x) = W(\Lambda x) - F(x),$$

where Λ: $\mathcal{X} \to \mathcal{Y}$ is a linear operator, W: $\mathcal{Y}_a \to \mathbf{R}$ and F: $\mathcal{X}_a \to \mathbf{R}$ are two convex, Gâteaux differentiable real-valued functions, satisfying the *Legendre duality relations*

$$x^* = DF(x) \Leftrightarrow x = DF^*(x^*)$$
$$\Leftrightarrow \langle x, x^* \rangle = F(x) + F^*(x^*)$$

on $\mathcal{X}_a \times \mathcal{X}_a^*$, and

$$y^* = DW(y) \Leftrightarrow y = DW^*(y^*)$$
$$\Leftrightarrow \langle y; y^* \rangle = W(y) + W^*(y^*)$$

on $\mathcal{Y}_a \times \mathcal{Y}_a^*$, where F^*: $\mathcal{X}_a^* \to \mathbf{R}$ and W^*: $\mathcal{Y}_a^* \to \mathbf{R}$ are the *Legendre conjugates* of F and W, respectively.

In dynamical systems, if $\Lambda = d/dt$ is a differential operator, its adjoint associated with the standard bilinear forms in $\mathcal{L}^2$ is $\Lambda^* = -\, d/dt$. If W denotes the kinetic energy, F stands for potential energy, then the primal function $P(x) = W(\Lambda x) - F(x)$ is the *total action* of the system. The primal feasible set $\mathcal{X}_k \subset \mathcal{X}$, defined by

$$\mathcal{X}_k = \{x \in \mathcal{X}_a : \ \Lambda x \in \mathcal{Y}_a\},$$

is called the *kinetically admissible space*. Clearly, P: $\mathcal{X}_k \to \mathbf{R}$ is nonconvex.

The *Lagrangian form* associated with the nonconvex primal problems is defined by

$$L(x, y^*) = \langle \Lambda x; y^* \rangle - W^*(y^*) - F(x), \quad (19)$$

which is G âteaux differentiable on $\mathcal{X}_a \times \mathcal{Y}_a^*$. The critical condition $DL(\bar{x}, \bar{y}^*) = 0$ leads to the Lagrange equations:

$$\Lambda \bar{x} = DW^*(\bar{y}^*), \quad \Lambda^* \bar{y}^* = DF(\bar{x}).$$

Clearly, L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ is a supercritical function, and

$$P(x) = \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*), \quad \forall x \in \mathcal{X}_k.$$

Dually, for any given dual feasible $y^* \in \mathcal{Y}_s^*$,

$$P^d(y^*) = \sup_{x \in \mathcal{X}_a} L(x, y^*) = F^*(\Lambda^* y^*) - W^*(y^*),$$

where the dual feasible set $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$ is defined by

$$\mathcal{Y}_s^* = \{y^* \in \mathcal{Y}_a^* : \ \Lambda^* y^* \in \mathcal{X}_a^*\}.$$

The criticality conditions $DL(\bar{x}, \bar{y}^*) = 0$, $DP(\bar{x}) = 0$ and $DP^d(\bar{y}^*) = 0$ are equivalent to each other.

The *Hamiltonian* H: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ associated with the Lagrangian L is defined by

$$\begin{aligned} H(x, y^*) &= \langle \Lambda x; y^* \rangle - L(x, y^*) \\ &= W^*(y^*) + F(x). \end{aligned} \quad (20)$$

For d.c. programming, $H(x, y^*)$ is a convex function on $\mathcal{X}_a \times \mathcal{Y}_a^*$. The critical point $(\bar{x}, \bar{y}^*)$ of L satisfies the Hamiltonian *canonical form*

$$\Lambda \bar{x} = D_{y^*} H(\bar{x}, \bar{y}^*), \quad \Lambda^* \bar{y}^* = D_x H(\bar{x}, \bar{y}^*).$$

Particularly, if $W(\Lambda\ x) = 1/2\ \langle \Lambda x,\ C\Lambda\ x \rangle$ is a quadratic function, C: $\mathcal{Y}_a \to \mathcal{Y}_a^*$ is a symmetric operator such that the composite operator $A = \Lambda^*\ C\ \Lambda = A^*$ is selfadjoint, then the total action can be written as

$$P(x) = \frac{1}{2} \langle x, Ax \rangle - F(x).$$

Let $P^c(x) = -\, P^d(C\ \Lambda\ x)$; then the function P^c: $\mathcal{X}_a \to \mathbf{R}$

$$P^c(x) = \frac{1}{2} \langle x, Ax \rangle - F^*(Ax)$$

is the so-called *Clarke dual action* (see [1]).

Theorem 5 (*Clarke duality theorem*) *Let $A\colon \mathcal{X}_k \subset \mathcal{X}_a \to \mathcal{X}_a^*$ be a closed selfadjoint operator, and $\operatorname{Ker} A = \{x \in \mathcal{X}\colon A\, x = 0 \in \mathcal{X}^*\}$ the null space of A. If $\bar{x} \in \mathcal{X}_k$ is a critical point of P, then any vector $x \in \operatorname{Ker} A + \bar{x}$ is a critical point of P^c.*

Conversely, if there exists a $x_o \in \mathcal{X}_k$ such that $A\, x_o \in \mathcal{X}_a^*$, then for a given critical point $\bar{x}$ of P^c, any vector $x \in \operatorname{Ker} A + \bar{x}$ is a critical point of P.

Example 6 Let us consider a very simple one-dimensional optimization problem with constraint

$$\begin{cases} F(x) = & \frac{1}{2}kx^2 - fx \to \max \\ \text{s.t.} & a \le x \le b, \end{cases} \tag{21}$$

where $k > 0$ and $f \in \mathbf{R}$ are given constants. We assume that $-\infty < a < 0 < b < \infty$. Since $F(x)$ is strictly convex on the closed set $[a, b]$, the maximum is attained only on the boundary, i. e.

$$\sup_{x\in[a,b]} F(x) = \max\{F(a), F(b)\} < \infty .$$

The classical Lagrange multiplier method cannot be used for this nonconvex problem. To set this problem within our framework, we need only set $\mathcal{X} = \mathbf{R}$, $\mathcal{X}_a = [a, b]$ and let $\Lambda = 1$, so that

$$y = \Lambda x = x \in \mathcal{Y} = \mathbb{R}.$$

Thus, the range of the mapping $\Lambda\colon \mathcal{X}_a \to \mathcal{Y} = \mathbf{R}$ is $\mathcal{Y}_a = [a, b]$. Let

$$W(y) = \begin{cases} 0 & \text{if } y \in \mathcal{Y}_a, \\ +\infty & \text{if } y \notin \mathcal{Y}_a. \end{cases}$$

It is not difficult to check that $W\colon \mathcal{Y} \to \mathbf{R} \cup \{+\infty\}$ is convex. On $\mathcal{Y}_a$, W is finite and differentiable. Thus, the primal feasible space can be defined by

$$\mathcal{X}_k = \{x \in \mathcal{X}_a\colon\ \Lambda x = x \in \mathcal{Y}_a\} = [a, b].$$

Clearly, on $\mathcal{X}_k$ $P(x) = W(\Lambda\, x) - F(x) = F(x)$. The constrained maximization problem (21) is then equivalent to the standard nonconvex minimization problem $(\mathcal{P}_{\inf})$: $P(x) \to \min$, $\forall x \in \mathcal{X}_k$.

Since $F(x)$ is strictly convex and differentiable on $\mathcal{X}_a = [a, b]$, and

$$x^* = DF(x) = kx - f \in \mathcal{X}_a^*$$

is invertible, where

$$\mathcal{X}_a^* = [ak - f, bk - f] \subset \mathcal{X}^* = \mathbb{R},$$

the Legendre conjugate $P^c\colon \mathcal{X}_a^* \to \mathbf{R}$ can easily be obtained as

$$F^*(x^*) = \max_{x\in\mathcal{X}_a}\{xx^* - F(x)\} = \frac{1}{2k}(x^* + f)^2 .$$

By the *Fenchel transformation*, the conjugate of the nonsmooth function W can be obtained as

$$W^*(y^*) = \sup_{y\in\mathcal{Y}}\{yy^* - W(y)\} = \max_{y\in\mathcal{Y}_a} yy^*$$
$$= \begin{cases} by^* & \text{if } y^* > 0, \\ 0 & \text{if } y^* = 0, \\ ay^* & \text{if } y^* < 0. \end{cases}$$

It is convex and differentiable on $\mathcal{Y}_a^* = \mathcal{Y}^* = \mathbf{R}$.

On $\mathcal{X}_a \times \mathcal{Y}_a^* = [a, b] \times \mathbf{R}$, the Lagrange form for this nonconvex programming is well-defined by

$$L(x, y^*) = y^* \Lambda x - W^*(y^*) - F(x)$$
$$= \begin{cases} xy^* - by^* - \frac{1}{2}kx^2 + fx & \text{if } y^* \ge 0, \\ xy^* - ay^* - \frac{1}{2}kx^2 + fx & \text{if } y^* < 0. \end{cases}$$

Since both W^* and F are convex, $L(x, y^*)$ is a supercritical point function. If $x \in \mathcal{X}_k = [a, b]$, then

$$P(x) = \sup_{y^*\in\mathcal{Y}_a^*} L(x, y^*).$$

On the other hand, for any y^* in the dual feasible space

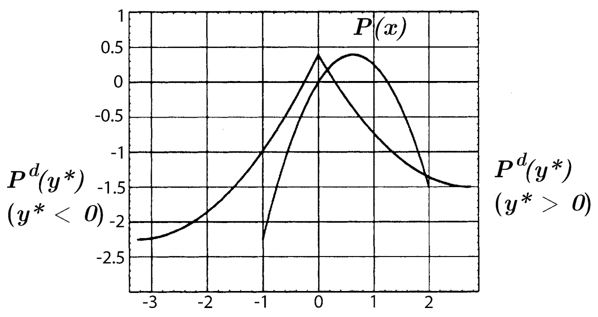
$$\mathcal{Y}_s^* = \{y^* \in \mathcal{Y}_a^* = \mathbb{R}\colon\ \Lambda^* y^* = y^* \in \mathcal{X}_a^*\}$$
$$= [ak - f, bk - f],$$

the dual function is obtained by

$$P^d(y^*) = \sup_{x\in\mathcal{X}_a} L(x, y^*)$$
$$= \sup_{x\in\mathbb{R}}\{(\Lambda x)y^* - F(x)\} - W^*(y^*)$$
$$= F^*(\Lambda^* y^*) - W^*(y^*),$$

where

$$F^*(\Lambda^* y^*) = \sup_{x\in\mathcal{X}_a}\{(\Lambda x)y^* - F(x)\}$$
$$= \sup_{x\in\mathbb{R}}\{x(y + f) - \frac{1}{2}kx^2\}$$
$$= \frac{1}{2k}(y^* + f)^2 .$$



Duality Theory: Biduality in Nonconvex Optimization, Figure 1
Biduality in constrained nonconvex optimization

Thus, the dual action P^d is well defined on $\mathcal{Y}_s^*$ by

$$P^d(y^*) = \begin{cases} \frac{1}{2k}(y^* + f)^2 - by^* & \text{if } y^* > 0, \\ \frac{1}{2k} f^2 & \text{if } y^* = 0, \\ \frac{1}{2k}(y^* + f)^2 - ay^* & \text{if } y^* < 0. \end{cases} \quad (22)$$

This is a double-well function on **R** (see Fig. 1.). The dual problem

$$(P^d_{\inf}): \quad P^d(y^*) \to \min, \quad \forall y^* \in \mathcal{Y}_s^*,$$

is a convex optimization problem on either

$$\mathcal{Y}_s^{*+} = \{y^* \in \mathcal{Y}_s^* : \; y^* > 0\}$$

or

$$\mathcal{Y}_s^{*-} = \{y^* \in \mathcal{Y}_s^* : \; y^* < 0\} .$$

In n-dimensional problems, this dual problem is much easier than the primal problem. The criticality condition leads to

$$\bar{y}^* = \begin{cases} bk - f & \text{if } \bar{y}^* > 0, \\ 0 & \text{if } \bar{y}^* = 0 \\ ak - f & \text{if } \bar{y}^* < 0. \end{cases}$$

It is easy to check that the following duality theorems hold:

$$\min_{x \in X_k} P(x) = \min_{y^* \in \mathcal{Y}_s^*} P^d(y^*),$$

$$\max_{x \in X_k} P(x) = \max_{y^* \in \mathcal{Y}_s^*} P^d(y^*).$$

The graphs of $P(x)$ and $P^d(y^*)$ are shown in Fig. 1.

See also

- ▶ Duality Theory: Monoduality in Convex Optimization
- ▶ Duality Theory: Triduality in Global Optimization
- ▶ History of Optimization
- ▶ Von Neumann, John

References

1. Gao DY (1999) Duality principles in nonconvex systems: Theory, methods and applications. Kluwer, Dordrecht

Duality Theory: Monoduality in Convex Optimization

DAVID YANG GAO
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 49-XX, 90-XX, 93-XX

Article Outline

Keywords

Duality; Convex optimization; Saddle Lagrangian; Legendre duality; Fenchel–Rockafellar duality; Complementarity; Linear programming; Center path

The concept of *duality* is one of the most successful ideas in modern mathematics and science. Inner beauty in natural phenomena is bound up with duality, which has always been a rich source of inspiration in human knowledge through the centuries. Duality in mathematics, roughly speaking, is a fundamental concept that underlies many aspects of *extremum principles* in natural systems. Eigenvectors, geodesics, minimal surfaces, KKT conditions, harmonic maps, Hamiltonian canonical equations and equilibrium states of many field equations are all critical points of certain functions on some appropriate constraint sets or manifolds. Considerable attention has been attracted on this fascinating research subject during the last years. A comprehensive study on duality theory in general nonconvex and nonsmooth systems is given in [1]. In global optimization problems, duality falls principally into three categories:

1) the classical *saddle Lagrange duality* (i. e. monoduality) in convex optimization;
2) the nice *biduality* in convex Hamilton systems or the *d.c. programming* (difference of convex functions); and
3) the interesting *triduality* in general nonconvex systems.

Saddle Lagrange Duality

Let $(\mathcal{X}, \mathcal{X}^*)$ and $(\mathcal{Y}, \mathcal{Y}^*)$ be two pairs real vector spaces, finite- or infinite-dimensional, and let $\langle *, * \rangle\colon \mathcal{X} \times \mathcal{X}^* \to \mathbf{R}$ and $\langle *; * \rangle\colon \mathcal{Y} \times \mathcal{Y}^* \to \mathbf{R}$ be certain *bilinear forms* which put the paired spaces $(\mathcal{X}, \mathcal{X}^*)$ and $(\mathcal{Y}, \mathcal{Y}^*)$ in *duality*, respectively. In classical convex optimization, a real-valued function L: $\mathcal{X} \times \mathcal{Y}^* \to \mathbf{R}$ is said to be a *saddle function* if it is convex in one variable and concave in the other one.

A pair $(\overline{x}, \overline{y}^*)$ is called a *right saddle point* of L on a subspace $\mathcal{X}_a \times \mathcal{Y}_{a*} \subset \mathcal{X} \times \mathcal{Y}^*$ if

$$L(\overline{x}, y^*) \leq L(\overline{x}, \overline{y}^*) \leq L(x, \overline{y}^*)$$

holds for any $(x, y^*) \in \mathcal{X}_a \times \mathcal{Y}^*_a$.

A pair $(\overline{x}, \overline{y}^*)$ is called a *left saddle point* of L on a subspace $\mathcal{X}_a \times \mathcal{Y}^*_a \subset \mathcal{X} \times \mathcal{Y}^*$ if it is a right saddle point of $-L$ on the subspace $\mathcal{X}_a \times \mathcal{Y}^*_a$.

A pair $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}^*_a$ is called a *critical point* of L if L is partially Gâteaux differentiable at $(\overline{x}, \overline{y}^*)$ and

$$D_x L(\overline{x}, \overline{y}^*) = 0, \quad D_{y^*} L(\overline{x}, \overline{y}^*) = 0,$$

where $D_x L\colon \mathcal{X}_a \to \mathcal{X}^*$ and $D_{y^*} L\colon \mathcal{Y}^*_a \to \mathcal{Y}$ denote, respectively, the partial Gâteaux derivatives of L with respect to x and y^*.

Any critical point of a Gâteaux differentiable saddle function is a saddle point. However, if $(\overline{x}, \overline{y}^*)$ is a saddle point of L it does not follow that L is a saddle function. In convex optimization problems, the following statements are of important theoretical value.

S1) Under certain necessary and sufficient conditions, suppose that

$$\inf_{x \in \mathcal{X}_a} \sup_{y^* \in \mathcal{Y}^*_a} L(x, y^*) = \sup_{y^* \in \mathcal{Y}^*_a} \inf_{x \in \mathcal{X}_a} L(x, y^*). \quad (1)$$

A statement of this type is called a *saddle-minimax theorem* and the pair $(\overline{x}, \overline{y}^*)$ is called a *saddle-minimax point* of L on $\mathcal{X}_a \times \mathcal{Y}^*_a$.

S2) Under certain conditions, suppose that a pair $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}^*_a$ exists such that

$$L(x, \overline{y}^*) \geq L(\overline{x}, \overline{y}^*) \geq L(\overline{x}, y^*) \quad (2)$$

holds for any $(x, y^*) \in \mathcal{X}_a \times \mathcal{Y}^*_a$. A statement of this type is called a *right saddle-point theorem*.

Let $\mathcal{X}_k$ be a subset of $\mathcal{X}_a$ such that $\mathcal{X}_k$ contains all point $u \in \mathcal{X}_a$ for which the supremum $\sup_{y^*} L(x, y^*)$ is finite, i. e.

$$\sup_{y^* \in \mathcal{Y}^*_a} L(x, y^*) < +\infty, \quad \forall x \in \mathcal{X}_k.$$

Dually, let $\mathcal{Y}^*_s$ be a subset of $\mathcal{Y}^*_a$ such that $\mathcal{Y}^*_s$ contains all points $y^* \in \mathcal{Y}^*_a$ for which the infimum $\inf_x L(x, y^*)$ is finite, i. e.

$$\inf_{x \in \mathcal{X}_a} L(x, y^*) > -\infty, \quad \forall y^* \in \mathcal{Y}^*_s.$$

The sets $\mathcal{X}_k$ and $\mathcal{Y}^*_s$ may be either empty or $\mathcal{X}_k = \mathcal{X}_a$ and $\mathcal{Y}^*_s = \mathcal{Y}^*_a$. The connection between the minimax theorem and the saddle-point theorem is given by the following results.

Theorem 1 (***Saddle-minimax theorem***) *Let L: $\mathcal{X}_a \times \mathcal{Y}^*_a \to \mathbf{R}$ be a given arbitrary function. If there exists a saddle-minimax point $(\overline{x}, \overline{y}^*) \in \mathcal{X}_a \times \mathcal{Y}^*_a$ such that*

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_a} \max_{y^* \in \mathcal{Y}^*_a} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}^*_a} \min_{x \in \mathcal{X}_a} L(x, y^*), \end{aligned} \quad (3)$$

then $(\overline{x}, \overline{y}^*)$ is a saddle point of L on $\mathcal{X}_a \times \mathcal{Y}^*_a$.

Conversely, if $L(x, y^*)$ possesses a saddle point $(\overline{x}, \overline{y}^*)$ on $\mathcal{X}_a \times \mathcal{Y}^*_a$, then the saddle-minimax theorem in the form

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_k} \max_{y^* \in \mathcal{Y}^*_a} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}^*_s} \min_{x \in \mathcal{X}_a} L(x, y^*) \end{aligned} \quad (4)$$

holds.

Let L: $\mathcal{X}_a \times \mathcal{Y}^*_a \to \mathbf{R}$ be a given arbitrary right saddle function. For any fixed $x \in \mathcal{X}_a$, let

$$P(x) = \sup_{y^* \in \mathcal{Y}^*_a} L(x, y^*). \quad (5)$$

Let $\mathcal{X}_k \subset \mathcal{X}_a$ be the domain of P such that $P\colon \mathcal{X}_k \to \mathbf{R}$ is finite and Gâteaux differentiable. Then the inf-problem

$$(\mathcal{P}_{\inf})\colon \quad P(x) \to \min, \quad \forall u \in \mathcal{X}_k, \tag{6}$$

is called the *primal problem*.

Dually, for any fixed $y^* \in \mathcal{Y}_a^*$, let

$$P^d(y^*) = \sup_{x \in \mathcal{X}_a} L(x, y^*) \tag{7}$$

with domain $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$, on which, $P^d\colon\mathcal{Y}_s^* \to \mathbf{R}$ is finite and Gâteaux differentiable. Thus, the sup-problem

$$(\mathcal{P}_{\sup}^d)\colon \quad P^d(y^*) \to \max, \quad \forall y^* \in \mathcal{Y}_s^*, \tag{8}$$

is referred as the *dual problem*. The problems $(\mathcal{P}_{\inf})$ and $(\mathcal{P}_{\sup}^d)$ are *realisable* if $\mathcal{X}_k$ and $\mathcal{Y}_s^*$ are not empty, i. e., there exists a pair $(\bar{x}, \bar{y}^*) \in \mathcal{X}_k \times \mathcal{Y}_s^*$ such that

$$P(\bar{x}) = \min_{x \in \mathcal{X}_k} P(x) = \inf_{x \in \mathcal{X}_k} P(x),$$
$$P^d(\bar{y}^*) = \max_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = \sup_{y^* \in \mathcal{Y}_s^*} P^d(y^*).$$

Theorem 2 (*Saddle duality theorem*) *Let $L\colon \mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ be a given arbitrary function such that P and P^d are well-defined by (5) and (7) on the open subsets $\mathcal{X}_k$ and $\mathcal{Y}_s^*$, respectively. If $(\bar{x}, \bar{y}^*)$ is a saddle point of L on $\mathcal{X}_k \times \mathcal{Y}_s^*$, P is Gâteaux differentiable at $\bar{x}$, and P^d is Gâteaux differentiable at $\bar{y}^*$, then $DP(\bar{x}) = 0$, $DP^d(\bar{y}^*) = 0$, and*

$$P(\bar{x}) = L(\bar{x}, \bar{y}^*) = P^d(\bar{y}^*). \tag{9}$$

Theorem 3 (*Weak duality theorem*) *The inequality*

$$P(x) \geq P^d(y^*) \tag{10}$$

holds for all $(x, y^*) \in \mathcal{X}_k \times \mathcal{Y}_s^*$.

Theorem 4 (*Strong duality theorem*) *$(\bar{x}, \bar{y}^*)$ is a saddle-point of L on $\mathcal{X}_k \times \mathcal{Y}_s^* \subseteq \mathcal{X}_a \times \mathcal{Y}_a^*$ if and only if the equality*

$$P(\bar{x}) = \inf_{x \in \mathcal{X}_k} P(x) = \sup_{y^* \in \mathcal{Y}_s^*} P^d(y^*) = P^d(\bar{y}^*) \tag{11}$$

holds.

Fenchel–Rockafellar Duality

Very often, the primal function $P\colon \mathcal{X}_k \to \mathbf{R}$ can be written as

$$P(x) = W(\Lambda x) - F(x),$$

where $\Lambda\colon \mathcal{X} \to \mathcal{Y}$ is a linear operator, $W\colon \mathcal{Y}_a \to \mathbf{R}$ and $F\colon \mathcal{X}_a \to \mathbf{R}$ are Gâteaux differentiable real-valued functions. The feasible set $\mathcal{X}_k \subset \mathcal{X}$ is then defined by

$$\mathcal{X}_k = \{x \in \mathcal{X}_a\colon \ \Lambda x \in \mathcal{Y}_a\}\,.$$

Clearly, $P\colon \mathcal{X}_k \to \mathbf{R}$ is convex if W is convex on $\mathcal{Y}_a$ and F is concave on $\mathcal{X}_a$.

The *conjugate function* $W^*\colon \mathcal{Y}_a^* \to \mathbf{R}$ of $W(y)$ is defined by the *Fenchel transformation*, i. e.

$$W^*(y^*) = \sup_{y \in \mathcal{Y}_a} \{\langle y; y^* \rangle - W(y)\}, \tag{12}$$

which is always l.s.c. and convex on $\mathcal{Y}^*$. The following *Fenchel–Young inequality*

$$W(y) \geq \langle y; y^* \rangle - W^*(y^*) \tag{13}$$

holds on $\mathcal{Y}_a \times \mathcal{Y}_a^*$. If W is strictly convex, and Gâteaux differentiable on $\mathcal{Y}_a \subset \mathcal{Y}$, then the following *Legendre duality relations*

$$y^* = DW(y) \Leftrightarrow y = DW^*(y^*)$$
$$\Leftrightarrow \langle y; y^* \rangle = W(y) + W^*(y^*)$$

hold on $\mathcal{Y}_a \times \mathcal{Y}_a^*$. In this case, we have $W(y) = W^{**}(y)$, the biconjugate of W, and the Fenchel transformation (12) is equivalent to the classical *Legendre transformation*

$$W^*(y^*) = \langle y(y^*); y^* \rangle - W(y(y^*)).$$

The *Lagrangian form* associated with $(\mathcal{P}_{\inf})$ is defined by

$$L(x, y^*) = \langle \Lambda x; y^* \rangle - W^*(y^*) - F(x), \tag{14}$$

which is Gâteaux differentiable on $\mathcal{X}_a \times \mathcal{Y}_a^*$. The critical condition $DL(\bar{x}, \bar{y}^*) = 0$ leads to the *Lagrange equations*:

$$\Lambda\bar{x} = DW^*(\bar{y}^*), \quad \Lambda^*\bar{y}^* = DF(\bar{x}), \tag{15}$$

where $\Lambda^*\colon \mathcal{Y}_a^* \to \mathcal{X}_a^*$ is the adjoint operator of Λ. Clearly, $L\colon \mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ is a right saddle function if

$F(x)$ is concave on $\mathcal{X}_a$. For convex function $W(y)$, we have

$$P(x) = \sup_{y^* \in \mathcal{Y}_a^*} L(x, y^*), \quad \forall x \in \mathcal{X}_k.$$

The Fenchel conjugate function of a concave function F: $\mathcal{X}_a \to \mathbf{R}$ is defined by

$$F^*(x^*) = \inf_{x \in \mathcal{X}_a} \{\langle x, x^* \rangle - F(x)\}. \tag{16}$$

Thus, for any given dual admissible $y^* \in \mathcal{Y}_s^*$ with

$$\mathcal{Y}_s^* = \{y^* \in \mathcal{Y}_a^* : \Lambda^* y^* \in \mathcal{X}_a^*\},$$

the Fenchel–Rockafellar dual function P^d: $\mathcal{Y}_k^* \to \mathbf{R}$ can be obtained as

$$P^d(y^*) = \inf_{x \in \mathcal{X}_a} L(x, y^*) = F^*(\Lambda^* y^*) - W^*(y^*).$$

If P is Gâteaux differentiable on $\mathcal{X}_k$, the critical condition $DP(\bar{x}) = 0$ leads to the *Euler–Lagrange equation* of the primal problem ($\mathcal{P}_{\inf}$):

$$\Lambda^* DW(\Lambda \bar{x}) - DF(\bar{x}) = 0. \tag{17}$$

Similarly, the critical condition $DP^d(\bar{y}^*) = 0$ gives the *dual Euler–Lagrange equation* of ($\mathcal{P}^d_{\sup}$):

$$\Lambda DF^*(\Lambda^* \bar{y}^*) - DW^*(\bar{y}^*) = 0. \tag{18}$$

Clearly, the critical point theorem (9) holds if the Lagrange equation (15), Euler–Lagrange equation (17) and its dual equation (18) are equivalent to each others.

For any given F and W, the weak duality theorem (10) always holds on $\mathcal{X}_k \times \mathcal{F}_s^*$. The difference inf P − sup P^d is the so-called *duality gap*. For convex primal problem, the duality gap is zero and the strong Lagrange duality theorem (11) holds, which is also referred as the *Fenchel–Rockafellar duality theory*.

Linear Programming and Central Path

Let us now demonstrate how the above scheme fits in with finite-dimensional linear programming. Let $\mathcal{X} = \mathcal{X}^* = \mathbf{R}^n$, $\mathcal{Y} = \mathcal{F}^* = \mathbf{R}^m$, with the standard inner products $\langle x, x^* \rangle = x^\intercal x^*$ in $\mathbf{R}^n$, and $\langle y; y^* \rangle = y^\intercal y^*$ in $\mathbf{R}^m$. For fixed $\bar{x}^* = c \in \mathbb{R}^n$ and $\bar{y} = b \in \mathbb{R}^m$, the primal problem is a constrained linear optimization problem:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} & \langle c, x \rangle \\ \text{s.t.} & \Lambda x = b, \quad x \geq 0, \end{cases} \tag{19}$$

where $\Lambda \in \mathbf{R}^{m \times n}$ is a matrix, and its adjoint is simply $\Lambda^* = \Lambda^\intercal \in \mathbf{R}^{n \times m}$. To reformulate this linear constrained optimization problem in the model form ($\mathcal{P}_{\inf}$), we need to set $\mathcal{X}_a = \{x \in \mathbf{R}^n x \geq 0\}$, which is a convex cone in $\mathbf{R}^n$, $\mathcal{Y}_a = \{y \in \mathbf{R}^m : y = b\}$, a hyperplane in $\mathbf{R}^m$, and let

$$F(x) = -\langle c, x \rangle, \quad \forall x \in \mathcal{X}_a,$$
$$W(y) = 0, \quad \forall y \in \mathcal{Y}_a.$$

Thus on the primal feasible set

$$\mathcal{X}_k = \{x \in \mathbb{R}^n : \Lambda x = b, \ x \geq 0\}$$

we have $P(x) = W(\Lambda x) - F(x) = \langle c, x \rangle$. The conjugate functions in this elementary case may be calculated at once as

$$W^*(y^*) = \sup_{y \in \mathcal{Y}_a} \langle y; y^* \rangle = \langle b; y^* \rangle,$$
$$\forall y^* \in \mathcal{Y}_a^* = \mathbb{R}^m,$$
$$F^*(x^*) = \inf_{x \in \mathcal{X}_a} \langle x, x^* + c \rangle = 0, \quad \forall x^* \in \mathcal{X}_a^*,$$

where $\mathcal{X}_a^* = \{x^* \in \mathbf{R}^n : x^* + c \geq 0\}$ is a polar cone of $\mathcal{X}_a$. Thus, on the dual feasible space

$$\mathcal{Y}_s^* = \{y^* \in \mathbb{R}^m : \Lambda^* y^* + c \geq 0\},$$

the problem dual to the linear programming (19) reads

$$\max_{p \in \mathbb{R}^m} P^d(y^*) = -\langle b; y^* \rangle, \quad \forall y^* \in \mathcal{Y}_s^*. \tag{20}$$

The Lagrangian L: $\mathcal{X}_a \times \mathcal{Y}_a^* \to \mathbf{R}$ associated with this constrained linear programming is

$$L(x, y^*) = \langle \Lambda x; y^* \rangle - \langle b; y^* \rangle + \langle c, x \rangle$$
$$= \langle x, \Lambda^* y + c \rangle - \langle b; y^* \rangle.$$

But for inequality constraints in $\mathcal{X}_a$, the Lagrange multiplier $x^* = \Lambda^* y^* \in \mathbf{R}^n$ has to satisfy the following *KKT optimality conditions*

$$\begin{aligned} &\Lambda x = b, \quad s = c + \Lambda^* y^*, \\ &x \geq 0, \quad s \geq 0, \quad s^\top x, = 0, \end{aligned} \tag{21}$$

where the vector $s \in \mathbf{R}^n$ is called the *dual slacks*. The problem of finding $(\bar{x}, \bar{y}^*, \bar{s})$ satisfying (21) is also known as the *mixed linear complementarity problem*.

By using the vector of dual slacks $s \in \mathbf{R}^n$, the dual problem can be rewritten as

$$\begin{cases} \max\limits_{p \in \mathbb{R}^m} & \langle b; p \rangle \\ \text{s.t.} & \Lambda^* y^* + c - s = 0, \quad s \geq 0. \end{cases} \tag{22}$$

We can see that the primal variable x is the Lagrange multiplier for the constraint $\Lambda^* y^* + c \geq 0$ in the dual problem. However, the dual variables y^* and s are respectively Lagrange multipliers for the constraints $\Lambda x = b$ and $x \geq 0$ in the primal problem. These choices are not accidents.

Theorem 5 *The vector $\bar{x} \in \mathbb{R}^n$ is a solution of (19) if and only if there exists a Lagrange multipliers $(\bar{y}^*, \bar{s}) \in \mathbb{R}^m \times \mathbb{R}^n$ for which the KKT optimality conditions (21) hold for $(\bar{x}, \bar{y}^*, \bar{s})$. Dually, the vector $(\bar{y}^*, \bar{s}) \in \mathbb{R}^m \times \mathbb{R}^n$ is a solution of (22) if and only if there exists a Lagrange multiplier $\bar{x} \in \mathbb{R}^n$ such that the KKT conditions (21) hold for $(\bar{x}, \bar{y}^*, \bar{s})$.*

The vector $(\bar{x}, \bar{y}^*, \bar{s})$ is called a *primal-dual solution* of (19). The so-called *primal-dual methods* in mathematical programming are those methods to find primal-dual solutions $(\bar{x}, \bar{y}^*, \bar{s})$ by applying variants of Newton's method to the three equations in (21) and modifying the search directions and steplengths so that the inequalities in (21) are satisfied at every iteration. If the inequalities are strictly satisfied, the methods are called *primal-dual interior-point methods*. In these methods, the so called *central path* $\mathcal{C}_{\text{path}}$ plays a vital role in the theory of primal-dual algorithms. It is a parametrical curve of strictly feasible points defined by

$$C_{\text{path}} = \left\{ (x_\tau, y^*_\tau, s_\tau)^\top \in \mathbb{R}^{2n+m} \colon \ \tau > 0 \right\}, \qquad (23)$$

where each point $(x_\tau, y^*_\tau, s_\tau)$ solves the following system:

$$\begin{aligned} &\Lambda x = b, \quad \Lambda^* y^* + c = s, \\ &x > 0, \quad s > 0, \quad u_i s_i = \tau, \quad i = 1, \ldots, n. \end{aligned} \qquad (24)$$

This problem has a unique solution $(x_\tau, y^*_\tau, s_\tau)$ for each $\tau > 0$ if and only if the *strictly feasible set*

$$\mathcal{F}_o = \left\{ (x, p, s) \colon \begin{array}{c} \Lambda x = b, \ \Lambda^* y^* + c = s, \\ x > 0, \ s > 0 \end{array} \right\}$$

is nonempty. A comprehensive study of the primal-dual interior-point methods in mathematical programming has been given in [3] and [2].

See also

- ▶ Duality Theory: Biduality in Nonconvex Optimization
- ▶ Duality Theory: Triduality in Global Optimization
- ▶ History of Optimization

References

1. Gao DY (1999) Duality principles in nonconvex systems: Theory, methods and applications. Kluwer, Dordrecht
2. Wright SJ (1996) Primal-dual interior-point methods. SIAM, Philadelphia
3. Ye Y (1997) Interior point algorithms: Theory and analysis. Discrete Math and Optim. Wiley/Interscience, New York

Duality Theory: Triduality in Global Optimization

DAVID YANG GAO
Virginia Polytechnic Institute and State University, Blacksburg, USA

MSC2000: 49-XX, 90-XX, 93-XX

Article Outline

Keywords

Duality; Nonconvexity; Global minimization; SuperLagrangian; Triduality; Triality

Consider the following general nonconvex extremum problem ($\mathcal{P}$):

$$P(x) = \Phi(x, \Lambda(x)) \to \text{extremum}, \quad \forall x \in \mathcal{X}, \quad (1)$$

where $\mathcal{X}$ is a locally convex topological vector space (l.c.s.), $P\colon \ \mathcal{X} \to \overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$ is a nonconvex and nonsmooth extended function, whose effective domain

$$\mathcal{X}_k = \operatorname{dom} P = \{x \in \mathcal{X} \colon \ |P(x)| < +\infty\}$$

is a nonempty convex subset of $\mathcal{X}$; the operator $\Lambda : \mathcal{X} \to \mathcal{Y}$ is a continuous, generally nonlinear, mapping from $\mathcal{X}$ to another l.c.s. $\mathcal{Y}$, and $\Phi\colon \ \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ is an associated extended function. Since the cost function $P(x)$ is usually nonconvex, the problem ($\mathcal{P}$) may possess many locally extremum (either minimum or maximum) solutions. The goal of *global optimization* is to find all the local extrema of $P(x)$ over the feasible set $\mathcal{X}_k$. Generally speaking, traditional direct approaches and algorithms for solving nonconvex, nonsmooth global optimization problems are usually very difficult. The classical saddle Lagrange duality methods as well as the well-known Fenchel-Rockafellar duality theory can be used mainly for solving convex problems. For nonconvex problems, there exists a so-called *duality gap* between the primal and the classical dual problems.

The *canonical dual transformation method* andassociated *triduality theory* were proposed originally in finite deformation theory [1]. The key idea of this method is to choose a suitable nonlinear operator Λ: $\mathcal{X} \to \mathcal{Y}$ such that $\Phi(x, y)$ is either convex or concave in each of its variables. This method can be used to solve many nonconvex, nonsmooth global optimization problems.

Canonical Dual Transformation

Let $(\mathcal{X}, \mathcal{X}^*)$ be a pair of real linear spaces, placed in duality by a bilinear form $\langle\cdot,\cdot\rangle : \mathcal{X} \times \mathcal{X}^* \to \mathbf{R}$. For a given extended real-valued function $P\colon \mathcal{X} \to \overline{\mathbf{R}}$, the *subdifferential* of P at $\overline{x} \in \mathcal{X}$ is a convex subset $\partial^- P(\overline{x}) \subset \mathcal{X}^*$ such that for each $\overline{x}^* \in \partial^- P(\overline{x})$, we have

$$\left\langle \overline{x}^*, x - \overline{x} \right\rangle \leq P(x) - P(\overline{x}), \quad \forall x \in \mathcal{X}.$$

Dually, the *superdifferential* of P at $\overline{x} \in \mathcal{X}$ is a convex subset $\partial^+ P(\overline{x}) \subset \mathcal{X}^*$ such that for each $\overline{x}^* \in \partial^+ P(\overline{x})$, we have

$$\left\langle \overline{x}^*, x - \overline{x} \right\rangle \geq P(x) - P(\overline{x}), \quad \forall x \in \mathcal{X}.$$

Clearly, we always have $\partial^+ P = -\,\partial^-\,(-P)$. In convex analysis, it is convention that ∂^- is simply written as ∂. In nonconvex analysis, ∂stands for either ∂^- or ∂^+, i. e.

$$\partial = \{\partial^-,\ \partial^+\}.$$

If P is smooth, Gâteaux-differentiable at $\overline{x} \in \mathcal{X}_a \subset \mathcal{X}$, then

$$\partial P(\overline{x}) = \partial^- P(\overline{x}) = \partial^+ P(\overline{x}) = \{DP(\overline{x})\},$$

where $DP : \mathcal{X}_a \to \mathcal{X}^*$ denotes the Gâteaux derivative of P at $\overline{x}$.

Definition 1 The set of functions $P\colon \mathcal{X} \to \overline{\mathbb{R}}$ which are either convex or concave is denoted by $\Gamma(\mathcal{X})$. In particular, let $\check{\Gamma}(\mathcal{X})$ denote the subset of functions $P \in \Gamma(\mathcal{X})$ which are convex and $\widehat{\Gamma}(\mathcal{X})$ the subset of $P \in \Gamma(\mathcal{X})$ which are concave.

The *canonical function space* $\Gamma_G(\mathcal{X}_a)$ is a subset of functions $P \in \Gamma(\mathcal{X}_a)$ which are Gâteaux differentiable on $\mathcal{X}_a \subset \mathcal{X}$ and the duality mapping $DP : \mathcal{X}_a \to \mathcal{X}_a^* \subset \mathcal{X}^*$ is invertible.

The *extended canonical function space* $\Gamma_0(\mathcal{X})$ is a subset of functions $P \in \Gamma(\mathcal{X})$which are either convex, lower semicontinuous or concave, upper semicontinuous, and if P takes the values $\pm\infty$, then P is identically equal to $\pm\infty$.

By the Legendre–Fenchel transformation, the *supconjugate function* of an extended function $P\colon\ \mathcal{X} \to \overline{\mathbb{R}}$ is defined by

$$P^\sharp(x^*) = \sup_{x\in\mathcal{X}}\{\langle x, x^*\rangle - P(x)\}.$$

By the theory of convex analysis, $P^\sharp\colon\ \mathcal{X}^* \to \vec{\mathbb{R}} := \mathbb{R}\cup\{+\infty\}$ is always convex and lower semicontinuous, i. e. $P^\sharp \in \check{\Gamma}_0(\mathcal{X}^*)$. Dually, the *subconjugate function* of P, defined by

$$P^\flat(x^*) = \inf_{x\in\mathcal{X}}\{\langle x, x^*\rangle - P(x)\},$$

is always concave and upper semicontinuous, i. e. $P^\flat \in \widehat{\Gamma}_0(\mathcal{X}^*)$, and $P^\flat = -P^\sharp$. Both the super- and subconjugates are called *Fenchel conjugate functions* and we write $P^* = \{P^\flat, P^\sharp\}$. Thus the extended Fenchel transformation can be written as

$$P^*(x^*) = \text{ext}\,\{\langle x, x^*\rangle - P(x)\colon\ \forall x \in \mathcal{X}\}\,, \tag{2}$$

where ext stands for extremum. Clearly, if $P \in \Gamma_0(\mathcal{X})$, we have the Fenchel equivalent relations, namely,

$$x^* \in \partial P(x) \Leftrightarrow x \in \partial P^*(x^*) \Leftrightarrow P(x) + P^*(x^*) = \langle x, x^*\rangle\,. \tag{3}$$

The pair (x, x^*) is called the *Fenchel duality pair* on $\mathcal{X} \times \mathcal{X}^*$ if and only if equation (3) holds on $\mathcal{X} \times \mathcal{X}^*$.

The conjugate pair (x, x^*) is said to be a *Legendre duality pair* on $\mathcal{X}_a \times \mathcal{X}_a^* \subset \mathcal{X} \times \mathcal{X}^*$ if and only if the equivalent relations

$$x^* = DP(x) \Leftrightarrow x = DP^*(x^*) \Leftrightarrow P(x) + P^*(x^*) = \langle x, x^*\rangle \tag{4}$$

hold on $\mathcal{X}_a \times \mathcal{X}_a^*$.

Let $(\mathcal{Y}, \mathcal{Y}^*)$ be an another pair of locally convex topological real linear spaces paired in separating duality by the second bilinear form $\langle\cdot;\cdot\rangle : \mathcal{Y} \times \mathcal{Y}^* \to \mathbf{R}$. The so-called *geometrical operator* $\Lambda : \mathcal{X} \to \mathcal{Y}$ is a continuous, Gâteaux differentiable operator such that for any given $x \in \mathcal{X}_a \subset \mathcal{X}$, there exists a $y \in \mathcal{Y}_a \subset \mathcal{Y}$ satisfying the *geometrical equation*

$$y = \Lambda(x).$$

The directional derivative of y at $\bar{x}$ in the direction $x \in \mathcal{X}$ is then defined by

$$\delta y(\bar{x}; x) := \lim_{\theta \to 0^+} \frac{y(\bar{x} + \theta x) - y(\bar{x})}{\theta} = \Lambda_t(\bar{x})x,$$

where $\Lambda_t(\bar{x}) = D\Lambda(\bar{x})$ denotes the Gâteaux derivative of the operator Λ at $\bar{x}$. For a given $y^* \in \mathcal{Y}^*$, $G_y^*(x) = \langle \Lambda(x); y^* \rangle$ is a real-valued function of x on $\mathcal{X}$. Its Gâteaux derivative at $\bar{x} \in \mathcal{X}_a$ in the direction $x \in \mathcal{X}$ is

$$\delta G_{y^*}(\bar{x}; x) = \langle \Lambda_t(\bar{x})x; y^* \rangle = \left\langle x, \Lambda_t^*(\bar{x}) y^* \right\rangle,$$

where $\Lambda_t^*(\bar{x}) \colon \mathcal{Y}^* \to \mathcal{X}^*$ is the adjoint operator of Λ_t associated with the two bilinear forms.

Let $\Phi\colon \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ be an extended function such that $P(x) = \Phi(x, \Lambda(x))$. If $\Phi\colon \mathcal{X} \times \mathcal{Y} \to \overline{\mathbb{R}}$ is an extended canonical function, i. e. $\Phi \in \Gamma_0(\mathcal{X}) \times \Gamma_0(\mathcal{Y})$, the duality relations between the paired spaces $(\mathcal{X}, \mathcal{X}^*)$ and $(\mathcal{Y}, \mathcal{Y}^*)$ can be written as

$$x^* \in \partial_x \Phi(x, y), \quad y^* \in \partial_y \Phi(x, y). \tag{5}$$

On the product space $\mathcal{X}_a \times \mathcal{Y}_a \subset \mathcal{X} \times \mathcal{Y}$, if the canonical function $\Phi(x, y)$ is finite and Gâteaux differentiable such that the feasible space $\mathcal{X}_k$ can be written as

$$\mathcal{X}_k = \{x \in \mathcal{X}_a\colon \Lambda(x) \in \mathcal{Y}_a\}, \tag{6}$$

then on $\mathcal{X}_k$, the critical condition $\delta P(\bar{x}; x) = \langle x, DP(\bar{x}) \rangle = 0$, $\forall x \in \mathcal{X}_k$, leads to the Euler equation

$$D_x\Phi(\bar{x}, \Lambda(\bar{x})) + \Lambda_t^*(\bar{x}) D_y\Phi(\bar{x}, \Lambda(\bar{x})) = 0, \tag{7}$$

where $D_x\Phi$ and $D_y\Phi$ denote the partial Gâteaux derivatives of Φ with respect to x and y, respectively. Since $\Phi \in \Gamma_G(\mathcal{X}_a) \times \Gamma_G(\mathcal{Y}_a)$ is a canonical function, the Gâteaux derivative $D\Phi : \mathcal{X}_a \times \mathcal{Y}_a \to \mathcal{X}_a^* \times \mathcal{Y}_a^* \subset \mathcal{X}^* \times \mathcal{Y}^*$ is a monotone mapping, i. e. there exists a pair $(\bar{x}^*, \bar{y}^*) \in \mathcal{X}^* \times \mathcal{Y}^*$ such that

$$-\bar{x}^* = D_x\Phi(\bar{x}, \Lambda(\bar{x})), \quad \bar{y}^* = D_y\Phi(\bar{x}, \Lambda(\bar{x})).$$

Thus, in terms of canonical dual variables $\bar{x}^*$ and $\bar{y}^*$, the Euler equation (7) can be written in the so-called *balance* (or *equilibrium*) equilibrium

$$\bar{x}^* = \Lambda_t^*(\bar{x})\bar{y}^*, \tag{8}$$

which linearly depends on the dual variable $\bar{y}^*$.

Definition 2 Suppose that for a given problem $(\mathcal{P})$, the geometrical operator $\Lambda : \mathcal{X} \to \mathcal{Y}$ can be chosen in such a way that $P(x) = \Phi(x, \Lambda(x))$, $\Phi \in \Gamma_G(\mathcal{X}_a) \times \Gamma_G(\mathcal{Y}_a)$ and $\mathcal{X}_k = \{x \in \mathcal{X}_a : \Lambda(x) \in \mathcal{Y}_a\}$. Then
1) the transformation $\{P; \mathcal{X}_k\} \to \{\Phi; \mathcal{X}_a \times \mathcal{Y}_a\}$ is called the *canonical transformation*, and $\Phi : \mathcal{X}_a \times \mathcal{Y}_a \to \mathbf{R}$ iscalled the *canonical function associated with* Λ;
2) the problem $(\mathcal{P})$ is called *geometrically nonlinear* (respectively, *geometrically linear*) if $\Lambda\colon \mathcal{X} \to \mathcal{Y}$ is nonlinear (respectively, linear); it is called *physically nonlinear* (respectively, *physically linear*) if the duality mapping $D\Phi : \mathcal{X}_a \times \mathcal{Y}_a \to \mathcal{X}_a^* \times \mathcal{Y}_a^*$ is nonlinear (respectively, linear); it is called *fully nonlinear* if it is both geometrically and physically nonlinear.

The canonical transformation plays a fundamental role in duality theory of global optimization. By this definition, the governing equation (7) for fully nonlinear problems canbe written in the *tricanonical forms*, namely,
1) geometrical equation: $y = \Lambda(x)$;
2) physical relations: $(-x^*, y^*) \in \partial\Phi(x, y)$;
3) balance equation: $x^* = \Lambda_t^*(x)\, y^*$.

Since $\Lambda : \mathcal{X} \to \mathcal{Y}$ is Gâteaux differentiable, for any given $x \in \mathcal{X}$ we have the *operator decomposition*

$$\Lambda(x) = \Lambda_t(x)x + \Lambda_c(x), \tag{9}$$

where $\Lambda_c = \Lambda - \Lambda_t$ is the *complementary operator* of Λ_t. By this operator decomposition, the relation between the two bilinear forms reads

$$\langle \Lambda(x); y^* \rangle = \left\langle x, \Lambda_t^*(x) y^* \right\rangle - G(x, y^*),$$

where $G(x, y^*) = \langle -\Lambda_c(x); y^* \rangle$ is the so-called *complementary gap function*, introduced in [2]. This gap plays an important role in the canonical dual transformation methods. A framework for the fully nonlinear system is

$$\begin{tikzcd}
x \in \mathcal{X} \arrow[r, leftrightarrow, "\langle x, x^* \rangle"] \arrow[d, "\Lambda_t + \Lambda_c = \Lambda"'] & \mathcal{X}^* \ni x^* \\
y \in \mathcal{Y} \arrow[r, leftrightarrow, "\langle y; y^* \rangle"] & \mathcal{Y}^* \ni y^* \arrow[u, "\Lambda_t^* = (\Lambda - \Lambda_c)^*"']
\end{tikzcd}$$

Extensive illustrations of the canonical transformation and the tricanonical forms in mathematical physics and variational analysis can be found in [1].

Very often, the extended canonical function Φ can be written in the form

$$\Phi(x, y) = W(y) - F(x),$$

where $F \in \Gamma(\mathcal{X})$ and $W \in \Gamma(\mathcal{Y})$ are extended canonical functions. The duality relations (5) in this special case take the forms

$$x^* \in \partial F(x), \quad y^* \in \partial W(y).$$

If $F \in \Gamma_G(\mathcal{X}_a)$ and $W \in \Gamma_G(\mathcal{Y}_a)$ are Gâteaux differentiable, the Euler equation (7) reads

$$\Lambda_t^*(\overline{x}) DW(\Lambda(\overline{x})) - DF(\overline{x}) = 0.$$

If $\Lambda : \mathcal{X} \to \mathcal{Y}$ is linear, and $W\ \mathcal{Y} \to \mathbf{R}$ is quadratic such that $DW = Cy$, where $C : \mathcal{Y} \to \mathcal{Y}^*$ is a linear operator, then the governing equations for linear system can be written as

$$\Lambda^* C \Lambda x = Ax = x^*.$$

For conservative systems, the operator $A = \Lambda^*\ C\Lambda$ is usually symmetric. In static systems, C is usually positive definite and the associated total potential P is convex. However, in dynamical systems, C is indefinite and P is called the total action, which is usually a d.c. function in convex Hamilton systems.

Triality Theory

We assume that for any given nonconvex extended function $P\colon\ \mathcal{X} \to \overline{\mathbb{R}}$, there exists a general nonlinear operator $\Lambda : \mathcal{X} \to \mathcal{Y}$ and a canonical function $W \in \Gamma(\mathcal{Y})$ suchthat the canonical transformation can be written as

$$P(x) = W(\Lambda(x)) - \langle x, c\rangle\,, \tag{10}$$

where $c \in \mathcal{X}^*$ is a given source variable. Since $F(x) = x, c$ is a linear function, the Hamiltonian $H(x, y^*) = W^*(y^*) + x, c$ is a canonical function on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}^*$ and the extended Lagrangian reads

$$L(x, y^*) = \langle \Lambda(x); y^*\rangle - W^*(y^*) - \langle x, c\rangle\,. \tag{11}$$

For a fixed $y^* \in \mathcal{Y}^*$, the convexity of $L(\cdot, y^*)\colon\ \mathcal{X} \to \overline{\mathbb{R}}$ depends on $\Lambda(x)$ and $y^* \in \mathcal{Y}^*$.

Let $\mathcal{Z}_a = \mathcal{X}_a \times \mathcal{Y}_a^* \subset \mathcal{Z}$ be the effectivedomain of L, and let $\mathcal{L}_c \subset \mathcal{Z}_a$ be a critical point set of L, i. e.

$$\mathcal{L}_c = \left\{(\overline{x}, \overline{y}) \in \mathcal{X}_a \times \mathcal{Y}_a^*\colon\ DL(\overline{x}, \overline{y}^*) = 0\right\}.$$

For any given critical point $(\overline{x}, \overline{y}^*) \in \mathcal{L}_c$, we let $\mathcal{X}_r \times \mathcal{Y}_r^*$ be its neighborhood such that on $\mathcal{X}_r \times \mathcal{Y}_r^*$, the pair $(\overline{x}, \overline{y}^*)$ is the only critical point of L. The following resultis of fundamental importance in global optimization.

Theorem 3 **(*Triality theorem*)** *Suppose that $W \in \check{\Gamma}(\mathcal{Y}_a)$ is convex, $(\overline{x}, \overline{y}^*) \in \mathcal{L}_c$ is a critical point of L and $\mathcal{X}_r \times \mathcal{Y}_r^*$ is a neighborhood of $(\overline{x}, \overline{y}^*)$.*

If $\left\langle \Lambda(x); \overline{y}^*\right\rangle$ is convex on $\mathcal{X}_r$, then

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_r} \max_{y^* \in \mathcal{Y}_r^*} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}_r^*} \min_{x \in \mathcal{X}_r} L(x, y^*). \end{aligned} \tag{12}$$

However, if $\left\langle \Lambda(x); \overline{y}^*\right\rangle$ is concave on $\mathcal{X}_r$, then either

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \min_{x \in \mathcal{X}_r} \max_{y^* \in \mathcal{Y}_r^*} L(x, y^*) \\ &= \min_{y^* \in \mathcal{Y}_r^*} \max_{x \in \mathcal{X}_r} L(x, y^*)\,, \end{aligned} \tag{13}$$

or

$$\begin{aligned} L(\overline{x}, \overline{y}^*) &= \max_{x \in \mathcal{X}_r} \max_{y^* \in \mathcal{Y}_r^*} L(x, y^*) \\ &= \max_{y^* \in \mathcal{Y}_r^*} \max_{x \in \mathcal{X}_r} L(x, y^*). \end{aligned} \tag{14}$$

Since $W \in \Gamma(\mathcal{Y}_a)$ is a canonical function, we always have

$$P(x) = \operatorname{ext}\{L(x, y^*)\colon\ y^* \in \mathcal{Y}^*\}\,, \quad \forall x \in \mathcal{X}_k. \tag{15}$$

On the other hand, for a given Gâteaux differentiable geometrical mapping $\Lambda : \mathcal{X}_a \to \mathcal{Y}_a$, the criticality condition $D_x L(\overline{x}, y^*) = 0$ leads to the equilibrium equation

$$\Lambda_t^*(\overline{x}) y^* = c. \tag{16}$$

If there exists a subspace $\mathcal{Y}_s^* \subset \mathcal{Y}_a^*$ such that for any $y^* \in \mathcal{Y}_s^*$ and a given source variable $c \in \mathcal{X}^*$, the equation (16) can be solved for $\overline{x} = \overline{x}(y^*)$, then by the operator decomposition (9), the dual function $P^d : \mathcal{Y}_s^* \to \mathbf{R}$ can be written explicitly in the form

$$\begin{aligned} P^d(y^*) &= \operatorname{sta}\{L(x, y^*)\colon\ x \in \mathcal{X}\} \\ &= -G^d(y^*) - W^*(y^*), \quad \forall y^* \in \mathcal{Y}_s^*, \end{aligned}$$

where $G^d : \mathcal{Y}^* \to \mathbf{R}$ is the so-called *pure complementary gap function*, defined by

$$G^d(y^*) = G(\bar{x}(y^*), y^*) = -\langle \Lambda_c(\bar{x}(y^*)); y^* \rangle .$$

For any given critical point $(\bar{x}, \bar{y}^*) \in \mathcal{L}_c$, we have $G^d(\bar{y}^*) = \langle \bar{x}, c\rangle - \langle \Lambda(\bar{x}(\bar{y}^*)); \bar{y}^* \rangle$. Thus, the Legendre duality relations among the canonical functions W and W^* lead to

$$P(\bar{x}) - P^d(\bar{y}^*) = 0, \quad \forall (\bar{x}, \bar{y}^*) \in \mathcal{L}_c. \tag{17}$$

This identity shows that there is no duality gap between the nonconvex function P and its canonical dual function P^d. Actually the duality gap, which exists in classical duality theories, is now recovered by the complementary gap function $G(\bar{x}, \bar{y}^*)$.

Theorem 4 (*Triduality theorem*) *Suppose that $W \in \check{\Gamma}(\mathcal{Y}_a)$ is a critical point of L and $\mathcal{X}_r \times \mathcal{Y}_r^*$ is a neighborhood of $(\bar{x}, \bar{y}^*)$. If $\langle \Lambda(x); \bar{y}^* \rangle$ is convex on $\mathcal{X}_r$, then*

$$P(\bar{x}) = \min_{x \in X_r} P(x) \Leftrightarrow P^d(\bar{y}^*) = \max_{y^* \in \mathcal{Y}_r^*} P^d(y^*).$$

However, if $\langle \Lambda(x); \bar{y}^* \rangle$ is concave on $\mathcal{X}_r$, then

$$P(\bar{x}) = \min_{x \in X_r} P(x) \Leftrightarrow P^d(\bar{y}^*) = \min_{y^* \in \mathcal{Y}_r^*} P^d(y^*);$$

$$P(\bar{x}) = \max_{x \in X_r} P(x) \Leftrightarrow P^d(\bar{y}^*) = \max_{y^* \in \mathcal{Y}_r^*} P^d(y^*).$$

Example 5 We now illustrate the application of the interesting triduality theory for solving the following nonconvex optimization problem in $\mathcal{X} = \mathbf{R}^n$,

$$P(x) = \frac{a}{2}(\frac{1}{2}\|Ax\|^2 - \mu)^2 - x^\top c \to \text{sta}, \quad \forall x,$$

where a, $\mu > 0$ are given parameters, $c \in \mathbf{R}^n$ is a given vector, and $A : \mathbf{R}^n \to \mathbf{R}^m$ is a matrix. The Euler equation associated with this nonconvex stationary problem is a nonlinear algebraic equation in $\mathbf{R}^n$

$$a(\frac{1}{2}\|A\bar{x}\|^2 - \mu)C\bar{x} = c,$$

where $C = A^\top A = C^\top \in \mathbf{R}^{nn}$. We are interested in finding all the critical points of P. To set this nonconvex problem in our framework, we let $\mathcal{X} = \mathbf{R}^n = \mathcal{X}^*$, and $\Lambda : \mathbf{R}^n \to \mathcal{Y} = \mathrm{R}$ a quadratic operator

$$y = \Lambda(x) = \frac{1}{2}\|Ax\|^2 - \mu = \frac{1}{2}x^\top Cx - \mu.$$

Since $F(x) = \langle x, c\rangle = x^\top c$ is a linear function on $\mathbf{R}^n$, the admissible space $\mathcal{X}_a = \mathcal{X} = \mathbf{R}^n$. By the fact that $x^* = DF(x) = c$, the range for the canonical mapping $DF : \mathcal{X} \to \mathcal{X}^* = \mathbf{R}$ is a hyperplane in $\mathbf{R}^n$, i. e.

$$\mathcal{X}_a^* = \{x^* \in \mathbb{R}^n : \ x^* = c\} .$$

The feasible set for the primal problem is $\mathcal{X}_k = \{x \in \mathcal{X}_a : \Lambda(x) \in \mathcal{Y}_a\} = \mathbf{R}^n$.

By the fact that $x^\top Cx \geq 0$, $\forall x \in \mathcal{X}_a = \mathcal{X} = \mathbf{R}^n$, the range for the geometrical mapping Λ: $\mathcal{X}_a \to \mathbf{R}$ is a closed convex set in $\mathbf{R}$

$$\mathcal{Y}_a = \{y \in \mathbb{R} : \ y \geq -\mu\} \subset \mathcal{Y} = \mathbb{R}.$$

On the admissible subset $\mathcal{Y}_a \subset \mathcal{Y} = \mathbf{R}$, the canonical function $W(y) = (1/2)ay^2$ is quadratic. The range for the constitutive mapping DW: $\mathcal{Y}_a \to \mathcal{Y}^* = \mathbf{R}$ is also a closed convex set in $\mathbf{R}$,

$$\mathcal{Y}_a^* = \{y^* \in \mathbb{R} : \ y^* \geq -a\mu\} .$$

On $\mathcal{Y}_a^*$, the Legendre conjugate of W isalso strictly convex

$$W^*(y^*) = \frac{1}{2}a^{-1}y^{*2}, \tag{18}$$

and the Legendre duality relations hold on $\mathcal{Y}_a \times \mathcal{Y}_a^*$.

On $\mathcal{X}_a \times \mathcal{Y}_a^* = \mathbf{R}^n \times \mathbf{R}$, the extended Lagrangian in this case reads

$$L(x, y^*) = \frac{1}{2}y^* x^\top Cx - \mu y^* - \frac{1}{2}a^{-1}y^{*2} - x^\top c.$$

It is easy to check that the dual function associated with L is

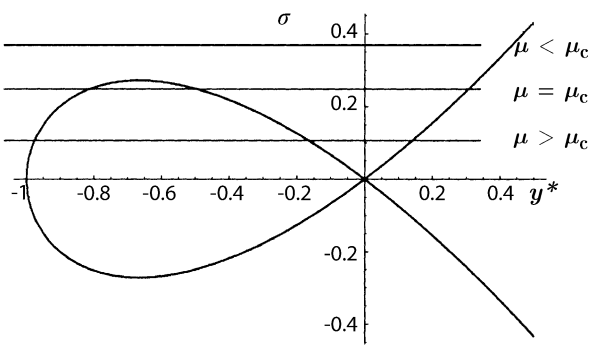
$$P^d(y^*) = \frac{1}{2}(y^*)^{-1}c^\top Cc - \mu y^* - \frac{1}{2a}y^{*2}.$$

The dual Euler–Lagrange equation is an algebraic equation in $\mathbf{R}$:

$$(\mu + a^{-1}y^*)y^{*2} = \frac{1}{2}\sigma^2, \tag{19}$$

where $\sigma^2 = c^\top Cc$ is a constant. Since $C \in \mathbf{R}^{nn}$ is positive definite, this equation holds only on $\mathcal{Y}_a^*$.

In algebraic geometry, the dual Euler–Lagrange equation (19) is the so-called singular algebraic curve in (y^*, σ)-space (see Fig. 1). For a given parameter μ and c



Duality Theory: Triduality in Global Optimization, Figure 1
Singular algebraic curve

$\in \mathbf{R}^n$, this dual equation has at most three real roots $y_k^* \in \mathcal{Y}_a^*$, $k = 1, 2, 3$, which leads to the primal solution

$$x_k = y_k^* C^+ c, \quad k = 1, 2, 3,$$

where C^+ stands for the generalized inverse of C. We know that each (x_k, y_k^*) is a critical point of L and

$$P(x_k) = L(x_k, y_k^*) = P^d(y_k^*), \quad k = 1, 2, 3.$$

In the case of $n = 1$, the cost function

$$P(x) = \frac{1}{2} a \left(\frac{1}{2} x^2 - \mu \right)^2 - cx$$

is a *double-well function* (see Fig. 2, solid line), which appears in many physical systems. The graph of the canonical dual function

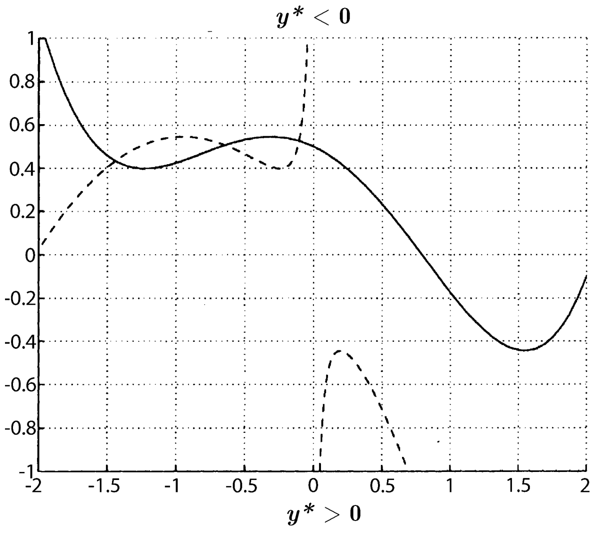
$$P^d(y^*) = \frac{1}{2} \frac{c^2}{y^*} - \mu y^* - \frac{y^{*2}}{2a}$$

has two branches (Fig. 2, dashed line). It is easy to prove (see [1]) that if $\mu > \mu_c = 1.5\,(\sigma/a)^{2/3}$, the dual Euler–Lagrange equation (19) has three roots $y_1^* > 0 > y_2^* > y_3^*$, corresponding to three critical points of P^d (see Fig. 2). Then, y_1^* is a global maximizer of P^d, $x_1 = \sigma/y_1^*$ is a global minimizer of P, P^d takes local minimum and local maximum values at y_2^* and y_3^*, respectively, $x_2 = \sigma/y_2^*$ is a local maximizer of P, while $x_3 = \sigma/y_3^*$ is a local minimizer.

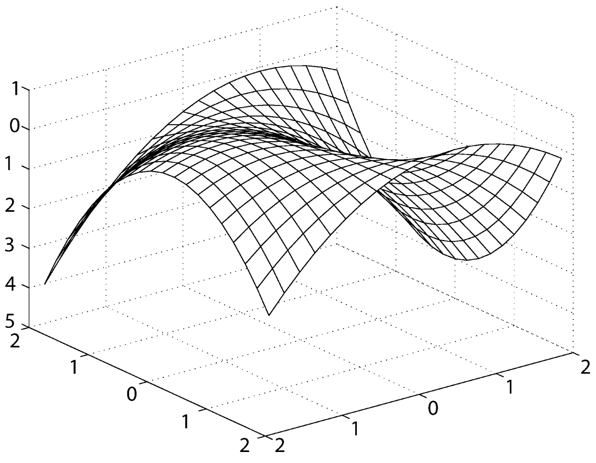
The Lagrangian associated with this double-well energy is

$$L(x, y^*) = \frac{1}{2} x^2 y^* - (\frac{1}{2a} y^{*2} + \mu y^*) - y^* x.$$

It is a saddle function for $y^* > 0$. If $y^* < 0$, it is a super-critical point function (see Fig. 3).



Duality Theory: Triduality in Global Optimization, Figure 2
Graphs of *P*(*u*) and itsdual *P*^{d}(*y)**



Duality Theory: Triduality in Global Optimization, Figure 3
Lagrangian for the double-well energy

See also

- ► Duality Theory: Biduality in Nonconvex Optimization
- ► Duality Theory: Monoduality in Convex Optimization
- ► History of Optimization
- ► Von Neumann, John

References

1. Gao DY (1999) Duality principles in non-convex systems: Theory, methods and applications. Kluwer, Dordrecht

2. Gao DY, Strang G (1989) Geometrical nonlinearity: Potential energy, complementary energy, and the gap functional. Quart Appl Math XLVII(3):487–504

Dykstra's Algorithm and Robust Stopping Criteria

Ernesto G. Birgin[1], Marcos Raydan[2]
[1] Department of Computer Science IME-USP, University of São Paulo, São Paulo, Brazil
[2] Departamento de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela

MSC2000: 90C25, 65K05, 65G505

Article Outline

Keywords

Convex optimization; Alternating projection methods; Dykstra's algorithm; Stopping criteria

Introduction

We consider the application of Dykstra's algorithm for solving the following optimization problem

$$\min_{x\in\Omega} \|x^0 - x\| , \qquad (1)$$

where x^0 is a given point, Ω is a closed and convex set, and $\|z\|^2 = \langle z, z\rangle$ defines a real inner product in the space. The solution x^* is called the projection of x^0 onto Ω and is denoted by $P_\Omega(x^0)$. Dykstra's algorithm for solving (1) has been extensively studied since it fits in many different applications (see [5,21,22,23,27,28,29,32,24,42,42,45]).

For simplicity, we consider the case

$$\Omega = \cap_{i=1}^{p} \Omega_i , \qquad (2)$$

where Ω_i are closed and convex sets in $\mathbb{R}^n$, for $i = 1, 2, \dots, p$, and $\Omega \neq \emptyset$. Moreover, we assume that for any $z \in \mathbb{R}^n$ the calculation of $P_\Omega(z)$ is not trivial; whereas, for each Ω_i, $P_{\Omega_i}(z)$ is easy to obtain as in the case of a box, an affine subspace, or a sphere. For the not feasible case (i. e., when $\Omega = \emptyset$) the behavior of Dykstra's algorithm is treated in [2,6,37].

Dykstra's alternating projection algorithm is a cyclic scheme for finding asymptotically the projection of a given point onto the intersection of a finite number of closed convex sets. Roughly speaking, it iterates by projecting in a clever way onto each of the convex sets individually. The algorithm was originally proposed by Dykstra [20] for closed and convex cones in the Euclidean space $\mathbb{R}^n$, and later extended by Boyle and Dykstra [7] for closed and convex sets in a Hilbert space. It was rediscovered by Han [30] using duality theory, and the linear rate of convergence was established by Deutsch and Hundal [18] for the polyhedral case (see also [19,43,44]).

Dykstra's algorithm belongs to the general family of alternating projection methods, that dates back to von Neumann [46] who treated the problem of finding the projection of a given point in a Hilbert space onto the intersection of two closed subspaces. Later, Cheney and Goldstein [15] extended the analysis of von Neumann's alternating projection scheme to the case of two closed and convex sets. In particular, they established convergence under mild assumptions. However, the limit point need not be the closest in the intersection. Therefore, the alternating projection method, proposed by von Neumann, is not useful for problem (1). Fortunately, Dykstra [20] found the clever modification of von Neumann's scheme for which convergence to the solution point is guaranteed. For a complete discussion on alternating projection methods see Deutsch [17].

Dykstra's algorithm has been extended in several different ways. Gaffke and Mathar [24] proposed, via duality, a family of simultaneous Dykstra's algorithm in Hilbert space. Later Iusem and De Pierro [37] established the convergence of the simultaneous version considering also the inconsistent case in the Euclidean space $\mathbb{R}^n$. Bauschke and Borwein [2] further analyzed Dykstra's algorithm for two sets, that appears frequently in applications and in particular generalized the results in [37]. In [36] it was established that for linear inequality constraints the method of Dykstra reduces to the method proposed by Hildreth [33] in his pioneer work on dual alternating projections. See also [40] for further analysis and extensions.

Dykstra's algorithm has also been generalized by Deutsch and Hundal [35] to an infinite family of sets, and also to allow a random ordering, instead of cyclic, of the projections onto the closed convex sets. More recently, it has also been generalized by Bregman et al. [9] to avoid the projection onto each one of the convex sets in every cycle. Instead, projections onto either a suitable half space of the intersection of two half spaces are used. Further results concerning the connection between Bregman distances and Dykstra's algorithm can be found in [3,4,8,14]. For the advantages of projecting cyclically onto suitable half spaces, see the previous work by Iusem and Svaiter [38,39].

A computational experiment comparing Dykstra's algorithm and the Halpern-Lions-Wittmann-Bauschke algorithm [1] on linear best approximation test problems can be found in [12].

Formulations

Dykstra's Algorithm

Dykstra's algorithm solves (1), (2) by generating two sequences: the iterates $\{x_i^k\}$ and the increments $\{y_i^k\}$. These sequences are defined by the following recursive formulae:

$$\begin{aligned} x_0^k &= x_p^{k-1}, \\ x_i^k &= P_{\Omega_i}(x_{i-1}^k - y_i^{k-1}), \qquad i = 1,2,\dots,p\,, \\ y_i^k &= x_i^k - (x_{i-1}^k - y_i^{k-1}), \qquad i = 1,2,\dots,p\,, \end{aligned} \tag{3}$$

for $k = 1, 2, \dots$ with initial values $x_p^0 = x^0$ and $y_i^0 = 0$ for $i = 1, 2, \dots, p$.

Remarks

1. For the sake of simplicity, the projecting control index $i(k)$ used in (3) is the most common one: $i(k) = k \bmod p + 1$, for all $k \geq 0$. However, more advanced control indices can also be used, as long as they satisfy some minimal theoretical requirements (see e. g., [35]).
2. The increment y_i^{k-1} associated with Ω_i in the previous cycle is always subtracted before projecting onto Ω_i. Only one increment (the last one) for each Ω_i needs to be stored.
3. If Ω_i is a closed affine subspace, then the operator P_{Ω_i} is linear and it is not required, in the kth cycle, to subtract the increment y_i^{k-1} before projecting onto Ω_i. Thus, for affine subspaces, Dykstra's procedure reduces to the alternating projection method of von Neumann [46].
4. For $k = 1, 2, \dots$ and $i = 1, 2, \dots, p$, it is clear from (3) that the following relations hold

$$x_p^{k-1} - x_1^k = y_1^{k-1} - y_1^k\,, \tag{4}$$

$$x_{i-1}^k - x_i^k = y_i^{k-1} - y_i^k\,, \tag{5}$$

where $x_p^0 = x^0$ and $y_i^0 = 0$, for all $i = 1, 2, \dots, p$.

For the sake of completeness we now present the key theorem associated with Dykstra's algorithm.

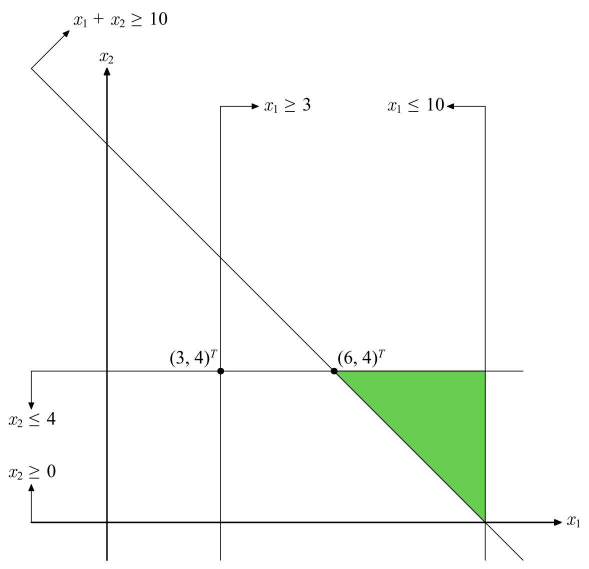
Theorem 1 Boyle and Dykstra, 1986 [7] *Let $\Omega_1, \dots, \Omega_p$ be closed and convex sets of $\mathbb{R}^n$ such that $\Omega = \cap_{i=1}^p \Omega_i \neq \emptyset$. For any $i = 1, 2, \dots, p$ and any $x^0 \in \mathbb{R}^n$, the sequence $\{x_i^k\}$ generated by (3) converges to $x^* = P_\Omega(x^0)$ (i. e., $\|x_i^k - x^*\| \to 0$ as $k \to \infty$).*

We now discuss the delicate issue of stopping Dykstra's algorithm within a certain previously established tolerance that indicates the distance of the current iterate to the unique solution.

Difficulties with some Commonly Used Stopping Criteria

In some applications it is possible to obtain a *somehow* natural stopping rule, associated with the problem at hand. For example, when solving a linear system, $Ax = b$, by alternating projection methods [10,25], the residual vector ($r(x) = b - Ax$) is usually available and yields some interesting and robust stopping rules. Another example appears in image reconstruction for which a *good and feasible* image tells the user that it is time to stop the process [13,16]. Similar circumstances are present in some other specific applications (e. g. saddle point problems [31], and molecular biology [28,29]).

However, in general, this is not the case, and we are left with the information produced only by the internal computations, i. e., the sequence of iterates and perhaps the sequence of increments, and some inner products. For this general case, a popular stopping rule



Dykstra's Algorithm and Robust Stopping Criteria, Figure 1
Feasible set $\boldsymbol{\Omega} = \boldsymbol{\Omega}_1 \cap \boldsymbol{\Omega}_2$ in $\mathbb{R}^2$

is to monitor the subsequence of projections onto one particular convex set, Ω_i, and stop the process when the distance, in norm, of two consecutive projections is less than or equal to a previously established tolerance [26,27,32,41].

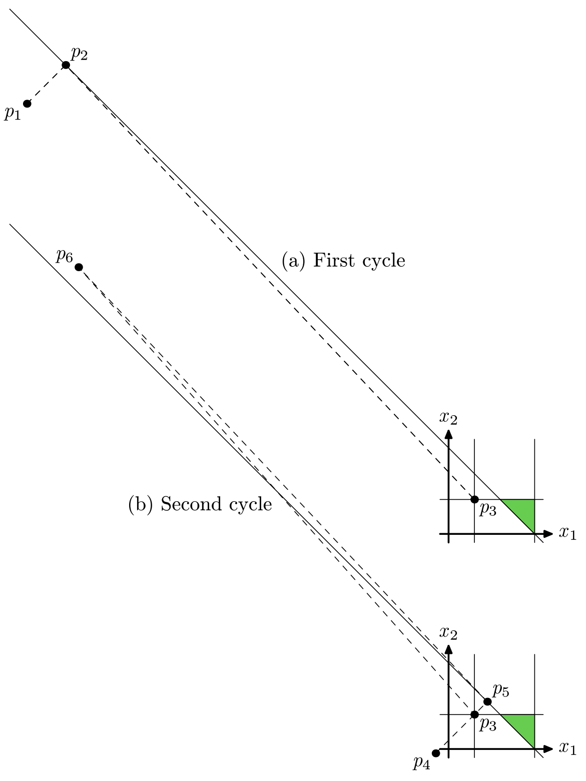
Another commonly used criterion, that is claimed to improve the previous one (e. g. [7,22,28,45]) is to somehow compute an average of all the projections at each cycle of projections, and then stop the process when the distance, in norm, of two consecutive of those average projections is less than or equal to a previously established tolerance.

Finally, we would like to mention that another criterion, that is also designed to improve any of the two criteria above, is to check any of the previously described rules during N consecutive cycles, where N is a fixed positive integer.

None of these stopping rules is a trustable choice. In [6], Birgin and Raydan presented the example below to establish that they can fail even for a two dimensional problem. (see Figs. 1 and 2).

Consider the closed and convex set $\Omega = \Omega_1 \cap \Omega_2$, where $\Omega_1 = \{x \in \mathbb{R}^2 \mid x_1 + x_2 \geq 10\}$ is a half space and $\Omega_2 = \{x \in \mathbb{R}^2 \mid 3 \leq x_1 \leq 10, 0 \leq x_2 \leq 4\}$ is a box. This closed and convex set in $\mathbb{R}^2$ is shown in Fig. 1.

Let $x^0 = (-49, 50)^T$ and let us use Dykstra's algorithm to find the closest point to x^0 in Ω. In Fig. 2



Dykstra's Algorithm and Robust Stopping Criteria, Figure 2
First two cycles of Dykstra's algorithm to find the projection of $x^0 = (-49, 50)^T$ onto $\boldsymbol{\Omega} = \boldsymbol{\Omega}_1 \cap \boldsymbol{\Omega}_2$

we can see the first two cycles of this convergent process. Since $y_1^0 = y_2^0 = 0$ (null initial increments) then for the first cycle we project x^0 onto Ω_1 to obtain $p_2 = x_1^1 = (-44.5, 54.5)^T$ and then we project p_2 onto Ω_2 to obtain $p_3 = x_2^1 = (3,4)^T$. For the second cycle, the increments are not null ($y_1^1 = (4.5, 4.5)^T$ and $y_2^1 = (47.5, -50.5)^T$), and we start from p_3. First we project $p_4 = p_3 - y_1^1$ onto Ω_1 to obtain $p_5 = x_1^2$. Then we project $p_6 = p_5 - y_2^1$ onto Ω_2 to obtain p_3 again. Hence $x_2^2 = x_2^1$. The increment associated with Ω_2 is large enough to take the iterate back to the quadrant where the projection onto the box is again p_3. As discussed in [6], this phenomenon will occur until cycle 32, i. e., $p_3 = x_2^1 = x_2^2 = \cdots = x_2^{32}$.

Moreover, by choosing x^0 far enough, this misleading event can be repeated for as many cycles as any previously established positive integer N. Eventually the size of the increments will be reduced and convergence to x^* will be observed.

Robust Stopping Criteria

After a close inspection of the proof of the Boyle and Dykstra's theorem, Birgin and Raydan [6] proposed some robust stopping criteria for Dykstra's algorithm. For that they first established the following result.

Theorem 2 *Let x^0 be any element of $\mathbb{R}^n$. Consider the sequences $\{x_i^k\}$ and $\{y_i^k\}$ generated by (3) and define c^k as*

$$c^k = \sum_{m=1}^{k}\sum_{i=1}^{p}\|y_i^{m-1} - y_i^m\|^2 + 2\sum_{m=1}^{k-1}\sum_{i=1}^{p}\langle y_i^m, x_i^{m+1} - x_i^m\rangle \,. \quad (6)$$

Then, in the kth cycle of Dykstra's algorithm,

$$\|x^0 - x^*\|^2 \geq c^k \,. \quad (7)$$

Moreover, at the limit when k goes to infinity, equality is attained in (7).

Based on the previous theorem, let us now write c^k as follows:

$$c^k = c_L^k + c_S^k \,,$$

where

$$c_L^k = \sum_{m=1}^{k} c_I^m \,, \quad (8)$$

$$c_I^m = \sum_{i=1}^{p}\|y_i^{m-1} - y_i^m\|^2 \quad (9)$$

and

$$c_S^k = 2\sum_{m=1}^{k-1}\sum_{i=1}^{p}\langle y_i^m, x_i^{m+1} - x_i^m\rangle \,.$$

Both c_L^k and c_S^k are monotonically nondecreasing by definition. Moreover in [6], the following theorem is also established.

Theorem 3 *Consider the sequences $\{x_i^k\}$ and $\{y_i^k\}$ generated by (3), and c^k, c_L^k and c_I^k as defined in (6), (8) and (9), respectively. For any $k \in \mathbb{N}$, if $x^k \neq x^*$ then $c_I^{k+1} > 0$ and, hence, $c_L^k < c_L^{k+1}$ and $c^k < c^{k+1}$.*

The results established in Theorems 2 and 3 are combined in [6] to propose robust stopping criteria. Notice that $\{c_L^k\}$ and $\{c^k\}$ are monotonically increasing and convergent, and also that $\{c_I^k\}$ converges to zero. Therefore we can stop the process when

$$c_I^k = \sum_{i=1}^{p}\|y_i^{k-1} - y_i^k\|^2 \leq \varepsilon$$

or, similarly, when

$$c^k - c^{k-1} = c_I^k + 2\sum_{i=1}^{p}\langle y_i^{k-1}, x_i^k - x_i^{k-1}\rangle \leq \varepsilon \,, \quad (10)$$

where $\varepsilon > 0$ is a sufficiently small tolerance. As c^k may grow fast, computing $c^k - c^{k-1}$ may give inaccurate results due to loss of accuracy in floating point representation and, hence, cancellation. So, for the criterion in (10), it is recommendable to test convergence with the second expression.

The computation of c_I^k involves the squared-norm $\|y_i^{k-1} - y_i^k\|^2$, for $i = 1, 2, \ldots, p$. By (5), $y_i^k = y_i^{k-1} + v$, where $v = x_i^k - x_{i-1}^k$ is a temporary n-dimensional array needed in the computation of Dykstra's algorithm. So, the computational cost involved in the calculation of c_I^k is just the cost of the extra inner product $\langle v, v\rangle$ at each iteration.

The computation of c^k involves the calculation of c_I^k plus an extra term. The computational of this extra term is also small and involves an inner product and the difference of two vectors per iteration. But, in contrast with the computation of c_I^k which does not require additional savings, the computation of the extra term requires to save p extra n-dimensional arrays (the same amount of memory required in Dykstra's algorithm to save the increments). So, the computation of c^k requires some additional calculations and memory savings, and hence it is more expensive. However, it also has the advantage of revealing the optimal distance: $\|x^0 - x^*\|^2$, that could be of interest in some applications.

We close this section with some comments concerning the behavior of the stopping criteria when the problem is not feasible. In this case ($\Omega = \emptyset$), there is no solution and we know from Theorem 3 that the sequences $\{c_L^k\}$ and $\{c^k\}$ are monotonically increasing. Moreover, under some mild assumptions on the sets Ω_i, the sequences $\{x_i^k\}$ converge for $1 \leq i \leq p$, and there exists a real constant $\delta > 0$ such that $\sum_{i=1}^{p}\|x_{i-1}^k - x_i^k\|^2 \geq \delta$ for all k. A discussion on this topic is presented in [2, Section 6], including a notion of *distance* between all the sets Ω_i (see also [37]). Now using (5), we obtain

$$\sum_{i=1}^{p} \|x_{i-1}^k - x_i^k\|^2 = \sum_{i=1}^{p} \|y_i^{k-1} - y_i^k\|^2 = c_I^k \,.$$

Therefore, the sequence $\{c_I^k\}$ remains bounded away from zero, whereas $\{c_L^k\}$ and $\{c^k\}$ tend to infinity. Consequently, none of the proposed stopping criteria will be satisfied for any k, as expected.

References

1. Bauschke HH (1996) The approximation of fixed points of compositions of nonexpansive mappings in Hilbert space. J Math Anal Appl 202:150–159
2. Bauschke HH, Borwein JM (1994) Dykstra's alternating projection algorithm for two sets. J Approx Theory 79:418–443
3. Bauschke HH, Borwein JM (1997) Legendre functions and the method of random Bregman projections. J Convex Anal 4:27–67
4. Bauschke HH, Lewis AS (2000) Dykstra's algorithm with Bregman projections: a convergence proof. Optim 48:409–427
5. Birgin EG, Martínez JM, Raydan M (2003) Inexact spectral gradient method for convex-constrained optimization. IMA J Numer Anal 23:539–559
6. Birgin EG, Raydan M (2005) Robust stopping criteria for Dykstra's algorithm. SIAM J Sci Comput 26:1405–1414
7. Boyle JP, Dykstra RL (1986) A method for finding projections onto the intersection of convex sets in Hilbert spaces. Lect Notes Stat 37:28–47
8. Bregman LM, Censor Y, Reich S (1999) Dykstra's algorithm as the nonlinear extension of Bregman's optimization method. J Convex Anal 6:319–333
9. Bregman LM, Censor Y, Reich S, Zepkowitz-Malachi Y (2003) Finding the projection of a point onto the intersection of convex sets via projections onto halfspaces. J Approx Theory 124:194–218
10. Bramley R, Sameh A (1992) Row projection methods for large non symmetric linear systems. J Sci Statist Comp 13:168–193
11. Bregman LM, Censor Y, Reich S (1999) Dykstra's algorithm as the nonlinear extension of Bregman's optimization method. J Convex Anal 6:319–333
12. Censor Y (2006) Computational acceleration of projection algorithms for the linear best approximation problem. Linear Algebr Appl 416:111–123
13. Censor Y, Herman GT (1987) On some optimization techniques in image reconstruction from projections. Appl Numer Math 3:365–391
14. Censor Y, Reich S (1998) The Dykstra's algorithm with Bregman projections. Commun Appl Anal 2:407–419
15. Cheney W, Goldstein A (1959) Proximity maps for convex sets. Proc Am Math Soc 10:448–450
16. Combettes PL (1996) The convex feasibility problem in image recovery. Adv Imag Electron Phys 95:155–270
17. Deutsch F (2001) Best Approximation in Inner Product Spaces. Springer, New York
18. Deutsch F, Hundal H (1994) The rate of convergence of Dykstra's cyclic projections algorithm: the polyhedral case. Numer Funct Anal Optim 15:537–565
19. Deutsch F (1995) Dykstra's cyclic projections algorithm: the rate of convergence. In: Singh SP (ed) Approximation Theory, Wavelets and Applications. Kluwer, Dordrecht, pp 87–94
20. Dykstra RL (1983) An algorithm for restricted least-squares regression. J Am Stat Assoc 78:837–842
21. Eberle MG, Maciel MC (2003) Finding the Closest Toeplitz Matrix. Comput Appl Math 22:1–18
22. Escalante R, Raydan M (1996) Dykstra's Algorithm for a Constrained Least-Squares Matrix Problem. Numer Linear Algebr Appl 3:459–471
23. Escalante R, Raydan M (1998) On Dykstra's algorithm for constrained least-squares rectangular matrix problems. Comput Math Appl 35:73–79
24. Gaffke N, Mathar R (1986) A cyclic projection algorithm via duality. Metr 36:29–54
25. Garcia-Palomares UM (1999) Preconditioning projection methods for solving algebraic linear systems. Numer Algorithm 21:157–164
26. Glunt W (1995) An alternating projection method for certain linear problems in a Hilbert space. IMA J Numer Anal 15:291–305
27. Glunt W, Hayden TL, Hong S, Wells J (1990) An alternating projection algorithm for computing the nearest Euclidean distance matrix. SIAM J Matrix Anal Appl 11: 589–600
28. Glunt W, Hayden TL, Raydan M (1993) Molecular conformations from distance matrices. J Comput Chem 14:114–120
29. Glunt W, Hayden TL, Liu WM (1991) The embedding problem for predistance matrices. Bull Math Biol 53:769–796
30. Han SP (1988) A successive projection method. Math Program 40:1–14
31. Hernández-Ramos LM (2005) Alternating oblique projections for coupled linear systems. Numer Algorithm 38:285–303
32. Higham N (2002) Computing the nearest correlation matrix – a problem from finance. IMA J Numer Anal 22:329–343
33. Hildreth C (1957) A quadratic programming procedure. Naval Res Log Quart 4:79–85
34. Hu H, Olkin I (1991) A numerical procedure for finding the positive definite matrix closest to a patterned matrix. Stat Probab Lett 12:511–515
35. Hundal H, Deutsch F (1997) Two generalizations of Dykstra's cyclic projections algorithm. Math Program 77:335–355

36. Iusem A, De Pierro A (1990) On the convergence properties of Hildreth's quadratic programming algorithm. Math Prog 47:37–51
37. Iusem A, De Pierro A (1991) On the convergence of Han's method for convex programming with quadratic objective. Math Prog 52:265–284
38. Iusem A, Svaiter BF (1995) A row-action method for convex programming. J Math Prog 64:149–171
39. Iusem A, Svaiter BF (1995) Primal-dual row-action method for convex programming. J Optim Theor Appl 86:73–112
40. Lent A, Censor Y (1980) Extensions of Hildreth's row-action method for quadratic programming. SIAM J Control Optim 18:444–454
41. Mendoza M, Raydan M, Tarazaga P (1998) Computing the nearest diagonally dominant matrix. Numer Linear Algebr Appl 5:461–474
42. Monsalve M, Moreno J, Escalante R, Raydan M (2003) Selective alternating projections to find the nearest SSD+ matrix. Appl Math Comput 145:205–220
43. Morillas PM (2005) Dykstra's algorithm with strategies for projecting onto certain polyhedral cones. Appl Math Comput 167:635–649
44. Perkins C (2002) A Convergence Analysis of Dykstra's Algorithm for Polyhedral Sets. SIAM J Numer Anal 40:792–804
45. Raydan M, Tarazaga P (2002) Primal and polar approach for computing the symmetric diagonally dominant projection. Numer Linear Algebr Appl 9:333–345
46. von Neumann J (1950) Functional operators vol. II. The geometry of orthogonal spaces. Annals of Mathematical Studies 22, Princeton University Press. This is a reprint of mimeographed lecture notes first distributed in 1933

Dynamic Programming: Average Cost Per Stage Problems

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L99

Article Outline

Keywords

Dynamic programming; Infinite horizon problems; Average cost per stage

Dynamic programming deals with optimal decision making problems in the presence of uncertainty that addresses systems in which events occur sequential. In general the state transitions are described by stationary dynamic systems of the form:

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, \dots,$$

where for each time instance (stage) k, the state of the system is an element of the space S, the control action u that is to be implemented so as to achieve optimality belong to a space C, and finally the uncertainty is modeled through a set of random disturbances ω that belong to a *countable set* D. Furthermore, it is assumed that the control u_k is constrained to take values in a given nonempty set $U(x_k) \in C$, which depends of the current state x_k. The random disturbances ω_k, $k = 0, \dots,$ have identical statistics and the probability distributions $\mathsf{P}(\cdot|x_k, u_k)$ are defined on D. These may depend explicitly on x_k and u_k but not on prior disturbances. Given an initial state x_0, we seek a policy π such that $\pi = \{\mu_0, \mu_1, \dots\}$ for which:

$$\mu_k\colon \ S \to C \to, \quad \mu_k(x_k) \in U(x_k), \quad \forall x_k \in S,$$

that minimizes a cost function defined as:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left\{\sum_{k=1}^{N-1} \alpha^k g(x_k, \mu_k(x_k), \omega_k)\right\}.$$

The function $g()$ is the cost per stage such that: g: $S \times C \times D \to \mathbf{R}$ and is assumed to be given. Finally, the parameter α is termed *discount factor* and it holds that: $0 < \alpha \leq 1$. We denote by Π the set of all admissible policies $\pi = \{\mu_0, \mu_1, \dots\}$, that is the set of all sequences of such functions for which:

$$\mu_k\colon \ S \to C, \quad \mu_k(x_k) \in U(x_k), \quad \forall x_k \in S.$$

The optimal cost function J^* is then defined as:

$$J^* = \min_{\pi\in\Pi} J_\pi(x), \quad x \in S.$$

An admissible policy of the form $\pi = \{\mu, \mu, \dots\}$ is termed *stationary* and its corresponding cost is J_μ.

When studying problems of this kind the assumption is made that either the discount factor is $\alpha < 1$ (*discounted problems*, [3]) or that naturally there exists a special cost-free absorbing state (*stochastic shortest path problems*, [3]). In either of these two cases the total expected cost is finite, and the minimization, the way it was previously stated, is well defined. In situations where either the discount factor is 1, or a terminal state does not exist it is more meaningful to optimize an average expected cost as:

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left\{ \sum_{k=0}^{N-1} g(x_k, u_k, \omega_k) \right\} .$$

The problems are known as *average cost per stage problems* and although they bare various similarities with both discounted and stochastic shortest path problems, they have some distinct characteristics. One of the earliest studies was that of D. Blackwell, [7], which made the connection between the optimal average return and the optimal return for values of $\alpha \to 1$. Precisely these characteristics make the analysis of average cost per stage problems the target of intense research, [1]. Connections are made, for developing the associated theory, with both the associated discounted problem, but also recently with an associated stochastic shortest path problem, [4].

Since the theory for analyzing average cost dynamic programming problems has been largely based on the associated theory for discounted and stochastic shortest path problems, most of the results and computational methods bare major similarities. As a prelude to what follows, it should be pointed out that for the average cost per stage problems:

1) the optimal average cost per stage is independent of the initial state for most problems;
2) *Bellman's equation* will take a slightly modified form that would include *differential cost* for each state;
3) there exist computational analogues of all methods developed for either discounted or stochastic shortest path problems.

The cost function of average cost per stage problems are closely related the associated α-discounted problem for a given stationary policy as follows:

- For any stationary policy μ and for any $\alpha \in (0, 1)$ we have:

  $$J_{\alpha,\mu} = (1-\alpha)^{-1} J_\mu + h_\mu + O(|1-\alpha|),$$

  where

  $$J_\mu = \left(\lim_{n\to\infty} \frac{1}{N} \sum_{k=1}^{N-1} P_\mu^k \right) g_\mu$$

  is the average cost corresponding to policy μ, for a process with a transition probability matrix P_μ and costs g_μ. The matrix O is such that: $\lim_{\alpha\to 1} O(|1-\alpha|) = 0$, and the vector h_μ satisfies: $J_\mu + h_\mu = g_\mu + P_\mu h_\mu$.

In the above, the matrix P_μ is the *transition probability matrix* for a given stationary policy μ, given by:

$$P_\mu = \begin{pmatrix} p_{11}(\mu(1)) & \cdots & p_{1n}(\mu(1)) \\ \cdots & \cdots & \cdots \\ p_{n1}(\mu(1)) & \cdots & p_{nn}(\mu(n)) \end{pmatrix}$$

and g_μ the associated cost vector:

$$g_\mu = \begin{pmatrix} g(1, \mu(1)) \\ \cdots \\ g(n, \mu(n)) \end{pmatrix} .$$

The vector h_μ is termed *differential cost vector*, and it represents the difference in N-stage expected optimal cost due to starting at stage i rather than starting at stage j. The key optimality results irrespective of initial states is based on ideas first formulated in [6]. An important element of this analysis is that of a *unichain policy*. Given a *stationary-state Markov chain*, [10], the subset of states that communicate, i. e., there exist transitions k_1 and k_2 for which state transitions probabilities $p_{ij}^{k_1}$ and $p_{ji}^{k_2}$ are positive, is termed a *recurrent class of states*. States that do not belong to a recurrent class are termed *transient*. A stationary policy whose associated Markov chain has a single recurrent class and a possibly empty set of transient states is called *unichain*. In view of the above, the form of the Bellman's equation for characterizing an optimal policy, [9], for the average cost per stage problem takes the following from:

- Assume that any of the following conditions hold:

  1) Every policy that is optimal within the class of stationary policies is unichain.
  2) For every two states i and j, there exists a stationary policy $\pi(i, j)$, such that for some k:

     $$\mathsf{P}(x_k = j | x_0 = i,\ \pi) > 0 .$$

  3) There exist a state t, a constant $L > 0$, and $\overline{\alpha} \in (0, 1)$ such that:

     $$|J_\alpha(i) - J_\alpha(t)| \le L ,$$
     $$i = 1, \dots, n ,$$
     $$\alpha \in (\overline{\alpha}, 1) ,$$

     where J_α is the α-discounted optimal cost vector.

Then:

1) The optimal average cost per stage cost has the same value, λ, for all intimal states, and it satisfies:

$$\lambda = \lim_{\alpha \to 1} (1-\alpha) J_\alpha(i),$$
$$i = 1, \ldots, n.$$

2) For any state t, the vector of differential cost, h is given by:

$$h(i) = \lim_{\alpha \to 1} (J_\alpha(i) - J_\alpha(t)),$$
$$i = 1, \ldots, n,$$

together with λ, satisfies Bellman's equation:

$$\lambda + h(i) = \lim_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u) h(j) \right],$$
$$i = 1, \ldots, n,$$

and λ is the optimal average cost per stage for all states i, i. e.,

$$\lambda = J^*(x) = \min_{\pi} J_\pi(i),$$
$$i = 1, \ldots, n.$$

The above result is also discussed in [2] where the minimization of an expected cost without discounting is considered. All classical methods for computing optimal policies and costs in dynamic programming have their counterparts for addressing average cost per stage problems. Certain alterations are nevertheless necessary. Let us consider first the *value iteration* method exhaustively analyzed in [11,12]. This is a method based on the premise that the limit of steps of the basic dynamic programming algorithm:

$$\lim_{k \to \infty} \frac{1}{k} T^k J = J^*.$$

Two issues arise with average cost per stage problems. First, some elements of the sequence $T^k J$ may diverge to $+\infty$ or $-\infty$ making the numerical calculation troublesome. Furthermore, since we found that the quantity described as the differential cost is important it would be appropriate to develop methods that allow the parallel computation of h as well. [14] developed the fundamentals based on which a *relative value iteration* of the form:

$$h^{k+1}(i) = (Th^k)(i) - (Th^k)(t),$$
$$i = 1, \ldots, n,$$

for some fixed state t, converges to vector h such that $(Th)(t)$ is equal to the optimal average cost per stage for all initial states, and h is the associated differential cost vector. [3] discusses various technical details required for proving convergence. Tight bounds that could improve the computational behavior of the value iteration method were proposed by [8] which modified the approach set forth in [14] to prove that upper and lower bounds on the maximal gain could be readily obtained. These are given according to:

$$\underline{c}_k \leq \underline{c}_{k+1} \leq \lambda \leq \overline{c}_{k+1} \leq \overline{c}_k,$$

where λ is the optimal average cost per stage for all initial states and

$$\underline{c}_k = \min_i [(Th^k)(i) - h^k(i)],$$
$$\overline{c}_k = \max_i [(Th^k)(i) - h^k(i)].$$

Recently, [4], by exploiting the connection between the average cost and the stochastic shortest path problem developed a new value iteration method by making use of *weighter sup-norm contraction* arising in the stochastic shortest path problem. One of the key advantages of this approach is that it admits a *Gauss–Seidel* implementation, thus it is amenable to a distributed implementation. *Policy iteration* methods can also be developed. The policy iteration algorithms generate sequences of stationary policies, each with improved cost over the preceding one. These methods are comprised of two basic steps, a policy evaluation and a policy improvement step. During the first step, for a given stationary policy, μ^k, we obtain the corresponding average and differential costs via the solution of the following system of equations which solution provides the kth iterate of λ, and h:

$$\lambda^k + h^k(i) = g(i, \mu^k(i)) + \sum_{j=1}^{n} p_{ij}(\mu^k(i)) h^k(j),$$
$$j = 1, \ldots, n.$$

The policy improvement step, consists of finding a policy μ^{k+1}, where for all state i, is such that:

$$g(i,\mu^{k+1}(i)) + \sum_{j=1}^{n} p_{ij}(\mu^{k+1}(i))h^k(j)$$
$$= \min_{u\in U(i)}\left[g(i,u) + \sum_{j=1}^{n} p_{ij}(u)h^k(j)\right].$$

In [7], the scope of policy iteration is expanded so as to address problems in which the optimal average cost per stage is not the same for every initial state. It can also be shown, [3], that the optimal vector (λ^*, h^*) is equivalent to the optimal solution of the following linear program:

$$\begin{cases} \max & \lambda \\ \text{s.t.} & \lambda + h(i) \le g(i,u) + \sum_{j=1}^{n} p_{ij}(u)h(j) \\ & u \in U(i) \\ & i = 1,\dots,n. \end{cases}$$

In [9] the *dual problem* of the above-mentioned formulation is considered, whose optimal value is the optimal value of the *primal problem*. The form of the dual is:

$$\begin{cases} \min & \sum_{i=1}^{n} \sum_{u\in U(i)} q(i,u)g(i,u) \\ \text{s.t.} & \sum_{u\in U(i)} q(j,u) = \sum_{i=1}^{n} \sum_{u\in U(i)} q(i,u)p_{ij}(u) \\ & \quad j = 1,\dots,n \\ & \sum_{i=1}^{n} \sum_{u\in U(i)} q(i,u) = 1 \\ & q(i,u) \ge 0, \quad i = 1,\dots,n \\ & u \in U(i). \end{cases}$$

Simulation-based methods are presented in [3,5] that use the basic concepts of *Monte-Carlo simulation* as well as ideas of *reinforcement learning*, [13].

See also

- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Arapostathis A, Borkar VK, Fernández-Gaucherand E, Ghosh ML, Markus SI (1993) Discrete-time controlled markov processes with average cost criterion: A survey. SIAM J Control Optim 31:282–344
2. Bather J (1973) Optimal decision procedures for finite Markov chains. Part I: Example. Adv Appl Probab 5:328–339
3. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
4. Bertsekas DP (1998) A new value iteration method for the average cost dynamic programming problem. SIAM J Optim 36:742–759
5. Bertsekas DP, Tsitsiklis JN (1997) Neuro-dynamic programming. Athena Sci., Belmont, MA
6. Blackwell D (1962) Discounted dynamic programming. Ann Math Statist 33:719–726
7. Blackwell D (1962) Discrete dynamic programming. Ann Math Statist 33:719–726
8. Odoni AR (1969) On finding the maximal gain for markov decision processes. Oper Res 17:857–860
9. Ross SM (1970) Applied probability models with optimization applications. Dover, Mineola, NY
10. Ross SM (1985) Probability models. Acad. Press, New York
11. Schweitzer PJ, Federgruen A (1977) The assymptotic behavior of undiscounted value iteration in Markov decision problems. Math Oper Res 2:360–381
12. Schweitzer PJ, Federgruen A (1978) The functional equation of undiscounted Markov renewal programming. Math Oper Res 3:308–321
13. Singh SP (1994) Reinforcement learning algorithms for average–payoff Markovian decision processes. Proc. 12th Nat. Conf. Artificial Intelligence
14. White DJ (1963) Dynamic programming, Markov chains and the method of successive approximation. J Math Anal Appl 6:373–376

Dynamic Programming in Clustering

L. J. HUBERT[1], P. ARABIE[2], J. MEULMAN[3]
[1] University Illinois, Champaign, USA
[2] Rutgers University, Newark, USA
[3] Leiden University, Leiden, The Netherlands

MSC2000: 90C39, 62H30

Article Outline

Keywords

Combinatorial optimization; Dynamic programming; Clustering; Classification; Ultrametric; Unidimensional scaling; Seriation

Many of the data analysis tasks that arise in the field of *classification* and *clustering* can be given some type of combinatorial characterization that involves the identification of object groupings, *partitions*, or *sequences*. These combinatorial structures are generally defined by certain properties of optimality for some loss (or merit) criterion based on data given in the form of an $n \times n$ *symmetric proximity* matrix $\mathbf{P}$ between distinct pairs of objects from a set $S = \{O_1, \dots, O_n\}$. The focus of this entry will be solely in this context and the use of a general optimization strategy referred to as the *General Dynamic Programming Paradigm* (GDPP), which allows the construction of *recursive procedures* to solve a range of *combinatorial optimization* tasks encountered in the field of classification and clustering. The GDPP will be presented in a general form below with later sections indicating how it can be operationalized for a number of specific problem types. For a more extensive presentation of the topics introduced in this entry and for generalizations to proximity matrices that may not be symmetric or that are defined between objects from (two) distinct sets, the monograph [12] should be consulted. This latter source also provides numerical illustrations for the topics introduced here plus instructions on how to obtain a collection of programs (available on the World Wide Web) to carry out the various optimization tasks presented in this entry and [12]. For a recent and comprehensive review of cluster analysis and the use of mathematical programming techniques in general, see [7].

The GDPP

To present the GDPP, a collection of K sets of entities is first defined, $\Omega_1, \dots, \Omega_K$, where it is possible by some operation to transform entities in Ω_{k-1} to certain entities in Ω_k for $2 \le k \le K$. Each such transformation can be assigned a *merit* (or *cost*) value based *only* on the entity in Ω_{k-1} and the transformed entity in Ω_k. An entity in Ω_k is denoted by A_k, and $\mathcal{F}(A_k)$ is the optimal value that can be assigned to A_k based on the sum of the merit (or cost) increments necessary to transform an entity in Ω_1, step-by-step, to $A_k \in \Omega_k$. If $A_{k-1} \in \Omega_{k-1}$ can be transformed into $A_k \in \Omega_k$, the merit (or cost) of that single transition will be denoted by $M(A_{k-1}, A_k)$ (or $C(A_{k-1}, A_k)$), and where the latter *does not depend* on how A_{k-1} may have been arrived at starting from an entity in Ω_1. Given these conditions, and assuming the values $\mathcal{F}(A_1)$ for $A_1 \in \Omega_1$ are available to initialize the recursive system, $\mathcal{F}(A_k)$ may be constructed for $k = 2, \dots, K$ (when merit is to be maximized) as

$$\mathcal{F}(A_k) = \max[\mathcal{F}(A_{k-1}) + M(A_{k-1}, A_k)],$$

where $A_k \in \Omega_k$, $A_{k-1} \in \Omega_{k-1}$, and the maximum is taken over all A_{k-1} that can be transformed into A_k. Or, if cost is to be minimized,

$$\mathcal{F}(A_k) = \min[\mathcal{F}(A_{k-1}) + C(A_{k-1}, A_k)].$$

In addition, both max/min and min/max forms could be considered as:

$$\mathcal{F}(A_k) = \max[\min(\mathcal{F}(A_{k-1}), M(A_{k-1}, A_k))],$$
$$\mathcal{F}(A_k) = \min[\max(\mathcal{F}(A_{k-1}), C(A_{k-1}, A_k))].$$

The leading maximization or minimization is over all $A_{k-1} \in \Omega_{k-1}$ that can be transformed into A_k. In all instances, an optimal solution is identified by some value for $\mathcal{F}(A_K)$ for a specific $A_K \in \Omega_K$, and the actual optimal solution obtained by working backwards through the recursion to see how $\mathcal{F}(A_K)$ was constructed.

Partitioning

The most direct characterization of the partitioning task can be stated as follows: given $S = \{O_1, \ldots, O_n\}$ and $\mathbf{P} = \{p_{ij}\}$, find a collection of M mutually exclusive and exhaustive subsets (or clusters) of S, say, $S_1, \ldots, S_M$, such that for some measure of *heterogeneity* $H(\cdot)$ that attaches a value to each possible subset of S, either the sum $\sum_{m=1}^{M} H(S_m)$, or alternatively, $\max[H(S_1), \ldots, H(S_M)]$, is minimized. This stipulation assumes that heterogeneity has a cost interpretation and that smaller values of the heterogeneity indices represent the'better' subsets (or clusters). If $H(S_m)$ for some $S_m \subseteq S$ depends only on those proximities from $\mathbf{P}$ that are within S_m and/or between S_m and $S - S_m$, an application of the GDPP is possible. Define K to be M, and let each of the sets $\Omega_1, \ldots, \Omega_M$ contain *all* of the $2^n - 1$ nonempty subsets of the n object subscripts; $\mathcal{F}(A_k)$ is the optimal value for a partitioning into k classes of the object subscripts present in A_k. A transformation of an entity $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if $A_{k-1} \subset A_k$, with cost $C(A_{k-1}, A_k) \equiv H(A_k - A_{k-1})$. Thus, beginning with the heterogeneity indices $H(A_1)$ for *every* subset $A_1 \subseteq S$, the recursion can be carried out, with the optimal solution represented by $\mathcal{F}(A_M)$ when $A_M = S$.

The first discussion of this general type of recursive solution for the partitioning task was in [14] but limited to one specific measure of subset heterogeneity defined by the sum of proximities within a subset divided by twice the number of objects in the subset. If the original proximities in $\mathbf{P}$ happened to be squared *Euclidean distances* between numerically given vectors (or profiles) for the n objects over some set of variables, then this subset heterogeneity measure is equivalent to the sum of squared Euclidean distances between each profile and the mean profile for the subset (this quantity is usually called the *sum of squared error* or the *k-means criterion*, e. g., see [16], p. 52) A major advantage of the GDPP formulation is that a variety of heterogeneity measures can be considered under a common rubric, with the sole requirement that the measure chosen be dependent only on the proximities within a subset and/or between the subset and its complement. For example, in [12] some twelve different alternatives are illustrated using a program implementation that can effectively deal with object set sizes in their lower 20's with the type of computational equipment and storage capacity now commonly available. As noted in a later section, it is also possible to extend the GDPP heuristically to allow for much larger object set sizes, although an absolute guarantee of globally optimality for the identified structures is sacrificed.

Admissibility Restrictions on Partitions

A specific restriction discussed at some length in the literature (see [8, Chapt. 5], [16, pp. 61–64], [6]) that would permit the construction of optimal partitions (subject to the restriction) for very large object sets is when there is an a priori assumed object *ordering* along a continuum that can be taken without loss of generality as $O_1 \prec \cdots \prec O_n$, and the only *admissible clusters* are those for which the objects in the cluster form a consecutive sequence or segment. Thus, an optimal partition will consist of M clusters, each of which defines a consecutive segment along the given object ordering. To tailor the GDPP to a consecutive-ordering admissibility criterion, each of the sets $\Omega_1, \ldots, \Omega_M$ is now defined by the n subsets of S that contain the objects $\{O_1, \ldots, O_i\}$ for $1 \le i \le n$; $\mathcal{F}(A_k)$ is the optimal value for a partitioning of A_k into k classes; a transformation of an entity $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if $A_{k-1} \subset A_k$; and the cost of the transition is $H(A_k - A_{k-1})$, where $A_k - A_{k-1}$ must contain a consecutive sequence of objects. Again, $\mathcal{F}(A_M)$ for $A_M = S$ identifies an optimal solution.

The selection of some prespecified ordering that constrains admissible clusters in a partition obviously does not lead necessarily to the same unconstrained optimal partitions, even though the identical subset heterogeneity measure and optimization criterion are being used. There are, however, several special instances where the original proximity matrix $\mathbf{P}$ is appropriately defined and/or patterned so that the imposition of a particular order constraint does invariably lead to partitions that would also be optimal even when no such order constraint was imposed. One such result dates back to W.D. Fisher [6] who showed that when proximities are squared differences between the values on some (unidimensional) variable, and the order constraint is derived from the ordering of the objects on this variable, then the selection of the sum of squared error as the subset heterogeneity measure, and minimizing this sum as an optimization criterion, leads to partitions that are not only optimal under the order constraint but also optimal when unconstrained, i. e., an unconstrained optimal partition will include only those subsets defined by objects consecutive in the given order. (The subset heterogeneity measure in this unidimensional case reduces to the sum of squared deviations of the univariate values for the objects from their mean value within the subset.) A more general result appears in [3] where the special case is discussed when a proximity matrix **P** can be row- and column-reordered to display an anti-Robinson form (a matrix pattern first introduced in [15]). As stated more formally below, a matrix has an *anti-Robinson* form if the entries within each row and column of **P** never decrease when moving away from a main diagonal entry in any direction. For certain subset heterogeneity measures and optimization criteria, imposing the order constraint that displays the anti-Robinson pattern in the row- and column-reordered proximity matrix leads to partitions that are also optimal when unconstrained.

The choice of an ordering that can be imposed to constrain the search domain for optimal partitions could be directly tied to a task, discussed later, of finding an (optimal) sequencing of the objects along a continuum. Somewhat more generally, one possible data analysis strategy for seeking partitions as close to optimal as possible, would be to construct an object ordering through an initial optimization process, and possibly one based on another analysis method that could then constrain the domain of search for an optimal partition. Obviously, if one were successful in generating an appropriate object ordering, partitions that would be optimal when constrained would also be optimal without the constraint. The obvious key here is to have some mechanism for identifying an appropriate order to give this possible equivalence (between an optimal constrained partition and one that is optimal without constraint) a chance to succeed. As one example of how such a process might be developed for constructing partitions based on an empirically generated ordering for the objects, a three-stage process is proposed in [1] and [2]. First, the objects to be partitioned are embedded in a *Euclidean representation* with a specific *multidimensional scaling* strategy. Second, by heuristic methods, a path among the n objects in the Euclidean representation is identified (hopefully, with close to minimal length) and used to define a prior ordering for the objects and to constrain the subsets present in a partition. Finally, a recursive strategy of the same general form just described is carried out to obtain a partitioning of S.

Hierarchical Clustering

The problem of hierarchical clustering will be characterized by the search for an optimal collection of partitions of S, which are denoted generically as $\mathcal{P}_1, \ldots, \mathcal{P}_n$. Here, $\mathcal{P}_1$ is the (trivial) partition where all n objects from S are placed into n separate classes, $\mathcal{P}_n$ is the (also trivial) partition where a single subset contains all n objects, and $\mathcal{P}_k$ is obtained from $\mathcal{P}_{k-1}$ by uniting some pair of classes present in $\mathcal{P}_{k-1}$. As an optimization criterion the sum of transition costs is minimized, irrespective of how the costs might be defined, between successive partitions in a hierarchy. Specifically, suppose $T(\mathcal{P}_{k-1}, \mathcal{P}_k)$ denotes some measure of transition cost between two partitions $\mathcal{P}_{k-1}$ and $\mathcal{P}_k$, where $\mathcal{P}_k$ is constructed from $\mathcal{P}_{k-1}$ by uniting two classes in the latter partition. An optimal *partition hierarchy* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ will be one for which the sum of the transition costs, $\sum_{k>2} T(\mathcal{P}_{k-1}, \mathcal{P}_k)$, is minimized. To apply the GDPP, first define n sets $\Omega_1, \ldots, \Omega_n$, where Ω_k contains all partitions of the n objects in S into $n - k + 1$ classes. The value $\mathcal{F}(A_k)$ for $A_k \in \Omega_k$ is the optimal sum of transition costs up to the partition A_k; a transformation of an entity $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if A_k is obtainable from A_{k-1} by uniting two classes in A_{k-1}, and has cost $C(A_{k-1}, A_k) \equiv T(A_{k-1}, A_k)$. Beginning with an assumed value for $\mathcal{F}(A_1)$ of 0 for the single entity $A_1 \in \Omega_1$ (which is the partition of S into n subsets each containing a single object), and constructing $\mathcal{F}(A_k)$ recursively for $2, \ldots, n$, an optimal solution is identified by $\mathcal{F}(A_n)$ for the single entity $A_n \in \Omega_n$ defined by the partition containing all n objects in a single class.

A concept routinely encountered in discussions of hierarchical clustering is that of an *ultrametric*, which can be characterized by any nonnegative $n \times n$ symmetric dissimilarity matrix for distinct pairs of the objects in S, denoted generically as $\mathbf{U} = \{u_{ij}\}$, where $u_{ij} = 0$ if and only if $i = j$ and the entries in **U** satisfy the ultrametric inequality: $u_{ij} \leq \max\{u_{ik}, u_{jk}\}$ for $1 \leq i, j, k \leq n$. Any ultrametric identifies a specific partition hierarchy, $\mathcal{P}_1, \ldots, \mathcal{P}_n$, where those object pairs defined between subsets united in $\mathcal{P}_{t-1}$ to form $\mathcal{P}_t$ all have a common ultrametric value; moreover, this latter value is not smaller than those for object pairs defined within these same subsets. One approach to the development of hierarchical clustering methods is by directly fitting an ultrametric to **P** minimizing a loss criterion defined by an L_p-norm between $\{p_{ij}\}$ and a (to be identified) ultrametric matrix $\{u_{ij}\}$. To be specific, for a given partition hierarchy, $\mathcal{P}_1, \ldots, \mathcal{P}_n$, let $C^{(u)}_{t-1}$ and $C^{(v)}_{t-1}$ denote the two classes united in $\mathcal{P}_{t-1}$ to form $\mathcal{P}_t$, and specify b_{t-1} to be some appropriate aggregate (or 'average') value of the proximities for object pairs between $C^{(u)}_{t-1}$ and $C^{(v)}_{t-1}$. The loss functions used to index the adequacy of a given partition hierarchy in producing an ultrametric fitted to **P** are for the L_1-*norm*:

$$\sum_{t=2}^{n} \sum_{\substack{O_{i'} \in C^{(u)}_{t-1}, \\ O_{j'} \in C^{(v)}_{t-1}}} \left| p_{i'j'} - b_{t-1} \right| ,$$

where b_{t-1} is the median proximity between $C^{(u)}_{t-1}$ and $C^{(v)}_{t-1}$; for the L_2-*norm*:

$$\sum_{t=2}^{n} \sum_{\substack{O_{i'} \in C^{(u)}_{t-1}, \\ O_{j'} \in C^{(v)}_{t-1}}} (p_{i'j'} - b_{t-1})^2 ,$$

where b_{t-1} is the mean proximity between $C^{(u)}_{t-1}$ and $C^{(v)}_{t-1}$; and for the L_∞-*norm*:

$$\sum_{t=2}^{n} \max_{\substack{O_{i'} \in C^{(u)}_{t-1}, \\ O_{j'} \in C^{(v)}_{t-1}}} \left| p_{i'j'} - b_{t-1} \right| ,$$

where b_{t-1} is the average of the minimum and maximum proximities between $C^{(u)}_{t-1}$ and $C^{(v)}_{t-1}$. For all three L_p-norms, an optimal ultrametric will be one for which the order constraint on the between-subset aggregate values holds: $b_1 \leq \cdots \leq b_{n-1}$, and the norm is minimized. For such an optimal solution, $b_1, \ldots, b_{n-1}$ define the distinct entries in an (optimal) fitted ultrametric. To implement a *dynamic programming* approach for locating an optimal ultrametric, $C(A_{k-1}, A_k)$ is the incremental cost of transforming A_{k-1} to A_k characterized by the appropriate L_p-norm when that pair of subsets in A_{k-1} is united to form A_k. As developed in detail in [11], an explicit admissibility criterion must also be imposed for defining a permissible transition from A_{k-1} to A_k that could ensure a nondecreasing sequence of between-subset aggregate values.

Constrained Hierarchical Clustering

Analogously to the admissibility conditions for partitions, one constraint that might be imposed on each partition in Ω_k is for the constituent subsets to contain objects consecutive in some given *ordering* (which could be taken as $O_1 \prec \cdots \prec O_n$ without loss of any generality). Thus, Ω_k will be redefined to contain those partitions that include $n - k + 1$ classes, and where each class is a segment in the given object ordering.

Optimal Sequencing of an Object Set

A combinatorial optimization task closely related to both partitioning and hierarchical clustering is the search for an optimal *sequencing* of the object set S based on the proximity matrix **P**. A best reordering is sought for the rows and columns of **P** that will optimize, over all possible row/column reorderings, some specified measure of patterning for the entries of the reordered matrix. Irrespective of the particular measure chosen, the GDPP is specialized as follows: A collection of sets $\Omega_1, \ldots, \Omega_n$ is defined, where Ω_k includes all the subsets that have k members from the integer set $\{1, \ldots, n\}$. The value $\mathcal{F}(A_k)$ is the optimal contribution to the total measure of matrix patterning for the objects in A_k when they occupy the first k positions in the (re)ordering. A transformation is now possible between $A_{k-1} \in \Omega_{k-1}$ and $A_k \in \Omega_k$ if $A_{k-1} \subset A_k$ (i. e., A_{k-1} and A_k differ by one integer). The contribution to the total measure of patterning generated by placing the single integer in $A_{k-1} - A_k$ at the kth order position is $M(A_{k-1}, A_k)$. As always, the validity of the recursive process will require the incremental merit index, $M(A_{k-1}, A_k)$, to depend only on the unordered sets A_{k-1} and A_k, and the complement $S - A_k$, and specifically *not* on how A_{k-1} may have been obtained beginning with Ω_1. Assuming $\mathcal{F}(A_1)$ for all $A_1 \in \Omega_1$ are available, the recursive process can be carried out from Ω_1 to Ω_n, with $\mathcal{F}(A_n)$ for the single set $A_n = \{1, \ldots, n\} \in \Omega_n$ defining the optimal value for the specified measure of matrix patterning.

Two general classes of measures of matrix patterning are mentioned here. The first is a row and column *gradient index* motivated by the ideal of reordering a symmetric proximity matrix to have an anti-Robinson form (and which is the same structure noted briefly in the clustering context when an optimal order-constrained partition might also be optimal when unconstrained). Specifically, suppose $\rho(\cdot)$ is some permutation of the first n integers that reorders both the rows and columns of $\mathbf{P}$ (i. e., $\mathbf{P}_\rho \equiv \{p_{\rho(i)}\rho(j)\}$). As noted earlier, the reordered matrix $\mathbf{P}_\rho$ is said to have an anti-Robinson form if the entries within the rows and within the columns of $\mathbf{P}_\rho$ moving away from the main diagonal in any direction never decrease; or formally, two gradient conditions must be satisfied:

- within rows: $p_{\rho(i)\rho(k)} \leq p_{\rho(i)\rho(j)}$ for $1 \leq i < k < j \leq n$;
- within columns: $p_{\rho(k)\rho(j)} \leq p_{\rho(i)\rho(j)}$ for $1 \leq i < k < j \leq n$.

It might be noted that whenever $\mathbf{P}$ is an ultrametric, or if $\mathbf{P}$ has an exact Euclidean representation in a single dimension (i. e., $\mathbf{P} = \{|x_j - x_i|\}$, for some collection of coordinate values, $x_1, \ldots, x_n$), then $\mathbf{P}$ can be row/column reordered to display a perfect anti-Robinson pattern. Thus, the notion of an anti-Robinson form can be interpreted as generalizing either a perfect discrete classificatory structure induced by a partition hierarchy (through an ultrametric) or as the pattern expected in $\mathbf{P}$ if there exists an exact *unidimensional Euclidean representation* for the objects in S. In any case, if a matrix can be row/column reordered to display an anti-Robinson form, then the objects are orderable along a continuum so that the degree of separation between objects in the ordering is reflected perfectly by the dissimilarity information in $\mathbf{P}$, i. e., for the object ordering, $O_{\rho(i)} \prec O_{\rho(k)} \prec O_{\rho(j)}$ (for $i < k < j$), $p_{\rho(i)\rho(k)} \leq p_{\rho(i)\rho(j)}$ and $p_{\rho(k)\rho(j)} \leq p_{\{\rho(i)\rho(j)}$.

A natural (merit) measure of how well a reordered proximity matrix $\mathbf{P}_\rho$ satisfies these two gradient conditions would rely on an aggregate index of the violations/nonviolations over all distinct object triples, as given by the expression:

$$\sum_{i<k<j} f(p_{\rho(i)\rho(k)}, p_{\rho(i)\rho(j)}) + \sum_{i<k<j} f(p_{\rho(k)\rho(j)}, p_{\rho(i)\rho(j)}), \quad (1)$$

where $f(\cdot, \cdot)$ is some function indicating how a violation/nonviolation of a particular gradient condition for an object triple within a row or within a column (and defined above the main diagonal of $\mathbf{P}_\rho$) is to be counted in the total measure of merit. The one option concentrated on here will be $f(z, y) = \text{sign}(z - y) = +1$ if $z > y$; 0 if $z = y$; and -1 if $z < y$; thus, the (raw) number of satisfactions minus the number of dissatisfactions of the gradient conditions *within rows* above the main diagonal of $\mathbf{P}_\rho$ is given by the first term in (1), and the (raw) number of satisfactions minus dissatisfactions of the gradient conditions *within columns* above the main diagonal of $\mathbf{P}_\rho$ is given by the second term. To carry out the GDPP based on the measure in (1), an explicit form must be given for the incremental contribution, $M(A_{k-1}, A_k)$, to the total merit measure of patterning generated by placing the single integer $A_k - A_{k-1}$ at the kth order position. For any ordering $\rho(\cdot)$ of the rows and columns of $\mathbf{P}$, the merit increment for placing an integer, say, k' ($\equiv \rho(k)$) (i. e., $\{k'\} = A_k - A_{k-1}$) at the kth order position can be defined as $\sum_{k=1}^{n} I_{\text{row}}(\rho(k)) + \sum_{k=1}^{n} I_{\text{col}}\rho(k))$, where

$$I_{\text{row}}(\rho(k)) = \sum_{i' \in A_{k-1}} \sum_{j' \in S - A_k} f(p_{i'k'}, p_{i'j'}),$$

$$I_{\text{col}}(\rho(k)) = \sum_{i' \in A_{k-1}} \sum_{j' \in S - A_k} f(p_{k'j'}, p_{i'j'}),$$

and $A_{k-1} = \{\rho(1), \ldots, \rho(k-1)\}$, $S - A_k = \{\rho(k+1), \ldots, \rho(n)\}$. Thus, letting $\mathcal{F}(A_1) = 0$ for all $A_1 \in \Omega_1$, and using the specification for $f(\cdot, \cdot)$ suggested above, the recursion can be carried out to identify an optimal row/column reordering of the given proximity matrix $\mathbf{P}$ to maximize this gradient measure over all row/column reorderings of $\mathbf{P}$.

A second class of measures of matrix patterning can be derived indirectly from the auxiliary problem of attempting to fit a given proximity matrix $\mathbf{P}$ by some type of unidimensional scaling representation (i. e., a *seriation*). Suppose the search is for a set of n ordered coordinate values, $x_1 \leq \cdots \leq x_n$ (such that $\sum_k x_k = 0$), and a permutation $\rho(\cdot)$ to minimize the *least squares criterion*

$$\sum_{i<j} (p_{\rho(i)\rho(j)} - |x_j - x_i|)^2.$$

After some algebraic reduction (see [5]), this latter least squares criterion can be rewritten as

$$\sum_{i<j} p_{ij}^2 + n \sum_k [x_k - \left(\frac{1}{n}\right) G(\rho(k))]^2 - \frac{1}{n} \sum_k [G(\rho(k))]^2,$$

where

$$G(\rho(k)) = \sum_{i=1}^{k-1} p_{\rho(k)\rho(i)} - \sum_{i=k+1}^{n} p_{\rho(k)\rho(i)}.$$

If the measure

$$\sum_{k=1}^{n} [G(\rho(k))]^2 \qquad (2)$$

is maximized over all row/column reorderings of **P**, and denoting the optimal permutation by $\rho^*(\cdot)$, then $G(\rho^*(1)) \leq \cdots \leq G(\rho^*(n))$, and the optimal coordinates can be retrieved as $x_k = (1/n)G(\rho^*(k))$, for $1 \leq k \leq n$. To execute the GDPP recursion using (2), the merit increment for placing the integer, say k' ($\equiv \rho(k)$) (i. e., $\{k'\} = A_k - A_{k-1}$) in the kth order position can be written as $[G(\rho(k))]^2$, where

$$G(\rho(k)) = \sum_{i' \in A_{k-1}} p_{k'i'} - \sum_{j' \in S - A_k} p_{k'j'},$$

with $A_{k-1} = \{\rho(1), \ldots, \rho(k-1)\}$, $S - A_k = \{\rho(k+1), \ldots, \rho(n)\}$, and $\mathcal{F}(A_1)$ for $A_1 = \{k'\} \in \Omega_1$ defined by

$$\left(\sum_{j' \in S - \{k'\}} p_{k'j'} \right)^2 .$$

Optimal Sequencing Based on the Construction of Optimal Paths

To tailor the GDPP (and for the moment emphasizing the minimization of the sum of adjacent object proximities in constructing a path among the objects in S), a collection of sets $\Omega_1, \ldots, \Omega_n$ is defined so that each entity in Ω_k, $1 \leq k \leq n$, is now an ordered pair (A_k, j_k). Here, A_k is a k element subset of the n subscripts on the objects in S, and j_k is one subscript in A_k (to be interpreted as the subscript for the last-placed object in a sequencing of the objects contained within A_k). The function value $\mathcal{F}((A_k, j_k))$ is the optimal contribution to the total measure of matrix patterning for the objects in A_k when they are placed in the first k positions in the (re)ordering, and the object with subscript j_k occupies the kth. A transformation is possible between $(A_{k-1}, j_{k-1}) \in \Omega_{k-1}$ and $(A_k, j_k) \in \Omega_k$ if $A_{k-1} \subset A_k$ and $A_k - A_{k-1} = \{j_k\}$ (i. e., A_{k-1} and A_k differ by the one integer j_k). The cost increment $C((A_{k-1}, j_{k-1}), (A_k, j_k))$ is simply $p_{(j_{k-1})j_k}$ for the contribution to the total measure of patterning generated by placing the object with the single integer subscript in $A_k - A_{k-1}$ at the kth order position (i. e., the proximity between the adjacently-placed objects with subscripts j_{k-1} and j_k). The type of GDPP recursion used for the construction of optimal *linear paths* can be modified easily for the construction of optimal *circular paths*: choose object O_1 as an (arbitrary) origin and force the construction of the optimal linear paths to include O_1 as the initial object by defining $\mathcal{F}((A_1, j_1)) = 0$ for $j_1 = 1$ and $A_1 = \{1\}$, and otherwise by a very large positive or negative value (depending on whether the task is a minimization or a maximization, respectively). The function values $\mathcal{F}((A_n, j_n))$ for all j_n, $1 \leq j_n \leq n$ for $(A_n, j_n) \in \Omega_n$ and $A_n = \{1, \ldots, n\}$ can then be used to obtain the optimal circular paths depending on the chosen optimization criteria as follows:

- *minimum path length*:
$$\min[\mathcal{F}((A_n, j_n)) + p_{j_n 1}];$$
- *maximum path length*:
$$\max[\mathcal{F}((A_n, j_n)) + p_{j_n 1}];$$
- *minimax path length*:
$$\min[\max(\mathcal{F}((A_n, j_n)), p_{j_n 1})];$$
- *maximin path length*:
$$\max[\min(\mathcal{F}((A_n, j_n)), p_{j_n 1})].$$

For the first discussions in the literature on constructing optimal paths through DP, see [4,9]; for applications to a variety of data analysis tasks, see [13].

Optimal Ordered Partitions

The task of constructing an *ordered partition* of an object set $S = \{O_1, \ldots, O_n\}$ into M ordered classes, $S_1 \prec \cdots \prec S_M$, using some (merit) measure of matrix patterning and a proximity matrix **P**, can be approached through the GDPP recursive process applied to the partitioning task but with appropriate variation in defining the merit increments. Explicitly, the sets $\Omega_1, \ldots, \Omega_M$ will each contain all $2^n - 1$ nonempty subsets of the n object subscripts; $\mathcal{F}(A_k)$ for $A_k \in \Omega_k$ is the optimal value for placing k classes in the first k positions, and the subset A_k is the union of these k classes. A transformation from $A_{k-1} \in \Omega_{k-1}$ to $A_k \in \Omega_k$ is possible if $A_{k-1} \subset A_k$; the merit increment $M(A_{k-1}, A_k)$ is based on placing the class $A_{k-1} - A_k$ at the kth position (which will depend on A_{k-1}, A_k, and $S - A_k$). Beginning with $\mathcal{F}(A_1)$ for all $A_1 \in \Omega_1$ (i. e., the merit of placing the class A_1 at the first position), the recursion proceeds from Ω_1 to Ω_M, with $\mathcal{F}(A_M)$ for $A_M = S \in \Omega_M$ defining the optimal merit value for an ordered partition into M classes.

To generalize the gradient measure given in (1), the merit increment for placing the class $A_k - A_{k-1}$ at the kth order position is $I_{\text{row}}(A_k - A_{k-1}) + I_{col(A_k - A_{k-1})}$, where

$$I_{\text{row}}(A_k - A_{k-1}) = \sum_{i' \in A_{k-1}} \sum_{k' \in A_k - A_{k-1}} \sum_{j' \in S - A_k} f(p_{i'k'}, p_{i'j'}),$$

and

$$I_{\text{col}}(A_k - A_{k-1}) = \sum_{i' \in A_{k-1}} \sum_{k' \in A_k - A_{k-1}} \sum_{j' \in S - A_k} f(p_{k'j'}, p_{i'j'}).$$

To initialize the recursion, let $\mathcal{F}(A_1) = 0$ for all $A_1 \in \Omega_1$.

A merit measure based on a coordinate representation for each of the M ordered classes, $S_1 \prec \cdots \prec S_M$, that generalizes (2) can also be developed directly. Here, M coordinates, $x_1 \leq \cdots \leq x_M$, are to be identified so that the residual sum-of-squares

$$\sum_{k \leq k'} \sum_{\substack{i_k \in S_k, \\ j_{k'} \in S_{k'}}} (p_{i_k j_{k'}} - |x_{k'} - x_k|)^2,$$

is minimized (the notation $pi_k j_{k'}$) indicates those proximities in **P** defined between objects with subscripts $i_k \in S_k$ and $j_{k'M} \in S_{k'}$). A direct extension of the argument that led to optimal coordinate representation for single objects would require the maximization of

$$\sum_{k=1}^{M} \left(\frac{1}{n_k}\right) (G(A_k - A_{k-1}))^2, \tag{3}$$

where

$$G(A_k - A_{k-1}) = \sum_{k' \in A_k - A_{k-1}} \left(\sum_{i' \in A_{k-1}} p_{k'i'} - \sum_{i' \in S - A_k} p_{k'i'} \right),$$

and n_k denotes the number of objects in $A_k - A_{k-1}$. The merit increment for placing the subset $A_k - A_{k-1}$ at the kth order position would be $(1/n_k)(G(A_k - A_{k-1}))^2$, with the recursion initialized by

$$\mathcal{F}(A_1) = \left(\frac{1}{n_1}\right) \left(\sum_{k' \in A_1} \sum_{i' \in S - A_1} p_{k'i'} \right)^2,$$

for all $A_1 \in \Omega_1$. If an optimal ordered partition that maximizes (3) is denoted by $S_1^* \prec \cdots \prec S_M^*$, the optimal coordinates for each of the M classes can be given as

$$x_k^* = \left(\frac{1}{n}\right) \left(\frac{G(S_k^*)}{n_k}\right),$$

where $x_1^* \leq \cdots \leq x_M^*$, and $\sum_k n_k x_k^* = 0$. A more complete discussion of constructing optimal ordered partitions appears in [10].

Heuristic Applications of the GDPP

When faced with the task of finding a single optimal partition for a (large) object set S, if one had knowledge that for an optimal M-class partition the classes could be allocated to two (or more) groups, then the aggregate collections of the objects within these latter groups could be separately and optimally partitioned and an optimal M-class partition for the complete object set identified directly. Or, if it were known that certain elemental subsets of the objects in S had to appear within the classes of an optimal M-class partition, one could begin with these elemental subsets as the objects to be analyzed, and an optimal M-class partition could again be retrieved. The obvious difficulty is to identify either the larger aggregate groups that might be dealt with separately, or an appropriate collection of elemental subsets, and in a size and number that might be handled by the recursive optimization strategy. For the latter task of identifying elemental subsets, one possible approach would be to begin with a partition of S into several classes (possibly obtained through another heuristic process), and where each class contained a number of objects that could be optimally analyzed. Based on these separate subset analyses, a (tentative) collection of elemental subsets would be identified. These could then be used to obtain a subdivision of S, and again within each group of this subdivision, the objects could be optimally partitioned to generate a possibly better collection of elemental subsets. This process could be continued until no change occurred in the particular elemental subsets identified. As an alternative, one could start with some collection of tentative *elemental subsets* obtained through another (*heuristic*) optimization strategy and try, if possible, to improve upon these through the same type of procedure. This latter approach is illustrated in [12]. Similarly, the tasks of constructing a (hopefully optimal) partition hierarchy or object order for a (large) set could be approached through the identification of a collection of elemental subsets, which would then be operated on as the basic entities for the generation of a partition hierarchy or an object sequence.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Alpert CJ, Kahng AB (1995) Multiway partitioning via geometric embeddings, orderings, and dynamic programming. IEEE Trans Computer-Aided Design Integr Circuits and Syst 14:1342–1357
2. Alpert CJ, Kahng AB (1997) Splitting an ordering into a partition to minimize diameter. J Class 14:51–74
3. Batagelj V, Korenjak-Černe S, Klavžar S (1994) Dynamic programming and convex clustering. Algorithmica 11:93–103
4. Bellman R (1962) Dynamic programming treatment of the traveling salesman problem. J ACM 9:61–63
5. Defays D (1978) A short note on a method of seriation. British J Math Statist Psych 31:49–53
6. Fisher WD (1958) On grouping for maximum heterogeneity. J Amer Statist Assoc 53:789–798
7. Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Math Program 79:191–215
8. Hartigan JA (1975) Clustering algorithms. Wiley, New York
9. Held M, Karp RM (1962) A dynamic programming approach to sequencing problems. J Soc Indus Appl Math 10:196–210
10. Hubert LJ, Arabie P, Meulman J (1997) The construction of globally optimal ordered partitions. In: Mirkin B, McMorris FR, Roberts FS, Rzhetsky A (eds) Mathematical hierarchies and biology. DIMACS, Amer. Math. Soc., Providence, RI, pp 299–312
11. Hubert LJ, Arabie P, Meulman J (1997) Hierarchical clustering and the construction of (optimal) ultrametrics using L_p-norms. In: Dodge Y (ed) L_1-statistical procedures and related topics. vol 31 of Lecture Notes. Inst. Math. Statist., Berkley, U.S., pp 457–472
12. Hubert LJ, Arabie P, Meulman J (2001) Combinatorial data analysis: Optimization by dynamic programming. SIAM, Philadelphia
13. Hubert LJ, Baker FB (1978) Applications of combinatorial programming to data analysis: The traveling salesman and related problems. Psychometrika 43:81–91
14. Jensen RE (1969) A dynamic programming algorithm for cluster analysis. J Oper Res Soc Amer 7:1034–1057
15. Robinson WS (1951) A method for chronologically ordering archaeological deposits. Amer Antiq 16:293–301
16. Späth H (1980) Cluster analysis algorithms. Horwood, Westergate

Dynamic Programming: Continuous-time Optimal Control

William R. Esposito
Department Chemical Engineering,
Princeton University, Princeton, USA

MSC2000: 49L20, 34H05, 90C39

Article Outline

Keywords

Dynamic programming; Continuous-time optimal control

Even though *dynamic programming* [1] was originally developed for systems with discrete types of decisions, it can be applied to continuous problems as well. In this article the application of dynamic programming to the solution of continuous time optimal control problems is discussed.

Problem Formulation

Consider the following continuous time dynamical system:

$$\begin{cases} \dot{z}(t) = f(z(t), u(t)), & \\ z(0) = z_0, & 0 \le t \le T, \end{cases} \tag{1}$$

where $z(t) \in \mathbf{R}^n$ is the state vector at time t with time derivative given by $\dot{z}(t)$, $u(t) \in U \subset \mathbf{R}^m$ is the control vector at time t, U is the set of control constraints, and T is the terminal time. The function $f(z(t), u(t))$ is continuously differentiable with respect to z and continuous with respect to u. The set of admissible control trajectories are given by the piecewise constant functions, $\{u(t)\colon u(t) \in U, \forall t \in [0, T]\}$. It is assumed that for any admissible control trajectory, that a state trajectory $z^u(t)$ exists and is unique. For a full treatment of existence and uniqueness, see [4].

The objective is to determine a control trajectory and the corresponding state trajectory which minimizes a cost function of the form:

$$h(z^u(T)) + \int_0^T g(z^u(t), u(t))\, dt, \tag{2}$$

where the functions g, and h are continuously differentiable with respect to both z and u.

Example

As a simple example, consider the problem of moving a unit mass from an initial point to a given final point. The position of the mass along a line is given by the state $z_1(t)$ and its velocity by $z_2(t)$. The control $u(t)$ is the force applied to the mass, and is bounded $u(t) \in [-1, 1]$. This system is described by:

$$\begin{aligned} &\dot{z}_1(t) = z_2(t), \quad \dot{z}_2(t) = u(t), \\ &z(0) = [z_1(0), z_2(0)], \quad t \in [0, T], \\ &u(t) \in [-1, 1]. \end{aligned}$$

The objective is to move this mass as near to the final state point, $[\bar{z}_1, \bar{z}_2]$, as possible. This can be formulated as the minimization of the square error at the final time point.

$$\min_{u(t)} \sum_{i=1}^{2} (z_i(T) - \bar{z}_i(T))^2 .$$

Converting this cost function into the form given by (2)) results in:

$$\begin{aligned} &h(z(T)) = \sum_{i=1}^{2} (z_i(T) - \bar{z}_i(T))^2 , \\ &g(z^u(t), u(t)) = 0, \quad \forall t \in [0, T]. \end{aligned}$$

Hamilton–Jacobi–Bellman Equation

The time horizon is divided into N equally spaced intervals with δ = T/N. This converts the problem into the discrete-time domain and the dynamic programming approach can be applied. Once the approach is applied, the result is converted back into the continuous-time domain by taking the limit as $\delta \to 0$. The result is the following partial differential equation,

$$\begin{aligned} &0 = \min_{u \in U} \\ &\left[g(z,u) + \nabla_t J^*(t,z) + \nabla_x J^*(t,z)^\top f(z,u)\right], \\ &J^*(T,z) = h(z), \quad \forall z, \end{aligned} \tag{3}$$

where $J^*(t, z)$ is the optimal cost-to-go function. This equation is called the *Hamilton–Jacobi–Bellman equation*. It is also referred to as the *continuous-time analog of the dynamic programming equation.*

Pontryagin Minimum Principle

It is possible to derive the *Pontryagin minimum principle* using the Hamilton–Jacobi–Bellman equation given above. Using the system given by (1), the classic principle results:

$$
\begin{aligned}
&\dot{p}(t) = -\nabla_z H(z^*(t), u*(t), p(t)),\\
&p(T) = \nabla h(z^*(T)),\\
&H(z^*(t), u^*(t), p(t))\\
&\quad = g(z^*(t), u^*(t)) + p^\top(t) f(z^*(t), u^*(t)),\\
&u^*(t) = \arg\min_{u \in U} H(z^*(t), u(t), p(t)),
\end{aligned}
$$

where $z^*(t)$ and $u^*(t)$ are the optimal state and control trajectories, respectively.

A more detailed description of these two results are given in the following sections. For dynamic programming and optimal control problems, see [2] as well as the classic optimal control text [3].

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ High-order Maximum Principle for Abnormal Extremals
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Pontryagin Maximum Principle
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
3. Bryson AE, Ho Y (1975) Applied optimal control. Hemisphere, Washington, DC
4. Pontryagin LS (1962) Ordinary differential equations. Addison-Wesley, Reading, MA

Dynamic Programming: Discounted Problems

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C39

Article Outline

Keywords

Dynamic programming; Infinite horizon problems; Discounted problem

Dynamic programming addresses models of decision making systems of an inherent sequential character. The problem of interest is defined as follows. We consider a discrete-time dynamic system:

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, 1, \ldots .$$

The state transitions, f, that define the evolution of the system from time k to time $k + 1$ depend on the current state of the system, x_k, external disturbances, ω_k, which are considered to be random variables, and finally on a set of control, or policy, actions, u_k. The state of the system, x_k, $k = 0, 1, \ldots$, is an element of a space S, the control variables, u_k, $k = 0, 1, \ldots$, belong to space C, and the random external disturbance belongs to a countable space D. The control variables are such that: $u_k \in U(x_k) \subset C$, $k = 0, 1, \ldots$, and depend on the current state x_k, $k = 0, 1, \ldots$. The random disturbances, ω_k, $k = 0, 1, \ldots$, have identical, known, distributions which depend on the current state and control, $P(\omega_K \mid x_k, u_k)$. Note that ω_k does not depend on previous values of the disturbances, but may depend explicitly on the values of x_k, and u_k. Given an initial state x_0, the problem is to find a control law $\pi = \{\mu_0, \mu_1, \ldots\}$, belonging to the set of *admissible policies*, Π, which is the set of all sequences of functions $\pi = \{\mu_0, \mu_1, \ldots\}$ with:

$$\mu_k:\ S \to C, \quad \mu_x(x_k) \in U(x_k),$$
$$\forall x_k \in S, \quad k = 0, 1, \ldots,$$

that minimizes the cost functional:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left\{ \sum_{k=0}^{k=N-1} \alpha^k g_k(x_k, u_k, \omega_k) \right\} .$$

The optimal cost function J^* is thus defined as:

$$J^*(x) = \min_{\pi\in\Pi} J_\pi(x), \quad x \in S.$$

The cost, $J_\pi(x_0)$, for any $x_0 \in S$ and a given policy π, represents the limit of the expected finite horizon costs and these are well defined. The *discounted problems with bounded cost per stage* are such that the following assumption holds:

Assumption 1

1) $\forall(x, u, \omega) \in S \times C \times D$ the functions defining the cost per stage g are *uniformly* bounded:

$$0 \le |g_k(x_k, u_k, \omega_k)| \le M;$$

2) $M \in \mathbf{R}$, and $0 < \alpha < 1$.

This type of problem was first address through the pioneering work of D. Blackwell, [6]. The scalar, α, is the *discount factor*, and the range of its admissible values implies that future costs matter less that costs incurring at the present time, particularly when the cost per stage has a monetary interpretation. Mathematically, the presence of the discount factor guarantees the finiteness of the cost functional provided that the per stage costs are bounded uniformly. Furthermore, although the assumption of an infinite number of stages may never be satisfied in practice, it constitutes a reasonable approximation for problems involving a large number of stages. A rather typical example of a dis counted infinite horizon dynamic problem is the so-called *asset selling problem* where the reward for selling a particular asset at a given time k diminishes as time progresses.

For any function J: $S \to \mathbf{R}$ we define the operator $(T(\cdot))$ as:

$$(TJ)(x) = \min_{u\in U(x)} \mathsf{E}\{g(x, u, \omega) + \alpha J(f(x, u, \omega))\}.$$

This is in essence the function obtained when applying the standard dynamic programming mapping to J. Note that (TJ) represents essentially the optimal cost for a one-stage problem that has stage cost g and terminal cost αJ. For this operator, it can be shown, [4], that:

- For any bounded function J, the optimal cost function satisfies:

$$J^*(x) = \lim_{N\to\infty} (T^N J)(x), \quad \forall x \in S.$$

In other words, the *dynamic programming algorithm* converges to the optimal cost function. The above result relies on Assumption 1. It should be noted that the operator (TJ) can be shown to be:

1) *monotonic*:

$$J(x) \le J'(x) \;\Rightarrow\; (T^k J)(x) \le (T^k J')(x)$$

for any functions J: $S \to \mathbf{R}$ and J': $S \to \mathbf{R}$ and,

2) *contractive*:

$$\max_{x\in S} \left|(T^k J)(x) - (T^k J'(x)\right| \le \alpha^k \max_{x\in S} \left|J(x) - J'(x)\right|$$

for any bounded functions J: $S \to \mathbf{R}$ and J': $S \to \mathbf{R}$.

Both of these properties are important so as to show not only theoretical convergence of the dynamic programming algorithm but also to construct numerical solution schemes. Furthermore, the optimal cost function J^* can be shown to satisfy *Bellman's equation*, i. e., $\forall x \in S$:

$$J^*(x) = \min_{u \in U(x)} \mathsf{E}\{g(x,u,\omega) + \alpha J^*(f(x,u,\omega))\};$$

in other words: $J^* = TJ^*$. This proposition essentially defines the necessary and sufficient condition for the optimality of a policy μ, i. e. $\mu(x)$ is optimal if and only if it attains the minimum in Bellman's equation for every $x \in S$.

For the case where the state, control and disturbance space are finite, i. e., each set has a definite number of elements which can be found in principle, [9], several approaches exist for numerically solving the discounted problem with bounded cost per stage. It should be pointed out that under these conditions the problem is equivalent to a *finite-state Markov chain*. The first, *value iteration*, is based on a successive computation of TJ, T^2J, …, since we know that $\lim_k \to \infty (T^kJ) = J^*$. Recall that the operator (TJ) is defined as the minimum over all possible disturbances with respect to the controls. Therefore, asymptotically we approach the optimal cost as well as the optima policy. Tight upper and lower bounds on the iterations can be derived, [3,4], which substantially improve the convergence rate of the successive approximations. More specifically, it can shown that for every vector J, state i, and time k:

$$(T^kJ)(i) + \underline{c}_k \le J^*(i) \le (T^kJ)(i) + \overline{c}_k,$$

where:

$$\underline{c}_k = \frac{\alpha}{1-\alpha} \min_{i=1,\dots,n} [(T^kJ)(i) - (T^{k-1}J)(i)],$$
$$\overline{c}_k = \frac{\alpha}{1-\alpha} \max_{i=1,\dots,n} [(T^kJ)(i) - (T^{k-1}J)(i)].$$

In fact, these error bounds can be used so as to further prove the finite convergence of the value iteration after $k < k'$ steps, $k' \in \mathbf{N}$. It can also be observed that instead of performing the value iteration simultaneously for all policies, one can perform the iteration in a *Gauss–Seidel* fashion, [10]. The contractive characteristics of the operator (FJ) make it possible to develop similar schemes. Instead of iterating on the operator (TJ), we define a new sequence based on the operator (FJ):

$$(FJ)(1) = \min_{u \in U(1)} \left[g(1,u) + \alpha \sum_{j=1}^{N} p_{1j}(u)J(j) \right],$$
$$(FJ)(i) = \lim_{u \in U(i)} \left[g(i,u) + \alpha \sum_{j=1}^{i-1} p_{ij}(FJ)(j) + \alpha \sum_{j=i}^{n} p_{ij}(u)J(j) \right],$$
$$i = 1, \dots, n\,.$$

In fact, when the error bounds are not used, a very interesting property can be shown, [4]:

- If J satisfies:

  $$J(i) \le (TJ)(i) \le J^*(i), \quad i = 1, \dots, n,$$

  then:

  $$(T^kJ)(i) \le (F^kJ)(i) \le J^*(i),$$
  $$i = 1, \dots, n; \quad k = 1, 2, \dots.$$

In other words, the Gauss–Seidel iteration converges faster than the ordinary, i. e., *Jacobi*, value iteration. An excellent treatment of the comparisons between Gauss–Seidel and Jacobi iterations and their parallel implementation can be found in [14]. Although the value iteration can be shown to be convergent even when the state and control spaces are infinite, the actual implementation can only proceed via approximations. In other words, instead of actually computing TJ we can only compute J', such that: $\max_{x \in S} |J'(x) - (TJ)(x)| \le \epsilon$. For such approximate methods to be in order, we do not necessarily need infinite spaces but even spaces with a very large number of states in which the actual computation is deemed inappropriate. Any function J' that satisfies the above criterion can in principle be used. Details regarding discretization approaches and computational techniques for addressing infinite state spaces can be found in [2,11].

The value iteration, thus far presented, is based on successive evaluations of the cost functions. Early on, [1], it was suggested that an alternate approach is to iterate on policies so as to generate sequences of stationary policies with improved, over the preceding one, costs. This method is know as the *policy iteration*. The method proceeds in three steps:

1) initialize control policy, u^0.
2) given a stationary policy, μ^k, evaluate the cost function J_{μ^k} by solving:

$$(I - \alpha P_{\mu^k}) J_{\mu^k} = g_{\mu^k}.$$

3) Obtain a new policy such that it satisfies: $T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$.

In the above, the matrix P_μ is the *transition probability matrix* for a given stationary policy μ, given by:

$$P_\mu = \begin{pmatrix} p_{11}(\mu(1)) & \cdots & p_{1n}(\mu(1)) \\ \cdots & \cdots & \cdots \\ p_{n1}(\mu(1)) & \cdots & p_{nn}(\mu(n)) \end{pmatrix}$$

and g_μ the associated cost vector:

$$g_\mu = \begin{pmatrix} g(1, \mu(1)) \\ \cdots \\ g(n, \mu(n)) \end{pmatrix}.$$

Termination is detected once $J_{\mu^k} = TJ_{\mu^k}$, i. e., a *fixed point* of the operator TJ has been identified. Notice that because of the assumption that the policy space is finite, the algorithm will terminate in a finite number of steps. Similarly to the value iteration, infinite state and control spaces pause problems when implementing policy iterations. Specifically, the policy evaluation and policy improvement steps can only be performed via approximations.

In [5] an *adaptive aggregation method* is proposed so as to address the issue of occasional slow convergence. The fundamental premise is to lump states of the original problem so as to generate a smaller dimension problem. In other words, the state space S is partitioned into smaller-dimensional spaces as: $S = S_1 \cup \cdots \cup S_m$. Given such a partitioning one can further define the transition probabilities for the aggregate states as:

$$r_{ij} = \sum_{s \in S_i} q_{is} \sum_{t \in S_j} p_{st}(\mu(s)),$$

which is the probability that the next state will belong to S_j given that the current state is S_i. q_{ij} are the elements of an $m \times n$ matrix Q, such that $q_{is} \neq 0$, if $s \in S$.

Finally, [7], noticed that since in the limit $J \leq J^* = TJ^*$, the optimal policy can be derived as the solution of the following *linear programming problem*:

$$\begin{cases} \max & \sum_{i \in S} \lambda_i \\ \text{s.t.} & \lambda_i \leq g(i, u) + \alpha \sum_{j=1}^{n} p_{ij}(u) \lambda_i, \\ & u \in U(i), \\ & i = 1, \dots, n. \end{cases}$$

In the above formulation $p_{ij}(u)$ denote the transition probabilities: $p_{ij}(u) = \{ \mathrm{P}(x_{k+1} = j | \ x_k = i, u_k = u)\}$, $i, j \in S$, $u \in U(i)$. These can either be given or derived based on the discrete dynamic system, $x_{k+1} = f(x_k, u_k, \omega_k)$, and the known probability distribution $\mathrm{P}(\cdot|x, u)$ of the input disturbance ω_k. Linear programming formulations can also be used to derived cost and policy evaluation approximations. One possibility is to approximate J^* by a set of known basis functions as: $J'(x, r) = \sum_{k=1}^{m} r_k \omega_k(x)$. The vector r is an m-dimensional vector of known parameters, and for each state x we have chose a set of known scalars $\omega_k(x)$. The vector r can be determined as the solution of:

$$\begin{cases} \max & \sum_{x \in S'} J'(x, r) \\ \text{s.t.} & J'(x, r) \leq g(x, u) + \alpha \sum_{y \in S} p_{xy} J'(y, r), \\ & x \in S' \subset S, \\ & u \in U'(x) \subset U(x). \end{cases}$$

Furthermore, the cost function J_μ for a given policy μ can be approximated via linear programming formulations by identifying a vector r so as to minimize: $\max_{x \in S} | J'(x, r) - J_\mu(x)|$. This can be shown, [4], to be equivalent to solving:

$$\begin{cases} \min & z \\ \text{s.t.} & \Big| J'(x, r) - g(x, \mu(x)) \\ & \qquad - \alpha \sum_{y \in S} p_{xy}(\mu(x)) J'(y, r) \Big| \leq z, \\ & x \in S' \subset S. \end{cases}$$

Extensions of the general ideas are discussed in [12] where work on including constraints in the general formulation of the discounted dynamic programming problem is presented. Furthermore, [8] expanded the scope of these models so as to address dynamic programming optimization problems involving multiple criteria by identifying the set of non-inferior, *Pareto optimal*, solutions.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Bellman R (1957) Applied dynamic programming. Princeton Univ. Press, Princeton
2. Bertsekas DP (1975) Converence of discretization procedures in dynamic programming. IEEE Trans Autom Control AC-20:415–419
3. Bertsekas DP (1976) On error bounds for successive approximation methods. IEEE Trans Autom Control AC-21:396–396
4. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
5. Bertsekas DP, Castanon DA (1989) Adaptive aggregation methods for infinite horizon dynamic programming. IEEE Trans Autom Control AC-34:589–598
6. Blackwell D (1965) Discounted dynamic programming. Ann Math Statist 33:719–726
7. D'Epenoux F (1963) Sur un probleme de production et de stockage dans l'aleatoire. Managem Sci 10:98–108English transl.
8. Ghosh MK (1990) Markov decision processes with multiple costs. Oper Res Lett 9:257–260
9. Kolmogorov AN, Fomin SV (1970) Introductory real analysis. Dover, Mineola, NY
10. Kushner HJ (1971) Introduction to stochastic control. Holt, Rinehart and Winston, New York
11. Puterman ML (1978) Dynamic programming and its applications. Acad. Press New York, New York
12. Ross KW (1989) Randomized and past–dependent policies for Markov decision processes with multiple constraints. Oper Res 37:474–477
13. Shapley LS (1953) Stochastic games. Proc Nat Acad Sci USA 39
14. Tsitsiklis JN (1989) A comparison of Jacobi and Gauss–Seidel parallel iterations. Appl Math Lett 2:167–170

Dynamic Programming: Infinite Horizon Problems, Overview

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C39, 90C40

Article Outline

Keywords

Dynamic programming; Infinite horizon problems

Dynamic programming deals with situations where optimal decisions are being sought in systems operating in stages. Events occur in a specific order, such that the decision at time $k+1$ depends on the state of the system at time k. In general the key variables of the basic formulation are as follows:

- k represents discrete time;
- x_k represents the state of the system at time k;
- $\mu(x_k)$ represents the control, or decision, variable to be selected at time k;
- ω_k represents a random disturbance occurring at time k;
- N represents the time horizon.

Given the aforementioned variables, the basic dynamic programming formulation requires the following ingredients:

- a discrete-time dynamic system:

$$x_{k+1} = f_k(x_k, \mu_k, \omega_k);$$

- an additive cost function of the form:

$$g_N(x_N) + \sum_{k=1}^{k=N-1} g_k(x_k, \mu_k, \omega_k),$$

where g_k corresponds to the cost incurred at time k. One is therefore wishing to identify that control policy, $\pi = \{\mu_0, \dots, \mu_{N-1}$, which minimizes the expected cost:

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0) = \mathsf{E}\left\{ g_N(x_N) + \sum_{k=1}^{k=N-1} g_k(x_k, \mu_k, \omega_k) \right\}.$$

The expectancy operator is needed since the presence of the random parameters ω_k the cost function becomes itself a random variable. As further complication, one might also minimize the expected cost not only for a given initial state of the system, x_0, but also with respect to all possible initial states.

Infinite horizon problems are further characterized by the fact that the number of stages N is infinite. In such a case, the cost functional over an infinite number of stages for a given control policy $\pi = \{\mu_0, \mu_1, \dots\}$, and initial state x_0, is given by:

$$J_\pi(x_0) = \lim_{n \to \infty} \mathsf{E}\left\{ \sum_{k=1}^{k=N-1} \alpha^k g_k(x_k, \mu_k, \omega_k) \right\}.$$

The factor α is termed *discount factor* and is a positive scalar $0 < \alpha \leq 1$ which simply implies that future costs matter less than similar costs incurred at the present time. Infinite horizon problems are by definition the limit of the corresponding N-stage problem, as $N \to \infty$. Three points are pivotal in the analysis of infinite-dimensional dynamic programming problems:

- The optimal cost for the infinite horizon is the limit of the corresponding N-stage optimal cost, i. e., $J^* = \lim_{N \to \infty} J_N$.
- The optimal costs satisfy *Bellman's equation*, i. e.,

$$J^*(x) = \min_{u \in U(x)} \mathsf{E}\{g(x, \mu, w) + J^*(f(x, \mu, w))\}.$$

- If the optimal policy that correspond to the minimum of Bellman's equation is $\mu(x)$, then the policy $\pi = \{\mu, \mu, \dots\}$ should be optimal.

The assumption of an infinite number of stages may not be satisfied in practice but is a very important one in terms of analyzing the asymptotic behavior of systems involving a finite but large number of stages. Depending on the nature of the cost per stage and the discount factor, the following categories of infinite horizon dynamic programming problems can be identified, [1]:

- *stochastic shortest path problems*: this problem is actually a generalization of the *deterministic shortest path problem* in the sense that we select not a successor but rather a probability distribution $p_{ij}(\mu)$. Obviously, if the probability $p_{ij}(\mu) = 1$ for a unique state j, then we recover the deterministic shortest path problem. One key feature of the stochastic shortest path problem is that the termination state t is cost-free termination state such that once the system reaches that state it never leaves from it. In other words, $p_{tt}(\mu) = 1$ and $g(t, \mu) = 0$, for all policies μ. In effect, the horizon is finite but the actual length is random. Furthermore, there exists at least one policy for which the destination state will be reached inevitably. A key assumption required for guaranteeing eventual termination states that there exists an integer m such that for every initial state and policy, there is a positive probability that the termination state will be reached after no more than m stages.
- *discounted problems with bounded cost per stage*: this type of infinite horizon dynamic programming encompasses problems for which:

$$|g(x, \mu, \omega)| \leq M, \ \forall (x, \mu, \omega) \in S \times C \times D,$$

i. e., there exists a finite scalar, M, that bounds the per stage cost. Furthermore, the discount factor is such that $0 < \alpha < 1$.
Both of these conditions are important so as to show that:

$$\lim_{K \to \infty} \mathsf{E}\left\{ \sum_{k=N}^{K} \alpha^k g(x_k, \mu(x_k), \omega_k) \right\} \to 0.$$

Boundedness and discounting results in successive approximation mappings which are *contraction mappings*, [2], thus proving the convergence of such schemes to the optimal solution of the discounted with bounded costs infinite horizon dynamic programming problems.
- *undiscounted problems*: this type of infinite horizon problems covers situations in which the discount

factor $\alpha = 1$, which greatly complicates the analysis. The key distinction is that the lack of a discount factor may result in infinite costs even when the per cost stage is bounded.

- *average cost per stage problems*: In cases where neither discounting nor a cost-free termination state exists, it is often meaningful to optimize the average per stage cost starting from state i.

$$J(i) = \lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left\{ \sum_{k=1}^{N-1} g(x_k, \mu_k(x_k), \omega_k) : \ x_0 = i \right\} .$$

In essence what is assumed is that for most problems of interest the average and the optimal per stage cost are independent of the initial state. As a result, costs that incurred in the early stages do not matter since their contributions vanishes, i. e.,

$$\lim_{N\to\infty} \frac{1}{N} \mathsf{E}\left\{ \sum_{k=0}^{K} g(x_k, \mu_k(x_k), \omega_k) \right\} = 0 .$$

For discrete state and transition spaces, it is helpful to consider the associated finite-state Markov chain. Let the state space S consist of n states, denoted by $1, \dots, n$:

$$S = \{1, \dots, n\} .$$

The transitions probabilities from state i to state j are:

$$p_{ij}(u) = \mathsf{P}(x_{k+1} = j | x_k = i, \ u_k = u), \qquad i, j \in S, \quad u \in U(i) .$$

The dynamics of the state transitions $x_{k+1} = f(x_k, u_k, \omega_k)$ can actually be used to compute the state transitions. Given the above, the per stage expected cost can be expressed as: $g(i, u) = \sum_{j=1}^{n} p_{ij}(u)\, g'(i, u, j)$ Given the above definitions, a very important mapping can now be defined:

$$(TJ)(i) = \min_{u\in U(i)} \left[g(i, u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j) \right] , \qquad i = 1, 2, \dots ,$$

and also

$$(T_\mu J)(i) = g(i, \mu(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\mu(i)) J(j), \qquad i = 1, 2, \dots ,$$

This operator can actually be written as:

$$T_\mu J = g_\mu + \alpha P_\mu J .$$

Therefore, a stationary policy has a corresponding cost, J_μ, which is the solution to the equation:

$$(I - \alpha P_\mu) J_\mu = g_\mu .$$

Computationally, two major families of approaches exists for determining the optimal additive costs and the optimal policies. The first one, *value iteration*, is based on the idea of successive approximations. It can be shown, under conditions depending of the specific type of infinite horizon problem, that:

$$\lim_{k\to\infty} (T^k J)(i) = J^*(i) .$$

This property essentially implies that the successive application of the mapping (TJ) will in the limit provide the optimal cost.

On the other hand *policy iteration* operates on the policy space and tries to identify a converging sequence of stationary policies converging to the optimal one. In all cases, the following basic three steps define the iteration:

- *Initialization*: guess an initial stationary policy, mu^0.
- *Policy evaluation*: given a stationary policy, μ^k, compute the corresponding cost function, J_{μ^k} from the system:

$$(I - \alpha P_{\mu^k}) J_{\mu^k} = g_{\mu^k} .$$

- Policy improvement: obtain a new stationary policy satisfying:

$$T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k} .$$

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control

- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Optimization Strategies for Dynamic Systems

References

1. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
2. Shapley LS (1953) Stochastic games. Proc Nat Acad Sci USA 39

Dynamic Programming: Inventory Control

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20

Article Outline

Keywords

Dynamic programming; Inventory control

Consider the problem of ordering a quantity of a certain item at each of the N periods so as to meet some stochastic demand. In mathematical terms the problem is defined as follows:

- x_k, the stock of a particular commodity available at the beginning of the kth period.
- u_k the stock to be ordered and immediately delivered at the beginning of the kth period.
- ω_k the demand during the kth period, whose probability distribution is assumed to be known.

The demand distributions are assumed to be independent random variables for each time period k. A simple stock balance at the beginning of each time period provides the description of the discrete-time evolution equation as:

$$x_{k+1} = x_k + u_k - \omega_k.$$

In other words, the state of the system (stock) at the beginning of period $k + 1$ was the state of the system (stock) at period k plus the ordered stock minus the demand at period k. The form of the replenishment of policy is very important and sits at the hart of the analysis of similar problems. The one just presented is, as will be seen, one of the two major assumptions regarding the stock balance equations. Given the above definitions, the cost incurred at period k has two components:

- a cost $r(x_k)$ representing either a penalty for positive stock, storage, or negative stocks, shortage for unfilled demand.
- a surcharging cost, cu_k, where c is the per unit surcharged cost.

The problem just described is known as the *inventory control problem*, one of the most important ones in the area of operations research. The preceding formulation illustrates the main characteristics of the inventory control problem:

- a discrete-time system that defines the system evolution in time of the form:

$$x_k = f_k(x_k, u_k, \omega_k);$$

- a set of independent random disturbances, representing commodity demands;
- a set of control constraints that depend on the state of the system at time k, x_k, that is $u_k \in U(x_k)$;
- a period of N time intervals over which the operating cost has an additive form as:

$$\mathrm{E}\left(g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, \omega_k)\right);$$

and, finally,

- we wish to optimally select the control actions at every time interval k, so as to optimize over all possible control policies the cost of operating the inventory system.

Clearly, the above definition of the inventory control problem, formulates the problem as *dynamic programming* problem in which we try to minimize an expected additive cost function.

Stochastic inventory problems were first considered by [6,10], were an abstract stochastic inventory model that allowed for possible constraints on the inventory after ordering were considered. In the literature of stochastic inventory models, there are two different assumptions about the excess demand unfilled from existing inventories: the *backlog assumption* and the *lost sales assumption*. These affect the form of the stock balance equation. The backlog assumption is historically more popular in the literature because of the inventory studies with spare parts inventory management problems. This assumption essentially states that an unfilled demand is being accumulated and satisfied at later times. The lost sales assumption states that unfilled demand is lost, which is the situation arising in retail establishments. Under either assumption, an important issue has been to establish the optimality of the (s, S)-type policy, [7]. It defines a very simple replenishment rule:

$$\mu^*(x_k) = \begin{cases} S_k - x_k, & x_k < s_k, \\ 0, & x_k \geq s_k. \end{cases}$$

The above rule is referred to as the (s, S) *policy*, implying that when the current level is less than the reorder point, s, an order up to the reorder level, S, has to be placed. Under an (s, S) policy if the inventory level at the beginning of a period is less than the reorder point s, then a sufficient quantity must be re-ordered to achieve an inventory level S upon replenishment. The key concept of *K-convexity*, [1], was instrumental in proving the optimality of the (s, S) policies. A real-valued function g is *K-convex*, where $K \geq 0$, if:

$$K + g(z + y) \geq g(y) + z\left(\frac{g(y) - g(y - b)}{b}\right), \quad \forall z \geq 0,\ b > 0,\ y.$$

The parameter K is the fixed cost associated with a positive inventory order:

$$C(u) = \begin{cases} K + cu, & u > 0, \\ 0, & u = 0. \end{cases}$$

The concept of K-convexity is one of the most important tools for the analysis of inventory control problem. It essentially expands the concept of convexity and is instrumental in proving the optimality of policies in inventory control problems. Regarding K-convexity, [2], the following hold true:

1) A real-valued convex function g is also 0-convex and hence also K-convex for all $K \geq 0$.
2) If $g_1(y)$ and $g_2(y)$ are K-convex and L-convex ($K \geq 0$, $L \geq 0$), respectively, then $\alpha g_1(y) + \beta g_2(y)$ is $(\alpha K + \beta L)$-convex for all $\alpha > 0$ and $\beta > 0$.
3) If $g(y)$ is K-convex and ω is a random variable, then $\mathsf{E}_\omega\{g(y - \omega)\}$ is also K-convex, provided $\mathsf{E}_\omega\{|g(y - \omega)|\} < \infty$, for all y.
4) If g is a continuous K-convex function and $g(y) \to \infty$ as $|y| \to \infty$, then there exist scalars s and S with $s \leq S$ such that:
   a) $g(S) \leq g(y)$, for all scalars y;
   b) $g(S) + K = g(s) < g(y)$, for all $y < s$;
   c) $g(y)$ is a decreasing function on $(-\infty, s)$;
   d) $g(y) \leq g(z) + K$, for all y, z with $z \leq y \leq z$.

If we further define a holding/storage cost as:

$$r(x) = p \max(0, -x) + h \max(0, x),$$

the function H as:

$$H(y) = p\mathsf{E}\left(\max(0, \omega_k - y)\right) + \mathsf{E}\left(\max(0, y - \omega_k)\right).$$

Application of the *dynamic programming algorithm* for zero final cost gives:

$$J_k(x_k) = \min \left\{ G_k(x_k), \min_{u_k > 0}\{K + cu_k + G_k(x_k + u_k)\} \right\},$$

with $G_k(x_k)$ defined as:

$$G_k(y) = cyH(y) + \mathsf{E}(J_{k+}1(y - \omega))$$

and $y_k = x_k + u_k$. Because of the K-convexity of G, it can actually be shown, [7], that the (s, S) policy is optimal. See [11] for the optimality of the (s, S) policy in the case of lost sales. For the case where the unfilled demand is not backlogged but rather lost, the system dynamic equation is defined as:

$$x_{k+1} = \max(0, x_k + u_k - \omega_k).$$

Additional K-convexity results and the optimality of the (s, S) for the case of lost-sales is presented and analyzed in [4]. Finite storage capacity in most real-life situations imposes an upper bound on theory that can be kept. The recent analysis of [3] considers the multi-product inventory model with stochastic demands and warehousing constraints. This is a fairly general model in that it does not allow for surplus disposal, $u_k \geq 0$, and imposes constraints on the stored stock, $x_k + u_k \in \Gamma$.

Similar ideas pertain the analysis of inventory control problems over an infinite horizon. *Infinite horizon problems* need not necessarily correspond to physically realistic situations, but nevertheless, they define the vehicle for a thorough analysis of the asymptotic response of the inventory system. A discounted version of the backlogged problem can be stated as:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathsf{E}\left(\sum_{k=1}^{N-1} \alpha^k (c\mu_k(x_k) + H(x_k + \mu(x_k) - \omega_k))\right),$$

where:

$$H(y) = p \max(0, -y) + h \max(0, y).$$

The case of $\alpha < 1$, i. e., a *discounted infinite horizon problem*, has also been analyzed, [8], and the existence of an optimal state-dependent (s, S)-type policy for problems with discounted costs was rigorously established.

The classical papers [5,10] were also devoted to stochastic inventory problems with the criterion of long-run average cost. In other words, one is interested in minimizing an average expected cost within an infinite horizon

$$J_\pi(x_0) = \lim_{N\to\infty} \frac{1}{N} \times \mathsf{E}\left(\sum_{k=1}^{N-1} \alpha^k (c\mu_k(x_k) + H(x_k + \mu(x_k) - \omega_k))\right).$$

This analysis also concludes that (s, S)-type policies are as well optimal for the long time average-return problem. The (s, S)-type of optimal policies are very important and they have be shown to be optimal for a wide variety of inventory problems including systems with continuous demands and discrete order sizes, [9], in other words for the cases where the orders u_k are assumed to be nonnegative integers, as well as the case where special structure in the form of periodicity of various components of the formulation such as demands, prices, and cost, [2].

Undoubtably, one of the most appealing features of inventory theory has been the fact that (s, S) policies are optimal for the class of dynamic inventory problems with random demands. However, real-life inventory problems impose constraints that make the assumption imposed on the analysis apparently too restrictive. The nature of demand, for instance, is an important factor in determining optimal policies. Classical models have assumed demand in each period to be a random variable independent of demands in other periods and of environmental factors at other times. Nevertheless, fluctuating economic conditions and uncertain market conditions can have a major effect. Furthermore, various constraints are observed in real life that limit the nature of ordering decisions nd inventory levels. The recent work of [8] addresses similar issues so as to incorporate cyclic or seasonal demand, as well as constraints imposed on the ordering periods, storage and service level constraints. Nevertheless, it is still shown that (s, S) policies are also optimal for these types of generalized models.

See also

- ► Dynamic Programming: Average Cost Per Stage Problems
- ► Dynamic Programming in Clustering
- ► Dynamic Programming: Continuous-time Optimal Control
- ► Dynamic Programming: Discounted Problems
- ► Dynamic Programming: Infinite Horizon Problems, Overview
- ► Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ► Dynamic Programming: Optimal Control Applications
- ► Dynamic Programming: Stochastic Shortest Path Problems
- ► Dynamic Programming: Undiscounted Problems
- ► Hamilton–Jacobi–Bellman Equation
- ► Multiple Objective Dynamic Programming
- ► Neuro-dynamic Programming

References

1. Bertsekas DP (1976) Dynamic programming and stochastic control. Acad. Press, New York
2. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
3. Beyer D, Sethi SP, Sridhar R (1997) Stochastic multi–product inventory models with limited storage. Working Paper The Univ. Texas at Dallas, Richardson, TX

4. Cheng F, Sethi SP (1997) Optimality of state-dependent (*s*, *S*) policies with Markovian demand and lost sales. Production and Operations Management
5. Iglehart D (1963) Optimality of (*s*, *S*) policies in the infinite horizon dynamic inventory problem. Managem Sci 9: 259–267
6. Ignall EJ, Veinott A (1969) Optimality of myopic inventory policies for several substitue products. Managem Sci 18:284–204
7. Scarf H (1960) The optimality of (*s*, *S*) policies in the dynamic inventory problem. In: Arrow J, Karlin S, Suppes P (eds) Math. Methods in Social Sciences. Stanford Univ. Press, Palo Alto, CA
8. Sethi SP, Cheng F (1997) Optimality of (*s*, *S*) policies in invetory models with markovian demand. Oper Res 45: 931–939
9. Tsitsiklis JN (1984) Periodic review inventory systems with continuous demand and discrete order sizes. Managem Sci 10:1250–1254
10. Veinott A (1965) Optimal policy for a multi-product, dynamic nonstationary inventory problem. Managem Sci 12:206–222
11. Veinott A (1966) On the optimality of (*s*, *S*) inventory policies: New conditions and a new proof. SIAM J Appl Math 14:1067–1083

Dynamic Programming and Newton's Method in Unconstrained Optimal Control

JOSEPH C. DUNN
Math. Department, North Carolina State University, Raleigh, USA

MSC2000: 49M29, 65K10, 90C06

Article Outline

Keywords

Dynamic programming; Newton method; Unconstrained optimal control

Discrete-Time Optimal Control

The cost function in the standard k-stage discrete-time optimal control problem is defined by

$$J = l_{k+1}(x_{k+1}) + \sum_{i=1}^{k} l_i(u_i, x_i) \tag{1}$$

and the recursion

$$\begin{cases} x_1 = a, \\ x_{i+1} = f_i(u_i, x_i), \quad i = 1, \dots, k. \end{cases} \tag{2}$$

In these equations, u_i is a *control vector* in $\mathbf{R}^m$ and x_i is a *state vector* in $\mathbf{R}^n$. For each vector-valued k-tuple $u = (u_1, \dots, u_k)$ in the direct sum $\mathbf{R}^{km} = \oplus_{i=1}^{k} \mathbf{R}^m$, there is a unique state vector-valued $(k+1)$-tuple $x(u) = (x_1(u), \dots, x_{k+1}(u)) \in \mathbf{R}^{(k+1)n}$ satisfying (2), and a corresponding unique value $J(u)$ in (1). For present purposes, the state transition functions f_i, the terminal loss function l_{k+1} and the stage-wise loss functions l_i, $i = 1, \dots, k$, are assumed to be twice continuously differentiable. The functions $x(\cdot)$ and $J(\cdot)$ are then also twice continuously differentiable, and Newton's method is formally applicable to the problem of minimizing $J(\cdot)$ over $\mathbf{R}^{km}$. Moreover, for fixed m and n the $km \times km$ linear system associated with the Newton iteration map for (1), (2) can be solved efficiently with *dynamic programming recursions* in $O(k)$ floating point operations as k increases without bound. In contrast, it requires $O(k^3)$ floating point operations to assemble and solve the Newtonian linear system for a general cost function J on $\mathbf{R}^{km}$ by standard Gaussian elimination methods.

The following discussion conforms to [4]. See [7] and [8] for an alternative development with connections to *differential dynamic programming*, and for a related but nonequivalent treatment of discrete-time optimal control problems based on the Riccati transformation. For analogous constructions in the setting of continuous-time optimal control problems, see [4] and the original papers [5] and [6]. For extensions to Newtonian projection methods and input-constrained optimal control problems, see [2] and [3].

Newton's Method

If J is any continuously differentiable real-valued function on $\mathbf{R}^N$ with global or local minimizing vectors $\overline{u}$, then all such vectors must satisfy the *first order necessary condition*,

$$\nabla J(u) = 0, \tag{3}$$

where

$$\nabla J(u) = \left(\frac{\partial J}{\partial u_1}(u), \dots, \frac{\partial J}{\partial u_N}(u) \right).$$

A solution of (3) is called a *stationary point*.

Condition (3) comprises N scalar equations in N scalar unknowns u_i. If J is a quadratic function, then (3) is a linear system which can be treated with standard elimination algorithms or other techniques capable of exploiting whatever structure may exist in the coefficient matrix for (3). On the other hand, if J is a nonquadratic nonlinear function, then (3) is a nonlinear system and iterative methods are generally needed to generate successive approximations to a solution of (3). One such method is Newton's recursive linearization scheme,

$$\begin{aligned} &u \to u + \overline{v}, \\ &\nabla J(u) + \nabla^2 J(u)\overline{v} = 0. \end{aligned} \tag{4}$$

When J is twice continuously differentiable, the vector-valued map $\nabla J(\cdot)\colon \mathbf{R}^N \to \mathbf{R}^N$ is continuously differentiable and its first differential at u is the Hessian operator $\nabla^2 J(u)\colon \mathbf{R}^N \to \mathbf{R}^N$ defined by

$$(\nabla^2 J(u)v)_i = \sum_{i=1}^{N} \frac{\partial^2 J}{\partial u_i \partial u_j}(u) v_j$$

for $i = 1, \dots, N$. In such cases, (4) is formally applicable to the nonlinear system (3). Furthermore, if $\nabla^2 J(\overline{u})$ is invertible at a solution $\overline{u}$ for (3), then in some neighborhood N of $\overline{u}$, $\nabla^2 J(u)$ is also invertible and for each starting point $u^0 \in N$, the iteration (4) generates a sequence of vectors, $u^0, u^1, \dots$, which remain in N and converge rapidly to $\overline{u}$. More precisely, either $u^i = \overline{u}$ eventually, or the errors $\left\| u^i - \overline{u} \right\| = (\langle u^i - \overline{u}, u^i - \overline{u} \rangle)^{\frac{1}{2}}$ satisfy the *superlinear convergence condition*,

$$\lim_{i \to \infty} \frac{\left\| u^{i+1} - \overline{u} \right\|}{\left\| u^i - \overline{u} \right\|} = 0. \tag{5}$$

A solution of (3) at which $\nabla^2 J(u)$ is invertible is said to be a *regular stationary point*. Note that solutions of (3) can be local maximizers or saddle points of J, and that regular points of this kind can also attract the Newton iterates. Hence for minimization problems, a simple steepest descent iteration is often employed at the outset to seek out likely starting points u^0 for (4) near some *regular local minimizer* for J.

If J is twice continuously differentiable, then every global or local minimizer $\overline{u}$ must also satisfy the *second order necessary condition*,

$$\forall v \in \mathbb{R}^N, \quad \left\langle v, \nabla^2 J(\overline{u})v \right\rangle \geq 0, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product,

$$\langle v, w \rangle = \sum_{i=1}^{N} v_i w_i .$$

In $\mathbf{R}^N$, $\overline{u}$ is therefore a regular local minimizer if and only if $\nabla^2 J(\overline{u})$ is *positive definite*, i. e.,

$$\forall v \in \mathbb{R}^N, \quad v \neq 0 \Rightarrow \left\langle v, \nabla^2 J(\overline{u})v \right\rangle > 0. \tag{7}$$

The gap between (6) and (7) is not large, hence regular local minimizers are commonly encountered in $\mathbf{R}^N$.

By continuity, property (7) extends to $\nabla^2 J(u)$ in some neighborhood of $\overline{u}$. At each fixed u in this neighborhood, the linear system in Newton's iteration (4) is equivalent to a corresponding unconstrained *accessory minimum problem*

$$\overline{v} \in \arg\min_{v \in \mathbb{R}^N} \phi(v) \tag{8}$$

with a strictly convex quadratic cost function

$$\phi(v) = \langle \nabla J(u), v \rangle + \frac{1}{2} \left\langle v, \nabla^2 J(u)v \right\rangle, \tag{9}$$

and a unique global minimizer $\overline{v}$. This equivalence is computationally significant for unconstrained minimization problems in general and discrete-time optimal control problems in particular.

The Accessory Minimum Problem

If J is the cost function of a discrete-time optimal control problem, then it can be shown that the accessory minimum problem (8)–(9) is also a discrete-time control problem with quadratic loss functions and linear state transition functions. Near a regular minimizer for J, this linear-quadratic (LQ) problem has a strictly convex cost function that can be minimized with dynamic programming recursions. The required control-theoretic construction for ϕ and the related dynamic programming algorithms are outlined below.

In the following development, the symbol u may denote a vector in $\mathbf{R}^m$ or a vector-valued k-tuple in $\mathbf{R}^{km}$. Similarly, x may indicate a vector in $\mathbf{R}^n$ or a vector-valued $(k+1)$-tuple in $\mathbf{R}^{(k+1)n}$, and the bracket $\langle \cdot, \cdot \rangle$ may denote the Euclidean inner product in any of the spaces $\mathbf{R}^m$, $\mathbf{R}^n$, $\mathbf{R}^{km}$, or $\mathbf{R}^{(k+1)n}$. In each case, the correct interpretation is always clear from the context. Now suppose that $J(\cdot)$ is defined by (1)–(2) on $\mathbf{R}^{km}$, and fix $u \in \mathbf{R}^{km}$. Then for all $v \in \mathbf{R}^{km}$ the chain rule gives,

$$\begin{aligned}\langle \nabla J(u), v\rangle &= \frac{d}{ds} J(u+sv)|_{s=0} \\ &= \sum_{i=1}^{k+1} \langle \nabla_x l_i, y_i \rangle + \sum_{i=1}^{k} \langle \nabla_u l_i, v_i \rangle \end{aligned} \tag{10}$$

and

$$\begin{aligned}&\left\langle v, \nabla^2 J(u) v \right\rangle = \frac{d^2}{ds^2} J(u+sv)|_{s=0} \\ &= \sum_{i=1}^{k+1} \langle \nabla_x l_i, z_i \rangle + \sum_{i=1}^{k+1} \left\langle y_i, \nabla_{xx}^2 l_i y_i \right\rangle \\ &\quad + 2\sum_{i=1}^{k} \left\langle y_i, \nabla_{xu}^2 l_i v_i \right\rangle + \sum_{i=1}^{k} \left\langle v_i, \nabla_{uu}^2 l_i v_i \right\rangle, \end{aligned} \tag{11}$$

where

$$\begin{aligned} y_i &= \frac{d}{ds} x_i(u+sv)|_{s=0}, \\ z_i &= \frac{d^2}{ds^2} x_i(u+sv)|_{s=0}, \end{aligned} \tag{12}$$

and where all partial gradients and Hessians of l_{k+1} and l_i, $i = 1, \ldots, k$, are evaluated at $x_{k+1}(u) \in \mathbf{R}^n$ and $(u_i, x_i(u)) \in \mathbf{R}^m \oplus \mathbf{R}^n$, respectively.

Equations (2) and (12) and the chain rule also establish that y_i and z_i are recursively generated by the equations of variation,

$$\begin{cases} y_1 = 0, \\ y_{i+1} = A_i y_i + B_i v_i, \quad i = 1, \ldots, k, \end{cases} \tag{13}$$

and

$$\begin{cases} z_1 = 0, \\ z_{i+1} = A_i z_i \\ \quad + (C_i y_i) y_i + 2(D_i y_i) v_i + (E_i v_i) v_i \end{cases} \tag{14}$$

for $i = 1, \ldots, k$, with linear differential maps,

$$A_i = \frac{\partial f_i}{\partial x}, \qquad B_i = \frac{\partial f_i}{\partial u}$$

and

$$C_i = \frac{\partial^2 f_i}{\partial x \partial x}, \qquad D_i = \frac{\partial^2 f_i}{\partial x \partial u}, \qquad E_i = \frac{\partial^2 f_i}{\partial u \partial u}$$

evaluated at $(u_i, x_i(u))$. Useful control-theoretic representations for $\nabla J(u)$ and ϕ can now be constructed by removing y_i from formula (10) and z_i from formula (11) with the aid of an *adjoint recursion* for (13) and (14).

Equations (13) and (14) are special instances of

$$\begin{cases} w_1 = 0, \\ w_{i+1} = A_i w_i + \xi_i, \quad i = 1, \ldots, k, \end{cases} \tag{15}$$

with $w = (w_1, \ldots, w_{k+1}) \in \mathbf{R}^{(k+1)n}$ and $\xi = (\xi_1, \ldots, \xi_k) \in \mathbf{R}^{kn}$. For each ξ, there is a unique $w = \Phi\xi$ satisfying (15), and the resulting correspondence defines a linear map $\Phi\colon \mathbf{R}^{kn} \to \mathbf{R}^{(k+1)n}$. The map Φ has an associated *adjoint linear map* $\Phi^*\colon \mathbf{R}^{(k+1)n} \to \mathbf{R}^{kn}\}$ which, in principle, is uniquely determined by the requirement,

$$\langle \Phi^* \eta, \xi \rangle = \langle \eta, \Phi\xi \rangle\,, \tag{16}$$

imposed for all $\xi \in \mathbf{R}^{kn}$ and $\eta \in \mathbf{R}^{(k+1)n}$. The matrix representor for Φ^* in the standard basis for $\mathbf{R}^{(k+1)n}$ is obtained by transposing the analogous matrix representor for Φ; however, the adjoint map can also be computed directly with recursions derived from (15), *without prior construction of* Φ. More precisely, for each $\eta \in \mathbf{R}^{(k+1)n}$,

$$(\Phi^* \eta)_i = \psi_{i+1} \tag{17}$$

for $i = 1, \ldots, k$, where ψ is the unique solution of the adjoint recursion,

$$\begin{cases} \psi_{k+1} = \eta_{k+1}, \\ \psi_i = A_i^* \psi_{i+1} + \eta_i, \quad i = 1, \ldots, k. \end{cases} \tag{18}$$

To see this, note that if w and ψ are solutions of (15) and (18) respectively, then

$$\begin{aligned}\langle \eta_{k+1}, w_{k+1}\rangle &= \langle \psi_{k+1}, w_{k+1}\rangle - \langle \psi_1, w_1\rangle\\ &= \sum_{i=1}^{k}(\langle \psi_{i+1}, w_{i+1}\rangle - \langle \psi_i, w_i\rangle)\\ &= \sum_{i=1}^{k}\langle \psi_{i+1}, A_i w_i + \xi_i\rangle\\ &\quad - \sum_{i=1}^{k}\langle A_i^* \psi_{i+1} + \eta_i, w_i\rangle\\ &= \sum_{i=1}^{k}\langle \psi_{i+1}, \xi_i\rangle - \sum_{i=1}^{k}\langle \eta_i, w_i\rangle\,.\end{aligned}$$

Hence for all $\xi \in \mathbf{R}^{kn}$ and $\eta \in \mathbf{R}^{(k+1)n}$ condition (16) gives,

$$\langle \Phi^* \eta, \xi\rangle = \langle \eta, \Phi\xi\rangle = \langle \eta, w\rangle = \sum_{i=1}^{k}\langle \psi_{i+1}, \xi_i\rangle\,,$$

and this establishes (17).

With the preceding formulas, it is now possible to write ϕ as a sum of linear and quadratic terms in the variables $v_1, \ldots, v_k$ and $y_1, \ldots, y_{k+1}$, with coefficients derived from the partial gradients and Hessians of *Hamiltonian functions*,

$$\begin{aligned}&H_i(u_i, x_i, \psi_{i+1})\\ &\qquad = l_i(u_i, x_i) + \langle \psi_{i+1}, f_i(u_i, x_i)\rangle\,. \qquad (19)\end{aligned}$$

Fix u and v in $\mathbf{R}^{km}$, let $\eta_i(u) = \nabla_x\, l_i$ for $i = 1, \ldots, k+1$, and let $\psi(u) \in \mathbf{R}^{(k+1)n}$ be the corresponding solution of the adjoint recursion,

$$\begin{cases}\psi_{k+1} = \nabla_x l_{k+1},\\ \psi_i = A_i^* \psi_{i+1} + \nabla_x l_i, \quad i = 1, \ldots, k.\end{cases} \qquad (20)$$

In addition, let y and z be the unique solutions of (13) and (14) respectively. Then with reference to (15)–(17) and (20),

$$\begin{aligned}\sum_{i=1}^{k+1}\langle \nabla_x l_i, y_i\rangle &= \sum_{i=1}^{k}\langle \psi_{i+1}, B_i v_i\rangle\\ &= \sum_{i=1}^{k}\langle B_i^* \psi_{i+1}, v_i\rangle\end{aligned}$$

and

$$\begin{aligned}\sum_{i=1}^{k+1}\langle \nabla_x l_i, z_i\rangle &= \sum_{i=1}^{k}\langle \psi_{i+1}, (C_i y_i) y_i\rangle\\ &+ 2\sum_{i=1}^{k}\langle \psi_{i+1}, (D_i y_i) v_i\rangle + \sum_{i=1}^{k}\langle \psi_{i+1}, (E_i v_i) v_i\rangle\,.\end{aligned}$$

When these expressions are substituted into (9)–(11), it follows from (19) that ϕ is prescribed by

$$\phi(v) = q_{k+1}(y_{k+1}) + \sum_{i=1}^{k} q_i(v_i, y_i) \qquad (21)$$

and the recursions,

$$\begin{cases}y_1 = 0,\\ y_{i+1} = A_i y_i + B_i v_i, \quad i = 1, \ldots, k,\end{cases} \qquad (22)$$

where A_i and B_i are the differential maps

$$A_i = \frac{\partial f_i}{\partial x}, \qquad B_i = \frac{\partial f_i}{\partial u}$$

as before, and the loss functions q are given by

$$q_{k+1}(y) = \frac{1}{2}\langle y, Q_{k+1} y\rangle$$

and

$$\begin{aligned}q_i(v, y) &= \langle r_i, v\rangle + \frac{1}{2}\langle y, Q_i y\rangle\\ &\qquad + \langle y, R_i v\rangle + \frac{1}{2}\langle v, S_i v\rangle\,,\end{aligned}$$

for $i = 1, \cdots, k$, with

$$\begin{aligned}Q_{k+1} &= \nabla^2_{xx} l_{k+1},\\ r_i &= \nabla_u H_i,\end{aligned}$$

and

$$Q_i = \nabla^2_{xx} H_i, \quad R_i = \nabla^2_{xu} H_i, \quad S_i = \nabla^2_{uu} H_i$$

for $i = 1, \ldots, k$. Moreover, the cost gradient $\nabla J(u)$ is separately recoverable from

$$\begin{aligned}\nabla J(u) &= (r_1, \ldots, r_k)\\ &= (\nabla_u H_1, \ldots, \nabla_u H_k)\,. \qquad (23)\end{aligned}$$

In these equations, the Hessian of l_{k+1} is evaluated at $x_{k+1}(u)$ and the Hamiltonian gradients and Hessians are evaluated at $(u_i, x_i(u), \psi_{i+1}(u))$.

Dynamic Programming Recursions

Recall that $\nabla^2 J(u)$ is positive definite in some neighborhood of a regular local minimizer for J. When $\nabla^2 J(u)$ is positive definite, the quadratic accessory minimum problem with cost function (21)–(22) can be solved by dynamic programming techniques, which rest on a simple embedding scheme and a few elementary theorems stated below without proof. For a fuller discussion of dynamic programming, see [1].

For $j = 1, \dots, k$ and $y \in \mathbf{R}^n$, define the family of cost functions $\phi_j(\cdot;y)$: $\mathbf{R}^{(k+1-j)m} \to \mathbf{R}^1$ by

$$\phi_j(v_j, \dots, v_k; y) = q_{k+1}(y_{k+1}) + \sum_{i=j}^{k} q_i(v_i, y_i) \quad (24)$$

and the recursions,

$$\begin{cases} y_j = y \\ y_{i+1} = A_i y_i + B_i v_i, \quad i = j, \dots, k, \end{cases} \quad (25)$$

where q_i, A_i and B_i are u-dependent entities defined as before. Evidently, the cost function ϕ in (21)–(22) is recovered from the equation,

$$\phi(v) = \phi_1(v_1, \dots, v_k; 0). \quad (26)$$

Moreover, the cost functions ϕ_j are recursively generated by

$$\phi_k(v_k; y) = q_k(v_k; y) + q_{k+1}(A_k y + B_k v_k)$$

and

$$\phi_j(v_j, \dots, v_k; y) = q_j(v_j; y) + \phi_{j+1}(v_{j+1}, \dots, v_k; A_j y + B_j v_j),$$

for $j = k - 1, \dots, 1$. It is likewise readily seen that

$$\phi_j(v_j, \dots, v_k; 0) = \phi(0, \dots, 0, v_j, \dots, v_k),$$
$$\nabla^2_{vv}\phi_j(v_j, \dots, v_k; y) = \nabla^2_{vv}\phi_j(v_j, \dots, v_k; 0),$$

and

$$\nabla^2_{vv}\phi(v) = \nabla^2 J(u)$$

for $j = 1, \dots, k$, $v \in \mathbf{R}^{km}$ and $y \in \mathbf{R}^n$. Note also that since $\phi(\cdot)$ and $\phi_j(\cdot;y)$ are quadratic functions, their corresponding v-Hessians are independent of v as well as y. These facts and the basic principles of dynamic programming yield the following theorems.

Theorem 1 *The following statements are equivalent:*

1) *The quadratic function $\phi(\cdot)$ has a unique global minimizer $\overline{v} \in \mathbb{R}^{km}$.*
2) *$\nabla^2 J(u)$ is positive definite.*
3) *For all $j = 1, \dots, k$, $\nabla^2_{vv}\phi_j$ is positive definite.*
4) *For all $j = 1, \dots, k$ and all $y \in \mathbf{R}^n$, the quadratic function $\phi_j(\cdot;y)$ has a unique global minimizer $(\overline{v}_j, \dots, \overline{v}_k) \in \mathbb{R}^{(k+1-j)m}$*

Theorem 2 *The following statements are equivalent:*

1) *For all $j = 1, \dots, k$ and $y \in \mathbf{R}^n$,*

$$\phi_j^0(y) := \inf_{v_j, \dots, v_k} \phi_j(v_j, \dots, v_k; y) > -\infty, \quad (27)$$

2) *The real-valued functions $\phi_1^0(\cdot), \dots, \phi_k^0(\cdot)$ satisfy the backward functional recursion,*

$$\phi_j^0(y) = \inf_{v \in \mathbb{R}^m} [q_j(v, y) + \phi_{j+1}^0(A_j y + B_j v)] \quad (28)$$

for $j = k, \dots, 1$, with

$$\phi_{k+1}^0(y) := q_{k+1}(y).$$

Theorem 3 *Let r_i, A_i, B_i, Q_i, R_i, and S_i be the vectors and linear maps appearing in the representations (21)–(22) and (24)–(25) for the functions $\phi(\cdot)$ and $\phi_j(\cdot;y)$. Then the following statements are equivalent for all $\overline{v} = (\overline{v}_1, \dots, \overline{v}_k) \in \mathbb{R}^{km}$:*

1) *$\overline{v}$ is the unique global minimizer for ϕ in $\mathbf{R}^{km}$, i. e.,*

$$\arg\min_{v \in \mathbb{R}^{km}} \phi(v) = \{\overline{v}\}.$$

2) *The vector $\overline{v} \in \mathbb{R}^{km}$ is generated by the forward recursions*

$$\begin{cases} y_1 = 0, \\ y_{j+1} = A_j y_j + B_j \overline{v}_j, \end{cases} \quad (29)$$

$$\begin{aligned} \{\overline{v}_j\} &= \arg\min_{v \in \mathbb{R}^m} \left[q_j(v, y_j) + \phi_{j+1}^0(A_j y_j + B_j v)\right] \\ &= \{\gamma_j + \Gamma_j y_j\}, \end{aligned} \quad (30)$$

for $j = 1, \dots, k$, where

$$q_j(v, y) = \langle r_j, v\rangle + \frac{1}{2}\langle y, Q_j y\rangle + \langle y, R_j v\rangle + \frac{1}{2}\langle v, S_i v\rangle,$$

$$\phi_j^0(y) = \alpha_j + \langle \beta_j, y\rangle + \frac{1}{2}\langle y, \Theta_j y\rangle,$$

$S_j + B_j^* \Theta_{j+1} B_j$ is positive definite,

$$\gamma_j = -(S_j + B_j^* \Theta_{j+1} B_j)^{-1}(r_j + B_j^* \beta_{j+1}), \quad (31)$$

$$\Gamma_j = -(S_j + B_j^* \Theta_{j+1} B_j)^{-1}(R_j^* + B_j^* \Theta_{j+1} A_j), \quad (32)$$

and the linear maps Θ_j, vectors β_j, and numbers α_j satisfy the backward recursions,

$$\begin{cases} \Theta_{k+1} = 0, \\ \Theta_j = Q_j + A_j^* \Theta_{j+1} A_j \\ \quad + (R_j^* + B_j^* \Theta_{j+1} A_j)^* \Gamma_j, \end{cases} \quad (33)$$

$$\begin{cases} \beta_{k+1} = 0, \\ \beta_j = (A_j + B_j \Gamma_j)^* \beta_{j+1} + \Gamma_j^* r_j, \end{cases} \quad (34)$$

and

$$\begin{cases} \alpha_{k+1} = 0, \\ \alpha_j = \alpha_{j+1} - \frac{1}{2}\left\langle \gamma_j, (S_j + B_j^* \Theta_{j+1} B_j)\gamma_j \right\rangle \end{cases} \quad (35)$$

for $j = k, \ldots, 1$.

These theorems support the following efficient scheme for computing the Newton increment $\overline{v}$ in (4).

```
1  Given $u \in \mathbf{R}^{km}$, solve the forward recursion (2) for $x(u)$, and construct the corresponding linear maps $A_j$ and $B_j$, and vectors $\eta_j = \nabla_x l_j$.
2  Solve the backward adjoint recursion (20) for $\psi(u)$ and construct the corresponding vectors $r_j$.
3  Construct the linear maps $Q_i$, $R_i$ and $S_i$, solve the backward dynamic programming recursions (33) and (34) for $\Theta_j$ and $\beta_j$, and compute $\gamma_j$ and $\Gamma_j$ in (31) and (32).
4  Solve the forward recursions (29)–(30) for $y$ and $\overline{v}$.
```

Algorithm

Stages 1 and 2 in the foregoing algorithm are always well-posed, and yield the cost gradient $\nabla J(u)$ (see (23)). The calculation for the Newton increment $\overline{v}$ is well-posed if and only if stage 3 produces invertible linear maps $S_j + B_j^* \Theta_{j+1} B_j$ for $j = k, \ldots, 1$. The calculation for $\overline{v}$ is well-posed *and* stage 3 concludes with k positive definite linear maps $S_j + B_j^* \Theta_{j+1} B_j$ if and only if $\nabla^2 J(u)$ is positive definite. If a positive semidefinite, indefinite or singular linear map $S_j + B_j^* \Theta_{j+1} B_j$ is encountered at some point in stage 3, it follows that $\nabla^2 J(u)$ is not positive definite and the accessory minimum problem may have no global minimizers or stationary points, or infinitely many such points. In such cases, it may be advantageous or even necessary to abort stage 3 and abandon Newton's method temporarily in favor of a descent iteration that employs the negative gradient $-\nabla J(u)$ computed in stages 1 and 2, or some other descent direction. Alternative *quasi-Newtonian* descent directions can be obtained by replacing S_j in stage 3 with $S_j + \lambda_j I$ where $\lambda_j I$ is a positive shift added where necessary to maintain positive definiteness of $S_j + \lambda_j I + B_j^* \Theta_{j+1} B_j$. This variant of stage 3 is automatically well-posed and produces the unique global minimizer $\overline{v}$ of the perturbed cost function,

$$\phi(v) + \frac{1}{2}\langle v, \Lambda(u)v \rangle ,$$

where

$$\Lambda(u)v = (\lambda_1 v_1, \ldots, \lambda_k v_k).$$

By construction, $\nabla^2 J(u) + \Lambda(u)$ is positive definite and

$$\overline{v} = -(\nabla^2 J(u) + \Lambda(u))^{-1} \nabla J(u).$$

Hence $\overline{v}$ is a descent vector for J. On the other hand, the simple steepest descent direction $-\nabla J(u)$ may be more cost-efficient, particularly when u is far from a local minimizer for J.

If the work required to compute the differentials for f_j and l_j in each time step j is uniformly bounded in j, with m and n fixed, then the number of arithmetic operations required to execute the foregoing algorithm (or its shifted variants) is directly proportional to k. This compares very favorably with the standard $O(k^3)$ estimate for general Newtonian calculations in $\mathbf{R}^{km}$.

Finally, references [9] and [10] revise the basic serial dynamic programming algorithm for parallel computation, and thereby achieve significant reductions in the *time* needed to calculate each Newton iteration.

See also

► Automatic Differentiation: Calculation of Newton Steps

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ Interval Newton Methods
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Nondifferentiable Optimization: Newton Method
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control
- ▶ Unconstrained Nonlinear Optimization: Newton–Cauchy Framework

References

1. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Bertsekas DP (1982) Projected Newton methods for optimization methods with simple constraints. SIAM J Control Optim 20:221–246
3. Dunn JC (1988) A projected Newton method for minimization problems with nonlinear inequality constraints. Numerische Math 53:377–409
4. Dunn JC, Bertsekas DP (1989) Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems. J Optim Th Appl 63: 23–38
5. Merriam CW (1964) An algorithm for the iterative solution of a class of two-point boundary value problems. SIAM J Control Optim Ser A 2:1–10
6. Mitter SK (1966) Successive approximation methods for the solution of optimal control problems. Automatica 3:135–149
7. Pantoja J (1988) Differential dynamic programming and Newton's method. Internat J Control 47:1539–1553
8. Polak E (1971) Computational methods in optimization. Acad. Press, New York
9. Wright SJ (1990) Solution of discrete-time optimal control problems on parallel computers. Parallel Comput 16:221–238
10. Wright SJ (1991) Partitioned dynamic programming for optimal control. SIAM J Optim 1:620–642

Dynamic Programming: Optimal Control Applications

REIN LUUS
Department Chemical Engineering, University Toronto, Toronto, Canada

MSC2000: 93-XX

Article Outline

Keywords

Optimal control; Optimization; Approximation algorithms; Singular control; Time-delay; Nonseparable problem; Sensitivity

A very powerful method for optimization of a system that can be separated into stages is *dynamic programming* developed by R. Bellman [1]. The main concept of this technique lies in the *principle of optimality* which can be stated as follows:

> An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Many engineering systems are in the form of individual stages, or can be broken into stages, so the idea of breaking up a complex problem into simpler subproblems so that optimization could be carried out systematically by optimizing the subproblems was received enthusiastically. Numerous applications of the principle of optimality in dynamic programming were given in [2], and there was a great deal of interest in applying dynamic programming to optimal control problems. In the 1960s many books and numerous papers were written to explore the use of dynamic programming as a means of optimization for optimal control problems. Since an optimal control problem, involving *optimization over a trajectory*, can be broken into a sequence of time stages, it appeared that dynamic programming would be ideally suited for such problems.

Although dynamic programming could be successfully applied to some simple optimal control problems, one of the greatest problems in using dynamic programming, however, was the *interpolation problem* encountered when the trajectory from a grid point did not reach exactly the *grid point* at the next stage [12]. This interpolation difficulty coupled with the dimensionality restriction and the requirement of a very large number of grid points limited the use of dynamic programming to only very simple optimal control problems. The limitations imposed by the '*curse of dimensionality*' and the '*menace of the expanding grid*' for solving optimal control problems kept dynamic programming from being used for practical types of optimal control problems, until R. Luus [14] suggested effective means of overcoming both the interpolation and the dimensionality problems.

Optimal Control Problem

We consider the continuous dynamic system described by the vector differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1}$$

with the initial state $\mathbf{x}(0)$ given, where $\mathbf{x}$ is an $(n \times 1)$ state vector and $\mathbf{u}$ is an $(m \times 1)$ control vector bounded by

$$\alpha_j \le u_j(t) \le \beta_j, \quad j = 1, \dots, m. \tag{2}$$

The *performance index* associated with this system is a scalar function of the state at the given final time t_f; i. e.,

$$I = \Phi(\mathbf{x}(t_f)). \tag{3}$$

We may have also state constraints, but for simplicity we shall leave these for later. The optimal control problem is to find the control $\mathbf{u}$ in the time interval $0 \le t < t_f$, so that the performance index in (3) is either minimized or maximized. To set up the problem into a staged form, we may approximate the optimal control problem by requiring a *piecewise constant control* policy instead of a continuous control policy, over P stages, each of length L, so that

$$L = \frac{t_f}{P}, \tag{4}$$

and we can consider the system at the grid points set up at these P stages. We may also use a piecewise linear approximation and the stages do not necessarily have to be of equal length.

Iterative Dynamic Programming

M. DeTremblay and Luus [10] suggested that instead of interpolation, an approximation can be used when the trajectory from a grid point does not reach a grid point at the next stage. They suggested that the control policy that was found to be optimal for the closest grid point be used to continue the integration to the next stage. If a large number of grid points are taken at each stage then a reasonable approximation may be obtained, but the resulting control policy can still be quite far from the optimum. Therefore, this simplification by itself gives only a crude approximation.

However, by making a small change to the procedure, the *accuracy* with which the optimum is obtained can be improved substantially. This change requires the use of the procedure repeatedly in an iterative fashion [16], so that after every iteration, where the best value is used as the center point, the regions for the allowable values for control and for the grid points are reduced in size. The idea of *region reduction* in optimization was successfully used by Luus and T.H.I. Jaakola [37] in *direct search optimization*. As was shown by Luus [16], dynamic programming can be used in this fashion to give a sufficiently accurate optimal control policy. Although easy to program, the method was not computationally attractive until the idea of using *accessible states* as grid points [14]. With this latter change, the method was recognized as a *feasible approach* to solving optimal control problems.

The advantage of generating the state grid points is also that the dimensionality of the state vector then does not matter. The application of the method to a nonlinear system described by 7 differential equations and having 4 control variables was solved quite easily [15]. Also the method was used for system of *difference equations*, which is actually easier, since no discretization is necessary [40]. In essence, the 'curse of dimensionality' was eliminated and the new computational procedure became known as *iterative dynamic programming* (*IDP*).

Early Applications of IDP

Iterative dynamic programming provided a very convenient way of investigating the effect of the choice of the final time in optimal control problems [18]. However, by generating the grid points, it was no longer possible to guarantee a *global optimum*. This was illustrated by Luus and M. Galli [36]. Even the use of a very large number of grid points does not guarantee getting the global optimum. In fact, the number of grid points can be quite small in many cases and the global optimum is still obtained with good accuracy [4].

A very challenging problem is the bifunctional catalyst problem, where it is necessary to determine the blend of the catalyst along a tubular reactor to maximize the yield of a desired component [35]. By using *successive quadratic programming* (*SQP*) and starting from 100 randomly chosen starting points, 26 local optima were located, but the global optimum was not obtained. With IDP, however, the global optimum was readily obtained with the use of a single grid point [34]. To avoid the numerous local optima, all that was required for this system was to take a sufficiently large initial region size for the control.

Although the optimal control of *fed-batch reactors* was very difficult to obtain by methods based on *Pontryagin's maximum principle*, iterative dynamic programming provided a reliable means of obtaining the global optimum, and the results were even marginally better than had been previously reported [20,22]. The additional advantage of IDP is that the computations are straightforward and the algorithm can be easily programmed to run on a personal computer.

Choice of Candidates for Control

In the early work with IDP, the test values for the control variables were chosen over a uniform distribution. This was easy to program and was easy to visualize. For each control variable we could have a minimum of 3 values, namely $-r$, 0, and r, where r is the region size. For m control variables we must examine then 3^m candidates at each grid point. This is fine if m is less than 4, but if m is large, this number becomes excessively large.

An alternative method for choosing candidates for control was suggested by V. Tassone and Luus [47], but a better approach as shown by B. Bojkov and Luus [3] was to choose such candidates at random inside the allowable range. This meant that in theory there was no upper limit on m. Conceptually m could be greater than 100. In fact, IDP was used successfully on a system with 130 differential equations and 130 control variables [21] and later with 250 differential equations with 250 control variables [26].

Piecewise Linear Continuous Control

In the early work with IDP, the given time interval was divided into P time stages of equal length and at each time-stage we would have constant control. In many cases the optimal control policy is quite smooth, and therefore it may be beneficial to approximate the control policy by linear sections. This, indeed, gives a better result with a smaller number of time stages as was shown by Luus [23], and allowed an optimal control policy for very high-dimensional systems to be determined accurately [15,26]. For a *piecewise linear control* we calculate the control policy in the time interval (t_k, t_{k+1}) by the expression

$$\mathbf{u}(t) = \mathbf{u}(k) + \frac{\mathbf{u}(k+1) - \mathbf{u}(k)}{L}(t - t_k), \qquad (5)$$

where $\mathbf{u}(k)$ is the value of $\mathbf{u}$ at the time t_k and $\mathbf{u}(k+1)$ is the value of $\mathbf{u}$ at time t_{k+1}.

Algorithm for IDP

To illustrate the underlying logic in IDP, an algorithm is given to solve the optimal control problem as outlined in (1)-(4), where it is required to minimize the performance index in (3) with the use of piecewise constant control over P stages, each of same length:

1) Divide the time interval $[0, t_f]$ into P time stages, each of length L.
2) Choose the number of test values for $\mathbf{u}$, denoted by R, an initial control policy and the initial region size $\mathbf{r}_{in}$; also choose the region contraction factor γ used after every iteration and the number of grid points N.
3) Choose the total number of iterations to be used in every pass and set the iteration number index to $j = 1$.
4) Set the region size vector $\mathbf{r}^{(j)} = \mathbf{r}_{in}$.
5) By using the best control policy (the initial control policy for the first iteration) as reference, integrate (1) from $t = 0$ to t_f N times with different values for control inside the allowable region to generate N**x**-trajectories and store the values of $\mathbf{x}$ at the beginning of each time stage as grid points, so that $\mathbf{x}(k-1)$ corresponds to the value of $\mathbf{x}$ at beginning of stage k.
6) Starting at stage P, corresponding to time $t_f - L$, for each of the N stored values for $\mathbf{x}(P-1)$ from step 5 (grid points) integrate (1) from $t_f - L$ to t_f, with each of the R allowable values for the control vector calculated from

$$\mathbf{u}(P-1) = \mathbf{u}(P-1)^{*(j)} + \mathbf{D}\mathbf{r}^{(j)}, \qquad (6)$$

where $\mathbf{u}(P-1)^{*(j)}$ is the best value obtained in the previous iteration and $\mathbf{D}$ is a diagonal matrix of different random numbers between -1 and 1. Out of the R values for the performance index, choose the control values that give the minimum value, and store these values as $\mathbf{u}(P-1)$. We now have the best control for each of these N grid points.
7) Step back to stage $P-1$, corresponding to time $t_f - 2L$, and for each of the N grid points do the following calculations. Choose R values for $\mathbf{u}(P-2)$ as in the previous step, and by taking as the initial state $\mathbf{x}(P-2)$ integrate (1) over one stage length. Continue integration over the last time stage by using the stored value of $\mathbf{u}(P-1)$ from step 6 by choosing the control policy corresponding to the grid point that is closest to the values of the state vector that has been reached. Compare the R values of the performance index and store the $\mathbf{u}(P-2)$ that gives the minimum value for the performance index.
8) Continue the procedure until stage 1, corresponding to the initial time $t = 0$ and the given initial state, is reached. This stage has only a single grid point, since the initial state is specified. As before, integrate (1) and compare the R values of the performance index and store the control $\mathbf{u}(0)$ that gives the minimum performance index. Store also the corresponding $\mathbf{x}$-trajectory.
9) Reduce the region for allowable control

$$\mathbf{r}^{(j+1)} = \gamma \mathbf{r}^{(j)}, \qquad (7)$$

where j is the iteration number index. Use the best control policy from step 8 as the midpoint for the allowable values for the control denoted by the superscript $*$.
10) Increment the iteration index j by 1 and go to step 5 and continue the procedure for the specified number of iterations and interpret the results.

The application of this algorithm is illustrated with several examples in [33], where also the computer program in FORTRAN is given for IDP.

Time-Delay Systems

The great advantage of IDP over Pontryagin's maximum principle is that no auxiliary variables have to be calculated and no derivatives are required. The state equation is integrated forward and there is no need to integrate any equations backward. Therefore, the method is applicable to more complex systems, such as *time-delay systems*. The initial attempt to apply IDP to time-delay systems was made by S.A. Dadebo and Luus [8]. By using piecewise linear continuous control very good results for a difficult nonlinear time-delay CSTR system were obtained by Luus et al. [43]. The method is further illustrated in [32].

State Constraints

Control constraints actually simplify the problem by decreasing the range over which the admissible values of control are to be taken. Research in how to handle *state constraints* is still continuing, but already very useful results have been obtained. As was shown in [39] and [17], the use of penalty functions appears to be the best way to deal with state constraints. The best type of penalty function has not yet been firmly established. Although Dadebo and K.B. McAuley [9] suggested the use of absolute value type of penalty function for state equality constraints, the recent work of Luus [27], and Luus and C. Storey [41] show that a quadratic penalty function with shifting terms also works very well. The advantage of using the quadratic penalty function with shifting terms is that, at the optimum, the shifting terms yield useful sensitivity information with respect to the constraints. Handling of state inequality constraints can be achieved by introducing through differential equations auxiliary variables that are increased in value whenever the constraint is violated and then including these auxiliary variables at the final time as penalty functions in the augmented performance index [46]. The use of differential equations is better than the use of difference equations as was used by Luus [17], because this will prevent violation of the constraint inside a time stage. The auxiliary variables when incorporated into the augmented performance index through a penalty function with a sufficiently large penalty function factor thus prevent a violation of the state constraint anywhere in the time interval.

Singular Control Problems

When Pontryagin's maximum principle is used, computational difficulties arise if the Hamiltonian is not an explicit function of the control for a portion of the trajectory. Such problems do not arise when IDP is used, and therefore this area was investigated by using IDP. The early work [19] showed that IDP can be used without much difficulty for such problems, and Luus [29] was able to obtain solutions to *singular control problems* that had eluded many investigators. For these problems the main difficulty is the very low *sensitivity* of the performance index on control.

Sensitivity of Control Policy

Especially for batch reactors, it is found that the cause of computational difficulties lies in the sensitivity of control policy with respect to the yield that is to be maximized [24]. Whereas we are not concerned with more than four figure accuracy in the yield, we would nevertheless like to know what the optimal control policy is. The very low sensitivity was brought out by Luus [25] where in the optimal control of a fed-batch reactor, it was shown that the optimal control policy is relatively smooth.

Use of Variable Stage-Lengths

Bojkov and Luus [5,6] suggested the use of flexible stage-lengths in IDP for *time optimal control problems* where the time of *switching* is very important. For general type of optimal control problems the use of *variable stage-lengths* enabled the optimum to be more accurately obtained, and in some instances the local optima encountered with the use of stages of fixed length could be avoided [7]. The problem of applying this idea to problems where the final time was specified was overcome by the use of shifting terms in a quadratic penalty function [27]. The use of flexible stage-lengths provides a means of obtaining accurate *switching times* and allowed some optimal control problems, that had gone by unsolved for several decades, to be readily solved [29]. The use of variable stage lengths and a quadratic penalty function with shifting terms enables time optimal control problems to be solved directly [31], so that a difficult boundary value problem is avoided. Further illustration of the usefulness of variable stage lengths is given in [30] and [33].

Nonseparable Problems

Problems where the performance index is a function of all the control variables and states, and where separation into stages as required for dynamic programming is not possible, constitutes and interesting class of problems. D. Li and Y.Y. Haimes [13] suggested a method of tackling such problems, and Luus and Tassone [42] considered the application of IDP to *nonseparable problems*. Luus [28] showed that even for complex nonseparable problems a large number of grid points is not necessary, and the optimum can be obtained quite readily. If the number of stages is large, then IDP has a great advantage over direct search optimization where optimization is carried out simultaneously over all the stages. The best means for the application of IDP to nonseparable problems are still to be determined. The strategy of using the values for some of the variables from previous iteration appears to work very well, however.

Future Directions

Iterative dynamic programming has been developed into a useful optimization procedure. As has been shown in [11], IDP has certain advantages over other optimization procedures for the optimization of a fed-batch reactor. The *reliability* of getting the *global optimum* is very high. Now that the personal computers have become very powerful, the method can be easily used on very complex optimal control problems. When G. Marroquin and W.L. Luyben [44] suggested operating a batch reactor at its best isothermal temperature as the set point, the computational power of the existing computers was relatively low and the cost of computation was very high. It appeared then that optimal control could not be used for realistic systems. Now, however, we can, in effect, have a *feedback control* if the measurements of the pertinent state variables can be done sufficiently fast, by solving the optimal control problem many times during the time of operation of the batch reactor. If the trend in the enhancement of computer speed continues, we can use realistic models and carry out optimization 'on-line', so that optimal control calculations can be carried out during the operation and the required changes in the control can be immediately implemented. Then the optimal control's application will not be only for investigation of design possibilities, but will constitute an important part of the actual operation of the process.

The viability of using IDP for on-line optimal control has been illustrated for reactor control by Luus and O.N. Okongwu [38].

Since derivatives are not required in the use of IDP, the method is applicable to more general types of optimal control problems. Also, since no *auxiliary variables* are necessary, except to handle state inequality constraints, the method is easier to use than variational methods based on Pontryagin's maximum principle. As convergence properties of IDP are studied in greater detail, further improvements will inevitably be introduced, to make IDP even more useful. Luus [30] showed that variable stage lengths can be incorporated into optimal control problems where *state inequality constraints* are also present, by combining the approach of Bojkov and Luus [7] along with that of W. Mekarapiruk and Luus [46]. Although the best choice for the *penalty function* to be used in IDP has not yet been established, good progress has been made in this field [45] and further research in this area is continuing. Furthermore, since no *derivatives* are required for IDP, the method should have important applications where *nondifferentiable functions* are encountered.

See also

- ▶ Control Vector Iteration
- ▶ Duality in Optimal Control with First Order Differential Equations
- ▶ Dynamic Programming: Average Cost per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-Time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Infinite Horizon Control and Dynamic Games
- ▶ MINLP: Applications in the Interaction of Design and Control
- ▶ Multi-objective Optimization: Interaction of Design and Control

- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-Dynamic Programming
- ▶ Optimal Control of a Flexible Arm
- ▶ Optimization Strategies for Dynamic Systems
- ▶ Robust Control
- ▶ Robust Control: Schur Stability of Polytopes of Polynomials
- ▶ Semi-Infinite Programming and Control Problems
- ▶ Sequential Quadratic Programming: Interior Point Methods for Distributed Optimal Control Problems
- ▶ Suboptimal Control

References

1. Bellman R (1957) Dynamic programming. Princeton Univ. Press, Princeton
2. Bellman R, Dreyfus S (1962) Applied dynamic programming. Princeton Univ. Press, Princeton
3. Bojkov B, Luus R (1992) Use of random admissible values for control in iterative dynamic programming. Ind Eng Chem Res 31:1308–1314
4. Bojkov B, Luus R (1993) Evaluation of the parameters used in iterative dynamic programming. Canad J Chem Eng 71:451–459
5. Bojkov B, Luus R (1994) Time-optimal control by iterative dynamic programming. Ind Eng Chem Res 33:1486–1492
6. Bojkov B, Luus R (1995) Time optimal control of high dimensional systems by iterative dynamic programming. Canad J Chem Eng 73:380–390
7. Bojkov B, Luus R (1996) Optimal control of nonlinear systems with unspecified final times. Chem Eng Sci 51:905–919
8. Dadebo S, Luus R (1992) Optimal control of time-delay systems by dynamic programming. Optimal Control Appl Meth 13:29–41
9. Dadebo SA, McAuley KB (1995) Dynamic optimization of constrained chemical engineering problems using dynamic programming. Comput Chem Eng 19:513–525
10. DeTremblay M, Luus R (1989) Optimization of non-steady-state operation of reactors. Canad J Chem Eng 67:494–502
11. Hartig F, Keil FJ, Luus R (1995) Comparison of optimization methods for a fed-batch reactor. Hungarian J Ind Chem 23:141–148
12. Lapidus L, Luus R (1967) Optimal control of engineering processes. Blaisdell, Waltham, pp 84–86
13. Li D, Haimes YY (1990) New approach for nonseparable dynamic programming problems. JOTA 66:311–330
14. Luus R (1989) Optimal control by dynamic programming using accessible grid points and region reduction. Hungarian J Ind Chem 17:523–543
15. Luus R (1990) Application of dynamic programming to high-dimensional nonlinear optimal control problems. Internat J Control 52:239–250
16. Luus R (1990) Optimal control by dynamic programming using systematic reduction in grid size. Internat J Control 19:995–1013
17. Luus R (1991) Application of iterative dynamic programming to state constrained optimal control problems. Hungarian J Ind Chem 19:245–254
18. Luus R (1991) Effect of the choice of final time in optimal control of nonlinear systems. Canad J Chem Eng 69: 144–151
19. Luus R (1992) On the application of iterative dynamic programming to singular optimal control problems. IEEE Trans Autom Control 37:1802–1806
20. Luus R (1993) Application of dynamic programming to differential-algebraic process systems. Comput Chem Eng 17:373–377
21. Luus R (1993) Application of iterative dynamic programming to very high-dimensional systems. Hungarian J Ind Chem 21:243–250
22. Luus R (1993) Optimization of fed-batch fermentors by iterative dynamic programming. Biotechnol and Bioengin 41:599–602
23. Luus R (1993) Piecewise linear continuous control by iterative dynamic programming. Ind Eng Chem Res 32:859–865
24. Luus R (1994) Optimal control of batch reactors by iterative dynamic programming. J Process Control 4:218–226
25. Luus R (1995) Sensitivity of control policy on yield of a fed-batch reactor. Proc. IASTED Internat. Conf. on Modelling and Simulation, Pittsburgh, PA, April 27-29, 1995, pp 224–226
26. Luus R (1996) Numerical convergence properties of iterative dynamic programming when applied to high dimensional systems. Chem Eng Res Des 74:55–62
27. Luus R (1996) Use of iterative dynamic programming with variable stage lengths and fixed final time. Hungarian J Ind Chem 24:279–284
28. Luus R (1997) Application of iterative dynamic programming to optimal control of nonseparable problems. Hungarian J Ind Chem 25:293–297
29. Luus R (1997) Use of iterative dynamic programming for optimal singular control problems. Proc. IASTED Internat. Conf. on Control, Cancun, Mexico, May 28-31, 1997, pp 286–289
30. Luus R (1997) Use of variable stage-lengths for constrained optimal control problems. Hungarian J Ind Chem 25: 299–304
31. Luus R (1998) Direct approach to time optimal control by iterative dynamic programming. Proc. IASTED Internat. Conf. on Intelligent Systems and Control, Halifax, Nova Scotia, Canada, June 1-4, 1998, pp 121–125
32. Luus R (1998) Iterative dynamic programming: from curiosity to a practical optimization procedure. Control and Intelligent Systems 26:1–8

33. Luus R (2000) Iterative dynamic programming. Chapman and Hall/CRC, London
34. Luus R, Bojkov B (1994) Global optimization of the bifunctional catalyst problem. Canad J Chem Eng 72:160–163
35. Luus R, Dittrich J, Keil FJ (1992) Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. Canad J Chem Eng 70:780–785
36. Luus R, Galli M (1991) Multiplicity of solutions in using dynamic programming for optimal control. Hungarian J Ind Chem 19:55–62
37. Luus R, Jaakola THI (1973) Optimization by direct search and systematic reduction of the size of search region. AIChE J 19:760–766
38. Luus R, Okongwu ON (1999) Towards practical optimal control of batch reactors. Chem Eng 75:1–9
39. Luus R, Rosen O (1991) Application of iterative dynamic programming to final state constrained optimal control problems. Ind Eng Chem Res 30:1525–1530
40. Luus R, Smith SG (1991) Application of dynamic programming to high-dimensional systems described by difference equations. Chem Eng Techn 14:122–126
41. Luus R, Storey C (1997) Optimal control of final state constrained systems. Proc. IASTED Internat. Conf. on Modelling, Simulation and Control, Singapore, Aug. 11-13, 1997, pp 245–249
42. Luus R, Tassone V (1992) Optimal control of nonseparable problems by iterative dynamic programming. Proc. 42nd Canad. Chemical Engin. Conf., Toronto, Canada, October, 18-21, 1992, pp 81–82
43. Luus R, Zhang X, Hartig F, Keil FJ (1995) Use of piecewise linear continuous control for time-delay systems. Ind Eng Chem Res 34:4136–4139
44. Marroquin G, Luyben WL (1973) Practical control studies of batch reactors using realistic mathematical models. Chem Eng Sci 28:993–1003
45. Mekarapiruk W, Luus R (1997) Optimal control of final state constrained systems. Canad J Chem Eng 25:806–811
46. Mekarapiruk W, Luus R (1997) Optimal control of inequality state constrained systems. Ind Eng Chem Res 36:1686–1694
47. Tassone V, Luus R (1993) Reduction of allowable values for control in iterative dynamic programming. Chem Eng Sci 48:3864–3867

Dynamic Programming: Stochastic Shortest Path Problems

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C40

Article Outline

Keywords

Dynamic programming; Infinite horizon problems; Stochastic shortest path

The *shortest path problem* is considered to be one of the classical and most important combinatorial optimization problems. Given a directed graph and a length α_{ij} for each arc (i, j), the problem is to find a path of minimum length that leads from any node i to a node t, called the destination node. So, for each node i, we need to optimally identify a successor node $u(i)$ so as to reach the destination at the minimum sum of arc lengths over all paths that start at i and terminate at t. Of particular relevance is, in the area of distributed computation, the problem of data routing within a computer communication network. In such a case, the cost associated with a particular link (i, j) is related to an average delay. The *stochastic shortest path problem* is a generalization whereby for each node i we must select a probability distribution over all possible successor nodes j out of a given set of probability distributions $p_{ij}(u)$, parameterized by a control $u \in U(i)$. Clearly, the path traversed and its length are random variables, but the optimal path should lead to the destination with probability 1 and have the minimum expected length. Furthermore, if the probability distributions are such that they assign a probability of 1 to a single successor we then recover the deterministic shortest path problem. Clearly, sequential decisions have to be made optimally so as to determine the sequence of controls that would produce for any current state, i. e. node, a successor state, i. e. node, so as to minimize the expected length for reaching the terminal state. If we were to assume that a particular policy π, i. e., set of control actions, has been selected the total expected cost starting from an initial state i, using this policy would be:

$$J^{\pi}(i) = \lim_{N\to\infty} \mathrm{E}\left\{\sum_{k=0}^{N-1} \alpha^k g(i_k, \mu_k(i), i_{k+1}) \colon\ i_0 = i\right\} .$$

The optimal cost-to-go starting from state i is denoted by

$$J^* = \min_\pi J^\pi(i).$$

This problem is defined as s special case of the *total cost infinite horizon problem*. The need to assume an infinite horizon is not required by the actual description of the problem, but rather it is a necessity since the actual length is random as well as unknown. The following are the key characteristics of the stochastic shortest path problem description within the infinite horizon dynamic programming framework:

1) There is no discounting, $\alpha = 1$.
2) The state space is $S = \{1, \ldots, n, t\}$.
3) The transitions probabilities are:

$$p_{ij}(u) == \mathsf{P}(x_{k+1} = j | x_k = i,\ u_k = u),\\ i, j \in S, \quad u \in U(i).$$

4) The state t is absorbing, that is,

$$p_{tt}(u) = 1, \quad \forall u \in U(t);$$

the state t, the termination state, is special in the sense that reaching it is inevitable.
5) The control set $U(i)$ is finite.
6) The destination is cost-free, i. e.,

$$g(t, u, t) = 0, \quad \forall u \in U(t).$$

If we denote by $g'(i, u, j)$ the cost of moving from i to j using control u, then the expected cost per stage will be defined as:

$$g'(i, u) = \sum_{j=1}^{n} p_{ij}(u) g(i, u, j).$$

The concept of the absorbing state implies that this state will either be reached inevitably, or there is an incentive to reach it with the minimum expected cost. The stochastic shortest path problem was first formulated in [3] while addressing a fundamental problem in control theory, namely finding the input that would take a given system to a specified terminal state at minimum cost. A fundamental assumption regarding the types of stochastic shortest path problems that can be analyzed states that:

Assumption 1 There exists at least one proper policy.

A *proper policy* μ is a *stationary policy* which, when used, results in a positive probability that the destination state will be reached after at most n stages, regardless of the initial state. That is:

$$\rho_\mu = \max_{i=1,\ldots,n} \mathsf{P}\{x_n \neq t \colon\ x_0 = i,\ \mu\} < 1.$$

A stationary policy is a policy of the form $\pi = \{\mu, \mu, \ldots\}$.

For analysis purposes, the following operator for any vector J is defined:

$$(TJ)(i) = \min_{u \in U(i)} \left[g(i, u) + \sum_{j=1}^{n} p_{ij}(u) J(j) \right], \\ i = 1, \ldots, n,$$

which is obtained by applying one iteration of the basic *dynamic programming algorithm* to the cost function J, by realizing that the expectancy operator can be reformulated based on the functional form of the state transition probabilities p_{ij}. It can actually be shown, [1], that T is a *contraction mapping* with respect to a *weighter sup norm*. In other words there exist positive constants $v_1, \ldots, v_n$, and some γ with $0 < \gamma < 1$, such that for all J_1, J_2:

$$\max_{i=1,\ldots,n} \frac{1}{v_i} |(TJ_1)(i) - (TJ_2)(i)| \\ \leq \gamma \max_{i=1,\ldots,n} \frac{1}{v_i} |J_1(i) - J_2(i)| .$$

Furthermore, the operator T is *monotone*, that is: for any vector J and $\bar{J}$ such that $J(i) \leq \bar{J}(i)$, $i = 1, \ldots, n$, and for any stationary policy μ we have:

$$(T^k J)(i) \leq (T^k \bar{J})(i),\\ (T^k_\mu J)(i) \leq (T^k_\mu \bar{J})(i),\\ i = 1, \ldots, n, \quad k = 1, 2, \ldots .$$

The main results of the theoretical analysis of stochastic shortest path problems are analogous to those for *discounted problems*:

i) The optimal cost vector is a solution to *Bellman's equation*: $J^* = TJ^*$.
ii) For every proper policy the cost vector J satisfies:

$$\lim_{k \to \infty} (T^k J)(i) = J^*(i), \quad i = 1, \ldots, n.$$

iii) A stationary policy μ is optimal if and only if $T_\mu J^* = TJ^*$.

iv) For every proper policy μ:

$$\lim_{k\to\infty} T_\mu^k J = J^\mu ,$$
$$J^\mu = T_\mu J^\mu .$$

A thorough analysis of the computational complexity of stochastic shortest path problems has been presented in [5].

In order to address computationally stochastic shortest path problems the general methods, i. e., *value iteration*, and *policy iteration*, as well as approximation schemes along the lines presented in [1] as developed for discounted problems. A detailed account can also be found in [6]. Value iteration is a principal method for calculating optimal cost J^* by generating sequences $T^k J$ starting from some J. Issues related to the *Gauss–Seidel* implementation of the value iteration are discussed in [2]. Although in principle an infinite number of iterations will be required, under certain conditions finite convergence can be achieved. An alternative way is to perform policy iterations, in the sense that starting with a proper policy μ_0, a sequence of policies converging to the optimal one is constructed. According to property iv), for any given policy μ, the cost vector can be evaluate as the solution of a system of linear equations:

$$J(i) = \sum_{j=1}^{n} p_{ij}(\mu_k(i))(g(i,\mu_k(i),j) + J(j)), \qquad i = 1,\dots,n .$$

A policy improvement can be know determined as in:

$$\mu_{k+1}(i) = \arg\min_{u\in U(i)} \sum_{j=0}^{n} p_{ij}(u)(g(i,u,j) + J^{\mu_k}(j)) .$$

These approaches assume that mathematical models for the cost structure and the transition probabilities of the system exist. In may cases however, such information is not available and methods based on simulation have been developed. This information can be derived by simulating, for given control and state spaces, the system's response so as to derive the associated transition costs $g(i,u,j)$. The ideas of *Monte-Carlo simulation* can be utilized so as to use simulation for policy evaluations. A straightforward way of computing the corresponding cost vector J_μ for a given policy μ, is to generate many sample trajectories starting at i, average the corresponding costs, therefore obtaining an estimate for $J_\mu(i)$. An alternative way, is to perform an infinite (large) number of simulation runs from various initial states up to the destination state, and any time that state i is encountered we record the corresponding cost of reaching state t:

$$c(i,m) = g(i,i_1) + g(i_1,i_2) + \dots + g(i_N,t) .$$

By averaging the simulations we obtain:

$$J_\mu(i) = \lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^{M} c(i,m) .$$

The iterative implementation of the update process results in:

$$\begin{aligned} J_\mu(i_k) = J_\mu(i_k) &+ \gamma_k \left(g(i_k,i_{k+1}) + g(i_{k+1},i_{k+2})\right. \\ &\left.+ \dots + g(i_N,t) - J_\mu(i_k)\right) , \\ &k = 1,\dots,N , \\ &\gamma_k = \frac{1}{m} , \quad m = 1,2,\dots \end{aligned}$$

Using simulation to perform the policy evaluation as just described, can be utilized so as to improve on the actual policies in order to achieve optimality. The concept of temporal differences, [7], was proposed recently as an alternative way so as to develop policy iterations, [1,2]. This concept originated in the field of reinforcement learning, and the key premise is to adjust the estimations appropriately so as to modify prior predictions when a temporal difference if observed, by essentially looking back in time and correcting previous predictions. The *temporal difference* is defined as the quantity:

$$d_k = g(i_k,i_{k+1}) + J_{\mu_{k+1}} - J_{\mu_k} , \qquad k = 1,\dots,N .$$

The temporal difference represents the difference between the current estimate $J_\mu(i_k)$ of expected *cost-to-go* to the termination state and the predicted cost-to-go to the termination state $g(i_k, i_k+1)+ J_{\mu_{k+1}}$. The key idea of the Monte-Carlo simulation using temporal differences is to update the individual cost-to-go as soon as the cost to go of the successor has been estimated. In other words $J_\mu(i_1)$ is update as soon as $g(i_1, i_2)$ and i_2 are generated during the simulation runs. Then update both $J_\mu(i_1)$ and $J_\mu(i_2)$ immediately after $g(i_2, i_3)$ and i_3 are generated, etc.

For the cases where the number of stages becomes prohibitively large, approximations schemes can be used so as to derive accurate estimates of either the optimal cost, J^*, or the optimal policy, μ. By approximating the optimal cost, J^*, we need to generate for a given state i and approximation $J'(i, r)$ of $J^*(i)$, where r a parameter vector that is to be determined by using some type of least squares minimization. Once the cost is know, it can then be used so as to generate suboptimal policies as:

$$\mu'(i) = \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + J'(j,r)) \,.$$

The type of the approximation is nonunique but usually the approximations are of the form:

$$J'(i,r) = \sum_{k=1}^{m} r_k w_k(i).$$

In essence the approximation is a linear combination of a set of basis functions.

Recently (1994), [8], presented an approximation scheme referred to as *feature-based aggregation*. The idea is to develop an approximation by making use of the fact that several states may share some common characteristics (features). For a stochastic shortest path problem with n states, one can identify m disjoint subsets S_k, $k = 1, \ldots, m$, such that:

$$S = S_1 \cup \cdots \cup S_m .$$

The basis functions $\omega_k(i)$ can therefore be defined as:

$$\omega_k(i) = \begin{cases} 1 & \text{if } i \in S_k , \\ 0 & \text{if } i \in S_k . \end{cases}$$

The approximate cost can thus be defined as:

$$J'(i,r) = \sum_{k=1}^{m} r_k \omega_k(i).$$

The optimal vector r can be determined as the solution of the aggregate stochastic shortest path problem, for which the aggregate aggregate transition probabilities q_{ki} express the probability of moving from any state in S_k to state i. The vector r solves the corresponding Bellman's equation of the aggregate problem:

$$r_k = \sum_{i=1}^{n} q_{ki} \times \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \sum_{s=1}^{m} r_s \omega_s(j) \right) , \quad k = 1, \ldots, m.$$

For the aggregate problem, the simulation ideas previously developed can be utilized so as to obtain the optimal vector r and therefore obtain the required approximation.

Approximation and simulation schemes can also be combined so as to provide alternatives to performing policy iterations, [1]. For a given stationary policy μ, a number of simulations, M, can be performed so as to obtain the estimates $c(i, m)$. subsequently, a least squares optimization can be solved to provide an approximation to the costs $J_\mu{}'(i, r$, and the coefficients r are derived by solving the following optimization problem:

$$\min_r \sum_{i\in S} \sum_{m=1}^{M} \left| J'(i,r) - c(i,m) \right|^2 .$$

Once the costs function have been determined, and improved policy, $\overline{\mu}(i)$ is identified as:

$$\overline{\mu}(i) = \arg\min_{u\in U(i)} \sum_j p_{ij}(u)(g(i,u,j) + J'(i,r)).$$

In essence, the method iterates between a policy evaluation and a policy improvement step, using both simulation techniques for obtaining, for a given state i and policy μ, sample costs and approximation techniques for obtaining representation of these costs. There are subsequently used so as to estimate improved policies and the iterations continue. The concept of *Q-learning*, [9], was recently proposed as an alternative way of implementing the concept of re-enforcement learning in the solution of dynamic programming, [4].

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Undiscounted Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming
- ▶ Optimization Strategies for Dynamic Systems

References

1. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
2. Bertsekas DP, Tsitsiklis JN (1997) Neuro-dynamic programming. Athena Sci., Belmont, MA
3. Eaton JH, Zadeh LA (1962) Optimal pursuit strategies in discrete state probabilistic systems. Trans ASME Ser DJ Basic Eng 84:23–29
4. Littman ML (1996) Algorithms for sequential decision making. PhD Thesis Brown Univ.
5. Psaraftis HN, Tsitsiklis JN (1993) Dynamic shortest paths in acyclic networks with markovian arc costs. Oper Res 41:91–101
6. Puterman ML (1994) Markov decision processes - Discrete stochastoc dynamic programming. Wiley, New York
7. Sutton RS (1988) Learning to predict by the method of temporal differences. Machine Learning 3:9–44
8. Tsitsiklis JN (1994) Asynchronous stochastic aggregation and Q-learning. Machine Learning 16:185–202
9. Watkins CJ (1989) Learning from delayed rewards. PhD Thesis Cambridge Univ.

Dynamic Programming: Undiscounted Problems

IOANNIS P. ANDROULAKIS
Department of Biomedical Engineering,
Rutgers University, Piscataway, USA

MSC2000: 49L20, 90C40

Article Outline

Keywords

Dynamic programming; Infinite horizon problems; Discounted problem

Total cost *infinite horizon problems* deal with optimal decision making problems in the presence of uncertainty of systems in which events occur sequential. In general, the state transitions are described by a stationary dynamic system of the form:

$$x_{k+1} = f(x_k, u_k, \omega_k), \quad k = 0, 1, \dots,$$

where for each time instance (stage) k, the state of the system is an element of the space S, the control action u that is to be implemented so as to achieve optimality belong to a space C, and finally the uncertainty is modeled through a set of random disturbances ω that belong to a *countable set* D. Furthermore, it is assumed that the control u_k is constrained to take values in a given nonempty set $U(x_k) \in C$, which depends of the current state x_k. The random disturbances ω_k, $k = 0, 1, \dots$, have identical statistics and the probability distributions $\mathcal{P}(\cdot|x_k, u_k)$ are defined on D. These may depend explicitly on x_k and u_k but not on prior disturbances. Given an initial state x_0, we seek a policy π such that $\pi = \{\mu_0, \mu_1, \dots\}$ for which:

$$\begin{aligned} &\mu_k\colon\ S \to C, \\ &\mu_k(x_k) \in U(x_k), \quad \forall x_k \in S, \end{aligned}$$

that minimizes a cost function defined as:

$$J_\pi(x_0) = \lim_{N\to\infty} \mathrm{E}\left\{\sum_{k=1}^{N-1} \alpha^k g(x_k, \mu_k(x_k), \omega_k)\right\}.$$

The function $g()$ is the cost per stage such that: g: $S \times C \times D \to \mathbf{R}$ and is assumed to be given. Finally, the parameter α is termed *discount factor* and it holds that: $0 < \alpha \leq 1$. We denote by Π the set of all admissible policies $\pi = \{\mu_0, \mu_1, \dots\}$, that is, the set of all sequences of such functions for which:

$$\mu_k : S \to C,$$
$$\mu_k(x_k) \in U(x_k), \quad \forall x_k \in S.$$

The optimal cost function J^* is then defined as:

$$J^* = \min_{\pi \in \Pi} J_\pi(x), \quad x \in S.$$

An admissible policy of the form $\pi = \{\mu, \mu, \dots\}$ is termed *stationary* and its corresponding cost is $J\mu$.

Nevertheless, it is often the case that either the discount factor, α, does not have to be less than one, or even the cost per stage does not have to be bounded from either above or bellow. If that is the case, then it is quite possible that for some initial states x_0, the cost functional $J_\pi(x_0)$ may become infinite.

D. Blackwell [4] was among the first to analyze the case in which the discount factor α becomes 1. His approach was based on the idea of studying the behavior as the discount factor approaches 1. Based on the ideas introduced in [7], undiscounted problems are analyzed under either of the following assumptions:

- Positivity assumption:

$$0 \le g(x,u,\omega), \quad \forall (x,u,\omega) \in S \times C \times D;$$

- Negativity assumption:

$$g(x,u,\omega) \le 0, \quad \forall (x,u,\omega) \in S \times C \times D.$$

Having costs per stages being bounded from either above or below may result in the complication of having unbounded costs for some initial states. Therefore, the assumption will be made that ∞, $(-\infty)$, are admissible costs J_π, under the positivity (negativity) assumption. Defining the following two mappings, greatly simplifies the analysis. For any function J defined in S that takes the values $[0, +\infty]$ under the positivity assumption, or the values $[-\infty, 0]$ under the negativity assumption, the mappings T and T_μ are defined as:

$$(TJ)(x) = \min_{u \in U(x)} \mathsf{E}\{g(x,u,\omega) + J(f(x,u,\omega)\}.$$

Furthermore, for any admissible *stationary policy*, the mapping T_μ is defined as:

$$(T_\mu J)(x) = \mathsf{E}\{g(x,\mu(x),\omega) + J(f(x,\mu(x),\omega)\}.$$

Under both the positivity or negativity assumptions, it can be shown, [1], that *Bellman's equation* is satisfied:

- Under either the positivity or the negativity assumption, the optimal cost function, J^* satisfies:

$$J^*(x) = \min_{u \in U(x)} \mathsf{E}\{g(x,u,\omega) + J^*(f(x,u,\omega)\}$$

Clearly, the optimality conditions requires that:

$$J^* = TJ^*.$$

Equivalently, for any stationary policy, it holds true that:

$$J_\mu = T_\mu(J_\mu).$$

It is to be noted though that for undiscounted problems, $\alpha = 1$, the function J^* need not be the unique function minimizes Bellman's equation. In other words, the mapping T does not have a unique fixed point. Nevertheless, the optimal cost vector, J^*, is the smallest fixed point, under the positivity assumption, or the largest fixed point, under the negativity assumption.

- Under the positivity assumption, if $J' : S \to [0, +\infty]$ satisfies $J' = TJ'$, then:

$$J^* \le J'.$$

- Under the negativity assumption, if $J' : S \to [-\infty, 0]$ satisfies $J' = TJ'$, then:

$$J' \le J^*.$$

It should be pointed out, that in the analysis of undiscounted problems the concept of monotonicity plays a key role. The following *monotone convergence theorem* summarizes the key properties, [3]:

Theorem *Let $P = (p_1, p_2, \dots)$ be a probability distribution over $S = \{1, 2, \dots\}$. Let $\{h_N\}$ be a sequence of extended real-valued functions on S such that for all $i \in S$ and $N = 1, 2, \dots$:*

$$0 \le h_N(i) \le h_{N+1}(i).$$

Let $h: S \to [0, \infty]$ be the limit function:

$$h(i) = \lim_{N\to\infty} h_N(i).$$

Then:

$$\lim_{N\to\infty} \sum_{i=1}^{\infty} p_i h_N(i) = \sum_{i=1}^{\infty} p_i \lim_{N\to\infty} h_N(i) = \sum_{i=1}^{\infty} p_i h(i).$$

As examples of undiscounted problems, let us consider two types of problems that define interesting classes that can be cast as undiscounted dynamic programming problems, namely the *optimal stopping* and the *optimal gambling* strategy problem. The former defines a situation in which at each state x of the state space there are two possible actions that are available. One may either decide to stop, by selecting control u_1, and pay a terminal cost $t(x)$, or, select control u_2, pay a cost $c(x)$ and continue the process, with a new state be given by:

$$x_{k+1} = f(x_k, \omega_k).$$

For completeness purposes one also defines a termination state s that is entered once the stopping decision is made. In other words,

$$x_{k+1} = s \quad \text{if } u_k = u_1 \text{ or } x_k = s.$$

If it is further assumed that both the termination and the continuation costs are positive (or negative), then the problem satisfies the positivity (negativity) assumption. The mapping T defined earlier takes now the form:

$$(TJ)(x) = \min\{t(x), c(x) + \mathsf{E}\{J(f(x,\omega))\}\}, \quad \forall x \in S.$$

The objective is to find the optimal stopping policy that minimizes the total expected costs over an infinite number of stages. Insofar regarding external disturbances, ω, it is assumed that they have the same probability distribution for all time instances and depend only on the current state x_k.

The early work [5] details the gambling problem, but was also one of the early works on undiscounted problems. The problem is defined as one in which a player may stake at any time k any amount $u_k \geq 0$ that does not exceed his/her current fortune, x_k. The stake is won back with probability p, and lost with probability $1-p$. The discrete-time state evolution is described by:

$$x_{k+1} = x_k + \omega_k u_k, \quad k = 1, 2, \ldots.$$

The disturbance ω_k is considered to be 1 with probability p, and -1 with probability $1-p$. The gambling is continued until reaching given fortune or loosing the entire initial capital. The problem is to determine that optimal gambling strategy that maximizes the probability of reaching the target fortune. As gambling strategy is defined the specific rule that specifies what the stakes should be at time k. It can be shown that the *bold strategy* is an optimal policy. The bold strategy is defined as:

$$\mu^*(x) = \begin{cases} x, & 0 < x \leq \frac{1}{2}, \\ 1-x, & \frac{1}{2} \leq x < 1. \end{cases}$$

As suggested by the theory of undiscounted problems, the bold strategy is simply an optimal strategy and others can also be derived, [5].

From a computational standpoint it is important to know whether the method of *successive approximations*, i. e., *value iteration*, converges to the optimal cost function. In other words, it is important to know whether the basic dynamic programming algorithm converges. Under either of the two basic assumptions we have:

- Positivity assumption:

$$J_0 \leq T(J_0) \leq \cdots \leq T^k(J_0) \leq \cdots.$$

- Negativity assumption:

$$J_0 \geq T(J_0) \geq \cdots \geq T^k(J_0) \geq \cdots.$$

In either case

$$J_\infty(x) = \lim_{k\to\infty} T^k(J_0)(x), \quad x \in S.$$

In other words, the sequence generated by successive approximation, i. e., by successively applying the mapping T, converges and the limit is well defined, including the values of $+\infty$ and $-\infty$. For the value iteration method though to be valid we also need to have that $J_\infty = J^*$. In order for the above to be true under the positivity assumptions, an additional condition needs to be satisfied, [2]:

- Let the positivity assumption be satisfied and assume that the sets:

$$U_k(x,\lambda) = \Big\{u \in U(x): \mathsf{E}\{g(x,u,\omega) + T^k(J_0)(f(x,u,\omega))\} \leq \lambda\Big\}$$

  are *compact* subsets of a Euclidean space for every $x \in S$, $\lambda \in \mathbf{R}$, and for all k greater than some integer $\bar{k}$. Then:

$$J_\infty = T(J_\infty) = J^*.$$

It should be noted that since $U(x)$ is assumed to be finite, the above condition is satisfied. A detailed account of the value iteration method of undiscounted Markov decision problems can also be found in [6].

It can also be shown, [3], that it is possible to devise computational methods based on mathematical programming when the state, control, and disturbance spaces are finite. Under the negativity assumption, the vector J solves the following linear programming problem:

$$\begin{cases} \max & \sum_{i=1}^{n} \lambda_i \\ \text{s.t.} & \lambda_i \le g(i,u) + \sum_{j=1}^{n} p_{ij}(u)\lambda_j \\ & i = 1, 2, \dots . \end{cases}$$

Under the positivity assumption, the corresponding program takes the form:

$$\begin{cases} \min & \sum_{i=1}^{n} \lambda_i \\ \text{s.t.} & \lambda_i \ge \min_{u \in U(x)} \left\{ g(i,u) + \sum_{j=1}^{n} p_{ij}(u)\lambda_j \right\}, \\ & i = 1, 2, \dots . \end{cases}$$

Unfortunately, this two-level optimization problem is neither linear nor convex, and therefore its solution highly nontrivial.

See also

- ▶ Dynamic Programming: Average Cost Per Stage Problems
- ▶ Dynamic Programming in Clustering
- ▶ Dynamic Programming: Continuous-time Optimal Control
- ▶ Dynamic Programming: Discounted Problems
- ▶ Dynamic Programming: Infinite Horizon Problems, Overview
- ▶ Dynamic Programming: Inventory Control
- ▶ Dynamic Programming and Newton's Method in Unconstrained Optimal Control
- ▶ Dynamic Programming: Optimal Control Applications
- ▶ Dynamic Programming: Stochastic Shortest Path Problems
- ▶ Hamilton–Jacobi–Bellman Equation
- ▶ Multiple Objective Dynamic Programming
- ▶ Neuro-dynamic Programming

References

1. Bertsekas DP (1976) Dynamic programming and stochastic control. Acad. Press, New York
2. Bertsekas DP (1977) Monotone mappings with applications in dynamic programing. SIAM J Control Optim 15 , 438–464
3. Bertsekas DP (1995) Dynamic programming and optimal control. Athena Sci., Belmont, MA
4. Blackwell D (1962) Discrete dynamic programming. Ann Math Statist 33:719–726
5. Dubins L, Savage LM (1966) How to gabmble if you must. McGraw-Hill, New York
6. Schweitzer PJ, Federgruen A (1977) The asymptotic behavior of undiscounted value iteration in Markov decision problems. Math Oper Res 2:360–381
7. Strauch R (1966) Negative dynamic programming. Ann Math Statist 37:871–890

Dynamic Traffic Networks

ANNA NAGURNEY
University Massachusetts, Amherst, USA

MSC2000: 90B15

Article Outline

Keywords

Projected dynamical system; Trip-route choice adjustment process; Day-to-day dynamic travel behavior; System stability; Asymptotical system stability; Discrete-time algorithms; Regular link cost function

Congested urban transportation networks represent complex systems in which travelers interact so as to determine unilaterally their cost-minimizing routes of travel between their points of origin and their destinations. The governing concept here is that of 'user-optimization', which dates to J.G. Wardrop [19] (and was so termed by S.C. Dafermos and F.T. Sparrow [4]), and states that, in equilibrium, all used paths connecting an origin/destination pair of nodes will have travel costs that are equal and minimal.

The complexity of user-optimized transportation networks, sometimes also referred to as the 'traffic assignment' problem, has stimulated much research in the past several decades, both from methodological perspectives, as well as in terms of practical application. Notable developments include: the proof by M.J. Beckmann, C.B. McGuire, and C.B. Winsten [1] that, under certain symmetry assumptions on the link travel cost functions (and travel disutility functions), the traffic network equilibrium solution also satisfies the Kuhn–Tucker conditions of an appropriately constructed optimization problem, and the identification by Dafermos [2] that the traffic network equilibrium conditions, as formulated by M.J. Smith [17] (without any imposition of a symmetry assumption), satisfy a variational inequality problem. Books that discuss methodological approaches to static traffic equilibrium problems include [9,14,16] (see also [6]).

The study of dynamic travel route choice models on general transportation networks, where time is explicitly incorporated into the framework, was initiated by D.K. Merchant and G.L. Nemhauser [8], who focused on dynamic system-optimal networks with the characteristic of many origins and a single destination. In system-optimal networks, in contrast to user-optimal networks, one seeks to determine the path flow and link load patterns that minimize the total cost in the network, rather than the individual path travel costs.

M.J. Smith [18], in turn, proposed a dynamic traffic user-optimized model with fixed demands. H. Mahmassani [7] also proposed dynamic traffic models and investigated them experimentally. The recent book [15] provides an overview of the history of dynamic traffic network models and discusses distinct approaches for their analysis and computation.

Here we present a dynamic traffic model with elastic demands proposed by P. Dupuis and A. Nagurney [5], who, along with D. Zhang and Nagurney [22], established the foundations for a new methodology, that of 'projected dynamical systems' theory. The notable feature of a projected dynamical system is that its set of stationary points coincides with the set of solutions of the corresponding variational inequality problem. There, thus, exists a fundamental linkage between the static world of finite-dimensional variational inequality problems and the dynamic world exhibited by a new class of dynamical system.

The dynamic adjustment process that is presented here models the travelers' *day-to-day* dynamic behavior of making trip decisions and route choices associated with a travel disutility perspective. Subsequently, some of the stability results of this travel-route choice adjustment process obtained by Zhang and Nagurney [21] are reviewed, which address whether and how the travelers' dynamic behavior in attempting to avoid congestion leads to a traffic equilibrium pattern. Finally, we recall the discrete-time algorithms devised for the computation of traffic network equilibria with elastic demands and with known travel disutility functions. The convergence of these discrete-time algorithms was established by Zhang and Nagurney [10,13]. Additional dynamic traffic network models, as well as qualitative and numerical results, using this methodology can be found in [11,12], and [22]. For alternative dynamic traffic network models and approaches, see [15], and the references therein.

A Dynamic Traffic Network Model

The model that we present is due to Dupuis and Nagurney [5]. It is a dynamic counterpart to the static traffic network equilibrium model with elastic travel demands developed by Dafermos [3].

We consider a network $[N, L]$ consisting of nodes $[N]$ and directed links $[L]$. Let a denote a link of the network connecting a pair of nodes, and let p denote a path (assumed to be acyclic) consisting of a sequence of links connecting an origin/destination (O/D) pair w. P_w denotes the set of paths connecting the O/D pair w with n_{P_w} paths. We let W denote the set of O/D pairs and P the set of paths in the network.

Let x_p represent the flow on path p and let f_a denote the load on link a. The following conservation of flow equation must hold for each link a:

$$f_a = \sum_p x_p \delta_{ap} ,$$

where $\delta_{ap} = 1$, if link a is contained in path p, and 0 otherwise. The expression states that the load on a link a is equal to the sum of all the path flows on paths that contain the link a.

Moreover, if we let d_w denote the demand associated with an O/D pair w, then we must have that for each O/D pair w

$$d_w = \sum_{p \in P_w} x_p,$$

where $x_p \geq 0$, for all p, that is, the sum of all the path flows on paths connecting the O/D pair w must be equal to the demand d_w. Let x denote the column vector of path flows with dimension n_P.

Let c_a denote the user cost associated with traversing link a, and let C_p denote the user cost associated with traversing path p. Then

$$C_p = \sum_a c_a \delta_{ap}.$$

In other words, the cost of a path is equal to the sum of the costs on the links comprising that path. We group the link costs into the column vector c with n_A components, and the path costs into the column vector C with n_P components. We also assume that we are given a travel disutility function λ_w for each O/D pair w. We group the travel disutilities into the column vector λ with J components.

We assume that, in general, the cost associated with a link may depend upon the entire link load pattern, that is,

$$c_a = c_a(f)$$

and that the travel disutility associated with an O/D pair may depend upon the entire demand pattern, that is,

$$\lambda_w = \lambda_w(d),$$

where f is the n_A-dimensional column vector of link loads and d is the J-dimensional column vector of travel demands.

We now, for completeness, recall the traffic network equilibrium conditions.

Definition 1 (***traffic network equilibrium***, [1,3]) A vector $x^* \in \mathbf{R}_+^{np}$, which induces a vector d^* through the demand equations, is a traffic network equilibrium if for each path $p \in P_w$ and every O/D pair w:

$$C_p(x^*) \begin{cases} = \lambda_w(d^*), & \text{if} \quad x_p^* > 0 \\ \geq \lambda_w(d^*), & \text{if} \quad x_p^* = 0. \end{cases}$$

In equilibrium, only those paths connecting an O/D pair that have minimal user costs are used, and their costs are equal to the travel disutility associated with traveling between the O/D pair.

The equilibrium conditions have been formulated as a variational inequality problem by Dafermos (cf. [2,3]). In particular, we have:

Theorem 2 [3] *$(x^*, d^*) \in K^1$ is a traffic network equilibrium pattern, that is, satisfies the equilibrium conditions if and only if it satisfies the variational inequality problem (path flow formulation):*

$$\left\langle C(x^*)^\top, x - x^* \right\rangle - \left\langle \lambda(d^*)^\top, d - d^* \right\rangle \geq 0, \qquad \forall (x, d) \in K^1,$$

where $K^1 \equiv \{(x, d) : x \geq 0$, and the demand constraints hold\}.

Note that, in view of the demand constraints, one may define $\widehat{\lambda}(x) \equiv \lambda(d)$, in which case one may rewrite the variational inequality in the path flow variables x only, that is, we seek to determine $x^* \in \mathbf{R}_+^{np}$, such that

$$\left\langle (C(x^*) - \overline{\lambda}(x^*))^\top, x - x^* \right\rangle \geq 0, \qquad \forall x \in \mathbb{R}_+^{n_P},$$

where $\overline{\lambda}(x)$ is the $n_{P_{w_1}} \times \dots n_{P_{w_J}}$-dimensional column vector with components:

$$(\widehat{\lambda}_{w_1}(x), \dots, \widehat{\lambda}_{w_1}(x), \dots, \widehat{\lambda}_{w_J}(x), \dots, \widehat{\lambda}_{w_J}(x)),$$

where J is the number of O/D pairs. If we now let $F(x) \equiv (C(x) - \overline{\lambda}(x))$ and $K \equiv \{x : x \in \mathbf{R}_+^{np}\}$, then, clearly, this inequality can be placed into standard variational inequality form.

The Trip-Route Choice Adjustment Process

The dynamical system, first presented in [5], whose stationary points correspond to solutions of the latter variational inequality problem above, is given by:

$$\dot{x} = \Pi_K(x, \overline{\lambda}(x) - C(x)), \qquad x(0) = x_0 \in K,$$

where, assuming that the feasible set K is a convex polyhedron (as is the case here), and given $x \in K$ and $v \in \mathbf{R}^n$, we define the projection of the vector v at x (with respect to K) by

$$\Pi_K(x, v) = \lim_{\delta \to 0} \frac{P_K(x + \delta v) - x}{\delta},$$

where P_K is defined as:

$$P_K(x) = \arg\min_{z \in K} \|x - z\|,$$

and $\|\cdot\|$ denotes the Euclidean norm.

This dynamical system is a *projected dynamical system* (cf. [10,22]), since the right-hand side, which is a projection operator, is discontinuous.

The adjustment process interpretation of the dynamical system, as discussed in [5], is as follows: Users of a transportation network choose, at the greatest rate, those paths whose differences between the travel disutilities (demand prices) and path costs are maximal; in other words, those paths whose costs are minimal relative to the travel disutilities. If the travel cost on a path exceeds the travel disutility associated with the O/D pair, then the flow on that path will decrease; if the travel disutility exceeds the cost on a path, then the flow on that path will increase. If the difference between the travel disutility and the path cost drives the path flow to be negative, then the projection operator guarantees that the path flow will be zero. The process continues until there is no change in path flows, that is, until all used paths have path costs equal to the travel disutilities, whereas unused paths will have costs which exceed the disutilities. Specifically, the travelers adjust their route choices until an equilibrium is reached.

The following example, given in a certain discrete-time realization, shows how the dynamic mechanism of the above trip-route choice adjustment would reallocate the traffic flow among the paths and would react to changes in the travel disutilities.

Example 3 Consider a simple transportation network consisting of two nodes, with a single O/D pair w, and two links a and b representing the two disjoint paths connecting the O/D pair. Suppose that the link costs are:

$$c_a(f_a) = f_a + 2, \qquad c_b(f_b) = 2f_b,$$

and the travel disutility function is given by:

$$\lambda_w(d_w) = -d_w + 5.$$

Note that here a path consists of a single link and, hence, we can use x and f interchangeably. Suppose that, at time $t = 0$, the flow on link a is 0.7, the flow on link b is 1.5; hence, the demand is 2.2, and the travel disutility is 2.8, that is,

$$x_a(0) = 0.7, \qquad x_b(0) = 1.5,$$
$$d_w(0) = 2.2, \qquad \lambda_w(0) = 2.8,$$

which yields travel costs: $c_a(0) = 2.7$ and $c_b(0) = 3.0$.

According to the above trip-route choice adjustment process, the flow changing rates at time $t = 0$ are:

$$\dot{x}_a(0) = \lambda_w(0) - c_a(0) = 0.1,$$
$$\dot{x}_b(0) = \lambda_w(0) - c_b(0) = -0.2.$$

If a time increment of 0.5 is used, then at the next moment $t = 0.5$, the flows on link a and link b are:

$$x_a(0.5) = x_a(0) + 0.5\dot{x}_a(0)$$
$$= 0.7 + 0.5 \times 0.1 = 0.75,$$
$$x_b(0.5) = x_b(0) + 0.5\dot{x}_b(0)$$
$$= 1.5 - 0.5 \times 0.2 = 1.4,$$

which yields travel costs: $c_a(0.5) = 2.75$ and $c_b(0.5) = 2.8$, a travel demand $d_w(0.5) = 2.15$, and a travel disutility $\lambda_w(0.5) = 2.85$. Now, the flow changing rates are given by:

$$\dot{x}_a(0.5) = \lambda_w(0.5) - c_a(0.5)$$
$$= 2.85 - 2.75 = 0.1,$$
$$\dot{x}_b(0.5) = \lambda_w(0.5) - c_b(0.5)$$
$$= 2.85 - 2.8 = 0.05.$$

The flows on link a and link b at time $t = 1.0$ would, hence, then be:

$$x_a(1.0) = x_a(0.5) + 0.5\dot{x}_a(0.5)$$
$$= 0.75 + 0.5 \times 0.1 = 0.80,$$
$$x_b(1.0) = x_b(0.5) + 0.5\dot{x}_b(0.5)$$
$$= 1.4 + 0.5 \times 0.05 = 1.425,$$

which yields travel costs: $c_a(1.0) = 2.80$ and $c_b(1.0) = 2.85$, a travel demand $d_w(1.0) = 2.225$, and a travel disutility $\lambda_w(1.0) = 2.775$. Now, the flow changing rates are given by:

$$\begin{aligned}\dot{x}_a(1.0) &= \lambda_w(1.0) - c_a(1.0)\\ &= 2.775 - 2.800 = 0.025,\\ \dot{x}_b(1.0) &= \lambda_w(1.0) - c_b(1.0)\\ &= 2.775 - 2.850 = -0.075.\end{aligned}$$

The flows on link a and link b at time $t = 1.5$ would be:

$$\begin{aligned}x_a(1.5) &= x_a(1.0) + 0.5\dot{x}_a(1.0)\\ &= 0.8 - 0.5 \times 0.025 = 0.7875,\\ x_b(1.5) &= x_b(1.0) + 0.5\dot{x}_b(1.0)\\ &= 1.425 - 0.5 \times 0.075 = 1.3875,\end{aligned}$$

which yields travel costs: $c_a(1.5) = 2.7875$ and $c_b(1.5) = 2.775$, a travel demand $d_w(1.5) = 2.175$, and a travel disutility $\lambda_w(1.0) = 2.82$.

In this example, hence, as time elapses, the trip-route choice adjustment process adjusts the flow volume on the two links so that the difference between the travel costs of link a and link b is being reduced, from 0.3, to 0.05, and, finally, to 0.0125; and, the difference between the disutility and the travel costs on the used links is also being reduced from 0.2, to 0.1, and to 0.045. In fact, the traffic equilibrium with: $x_a^* = 0.8$ and $x_b^* = 1.4$, which induces the demand $d_w^* = 2.2$, is almost attained in only 1.5 time units.

Stability Analysis

We now present the stability results of the trip route choice adjustment process. The results described herein are due to Zhang and Nagurney [21]. For example, the questions that motivate transportation planners and analysts to study the stability of a transportation system include: Will any initial flow pattern be driven to an equilibrium by the adjustment process? In addition, will a flow pattern near an equilibrium always stay close to it? These concerns of *system stability* are important in traffic assignment and form, indeed, a critical base for the very concept of an equilibrium flow pattern.

For the specific application of transportation network problems, the following definitions of stability of the transportation system and the local stability of an equilibrium are adapted from the general stability concepts of projected dynamical systems (cf. [22]).

Definition 4 (*stability at an equilibrium*) An equilibrium flow pattern x^* is stable if it is a global monotone attractor for the corresponding route choice adjustment process.

Definition 5 (*asymptotical stability at an equilibrium*) An equilibrium flow pattern x^* is asymptotically stable if it is a strictly global monotone attractor for the corresponding route choice adjustment process.

Definition 6 (*stability of the system*) A route choice adjustment process is stable if all its equilibrium flow patterns are stable.

Definition 7 (*asymptotical stability of the system*) A route choice adjustment process is asymptotically stable if all its equilibrium flow patterns are asymptotically stable.

We now present the stability results in [21] for the trip-route choice adjustment process.

Theorem 8 ([21]) *Suppose that the link cost functions c are monotone increasing in the link load pattern f and that the travel disutility functions λ are monotone decreasing in the travel demand d. Then the trip-route choice adjustment process is stable.*

Theorem 9 ([21]) *Assume that there exists some equilibrium path flow pattern. Suppose that the link cost functions c and negative disutility functions $-\lambda$ are strictly monotone in the link load f and the travel demand d, respectively. Then the trip-route choice adjustment process is asymptotically stable.*

The first theorem states that, provided that monotonicity of the link cost functions and the travel disutility functions holds true, then any flow pattern near an equilibrium will stay close to it forever. Under the strict monotonicity assumption, on the other hand, the second theorem can be interpreted as saying that any initial flow pattern will eventually be driven to an equilibrium by the route choice adjustment process.

Discrete Time Algorithms

The Euler method and the Heun method were employed in [13] and [10] for the computation of solutions to dynamic elastic demand traffic network problems with known travel disutility functions, and their convergence was also established therein. We refer the reader to these references for numerical results, including traffic network examples that are solved on a massively parallel computer architecture.

In particular, at iteration τ, the *Euler method* computes

$$x^{\tau+1} = P_K(x^\tau - a_\tau F(x^\tau)),$$

whereas, according to the *Heun method*, at iteration τ one computes

$$x^{\tau+1} = P_K\left(x^\tau - a_\tau \frac{1}{2}\big[F(x^\tau) + F(P(x^\tau - a_\tau F(x^\tau)))\big]\right).$$

In the case that the sequence $\{a_\tau\}$ in the Euler method is fixed, say, $\{a_\tau\} = \rho$, for all iterations τ, then the Euler method collapses to a projection method (cf. [2,6,9], and [14]).

In the context of the dynamic traffic network problem with known travel disutility functions, the projection operation in the above discrete-time algorithms can be evaluated explicitly and in closed form. Indeed, each iteration τ of Euler method takes the form: For each path $p \in P$ in the transportation network, compute the path flow $x_p^{\tau+1}$ according to:

$$x_p^{\tau+1} = \max\{0, x_p^\tau + a_\tau(\lambda_w(d^\tau) - C_p(x^\tau))\}.$$

Each iteration of the Heun method, in turn, consists of two steps. First, at iteration τ one computes the approximate path flows:

$$\overline{x}_p^\tau = \max\{0, x_p^\tau + a_\tau(\lambda_w(d^\tau) - C_p(x^\tau))\}, \quad \forall p \in P,$$

and updates the approximate travel demands:

$$\overline{d}_w^\tau = \sum_{p \in P_w} \overline{x}_p^\tau, \qquad \forall w \in W.$$

Let

$$\overline{x}^\tau = \{\overline{x}_p^\tau, p \in P\}$$

and

$$\overline{d}^\tau = \{\overline{d}_w^\tau, w \in W\}.$$

Then, for each path $p \in P$ in the transportation network one computes the updated path flows $x_p^{\tau+1}$ according to:

$$x_p^{\tau+1} = \max\Big\{0, x_p^\tau + \frac{a_\tau}{2}[\lambda_w(d^\tau) - C_p(x^\tau) + \lambda_w(\overline{d}^\tau) - C_p(\overline{x}^\tau)]\Big\}, \quad \forall p \in P,$$

and updates the travel demands $d_w^{\tau+1}$ according to:

$$d_w^{\tau+1} = \sum_{p \in P_w} x_p^{\tau+1}, \qquad \forall w \in W.$$

It is worth noting that both the Euler method and the Heun method at each iteration yield subproblems in the path flow variables, each of which can be solved not only in closed form, but also, simultaneously. Hence, these algorithms in the context of this model can be interpreted as massively parallel algorithms and can be implemented on massively parallel architectures. Indeed, this has been done so by Nagurney and Zhang [13] (see also [11] for the case where the demand functions are given, rather than the travel disutility functions).

In order to establish the convergence of the Euler method and the Heun method, one regularizes the link cost structures.

Definition 10 (*regular cost function*) The link cost function c is called regular if, for every link $a \in L$,

$$c_a(f) \to \infty, \qquad \text{as} \quad f_a \to \infty,$$

holds uniformly true for all link flow patterns.

We note that the above regularity condition on the link cost functions is natural from a practical point of view and it does not impose any substantial restrictions. In reality, any link has an upper bound in the form of a capacity. Therefore, letting $f_a \to \infty$ is an artificial device under which one can reasonably deduce that $c_a(f) \to \infty$, due to the congestion effect. Consequently, any practical link cost structure can be theoretically extended to a regular link cost structure to allow for an infinite load.

The theorem below shows that both the Euler method and the Heun method converge to the traffic network equilibrium under reasonable assumptions.

Theorem 11 ([10,13]) *Suppose that the link cost function c is regular and strictly monotone increasing, and that the travel disutility function λ is strictly monotone decreasing. Let* $\{a_\tau\}$ *be a sequence of positive real numbers that satisfies*

$$\lim_{\tau \to \infty} a_\tau = 0$$

and

$$\sum_{\tau=0}^{\infty} a_\tau = \infty .$$

Then both the Euler method and the Heun method produce sequences $\{x^\tau\}$ *that converge to some traffic network equilibrium path flow pattern.*

See also

- ▶ Auction Algorithms
- ▶ Communication Network Assignment Problem
- ▶ Cost Approximation Algorithms
- ▶ Directed Tree Networks
- ▶ Equilibrium Networks
- ▶ Evacuation Networks
- ▶ Generalized Networks
- ▶ Maximum Flow Problem
- ▶ Minimum Cost Flow Problem
- ▶ Network Design Problems
- ▶ Network Location: Covering Problems
- ▶ Nonconvex Network Flow Problems
- ▶ Piecewise Linear Network Flow Problems
- ▶ Shortest Path Tree Algorithms
- ▶ Steiner Tree Problems
- ▶ Stochastic Network Problems: Massively Parallel Solution
- ▶ Survivable Networks
- ▶ Traffic Network Equilibrium

References

1. Beckmann MJ, McGuire CB, Winsten CB (1956) Studies in the economics of transportation. Yale Univ. Press, New Haven, CT
2. Dafermos S (1980) Traffic equilibrium and variational inequalities. Transport Sci 14:42–54
3. Dafermos S (1982) The general multimodal traffic equilibrium problem with elastic demand. Networks 12:57–72
4. Dafermos SC, Sparrow FT (1969) The traffic assignment problem for a general network. J Res Nat Bureau Standards 73B:91–118
5. Dupuis P, Nagurney A (1993) Dynamical systems and variational inequalities. Ann Oper Res 44:9–42
6. Florian M, Hearn D (1995) Network equilibrium models and algorithms. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL (eds) Network Routing. Handbook Oper Res and Management Sci. Elsevier, Amsterdam, pp 485–550
7. Mahmassani H (1991) Dynamic models of commuter behavior: Experimental investigation and application to the analysis of planned traffic disruptions. Transport Res 24A:465–484
8. Merchant DK, Nemhauser GL (1978) A model and an algorithm for the dynamic traffic assignment problems. Transport Sci 12:183–199
9. Nagurney A (1999) Network economics: A variational inequality approach, 2nd edn. Kluwer, Dordrecht
10. Nagurney A, Zhang D (1996) Projected dynamical systems and variational inequalities with applications. Kluwer, Dordrecht
11. Nagurney A, Zhang D (1997) Massively parallel computation of dynamic traffic networks modeled as projected dynamical systems. In: Pardalos PM, Hearn DW, Hager WW (eds) Network Optimization. Lecture Notes Economics and Math Systems. Springer, Berlin, pp 374–396
12. Nagurney A, Zhang D (1997) Projected dynamical systems in the formulation, stability analysis, and computation of fixed demand traffic network equilibria. Transport Sci 31:147–158
13. Nagurney A, Zhang D (1998) Massively parallel implementation of a discrete time algorithm for the computation of dynamic elastic demand traffic problems modeled as projected dynamical systems. J Econom Dynam Control 22:1467–1485
14. Patriksson M (1994) The traffic assignment problem. VSP, Utrecht
15. Ran B, Boyce DE (1996) Modeling dynamic transportation network, 2nd revised edn. Springer, Berlin
16. Sheffi Y (1985) Urban transportation networks. Prentice-Hall, Englewood Cliffs, NJ
17. Smith MJ (1979) Existence, uniqueness, and stability of traffic equilibria. Transport Res 13B:295–304
18. Smith MJ (1984) The stability of a dynamic model of traffic assignment-An application of a method of Lyapunov. Transport Sci 18:245–252
19. Wardrop JG (1982) Some theoretical aspects of road traffic research. Proc Inst Civil Engineers II:325–278
20. Zhang D, Nagurney A (1995) On the stability of projected dynamical systems. J Optim Th Appl 85:97–124
21. Zhang D, Nagurney A (1996) On the local and global stability of a travel route choice adjustment process. Transport Res 30B:245–262
22. Zhang D, Nagurney A (1997) Formulation, stability, and computation of traffic network equilibria as projected dynamical systems. J Optim Th Appl 93:417–444