Optimization Lab Sessions: Major Features and Applications of IBM CPLEX

Juan Manuel Garcia-López, Kseniia Ilchenko and Olga Nazarenko

Abstract The Optimization and Decision Support Systems for Supply Chains (Odss4SC) summer school has taken place in Portalegre, Portugal, for three editions (2012, 2013 and 2014) within the Erasmus Intensive Programme. The audience is formed by M.Sc./Ph.D. students on Engineering and Logistics, coming from multiple countries. Some of the goals of this school include studying "green" logistics aspects and using optimization models for solving specific problems, with a practical approach supported by computational sessions (Optimization Labs). During these Optimization Lab Sessions the participants were introduced to the IBM ILOG CPLEX Optimization Studio to develop optimization projects. Participants were acquainted with the concepts, architecture, components, processes, and procedures necessary to build optimization models. Topics covered in these sessions included working with Optimization Programming Language (OPL) to write mathematical programming (MP) models, linking to data sources, flow control and performance tuning of CPLEX optimizer. After the Lab sessions, the participants had acquired a general idea about Optimization engines and how to use them for their Supply Chain Optimization problems.

Keywords Optimization · Mathematical programming · Constraint programming · IBM ILOG CPLEX

J.M. Garcia-López (✉)
IBM—ILOG Decision Optimization, C/Santa Hortensia, 26-28, 28002 Madrid, Spain
e-mail: jm.garcia@es.ibm.com

K. Ilchenko · O. Nazarenko
Kyiv Polytechnic Institute, National Technical University of Ukraine, 37 Peremohy Ave., Kyiv, Ukraine
e-mail: ilchenko@wdc.org.ua

O. Nazarenko
e-mail: olga.nazarenko@ukr.net

A.P. Barbosa-Póvoa et al. (eds.), *Optimization and Decision Support Systems for Supply Chains*, Lecture Notes in Logistics,
DOI 10.1007/978-3-319-42421-7_10

1 Introduction

An introductory session about IBM CPLEX package was delivered at each edition of the program. IBM ILOG CPLEX Optimization Studio speeds development and deployment of optimization models, combining leading solver engines with a tightly Integrated Development Environment (IDE) and modelling language. Techniques used include linear, quadratic and mixed integer programming. IBM ILOG CPLEX Optimization Studio ensures reliable development and maintenance using a transparent modelling language and intuitive tools for model testing, profiling, and tuning.

The session for each edition had an approximate duration of 8 h, with a mix of presentations delivered by the instructor and guided exercises ("Labs").

The goal aimed with these CPLEX sessions was to introduce the students to real, first-class software for modeling and solving Supply Chain problems. It was not intended as a training session in the product, as the duration was obviously insufficient for achieving an effective proficiency level.

The students were evaluated after the session with a short quiz. In Sect. 2 is an excerpt of the questions presented to the students, with enriched answers that can be also useful for researchers and practitioners. The students were asked to answer 'True' or 'False', but long answers are provided here as explanations. Notions and concepts are revisited, and the following text can support further computational Optimization developments on IBM CPLEX.

Fourteen questions were asked in the lab session of the first course edition (2012). In subsequent editions of the course (2013, 2014) twenty questions were asked. The questions are selected and presented here in two separated sets: one dedicated to topics in Mathematical Programming and Constraint Programming (Sect. 2); other addressing topics and applications of IBM CPLEX (Sect. 3).

2 Topics in Mathematical Programming and Constraint Programming

This section based on fundamental works that present main approaches and methods in Mathematical Programming and Constraint Programming. In accordance to this, the materials are presented as overall results of theoretical investigations of sources (Bradley et al. 1977; Imamoto and Tang 2008; Rossi et al. 2006; Williams 1999).

1. ***Constraint programming (CP) is a branch of mathematical programming (MP).***

False: CP has its origins in computer science. It uses constructive search algorithms to find feasible solutions. MP has its origins in Operations Research, and

uses algorithms such as simplex and branch-and-bound, with a focus on finding optimal solutions.

The word "programming" causes a common misconception with inclusion of CP to MP. CP can be described as a "programming" in the sense of a computer program: the statements invoke procedures, and control is passed from one statement to another. Moreover, MP only declares algorithms without real programming. The word "programming" was implemented to MP by George Dantzig's application of linear programming to logistics in the military sphere.

A CP optimization model has the same structure as a MP model: a set of decision variables, an objective function to maximize or minimize, and a set of constraints. Still CP and MP came from different branches of science.

2. ***A feasible solution can be, but is not guaranteed to be, an optimal solution***.

True: For a solution to be optimal, it is by definition also feasible.

Feasible solution is an element of the feasible region. The feasible region is the set of all possible solutions of an optimization problem. Moreover, the feasible solution best matched to the problem is called optimal.

For example, there is the set of marks {1, 2, 3, 4, 5}, where the higher value correlates with better marks. Thus, the optimal (in case, if the student wants to be the best in the class) is 5, but all the marks are feasible. The illustration of feasible and optimal solutions for abstract linear programming problem with maximization of objective function is presented on Fig. 1.

3. ***The following constraint is valid for a linear programming problem, where x and y are variables, z is a data item: $2x + 3y \leq z^2$.***

True: As long as z is a data item, then z^2 is also a data item.

LP concept is based on four main components: decision variables (the values to be determined), objective function (shows how the decision variables affect the cost or value to be optimized), constraints and data. Constraints represent how the decision variables use resources, which are available in limited quantities. Data quantifies the relationships represented in the objective function and the constraints.

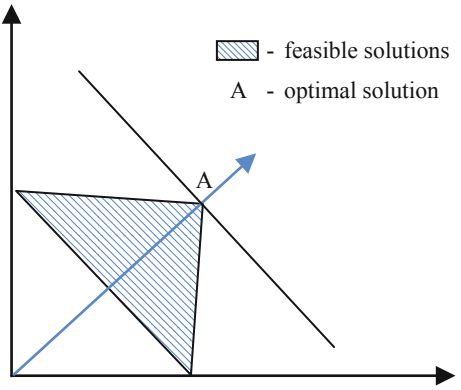


Fig. 1 Feasible and optimal solutions

In a linear program, the objective function and the constraints are linear relationships.

Thus, it is possible to present the CP constraint as $a_1x_1 + a_2x_2 \leq (\geq \text{ or } =)\ z$, where a_1 and a_2—some parameters, x_1, x_2—variables, and z is data.

To sum up, the presented formula is a linear constraint with variables at the left side and data at the right side.

4. ***Hard constraints can be converted to soft constraints to help resolve infeasibilities***.

True: Removing hard constraints and weighting them in the objective function helps in dealing with infeasibilities. However, this technique might yield a problem solution meaningless, if the removed constraint cannot be considered "optional" by the user.

Each inequality that defines the feasible region prohibits certain assignments to the structure. Such constraints are known as hard constraints. If there are many constraints imposed on problem variables, it could be impossible to satisfy them all, such problems are called over-constrained. Therefore, a soft constraint is used for finding some additional decisions. Soft constraints give more scope for relaxing the restrictions and attaching concepts such as cost and priorities to the constraints depending on the preference of the constraint being satisfied in the Constraint Satisfaction Problem. The concept of soft constraints is that constraints are ranked according to their importance and the form that satisfies the maximum high-ranking constraints is considered the optimal.

Hard constraints are those, which we definitely want to be true. These might relate to the successful assembly of a mechanism. Soft constraint are those we would like to be true—but not at the expense of the others. These might say that a mechanism must follow a given path. There is no point in trying to match every point exactly if this can only be done by breaking the assembly of the links.

5. ***An unbounded variable will lead to an objective that can be minimized or maximized to infinity***.

False: An unbounded model will lead to an objective that can be minimized or maximized to infinity. Whether an unbounded model is determined by the constraint definitions.

6. ***A piecewise linear function can be used to approximate convex nonlinear functions***.

True: This technique should not be used for nonconvex nonlinear functions.

A piecewise linear function is a function composed of straight-line sections. A convex function is a continuous function whose value at the midpoint of every interval in its domain does not exceed the arithmetic mean of its values at the ends of the interval. The simplest example of a convex function is an affine function (the sum of a linear form and a constant): $f = a^Tx + b$.

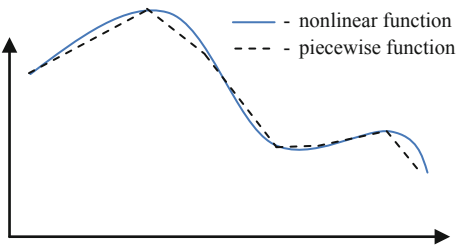


Fig. 2 An approximation of nonlinear function

Approximation of nonlinear function by piecewise linear is a common practice that simplifies the solving. Piecewise functions allow the representation of functions with any accuracy by simply increasing the number of segments until the desired accuracy is met. The approximation of nonlinear function by piecewise linear is shown on Fig. 2.

7. ***Mixed-integer programming is often used for investment planning***.

True: In MIP, for example, the yes/no decisions required for investment planning are represented by binary decision variables. A mixed-integer programming (MIP) problem is such problem where some of the decision variables are constrained to be integer values at the optimal solution.

The typical problem in investment planning can be formulated in such way: an investor needs to plan the budget and the portfolio of the start-up projects for 3 years and has picked up 3 projects he wants to invest in. This problem can be presented in binary way because there are only two decisions: to invest or not to invest in the start-ups. The binary decision variable is 1 if the project is invested in the year, and 0 if there will be no investments for concrete project in concrete year.

Not only investment planning problems are often solved by MIP. The processes of labor planning and logistics are also very popular economic applications for this method.

8. ***Rounding is an efficient method for creating integer solutions to relaxed mixed-integer problems***.

False: Rounding is not the best way to create integer solution to relaxed MIP. Even more, it could give wrong result.

The optimal integer solution can be not obtained by rounding. The closest point to the optimal solution can be even not feasible. Also, the nearest feasible integer point can be far away from the optimal integer point (Fig. 3). Thus, it is not sufficient to simply round the solutions.

9. ***It is important always to use integer variables when a model involves the production of whole items***.

False: While it might be important to use integer variables to plan production of high-value whole items, such as airplanes, it is usually better to use continuous

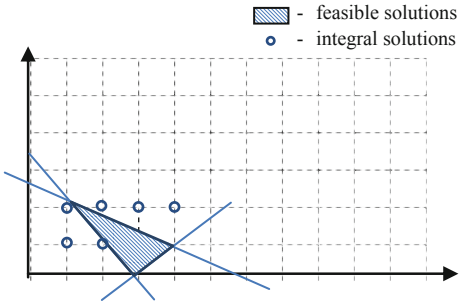


Fig. 3 Rounding solution in relaxed MIP

variables to plan for a large number of small items, for example the production of tennis balls, but, in the end, rounding can pose some (small) difficulties.

10. ***Introducing logical constraints in an LP model results in a Mixed-Integer model***.

True. MIP is an optimization method that combines continuous and discrete variables. Involving the logical constraints in an LP model means the usage of binary constraints. Such type of constraints is integer. Therefore, it is possible to speak about a transition to MIP.

11. ***Constraint Programming involves a fine-grained enumeration of time in order to determine start and end times of tasks to be scheduled***.

True. Constraint programming is a type of programming where the relations between variables are described by constraints which must be satisfied at the same time.

A fine-grained description of a system is a detailed model of it. At the same way, the fine-grained enumeration of time helps to determine the start and the end of the process.

12. ***In order to solve a CP problem, you must specify a search phase***.

False: Specifying a search phase is optional, but default search strategy works well in many occasions.

It is necessary to mention that a search phase is a way to guide search types in CP. A search phase defines instantiation strategies to help search the algorithm. A search phase also determines mono-criterion or multi-criteria type of problem. The first one consists of an array of integers to instantiate (or fix), and a variable chooser that defines how the next variable to instantiate is chosen, and a value chooser that defines how values are chosen when variables are instantiated. The second one can have either two different search phases or several decision variables.

CPLEX Studio gives the possibility to change the search algorithms from the IDE settings editor or by changing a CP parameter. In case of missing the specification of search phase, the default algorithm will work. Therefore, the specification of a search phase is desirable but not required.

3 Topics in IBM CPLEX

This section has more practical features, in comparison to the previous one; thus, the answers mainly based on IBM Tutorials.[1,2,3,4,5,6]

13. ***Once an OPL project has been created in CPLEX Studio, it can be solved only from inside CPLEX Studio***.

False: The models created by CPLEX Optimization Studio can be integrated into external applications written in Java, C++, .NET or any other language that is call-compatible with them, with the help of application programming interfaces (APIs). Thus, there is no necessity to rewrite the model in other programming languages. This capability enables to include optimization components to other applications.

14. ***When solving an LP model in CPLEX Studio, the user must choose which LP optimizer (for example, Simplex, Dual-Simplex, or Barrier) to use***.

False: The user is allowed to choose the LP optimizer, but by default CPLEX Optimizer will automatically choose a particular LP optimizer based on the problem structure.

Default settings will result in a call to an optimizer that is appropriate to the class of problem you are solving. However, it is possible to choose a different optimizer for special purposes. An LP problem can be solved with usage of CPLEX optimizers: dual simplex, primal simplex, barrier, and even the network optimizer (if the problem can be described through network structure). The variants of optimizers for each type of problem are presented in Table 1.

---

[1]IBM ILOG CPLEX Optimization Studio OPL Language User's Manual. © Copyright IBM Corp. 1987, 2015. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/opl_languser.pdf?lang=en.

[2]IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. © Copyright IBM Corporation 1987, 2015. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/usrcplex.pdf?lang=en.

[3]IBM ILOG CPLEX Optimization Studio CP Optimizer User's Manual. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/usrcpoptimizer.pdf?lang=en.

[4]IBM ILOG CPLEX Optimization Studio V12.6.2 documentation. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html.

[5]IBM ILOG CPLEX Optimization Studio. http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

[6]IBM Decision Optimization Center V3.8 documentation. http://www-01.ibm.com/support/knowledgecenter/SSQVNT_3.8.0/ilog.odms.ide.odm.enterprise.help/ODME/ODMEhome.html.

Table 1 Comparison the optimizers for different types of problems

Type of a problem/optimizer	Dual	Primal	Mixed integer	Network	Barrier
LP	+	+		+ (for some problems)	+
QP	+	+		+ (for some problems)	+
MIP			+		
Network				+	
QCP					+

Therefore, it is possible to solve a LP problem without choosing the optimizer

15. ***The Network Optimizer will always improve the solution time for a model with a network structure***.

False: When faced with a difficult network model, it is a good idea to try the Network Optimizer, although an improvement is not guaranteed.

The ILOG CPLEX Network Optimizer recognizes a special class of linear programming problems with network structure. It uses network algorithms on that part of the problem to find a solution from which it then constructs an advanced basis for the rest of the problem. Each problem has different structure and level of difficulty, thus, there is a high possibility that simple LP models will be solved faster by other optimizers.

16. ***A good way to deal with uncertainty is to solve and compare several scenarios of the same basic model***.

True: Scenario planning is the development of number of the ways for solving the problem, which deals with uncertainty.

Scenario planning involves the development of scenarios that capture a range of plausible future conditions and particularly appropriate in complex situations where uncertainties about future conditions and the effectiveness of management actions are uncontrollable. While there are key steps in the process, there is no single established methodology for conducting scenario planning, or even discrete types of scenario planning approaches. But, in general, scenario planning is the development of number of the ways for solving the problem.

IBM ILOG ODM Enterprise is a software enterprise platform for developing and deploying solutions based on optimization algorithms, e.g. CPLEX, for decision makers, especially in business. IBM ILOG ODM Enterprise facilitates scenario creation and comparison. Therefore the scenario management and analysis are provided by this software.

17. ***Data sparsity can be exploited to create only the essential variables and constraints, thus reducing memory requirements***.

True: CPLEX Studio allows one to create variables and constraints for only the relevant data. This is helpful when dealing with very large problems, where limited memory might become an issue.

Sparsity is defined as the fraction of zeros in a matrix: a sparse matrix has a large number of zeros compared to non-zeros. An optimization model usually contains data structures that are in effect combinations of other data structures, such a tuple that includes other tuples as elements, or a multi-dimensional array that uses a different set to index each dimension. However, not all possible combinations are valid. Therefore, removing these combinations leads to time and memory saving.

It is possible to use tuple sets to instantiate only valid combinations, create a sparse array by indexing on the tuple set and sparse data structures.

To sparse data structure, it is necessary to do the following steps:

– use tuple sets to initialize only valid combinations:

```
tuple ExampleData
{

string ProjectID;
string AppsID;

};
```

– creation of a set of all valid pairs:

```
(ExampleData) ProjectApps=…
```

– creation of a sparse array through Boolean variable:

```
dvar Boolean i [ProjectApps];
```

18. ***CPLEX Studio and ODM Enterprise are used to develop packaged optimization-based applications***.

False: CPLEX Studio and ODM Enterprise are used to develop custom optimization-based applications.

ODM Enterprise allows the development and deployment of applications that are scaled to meet individual customers' needs, from standalone Individual (desktop) configurations to enterprise configurations.

CPLEX Studio is a core part of ODM Enterprise, an analytical decision support toolkit for rapid development and deployment of optimization models. CPLEX Studio is enabled for creation the application for decision-making support. Therefore, it is also an instrument for developing custom optimization-based applications.

Table 2 MP and CP engines

MP engine	CP engine
Usage of a combination of relaxations that were strengthened by cutting-planes and "branch and bound" approach	Making a decision on variables and values and, after each decision, performs a set of logical inferences to reduce the available options for the remaining variables' domains
Usage of a lower bound proof provided by cuts and linear relaxation	Proving an optimality by showing that no better solution than the current one can be found
Requirement of the model falls in a well-defined mathematical category	Not making assumptions on the mathematical properties of the solution space

19. ***CPLEX Optimizer includes both MP and CP optimization engines***.

True. MP and CP are both critical to solving complex planning and scheduling problems. Therefore, both engines include to the CPLEX Optimizer. Some of their differences that complemented each other are presented in Table 2.

20. ***CPLEX Studio is embedded in ODM Enterprise for rapid prototyping***.

True. CPLEX Studio is at the core of applications developed as the integrated system IBM ILOG ODM Enterprise.

This system maintains optimization technology-based analytical decision support. ODM Enterprise enables rapid prototyping the flexible decision support applications. Applications based on ODM Enterprise allow users to adjust assumptions, operating constraints and goals.

21. ***A run configuration in CPLEX Studio can contain more than one model***.

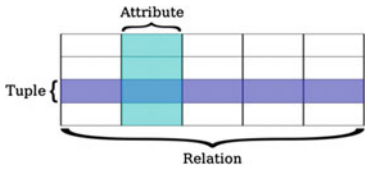
False: A run configuration in CPLEX Studio can contain only one model.

Run configurations are a way of handling model, data, and settings files within a project. Run configurations allows to combine models, data sets and optimizer settings for execution, providing a convenient method for testing a model's behavior across data instances, or test different models on the same data set, or test different settings on model/data combinations without having to resort to writing OPL Script or coding. Basically, a run configuration is a variation of a given project for execution purposes, so it must specify which precise model OPL needs to solve. Therefore, it combines a single model file and, optionally, one or more data files and one or more settings files within the project.

22. ***A tuple is analogous to a database table***.

False: A tuple is the analogous to a ROW in a database table.

A tuple is an ordered set of values represented by an array. Thus, the set of tuples can create the table (Fig. 4).



Tuples can be presented in such ways:
tuple = ( ) # Empty tuple
tuple = 1, # One element tuple
tuple = (1) # not a tuple!
tuple = (1, 2, 3) # 3 element tuple
tuple = 1, 2, 3 # The same tuple

Fig. 4 Tuple presentation in the table form

23. ***The following declaration in the OPL .dat file is correct: float production Cost = 2.5***.

False: The declaration of type 'float' is used in the model, not in the .dat file.

Float is a reserved word for the data type that is used in model files (.mod), but a .dat file is for saving the data in a file, with the extension .dat. Therefore, the declaration is right, but it could not be in .dat file, but in the model file.

24. ***You should label all constraints to be able easily track conflicts***.

False: There is no need to label all of them. However, only labeled constraints are considered by the relaxation and conflict search process.

It is possible to identify constraints by attaching labels to them. This practice is recommended but not required. Furthermore, it is possible to underline both advantages and disadvantages.

Constraint labels enable to benefit from the expand feature in the IDE Problem Browser to find which constraints are tight in a given application or to find dual variable values in linear programs. In addition, it is possible to access the slack and dual values for labeled constraints when a solution is available. Only labeled constraints are considered by the relaxation and conflict search process in infeasible models. However, labels take some performance and memory that can be significant for large models. Nevertheless, there still no need to label all constraints.

4 Conclusions

CPLEX lab sessions were included in the course as a means for the participants to have a close experience with real-world Operations Research software, which they might apply in their future Supply Chain engagements. These lab sessions were short on purpose, to adapt to the tight course schedule. However, all capabilities of the software were presented, to serve as an introductory session rather than as a learning course. The student willing to deepen in his or her skills might then know what to expect and what materials to use.

A quiz with true/false questions was presented to the course participants with the intention to assess their understanding of the topics related to the CPLEX tools. Although some of the questions might be seen as tricky for a "true/false" quiz, this format was chosen for brevity and ease of delivery.

References

Bradley, S., Hax, A., & Magnanti, T. (1977). *Applied mathematical programming*. Reading, MA, USA: Addison-Wesley.

Imamoto, A., Tang, B. (2008). Optimal piecewise linear approximation of convex functions. In *Proceedings of the World Congress on Engineering and Computer Science, WCECS 2008*, October 22–24, 2008, San Francisco, USA.

Rossi, F., van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming* (978 p). Elsevier.

Williams, H. P. (1999). *Model building in mathematical programming*. New York: Wiley.