Linear and Mixed Integer Programming 30

Hartmut Stadtler

Linear Programming (LP) is one of the most famous optimization techniques introduced independently by Kantarowitsch in 1939 and by Dantzig in 1949 (Krekó 1973). LP is applicable in decision situations where quantities (variables) can take any real values only restricted by linear (in-) equalities, e.g. for representing capacity constraints. Still, LP has turned out to be very useful for many companies so far. LP is used in APS e.g. in Master Planning as well as in Distribution and Transport Planning. Very powerful solution algorithms have been developed (named solvers), solving LP models with thousands of variables and constraints within a few minutes on a personal computer.

In case some decisions can only be expressed by integer values, e.g. the number of additional shifts for a given week, LP usually will not provide a feasible solution. Similarly, logical implications might be modeled by binary variables. As an example consider the decision whether to setup a flow line for a certain product or not: A value of "0" will be attributed to a decision "no" and a value of "1" to "yes". Still, the corresponding model may be described by linear (in-) equalities. In case the model solely consists of integer variables, it is called a pure *Integer Programming* (IP) model. If the model contains both real and integer variables a *Mixed Integer Programming* (MIP) model is given.

Thus, both LP and MIP comprise special model types and associated solution algorithms. Numerous articles and textbooks have been written on LP and MIP (e.g. Martin 1999; Winston 2004; Wolsey 1998) representing a high level of knowledge which cannot be reviewed here. In order to give an understanding of LP and MIP, only the basic ideas will be provided in the following by means of an example.

First, an LP model is presented and solved graphically (Sect. 30.1). This model is then converted into an IP model and solved by Branch and Bound (Sect. 30.2),

H. Stadtler (✉)
Institute for Logistics and Transport, University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
e-mail: hartmut.stadtler@uni-hamburg.de

H. Stadtler, C. Kilger and H. Meyr (eds.), *Supply Chain Management and Advanced Planning*, Springer Texts in Business and Economics,
DOI 10.1007/978-3-642-55309-7_30, © Springer-Verlag Berlin Heidelberg 2015

where for each submodel a LP model is solved graphically. Finally, a few remarks and recommendations regarding the effective use of LP and MIP complements this chapter (Sect. 30.3).

30.1 Linear Programming

A hypothetical production planning problem is considered here, where two products A and B can be produced within the next month. The associated production amounts are represented by (real) variables x_1 and x_2 measured in ten tons. Both products have to pass through the same production process. The available capacity is 20 days (on a two shift basis). The production of ten tons, or one unit, of product A lasts 5 days, while the respective coefficient for product B is 4 days. This situation is represented by inequality (30.2).
LP model:

$$\text{Max!} \quad 19x_1 + 16x_2 \tag{30.1}$$

subject to

$$5x_1 + 4x_2 \leq 20 \tag{30.2}$$

$$-x_1 + 2x_2 \leq 5 \tag{30.3}$$

$$2x_1 + 5x_2 \geq 10 \tag{30.4}$$

$$x_1 \geq 0, x_2 \geq 0 \tag{30.5}$$

Inequality (30.3) represents the demand constraints, stating that only sales of product B are limited. However, we might increase sales if we also offer product A: For every two units of product A we can extend sales of product B by one unit (the reason may be that one has to offer a complete product range to some customer groups in order to sell product B). Although we aim at maximizing our revenue (30.1), we also want to make sure that a contribution margin of at least \$10,000 is reached within the next month (30.4). Note, the dimension "one thousand" is scaled down to "one" for the contribution margin constraint. Obviously, one cannot produce negative amounts which is reflected by the non-negativity constraints (NNC, see (30.5)).

This small LP model can be solved algebraically by the Simplex algorithm (or one of its variants, see Martin 1999). However, we will resort to a graphical representation (Fig. 30.1). Variables x_1 and x_2 depict the two dimensions. Inequalities restrict the combination of feasible values of variables. The limits of the corresponding set of feasible solutions are illustrated by a line (see Fig. 30.1). Whether the set of feasible solutions lies below or above a line is depicted by three adjacent strokes being part of the set of feasible solutions.

The intersection of all the (in-) equalities of a model defines the set of feasible solutions (shaded area in Fig. 30.1). For a given objective function

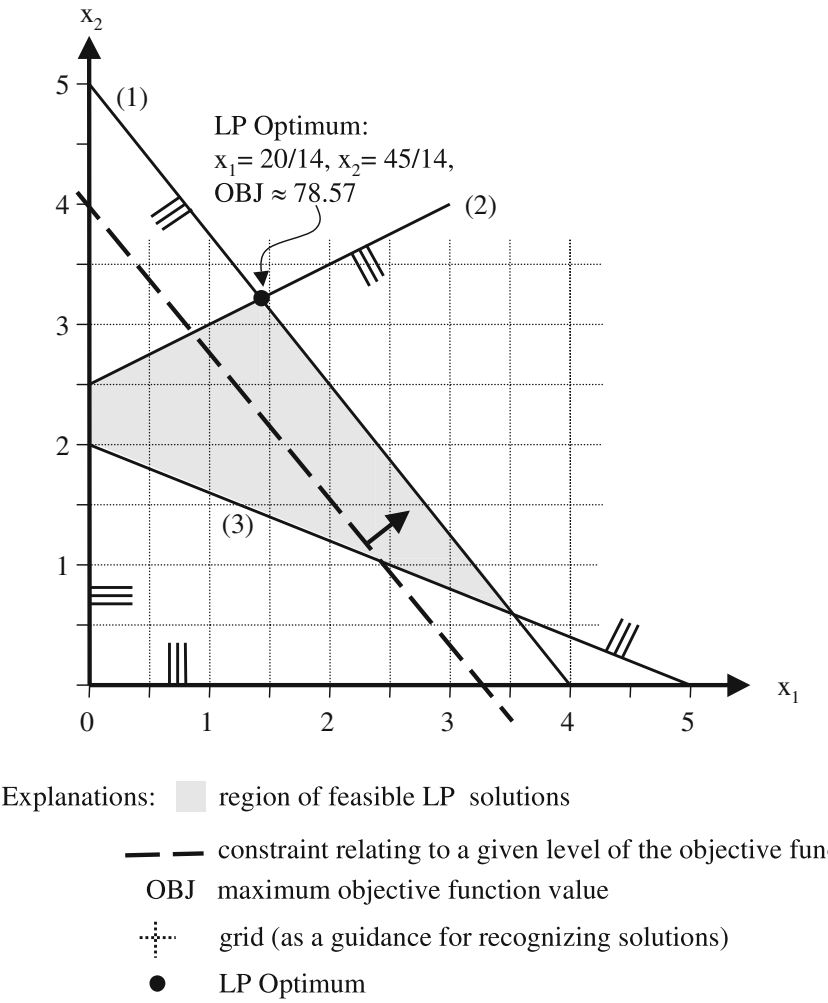

Fig. 30.1 Graphical representation of an LP model

value the objective function itself is an equation (see dashed line in Fig. 30.1, corresponding to a value of 76 [$000]). Since we do not know the optimal value of the objective function we can try out several objective function values. An arrow shows the direction in which the objective function value can be increased. Actually, we can move the dashed line further to the right. The maximum is reached once it cannot be moved any further without leaving the set of feasible solutions. This is the case for $x_1 = 20/14$ and $x_2 = 45/14$ resulting in a revenue of 78.57 [$000]. The optimal solution has been reached at the intersection of inequalities (1) and (2). It can be shown that it suffices to look for an optimal solution only at the intersections

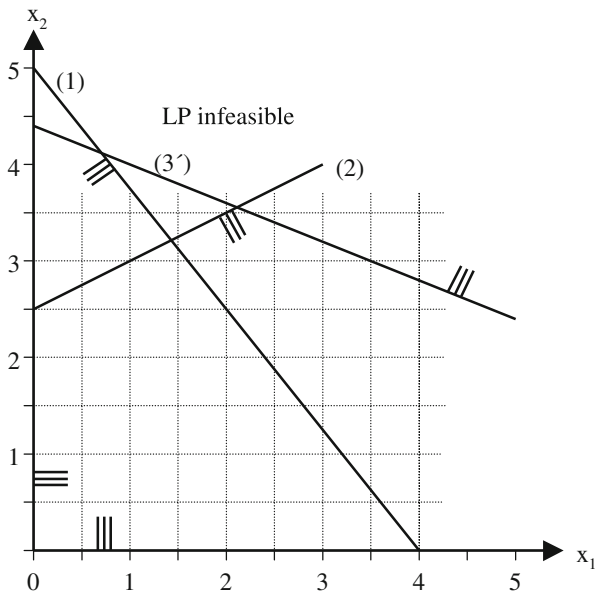

Fig. 30.2 An infeasible LP model

of (in-) equalities limiting the set of feasible solutions or, graphically speaking, at the "corners" of the shaded area.

The Simplex algorithm (and its variants) carries out the search for an optimal solution in two phases, namely

- Creating an initial feasible solution
- Finding an optimal solution.

In our example a first feasible solution may be $x_1 = 0$ and $x_2 = 2$ with a revenue of $2 \cdot 16 = 32$ [\$000]. Now, the second phase is started, probably generating an improved second solution, e.g. $x_1 = 0$ and $x_2 = 2.5$ with a revenue of 40 [\$000]. In the next iteration variable x_1 will be introduced, resulting in the optimal LP solution.

However, an initial feasible solution may not always exist. As an example, assume that a minimum contribution margin of 22 [\$000] is required (see inequality (3') in Fig. 30.2). The set of feasible solutions is empty and thus no feasible solution exists.

Now consider the situation where there is no production constraint [i.e. eliminating inequality (1)], resulting in an unrestricted shaded area (Fig. 30.3) and an unbounded objective function value. This case will also be detected in the first phase. Actually, an unbounded solution indicates that the model or the data have not been created correctly.

We would like to point out that an LP solution does not only provide optimal values for the decision variables. It also shows the *dual values* associated with the (in-) equalities of an LP model. As an example consider the production capacity (30.2). If we were able to increase the number of working days from 20 to 21, the optimal objective function value would rise from 1100/14 to 1154/14.

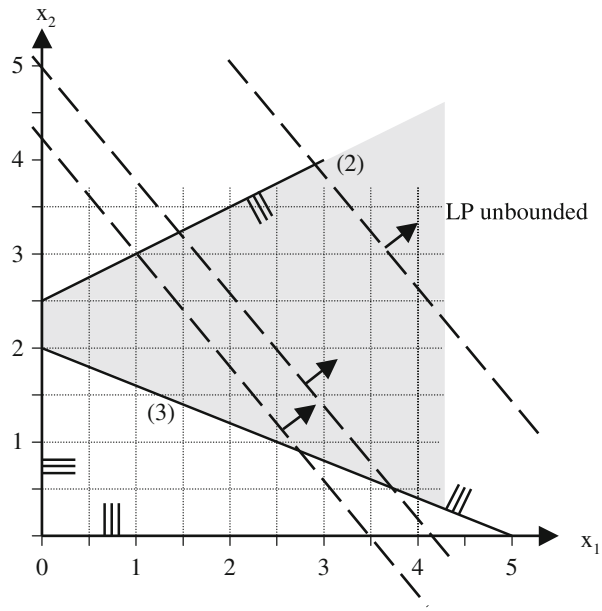

Fig. 30.3 An unbounded LP model

Thus an additional capacity unit has a dual value of 3.86 [$000]. Management now may look for options to extend capacity which are worth further revenues of 3.86 [$000] per working day. Note that only inequalities which are binding in the optimal solution may have a positive dual value. Although dual values have to be interpreted with caution, they are a fruitful source for finding ways to improve the current decision situation.

As has already been stated at the beginning of this chapter, very powerful solution algorithms and respective standard software exist for solving LP and MIP models (e.g. IBM ILOG CPLEX Optimizer 2014 and FICO Xpress Optimization Suite 2014). However, users of an APS do not have to deal with these solvers directly. Instead, special modeling features have been selected within APS modules for building correct models. Still, care should be taken regarding the numbers entering the model. If possible, appropriate scaling should be introduced first, such that the coefficients of variables are in the range from 0.01 to 100 to avoid numerical problems.

30.2 Pure Integer and Mixed Integer Programming

Now let us assume that a product can only be produced in integer multiples of ten [tons], since this is the size of a tub which has to be filled completely for producing either product A or B. Then the above model (30.1)–(30.4) has to be complemented by the additional constraints

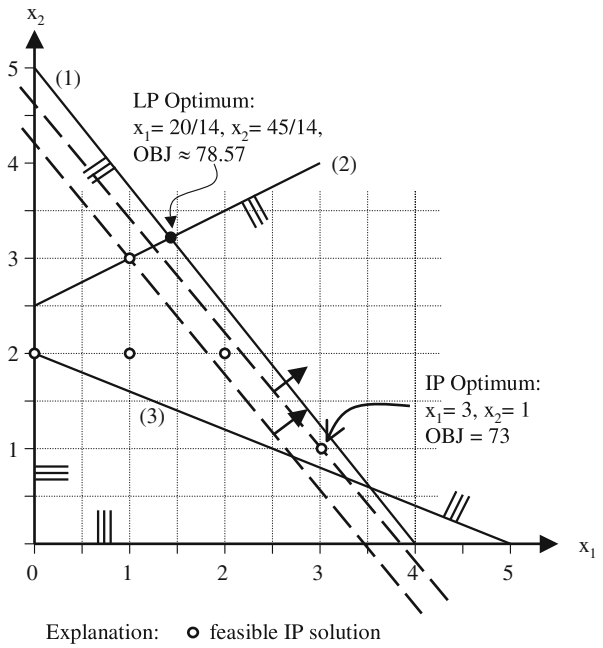

Fig. 30.4 A graphical representation of an IP model

$$x_1 \in N_0\,, \quad x_2 \in N_0 \tag{30.6}$$

The set of feasible solutions reduces drastically (see the five integer solutions in Fig. 30.4). Still, in practice the number of solutions to consider before an integer solution has been proven to be optimal may be enormous.

As can be seen from Fig. 30.4 a straightforward idea, namely rounding the optimal LP solution to the next feasible integer values ($x_1 = 1$, $x_2 = 3$ with a revenue of 67 [$000]), does not result in an optimal integer solution (which is $x_1 = 3$, $x_2 = 1$ and with a revenue of 73 [$000]).

Anyway, an intelligent rounding heuristic might be appropriate for some applications. Hence, some APS incorporate rounding heuristics which usually require much less computational efforts than Branch and Bound which is explained next.

Four building blocks have to be considered describing a *Branch and Bound* algorithm, namely

- Relaxation
- Separation rules
- Search strategy
- Fathoming rules.

The two building blocks *separation rules* and *search strategy* relate to "branch" while *relaxation* and *fathoming rules* concern "bound". These building blocks will now be explained by solving our example.

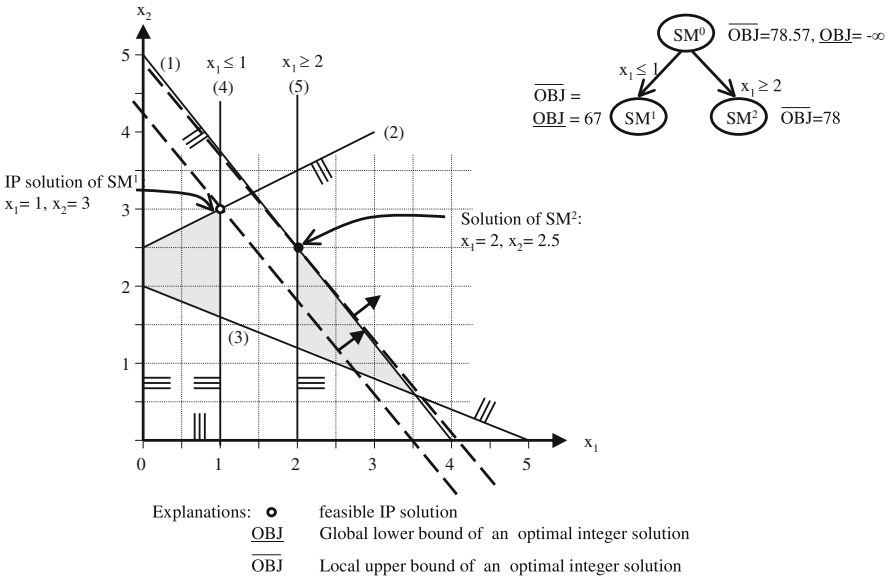

Fig. 30.5 A graphical representation of the first and second submodel

Although solving the associated LP model directly usually does not yield an optimal integer solution, we can conclude that the set of feasible integer solutions is a subset of the set of feasible LP solutions. So, if we were able to cut off some parts of the non-integer solution space, then we would finally arrive at an integer solution.

Consequently, we first *relax* the integer requirements (30.6) in favor of the non-negativity constraints (30.5). The resultant model is called an *LP relaxation*. If we solve an LP relaxation of a maximization problem, the optimal objective function will be an *upper bound* for all integer solutions contained in the associated set of feasible (integer) solutions. Hence, if the solution of an LP relaxation fulfills the integer requirements (30.6), it will be an optimal integer solution for this (sub-) model.

Next, submodels are created by introducing additional constraints, such that a portion of the real-valued non-integer solution space is eliminated (see Fig. 30.5). Here, the constraint $x_1 \leq 1$ is added resulting in submodel SM^1, while constraint $x_1 \geq 2$ yields submodel SM^2. Now, we have to solve two submodels with a reduced set of feasible solutions. Note that the union of the set of feasible *integer* solutions of both submodels matches the initial set of feasible integer solutions, i.e. no integer solution is lost by *separation*.

Submodel SM^1 results in a first *integer* solution ($x_1 = 1$, $x_2 = 3$ with a revenue of 67 [$000] representing the local upper bound of SM^1). Subsequently, we will only be interested in solutions with a revenue of more than 67 [$000]. Thus, we set the global lower bound to 67 [$000] ($\underline{OBJ} = 67$). The term "global" is used in order to refer to our original IP model. Since submodel SM^1 has resulted in an *integer*

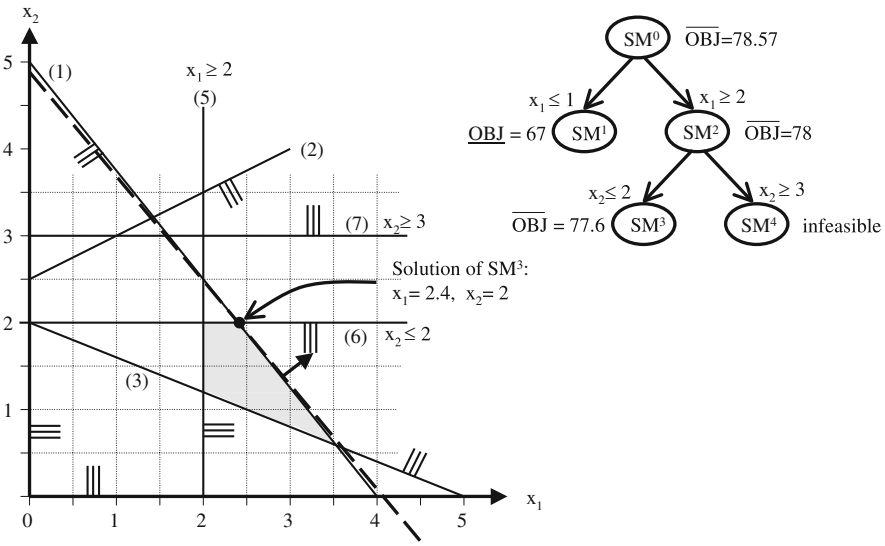

Fig. 30.6 A graphical representation of the third and fourth submodel

solution (and cannot yield a better solution) it will be discarded from our list of open submodels, i.e. it is *fathomed*.

The second submodel has a local upper bound of 78 [$000] which is clearly better than our current global lower bound, but its solution is non-integer valued ($x_2 = 2.5$).

The search for an optimal solution can be represented by a *search tree* (see right hand side of Fig. 30.5). Each node corresponds to an LP (sub-)model.

Now an unfathomed submodel has to be chosen for further investigations. However, only submodel SM^2 is unfathomed here. Subsequently, one has to decide on the non-integer-valued variable to branch on. These two choices make up the *search strategy* and may have a great impact on the number of submodels to solve and hence the computational effort.

The only variable which is non-integer valued in the optimal solution for submodel SM^2 is x_2. Two new submodels are created, submodel SM^3 with the additional constraint $x_2 \leq 2$ and submodel SM^4 with the additional constraint $x_2 \geq 3$. Note that all additional constraints that have been generated on the path from the origin (SM^0) to a given submodel in the search tree have to be taken into account (here $x_1 \geq 2$).

Since, there *is no feasible (real valued) solution* for submodel SM^4 (see Fig. 30.6) it may be *fathomed*. For submodel SM^3 a non-integer valued solution with an upper bound of 77.6 [$000] is calculated. Since this local upper bound exceeds the global lower bound (i.e. the best objective function value known) submodel SM^3 must not be fathomed.

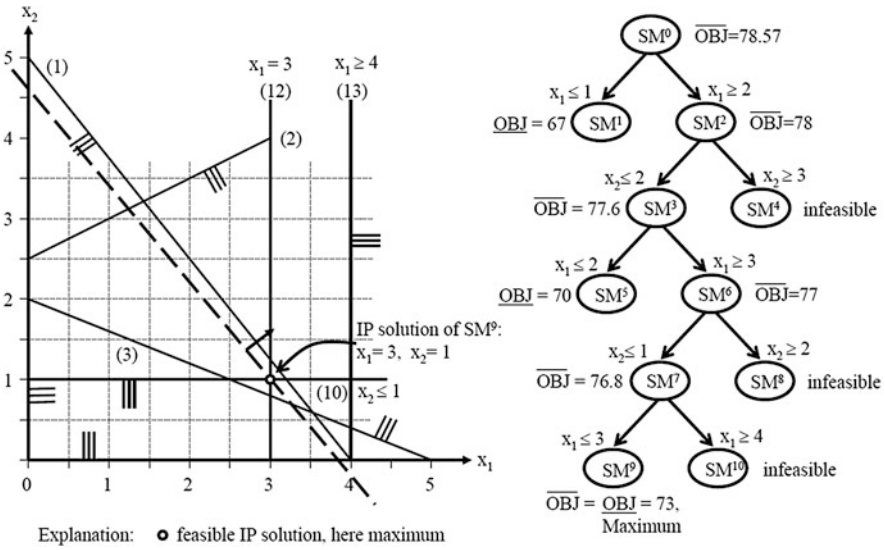

Fig. 30.7 A graphical representation of the optimal integer solution and the complete search tree

It now takes three further separations until we reach submodel SM^9 (Fig. 30.7), where the LP relaxation yields an integer solution with an objective function value of 73 [$000].

Usually, there will be some unfathomed submodels which have been generated in the course of the search. An unfathomed submodel has to be selected for a further separation until all submodels are fathomed. Then the best feasible integer solution found will be the optimal one for the initial IP model.

In our example, the search ends once it has been found out that submodel SM^{10} has no feasible solution. Now we have proven that the solution to submodel SM^9 is optimal.

Finally, we would like to add that the Branch and Bound scheme is almost the same for MIP models. As an example consider that only x_2 has to take integer values. Then we would start separating on variable x_2 (i.e. $x_2 \leq 3$ and $x_2 \geq 4$). Only constraint $x_2 \leq 3$ results in a feasible solution for the LP relaxation. Since it is also feasible with respect to the mixed integer constraints it is the optimal solution, too.

30.3 Remarks and Recommendations

Although the examples presented are rather simple, they have illustrated the differences in solving an LP model and a MIP model. Generating an optimal solution for an LP model requires "some" Simplex iterations leading from one "corner" of the feasible solution space to the next and finally to the optimal one. However, solving a MIP model by Branch and Bound incurs solving an LP (sub-)

model for each node of the search tree—and there may be several thousand nodes to explore until an optimal solution has been proven.

One way to reduce the number of submodels to investigate is to *truncate the search effort*. For example, the user may either set a certain time limit for the search or indicate that the search has to be stopped once the k-th feasible integer solution has been found. However, the problem with truncation is that one does not know in advance at which point in time a feasible or good solution will be found.

Another option to limit the computational effort of Branch and Bound is to specify in advance that the search for an improved solution should be stopped, once we are sure that there is no feasible integer solution which is at least $\delta\%$ better than our current best solution. This allows us to calculate an aspiration level in the course of Branch and Bound, simply by multiplying the objective function value of the current best solution by $(1 + \delta\%)$. The question whether there exists a feasible integer solution with an objective function value no less than the aspiration level is known from the maximum upper bound of *all* unfathomed submodels. If the maximum is less than our aspiration level the search is stopped.

In our example (see the search tree in Fig. 30.7) we now assume $\delta = 10$. Having generated the first integer solution ($\underline{\text{OBJ}} = 67$) the aspiration level is 73.7 [\$000]. Since the maximum of the upper bounds of unfathomed submodels is 78 [\$000] (submodel 2) the search will continue. Having reached the second integer solution with an objective function value of 70 [\$000], an aspiration level of 77 [\$000] is calculated. In this example the search stops once the maximum upper bound of all unfathomed submodels falls below 77 [\$000] which is true after having generating submodel 8.

The number of submodels to solve largely depends on the relative difference between the objective function value of the LP relaxation and the optimal integer solution, named *integrality gap*. For our example the integrality gap is rather modest (e.g. $(78.57 - 73)/73 = 0.076$ or 7.6 %). The smaller the integrality gap is, the greater is the chance to fathom submodels and thus to keep the search tree small. Today much effort is invested in deriving additional *valid inequalities (cuts)* to yield small integrality gaps for each submodel generated within Branch and Bound (see Wolsey 1998; Pochet and Wolsey 2006).

A further option applied by advanced MIP solvers to reduce the search effort is *preprocessing*. Here one investigates the interactions of the model's constraints in order to restrict or even fix the values of some integer variables before starting Branch and Bound. For our example, one might conclude that the set of feasible integer values for x_1 will be restricted to {0,1,2,3} and to {1, 2, 3} for x_2.

We would like to add that the logic of Constraint Programming (see Lustig and Puget 2001; Milano and Wallace 2010), previously considered as an alternative to MIP solvers, now is (partly) integrated into most recent MIP solvers. A survey of the latest LP software is provided by Fourer (2013). As a last resort to keep CPU times low more powerful hardware may be used. For example several submodels generated in the course of Branch and Bound now can be solved effectively in parallel by multiprocessor computers.

A frequently asked question is: "Will our Master Planning model be solvable within reasonable CPU-times?" Before answering this question one has to differentiate whether the Master Planning model is to be solved by an LP or a MIP solver or a simple heuristic.

As already stated, purely linear models are much easier to solve than MIP models. Actually, solution capabilities of state-of-the-art LP solvers should be sufficient for solving almost all reasonable real world applications. However, if elapsed time plays a role a few experiments at an early stage of a project should clarify matters: The idea is to generate an LP model with only a subset J^r of all products J and/or with a reduced number of time periods T^r compared with T periods in the final model ($T^r << T$), but representing the same model structure, i.e. containing all types of constraints of the final model. Assuming that the reduced model requires a CPU-time CPU^r, a rule-of-thumb for calculating the CPU-time (CPU) of the final model is:

$$CPU \sim \left(\frac{T}{T^r} \cdot \frac{|J|}{|J^r|} \right)^3 \cdot CPU^r \tag{30.7}$$

This rule-of-thumb is derived from the observation that the computational time required for solving an LP by a Simplex method tends to be roughly proportional to the cube of the number of explicit constraints, so that doubling this number may multiply the computational time by a factor of approximately 8 (Hillier and Liebermann 2010, p. 138).

For MIP models an optimal solution usually cannot be expected within reasonable CPU times. Hence, the search for an optimal solution is truncated (see above). Also, remember that always a (relaxed) LP model has to be solved before the search for a MIP solution starts. In any case the CPU time limit must be sufficient for at least generating a first feasible MIP solution.

Again, preliminary experiments with the MIP model can provide valuable insights. One approach is to start with relaxing all integer requirements, resulting in a purely linear model. If this model is solved easily, then the most important integer requirements can be introduced and the associated computational effort observed. Then further variables may be declared integer until a good compromise between the solution effort, solution quality and model adequacy is reached.

A second related approach is to specify a complete MIP model including all desired integer requirements but to relax some integer requirements for variables in later periods in the planning horizon, where e.g. only a rough capacity check suffices.

A third approach supported by some software vendors is to use time or stage oriented decomposition. If this option is chosen, the overall model is partitioned into smaller submodels (automatically) which are then solved successively (e.g. a MIP model covering 13 periods is partitioned into four MIP models with four periods each, while there is one overlapping period). In the end the user will get a complete solution for the original decision problem.

In any case the user should use integer or binary variables carefully—a MIP model incorporating (only) 100 integer variables may already turn out to require excessive computational efforts.

References

FICO Xpress Optimization Suite. (2014). Homepage, http://www.fico.com/en/products/fico-xpress-optimization-suite/. Visited on Feb. 28, 2014.

Fourer, R. (2013). Linear programming. *OR/MS Today*, *40*, 40–53.

Hillier, F., & Liebermann, G. (2010). *Introduction to operations research* (9th ed.). Boston: McGraw-Hill.

IBM ILOG CPLEX Optimizer. (2014). Homepage, http://www-03.ibm.com/software/products/de/ibmilogcpleoptistud/. Visited on Feb. 28, 2014.

Krekó, B. (1973). *Lehrbuch der Linearen Optimierung* (6th ed.). Berlin: Deutscher Verlag d. Wiss..

Lustig, I., & Puget, J.-F. (2001). Program does not equal program: constraint programming and its relationship to mathematical programming. *INFORMS Journal on Computing*, *31*, 29–53.

Martin, R. (1999). *Large scale linear and integer optimization: A unified approach*. Boston: Kluwer Academic.

Milano, M., & Wallace, M. (2010). Integrating operations research in constraint programming. *Annals of Operations Research*, *175*, 37–76.

Pochet, Y., & Wolsey, L. (2006). *Production planning by mixed integer programming* (1st ed.). New York: Springer.

Winston, W. (2004). *Operations research: Applications and algorithms* (4th ed.). Belmont, CA: Thomson/Brook/Cole.

Wolsey, L. (1998). *Integer programming*. New York: Wiley.