Advanced Numerical Methods Based on Optimization

Marian Gaiceanu, Vasile Solcanu, Theodora Gaiceanu, and Iulian Ghenea

Abstract In this chapter the unconstrained and constrained optimization algorithms for numerical methods are envisaged. The numerical solutions to the fundamental problems in energy systems are provided. The recent heuristic methods used in power systems are highlighted, and the specific algorithms are proven by simulations. The selection of the best solution is one of the authors' concern. Therefore, the optimization algorithms along with numerical examples are delivered in this chapter.

Keywords Numerical methods · Newton · Newton-Raphson · Optimization algorithms · Trust region · Genetic · Mixed integer nonlinear programming · Heuristic method · Ant colony · PSO · Firefly · Simulated annealing · Differential evolution · Bee swarm · Unit-commitment · Very large scale systems

M. Gaiceanu (✉)
Department of Automatic Control and Electrical Engineering, University Dunarea de Jos Galati, Galati, Romania
e-mail: marian.gaiceanu@ugal.ro

V. Solcanu
University Dunarea de Jos Galati, Galati, Romania
e-mail: vasilesolcanu@dedeman.ro

T. Gaiceanu
Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi, Iasi, Romania
e-mail: gaiceanu.theodora@ac.tuiasi.ro

I. Ghenea
Doctoral School of Fundamental and Engineering Sciences, University Dunarea de Jos of Galati, Galati, Romania
e-mail: iulian.ghenea@ugal.ro

N. Mahdavi Tabatabaei and N. Bizon (eds.), *Numerical Methods for Energy Applications*, Power Systems,
https://doi.org/10.1007/978-3-030-62191-9_8

Nomenclature

BFGS	Broyhen-Fletcher-Goldfarb-Shanno
CD	Conjugate Directions
CNO	Constrained Nonlinear Optimization
DFP	Davidon-Fletcher-Powell
FR	Fletcher–Reeves
GA	Genetic Algorithms
HA	Heuristic Algorithms
LQR	Linear Quadratic Regulator
MINLP	Mixed Integer Nonlinear Programming
MMCG	Modified method of conjugate gradients
NM	Newton method
NO	Nonlinear Optimization
NRM	Newton-Raphson method
PR	Polak–Ribiere
PS	Power Systems
PSO	Particle Swarm Optimization
OC	Optimal Control
TSP	Travelling salesman problem
TR	Trust region
SA	Simulated Annealing
SCUC	Security-Constrained Unit CommitmentUnit Commitment
SIP	Semi-infinite programming
SDP	Semi-defined programming
UC	Unit Commitment
UNO	Unconstrained Nonlinear Optimization
VC	Variational Calculus

Symbols

C	convex set
d_k	descent direction
Δ	distance
F	convex function
$f(x)$	real function of variable x
$f'(x)$	first derivative
$grad(f(x^*))$	gradient of the multivariable function
$\nabla f(x^*, y^*)$	the gradient of function f at the minimum point (x^*, y^*)
$\nabla^2 f(x_k) = D2f_k$	Hessian matrix
$\mathbf{K}$ $(\mathbf{x}, t)$	integral cost
$\mathbf{K}$ $(\mathbf{x}_f, tf)$	final cost

$\mathbf{J}(\mathrm{u})$	performance index
$L(x, u, \tau)$	Lagangian
p	Lagrange multiplier
p_i^C	Cauchy point
$\mathbf{x}$	unknown vector
$s^*(\Delta)$	steepest descent direction
$\|s\| = \Delta$	norm
span	linear combination operator

1 Introduction

Optimization offers the best solution, out of the many possible solutions to a problem, according to a performance criterion imposed, respecting or not certain constraints. In the optimization problems, the performance criterion is chosen according to the proposed objectives, observing certain constraints or not. Multivariable functions are used in variational calculus (VC). VC deals with the functionals, the solution being obtained more often by minimizing the functional. By taking into account the Sect. 1.1 of this book, the evolution of the VC is optimal control (OC). OC is a result of the functional cost minimization, the control being dependent by the state feeback product of the system, the cost function, and a weighting matrix.

The problem of determining the shortest possible route between two points is the oldest optimization problem, the well-known solution being a straight line segment.

Greek mathematicians Zenodorus and Poppus studied the problem of Princess Dido (Elisa) from Tiria or the isoperimetric problem, inspired by the historical story of Vergilius (70-19 BC), Eneida, about the formation of Carthage (850 BC): to find a plane-generated curve of a given straight segment, covering the largest possible area. The Greeks knew the solution (the circle), but it was only in the 29th century rigorously demonstrated.

Heron of Alexandria gave another interpretation of the shortest path possible, in the paper Katoprika (Principles of optics), noting that if a light source emits a light beam reflected in a mirror; it will follow the shortest possible path from source to observer.

The origins of variational calculus date from 1662 with the Fermat principle the fastest path of a light beam which passes through a single optical environment is that of minimum time.

The first minimum time problem formulated and demonstrated was in 1697 ("brachystochrone problem"–"βραρχιστοζ-Short, χρονο-time"), by Johann Bernoulli. In this way the optimal command appeared.

Modern optimal control

The practical point ov view of OC was discovered by Riccati (1676–1754). Jacopo Riccati had obtained a scalar solution of the nowadays linear quadratic problem.

Kalman (1960), Athans and Falb (1966), push forward the development of the modern optimal control.

Dynamic programming and the principle of maximum

Dynamic programming orginated from Bellman (1950s), takes into account the nonlinear optimal command. The main disadvantage of its solution is the large memory requiremnts. The principle of maximum, developed by Pontryagin and the Soviet school, is a natural consequence of Weierstrass's necessary condition with limited command functions. To determine the shortest route, the optimal control maximizes the Hamiltonian within the boundary of definition. The maximum principle is normally found in dynamic programming.

The *nonlinear problem* of determining optimal trajectories was addressed in 1919, by Robert H. Goddard (1882–1945). Using variational calculus, in 1927 Hamel formulates the nonlinear problem, the analytical solution being found by Tsien and Evans in 1951.

The modern optimal control deliver a more robust solution (in frequency domain) through $H\infty$ and H_2 approaches. The robustness of the modern control is related to the systems with structural uncertains, and unmodelled dynamics.

Research in the field of optimization is on-going by using artificial intelligence, Genetic algorithms and metaheuristics approaches.

The objective of this chapter is to provide a methodology to solve the specific problems in power system applications. Moreover, due to the high importance of energy production, the solutions should be the best one. Therefore, the optimization algorithms are also taken into account. In order to facilitate the understanding of some algorithms, the authors provide numerical exmaples.

In this chapter the formulation of the optimization problems are presented. The Sect. 3 contains the classification of the optimal problems. Different types of the optimizations are included in Sect. 4. Unconstrained and constrained optimization algorithms for numerical methods are provided in Sects. 5–13. The recent heuristic methods and the specific algorithms are shown in Sect. 14. The Sect. 15 provides an overview of the genetic algorithms. In Sects. 16–19, different nature inspired optimization algorithms (metaheuristics) are found: ant colony optimization, Simulated Annealing, Particle Swarm Optimization, Bee Swarm, and Firefly model. The last Sects. (20, 21) are dedicated to very large scale neighborhood search, and security-constrained unit commitment. The chapter ends with the conclusions.

2 Formulation of the Optimization Problem

In order to formulate a problem of optimization of a dynamic system, the following steps must be taken into account: (1) the initial and final conditions; (2) the dynamics of the system; (3) the control limits; (3) the objectives of the problem; (4) performance criterion. The optimal solution will be an admissible control delivered by minimization of the peformance criterion [1].

Implementation of the optimal solution

With the analytical or numerical determination of the optimal solution, the natural problem of implementing this solution found for the considered dynamic system is posed.

In the references [1, 2] the optimal command was obtained either by solving a Riccati differential matrix equation (for problems with fixed final time), or from an Riccati algebraic matrix equation (for problems with infinite horizon or final free time). Riccati equations require a reverse integration in time (from the final step to the initial step) and the storage of specific coefficients on this interval, following the application of the optimal command in the direct sense of time (from the initial step to the final step). This procedure requires a large volume of calculus and is sensitive to the variation of the dynamic system parameters. The solutions that eliminate the mentioned disadvantages are presented in the technical papers [1, 2] are described modern methods of implementing the solutions within modern electric drive systems.

Types of constraints

Constraints are applying to control and states of the dynamic system under consideration. At the level of the relations, they must be within an allowable range (*inequality constraints*), and at the state level, as a result of the existence of differential equations that must be solved, their initial and final values (*equality* constraints) must be known. Another variable to consider is *time*. Therefore, *the initial and final time* are all *equality* constraints.

Performance index

The performance index (cost function) is designed by the designer to meet the objectives set. It usually includes a *final cost* and an *integral cost*, corresponding to the stationary and dynamic regimes. From a systemic point of view, the performance index contains system errors or states, and commands. For a system of order n (n states) with m inputs ($m \leq n$), the performance index is defined by the final cost K ($\mathbf{x}$, t) and the integral cost:

$$J(\mathbf{x_0}, t_0, \mathbf{u}, \mathbf{x}, t) = K(\mathbf{x}, t) + \int_{t_0}^{t} L(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau \tag{1}$$

or by knowing the final time, t_f, it becomes:

$$J(\mathbf{x_0}, t_0, \mathbf{u}) = J\left(\mathbf{x_0}, t_0, \mathbf{u}, \mathbf{x}_f, t_f\right) = K\left(\mathbf{x}_f, t_f\right) + \int_{t_0}^{t_f} L(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau, \quad (2)$$

where

- $J(\mathbf{x_0}, t_0, \mathbf{u})$ is the value of the system performance index for the command $\mathbf{u}$ applied,
- t_f the final time,
- $\mathbf{x}_f$ final state,
- K (x$_{f,}$ *t f) terminal cost*.

The performance index can stretch to infinity if no command exists to bring the system to a specified end state:

$$J(\mathbf{x_0}, t_0, \mathbf{u}) \to \infty. \quad (3)$$

It is noted with:

$$J(\mathbf{u}) = J(\mathbf{x_0}, t_0, \mathbf{u}) \quad (4)$$

if the initial state $\mathbf{x_0}$ is known at time $t_{0.}$

Solving the optimal control problem leads to the determination of *the optimal regulator.* The regulator will minimize the chosen performance index.

Example: If the energy absorbed by a dynamic system should be minimized, then the instantaneous quantities of the absorbed current and the supply voltage will be included in the performance index, and the controller will drive the system so as to minimize the energy absorbed from the power source.

Remarks:

1. Any solution to the optimal driving problem is called *optimal control;*
2. In the optimization problems, the poles of the system are not directly imposed, but result from the minimization of the chosen performance index;
3. Only the determination of the minimum of the performance index can be considered, because the problem of determining the maximum is similar to finding the minimum of the negative function.

3 Classification of the Optimal Problems

Optimal problem with quadratic functional applied to linear systems are well known as *linear quadratic regulators* (LQR) [2]. The LQR problems minimizes the consumed energy into system. LQR problems can be classified as:

- the problem of adjustment according to the state;
- the problem of the adjustment according to the output.

The linear quadratic optimal driving problems lead to solving different types of the Riccati equations: algebraic, differential (smooth case) or in differences (discrete case), solution that acts as a gain factor on the state feedback of the system.

The solution which minimizes the functional cost is named *optimal control* (OC). OC can be with or *without* constraints, with *fixed* or *free time* problems.

Particular cases:

(1) *The tracking problem,* when a certain trajectory becomes an objective to be achieved;
(2) *The problem* of *optimal control* with *fixed time,* when it is constraint to the final time t_f. The performance index becomes:

$$J(\mathbf{u}) = K(T) + \int_{t_0}^{T} L(x, u, \tau) d\tau \tag{5}$$

(3) *The problem* of control *with final state fixed, free final time,* when a desired state is required, $\mathbf{x}_f$.
(4) *The adjustment problem* when the fixed end point satisfy:$\mathbf{x}^* = \lim_{t \to \infty} \mathbf{x}(t)$, i.e. $\mathbf{x}^*$ is a state of equilibrium of the system.

4 Types of Optimizations

Continuous optimization

Models with continuous variables that can take any real value are problems of continuous optimization

Discrete optimization

The variables into the discrete optimization problems are included into a set of the discrete values.

Optimization without constraints—the constraints do not found on model variables.

Optimization with constraints—*the constraints* are presented on the model variables.

Optimization problems with ***constraints*** can be reduced to problems without constraints by replacing them with penalty terms in the chosen objective function.

The types of the optimization problems with constraints can be found by taking into account both the nature of the constraints and property of the functions (differentiable; non-differential). The constraints could be linear, nonlinear, or convex type.

Optimization problems without objective function

In practice there are problems without a special optimization objective, e.g. to find the variable values by satisfying the imposed model constraints.

This type of problem is named the *feasibility* problems

Optimization problems with objective functions

Usually, the one objective function into the optimization problems can be found. Moreover, there are special cases with multiple objective functions.

Problems of complementarity and variational inequalities are ubiquitous in engineering field, combined or not with economic field. The main objective is to satisfy the conditions of complementarity by the deducted solution.

The *optimization* problem with *multi-objective functions* occurs in logistics, or enginnering combined or not with economics. In these types of optimization problems the solution is a compromise of the goals conflict.

However, problems with multiple objectives can be reformulated as a single objective problem by forming a weighted combination of the functions objectives, or changing part of the function objectives by the adequate constraints.

Deterministic optimization

In this type of *optimization*, it is supposed that the required data to solve the problem is precisely known. Many times in practice, the data is missing (the often case is that there are errors in measurements, or for forecasting problems the future data cannot be known accurately).

Stochastic optimization

There are problems with the uncertain parameters into the used model in which the optimization stage is necessary. The robust control field takes into account these type of the processes conducting to a robust optimal solution. The robustness character is justified by controlling the process within a certain limits, but without knowing the all process data (structural uncertains). In stochastic problems, the advantages of knowing the probability distributions of the problem data are used; the objective is to find a solution to optimize the model performances for any data inputs.

Combinatorial optimization

The study of the arrangements of the elements of a *finite* set, according to a given *structure*, leads to *combinatorics*. In *combinatorial problems*, priorities are problems related to *existence* (there is a particular type of arrangement) and *counting* (the number of arrangements that can be formed):

The characteristic of combinatorial problems is that the number of these arrangements is finite.

The combinatorial optimization principle involves comparing the arrangements based on a criterion and selecting the best arrangement according to that criterion. The purpose of combinatorial optimization is to solve specific problems, i.e. to develop methods and algorithms for effectively finding the most suitable "arrangement" from the finite set of all possible arrangements.

Travelling salesman problem (TSP) is one of the combinatorial optimization problem.

Travelling salesman is located in town 0 must visit in localities *1,2,…,n* and finally returning from where he left off. Knowing the distances between localities (or the costs of traveling between them) the solution to find the route with the minimum total length (or the minimum total cost) is a concern. It can be easily deduced that the total number of the possible routes is $n!$, a route being perfectly determined by the order in which the ones will be visited in the localities. Once this order is established, the calculation of the length or costs of the corresponding route is a simple operation of summing the lengths or costs of the sections that make up the route.

The mathematical model of the TSP

By consider *0,1, …, n* the set of the towns, in which 0 is the town from which the travel salesman depart and arrive at the end of the travel. Is denoted by $\mathrm{c_{ij}}$ *cost of* displacement from the locality *i* to the $\mathrm{j} \neq \mathrm{i}$. If there is not direct connection between the *i* and *j* or in the case of $i = j$ then $\mathrm{c_{ij}} \to \infty$. A *route* will be described using the *bivalent* variables: if the route starts from city *i* directly to city *j*, $i \neq j$ then $\mathrm{x_i} = 1$, otherwise, $\mathrm{x_i} = 0$ (or if $i = j$).

In these conditions the following function is defined

$$f = \sum_{i,j=0}^{n} c_{ij} x_{ij}. \tag{6}$$

The constraints of the problem are given by the relations (7–8):

$$\sum_{j=0}^{n} x_{ij} = 1, \quad i = 0, 1, \ldots, n \tag{7}$$

and taking into account that from any city *i* the traveling salesman should straighten to the another single location,

$$\sum_{i=0}^{n} x_{ij} = 1, \quad j = \overline{0, n} \tag{8}$$

and at any time the travel salesman comes from a place previously visited.

Equations (6)–(8) represent the mathematical model of the *minimum tour* problem.

Heuristic methods for solving TSP

Usually, the optimal solutions of a TSP is very hard to find (if not impossible to find) if the number of the visiting cities are higher than of 50. To determine an acceptable solution more heuristic procedures have been developed. These procedures are attractive from two points of view:

- Can give a “guarantee certificate” for the obtained solution in the sense of the possible evaluation of the “maximum exceeds” from the optimal solution;

- The approximately solution can be found with a moderate *computational* effort into a *reasonable time*, that it meant those two parameters are *polynomial* dependent from the size of the problem (i.e. the number of the cities);

The heuristic methods builds a solution *by trial, by making at each iteration the best possible choice.* Unfortunately, this scheme *does not* usually lead on *the best globally solution.*

Next, several heuristics are presented to solve *the Euclidean problem of the travel salesman,* i.e. of the problem in which $c_{ij\,j}$ are *distances* satisfying both symmetry conditions: $c_{ij} = c_{ji}$, and triangle inequality $c_{ik} \leq c_{ij} + c_{jk}$.

Heuristics—*nearest neighbor*

- Departure from the town 0 to the *nearest* town;
- From the last visited village bows by the nearest unmarked village; in case that there is no longer single location to be visited, returns in place of departure;

Most combinatorial optimization problems are modeled using graph theory. Most of them can be described alternatively by linear programs with integer variables, especially bivalents, hence the close link between combinatorial optimization and integer programming.

Among the problems of combinatorial optimization are:

- the problem of the *minimum path* value between two nodes of a graph;
- the problem of *maximum flow*;
- the problem of *minimum tour*.

Another heuristic problem is the **problem of the minimum cost tree**: a number of "points" must be connected to facilitate the transmission of a certain service. Between points there are "potential links" whose realization involves a certain cost. The problem that arises is to see what connections will actually be made in such a way that any two points will be connected—directly or indirectly—in order to use the service, and the sum of the costs of the connections made will be minimal.

5 Unconstrained Optimization Problems

Overview

Numerical methods for solving unconstrained optimization problems are applied to the objective functions of one or more variables and aim to determine the global extremum point of the function.

From the calculated values point of view, the methods of solving the optimal problems without constraints can be classified in [3–8]:

- *Direct methods*—no derivatives of the objective function are required;
- *Indirect methods*—the derivatives of the objective function are used.

The difference between the direct and indirect methods is mainly based on the presence of the derivatives of the objective function. In the direct methods, there are no objective function derivatives.

The methods that use derivatives calculus have the advantage of a high convergence speed, but the volume of calculations increases, and errors can occur.

Depending on the used principle, the methods are divided into:

1. **Exploration** methods
2. **Removal** methods
3. **Search** methods (first order, second order)

1. The first two methods (**exploaration**, and **removal**) are used to find the domain in which the extremum of the function is located, but are used more in the case of functions of two or maximum three variables, since they are difficult to apply to functions of several variables for one-dimensional extremities.

2. ***Removal Methods.***

The objective function must satisfy the unimodality hypothesis (there is only one extreme point on the definition domain).

Principle: the considered area is divided into two parts by a (segment) separation plan; the value of the function is tested in the two sub-domains by a specific procedure and the one that is not of interest is eliminated; the procedure is continued for the remaining domain by dividing with a separation plan, etc., until a sufficiently small domain is reached, depending on the accuracy required.

3. **First order methods. Linear search methods**

They are the most efficient methods of solving unconstrained problems.

The principle of these methods consists in the iterative approximation of the extremum point, the procedure ending when the stop criterion is satisfied.

The general calculation formula is:

$$x^{i+1} = x^i + \theta_i h^i$$

where

h^i—is the direction of travel

θ_i—is a scalar that represents the step of movement in the h^i direction.

3.1 Indirect Search methods or *gradient methods* apply for derivable functions f (x) in relation to all arguments.

Methods of the II-nd order, in which the derivatives of the I-st order of the function are used:

$$x^{i+1} = x^i - \theta_i^* r^i, \quad r^i = \nabla f(x^i).$$

The step can be constant, variable (decreasing) or optim (*optimal gradient* method or *Cauchy method*).

3.1.1 Methods Based on Conjugate Directions Let x_0 the starting point of the algorithm and x^* the minimum point

$$x^* = x_0 + x_s,$$

$x_s \in \mathbb{R}^n$, unknown vector.

x_s is expressed in a base by using the conjugate directions.

Starting from x_0 and making successive steps along the axes of this base, there is the possibility that in n steps x^* to be reached.

The *conjugate directions (CD) method* can be regarded as an intermediate method between the *gradient* and the *Newton algorithms. The gradient method* has access to the values of the first derivatives order. In the Newton method, the second order derivatives are used. The CD method aims to accelerate the slow convergence rate of the gradient method and at the same time avoid the use of Hessian as in the Newton method. The particular case of the *conjugate directions* method is the *conjugate gradients method*, which was initially developed for quadratic problems. By approximation, this technique has been extended to general optimization problems because it can be argued that near a local minimum point the objective function is approximately quadratic.

This method consists of the following steps:

(a) $r_0 = \nabla f(x_0)$, and $d_0 = -\nabla f(x_0)$;
(b) $x_{k+1} = x_k + \alpha_k d_k$ for any $k \in [0, n-1]$, with $\alpha_k = -\frac{r_k^T d_k}{d_k^T \nabla^2 f(x_k) d_k}$;
(c) $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$, where $\beta_k = \frac{r_{k+1}^T d_k \nabla^2 f(x_k)}{d_k^T \nabla^2 f(x_k) d_k}$;
(d) at each iteration i, the variable x_0 is replaced by x_i. The above mentioned procedure repeats at each iteration.

This method has the following two main disadvanteges:

- requires the calculation of the Hessian objective function for each iteration,
- is not convergent for the general case.

3.1.2 Modified Method of Conjugate Gradients (MMCG) This method attempts to correct the disadvantages of the conjugate gradient method listed above. In this case, α_k it is calculated by using another method (ideal, Wolfe conditions or backtracking), and β_k is calculated with one of the two formulas proposed below:

- Fletcher–Reeves (FR): $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$,
- Polak–Ribiere (PR): $\beta_k = \frac{(r_{k+1}^T - r_k)^T r_{k+1}}{r_k^T r_k}$.

The PR method behaves better than the FR method.

Thus, for MMCG problems without constraint, the following steps are followed:

(a) $r_0 = \nabla f(x_0)$ and $d_0 = -\nabla f(x_0)$;
(b) $x_{k+1} = x_k + \alpha_k d_k$, for any $k \in [0, n-1]$, with α_k calculated with Wolfe conditions or backtracking;
(c) $d_{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$, where β_k is calculated using one of the FP or PR methods.
(d) at each iteration k, replace x_0 with x_k and repeat the process described above.

3.2 Second Order Methods

I. Newton Methods

I.1 N method (NM)

The NM (1669) is part of the second order optimization method category. The conditions of sufficient optimality of the second order are: if there is an x^* which satisfies the (a, b) conditions simultaneously:

(a) $\nabla f(x^*) = 0$,

(b) $\nabla^2 f(x^*) > 0$,

then x^* can be considered as a local minimum.

An important aspect of this method is that the descent direction, $d_k (k > 0)$, can be considered if Hessian matrix is positive definite:$\nabla^2 f(x_k) = D2f_k > 0$.

For each iteration of the algorithm, the following recursive equation (quadratic approximation deducted from Taylor series expansion of the objective function) should be taken into account:

$$x_{k+1} = x_k - \theta_k (D2f_k)^{-1} D1f_k,\ D1f_k = \nabla f(x_k),$$

with step $\theta_k = 1$, and descent direction d_k:

$$d_k = -(D2f_k)^{-1} D1f_k.$$

If the starting point x_0 is not placed into the small perturbations arround of the point x^* interval, the NM cannot guarantee the convergence of the method. This is the main disadvantage of the NM. Another disadvantage of the Newton method is that at each iteration the Hessian and its inverse should be calculated.

Therefore, an improvement has been made by choosing the steps $\theta_k \neq 1$ in optimal manner, resulting so-called *Newton‘s variable step method.*

I.2. Quasi-Newton method

To overcome the above mentioned disadvantages, the *quasi-Newton method* is used, in which the inverse of Hessian $(D2f_k)^{-1}$ is replaced with an approximated matrix

H_k. This matrix can be calculated much easier, and at the same time keeps the fast convergence speed of the NM.

The principle of the Newton methods consists in the successive approximation of the Hessian matrix or its inverse by appealing only to the first order derivatives of the objective function.

I.2.1 Davidon-Fletcher-Powell (DFP) method

Let B be the matrix that approximates the inverse of the Hessian matrix.

Initially, a point x_0 is chosen and a symmetrical matrix $B_0 > 0$ (usually $B_0 = I$).

Then, $\nabla f(x^i)$ and optimal step θ_i in the travel direction $h^i = -B_i \nabla f(x^i)$ are computed.

It is calculated $x^{i+1} = x^i + \theta_i h^i$.

Check if the stop criterion is met.

If it is *yes*, then *stop* the algorithm,

otherwise, $B_{i+1} = B_i + M_i + N_i$ is computed.

It follows the next iteration.

I.2.2 Broyhen-Fletcher-Goldfarb-Shanno Method (BFGS)

Initially, a point x_0 is chosen and a symmetrical matrix $B_0 > 0$ (usually, $B_0 = I$). Following this, $\nabla f\left(x^i\right)$ and optimal step θ_i on the travel direction $h^i = -B_i \nabla f\left(x^i\right)$ are computed.

It is calculated $x^{i+1} = x^i + \theta_i h^i$.

Check if the stop criterion is met, and if it *yes*, *stop* the algorithm. An update of the Hessian matrix value is delivered.

Continue with the next iteration.

Comparatively, the BFGS algorithm is less affected by the errors in the optimal step calculation than the DFP algorithm. As regards the numerical stability, BFGS is considered the most stable method. From the ones presented so far, it can be observed that quasi-Newton methods only require first order information (only gradient type calculations are used).

The NM is a good choice for the multi-variable equation systems solving, and to find the polynomials complex roots. Newton wanted to determine the solution of an "algebraic" given problem: F(x, y) = 0, in which the variable *y* is expressed as a series of powers in x.

III. Newton-Raphson methods (NRM)

The NRM supposes the determination of the second order derivatives of the objective function. The NRM has the main disadvantage the requirement of a large amount calculus, but this is compensated by the speed of convergence. The Raphson procedure is equivalent to the linear approximation. The first analysis of the NM convergence had been done in 1820 by the Cauchy, and Fourier [3, 7, 9].

III.1. Newton-Raphson method

The method is based on the linear approximation of the objective function by the Taylor series expansion. The general recurrence relationship is:

$$x^{i+1} = x^i - H_i^{-1}\nabla f\left(x^i\right)$$

where

H_i-the Hessian matrix of the objective function at iteration i

There are some disadvantages of the recurrence formula in this form: the method can converge to a saddle point or to an extremely relative point. Thus, the recurrence relation can be modified, eliminating the mentioned problems by moving at the optimum pace on the direction of movement. The relationship becomes:

$$x^{i+1} = x^i - \theta_i^* H_i^{-1}\nabla f\left(x^i\right)$$

For quadratic functions, the extremum point is reached in one step. For the other functions, the extreme point is not found in a single step, but convergence is fast.

Even with this new recurrence relation, the method cannot be used in the case of objective functions of many variables, since:

- nxn dimensional matrix is stored at each iteration
- Each iteration is calculated $H_i^{-1}\nabla f\left(x^i\right)$
- In certain situations it is impossible to calculate the elements of the Hessian matrix
- At each iteration, the inverse of the Hessian matrix is calculated.

III.2 The Marquardt method

It combines the advantages of the optimal gradient method (safe convergence and rapid decrease of the function value for the case of choosing the starting point far from the minimum point) and the Newton-Raphson method (rapid convergence if the starting point is near to the minimum point).

IV. Trust region method (TR)

In this method, based on the information gathered about the objective function f an objective function T_i is so built, that near the point x_i, it behaves just like function f. Since objective T_i is not always the best approximation of f we have to force the search for a minimizer of T_i to a certain area around x_i. In this regard, the step s is search such that the next subproblem will be solved by approximation:

$$\min_s \quad T_i(x_i + s),$$

in which $(x_i + s)$ is placed in the TR.

If the solution obtained does not lead to a significant decrease in f, it means that the TR is overgrown; therefore we reduce the step and proceed again to solve the above mentioned subproblem.

The procedure of applying TR method consists of:

1. Maximum distance Δ_i, i.e. the *radius* of the TR, is chosen;
2. Search for a direction d with a step s such as the maximum decreasing rate is obtained;

3. If the obtained result is unsatisfactory, proceed to the choice of a smaller Δ_i, and repeat the algorithm from *step* 1.

By considering the first two terms of the quadratic model of the function T_i, and the first two terms of the f Taylor's series expansion around x_i that are identical for each iteration x_i, the following relationship can be deducted:

$$T_i(s) = f_i + s^T \nabla f_i + \frac{1}{2} s^T B_k s,$$

with symmetric matrix B_k.

At the same time, by applying Taylor's theorem for a continuous and differentiable function f¸the approximation function around s point can be written:

$$f(x_i + s) = f_i + s^T \nabla f_i + \frac{1}{2} s^T \nabla^2 f(x_i + ts)s,$$

with $t \in (0, 1)$, a scalar number.

In this way, the following approximation T_i is adopted:

$$T_i(s) = f_i + s^T \nabla f_i + O(\|s\|^2).$$

The difference between T_i and $f(x_i + s)$ it is just $O(\|s\|^2)$. The approximation error is small if s is small.

As a consequence, the following problem should be solved: norm

$$\min T_i(s) = f_i + s^T \nabla f_i + \frac{1}{2} s^T B_k s, \text{ with norm } \|s\| \le \Delta_i, \tag{9}$$

$\Delta_i > 0$ being the trust-region radius.

Three strategies are described for identification of approximate solution, based on which at least one reduction of T_i is obtained, as well as the reduction obtained using the Cauchy point. This point is actually a T_i minimizer along the direction that offers the steepest descent—∇f_i within the TR.

V. The Cauchy Point

In the line search methods, the approximation is sometimes coarse from an optimal length of the step. However, this does not affect global convergence. In order to simplify the calculations, the same methods can be used in trust region methods. In other words, instead of finding an optimal solution to (8.93), to identify an approximate solution s_i is a better approach. Therefore, to achieve global convergence, this approximate solution should be placed within the trust region, and to provide a satisfactory reduction of the objective function.

The satisfactory reduction is obtained using the Cauchy point p_i^C which can be defined by using the following algorithm.

Algorithm:

– Looking for a vector p_i^S which is a linear solution of the (8.93) problem

$$p_i^S = \arg \min_{s \in R^n} f_i + s^T \nabla f_i, \quad \|s\| \le \Delta_i$$

– The $\tau_i > 0$ scalar is determined such that $T_i(\tau p_i^S)$ is minimized, i.e.

$$\tau_i = \arg \min_{\tau > 0} T_i(\tau p_i^S), \text{for } \left\| \tau p_i^S \right\| \le \Delta_i;$$

– Set

$$p_i^C = \tau_i p_i^S,$$

results:

$$p_i^C = -\tau_i \frac{\Delta_i}{\|\nabla f_i\|} \nabla f_i,$$

in which

$$\tau_i = 1, \text{for } \nabla f_i^T B_i \nabla f_i \le 0,$$

otherwise

$$\tau_\mathrm{i} = \min(\|\nabla f_\mathrm{i}\|^3 / (\Delta_i \nabla f_i^T B_i \nabla f_i), 1).$$

Always taking the Cauchy Point as the step of the method gives the abrupt descent method. For example, this is not the case for *steepest descent method* even if an optimal step size is chosen at each iteration.

Because the matrix B_i is used only to calculate the step length, the Cauchy point has a lower dependence on it.

A fast (superlinear) convergence is obtained if B_i is exactly Hessian $\nabla^2 f(x_i)$ or a quasi-Newton approximation. The matrix B_i is used for both the direction calculation, and the step length.

The *dogleg method* is one way to find the solution to the problem (9). Another ways, are to use *Steihaug method,* or *two-dimensional subspace* minimization.

In order to increase the clarity, the problem (9) can be simplified by considering the following single iteration problem:

$$\min_{\mathrm{s} \in \mathrm{R}^\mathrm{n}} T(s) \overset{def}{=} f + q^T + \frac{1}{2} s^T B s, \quad \|s\| \le \Delta \tag{10}$$

having the solution $s^*(\Delta)$.

VI: Dogleg method

The *Dogleg* method is used to find a two line segments approximate solution instead of a curve path for $s^*(\Delta)$. The origin is the initial value of the first path segment to the unconstrained minimizer with the steepest descent direction defined by:

$$s^U = -\frac{q^T q}{q^T Bq} q.$$

The second path segment starts from the point s^U to the point s^B having a trajectory defined by $d(\tau)$:

$$d(\tau) = \begin{cases} -\tau s^U, & 0 \le \tau \le 1 \\ s^U + (\tau - 1)(s^B - s^U), & 1 \le \tau \le 2 \end{cases}$$

with $\tau \in [0, 2]$.

If the norm $\|s^B\| \geqslant \Delta$, then the path $d(\tau)$ crosses the boundary of the TR, $\|s\| = \Delta$, at one point (Fig. 1).

VII. Two-dimensional subspace minimization

By extending the search to the entire surface covered by s^U and s^B(*two-dimensional subspace*), a simplification of the *dogleg* method is obtained.

In these conditions, when matrix $B > 0$, has the positive eigenvalues, the subproblem (10) becomes:

$$\min_{s} T(s) \overset{def}{=} f + q^T s + \frac{1}{2} s^T Bs, \ \|s\| \le \Delta, \ s \in span[q, B^{-1}q], \tag{11}$$

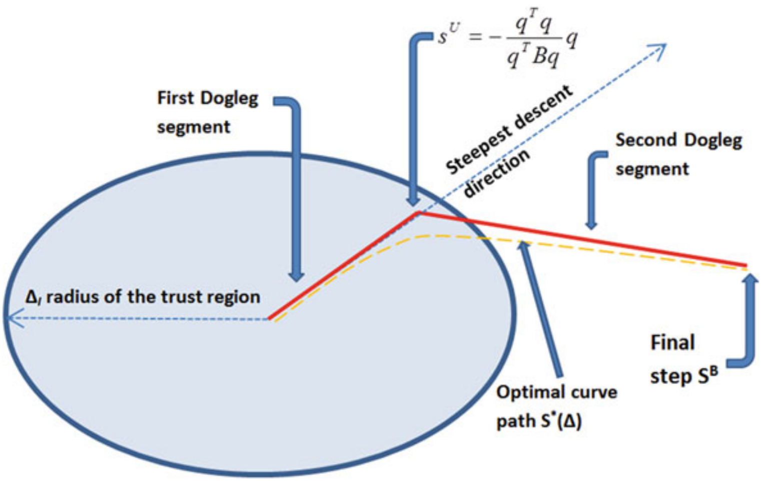

$s^U = -\frac{q^T q}{q^T Bq} q$
First Dogleg segment
Steepest descent direction
Second Dogleg segment
Δ, radius of the trust region
Final step S^B
Optimal curve path S*(Δ)

Fig. 1 Illustration of the Dogleg method

in which the span operator means the possible linear combinations set between vectors q and $B^{-1}q$.

The problem (11) can be solved relatively easy. The optimal solution to this subproblem: p_i^C—the Cauchy point.

In case of $B < 0$, the sub-problem (10) is changed to:

$$span[q, (B + kI)^{-1}q]\,, \text{for}\, k \in (-\lambda_1, -2\lambda_1], \tag{12}$$

where λ_1 indicates the most negative value of the eigenvalue of the matrix B.

The value of k is chosen such that $B + kI$ is positively defined.

In case of $\left\|(B + kI)^{-1}q\right\| \leq \Delta$, the new step is defined as:

$$s = -(B + kI)^{-1}q + a$$

with value of a chosen such that the condition $a^T(B + kI)^{-1}q \leq 0$ is satisfied.

VIII. Steihaug's approach

The above presented two methods (*dogleg, two-dimensional subspace minimization*) assume a linear system whose solutions depend on B or $(B + aI)$. But solving this system when B is big becomes costly. Therefore, in practical situations other techniques must be identified to find a solution to the subproblem (12) that does not depend on the exact solution of the linear system, but which nevertheless leads to an Cauchy point improvement.

In this regard, Steihaug proposed a method similar to the *conjugate gradient algorithm* (as it is previously described).

The main difference between the standard conjugate gradient algorithm and Steihaug's approach is that the algorithm is finalized when either the limits set by the trust region $\|s\| \leq \Delta$ are exceeded or when a negative curvature direction is reached in B.

In the *Newton trust region method*, the B matrix is chosen to be exactly Hessian $\nabla^2 f(x)$ (or approximations thereof). These methods have very good local and global convergence properties.

Unlike the line search methods where a fixed direction d_i and at each iteration a length of step α_i, is determined, in the trust region, firstly a radius of this region is chosen in the form of a maximum distance Δ_i, and, secondly, to both *direction* and *step* are explored such that the best improvement is possible. If the result is not satisfactory, the distance Δ_i is reduced, and a new minimizer should be determined. In general, as the radius of the trust region changes, the direction of the step changes. Very important at each stage is the magnitude of the trust region. Often, the size of the trust region is based on the previous obtained results (if the result is satisfactory) from the iteration algorithm. If the region is not big enough, the algorithm passes a good opportunity to take an increased step that will position the results much closer to the objective function minimizer. If it is too large, the model minimizer may be far from the objective function minimizer in the region and the size of the TR is reduced (repeat the algorithm).

IX. ***Generic TR Method***

In order to establish the border of the TR, a maximum radius should be chosen $\Delta_{\max} > 0$.

Similiarly, the distance $\Delta_0 \in (0, \Delta_{\max})$, and a fixed η within (0, 1/4) interval, the starting point $x_0 \in R_i$ within the TR R_i, B_0, $\varepsilon > 0$ should be initiated.

While $\|\nabla f(x_i)\|$ norm is larger or equal than ε, do

Compute y_{i+1} as the approximate minimiser of $x_{i+1} \approx \arg\min_{x\in R_i} T_i(x)$;

Determine x_{i+1}:

$$x_{i+1} = y_{i+1}, \text{ if } \frac{f(x_i) - f(y_{i+1})}{T_i(x_i) - T_i(y_{i+1})} > \eta,$$

$x_{i+1} = x_i$ otherwise;

Compute Δ_{i+1}:

$$\Delta_{i+1} = \frac{\Delta_i}{4}, \text{ if } \frac{f(x_i) - f(y_{i+1})}{T_{ki}(x_i) - T_i(y_{i+1})} < \frac{1}{4},$$

$$\Delta_{i+1} = \min(2\Delta_i, \Delta_{\max}), \text{ if } \frac{f(x_i) - f(y_{i+1})}{T_i(x_i) - T_i(y_{i+1})} > \frac{3}{4};$$

$\Delta_{i+1} = \Delta_i$, otherwise;

Build a new model function $T_{i+1}(x)$.

$$i \leftarrow i + 1$$

end.

3.2. Direct Search Methods

These methods are also used in the case of non-divisible functions for determining the values of the objective function. As disadvantages, convergence is slow, based on simple calculations at each iteration.

Direct search methods can be—*iterative methods*—points are getting closer and closer to the minimum.

As algorithm methods can be mentioned:

- Gauss-Seidel,
- Nelder-Mead
- downhill simplex, or amoeba method,
- Rosenbrock and Powell,
- SIMPLEX and COMPLEX.

3.2.1. Nelder-Mead [10]

This optimization method takes part from the direct search optimization methods.

By considering a quadratic function defined by the user as:

```
function y = fnm (x)
  y = 0
for k = 1 : size (x, 2) - 1
y = y + 68 * (x (k + 1) - x (k)) ^ 2 + (1 - x (ik)) ^ 2 - x (k) * x (k + 1);
end
endfunction
```

the problem is to find the optimal value through the Nelder-Mead method.

The optimal searching method is based on construction of new solutions by using a simplex structure and a transforming series. This method does not require the deduction of the gradients.

The detailed optimal structure solution can be found as:

opt = optimset (“Display”, “iter”);

The structured solution is composed by three elements: number of iteration, the function count, and the minimum value of the function at each iteration. The method can be used to improve the evolution of the optimization process.

Results: the minimum value of the quadratique function is found after 12 iterations as in the following structure (Fig. 2).

Iteration	Func-count	min f (x)
0	3	0
1	3	0
2	7	0
3	11	0
4	15	0
5	19	0
6	23	0
7	27	0
8	31	0
9	35	0
10	39	0
11	43	0
12	47	0

3.2.2. Rosenbrock Method [10]

The problem is to find the optimum value of the nonlinear function of two variables, $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$:

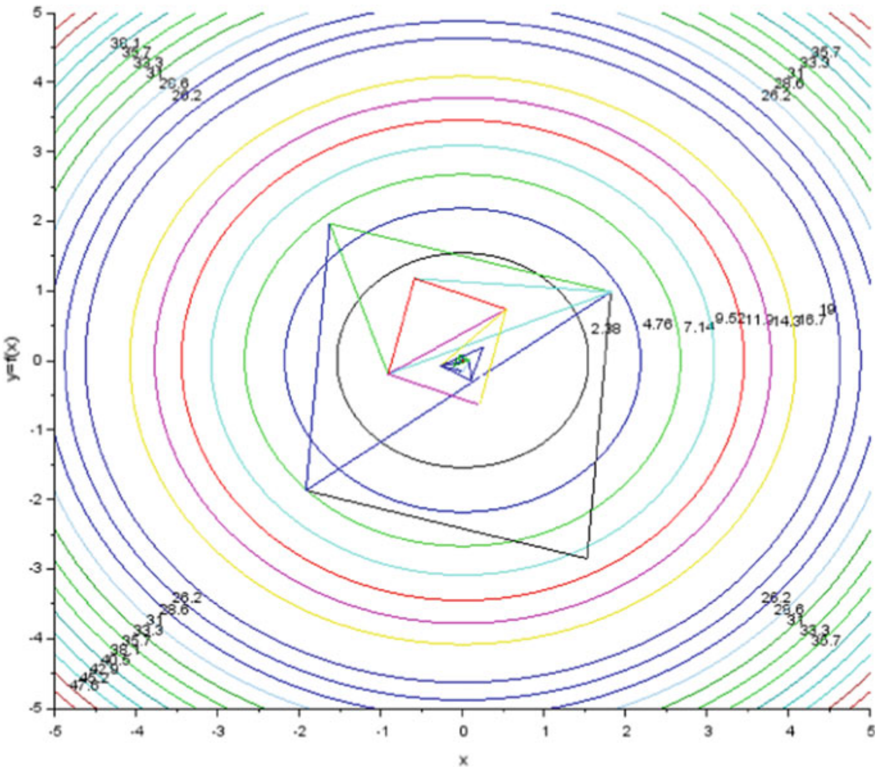


Fig. 2 Nelder Mead method for solving a quadratique function [10]

$$f(x_1, x_2) = (2 - x_1)^2 + 68\left(x_2 - x_1^2\right)^2$$

near to [−1.2 1.0].

The multivariable function mentioned above is known is Rosenbrock function.

This optimization problem takes part of the nonlinear optimization without constraints.

The optimum value is find by using the three basic methods:

(i) Quasi-Newton BFGS,
(ii) Quasi-Newton BFGS with limited memory;
(iii) QN for non-diferentiable objective function (local optimization without using the derivatives).

(i) Quasi-Newton BFGS method is a local optimization based method, and uses the specification of the first derivative respect to each variable.

The software implementation of the above mentioned optimization problem supposes:

– The definition of the index function:
  f = 60 *(x(2) – x(1)^2)^2 + (2 – x(1))^2;
– The first order derivative respect to $x_1(1) = D1(1)$:
  D1(1) = –272. * (x(2) – x(1)^2) * x(1) -2. * (2. –x(1));
– The first order derivative respect to $x_1(2) = D1(2)$:
  D1(2) = 136. * (x(2) – x(1)^2).

After definition of the cost function and its first derivatives, the starting point of the optimization algorithm: in this example the initial point is placed at $[-1.2\ 1.0]$ blue mark in Fig. 3a. The initial estimate of the solution coordinates should be also be provided: the initial guess is placed at [1.0 1.0] (red point on the Fig. 3a).

Contours of the particular Rosenbrock function of two variables (a = 2, b = 68) are plotted on the Fig. 3a, the optimization process is depicted on the Fig. 3b, and by using the QN-BFGS algorithm the minimum value of the objective function is found at $[x_{1\min}, x_{2\min}] = [2, 4]$, having the scalar value $f_{\min}(x_1, x_2) = 0$.

The optimum function value fopt, for the initial guess × 0 can be find as:

fopt1 = 0

placed at the optimum coordinates $[x_{1\min}, x_{2\min}] =$ xopt1

xopt1 = [2. 4.]

(ii) By using the quasi-Newton method based on Broyden-Fletcher-Goldfarb-Shanno (BFGS) with limited memory, the same values are found. This method is applied to large number of variables (more than 100):

fopt2 = 0

at the $[x_{1\min}, x_{2\min}] =$ xopt2

xopt2 = [2. 4.].

(iii) By using the third method, for non-differentiable functions, an approximation of the optimal solution is obtained as:

fopt3 = 0.0004828,

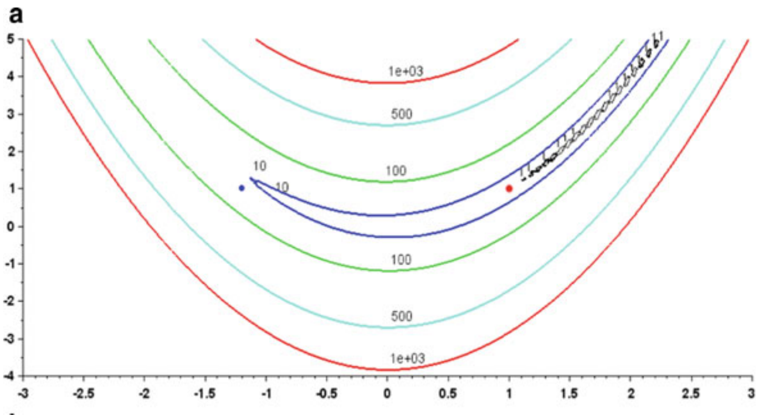
the optimum value of the nonlinear multivariable function being placed at,

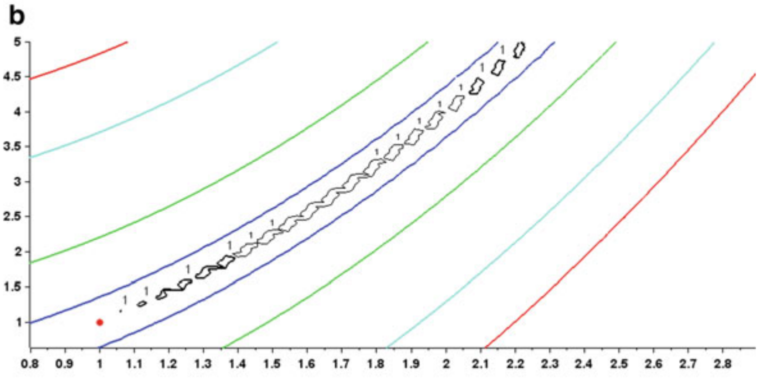
xopt3 = [1.9866225 3.9487826].

In order to find the optimal value of the cost function,

The optimization based on the Nelder-Mead search method does not use the information about the gradient of the function. The function value evolution during the unconstrained optimization process can be plotted for the above known function.

In Fig. 3c the dynamic of the optimization process is shown As can be seen, the last half of the iterations maintain the cost function value is almost maintained at the constant value. The optimal value of the cost function is 1.389381e-10 (Fig. 3c).





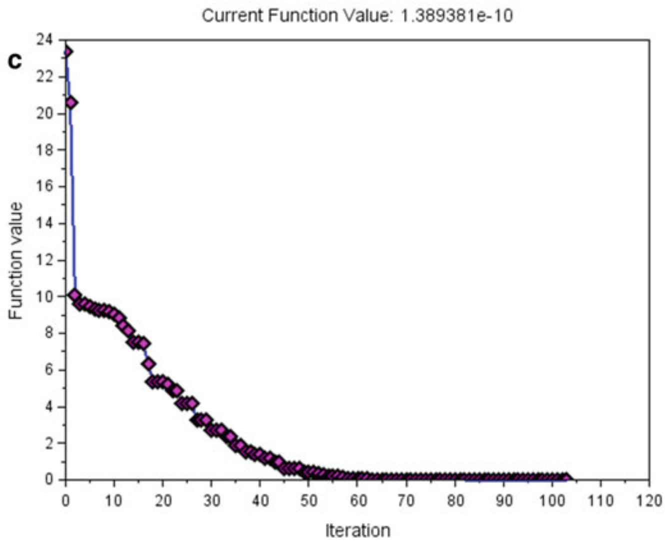


Fig. 3 **a** Contours of the particular Rosenbrock function of two variables (a = 2, b = 68). **b** Plot the initial guess of the optimum variables $[x_{1\min}, x_{2\min}] = [1]$ and the evolution of the optimization process. **c** Cost value: the evolution of the minimization process through the used Nelder-Mead algorithm

The dynamic of the transformations at each iterate, as well as the applied optimization procedure at each step can be viewed:

Iteration	Func-count	min f(x)	Procedure
0	3	23.4048	
1	3	20.5828	initial simplex
2	5	10.054469	expand
3	7	9.6029493	reflect
4	9	9.6029493	contract outside
5	11	9.4738768	contract inside
6	13	9.31917	contract inside
7	15	9.2507973	reflect
8	17	9.2507973	contract inside
9	19	9.1980872	expand
10	21	9.0477781	expand
11	23	8.8547479	expand
12	25	8.3861285	expand
13	27	8.1417887	expand
14	29	7.4969757	reflect
15	31	7.4969757	contract inside
16	33	7.4288297	expand
17	35	6.3523046	expand
18	37	5.3708076	expand
19	39	5.3708076	contract inside
20	40	5.3708076	reflect
21	42	5.2331712	reflect
22	44	4.8457422	reflect
23	45	4.8457422	reflect
24	47	4.1618679	reflect
25	49	4.1618679	contract inside
26	51	4.1442689	reflect
27	53	3.2469009	expand
28	55	3.2469009	contract inside
29	57	3.2469009	contract outside
30	59	2.6898366	expand
31	61	2.6898366	contract outside
32	63	2.6898366	contract outside
33	65	2.334585	expand
34	66	2.334585	reflect
35	68	1.8784037	expand

(continued)

(continued)

Iteration	Func-count	min f(x)	Procedure
36	70	1.8784037	contract inside
37	72	1.5530071	expand
38	74	1.5530071	contract outside
39	76	1.3613402	reflect
40	77	1.3613402	reflect
41	79	1.1890049	reflect
42	81	1.1890049	contract inside
43	83	0.9850215	expand
44	84	0.9850215	reflect
45	86	0.6400976	expand
46	88	0.6400976	contract inside
47	90	0.6400976	contract outside
48	92	0.6241326	reflect
49	94	0.435756	expand
50	96	0.435756	contract inside
51	98	0.435756	contract outside
52	100	0.3462121	expand
53	102	0.2962515	reflect
54	104	0.2115731	reflect
55	106	0.2115731	contract inside
56	108	0.1905142	expand
57	110	0.1339536	reflect
58	112	0.0576853	expand
59	114	0.0576853	contract inside
60	115	0.0576853	reflect
61	117	0.0548968	reflect
62	119	0.043614	contract inside
63	121	0.026894	reflect
64	123	0.026894	contract outside
65	125	0.0176597	contract outside
66	127	0.0176597	contract inside
67	129	0.0158285	contract outside
68	130	0.0158285	reflect
69	132	0.0134937	reflect
70	134	0.0133218	contract inside
71	136	0.0114001	reflect
72	138	0.0114001	contract inside

(continued)

(continued)

Iteration	Func-count	min f(x)	Procedure
73	140	0.0094043	expand
74	142	0.0073118	expand
75	144	0.0034168	expand
76	145	0.0034168	reflect
77	147	0.0003337	expand
78	148	0.0003337	reflect
79	149	0.0003337	reflect
80	151	0.0003337	contract inside
81	153	0.0003007	contract outside
82	155	0.0000446	contract inside
83	157	0.0000446	contract inside
84	159	0.0000178	contract outside
85	161	0.0000071	contract inside
86	163	0.0000058	contract inside
87	165	0.0000028	contract inside
88	167	0.0000006	contract inside
89	169	0.0000006	contract inside
90	171	0.0000005	contract inside
91	173	7.754D-08	contract inside
92	175	7.754D-08	contract inside
93	177	6.537D-08	contract outside
94	179	7.641D-09	contract inside
95	181	7.641D-09	contract inside
96	183	7.641D-09	contract outside
97	185	1.573D-09	contract inside
98	187	1.573D-09	contract inside
99	189	1.325D-09	contract inside
100	191	8.792D-10	reflect
101	193	1.389D-10	contract inside
102	195	1.389D-10	contract inside
103	197	1.389D-10	contract inside

The procedure *contract inside* means that the problem has a solution, and the termination criteria has been attained. In order to check the optimization progress, at each iterate there is a warning message regarding the convergence evolution of the algorithm. If the problem converges to the optimal solution, the following message could be viewed: Optimization terminated.

The variables **x** satisfies the termination criteria using OPTIONS.TolX of 0.0001 (tolerance) and cost function F(x) satisfies the convergence criteria using OPTIONS.TolFun of 0.0001.

By using Nelder-Mead method, the optimal variables pair is obtained,

xopt4 = [2.0000022 4.0000102],

conducting to an optimal cost function of:

fopt4 = 1.389D-10.

6 Nonlinear Optimization Without Constraints

The canonical form of *nonlinear programming* problems is [7] to find the optimal solution vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ such that the index function $z = f(x_1, x_2, \ldots, x_n)$ is minimized, having the $g_i(x_1, x_2, \ldots, x_n) \leq 0$, as constraints, with $i = 1, 2, \ldots, m$, and nonnegative conditions $x_1 \geq 0\ ,\ x_2 \geq 0\ ,\ \ldots\ x_n \geq 0$.

The canonical form can be synthesize as:

$$\begin{cases} \min f(\underline{\mathbf{x}}), & \underline{\mathbf{x}} \in R^n; \\ g_i(\underline{\mathbf{x}}) \leq 0, & i = \overline{1, m}; \\ \quad \underline{\mathbf{x}} \geq 0 & \end{cases}$$

In case of the *linear programming* problem, both the *index function*, *z*, and the *constraints*, g_i, are *linear.* Therefore, there is (at least) a general method of solving—for example the simplex method–in the nonlinear case there is no such method. However, substantial progress has been made in some special cases by imposing conditions on the functions f and g_i.

The non-linearity of the objective or some of the constraints leads to *complicating the* task of determining the optimum.

(1) From the beginning we will emphasize that in *nonlinear programming*—with a few exceptions—the methods of solving "theoretically" obtain the optimal solution as the *limit* of a series of solutions. Thus, a *concrete* process *of nonlinear optimization* is *finished* not due to the structure of the problem but by *the user's will* that limits the number of steps according to a whole series of factors such as: complexity of calculation, time available, performance of computing equipment, etc.
(2) it is possible that the objective function in (P) has more *local minima* on the set of admissible solutions A.

The possibility of the existence of several local minima of the objective function represents a serious difficulty in solving a nonlinear program. Indeed, in the formulation itself, such a problem requires the determination of the global minimum of the objective. However, all known non-linear optimization methods fail to determine at most a local minimum, with no guarantee that it coincides with the global minimum sought.

As we will see, if *A* is convex and the objective function is convex and minimized then it has at most a local minimum which - if any - is automatically global.

(3) Even though the constraints in (P) are *linear* but the objective remains *nonlinear* but *convex*, the optimal solution, although it is on the border of A, is not necessarily a *peak.*
(4) it is possible that the optimal solution is located inside the set A.

Classes of nonlinear problems

6.1 Unconstraint *Optimization Problems*

Unconstraint optimization problems have the general form:
determine $x^* \in R^n$ ^{n} which minimizes the value of the function

$$z = f(x_1, x_2, \ldots, x_n),$$

the minimum being taken after all $x \in R^n$ ^{n} where function *f* is defined.

6.2 Optimization Problems with Linear Constraints *and* Nonlinear *Objective Function*

In this class a special attention should be on the *quadratic programming* problems. The index function is a *polynomial* of the *second* degree in its variables:

$$f(x_1, x_2, \ldots, x_n) = c_0 + \sum_{i=\overline{1,n}} c_i x_i + \sum_{(i<j)=\overline{1,n}} c_i x_i x_j$$

The problems of quadratic programming are important for the following reasons:

- the fact that it models many practical situations with sufficient accuracy;
- it is solved by methods derived from the simplex method in a finite number of steps;
- solving many problems with linear constraints and nonlinear objective function can be reduced to solving a sequence of quadratic programming problems whose objective functions increasingly approximate the original nonlinear objective.

7 The Problems of Convex Programming

The problems of convex programming are characterized by:

- *convex* objective (index) *function* if it is *minimized* (equivalent: *concave* objective function if it is *maximized*);
- $g_i(x) \leq 0$, the *inequality* constraints are of the form $g_i(x) \leq 0$ in case of *convex* function g (equivalent $g_i(x) \geq 0$, in case of a *concave* function g);
- *equality* constraints are *linear*, a requirement motivated by the fact that linear functions are the only simultaneously convex and concave functions.

Convex problems have the following fundamental properties:

- the set of admissible solutions is *convex*;
- the objective function admits at most an optimal (minimum or maximum) *local*; automatically this will be a *global* optimum and will represent the optimal of the problem;
- if the *free* (*unconstrained*) optimal of the objective function is not an admissible solution then the *constrained* optimal is necessarily located on the *boundary of* set A.

The importance of this class of problems is *very high* due to the following reasons: Convex programming includes linear programming;

- In this area, the greatest research effort was made and the strongest *theoretical* results (such as nonlinear duality theory, Kuhn - Tucker optimality conditions) and *practical* (optimization methods and algorithms) were obtained;
- The whole mathematical formalism of modern economic theory is based on convexity assumptions.

8 The Problems of Separable Programming

The problems of separable programming are characterized by the fact that the index function f as well as the constraints g_I are *separable* within the meaning of the following definition:

The separable function $f(x_1, x_2, \ldots, x_n)$ should satisfy the following relation:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_i).$$

Separability is important because it facilitates optimization. For example, the optimization of a separable function without constraints is reduced to the independent optimization of terms.

9 The Problems of Non-convex Programming

The problems of non-convex programming bring together all the problems that do not satisfy the convexity hypotheses. These problems have several local minima. The current methods can cause such an optimum. However, these problems cannot guarantee that the solution is the global optimum. Fortunately, there are several types of non-convex problems, useful in practice that can be solved without special difficulties by special methods, added via *fractional* programming problems.

Sets and convex functions

By considering C as a set of the real *n*- dimensional space ($C \subseteq R^n$), the set is *convex* if for two containing points the segment joining them is inside of the C.

Mathematically,

C is a convex set if $(\forall)\quad x, y \in C,$ for any $\lambda \in [0, 1],$ the following relation is true $(1 - \lambda)x + \lambda y \in C$.

Supposing C is a convex set, and f a numerical function defined at all points of the set C, f is a *convex* function if:

$(\forall)\quad x, y \in C,$ for any $\lambda \in [0, 1],$ the following relation is true

$$f[(1-\lambda)x + \lambda y] - (1-\lambda)f(x) + \lambda f(y) \leq 0$$

Oppositely, the concave function should satisfy the folowing:

$$f[(1-\lambda)x + \lambda y] - (1-\lambda)f(x) + \lambda f(y) \geq 0.$$

For any different two points, *x*, *y*, with $(\forall)x, y \in C$, and $(\forall)\,\lambda \in (0, 1)$, the function f is named *strictly convex* (*strictly concave*) if the above mentioned relationships are strictely.

10 Nonlinear Optimization Without Constraints. The Convex Case

The general optimization problem:

(P) Find $x^* \in A \subseteq R^n$ with the property that the function value $f(x^*)$ satisfy the relation inf $\{f(x), x \in A\}$,

for any *x* from the admissible solutions (A) set of the problem (P), A is defined by a set of constraints:

$$gi(x) \leq 0, i \in M = \{1, 2, \ldots, m\}.$$

In order to simplify the explanation, any conditions of non-negativity $x_j \geq 0$ were included in the restriction block in the following form: $-x_j \leq 0$.

Suppose the functions f and g_1, g_2 …, g_m are defined entire the space R^n, ^{n} are convex and differentiable, and at least one of them is non-linear. In this way, the problem (P) is a *programming convex problem.*

Recall that in this context:

- A is a **closed** and **convex set**;
- any **local minimum** of function f on the set A it is a **global minimum**.

11 Optimization Methods with Constraints

Techniques for nonlinear software problems with constraints are superior to those without constraints.

If the constraints are linear, the algorithms are based on methods of optimizing the case without constraints. The basic approach for solving a multivariate constrained nonlinear problem is to reformulate it into a succession of related problems, so that each problem can be solved by simpler methods. As an example, Newton‘s methods for unconstrained problems based on a local quadratic approximation of the objective function can be developed for the unconstrained problem by restricting the region in which the quadratic model is valid. This constraint is, in principle, similar to the *trust region* method.

These optimization techniques are splitted in three categories:

(1) *methods of gradient*—based on the adaptation of the unconstrained general optimization scheme in case of constraints.
(2) methods based on *penalty functions*: solving the main problem reduces to more unconstrained optimized problems.
(3) methods based on the *plane sections*; its principle, these methods "approximate" by polyhedral set (a set that can be described by a system of linear inequalities); solving the problem is reduced to a sequence (infinite) of linear optimization problems made using the simplex algorithm.

11.1 Convex Programming. Kuhn—Tucker Optimality Conditions

Formulation of the conditions

It is considered the canonical form of a **convex programming:**

(P) Find the $\underline{\mathbf{x}}^* \in R^n$ ^{n} with the property that the value of the function f ($\mathbf{x}^*$) is minimal:

$$f(x^*) = \mathbf{min} f(x).$$

The minimum value is determined for any $\underline{\mathbf{x}} \in R^n$ by fulfilling the constraints:

$$q_i(\underline{x}) \leq 0, i = \overline{1, m}$$

and the non negativity conditions:

$$\underline{\mathbf{x}} \geq 0, x_j \geq 0\, j = 1, \ldots, n.$$

It is assumed that the functions f is defined in the space R^n, and q_i (q_1, q_2,..., qm) is also defined. Morevover, both of the functions are differentiable.

For each restriction $q_i(\underline{\mathbf{x}}) \underline{\leq} 0$ corresponds to a variable single nonnegative single u_i. Thus, the **Lagrangian of the** problem is constructed based on:

$$L(\underline{\mathbf{x}},\underline{\mathbf{u}}) = f(\underline{\mathbf{x}}) + \sum_{i=\overline{1,m}} u_i q_i(\underline{\mathbf{x}}),$$

where $\underline{\mathbf{u}} = \overline{u_1, u_m}$ are called **Lagrange multipliers components**.

If $\underline{\mathbf{u}} \geq 0$ ($\underline{\mathbf{u}} \in \mathbb{R}^m$), then L is a convex function and differentiable.

Assuming that the conditions of regularity Slater [11], the set of the admissible solutions is **within the relatively non-empty, the Kuhn - Tucker theorem is as follows:** the necessary and suffiecient conditions such that $\underline{\mathbf{x}}^* \in R^n$ be the minimum value of the problem (P) is to exist $\underline{\mathbf{u}}^* \in R^m$ such that the pair $(\underline{\mathbf{x}}^*, \underline{\mathbf{u}}^*)$ check the **Kuhn - Tucker optimality** relationship:

$\frac{\partial L}{\partial x_j} \geq 0, \quad j = \overline{1, n}; x_j \cdot \frac{\partial L}{\partial x_j} = 0$, respect with $x_j \geq 0$,

$\frac{\partial L}{\partial u_i} \leq 0, i = \overline{1, m}; u_i \cdot \frac{\partial L}{\partial u_i} = 0$, respect with $u_i \geq 0$.

Constrained optimization. Sufficient conditions for minimum point

Taken into consideration two continuously differentiable real-valued functions f, $g_1, g_2, \ldots, g_{m,}$ if there are vectors $\mathbf{x}_0 \in R^n$, $\lambda_0 \in R^m$, such that:

$$\nabla L(\mathbf{x_0}, \lambda_\mathbf{0}) = 0$$

and the positivity condition is respected:

$$(-1)^m \begin{bmatrix} \frac{\partial^2 L}{\partial x_1^2}(\mathbf{x}_0, \lambda_0) & \cdots & \frac{\partial^2 L}{\partial x_1 \partial x_p}(\mathbf{x}_0, \lambda_0) & \frac{\partial q_1}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial q_m}{\partial x_1}(\mathbf{x}_0) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 L}{\partial x_p \partial x_1}(\mathbf{x}_0, \lambda_0) & \cdots & \frac{\partial^2 L}{\partial x_p^2}(\mathbf{x}_0, \lambda_0) & \frac{\partial q_1}{\partial x_p}(\mathbf{x}_0) & \cdots & \frac{\partial q_m}{\partial x_p}(\mathbf{x}_0) \\ \frac{\partial q_1}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial q_1}{\partial x_p}(\mathbf{x}_0) & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial q_m}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial q_m}{\partial x_p}(\mathbf{x}_0) & 0 & \cdots & 0 \end{bmatrix} > 0$$

for $p = m + 1,\ldots,n$, then the stricttly local minimum of the function f is the *vector* $\mathbf{x_0}$

$$q_i(\mathbf{x}_0) = 0, i = \overline{1, m};$$

if $p = n$, the *bordered Hessian matrix* is obtained from the above last mentioned matrix.

11.2 Semi-defined Programming (SDP)

Semi-infinite programming problems ***(SIPs)*** **or SDPs** are components of the optimization family having an infinite number of variables or of constraints.

A SDP problem is *linear* if two conditions are fulfilled: (1) there is a linear *objective function* f; (2) there are affines constraint functions $g(t)$, for any $t \in T$.

A SDP problem is *convex* if the following two conditions are respected: (1) the objective function f is convex, (2) the constraint g (., t) is concave for all $t \in T$. In this case the considered set is *feasible.*

The function F is convex

$$F = \{\underline{\mathbf{x}} \in \mathrm{R}^{\mathrm{n}} : \mathbf{g}(\underline{\mathbf{x}}, t) \geq 0 \,\text{for all}\, t \in T\}.$$

In this case, at the same time the local and the global minimum are the same.

Semi-defined programming involves *second order cone programming* (SOCP), because SOCP constraints can be written as linear matrix inequalities.

SDP is a generalization of linear programming (LP). SDP is reduced to LP problem only in case of the diagonal matrices [8].

12 Mixed Integer Nonlinear Programming (MINLP)

The variables in the most used optimization problems are integer or discrete form. These can be modeled as MINLPs. The modeling variables can be integers (number of buildings), or binary (0 or 1) type (decision modelling). In addition, there may be continuous variables that may represent, for example, speed or torques. The nonlinearities can be found in the mathematical model of the studied process (the hysteresis of the magnetic materials), or in the decision variables. The objective function of the MILPs can be the costs minimization or the profits maximization.

13 Heuristic Methods

Heuristic Methods (HM) are faster than traditionally optimization algorithms

Classification of the metaheuristic algorithms (MA) for local and global search:

(a) The general structure of the local optimization algorithms

(b) Deterministic for local search. There are two mainly types of searching algorithms: Pattern, and Nelder Mead, respectively;
(c) Random local search: Matyas, and Solis-Wets, respectively;
(d) MA for global search: Local search with restarting; Local iterated search, and Simulated Annealing.

The use of the HM in MINLP

The HM is recommended to be used instead of the large computational burning time of the deterministic algorithms. Many times, these methods cannot guarantee that the solution can be found. However, HM are much faster than the conventional algorithms and simplify the optimization problems. Many of the HM mimic the known continuous methods.

Recently, there are many developed HM aimed to solve the practical problems of MINLP. These HM include procedure for rounding [12] and an attempt to generalize a method SQP [13, 14] the method of gradient descent [15–18], penalty function method [19, 20] and adaptive random search [21].

There are two mainly factors to succeed a HM: speed and reliability to find the suboptimal solution. The term of the suboptimal solution is used for any solution very closed to the optimal solution. This type of solution is very attractive in practice. From the NLP relaxation method a satisfable solution can be determined. The simplest HM are based on rounding. The method of solving the MINLP problem is based on the NLP technique combined with the relaxation of all the integer constraints. Finally, the value of the solution is rounded to a near integer point. There are two disadvantages of using thi method: 1. the value of the solution cannot be feasible, and 2. the obtained value for the objective could be too far from the value obtained in the deterministic case. In the paper [12] a “smart rounding” procedure was developed and a discrete line layout scheme was proposed. Recently, the developed smooth landings methods becomes very attractive. In the paper [15], for example, the authors generalize the ablest method of descending with the MINLP problem line search. They reformulate the optimization problem by using an inverse barrier function to obtain the MINLP problem without constraints.

Local optimization

Local optimization methods search the optimal solution x* in the vicinity of the studied point V (x*), i.e.

$$f\left(x^*\right) \leq f(x+e), \text{ with } \varepsilon > 0, \text{ sufficiently small.}$$

Remark: the initial approximation of the optimal solution should be known

The solution can be found in discrete space or in continuous space.

In the first case, the neighbourhood of an element is a finite set that can be thoroughly explored.

In the second case, there are two derived branches, function of the derivability property of the objective function:

(a) in case that the derivability feature is fulfilled by the objective function, two basic methods can be used: gradient, and Newton.
(b) in case that the derivability feature is not fulfilled by the objective function, the direct search methods (e.g. Nelder Mead), or the methods based on small random perturbations are recommended.

Global optimization:

- identifying the global optimality of a function: for a minimization problem, the following property should be attained $f(x*) \leq f(x)$, x^* is considered the global minimum value if the above mentioned property is true for all x;
- by using local search methods, if there are local optimimum points, the minimum value of the objective function can be blocked into the local optimum point.

14 Genetic Algorithms (GAs)

Genetic algorithms are some of the most popular evolutionary computing strategies. Among other applications, these have been successfully used for difficult optimization problems, with multimodal, discontinuous and non-differential objective functions [22]. Traditional optimization algorithms often fail in such cases.

Algorithm

```
Generate the initial population: $N$ random individuals
   **while** stop criterion not fulfilled,
      Selection stage of the individuals fittest for the next stage
          Reproduction stage: by applying crossover and mutation operators, new
individuals are created
          Recombination stage: new population are created
        **end while**
```

15 Ant Colony Optimization

Inspiration from the intelligence of the colonies led to some very successful optimization algorithms.

– Ant colony optimization—a way to solve optimization problems based on the way ants communicate indirectly with each other.

Ant colony optimization algorithms.
Ants are agents that [23]:

– move along the nodes in a graph
– ants choose where to go based on the power of pheromones.

The path of an ant is a specific solution of the candidate.

When an ant has finished a solution, the pheromone is placed in its path, depending on the quality of the solution.

This pheromone pathway affects the behavior of other ants through "stigmergy"

Optimization methods

1. Differential evolution (DE)

The general algorithm from the GA method has been applied in the case of DE. After population initialisation, the generation of descendant population through crossover and mutation is obtained (the first step is the reproduction). The next step is the selection of the best element in the population. By using DE method, a new candidate is building. As evolutionary strategy, the multi-modal test function has been used.

Taking 2 elements in the population having the size n = 1 for each element, considering 50 generations, choosing [−0,5 0] the domain of the function to be minimized with crossover probability of 0.55, the Differential evolution algorithm provides the minimum function value (Fig. 4).

2. Particle cluster optimization
3. Clonal selection
4. Ant colony-type optimization
5. Hill climbing and simulated riding

Hill climbing algorithm

Ideas similar to the ascending gradient method if the objective function is maximized

– They start from a randomly selected point in the search space p_0
– Current point $pc \leftarrow p_0$
– One or more neighboring points are generated, pv
– If the objective function in a neighboring point is better than the current one, then $pc \leftarrow pv$;
– Choose the first best neighbor (*Greedy*, *Simple HC*)
– The best neighbor is chosen (Steepest *Ascent HC*)

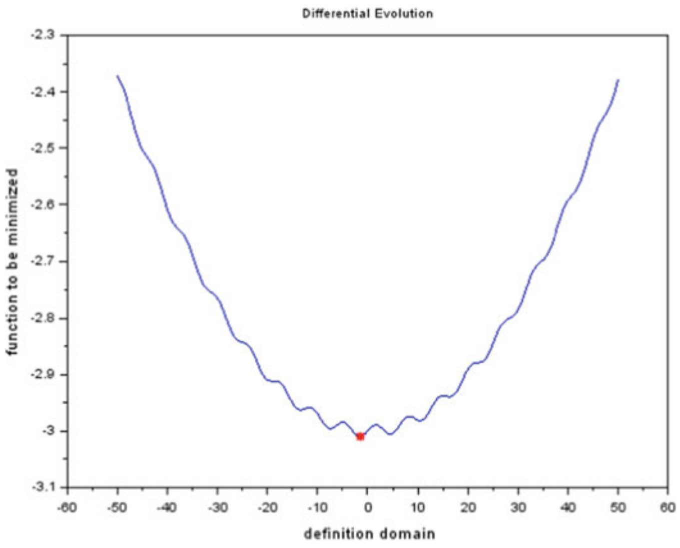


Fig. 4 Graphical representation of the function and elements of the population. Displaying the value of the best element in the population [10]

16 Simulated Annealing (SA)

Stochastic algorithm inspired by metallurgical quenching. Heating and then controlled cooling of a material increases the size of the crystals and reduces defects.

SA algorithm

Suppose a minimization problem, with the objective function E

- If the neighbor is better ($E_v < Ec$), then $pc \leftarrow pv$
- If the current state is better than the next one, the neighbor ($E_v > Ec$), then ΔE is caluclated;
- The difference of the objective functions is calculated: $\Delta E = Ev - Ec$
- It is considered the current temperature T, high at the beginning and decreasing in time.
- Probability of accepting the transition to the lower state is: $P = \exp(-\Delta E/T)$

Example: The function to optimize is the Rastrigin function by using SA method.

In Figs. 5–6, by taking two parameters of Rastrigin function, A = 1, n = 2, the dynamic process of the optimization, and the contour plot of the Rastrigin function during SA optimization are shown (Fig. 5).

The dynamic process of the optimization by using SA.

The intial random initialisation:

×0 = −0.0060798 0.0412484

y0 = 0.0358653.

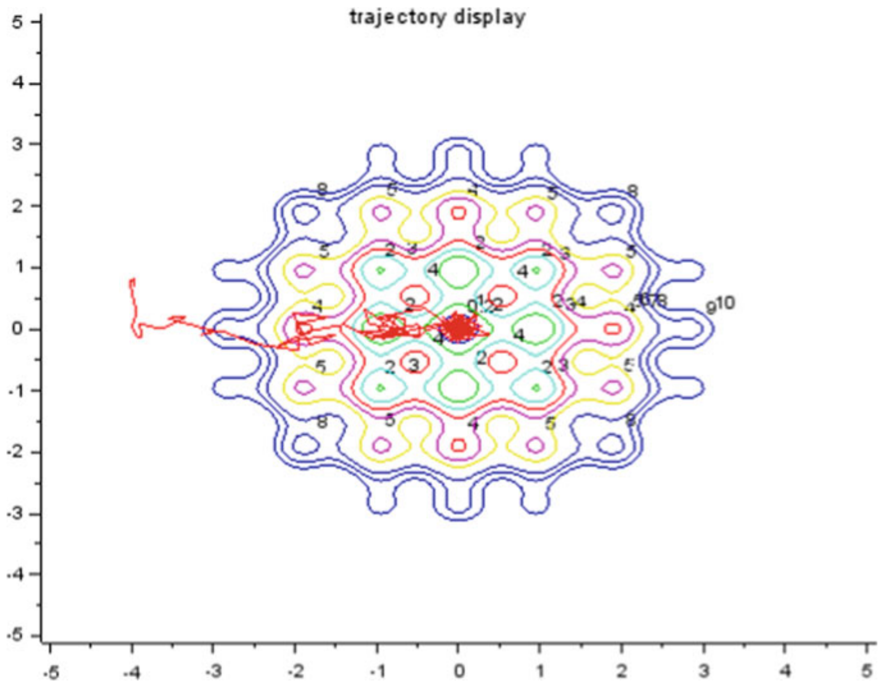


Fig. 5 Contour plot of the Rastrigin function during SA optimization and the dynamic process of the optimization process evolution (red line) [10]

After 2000 number of iterations, by using SA method of optimization (Fig. 7), the coordinates of minimum function obtained at (0.0023824 −0.0028886), corresponding value by Rastrigin function is 0.0002908 (Fig. 6).

By choose a large number of iterations more than 2000), the function has a global minimum at $\mathbf{x} = (x_1, x_2) = (0,0)$, and the value $f(\mathbf{x}) = 0$.

17 Particle Swarm Optimization (PSO)

Swarm intelligence [24] domain that includes intelligent techniques based on the collective behavior of systems with self-organization and without centralized control

Model of the particle assembly

The Particle Swarm Optimization (PSO) [4, 25] technique was proposed by James Kennedy and Russell Eberhart for nonlinear function optimization (1995).

The source of inspiration: the behavior of bird flocks, fish banks, and bees swarms, these are assimilated to a set of particles that move in the search space to identify the optimum [25].

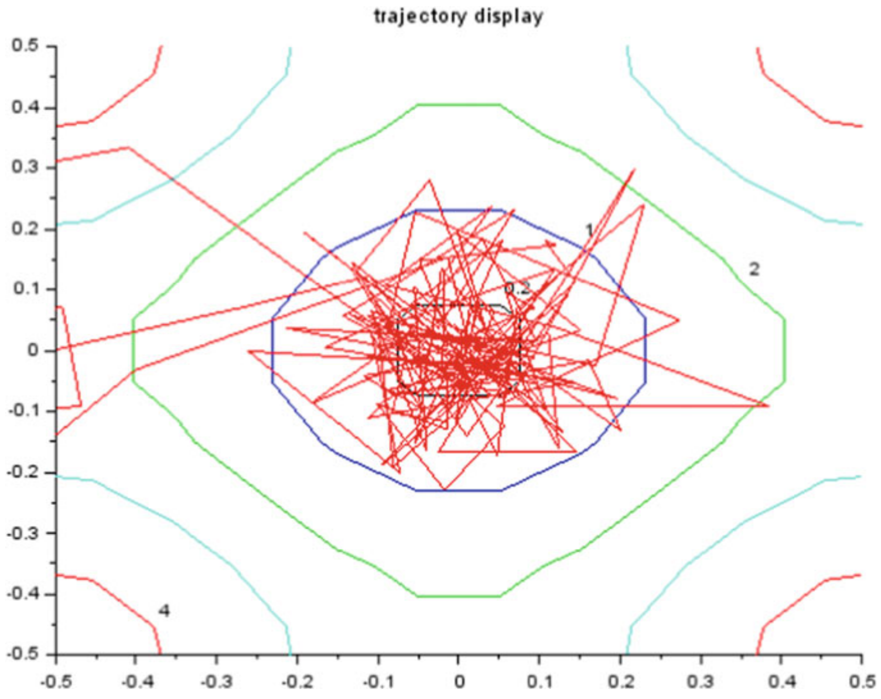


Fig. 6 The dynamic process of optimization (red line) in the specified [−0.5, 0.5] interval

It uses a set of particles whose positions are in the field of objective function and which are modified by an iterative process.

At each iteration, the new position of each particle is determined according to: the current position of the particle.

The best position encountered by the particle is the *local best solution*

Best position found by the whole is the *global best*.

PSO Algorithm:

```
    Initialization of particle positions

REPEAT

    speed calculation
    update positions

UNTIL < stop condition>
```

The PSO algorithm is applied to the Rastrigin function.

The same significance of the optimization process (for Rastrigin function) results are obtained as in SA method, discussed above (Figs. 8, 9, 10).

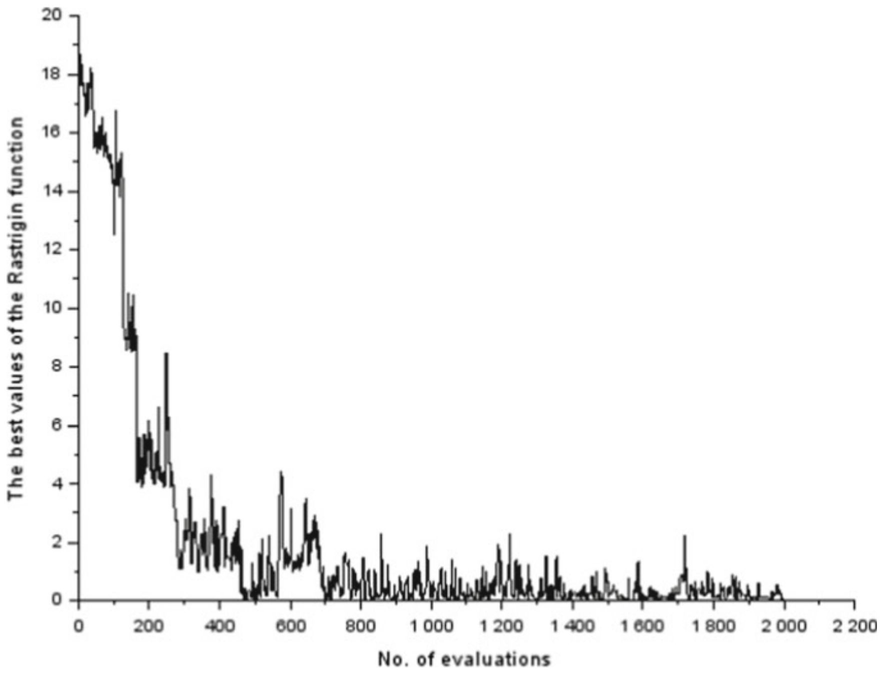


Fig. 7 Optimization process of Rastrigin function of two variables

18 The Model of the Bee Swarm

The source of inspiration: the intelligent behavior of bees in the process of identifying food sources (nectar)

It uses a population of "bees" consisting of three categories: Bees "allocated" to a food source (workers)

Observer bees
Scorpion bees

The model of the bee swarm

Step 1: Initialize the locations in the search space where the worker bees are placed

Step 2: How long the continuation condition is satisfied: The working bees transmit information about the quality of the location where they are to the observer bees; each bee observer selects a location; the selection is based on a probability distribution determined by the values of the associated scores;

Working bees explore the vicinity of their location and move to another neighboring location if it is better; if a worker bee does not discover a better configuration in a limited number of steps then it is relocated to a position determined by a bee researcher. Scorpion bees randomly change their position.

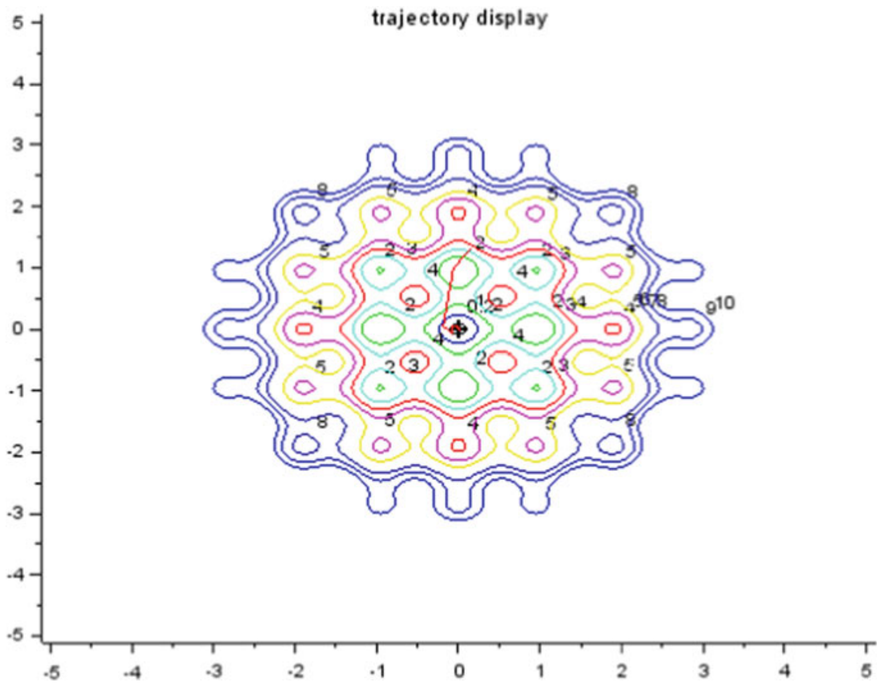


Fig. 8 Contour plot of the Rastrigin function during PSO optimization and the dynamic process of the optimization process evolution (red line) [10]

19 The Firefly Model

Firefly algorithm [26]

The source of inspiration: interactions between firefly based on the light signals they emit.

Main idea of implementation

- Each element of the population corresponds to the position of a firefly
- Each degree of firefly is associated with a degree of brightness (correlated with the value of the objective function associated with the corresponding element in the population).
- The movement of the firefly is guided both by the distance between their positions and the value of the brightness

– Position x_i is shifted to position x_j (if x_j has higher brightness) using the specific parameters (alpha, beta and gamma are control parameters and epsilon is a random value with normal distribution)

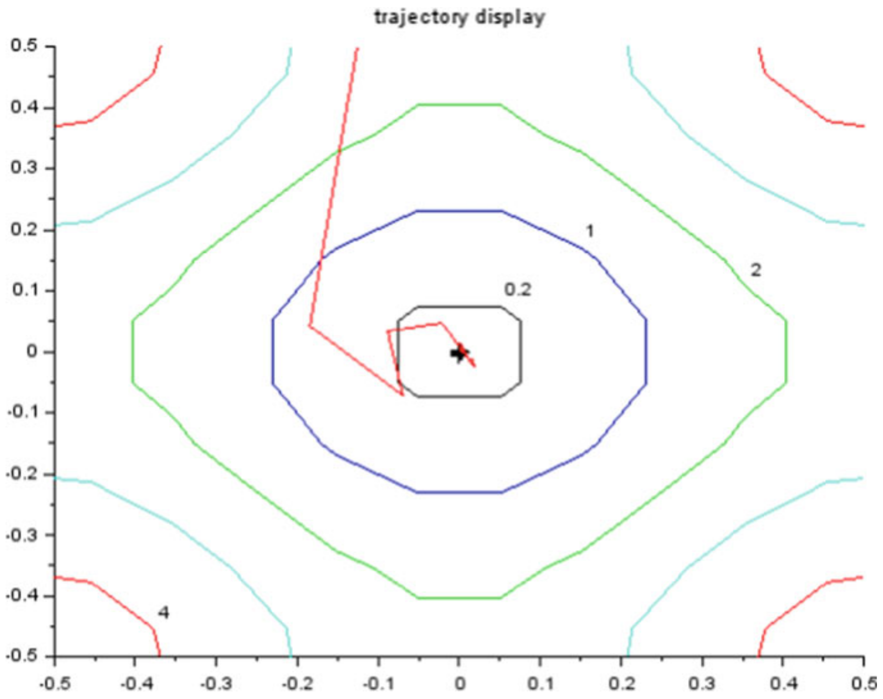


Fig. 9 The PSO dynamic process of optimization (red line) in the specified [-0.5, 0.5] interval [10]

Conclusions:

– Differential evolution has the same genetic operators as classical evolutionary algorithms, but their order and mode of operation is different
– There are many other optimization algorithms inspired by nature
– Particle cluster optimization is inspired by the behavior of birds
– The clonal selection is inspired by the immune system
– Ants colony-type optimization is inspired by the way they search for food
– The simulated ride is inspired by metallurgy.

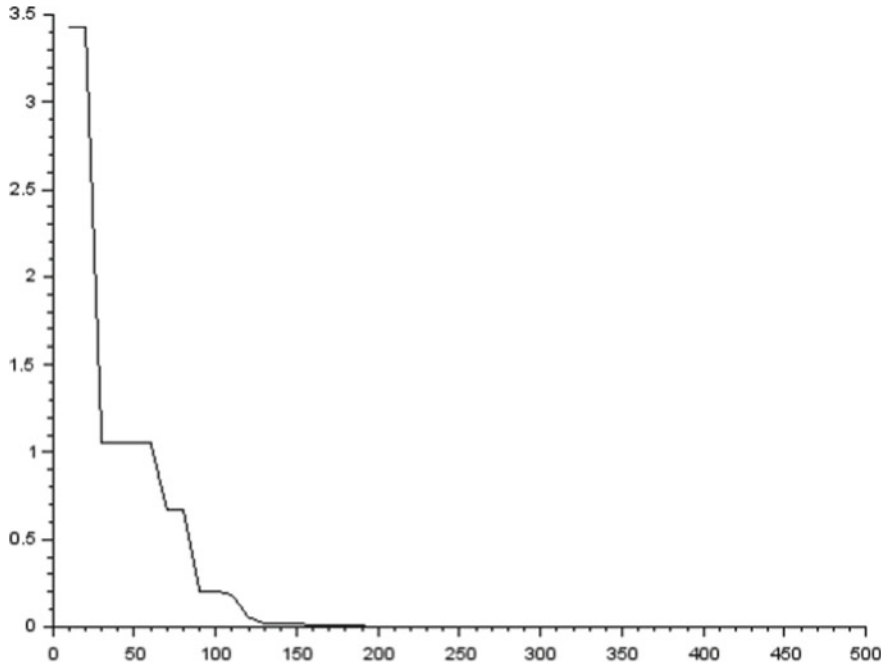


Fig. 10 Particle Swarm Optimization process of Rastrigin function of two variables. Convergence of the optimization process [10]

20 Very Large Scale Neighborhood Search

The main cause of generating complicated difficulties in solving real optimization problems is the **dimension**: such a problem complicated is **too high**. In mathematical programming the size of the problem is relatively, depending on the following parameters:

- Number of variables and the number of constraints;
  - Complexity of the function expressions and the constraints;
  - Performance of the numerical calculus equipment: hardware and software

  In principle, methods for solving large scale problem divides into:

- Direct Methods: they specializes a general procedure to the specifics of a particular class of optimization problems.

Let take into consideration he case of a linear program with upperr bounded variable:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j = b_i \quad , i = \overline{1, .m} \\ 0 \le x_j \le u_j \quad , j = \overline{1, .n} \\ (\max) f = \sum_{j=1}^{n} c_j x_j \end{cases}$$

The classical solution method involves the transformation of the upper conditions in equality:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j = b_i \quad , i = \overline{1, .m} \\ x_j + x_{n+j} = u_j, \quad j = \overline{1, .n} \\ x_j \ge 0 \quad , \\ (\max) f = \sum_{j=1}^{n} c_j x_j \end{cases}$$

The result consists of the program with m + n restriction and 2n variables of those bases were of the order m + n matrices.

- *Indirect methods*, based on the **decomposition large** problem into smaller, **interconnected** sub-problems. Subproblems may be solved **independently** (and if it is even possible **at the same time**), but the fact that the subproblems interacts involves existence of a **coordination** mechanism. Thus, solving the original problem is made in two levels.

    - at the First level—lower—subproblems are solved and the results are communicated:
    - Second level—higher—which analyzes the results and transmit the new parameters to the lower level.

At level one there is a resumption of calculations (re-optimization); new results sent to the upper level for analysing, so on.

Important is the fact that this iterative process converges.

The nonlinear-optimization (NO) is divided into: Unconstrained (UNO) and Constrained (CNO).

The algorithms used to solve *Nonlinear programming problems in Very Large Scale Systems* are as follows:

- Augmented Lagrangian Methods
- Sequential Quadratic Programming
- Feasible Sequential Quadratic Programming
- Reduced-Gradient Methods

Constrained Nonlinear Optimization

Constrained nonlinear optimization problems can be solved by using one of the following algorithms:

- **Interior-point (IP).** The method is used for large-scale problems.
  The IP algorithm estimates the Hessian matrices by using:
  - BFGS (dense)—Limited-memory Broyden Fletcher Goldfarb Shanno
  - Limited memory BFGS
  - Hessian-multiply function
  - Current Hessian (sparse or dense)
  - Finite difference of gradients. This case does not require the knowledge of sparsity structure
- **SQP** algorithm.
  This algorithm is specified to general nonlinear optimization problem.
- The **trust-region reflective (TRRA)** algorithm can be succesfull used in case of the bound constrained problems or linear equals only.

  For the TRRA, the Hessian matrices can be obtained by using:

- Finite difference of gradients, sparsity structure of the Hessian
- Current Hessian (sparse or dense)
- Hessian-multiply function

  Both methods, the IP and TRRA use lower memory usage.

21 Security-Constrained Unit Commitment (SCUC)

The ***Unit Commitment*** (UC) problem is based on the optimization algorithms and it used in power sytems. The idea is to coordinate a set of the electrical generators such that the energy demand with minimal costs is attained, or maximize the profits from energy production.

Coordination of generating units is a difficult task for several reasons:

– there are a large number of units;
– the costs of the energy production from different type of generators can varying. At the same time, different constraints conditions for each generator can be met (due to the different technologies of energy production);
– the generators are spread over a large area into a country. For this reason, the response capacity of the electrical network should be considered. The complex power flow data should be determined to assure the load demand.

SCUC commitment consists of two components, system security and economic dispatching. The objective of the problem is focused exclusively on the economic dispatch of generators with tender segments, without loading costs, starting costs and other costs incurred during operation. The lowest cost is desired depending on the system operators.

22 Conclusion

The complexity of the mathematical problems into the power systems is very high. Different methods should be used to find the adequate solution.

The class of the numerical methods is very usefull to increase the speed of finding the solution with high precision.

To facilitate the development of numerical problems, the need to establish procedures is stringent, this leads to algorithms development. However, the solution involves the following steps; modeling, choice of numerical methods, operation, results, and interpretation of the obtained results. In power systems, power is known rather than current; thus, the equations resulting in power term are more appropriately. This type of equations (power flow equations), are nonlinears. The solution of them is obtained by using iterative techniques from the numerical algorithms.

The authors of this chapter made an incursion in optimization problems and their solution through the numerical methods. In order to implement the optimization algorithms different programming languages have been used, as Matlab [27], and Scilab [10] Starting with the formulation of the optimization problem, the authors gives different solutions to the exposed topics: optimization of a real variable functions, the extremes of the functions defined over an interval, extremes of functions for which the derivative does not exist, the method of small pertutbations (variations), extremes of the multivariable functions, the minimum of a function of two variables with constraints, the Lagrange multiplier method for determining the minimum of a constraint function, types of optimizations, unconstrained optimization problems, nonlinear optimization without constraints, optimization problems with linear constraints and nonlinear objective function, the problems of convex programming, the problems of separable programming, the problems of non-convex programming, optimization methods with constraints, Mixed Integer Nonlinear Programming (MINLP), Very Large Scale Neighborhood Search, Security-Constrained Unit Commitment (SCUC).

Advanced optimization algorithms are introduced and the simulation results are provided (Genetic Algorithms, Particle Swarm Optimization, and Simulated Annealing).

References

1. Rosu E, Nichita C, Bivol I, Gaiceanu M (1999) Optimizarea energetica a sistemelor de conversie electromecanica (Energetic Optimization of the Electromechanical Conversion Systems). Technical Press

2. Gaiceanu M (2009) Optimal control of the electric drive systems. Galati Univerity Press (University)
3. Metode de optimizare numerica, Ion Necoara, acse.pub.ro
4. https://press.princeton.edu/sites/default/files/inline-files/Absil_Chap7.pdf.Last Accessed 2019
5. http://www.numerical.rl.ac.uk/people/nimg/oupartc/lectures/raphael/lectures/lec6slides.pdf. Last accessed 2019
6. Gould N (2006) An introduction to algorithms for continuous optimization. Oxford University Computing Laboratory and Rutherford Appleton Laboratory Copyright 2006 by Nicholas Ian Mark Gould
7. Botan C (2007) Optimization techniques, Politehnium
8. http://neos-guide.org. Last accessed 2019
9. www.math.ubc.ca. Last accessed 2019
10. http://www.scilab.org/. Last accessed 2019
11. Bolte J, Hochart A, Pauwels E (2018) Qualification conditions in semialgebraic programming. SIAM J Optim 28(2), 1867–1891. https://doi.org/10.1137/16m1133889
12. Olsen GR, Vanderplaats GN (1989) Method for nonlinear optimization with discrete design variables. AIAA J 27:1584–1589
13. Cha JZ, Mayne RW (1989) Optimization with discrete variables via recursive quadratic programming: Part 1—concepts and definitions. Trans ASME, J Mech, Transm, Autom Des 111:124–129
14. Cha JZ, Mayne RW (1989) Optimization with discrete variables via recursive quadratic programming: Part 2—algorithms and results. Trans ASME, J Mech, Transm, Autom Des 111:130–136
15. Amir HM, Hasegawa T (1989) Nonlinear mixed-discrete structural optimization. J Struct Eng 115:626–646
16. Bremicker M, Papalambros PY, Loh HT (1990) Solution of mixed—discrete structural optimization p roblems with a new sequential linearization algorithm. Comput Struct 37:451–461
17. Loh Han Tong, Papalambros PY (1991) "A sequential linearization approach for solving mixed-discrete nonlinear design optimization problems" Transactions of the ASME. J Mech Des 113:325–334
18. Loh Han Tong, Papalambros PY (1991) "Computational Implementation and tests of a sequential linearization algorithm for mixed-discrete nonlinear design problems" Transactions of the ASME. J Mech Des 113:335–345
19. Davydov EG, Sigal IKh (1972) Application of the penalty function method in integer programming problems. Eng Cybern 10:21–24
20. Li Han-Lin (1992) An approximate method for local optima for nonlinear mixed integer programming problems. Comput Oper Res 19:435–444
21. Salcedo RL (1992) Solving nonconvex nonlinear programming and mixed – integer nonlinear programming problems with adaptive random search. Ind Eng Chem Res 31:262–273
22. Houck CR, Joines JA, Kay MG (1996) Utilizing lamarckian evolution and the baldwin effect in hybrid genetic algorithms. meta-heuristic research and applications group. NCSU-IE Technical Report 96-01. Department of Industrial Engineering, North Carolina State University
23. Chandrasekhar A, Gordon DM, Navlakha S (2018) A distributed algorithm to maintain and repair the trail networks of arboreal ants. Sci Rep. 8(1), 9297. Published 2018 Jun 18. https://doi.org/10.1038/s41598-018-27160-3
24. Beni G, Wang J (1989) Swarm intelligence in cellular robotics systems. In: Proceeding of NATO advanced workshop on robots and biological system
25. Kennedy J, Eberhart R (n.d.) Particle swarm optimization. In: Proceedings of ICNN'95—international conference on neural networks. https://doi.org/10.1109/icnn.1995.488968
26. Karaboga D, Basturk B (n.d.) Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. Found Fuzzy Log Soft Comput 789–798. https://doi.org/10.1007/978-3-540-72950-1_77
27. https://www.mathworks.com. Last accessed 2019

28. http://www.particleswarm.info/. Last accessed 2019
29. Yang XS (2009) Firefly algorithms for multimodal optimization. Lect Notes Comput Sci 169–178 https://doi.org/10.1007/978-3-642-04944-6_14