HEURISTICS IN MIXED INTEGER PROGRAMMING

MATTEO FISCHETTI
DEI, Universitá di Padova, Padova, Italy

ANDREA LODI
DEIS, Universitá di Bologna, Bologna, Italy

INTRODUCTION

We consider a generic mixed integer linear program (MILP) in the form

$$\min c^{\mathrm{T}}x \tag{1}$$

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

$$x_j \in \{0,1\} \quad \forall j \in \mathcal{B}, \tag{3}$$

$$x_j \text{ integer} \quad \forall j \in \mathcal{G}, \tag{4}$$

$$x_j \text{ continuous}, \quad \forall j \in \mathcal{C}, \tag{5}$$

where A is an $m \times n$ input matrix, and b and c are input vectors of dimension m and n, respectively. Here, the variable index set $\mathcal{N} := \{1, \ldots, n\}$ is partitioned into $(\mathcal{B}, \mathcal{G}, \mathcal{C})$, where $\mathcal{B}$ is the index set of the 0–1 variables (if any), while the sets $\mathcal{G}$ and $\mathcal{C}$ index the general integer and the continuous variables, respectively. Bound on the variables, including the 0–1 bounds on the binary ones, are assumed to be part of system (2). Removing the integrality requirement on variables indexed by $\mathcal{I} := \mathcal{B} \cup \mathcal{G}$ leads to the *LP relaxation* $\min\{c^{\mathrm{T}}x : x \in P\}$ where $P := \{x \in \mathcal{R}^n : Ax \geq b\}$.

MILP heuristics aim at finding a feasible (and hopefully good) solution of the problem above, which is an NP-hard problem by itself. We next present the main ideas underlying some of the heuristics proposed in the literature. In this article, in particular, we focus on those algorithms developed with the aim of being tightly integrated within MILP solvers. We group and discuss the algorithms depending on the main ingredient (building block) they use, namely LP-based heuristics in the section titled "LP-Based Heuristics" and MILP-based approaches in the section titled "MILP-Based Heuristics."

With a little abuse of terminology, in what follows, we will say that a point x is *integer* if x_j is integer for all $j \in \mathcal{I}$ (no matter what the value of the other components), whereas the *rounding* $\tilde{x}$ of a given x will be defined as $\tilde{x}_j := [x_j]$ if $j \in \mathcal{I}$ and $\tilde{x}_j := x_j$ otherwise ($[\cdot]$ representing scalar rounding to the nearest integer).

We end the introduction by noting that, although Achterberg [1] has shown that the impact of heuristics is not dramatic in terms of ability of an MILP solver to prove optimality in a (much) quicker way, the psychological impact for the user who sees a high quality feasible solution provided very early is huge. For this reason, the availability of very effective general-purpose heuristics for MILP is among the most crucial improvements over the last 10 years.

LP-BASED HEURISTICS

In this section we address some basic heuristics that only assume the availability of a "black-box" LP solver.

Folklore

The so-called *rounding* as well as *diving* heuristics belong to folklore.

Rounding methods solve the LP relaxation to get an optimal point x^*, whose fractional components x_j^* with $j \in \mathcal{I}$ are rounded, for example, to their nearest integer value. Continuous variables, if any, can optimally be recomputed once the integer variables have been fixed by solving an LP.

Diving (also known as *Relax-and-Fix*) methods, instead, mimic an enumerative scheme by sequentially fixing some integer-valued variables that assume a fractional value in the solution of the current LP-relaxation. The sequence can be viewed

Wiley Encyclopedia of Operations Research and Management Science, edited by James J. Cochran

as "diving" on a branch-and-bound tree until a feasible (and hopefully good) MILP solution is found. A limited backtracking is often allowed to investigate alternative fixing patterns. Various variable-fixing (i.e., branching) policies have been used by different authors, all meant to hopefully reach deep branching nodes while maintaining a good LP relaxation lower bound value.

Pivoting Methods

These methods originate from the seminal work of Balas and Martin [2] in the late 1970s. The basic heuristic, called *Pivot and Complement* (P&C), applies to pure 0–1 integer linear programs (ILPs) with $\mathcal{G} = \mathcal{C} = \emptyset$. It is based on the observation that a feasible solution is just a basic solution of the LP relaxation where all 0–1 variables are nonbasic (either at their lower or upper bound), that is, only the slack (continuous) variables on Equation (2) are allowed to be basic. Given an optimal basis of the LP relaxation, P&C then performs a systematic series of "nonstandard" pivots in the attempt of driving as many 0–1 variables as possible out of the basis, while preserving primal feasibility and worsening the objective function as little as possible.

More specifically, the method prescribes the use of three types of pivots: (i) Type 1 pivots that maintain primal feasibility and increase the number of nonbasic 0–1 variables; (ii) Type 2 pivots that also maintain primal feasibility but leave the number of nonbasic 0–1 variables unchanged, while reducing the *infeasibility degree* of the current LP solution x^*, computed as $\sum_{j \in \mathcal{B}} \min\{x_j^*, 1 - x_j^*\}$; (iii) Type 3 pivots that increase the number of nonbasic 0–1 variables at the price of producing a primal-infeasible point—a situation to be repaired at a later step of the procedure.

Nonbasic 0–1 variables can also be complemented with the aim of improving the solution quality or reducing its primal infeasibility, and reduced costs are exploited to fix variables when mathematically correct. The P&C heuristic was later improved by Løkketangen *et al*. [3] by adding tabu search to the basic mechanisms.

In the mid-1980s Balas and Martin [4] introduced a generalization of P&C to MILPs, called *pivot and shift*, where general integer variables are treated as 0–1 variables "centered" on the current LP solution, and some new types of "nonstandard" pivots are introduced. An elaboration with the same name of this idea was later proposed by Balas *et al*. [5]. Different extensions called *pivot, cut, and dive* and *pivot and Gomory cut* were proposed by Nediak and Eckstein [6] and by Gosh and Hayward [7], respectively.

OCTANE and Line Search Methods

Line search methods are based on the following construction. One starts from an LP relaxation optimal vertex x^* and draws a line toward a second point, for example, y, to be chosen according to a certain criterion. Thus, moving in a discretized way from x^* to y traces a discrete sequence of (say) K points $x^k := x^* + \alpha_k(y - x^*)$ for $0 = \alpha_1 < \cdots < \alpha_K = 1$. Each x^k can therefore be associated (e.g., by rounding) with an integer point $\tilde{x}^k$ which is then checked for feasibility and possibly used to update a current-best MILP solution. Local shifts of the rounded variables are typically allowed in the attempt of improving the feasibility and/or quality of the rounded points.

Hillier [8] introduced this scheme by defining y so as to go inside P along a so-called *interior path*, the rational being supported by the geometrical intuition that (mainly for general integer MILPs) going inside P increases the chances the rounded point be feasible. A similar scheme was later adopted by Ibaraki *et al*. [9] and by Faaland and Hiller [10], where the sampling line segment $[x^*, y]$ is replaced by a sequence of linear segments $[x^*, y^1]$, $[y^1, y^2], \ldots$ approximating a curved search trajectory. Very recently, Naoum-Sawaya and Elhedhli [11] investigated the effect of replacing the LP solver by an analytic-center method.

OCTANE (for Octahedral Neighborhood Enumeration) is a line search method proposed by Balas *et al*. [12] for 0–1 ILPs. It also starts with an LP optimal vertex x^* and moves it along a certain search direction $y - x^*$. However, the integer (possibly infeasible) points discovered along this direction

are not defined through rounding, but exploit the following construction. Consider the center $x^0 = (1/2, \ldots, 1/2)$ of the unit hypercube. For any vertex $\tilde{x}$ of the unit hypercube (not necessarily in P), define the hyperplane $H(\tilde{x})$ passing through $\tilde{x}$ and orthogonal to $\tilde{x} - x^0$.[1] Starting from x^* and going toward y one crosses, in sequence, certain hyperplanes $H(\tilde{x}^1), H(\tilde{x}^2), \ldots$ whose generating 0–1 points $\tilde{x}^1, \tilde{x}^2, \ldots$ are checked for feasibility and used to possibly update the current-best feasible solution. *OCTANE* prescribes to visit only a limited number of hyperplanes, and exploits an effective method for their enumeration.

Feasibility Pump

The *Feasibility Pump* (FP) is a method originally proposed by Fischetti *et al*. [13] for 0–1 MILPs ($\mathcal{G} = \emptyset$), and then extended by Bertacco *et al*. [14] to the general case. It is based on the observation that a feasible MILP solution is a point x of P that coincides with its rounding. Replacing "coincides" with "is as close as possible" leads to the following iterative scheme. FP works with a *pair* of points $(x^*, \tilde{x})$ with $x^* \in P$ and $\tilde{x}$ integer, that are iteratively updated with the aim of reducing as much as possible their L_1-distance $\Delta(x^*, \tilde{x}) := \sum_{j \in \mathcal{I}} |x_j - \tilde{x}_j|$. In a sense, the method is aimed at "pumping" the integrality of $\tilde{x}$ into x^*, and the LP-feasibility of x^* into $\tilde{x}$—hence its name.[2] To be more specific, one starts with any $x^* \in P$, for example, an optimal LP vertex, and initializes an integer $\tilde{x}$ as the rounding of x^*. At each FP iteration, called a *pumping cycle*, $\tilde{x}$ is fixed and one finds a point $x^* \in P$ which is as close as possible to $\tilde{x}$ by solving the auxiliary LP $\min\{\Delta(x, \tilde{x}) : x \in P\}$. If $\Delta(x^*, \tilde{x}) = 0$, then x^* is integer and the method is completed. Otherwise, $\tilde{x}$ is replaced by the rounding of x^* so as to hopefully reduce $\Delta(x^*, \tilde{x})$ even further, and the process is iterated. The basic FP scheme above may stall in case $\Delta(x^*, \tilde{x})$ is not reduced at some iteration, hence a number of diversification mechanisms are introduced.

As stated, FP is meant to only produce a feasible MILP solution, as the objective function is only taken into account implicitly in the definition of the very first x^*. In order to produce good quality solutions, the original FP approach [13] uses a "sliding window" method that introduces the cut $c^{\mathrm{T}}x \le UB$ into the LP model and updates UB, on the fly, by taking an intermediate value between the optimal LP-relaxation and the UB values. (Note that the value UB can also be a "guess" of a feasible solution value and in such a case the cut can be possibly invalid.) A different approach, called *objective FP*, was developed by Achterberg and Berthold [15], who modified the objective function of the auxiliary LP to $\Delta(x^*, \tilde{x}) + \alpha\, c^{\mathrm{T}}x$, where α is a dynamically updated parameter. Very recently, Fischetti and Salvagnin [16] proposed a more elaborated FP version, called FP 2.0, where the original rounding operation used to construct $\tilde{x}$ from x^* is replaced by a more clever rounding heuristic based on constraint propagation—a basic tool in constraint programming (see ***Constraint Programming Links with Math Programming***).

MILP-BASED HEURISTICS

We next address a new generation of MILP heuristics that emerged in the late 1990s. Their hallmark is the use of a "black-box" external MILP solver to explore a solution neighborhood defined by invalid linear constraints. The use of an exact MILP solver inside an MILP heuristic may appear naive at first glance, but it turns out to be effective in the cases where the added invalid constraints lead to a structural simplification of the MILP at hand and allow, for example, for a more powerful instance preprocessing and/or for extensive node pruning.

Local Branching

The *local branching* (LB) scheme of Fischetti and Lodi [17] appears to be the first method

[1] The half-spaces induced by these hyperplanes that contain x^0 define an octahedron, hence the name of the method.

[2] The name was actually inspired by the *Electron Pump* of the Asimov's science fiction novel "The Gods Themselves".

embedding an MILP solver within a general MILP heuristic framework. Suppose a feasible *reference solution* $\overline{x}$ of an MILP with $\mathcal{B} \neq \emptyset$ is given, and one aims at finding an improved solution that is "not too far" from $\overline{x}$. To this end, one can define the *k-OPT neighborhood* $\mathcal{N}(\overline{x}, k)$ of $\overline{x}$ as the set of the MILP solutions satisfying the invalid *local branching constraint*

$$\Delta(x, \overline{x}) := \sum_{j \in \mathcal{B}: \overline{x}_j = 0} x_j + \sum_{j \in \mathcal{B}: \overline{x}_j = 1} (1 - x_j) \leq k, \quad (6)$$

for a small parameter k (typically, $k = 10$ or $k = 20$), and explore it by means of an external MILP solver, often heuristically, that is, within a prefixed number of branch-and-bound nodes. The method is in the spirit of local search metaheuristics and in particular of *large neighborhood search* [18], with the novelty that neighborhoods are obtained through a so-called *soft fixing*, that is, through invalid cuts to be added to the original MILP model. Diversification cuts can be defined in a similar way, thus leading to a flexible toolkit for the definition of metaheuristics for general MILPs. For example, Hansen *et al*. [19] embedded LB within variable neighborhood search, whereas Fischetti *et al*. [20] specialized LB to MILPs with a two-level variable structure. Integration between FP and LB was instead proposed by Fischetti and Lodi [21].

As its name suggests, LB can also be used within an exact enumerative scheme by branching on the "local" disjunction $\Delta(x, \overline{x}) \leq k$ (to be explored first) or $\Delta(x, \overline{x}) \geq k + 1$ with respect to the incumbent MILP solution $\overline{x}$ (no matter the LP solution at a given node). This branching strategy is intended to favor the detection of good solutions very early in the enumeration.

Relaxation Induced Neighborhood Search and Variants

The *relaxation induced neighborhood search* (RINS) framework of Danna *et al*. [22] also uses the (heuristic) solution of a simplified MILP through an external MILP solver as a main ingredient, but extends the idea by taking the solution of the LP relaxations into account. At specified nodes of the branch-and-bound tree, the current LP relaxation solution x^* and the incumbent $\overline{x}$ are compared and all integer-constrained variables that agree in value are fixed and projected out of the MILP. In this way, such an MILP is not only simplified but generally speaking (provided the number of components fixed is sufficiently large) it is also reduced in size. This is probably the main difference between LB and RINS, namely the fact that the former performs a *soft* fixing by means of the cardinality constraint (6), while in the latter the fixing is *hard* with clear benefits associated with dealing with smaller MILPs.

The RINS framework is also particularly suitable for an integration within a classical branch-and-bound (and then within classical MILP solvers) because it can be virtually invoked at any node of the tree where solutions of the LP relaxations are always different. Clearly, invoking the algorithm too often would result in slowing down the overall running time for proving optimality, thus RINS is only executed every L (say) nodes in the tree. On the other hand, the size of the resulting *reduced* MILP cannot be predicted in advance and the MILP is solved only if the reduction is relevant enough.

The *distance* induced neighborhood search (DINS) approach by Ghosh [23] is a RINS variant which takes an intermediate step between LB and RINS by performing both soft and hard fixing within the same algorithm. The idea behind DINS is that the most promising solutions are those "close" to the solution of the continuous relaxation, thus the neighborhood is defined by the distance inequality

$$\sum_{j \in \mathcal{B} \cup \mathcal{G}} |x_j - x_j^*| \leq \sum_{j \in \mathcal{B} \cup \mathcal{G}} |\overline{x}_j - x_j^*|, \quad (7)$$

where again x^* and $\overline{x}$ denote the current LP solution of the LP relaxation and the incumbent, respectively. For a variable x_j such that $|\overline{x}_j - x_j^*| < 0.5$ the only way for the new solution to decrease its distance with respect to $\overline{x}_j$ is to hard fix it, that is, $x_j = \overline{x}_j$. For the other variables, instead, a *rebounding* phase is performed before the neighborhood is explored (as usual by means of a call to an

external MILP solver), in which the bounds on the variables are set so as the absolute value of the distance between the new solution and $\overline{x}$ cannot increase. A bit more of flexibility can be granted by allowing some of the variables to increase their distance but reducing the overall distance (as guaranteed by Equation 7). This is again obtained by a sophisticated mixture of hard and soft fixing.

The relaxation *enforced* neighborhood search (RENS) scheme by Berthold [24] is another RINS approach whose main characteristic is to define a reduced MILP (to be explored by a call to an external solver) as the set of all integer solutions which can be obtained by rounding a solution $x^* \in P$. More precisely, for a given x^* the reduced MILP is constructed by

1. *hard fixing* those integer-constrained variables whose current value is already integer, that is, $x_j = x_j^*, \forall j \in F$ where $F := \{j : x_j^* = \lfloor x_j^* \rfloor, j \in \mathcal{B} \cup \mathcal{G}\}$;
2. *rebounding* the remaining variables by using the two rounded values as lower ℓ_j and upper u_j bounds, that is, $\ell_j = \lfloor x_j^* \rfloor$ and $u_j = \lceil x_j^* \rceil$ for all $j \in (\mathcal{B} \cup \mathcal{G}) \setminus F$.

Interestingly, RENS does not require an incumbent solution to work with and it can be used offline as a tool for the analysis of the effectiveness of MILP rounding heuristics. Note that pivot-and-shift [5] uses a similar mechanism to restrict the range of the integer variables depending on their LP value and such a rebounding is similar to that of DINS as well.

An Evolutionary Algorithm within MILP

The farthest (to date) step in the direction of using metaheuristic ideas within an MILP has been taken by Rothberg [25]. The resulting approach, often called *polishing* algorithm, implements an evolutionary heuristic which is invoked at selected nodes of a branch-and-bound tree and includes all classical ingredients of genetic computation. More precisely,

- *Population.* A fixed-size population of feasible solutions is maintained. Those solutions are either obtained within the tree (by other heuristics, including diving) or computed by the polishing algorithm itself.
- *Combination.* Two or more solutions (the parents) are combined with the aim of creating a new member of the population (the child) with "improved" characteristics. The RINS scheme is adopted, that is, all variables whose value coincides in the parents are fixed and the reduced MILP is heuristically solved by an external MILP solver within a limited number of branch-and-bound nodes. This scheme, clearly, is much more time-consuming than a classical combination step in evolutionary algorithms, but guarantees that the child solution is feasible (for the original problem).
- *Mutation.* A sufficient degree of diversification is guaranteed by performing a classical mutation action which consists in (i) selecting at random a *seed* solution in the population, (ii) fixing at random some of its variables, and (iii) heuristically solve the resulting reduced MILP.
- *Selection.* The parents' selection is performed in a simple way by randomly picking a solution in the population and then choosing, again at random, the second one but only among those solutions with a better objective value. Although it is easy to extend this criterion to select more than two parents, in Rothberg [25] either two or all solutions are used as parents, that is, in the latter case no selection is performed.

The polishing algorithm is then a fully general metaheuristic implemented within an MILP framework and, in turn, making use of an external MILP solver as main "engine."

REFERENCES

1. Achterberg T. Constraint integer programming. PhD thesis, Berlin: ZIB; 2007.
2. Balas E, Martin CH. Pivot-and-complement: a Heuristic for 0-1 programming. Manage Sci 1980;26:86–96.

3. Løkketangen A, Jornsten K, Storoy S. Tabu search within a Pivot and complement framework. Int Trans Oper Res 1994;1:305–317.
4. Balas E, Martin CH. Pivot and shift: a heuristic for mixed integer programming. Technical report, GSIA, Carnegie Mellon University; 1986.
5. Balas E, Schmieta S, Wallace C. Pivot and shift - a mixed integer programming Heuristic. Disc Optim 2004;1:3–12.
6. Nediak M, Eckstein J. Pivot, cut, and dive: a Heuristic for 0-1 mixed integer programming. J Heuristics 2007;13:471–503.
7. Ghosh S, Hayward RB. Pivot and Gomory cut: a MILP feasibility Heuristic. Technical report, University of Alberta; 2009.
8. Hillier FS. Efficient Heuristic procedures for integer linear programming with an interior. Oper Res 1969;17:600–637.
9. Ibaraki T, Ohashi T, Mine H. A Heuristic algorithm for mixed-integer programming problems. Math Program Study 1974;2:115–136.
10. Faaland BH, Hillier FS. Interior path methods for heuristic integer programming procedures. Oper Res 1979;27:1069–1087.
11. Naoum-Sawaya J, Elhedhli S. An interior point cutting plane Heuristic for mixed integer programming. Technical report, University of Waterloo; 2010.
12. Balas E, Ceria S, Dawande M, *et al*. OCTANE: a new Heuristic for pure 0-1 programs. Oper Res 2001;49:207–225.
13. Fischetti M, Glover F, Lodi A. The feasibility pump. Math Program 2005;104:91–104.
14. Bertacco L, Fischetti M, Lodi A. A feasibility pump Heuristic for general mixed-integer problems. Disc Optim 2007;4:63–76.
15. Achterberg T, Berthold T. Improving the feasibility pump. Disc Optim 2007;4:77–86.
16. Fischetti M, Salvagnin D. Feasibility pump 2.0. Math Program Comput 2009;1:201–222.
17. Fischetti M, Lodi A. Local branching. Math Program 2003;98:23–47.
18. Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget JF, editors. Principles and practice of constraint programming CP98. Volume 1520, Lecture notes in computer science. Heidelberg: Springer; 1998. pp. 417–431.
19. Hansen P, Mladenović N, Urosević D. Variable neighborhood search and local branching. Comput Oper Res 2006;33:3034–3045.
20. Fischetti M, Polo C, Scantamburlo M. A local branching Heuristic for mixed-integer programs with 2-level variables. Networks 2004;44:61–72.
21. Fischetti M, Lodi A. Repairing MILP infeasibility through local branching. Comput Oper Res 2008;35:1436–1445.
22. Danna E, Rothberg E, Le Pape C. Exploring relaxation induced neighborhoods to improve MILP solutions. Math Program 2005;102:71–90.
23. Ghosh S. DINS, a MILP improvement Heuristic. In: Fischetti M, Williamson DP, editors. Integer programming and combinatorial optimization, IPCO 2007. Volume 4513, Lecture notes in computer science. Springer; 2007. pp. 310–323.
24. Berthold T. RENS - relaxation enforced neighborhood search. Technical Report 07-28. Berlin: ZIB; 2007.
25. Rothberg E. An evolutionary algorithm for polishing mixed integer programming solutions. Informs J Comput 2007;19:534–541.