Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization

Günther R. Raidl[1] and Jakob Puchinger[2]

[1] Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria,
raidl@ads.tuwien.ac.at
[2] NICTA Victoria Laboratory,
University of Melbourne, Melbourne, Australia,
jakobp@csse.unimelb.edu.au

Summary. Several different ways exist for approaching hard optimization problems. Mathematical programming techniques, including (integer) linear programming based methods, and metaheuristic approaches are two highly successful streams for combinatorial problems. These two have been established by different communities more or less in isolation from each other. Only over the last years a larger number of researchers recognized the advantages and huge potentials of building hybrids of mathematical programming methods and metaheuristics. In fact, many problems can be practically solved much better by exploiting synergies between these different approaches than by "pure" traditional algorithms. The crucial issue is *how* mathematical programming methods and metaheuristics should be combined for achieving those benefits. Many approaches have been proposed in the last few years. After giving a brief introduction to the basics of integer linear programming, this chapter surveys existing techniques for such combinations and classifies them into ten methodological categories.

1 Introduction

Computationally difficult *combinatorial optimization problems* (COPs) frequently appear in many highly important, practical fields. Creating good timetables, determining optimal schedules for jobs which are to be processed in a production line, designing efficient communication networks, container loading, determining efficient vehicle routes, and various problems arising in computational biology are a few examples. All these problems involve finding values for discrete variables such that an optimal solution with respect to a given objective function is identified subject to some problem specific constraints.

Most COPs are difficult to solve. In theoretical computer science, this is captured by the fact that many such problems are NP-hard [38]. Because of the inherent

difficulty and the enormous practical importance of NP-hard COPs, a large number of techniques for solving such problems has been proposed in the last decades. The available techniques for solving COPs can roughly be classified into two main categories: *exact* and *heuristic* algorithms. Exact algorithms are guaranteed to find an optimal solution and to prove its optimality for every instance of a COP. The run-time, however, often increases dramatically with a problem instance's size, and often only small or moderately-sized instances can be practically solved to proven optimality. For larger instances the only possibility is usually to turn to heuristic algorithms that trade optimality for run-time, i.e. they are designed to obtain good but not necessarily optimal solutions in acceptable time.

When considering exact approaches, the following techniques have had significant success: *branch-and-bound*, *dynamic programming*, *constraint programming*, and in particular the large class of *integer (linear) programming* (ILP) techniques including linear programming and other relaxation based methods, cutting plane and column generation approaches, branch-and-cut, branch-and-price, and branch-and-cut-and-price. See e.g. [52, 59] for general introductions to these mathematical programming techniques.

On the heuristic side, *metaheuristics* (MHs) have proven to be highly useful in practice. This category of problem solving techniques includes, among others, simulated annealing, tabu search, iterated local search, variable neighborhood search, various population-based models such as evolutionary algorithms, memetic algorithms, and scatter search, and estimation of distribution algorithms such as ant colony optimization. See Chap. 1 of this book as well as e.g. [42, 48] for more general introductions to metaheuristics.

Looking at the assets and drawbacks of ILP techniques and metaheuristics, the approaches can be seen as complementary to a large degree. As a matter of fact, it appears to be natural to combine ideas from both streams. Nevertheless, such hybrid approaches became more popular only over the last years. Nowadays, a multitude of recent publications describe different kinds of such hybrid optimizers that are often significantly more effective in terms of running time and/or solution quality since they benefit from synergy. International scientific events such as the *Hybrid Metaheuristics* workshop series [13, 12, 6], which started in 2004, and the *First Workshop on Mathematical Contributions to Metaheuristics – Matheuristics 2006* further emphasize the promise that is believed to lie in such hybrid systems. In fact, the artificial term "matheuristics" has been established by the latter event for referring to combinations of metaheuristics and mathematical programming methods.

In the next section, we will continue with a brief introduction of previously suggested structural classifications of strategies for combining metaheuristics and exact optimization techniques. Sect. 3 gives an overview on the basics of prominent ILP techniques and introduces used notations. Various different methodologies of utilizing ILP techniques in metaheuristics and vice versa, including annotated references to successful examples, are then reviewed in Sects. 4 to 13. These MH/ILP hybridization methodologies are

- MHs for finding high-quality incumbents and bounds in branch-and-bound
- relaxations for guiding metaheuristic search
- using the primal-dual relationship in MHs
- following the spirit of local search in branch-and-bound
- ILP techniques for exploring large neighborhoods
- solution merging

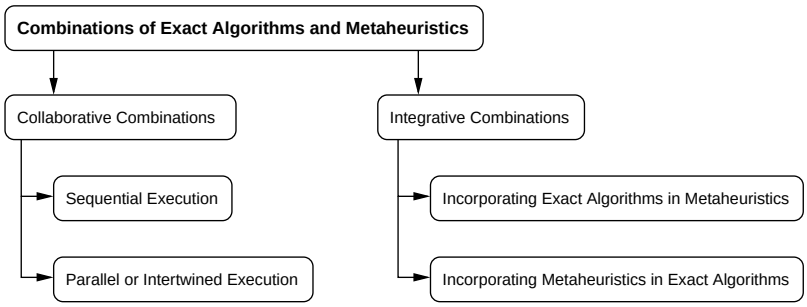

Fig. 1. Major structural classification of exact/metaheuristic combinations according to [67].

- ILP techniques as decoders for indirect or incomplete representations
- multi-stage approaches
- cut and column generation by metaheuristics
- strategic guidance of search and collaboration

2 Structural Models for Combining Metaheuristics with Exact Approaches

Overviews on various structural models of combining exact techniques and metaheuristics are given in [25, 67, 74].

Dumitrescu and Stützle [25] describe existing combinations which primarily focus on local search approaches that are strengthened by the use of exact algorithms. In their survey they concentrate on integration and exclude obvious combinations such as preprocessing.

In [67] we present a more general classification of existing approaches combining exact and metaheuristic algorithms for combinatorial optimization in which the following two main categories are distinguished, see also Fig. 1:

Collaborative Combinations. In a collaborative environment, the algorithms exchange information, but are not part of each other. Exact and heuristic algorithms may be executed sequentially, intertwined, or in parallel.

Integrative Combinations. In integrative models, one technique is a subordinate embedded component of another technique. Thus, there is a distinguished master algorithm, which can be either an exact or a metaheuristic algorithm, and at least one integrated slave.

Danna and Le Pape [21] present a similar classification of hybrid algorithms, further including constraint programming. The authors discern a *decomposition scheme* corresponding to the integrative combinations and a *multiple search scheme* corresponding to collaborative combinations. Four kinds of optimization algorithms are considered in particular, namely polynomial operations research algorithms, constraint programming, mixed integer programming, and various forms of local search

and metaheuristics. The main part of their article consists of examples from the literature illustrating six different collaborative schemes consisting of two of the above mentioned algorithm classes.

A taxonomy on hybrid metaheuristics in general has been proposed by Talbi [82]. Various hybridization schemes involving in particular *evolutionary algorithms* (EAs) are described by Cotta [19]. El-Abd and Kamel [26] particularly addressed cooperative parallel architectures.

Raidl [74] tries to unify previous classifications and taxonomies of hybrid metaheuristics and primarily distinguishes (a) the type of algorithms that are hybridized, (b) the level of hybridization (high- or low-level), (c) the order of execution (batch, interleaved, or parallel), and (d) the control strategy (integrative or collaborative).

3 Linear and Integer Programming at a Glance

This section gives a short overview of the main concepts in integer programming; for an in-depth coverage of the subject we refer to the books on linear optimization by Bertsimas and Tsitsiklis [11] and on combinatorial and integer optimization by Nemhauser and Wolsey [59] and Wolsey [88].

An integer (linear) program is an optimization problem involving integer variables, an objective function linearly depending on the variables, and a set of constraints expressed as linear (in)equalities. We consider the form

$$z_{\text{ILP}} = \min\{cx \mid Ax \geq b, x \geq 0, x \in \mathbb{Z}^n\}, \tag{1}$$

where x is the n-dimensional integer variable column vector and $c \in \mathbb{R}^n$ an n-dimensional row vector. Their dot-product cx is the *objective function* that should be minimized. Matrix $A \in \mathbb{R}^{m \times n}$ and the m-dimensional column vector $b \in \mathbb{R}^m$ together define m inequality constraints.

Maximization problems can be converted into minimization problems by simply changing the sign of c. Less-than constraints are similarly brought into greater-than-or-equal form by changing the sign of the corresponding coefficients, and equalities can be translated to pairs of inequalities. Thus, we can consider all kinds of linear constraints by appropriate transformations. Without loss of generality, we may therefore restrict our following considerations to minimization problems of the form (1).

A *mixed integer (linear) program* (MIP) involves a combination of integer and real-valued variables, but is otherwise defined in the same way.

3.1 Relaxations and Duality

One of the most important concepts in integer programming are *relaxations*, where some or all constraints of a problem are loosened or omitted. Relaxations are mostly used to obtain related, simpler problems which can be solved efficiently yielding bounds and approximate (not necessarily feasible) solutions for the original problem.

The *linear programming relaxation* of the ILP (1) is obtained by relaxing the integrality constraint, yielding the *linear program* (LP)

$$z_{\text{LP}} = \min\{cx \mid Ax \geq b, x \geq 0, x \in \mathbb{R}^n\}. \tag{2}$$

Large instances of such LPs can be efficiently solved in practice using simplex-based or interior-point algorithms. The linear programming relaxation always provides a lower bound for the original minimization problem, i.e. $z_{\mathrm{ILP}} \geq z_{\mathrm{LP}}$, since the search space of the ILP is contained in the one of the LP and the objective function remains the same.

According to linear programming theory, we can further associate a *dual problem* to each LP (2), which is defined by

$$w_{\mathrm{LP}} = \max\{ub \mid uA \leq c, u \geq 0, u \in \mathbb{R}^m\}. \tag{3}$$

The dual of the dual LP is the original (*primal*) LP again. Important relations between the primal problem and its dual are known as weak and strong duality theorems, respectively:

- The value of every finite feasible solution to the dual problem is a lower bound for the primal problem, and each value of a finite feasible solution to the primal problem is an upper bound for the dual problem. As a consequence, if the dual is unbounded, the primal is infeasible and vice versa.
- If the primal has a finite optimal solution z^*_{LP}, than its dual has the same optimal solution $w^*_{\mathrm{LP}} = z^*_{\mathrm{LP}}$ and vice versa.

The complementary slackness conditions follow from the strong duality theorem: Suppose x and u are feasible solutions for (2) and (3), respectively; then they are optimal if and only if the following conditions hold:

$$u(Ax - b) = 0 \quad \text{and} \tag{4}$$

$$x(c - uA) = 0. \tag{5}$$

In case of an integer linear problem, we have to differentiate between the notions of weak and strong duals. A *weak dual* of an ILP (1) is any maximization problem $w = \max\{w(u) \mid u \in S_{\mathrm{D}}\}$ such that $w(u) \leq cx$ for all $x \in \{Ax \geq b, x \geq 0, x \in \mathbb{Z}^n\}$. An obvious weak dual of (1) is the dual (3) of its LP relaxation (2). A *strong dual* w is a weak dual that further has an optimal solution u^* such that $w(u^*) = cx^*$ for an optimal solution x^* of (1). For solving ILPs, weak duals which are iteratively strengthened during the course of the optimization process are often utilized.

Another standard relaxation technique for ILPs, which often yields significantly tighter bounds than the LP relaxation, is *Lagrangian relaxation* [33, 34]. Consider the ILP

$$z_{\mathrm{ILP}} = \min\{cx \mid Ax \geq b, Dx \geq d, x \geq 0, x \in \mathbb{Z}^n\}, \tag{6}$$

where constraints $Ax \geq b$ are "nice" in the sense that the problem can be efficiently solved when the m' "complicating" constraints $Dx \geq b$ are dropped. Simply dropping these constraints of course yields a relaxation, however, the resulting bound will usually be weak due to the total ignorance of part of the inequalities. In Lagrangian relaxation, constraints $Dx \geq d$ are replaced by corresponding additional terms in the objective function:

$$z_{\mathrm{LR}}(\lambda) = \min\{cx + \lambda(d - Dx) \mid Ax \geq b, x \geq 0, x \in \mathbb{Z}^n\}. \tag{7}$$

Vector $\lambda \in \mathbb{R}^{m'}$ is the vector of Lagrangian multipliers, and for any $\lambda \geq 0$, $z_{\mathrm{LR}}(\lambda) \leq z_{\mathrm{ILP}}$, i.e. we have a valid relaxation of the ILP. We are now interested in finding

a specific vector λ yielding the best possible bound, which leads to the *Lagrangian dual problem*

$$z^*_{\mathrm{LR}} = \max_{\lambda \geq 0} \{z_{\mathrm{LR}}(\lambda)\}. \tag{8}$$

It can be shown that this Lagrangian dual is a piecewise linear and convex function, and usually, it can be well solved by iterative procedures like the subgradient method. A more elaborate algorithm that has been reported to converge faster on several problems is the volume algorithm [10], whose name is inspired by the fact that primal solutions are also considered, whose values come from approximating the volumes below active faces of the dual problem.

Given a solution λ to the Lagrangian dual problem (8) and a corresponding optimal solution x^* to the Lagrangian relaxation (7) which is also feasible to the original problem (6), i.e. $Dx^* \geq d$, the following complementary slackness condition holds: x^* is an optimal solution to the original problem (6) if and only if

$$\lambda(d - Dx^*) = 0. \tag{9}$$

It can be shown that the Lagrangian relaxation always yields a bound that is at least as good as the one of the corresponding linear relaxation, providing the Lagrangian dual problem is solved to optimality.

A third general-purpose relaxation technique for ILPs is *surrogate relaxation* [40]. Here, some or all constraints are scaled by surrogate multipliers and cumulated into a single inequality by addition of the coefficients. Similar as in Lagrangian relaxation, the ultimate goal is to find surrogate multipliers yielding the overall best bound. Unfortunately, this surrogate dual problem has not such nice properties as the Lagrangian dual problem and solving it is often difficult. However, if one is able to determine optimal surrogate multipliers, the bound obtained for the ILP is always at least as good as (and often better than) those obtained from linear and Lagrangian relaxation.

3.2 Cutting Plane Approach

When modeling COPs as ILPs, an important goal is to find a *strong* formulation, for which the LP relaxation provides a solution which lies in general not too far away from the integer optimum. For many COPs it is possible to strengthen an existing ILP formulation significantly by including further inequalities. Often, the number of such constraints grows exponentially with the problem size. This, however, means that already solving the LP relaxation by standard techniques might be too costly in practice due to the exponentially sized LP. Dantzig et al. [23] proposed the *cutting plane approach* for this purpose, which usually only considers a small subset of all constraints explicitly and nevertheless is able to determine an optimal solution to the whole LP.

This cutting plane approach starts with a small subset of initial inequalities and solves this reduced LP. Then, it tries to find inequalities that are not satisfied by the obtained solution but are valid for the original problem (i.e. contained in the full LP). These violated constraints are called *cuts* or *cutting planes*. They are added to the current reduced LP, and the LP is resolved. The whole process is iterated until no further cuts can be found. If the algorithm is able to provide a proof that

no further violated inequality exists, the finally obtained solution is also optimal with respect to the original full LP. The subproblem of identifying cuts is called *separation problem*, and it is of crucial importance to solve it efficiently, since many instances of it must usually be solved until the cutting plane approach terminates successfully.

Note that from a theoretical point of view it is possible to solve any ILP using a pure cutting plane approach with appropriate classes of cuts. There exist generic types of cuts, such as the Chvatal-Gomory cuts [88], which guarantee such a result. In practice, however, it may take a long time for such a cutting plane approach to converge to the optimum, partly because it is often a hard subproblem to separate effective cuts. The cutting plane method is therefore often combined with other methods, as we will see below.

3.3 Column Generation Approach

Instead of considering many inequalities, it is often also a reasonable option to formulate a problem in a strong way via a large number of variables, which correspond to columns in the coefficient matrix. The *(delayed) column generation approach* starts with a small subset of these variables and solves the corresponding restricted LP. Then, the algorithm tries to identify one or more not yet considered variables, whose inclusion might lead to an improved solution. This subproblem is called *pricing problem*, and for a minimization problem a variable is suitable in this sense if and only if it has negative reduced costs. After including such newly found variables in the restricted LP, the LP is resolved and the process iterated until it can be proven that no further variables with negative reduced costs exist, i.e. all variables *price out correctly*. An optimal solution for the original complete LP is then obtained. Column generation can be seen as the dual of the cutting plane approach, since inequalities correspond to variables in the dual LP.

A classical example where column generation is highly successful is the cutting stock problem [39]. A decision variable is defined for each possible cutting pattern, clearly yielding an exponential number of variables, and the pricing problem corresponds to the classical knapsack problem, which can be solved in pseudo-polynomial time. For a thorough review on column generation, we refer to [55].

A general technique for obtaining possibly strengthened ILP formulations is the Dantzig-Wolfe decomposition. It transforms original variables into linear combinations of extreme points and extreme rays of the original search space, yielding a potentially exponential number of variables. The resulting problems are usually solved by column generation.

3.4 Branch-and-Bound Methods

By solving the LP relaxation of an ILP problem, we usually only get a lower bound on the optimal integer solution value, and the solution will in general also contain fractional values. For hard COPs, this typically also holds for strengthened formulations and when cutting plane or column generation procedures have been applied, although the obtained bound might be much better. The standard way of continuing in order to finally determine an integer solution is *branch-and-bound* (B&B). This is a divide-and-conquer approach that solves an ILP by recursively splitting it into

disjoint subproblems. Bounds are calculated for the subproblems, and only those potentially holding an optimal solution are kept for further processing, whereas the others are pruned from the B&B tree.

The main idea in *LP-based B&B* is to use an LP relaxation of the ILP being solved in order to derive a lower bound for the objective function. A standard way for branching is to pick one of the fractional variables, say x_i with its current LP-value x_i^*, and define as first subproblem the ILP with the additional inequality $x_i \leq \lfloor x_i^* \rfloor$ and as second subproblem the ILP with inequality $x_i \geq \lceil x_i^* \rceil$. For these subproblems with the additional branching constraints, the LP is resolved, eventually leading to increased lower bounds. Usually, primal heuristics are also applied to each subproblem in order to possibly obtain an improved feasible solution and a corresponding global upper bound.

Combining B&B with cutting plane algorithms yields the highly effective class of *branch-and-cut algorithms* which are widely used in commercial ILP-solvers. Cuts are generated at the nodes of the B&B tree to tighten the bounds of the LP relaxations or to exclude infeasible solutions.

The combination of B&B with column generation results in *branch-and-price* algorithms, where new columns may be generated at each node in order to optimally solve their corresponding LP relaxations.

Finally, *branch-and-cut-and-price* refers to the combination of all of the above methods, often resulting in highly specialized and most powerful optimization algorithms.

We now turn to the different methodologies of hybridizing these ILP techniques (and some further mathematical programming approaches) with metaheuristics.

4 Metaheuristics for Finding High-Quality Incumbents and Bounds in B&B

Almost any effective B&B approach depends on some heuristic for deriving a promising initial solution, whose objective value is used as original upper bound. Furthermore, and as already mentioned, heuristics are typically also applied to some or all subproblems of the B&B tree in order to eventually obtain new incumbent solutions and corresponding improved upper bounds. In order to keep the B&B tree relatively small, good upper bounds are of crucial interest. Therefore, metaheuristics are often also applied for these purposes.

However, when performing a relatively expensive metaheuristic at each node of a large B&B tree in a straight-forward, independent way, the additional computational effort often does not pay off. Different calls of the metaheuristic might perform more or less redundant searches in similar areas of the whole search space. A careful selection of the B&B tree nodes for which the metaheuristic is performed and how much effort is put into each call is therefore crucial.

As an example, Woodruff [89] describes a chunking-based selection strategy to decide at each node of the B&B tree whether or not reactive tabu search is called. The chunking-based strategy measures a distance between the current node and nodes already explored by the metaheuristic in order to bias the selection toward distant points. Reported computational results indicate that adding the metaheuristic improves the B&B performance.

5 Relaxations for Guiding Metaheuristic Search

An optimal solution for a relaxation of the original problem often indicates in which areas of the original problem's search space good or even optimal solutions might lie. Solutions to relaxations are therefore frequently exploited in (meta-)heuristics. In the following, we study different possibilities for such approaches.

5.1 Creating Promising Initial Solutions

Sometimes an optimal solution to a relaxation can be repaired by a problem-specific procedure in order to make it feasible for the original problem and to use it as promising starting point for a subsequent metaheuristic (or exact) search. Often, the linear programming (LP) relaxation is used for this purpose, and only a simple rounding scheme is needed.

For example, Raidl and Feltl [75] describe a hybrid *genetic algorithm* (GA) for the generalized assignment problem, in which the LP relaxation of the problem is solved, and its solution is exploited by a randomized rounding procedure to create an initial population of promising integral solutions. These solutions are, however, often infeasible; therefore, randomized repair and improvement operators are additionally applied, yielding an even more meaningful initial population for the GA.

Plateau et al. [64] combine interior point methods and metaheuristics for solving the *multidimensional knapsack problem* (MKP). In a first step an interior point method is performed with early termination. By rounding and applying several different ascent heuristics, a population of different feasible candidate solutions is generated. This set of solutions is then used as initial population for a path-relinking/scatter search. Obtained results show that the presented combination is a promising research direction.

5.2 Guiding Repairing, Local Improvement, and Variation Operators

Beside initialization, optima of LP relaxations are often exploited for guiding local improvement or the repairing of infeasible candidate solutions. For example, in [73] the MKP is considered, and variables are sorted according to increasing LP-values. A greedy repair procedure considers the variables in this order and removes items from the knapsack until all constraints are fulfilled. In a greedy improvement procedure, items are considered in reverse order and included in the knapsack as long as no constraint is violated.

Many similar examples for exploiting LP solutions, also including a biasing of variation operators like recombination and mutation in EAs, exist.

5.3 Exploiting Dual Variables

Occasionally, dual variable values are also exploited. Chu and Beasley [15] make use of them in their GA for the MKP by calculating so-called *pseudo-utility ratios* for the primal variables and using them in similar ways as described above for the primal solution values. These pseudo-utility ratios tend to give better indications of the likeliness of the corresponding items to be included in an optimal solution; see [76] for more details on GA approaches for the MKP.

5.4 Variable Fixing: Reduction to Core Problems

Another possibility of exploiting the optimal solution of an LP relaxation is more direct and restrictive: Some of the decision variables having integral values in the LP-optimum are fixed to these values, and the subsequent optimization only considers the remaining variables. Such approaches are sometimes also referred to as *core methods*, since the original problem is reduced and only its "hard core" is further processed. Obviously, the selection of the variables in the core is critical.

The core concept has originally been proposed for the 0–1 knapsack problem [9] and also led to several very successful exact algorithms such as [63]. Puchinger et al. [72] extend this approach for the MKP and investigated several variants for choosing approximate cores. Considering binary decision variables $x_1, \ldots, x_n \in \{0, 1\}$, the basic technique first sorts all variables according to some specific efficiency measure and determines the so-called split-interval, which is the subsequence of the variables starting with the first and ending with the last fractional variable. Different efficiency measures are studied, and it is shown that the above already mentioned pseudo-utility ratios, which are determined from dual variable values, are in general a good choice for the MKP. The split interval is finally extended to an approximate core by adding $\delta > 0$ further variables on each side of the center of the split-interval. Empirical investigations in [72] indicate that already with $\delta = 0.1n$, high quality solutions with average optimality gaps less than 0.1% can be achieved when solving the remaining core problem to proven optimality. Applying an EA and relaxation guided variable neighborhood search to the reduced problem instances yields significantly better solutions in shorter time than when applying these metaheuristics to the original instances.

Staying with the MKP, another example for exploiting the LP relaxation within metaheuristics is the hybrid tabu search algorithm from Vasquez and Hao [86]. Here, the search space is reduced and partitioned via additional constraints fixing the total number of items to be packed. Bounds for these constraints are calculated by solving modified LP relaxations. For each remaining part of the search space, tabu search is independently applied, starting with a solution derived from the LP relaxation of the partial problem. The approach has further been improved in [87] by additional variable fixing. To our knowledge, this method is currently the one yielding the best results on a commonly used library of MKP benchmark instances.

5.5 Exploiting Lagrangian Relaxation

Also other relaxations besides the LP relaxation are occasionally successfully exploited in conjunction with metaheuristics. The principal techniques for such combinations are similar. A successful example is the hybrid Lagrangian GA for the prize collecting Steiner tree problem from Haouaria and Siala [47]. They perform a Lagrangian decomposition on a minimum spanning tree formulation of the problem and apply the volume algorithm for solving the Lagrangian dual. After termination, the genetic algorithm is started and exploits results obtained from the volume algorithm in several ways:

- Graph reduction: The volume algorithm creates a sequence of intermediate spanning trees as a by-product. All edges appearing in these intermediate trees are marked, and only this reduced edge set is further considered by the GA; i.e. a

core of edges is derived from the intermediate primal results when solving the Lagrangian dual.
- Initial population: A subset of diverse initial solutions is created by a Lagrangian heuristic, which greedily generates solutions based on the reduced costs appearing as intermediate results in the volume algorithm.
- Objective function: Instead of the original objective function, an alternate one is used, which is based on the reduced costs that are finally obtained by the volume algorithm. The idea is to guide the search into regions of the search space, where also better solutions with respect to the original objective function can presumably be found.

Pirkwieser et al. [62] described a similar combination of Lagrangian decomposition and a GA for the knapsack constrained maximum spanning tree problem. By Lagrangian relaxation, the problem is decomposed into a minimum spanning tree and a 0–1 knapsack problem. Again, the volume algorithm is employed to solve the Lagrangian dual. While graph reduction takes place as before, the objective function remains unchanged. Instead, final reduced costs are exploited for biasing the initialization, recombination, and mutation operators. In addition, the best feasible solution obtained from the volume algorithm is used as a seed in the GA's initial population. Results indicate that the volume algorithm alone is already able to find solutions of extremely high quality also for large instances. These solutions are polished by the GA, and in most cases proven optimal solutions are finally obtained.

6 Using the Primal-Dual Relationship in Metaheuristics

Using the primal-dual relationship in metaheuristics is a relatively recent approach; only a few papers have been published in this area. One idea is to take advantage of the complementary slackness conditions (5) or (9). Starting from a feasible dual solution u we try to find a primal feasible solution x satisfying these conditions with respect to u. On the other hand, if one searches in the dual as well as in the primal space, one may be able to give meaningful performance guarantees for heuristically obtained primal feasible solutions.

6.1 Generating Tight Bounds

Hansen et al. [44] present a primal-dual *variable neighborhood search* (VNS) for the *simple plant location problem* (SPLP). Since the tackled instances are too big to be solved by linear programming techniques, the authors propose to first perform a *variable neighborhood decomposition search* to the SPLP yielding a primal feasible solution. An initial, possibly infeasible, dual solution is then devised by exploiting the complementary slackness conditions. This solution is locally improved by applying *variable neighborhood descent* (VND), which also reduces a potential infeasibility. An exact dual solution is required to derive a correct lower bound for the SPLP. It is obtained by applying the recently developed sliding simplex method. The authors further use the generated bounds to strengthen a B&B algorithm exactly solving the SPLP. The presented computational experiments show the efficiency of the proposed approach, which is able to solve previously unsolved instances to proven optimality.

6.2 Integrating Primal and Dual Solution Approaches

Rego [77] describes a metaheuristic framework, called *relaxation adaptive memory programming* (RAMP), which combines principles of Lagrangian and surrogate relaxation with those of *adaptive memory programming* (AMP) [81]. He further proposes a primal-dual extension PD-RAMP and a specific implementation of PD-RAMP based on Lagrangian and surrogate constraint relaxation on the dual side and scatter search and path-relinking on the primal side.

Lagrangian and surrogate relaxation are combined into a cross-parametric relaxation method, which uses subgradient optimization to generate good surrogate constraints. Dual solutions are projected into the primal space by applying constructive and improvement heuristics. The approach yields primal solutions as well as dual bounds and may therefore be able to prove optimality or give performance guarantees for generated solutions. Using AMP for projecting solutions from the dual to the primal space yields the RAMP framework. The authors propose to use frequency based tabu search or a method were tabu search and path-relinking are combined. The primal-dual RAMP approach switches back and forth between a relaxation method and a path-relinking in the primal space, both updating the same reference set. The author describes preliminary computational experiments, where PD-RAMP is dominating the performance of the best known methods from the literature for different variants of the generalized assignment problem.

7 Following the Spirit of Local Search in B&B

Most metaheuristics are based on the principle of local search, i.e. starting from an initial solution, a certain neighborhood around it is investigated, and if a better solution can be identified, it becomes the new incumbent solution; this process is repeated. Thus, the central idea is to focus the search for better solutions on regions of the search space nearby already identified, good solutions.

In comparison, most B&B algorithms choose the next B&B tree node to be processed by a *best-first* strategy: a node with smallest lower bound is always selected, since it is considered to be most promising to contain an optimal solution. This approach is often the best strategy for minimizing the total number of nodes that need to be explored until finding an optimum and proving its optimality. However, good complete solutions and thus also tight upper bounds are often found late during this search. The best-first node selection strategy typically "hops around" on the search tree and in the search space, and does not stay focused on subregions. When no strong primal heuristic is applied for determining promising complete solutions, the best-first strategy is often combined with an initial *diving*, in which a depth-first strategy is followed at the beginning until some feasible solution is obtained. In depth-first search, the next node to be processed is always one that has been most recently been created by branching.

In the last years, several more sophisticated concepts have been proposed with the aim to intensify B&B-search in an initial phase to neighborhoods of promising incumbents in order to quickly identify high quality heuristic solutions. In some sense, we can consider these strategies to "virtually" execute a metaheuristic. We will review some of these strategies in the following.

7.1 Guided Dives

Danna et al. [22] describe *guided dives*, which are a minor, but effective modification of the already mentioned simple diving by temporarily switching to depth-first search. Consider a classical branching in LP-based B&B over a fractional variable, as described in Sect. 3.4. The subproblem to be processed next in case of guided dives is always the one in which the branching variable is allowed to take the value it has in a current incumbent solution. Diving is therefore biased towards the neighborhood of the given incumbent. Instead of performing only a single dive at the beginning, guided dives are repeatedly applied in regular intervals during the whole optimization. While this strategy is trivial to implement, experimental results indicate significant advantages over standard node selection strategies.

7.2 Local Branching

Fischetti and Lodi [31] propose *local branching*, an exact approach introducing the spirit of classical k-OPT local search in a generic branch-and-cut based MIP solver. They consider general MIPs with 0–1 variables. Let $x = (x_1, \ldots, x_n)$ be the vector of all variables and $\mathcal{B} \subseteq \{1, \ldots, n\}$ be the index set of the 0–1 variables. The following *local branching constraint* is used for defining a k-OPT neighborhood around a given incumbent solution $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$:

$$\Delta(x, \overline{x}) := \sum_{j \in \overline{\mathcal{S}}} (1 - x_j) + \sum_{x \in \mathcal{B} \setminus \overline{\mathcal{S}}} (x_j) \leq k, \tag{10}$$

where $\overline{\mathcal{S}} = \{j \in \mathcal{B} \mid \overline{x}_j = 1\}$ being the index set of 0–1 variables set to 1 in the incumbent solution. Note that $\Delta(x, \overline{x})$ resembles the classical Hamming distance between x and $\overline{x}$.

In the main algorithm, the whole problem is partitioned into the k-OPT neighborhood of an initial solution $\overline{x}$ and the rest by branching according to inequality (10) and the reverse constraint $\Delta(x, \overline{x}) \geq k + 1$, respectively. The MIP solver is then enforced to completely solve the k-OPT neighborhood before considering the rest.

If an improved solution $\overline{x}'$ has been found in the k-OPT neighborhood, a new subproblem $\Delta(x, \overline{x}') \leq k$ is split off from the rest and solved in the same way; this process is repeated until no further improvements can be achieved. Finally, the remaining problem corresponding to all not yet considered parts of the search space is processed in a standard way.

This basic mechanism is extended by introducing time limits, automatically modifying the neighborhood size k, and adding diversification strategies in order to improve performance. Furthermore, an extension of the branching constraint for general integer variables is also proposed. Reported results on various benchmark MIP instances using CPLEX[3] as MIP solver indicate the advantages of the approach in terms of an earlier identification of high-quality heuristic solutions.

Hansen et al. [46] present a variant of the local branching approach in which they follow more closely the standard VNS strategy [45] when switching between neighborhoods. Improved results are reported.

[3] http://www.ilog.com

Another variant of the original local branching scheme is described by Fischetti et al. [32]. They consider in particular problems in which the set of variables partitions naturally into two levels, with the property that fixing the values of the first-level variables yields a substantially easier subproblem.

Lichtenberger [53] describes an extended local branching framework in which several k-OPT neighborhoods induced by a set of candidate solutions can be processed in a pseudo-simultaneous (intertwined) way. This allows the "virtual" implementation of population-based metaheuristics like EAs on top of a B&B-based MIP solver. The framework was tested on the MKP. In order to keep the computational effort for processing the k-OPT neighborhoods reasonably low, an additional variable fixing strategy is applied.

7.3 The Feasibility Pump

Sometimes, it is already hard to identify any feasible initial solution for a MIP. For this purpose, Fischetti et al. [30] suggest an algorithm called *feasibility pump*. The method starts by solving the LP relaxation yielding a fractional solution x^*. A (usually infeasible) integer solution $\overline{x}$ is derived by simple rounding. From it, the nearest feasible point in the polytope defined by the LP relaxation is determined by solving a linear program with the Hamming distance $\Delta(x, \overline{x})$ as objective function. When the obtained solution is integral, a feasible solution for the original MIP has been found; otherwise, the process is repeated.

7.4 Relaxation Induced Neighborhood Search

Danna et al. [22] further suggest an alternative approach called *relaxation induced neighborhood search* (RINS) in order to explore the neighborhoods of promising MIP solutions more intensively. The main idea is to occasionally devise a sub-MIP at a node of the B&B tree that corresponds to a special neighborhood of an incumbent solution: First, variables having the same values in the incumbent and in the current solution of the LP relaxation are fixed. Second, an objective cutoff based on the objective value of the incumbent is set. Third, a sub-MIP is solved on the remaining variables. The time for solving this sub-MIP is limited. If a better incumbent could be found during this process, it is passed to the global MIP-search which is resumed after the sub-MIP termination. In the authors' experiments, CPLEX is used as MIP solver, and RINS is compared to standard CPLEX, local branching, combinations of RINS and local branching, and guided dives. Results indicate that RINS often performs best. The current version 10 of CPLEX also includes RINS as a standard strategy for quickly obtaining good heuristic solutions.

8 ILP Techniques for Exploring Large Neighborhoods

A common approach in more sophisticated local search based metaheuristics is to search neighborhoods by means of clever exact algorithms. If the neighborhoods are chosen appropriately, they can be relatively large and nevertheless an efficient search for the best neighbor is still reasonable. Such techniques are known as *very large-scale neighborhood* (VLSN) search [3]. Probably most of today's combinations

of local search based metaheuristics and ILP techniques follow this approach. In the following, we present some examples.

In *Dynasearch* [17, 18] exponentially large neighborhoods are explored by dynamic programming. A neighborhood where the search is performed consists of all possible combinations of mutually independent simple search steps, and one Dynasearch move corresponds to a set of independent moves that are executed in parallel in a single local search iteration. Independence in the context of Dynasearch means that the individual moves do not interfere with each other; in this case, dynamic programming can be used to find the best combination of independent moves. Dynasearch is restricted to problems where the single search steps are independent, and to our knowledge it has so far only been applied to problems where solutions are represented by permutations. Ergun and Orlin [28] investigated several such neighborhoods in particular for the traveling salesman problem.

For a class of partitioning problems, Thompson et al. [84, 85] suggest the concept of a cyclic exchange neighborhood, which is based on the transfer of single elements between an unrestricted number of subsets in a cyclic manner. A 2-exchange move can be seen as the simplest case of a cyclic exchange having length two. To efficiently determine a best cyclic exchange for a current solution, a weighted, directed graph is constructed, in which each arc represents a possible transfer of a single element and the arc's weight corresponds to the induced difference in the objective value of the solution. A best cyclic exchange can then be derived by finding a smallest negative-cost subset-disjoint cycle in this graph. The authors consider exact and heuristic methods for this purpose.

Puchinger et al. [71] describe a combined GA/B&B approach for solving a real-world glass cutting problem. The GA uses an order-based representation, which is decoded using a greedy heuristic. The B&B algorithm is applied with a certain probability enhancing the decoding phase by generating locally optimal subpatterns. Reported results indicate that the approach of occasionally solving subpatterns to optimality often increase the overall solution quality.

Büdenbender et al. [14] present a tabu search hybrid for solving a real-world direct flight network design problem. Neighborhoods are created by fixing a large subset of the integer variables corresponding to the performed flights and allowing the other variables to be changed. CPLEX is used to solve the reduced problems corresponding to these neighborhoods. Diversification is performed by closing flights frequently occurring in previously devised solutions.

Prandtstetter and Raidl [65] apply variable neighborhood search to the car sequencing problem and also use CPLEX for searching large neighborhoods. A subset of the scheduled cars is selected, removed from the schedule, and reinserted in an optimal way. The neighborhoods differ in the technique used to choose the cars and their number. Results indicate that this approach can compete well with leading algorithms from a competition organized by the French Operations Research Society ROADEF in 2005.

Hu et al. [49] propose a VNS metaheuristic for the generalized minimum spanning tree problem. The approach uses two dual types of representations and associated exponentially large neighborhoods. Best neighbors are identified by means of dynamic programming algorithms, and – in case of the so-called global subtree optimization neighborhood – by solving an ILP formulation with CPLEX. Experimental results indicate that each considered neighborhood contributes well to the

whole success, and the algorithm obtains significantly better solutions than previous metaheuristics.

Puchinger and Raidl [68] suggest a new variant of VNS: *relaxation guided variable neighborhood search*. It is based on the general VNS scheme and a new VND algorithm. The ordering of the neighborhood structures in this VND is determined dynamically by solving relaxations of them. The objective values of these relaxations are used as indicators for the potential gains of searching the corresponding neighborhoods. The proposed approach has been tested on the MKP. Computational experiments involving several ILP-based neighborhoods show that relaxation guided VNS is beneficial to the search, improving the obtained results. The concept is more generally applicable and seems to be promising for many other combinatorial optimization problems approached by VNS.

9 Solution Merging

In *evolutionary algorithms* (EAs), recombination is a traditionally essential operator. Its purpose is to derive a new candidate solution from two (or more) selected parental solutions by merging their attributes. Usually, this is done in a simple way, which is heavily based on random decisions. While such an operation is computationally cheap, created offspring is often worse than respective parent solutions, and many repetitions are typically necessary for achieving improvements.

As an alternative, one can put more effort into the determination of a new solution that is constructed entirely or mainly of attributes appearing in the parents. An established example from the domain of metaheuristics following this idea is *path-relinking* [43]. In the search space, this approach traces a path from one parent to another by always only exchanging a single attribute (or, more generally, performing a simple move towards the second parent). An overall best solution found on this path is finally taken as result.

This concept can further be extended by considering not just solutions on an individual path between two parents, but the whole subspace of solutions made up of parental properties only. An optimal *merging* operation returns a best solution from this set. Identifying such a solution often is a hard optimization problem on its own, but due to the limited number of different properties appearing in the parents, it can often be solved in reasonable time in practice.

Merging has already been successfully applied multiple times. Applegate et al. [7] were one of the first and describe such an approach for the traveling salesman problem. They derive a set of diverse tours by a series of runs of an iterated local search algorithm. The edge-sets of these solutions are merged and the traveling salesman problem is finally solved to optimality on this strongly restricted graph. In this way a solution is achieved that is typically superior to the best solution of the iterated local search.

Klau et al. [50] follow a similar idea and combine a memetic algorithm with integer programming to heuristically solve the prize-collecting Steiner tree problem. The proposed algorithmic framework consists of three parts: extensive preprocessing, a memetic algorithm, and an exact branch-and-cut algorithm applied as post-optimization procedure to the merged final solutions of the memetic algorithm.

Besides the one-time application of merging to a set of heuristically determined solutions, merging can also replace the classical crossover operator in EAs. Aggarwal

et al. [1] originally suggested such an approach for the independent set problem and called it *optimized crossover*. The subproblem of combining two independent sets to obtain the largest independent set in their union can be solved by an efficient algorithm.

Ahuja et al. [2] extend this concept to genetic algorithms for the quadratic assignment problem. They present a matching-based optimized crossover heuristic that finds an optimized child quickly in practice. This technique can also be applied to other assignment-type problems, as it relies on the structure of the problem rather than the objective function.

Cotta et al. [20] discuss the concept of merging in the light of a framework for hybridizing B&B with EAs. The authors recall the theoretical concepts on formal analysis (formae are generalized schemata), such as the dynastic potential of two chromosomes x and y, which is the set of individuals that only carry information contained in x and y. Based on these concepts the idea of dynastically optimal recombination is developed. This results in an operator exploring the potential of the recombined solutions using B&B, providing the best possible combination of the ancestors' features that can be attained without introducing implicit mutation. Extensive computational experiments on different benchmark sets show the usefulness of the approach.

Marino et al. [56] present an approach where a GA is combined with an exact method for the *linear assignment problem* (LAP) to solve the graph coloring problem. The LAP algorithm is incorporated into the crossover operator and generates an optimal permutation of colors within a cluster of nodes, thereby preventing the offspring from being less fit than its parents. The algorithm does not outperform other approaches, but provides comparable results. The main conclusion is that solving the LAP in the crossover operator strongly improves the performance of the GA in comparison to the GA using a classical crossover.

Clements et al. [16] propose a column generation approach in order to solve a production-line scheduling problem. Each feasible solution of the problem consists of a line-schedule for each production line. First, the *squeaky wheel optimization* (SWO) heuristic is used to generate feasible solutions to the problem. SWO is a heuristic using a greedy algorithm to construct a solution, which is then analyzed in order to find the problematic elements. Higher priorities, indicating that these elements should be considered earlier by the greedy algorithm, are assigned to them and the process restarts until a termination condition is reached. SWO is called several times in a randomized way in order to generate a set of diverse solutions. In the second phase, the line-schedules contained in these solutions are used as columns of a set-partitioning formulation for the problem, which is solved by a general purpose MIP solver. This process always provides a solution which is at least as good as, but usually better than the best solution devised by SWO. Reported results indicate that SWO performs better than a tabu search algorithm.

From a more theoretical point, Eremeev [27] studies the computational complexity of producing the best possible offspring in an optimized crossover for 0–1 ILPs. By means of efficient reductions of the merging subproblem, he shows the polynomial solvability for the maximum weight set packing problem, the minimum weight set partition problem, and for a version of the simple plant location problem.

For general mixed integer programming, Rothberg [79] describes a tight integration of an EA in a branch-and-cut based MIP solver. In regular intervals, a certain number of iterations of the EA is performed as B&B tree node heuristic. Recom-

bination follows the idea of solution merging by first fixing all variables that are common in selected parental solutions. The values of the remaining variables are then determined by applying the MIP solver to the reduced subproblem. Mutation is performed by selecting one parent, fixing a randomly chosen set of variables, and again solving the resulting reduced subproblem by the MIP solver. Since the number of variables to be fixed is a critical parameter, an adaptive scheme is used to control it. Performed experiments indicate that this hybrid approach is able to find significantly better solutions than other heuristic methods for several very difficult MIPs. The method is now also integrated in version 10 of the commercial MIP solver CPLEX.

Last but not least, it should be pointed out that there exists a strong relation between large neighborhood search and solution merging. In fact, solution merging can also be seen as exploring a large neighborhood defined by two or more parental solutions.

10 ILP Techniques as Decoders for Indirect or Incomplete Representations

Often, candidate solutions are only indirectly or incompletely represented in metaheuristics, and an "intelligent" decoding function is applied for determining an actual, complete solution. This in particular holds for many GAs. Sometimes, ILP techniques are successfully used for the decoding step.

It is relatively straight-forward to approach a MIP by splitting it into the integer and the continuous variable parts. One can then apply a metaheuristic to optimize the integer part only; before evaluating a solution, a linear programming solver is applied in order to augment the integer part with an optimal choice of continuous variable values. Such approaches are described in conjunction with GRASP by Net and Pedroso [60] and in conjunction with tabu search by Pedroso [61].

Glover [41] suggests a parametric tabu search for heuristically solving MIPs. This approach also makes use of an underlying LP-solver to obtain complete solution candidates. The current search point is indirectly represented by the LP relaxation of the MIP plus additional *goal conditions* that restrict the domains of a subset of the integer variables. These goal conditions are, however, not directly considered as hard constraints when applying the LP-solver, but are relaxed and brought into the objective function similarly as in Lagrangian relaxation. In this way, the approach can also be applied to problems where it is hard to find any feasible integer solutions (constraint satisfaction problems). Glover suggests a variety of intensification and diversification strategies based on adaptive tabu memory for making the heuristic search more efficient.

A more problem-specific example is the hybrid GA presented by Staggemeier et al. [80] for solving a lot-sizing and scheduling problem minimizing inventory and backlog costs of multiple products on parallel machines. Solutions are represented as product subsets for each machine at each period. Corresponding optimal lot sizes are determined when the solution is decoded by solving a linear program. The approach outperforms a MIP formulation of the problem directly solved by CPLEX.

11 Multi-Stage Approaches

Some optimization approaches consist of multiple sequentially performed stages, and different techniques are applied at the individual phases.

In many real-world applications, the problem naturally decomposes into multiple levels, and if the decision variables associated to the lower level(s) have a significantly weaker impact on the objective value than the higher-level variables, it is a reasonable approach to optimize the individual levels in a strictly sequential manner. Metaheuristics and ILP techniques can be considered and in combination be applied at the individual levels.

Multi-stage approaches are sometimes even applied when such a problem decomposition is not so obvious. For example, in Sect. 9, we considered approaches, where a metaheuristic is used to derive a set of heuristic solutions and an exact technique is used for merging them. Further examples are variable fixing strategies as described in Sect. 5.4.

Tamura et al. [83] tackle a job-shop scheduling problem and start from its ILP formulation. For each variable, they take the range of possible values and partition it into a set of subranges, which are then indexed. The encoded solutions of a GA are defined so that each position represents a variable, and its value corresponds to the index of one of the subranges. The fitness of such a chromosome is calculated using Lagrangian relaxation in order to obtain a bound on the optimal solution subject to the constraints that the values of the variables fall within the represented ranges. When the GA terminates, an exhaustive search of the region identified as the most promising is carried out to produce the final solution.

Lin et al. [54] propose an exact algorithm for generating the minimal set of affine functions that describes the value function of the finite horizon partially observed Markov decision process. In the first step a GA is used to generate a set Γ of witness points, which is as large as possible. In the second step a component-wise domination procedure is performed in order to eliminate redundant points in Γ. The set generated so far does not, in general, fully describe the value function. Therefore, a MIP is solved to generate the missing points in the final third step of the algorithm. Reported results indicate that this approach requires less time than some other numerical procedures.

Another kind of sequential combination of B&B and a GA has been described by Nagar et al. [58] for a two-machine flowshop scheduling problem in which solution candidates are represented as permutations of jobs. Prior to running the GA, B&B is executed down to a predetermined depth k and suitable bounds are calculated and recorded at each node of the explicitly stored B&B tree. During the execution of the GA each partial solution up to position k is mapped onto the corresponding tree node. If the associated bounds indicate that no path below this node can lead to an optimal solution, the permutation is subjected to a mutation operator that has been specifically designed to change the early part of the permutation in a favorable way.

12 Cut and Column Generation by Metaheuristics

In cutting plane and column generation based methods, which we addressed in Sects. 3.2 and 3.3, the dynamic separation of cutting planes and the pricing of

columns, respectively, is sometimes done by means of (meta-)heuristics in order to speed up the whole optimization process. We consider these hybrid approaches in the following in more detail.

12.1 Heuristic Cut Separation

In cutting plane and branch-and-cut algorithms, effective techniques are needed for deriving cuts, i.e. inequalities that are satisfied by feasible integer solutions but violated by the current solution to the LP relaxation. Although heuristic separation routines are commonly applied for this purpose, more sophisticated metaheuristics have only rarely been used.

An example is the work from Augerat et al. [8], who present a constructive algorithm, a randomized greedy method, and a tabu search for separating capacity constraints to solve a capacitated vehicle routing problem. The ILP formulation includes an exponential number of capacity constraints ensuring that for any given subset of customers S at least $\lceil \frac{d(S)}{C} \rceil$ vehicles are needed to satisfy the demand in S ($d(S)$ corresponds to the sum of the demands of the customers in set S and C is the capacity of one vehicle). A combination of a cutting plane algorithm and branch-and-bound is used to solve the problem optimally. The presented results indicate that using tabu search for identifying violated valid inequalities is promising and the use of metaheuristics in separation procedures is worth investigating.

Another example concerns the acceleration of Benders decomposition by local branching, as described by Rei et al. [78]. Benders decomposition is a promising solution approach in particular for MIPs with diagonal block structure. The basic principle is to project the MIP into the space of complicating integer variables only; real variables and the constraints involving them are replaced by corresponding constraints on the integer variables. These constraints, however, are not directly available but need to be dynamically separated in a cutting plane algorithm-like approach. According to the classical method, an optimal solution to the relaxed master problem (including only the already separated cuts) is needed and a linear program involving this solution must be solved in order to separate a single new cut. Rei et al. [78] improved this method by introducing phases of local branching on the original problem in order to obtain multiple feasible heuristic solutions. These solutions provide improved upper bounds on one hand, but also allow the derivation of multiple additional cuts before the relaxed master problem needs to be resolved. Tests on certain multicommodity flow formulations of a capacitated network design problem indicate the advantages over the traditional Benders decomposition approach.

12.2 Heuristic Column Generation

In column generation approaches and branch-and-price algorithms, it is important to have fast algorithms available for repeatedly solving the pricing subproblem, i.e. identifying a variable (column) with negative reduced costs. For many hard problems, however, this subproblem is also hard. Fast heuristics are therefore sometimes used for approaching the pricing problem. Note that it is fine when pricing in a column with negative reduced costs even when it is not one with minimum reduced costs. However, at the end of column generation it is necessary to prove that no further column with negative reduced costs exists, i.e. the pricing problem must

finally be solved exactly. Otherwise, no quality guarantees can be given for the final solution of the whole column generation or branch-and-price algorithm, and they must be considered to be heuristic methods only.

Most heuristic approaches for solving pricing problems are relatively simple construction methods. More sophisticated metaheuristics have so far been used less frequently. Filho and Lorena [29] apply a heuristic column generation approach to graph coloring. A GA is used to generate initial columns and to solve the pricing problem, which corresponds to the weighted maximum independent set problem, at every iteration. Column generation is performed as long as the GA finds columns with negative reduced costs. The master problem is solved using CPLEX. Some encouraging results are shown.

Puchinger and Raidl [66, 69] describe a branch-and-price approach for the three-stage two-dimensional bin packing problem. The pricing problem corresponds to the NP-hard three-stage two-dimensional knapsack problem with additional side-constraints coming from a special branching technique. Fast column generation is performed by applying a hierarchy of four methods: (a) a greedy heuristic, (b) an EA, (c) solving a restricted form of the pricing problem using CPLEX, and finally (d) solving the complete pricing problem using CPLEX. From this hierarchy, a strategy is always only applied when all lower level methods have been tried and were not successful in finding a column with negative reduced costs. Computational experiments on standard benchmark instances document the benefits of this fine-grained approach. The combination of all four pricing algorithms in the proposed branch-and-price framework yields the best results in terms of the average objective value, the average run-time, and the number of instances solved to proven optimality.

13 Strategic Guidance of Search and Collaboration

Last but not least, we consider approaches where metaheuristics are applied in order to explicitly guide ILP techniques and collaborative combinations where metaheuristics as well as ILP techniques provide each other mutual guidance.

13.1 Guidance of ILP Search

In principle, any metaheuristic that provides incumbent solutions to a B&B-based approach might already be considered to fall into this class of approaches; see also Sect. 4. Two more sophisticated methods, which go beyond this, are the following.

French et al. [35] suggest an EA/B&B hybrid to solve general ILPs. This hybrid algorithm combines the generic B&B of the MIP solver XPRESS-MP[4] with a steady-state EA. It starts with a B&B phase, in which information from the B&B tree nodes is collected in order to derive candidate solutions which are added to the originally randomly initialized EA-population. When a certain criterion is fulfilled, the EA takes over for a certain time using the augmented initial population. After termination of the EA, its best solutions are passed back and grafted onto the B&B tree. Full control is given back to the B&B-engine after the newly added nodes had been examined to a certain degree. Reported results on instances of the maximum

[4] http://www.dashoptimization.com/

satisfiability problem show that this hybrid approach yields better solutions than B&B or the EA alone.

Kotsikas and Fragakis [51] determine improved node selection strategies within B&B for solving MIPs by using genetic programming. After running B&B for a certain amount of time, information is collected from the B&B tree and used as a training set for genetic programming, which is performed to find a node selection strategy more appropriate for the specific problem at hand. The following second B&B phase then uses this new node selection strategy. Reported results show that this approach has potential, but needs to be enhanced in order to be able to compete with today's state-of-the-art node selection strategies.

13.2 Mutual Guidance

Several systems have been proposed where different optimization techniques, including metaheuristics and ILP methods, run in parallel or in an intertwined way and communicate with each other in order to provide mutual guidance.

Denzinger and Offerman [24] described a multi-agent based approach called TECHS (TEams for Cooperative Heterogenous Search). It consists of teams of one or more agents using the same search paradigm. Communication between the agents is controlled by so-called send- and receive-referees, in order to filter exchanged data. Each agent is in a cycle between searching and processing received information. In order to demonstrate the usefulness of TECHS, a system with multiple GA and B&B agents is considered for job-shop scheduling. GA and B&B agents exchange only positive information (solutions), whereas B&B agents can also exchange negative information (closed subtrees) among each other. Computational experiments show that this cooperation results in finding better solutions given a fixed time-limit and in finding solutions comparable to the ones of the best individual system alone in less total time.

Gallardo, Cotta, and Fernández [36] present another EA/B&B hybrid evaluated on the MKP. The algorithms are executed in an intertwined way and are cooperating by exchanging information. The EA provides bounds for B&B, while B&B provides best and partial solutions to the EA. In more detail, the EA is executed first until a certain convergence criterion is reached, yielding an initial bound. Then B&B is performed until it obtains an improved solution. Next, control is again given back to the EA, which possibly incorporates the new incumbent solution as well as some promising partial solutions from the ongoing B&B search into its population. Control is switched between the algorithms until a run-time limit is reached. Experimental results show that the collaborative approach yields better results than the individual techniques executed on their own.

In [37], the same authors described a refined variant of their approach, which uses beam search as truncated B&B. The method is also applied to the shortest common supersequence problem, where the results are again very encouraging.

Another cooperative approach involving a memetic algorithm and branch-and-cut has been described by Puchinger et al. [70] for the MKP. Both methods are performed in parallel and exchange information in a bidirectional asynchronous way. In addition to promising primal solutions, the memetic algorithm also receives dual variable values of certain LP relaxations and uses them for improving its repair and local improvement functions by updating the items' pseudo-utility ratios (see also Sect. 5.3).

The MALLBA project [4, 5] and its follow-up TRACER facilitate the direct development of parallel hybrid algorithms over local and wide area networks. It consists of a library of skeletons for combinatorial optimization, hiding complex parallelization and hybridization implementation details from the user. Several skeletons of exact and heuristic methods such as B&B, dynamic programming, tabu search, and GAs are available.

14 Conclusions

We have surveyed a multitude of examples where more powerful optimization systems were constructed by combining mathematical programming techniques and metaheuristics. Many very different ways exist for such hybridizations, and we have classified them into ten major methodological categories. The probably most traditional approach is to use some metaheuristic for providing high-quality incumbents and bounds to a B&B-based exact method. On the other hand, quickly solved relaxations or the primal-dual relationship are often used for guiding or narrowing the search in metaheuristics. A relatively new and highly promising stream are those methods in which B&B is modified in some way in order to follow the spirit of local search based metaheuristics. A nowadays frequently and successfully applied approach is large neighborhood search by means of ILP techniques. When extending this concept towards searching the neighborhood defined by the common and disjoint properties of two or more parental solutions, we come to solution merging approaches. Then, we have considered ILP techniques as decoders for indirect or incomplete representations. Furthermore, some problems are naturally approached by multi-stage approaches. So far less frequently applied, but in the opinion of the authors highly promising hybrid approaches are those where metaheuristics are utilized within more complex branch-and-cut and branch-and-price algorithms for cut separation and column generation, respectively. Last but not least we have considered collaborative hybrid systems in which one method provides some kind of strategic guidance for the other or even mutual guidance is achieved. As noted, some approaches from the literature can be considered to fall into several of the methodological categories we have identified.

Although a lot of experience already exists with such hybrid systems, it is usually still a tough question which algorithms and kinds of combinations are most promising for a new problem at hand. Despite the many successful examples of hybrids, the reader should also keep in mind that a more complex system does not automatically perform better than a simpler "pure" algorithm. Many less successful trials of combining mathematical programming techniques and metaheuristics also exist, but they are usually not published. The primary advice the authors are able to give for developing superior hybrid systems is to carefully study the literature looking for most successful approaches to similar problems and to adopt and eventually recombine (hybridize) their key-features. We hope that this chapter provides a good starting point and some references for this purpose.

Acknowledgements

This work is partly supported by the European RTN ADONET under grant 504438 and the "Hochschuljubiläumsstiftung" of Vienna, Austria, under contract number H-759/2005.

National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

1. C. Aggarwal, J. Orlin, and R. Tai. Optimized crossover for the independent set problem. *Operations Research*, 45:226–234, 1997.
2. R. Ahuja, J. Orlin, and A. Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 27:917–934, 2000.
3. R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.
4. E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, J. González, C. León, L. Moreno, J. Petit, J. Roda, A. Rojas, and F. Xhafa. MALLBA: Towards a combinatorial optimization library for geographically distributed systems. In *Proceedings of the XII Jornadas de Paralelismo*, pages 105–110. Editorial U.P.V., 2001.
5. E. Alba, F. A. M., Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, J. G. C., León, L. Moreno, J. Petit, J. Roda, A. Rojas, and F. Xhafa. MALLBA: A library of skeletons for combinatorial optimisation. In B. Monien and R. Feldman, editors, *Euro-Par 2002 Parallel Processing*, volume 2400 of *Lecture Notes in Computer Science*, pages 927–932. Springer, Berlin, Germany, 2002.
6. F. Almeida, M. J. Blesa Aguilera, C. Blum, et al., editors. *Hybrid Metaheuristics – Third International Workshop, HM 2006*, volume 4030 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 2006.
7. D. Applegate, R. Bixby, V. Chvátal, and W. Cook. On the solution of the traveling salesman problem. *Documenta Mathematica*, Extra Volume ICM III:645–656, 1998.
8. P. Augerat, J. Belenguer, E. Benavent, A. Corberan, and D. Naddef. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, 106(2):546–557, 1999.
9. E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.
10. F. Barahona and R. Anbil. The volume algorithm: Producing primal solutions with a subgradient method. *Mathematical Programming, Series A*, 87(3):385–399, 2000.
11. D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
12. M. J. Blesa Aguilera, C. Blum, A. Roli, et al., editors. *Hybrid Metaheuristics – Second International Workshop, HM 2005*, volume 3636 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 2005.
13. C. Blum, A. Roli, and M. Sampels, editors. *Hybrid Metaheuristics – First International Workshop, HM 2004*. Proceedings, Valencia, Spain, 2004.

14. K. Büdenbender, T. Grünert, and H.-J. Sebastian. A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. *Transportation Science*, 34(4):364–380, 2000.
15. P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.
16. D. Clements, J. Crawford, D. Joslin, G. Nemhauser, M. Puttlitz, and M. Savelsbergh. Heuristic optimization: A hybrid AI/OR approach. In A. Davenport and C. Beck, editors, *Proceedings of the Workshop on Industrial Constraint-Directed Scheduling*, 1997. Held in conjunction with the Third International Conference on Principles and Practice of Constraint Programming (CP97).
17. R. K. Congram. *Polynomially Searchable Exponential Neighbourhoods for Sequencing Problems in Combinatorial Optimisation*. PhD thesis, University of Southampton, Faculty of Mathematical Studies, UK, 2000.
18. R. K. Congram, C. N. Potts, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.
19. C. Cotta. A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Communications*, 11(3–4):223–224, 1998.
20. C. Cotta and J. M. Troya. Embedding branch and bound within evolutionary algorithms. *Applied Intelligence*, 18:137–153, 2003.
21. E. Danna and C. Le Pape. Two generic schemes for efficient and robust cooperative algorithms. In M. Milano, editor, *Constraint and Integer Programming*, pages 33–57. Kluwer Academic Publishers, 2003.
22. E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming, Series A*, 102:71–90, 2005.
23. G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large scale traveling salesman problem. *Operations Research*, 2:393–410, 1954.
24. J. Denzinger and T. Offermann. On cooperation between evolutionary algorithms and other search paradigms. In W. Porto et al., editors, *Proceedings of the 1999 Congress on Evolutionary Computation (CEC)*, volume 3, pages 2317–2324. IEEE Press, 1999.
25. I. Dumitrescu and T. Stuetzle. Combinations of local search and exact algorithms. In G. R. Raidl et al., editors, *Applications of Evolutionary Computation*, volume 2611 of *Lecture Notes in Computer Science*, pages 211–223. Springer, Berlin, Germany, 2003.
26. M. El-Abd and M. Kamel. A taxonomy of cooperative search algorithms. In Blesa Aguilera et al. [12], pages 32–41.
27. A. Eremeev. On complexity of optimized crossover for binary representations. In D. V. Arnold, T. Jansen, M. D. Vose, and J. E. Rowe, editors, *Theory of Evolutionary Algorithms*, number 06061 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
28. O. Ergun and J. B. Orlin. A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. *Discrete Optimization*, 3(1):78–85, 2006.
29. G. R. Filho and L. A. N. Lorena. Constructive genetic algorithm and column generation: an application to graph coloring. In L. P. Chuen, editor, *Proceedings of APORS 2000, the Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS*, 2000.

30. M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.
31. M. Fischetti and A. Lodi. Local Branching. *Mathematical Programming, Series B*, 98:23–47, 2003.
32. M. Fischetti, C. Polo, and M. Scantamburlo. Local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. *Networks*, 44(2):61–72, 2004.
33. M. L. Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 27(1):1–18, 1981.
34. A. Frangioni. About Lagrangian methods in integer optimization. *Annals of Operations Research*, 139(1):163–193, 2005.
35. A. P. French, A. C. Robinson, and J. M. Wilson. Using a hybrid genetic algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. *Journal of Heuristics*, 7:551–564, 2001.
36. J. E. Gallardo, C. Cotta, and A. J. Fernández. Solving the multidimensional knapsack problem using an evolutionary algorithm hybridized with branch and bound. In Mira and Álvarez [57], pages 21–30.
37. J. E. Gallardo, C. Cotta, and A. J. Fernández. On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(1):77–83, 2007.
38. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
39. P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.
40. F. Glover. Surrogate constraints. *Operations Research*, 16(4):741–749, 1968.
41. F. Glover. Parametric tabu-search for mixed integer programming. *Computers & Operations Research*, 33(9):2449–2494, 2006.
42. F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, 2003.
43. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–684, 2000.
44. P. Hansen, J. Brimberg, N. Mladenović, and D. Urosević. Primal-dual variable neighborhood search for the simple plant location problem. *INFORMS Journal on Computing*, to appear.
45. P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voß, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-heuristics: advances and trends in local search paradigms for optimization*, pages 433–438. Kluwer Academic Publishers, 1999.
46. P. Hansen, N. Mladenović, and D. Urosević. Variable neighborhood search and local branching. *Computers & Operations Research*, 33(10):3034–3045, 2006.
47. M. Haouaria and J. C. Siala. A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. *Computers & Operations Research*, 33(5):1274–1288, 2006.
48. H. Hoos and T. Stützle. *Stochastic Local Search – Foundations and Applications*. Morgan Kaufmann, 2004.
49. B. Hu, M. Leitner, and G. R. Raidl. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, to appear.

50. G. Klau, I. Ljubić, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, G. Raidl, and R. Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. In K. Deb et al., editors, *Genetic and Evolutionary Computation – GECCO 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 1304–1315. Springer, Berlin, Germany, 2004.
51. K. Kostikas and C. Fragakis. Genetic programming applied to mixed integer programming. In M. Keijzer et al., editors, *Genetic Programming – EuroGP 2004*, volume 3003 of *Lecture Notes in Computer Science*, pages 113–124. Springer, Berlin, Germany, 2004.
52. E. Lawler and D. Wood. Branch and bounds methods: A survey. *Operations Research*, 4(4):669–719, 1966.
53. D. Lichtenberger. An extended local branching framework and its application to the multidimensional knapsack problem. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria, March 2005.
54. A. Z.-Z. Lin, J. Bean, and C. C. White. A hybrid genetic/optimization algorithm for finite horizon partially observed Markov decision processes. *Journal on Computing*, 16(1):27–38, 2004.
55. M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.
56. A. Marino, A. Prügel-Bennett, and C. A. Glass. Improving graph colouring with linear programming and genetic algorithms. In K. Miettinen, M. M. Makela, and J. Toivanen, editors, *Proceedings of EUROGEN 99*, pages 113–118, Jyväskyiä, Finland, 1999.
57. J. Mira and J. Álvarez, editors. *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 2005.
58. A. Nagar, S. S. Heragu, and J. Haddock. A meta-heuristic algorithm for a bicriteria scheduling problem. *Annals of Operations Research*, 63:397–414, 1995.
59. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.
60. T. Neto and J. P. Pedroso. GRASP for linear integer programming. In J. P. Sousa and M. G. C. Resende, editors, *Metaheuristics: Computer Decision Making*, Combinatorial Optimization Book Series, pages 545–574. Kluwer Academic Publishers, 2003.
61. J. P. Pedroso. Tabu search for mixed integer programming. In C. Rego and B. Alidaee, editors, *Metaheuristic Optimization via Memory and Evolution*, volume 30 of *Operations Research/Computer Science Interfaces Series*, pages 247–261. Springer, Berlin, Germany, 2005.
62. S. Pirkwieser, G. R. Raidl, and J. Puchinger. Combining Lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. In C. Cotta and J. van Hemert, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2007*, volume 4446 of *Lecture Notes in Computer Science*, pages 176–187. Springer, Berlin, Germany, 2007.
63. D. Pisinger. An expanding-core algorithm for the exact 0–1 knapsack problem. *European Journal of Operational Research*, 87:175–187, 1995.
64. A. Plateau, D. Tachat, and P. Tolla. A hybrid search combining interior point methods and metaheuristics for 0–1 programming. *International Transactions in Operational Research*, 9:731–746, 2002.

65. M. Prandtstetter and G. R. Raidl. A variable neighborhood search approach for solving the car sequencing problem. In P. Hansen et al., editors, *Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search*, Tenerife, Spain, 2005.
66. J. Puchinger and G. R. Raidl. An evolutionary algorithm for column generation in integer programming: an effective approach for 2D bin packing. In X. Yao et al., editors, *Parallel Problem Solving from Nature – PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 642–651. Springer, Berlin, Germany, 2004.
67. J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, Part II*, volume 3562 of *Lecture Notes in Computer Science*, pages 41–53. Springer, Berlin, Germany, 2005.
68. J. Puchinger and G. R. Raidl. Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. *Journal of Heuristics*, to appear.
69. J. Puchinger and G. R. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, to appear.
70. J. Puchinger, G. R. Raidl, and M. Gruber. Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem. In *Proceedings of MIC 2005, the 6th Metaheuristics International Conference*, pages 775–780, Vienna, Austria, 2005.
71. J. Puchinger, G. R. Raidl, and G. Koller. Solving a real-world glass cutting problem. In J. Gottlieb and G. R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2004*, volume 3004 of *Lecture Notes in Computer Science*, pages 162–173. Springer, Berlin, Germany, 2004.
72. J. Puchinger, G. R. Raidl, and U. Pferschy. The core concept for the multidimensional knapsack problem. In J. Gottlieb and G. R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2006*, volume 3906 of *Lecture Notes in Computer Science*, pages 195–208. Springer, Berlin, Germany, 2006.
73. G. R. Raidl. An improved genetic algorithm for the multiconstrained 0–1 knapsack problem. In D. B. Fogel et al., editors, *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pages 207–211. IEEE Press, 1998.
74. G. R. Raidl. A unified view on hybrid metaheuristics. In Almeida et al. [6], pages 1–12.
75. G. R. Raidl and H. Feltl. An improved hybrid genetic algorithm for the generalized assignment problem. In H. M. Haddadd et al., editors, *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 990–995. ACM Press, 2004.
76. G. R. Raidl and J. Gottlieb. Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation Journal*, 13(4):441–475, 2005.
77. C. Rego. RAMP: A new metaheuristic framework for combinatorial optimization. In C. Rego and B. Alidaee, editors, *Metaheuristic Optimization via Memory and Evolution*, pages 441–460. Kluwer Academic Publishers, 2005.
78. W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. Technical Report C7PQMR PO2006-02-X, HEC Montréal, Canada, 2006.

79. E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, to appear.
80. A. T. Staggemeier, A. R. Clark, U. Aickelin, and J. Smith. A hybrid genetic algorithm to solve a lot-sizing and scheduling problem. In B. Lev, editor, *Proceedings of the 16th triannual Conference of the International Federation of Operational Research Societies*, Edinburgh, U.K., 2002.
81. E. D. Taillard, L.-M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive memory programming: A unified view of meta-heuristics. *European Journal of Operational Research*, 135:1–16, 2001.
82. E.-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–565, 2002.
83. H. Tamura, A. Hirahara, I. Hatono, and M. Umano. An approximate solution method for combinatorial optimisation. *Transactions of the Society of Instrument and Control Engineers*, 130:329–336, 1994.
84. P. Thompson and J. Orlin. The theory of cycle transfers. Technical Report OR-200-89, MIT Operations Research Center, Boston, MA, 1989.
85. P. Thompson and H. Psaraftis. Cycle transfer algorithm for multivehicle routing and scheduling problems. *Operations Research*, 41:935–946, 1993.
86. M. Vasquez and J.-K. Hao. A hybrid approach for the 0–1 multidimensional knapsack problem. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI 2001*, pages 328–333, Seattle, Washington, 2001. Morgan Kaufman.
87. M. Vasquez and Y. Vimont. Improved results on the 0–1 multidimensional knapsack problem. *European Journal of Operational Research*, 165:70–81, 2005.
88. L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.
89. D. L. Woodruff. A chunking based selection strategy for integrating metaheuristics with branch and bound. In S. Voss et al., editors, *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 499–511. Kluwer Academic Publishers, 1999.