Metaheuristic Hybrids

Günther R. Raidl, Jakob Puchinger, and Christian Blum

Abstract Over the last years, so-called hybrid optimization approaches have become increasingly popular for addressing hard optimization problems. In fact, when looking at leading applications of metaheuristics for complex real-world scenarios, many if not most of them do not purely adhere to one specific classical metaheuristic model but rather combine different algorithmic techniques. Concepts from different metaheuristics are often hybridized with each other, but they are also often combined with other optimization techniques such as branch-and-bound and methods from the mathematical programming and constraint programming fields. Such combinations aim at exploiting the particular advantages of the individual components, and in fact well-designed hybrids often perform substantially better than their "pure" counterparts. Many very different ways of hybridizing metaheuristics are described in the literature, and unfortunately it is usually difficult to decide which approach(es) are most appropriate in a particular situation. This chapter gives an overview on this topic by starting with a classification of metaheuristic hybrids and then discussing several prominent design templates which are illustrated by concrete examples.

Günther R. Raidl
Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria,
e-mail: raidl@ads.tuwien.ac.at

Jakob Puchinger
NICTA Victoria Laboratory, University of Melbourne, Australia,
e-mail: jakobp@csse.unimelb.edu.au

Christian Blum
ALBCOM Research Group, Universitat Politècnica de Catalunya, Barcelona, Spain,
e-mail: cblum@lsi.upc.edu

1 Introduction

Most of the other chapters of this book illustrate the existence of a large number of different metaheuristics. Simulated annealing, tabu search, evolutionary algorithms such as genetic algorithms and evolution strategies, ant colony optimization, particle swarm optimization, scatter search, path relinking, the greedy randomized adaptive search procedure, multi-start and iterated local search, and variable neighborhood search are—among others—prominent examples. Each of them has an individual historical background, follows certain paradigms and philosophies, and puts one or more particular strategic concepts in the foreground.

Over the last years a large number of algorithms were reported that do not purely follow the concepts of one single traditional metaheuristic, but they combine various algorithmic ideas, often originating from other branches of optimization and soft-computing. These approaches are commonly referred to as *metaheuristic hybrids* or *hybrid metaheuristics*. Note that the lack of a precise definition of these terms is sometimes subject to criticism. In our opinion, however, the relatively open nature of these terms is rather helpful, as strict borderlines between related fields of research are often a hindrance for creative thinking and the exploration of new research directions.

The motivation behind hybridizations of different algorithmic concepts is usually to obtain better performing systems that exploit and unite advantages of the individual pure strategies; i.e. such hybrids are believed to benefit from *synergy*. In fact, today it seems that choosing an adequate combination of multiple algorithmic concepts is the key for achieving top performance in solving most difficult problems. The vastly increasing number of reported applications of metaheuristic hybrids and dedicated scientific events such as the *Workshops on Hybrid Metaheuristics* [7, 15, 12, 22], the *Workshops on Matheuristics* [53, 64], and the conferences on the *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* [74] document the popularity, success, and importance of this specific line of research.

The idea of hybridizing metaheuristics is not new but dates back to their origins. At the beginning, however, such combinations were not very popular since several relatively strongly separated and sometimes competing communities of researchers existed who tended to consider "their" favorite class of metaheuristics generally superior to others and dogmatically followed their specific philosophies. For example the evolutionary computation community grew up quite isolated and followed quite strictly the biologically inspired thinking. The situation changed, according to many researchers, with the no free lunch theorems [104] when people recognized that there cannot exist a general optimization strategy which is always better than any other. In fact, solving a specific problem most effectively almost always requires a particularly tuned algorithm that is compiled of an adequate combination of sometimes very problem specific parts often originating from different metaheuristics and other algorithmic techniques. Exploiting problem specific knowledge in the best possible ways, picking the right algorithmic components, and combining them in the most appropriate way are key ingredients for leading optimization algorithms.

Unfortunately, developing a highly effective hybrid approach is in general a difficult task and sometimes even considered an art. Nevertheless, there are several strategies that have proven successful on many occasions, and they can provide some guidance. In the next section, we will start with a general classification of metaheuristic hybrids. The following sections will discuss the most prominent algorithmic templates of combinations and illustrate them with selected examples from the literature. For a more comprehensive review on hybrid metaheuristics, we recommend the book by Blum et al. [20].

2 Classification

Several classifications and taxonomies of hybrid metaheuristics can be found in the literature. Here we primarily follow the classification from Raidl [87] that combines aspects of the taxonomy introduced by Talbi [94] with the points-of-view from Cotta [29] and Blum et al. [21]. Differentiations with regard to parallel metaheuristics and hybridization of metaheuristics with exact optimization techniques are adopted from El-Abd and Kamel [37] and from Puchinger and Raidl [81], respectively. Figure 1 illustrates our classification.

We primarily distinguish hybrid metaheuristics according to four criteria, namely the kinds of algorithms that are hybridized, the level of hybridization, the order of execution, and the control strategy.

Hybridized algorithms. First, one might combine (parts of) different metaheuristic (MH) strategies, which is probably the most common approach. Second, highly problem specific algorithms, such as entire simulations for acquiring the quality of candidate solutions, are sometimes used in conjunction with metaheuristics. As a third class we consider the combination of metaheuristics with other more general techniques coming from fields like operations research (OR) and artificial intelligence (AI). Here, we can further distinguish between combinations with exact techniques or with other heuristics and soft-computing methods. Prominent examples for exact techniques that are often very successfully combined with metaheuristics are tree-search based methods such as branch-and-bound (B&B), dynamic programming, linear programming (LP) and mixed integer programming (MIP) methods as well as nonlinear programming techniques, and constraint programming (CP). For a survey dedicated to combinations of metaheuristics with MIP techniques see [88], for an overview on combinations of local search based methods with CP see [45], and for a review on combinations of local search methods with exact techniques see [36]. Examples of other heuristic and soft-computing techniques include neural networks, fuzzy logic, and several statistical techniques. As a fourth class we want to mention the combination of metaheuristics with a human interaction component, called *human guided search*. In particular for problems where it is difficult to quantify the quality of solutions in mathematically precise ways, where candidate solutions can be

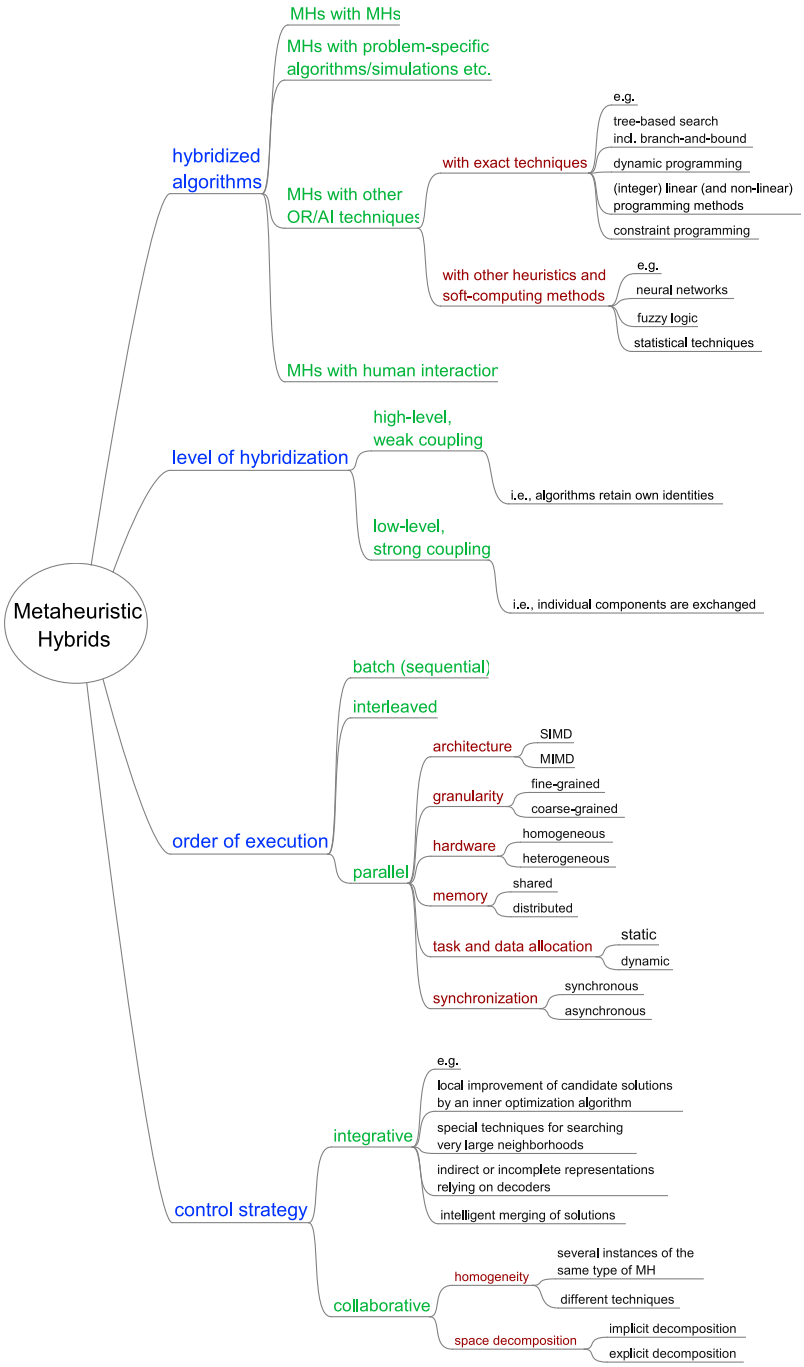

Fig. 1 Classification of metaheuristic (MH) hybrids based on Raidl [87].

well visualized, and where human intuition and intelligence present a particular advantage, such interactive systems are often highly effective in practice [60].

Level of hybridization. Hybrid metaheuristics can further be differentiated according to the *level* (or strength) at which the individual algorithms are coupled: *High-level* combinations retain in principle the individual identities of the original algorithms and cooperate over a relatively well defined interface; there is no direct, strong relationship of the internal workings of the algorithms. On the contrary, algorithms in *low-level* combinations strongly depend on each other; individual components or functions of the algorithms are exchanged.

Order of execution. In the simplest case, the *batch* execution, the individual algorithms are performed in a sequential way, and results of the first algorithm are used as input for the second. More sophisticated approaches apply the individual algorithms in an *intertwined* or even *parallel* way, and information is exchanged more frequently, usually in a bidirectional way. Parallel metaheuristics are an important research area by themselves and independent classifications of hybrid parallel approaches have been proposed in [6, 37]. They distinguish the following major criteria: (a) the architecture (SIMD: single instruction, multiple data streams versus MIMD: multiple instructions, multiple data streams), (b) the granularity of parallelization (fine- or coarse-grained), (c) the hardware (homogeneous or heterogeneous), (d) the memory (shared or distributed), (e) task and data allocation (static or dynamic), and (f) whether the parallel processes run asynchronously or are synchronized in some way.

Control strategy. Last but not least, we distinguish metaheuristic hybrids according to their control strategy, which can be either *integrative* (coercive) or *collaborative* (cooperative).

In the extremely popular integrative case, one algorithm is the subordinate, embedded component of another. Examples include the local improvement of candidate solutions by an inner optimization algorithm (as in memetic algorithms, see also Section 3), special techniques for searching large neighborhoods (see Section 9.1), indirect or incomplete representations relying on decoders (see Section 5), and intelligent merging (recombination) of solutions (see Section 6).

In contrast, in collaborative approaches the individual algorithms exchange information but are not part of each other. For example, the popular island model [26] for parallelizing evolutionary algorithms is of this type. Collaborative approaches can further be classified into homogeneous approaches, where several instances of one and the same algorithm are performed (as in traditional island models), and heterogeneous approaches. An example for the latter are *asynchronous teams* (A-Teams) [95]: An A-Team consists of a collection of agents and memories connected into a strongly cyclic directed network. Each of these agents is an optimization algorithm that works asynchronously on the target problem, on a relaxation of it, i.e. a superproblem, or on a subproblem. Information is exchanged via shared memory. Denzinger and Offermann [32] presented a similar multi-agent approach for achieving cooperation between search algorithms following differ-

ent search paradigms, such as B&B and evolutionary algorithms. Especially in collaborative combinations, another question is which search spaces are actually explored by the individual algorithms. Implicit decomposition results from different initial solutions, parameter settings, or random decisions, while an explicit decomposition is obtained when each algorithm works on its individually defined subspace. Effectively decomposing large problems is often an important issue in practice. Occasionally, problems decompose in a relatively natural way, see Sections 4 and 9, but most often finding a strong decomposition into weakly related or even unrelated subproblems is a difficult task, and (self-)adaptive schemes are sometimes applied.

Starting with the next section, we will consider several templates of implementing metaheuristic hybrids, which have successfully been applied on many occasions.

3 Finding Initial or Improved Solutions by Embedded Methods

The most natural way of hybridizing two optimization algorithms is probably to embed one algorithm into another for obtaining either promising starting solutions or for possibly improving intermediate solutions.

Problem-specific construction heuristics are often used for finding initial solutions which are then further improved by local search or metaheuristics. A frequently applied and more general strategy for obtaining initial solutions is to solve a relaxation of the original problem (e.g. the LP relaxation) and eventually repair the obtained solution in some heuristic way. Examples of such approaches can also be found in Section 7.

The *Greedy Randomized Adaptive Search Procedure* (GRASP) [40] systematically extends the principle of locally improving a starting solution by iterating a randomized construction process, and each of the resulting solutions is then used as a starting point for local search.

The so-called *proximate optimality principle* (POP) was first mentioned by Glover and Laguna in the context of tabu search [48]. It refers to the general intuition that good solutions are likely to have a similar structure and can therefore be found close to each other in the search space. Fleurent and Glover transfered this principle in [44] from complete to partial solutions in the context of GRASP. They suggested that mistakes introduced during the construction process may be undone by applying local search during (and not only at the end of) the GRASP construction phase. They proposed a practical implementation of POP in GRASP by applying local search at a few stages of the construction phase only. Another application of this concept can be found in [14] for the job shop scheduling problem.

Often local search procedures or more sophisticated improvement algorithms are applied within an outer metaheuristic for “fine-tuning” intermediate candidate solutions. While the outer metaheuristic is responsible for diversification, the inner improvement algorithm focuses on intensification. For example, most *memetic al-*

gorithms [69] rely on this principle: The outer metaheuristic is an evolutionary algorithm, and intermediate candidate solutions are locally improved. If each intermediate solution is always turned into a local optimum, the evolutionary algorithm exclusively searches the space of local optima (w.r.t. the neighborhood structure of the inner local improvement procedure) only. Memetic algorithms are often more successful than simple evolutionary algorithms, because intensification is typically considered a weakness of traditional evolutionary algorithms. By adjusting how much effort is spent in the local improvement, one can tune the balance between intensification and diversification. Note that the inner local improvement does not always have to be just a simple local search. Occasionally, more sophisticated strategies like tabu search or even exact techniques for solving a restricted problem are applied. This also leads to the related *large neighborhood search* methods, which we will consider in Section 9.1.

Another example is *Variable Neighborhood Search* (VNS) [55], where each candidate solution undergoes some kind of local improvement and a hierarchy of different neighborhood structures is utilized. Especially in *general variable neighborhood search*, a more sophisticated *variable neighborhood descent* is uses as the inner local improvement procedure.

Considering exact techniques, B&B approaches strongly rely on good upper and lower bounds in order to prune the search tree as strongly as possible. Metaheuristic techniques are frequently applied to obtain a promising initial solution or to improve intermediate solutions in order to find tight(er) bounds. Sections 6 and 8 contain several examples such as [31, 91] that also fall into this category.

4 Multi-Stage Approaches

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

In many complex real-world applications, the problem naturally decomposes into multiple levels. If the decision variables associated with the lower level(s) have a significantly weaker impact on the objective value than the higher-level variables or if the impact of these variable sets is only loosely correlated, it is a very reasonable approach to optimize the individual levels in a strictly sequential manner. Different techniques can be applied at the individual levels yielding simple but often very effective hybrid approaches.

For example, for certain vehicle routing applications where the aim is to deliver goods to customers, it may be a meaningful approach to first partition the customers into groups which are then independently treated by finding appropriate delivery tours; finally, the drivers' specific time-plans are derived from these tours. For certain job scheduling problems, it might be feasible to first assign the jobs to machines and then independently optimize the schedules for each machine. For large communication network design problems it might be wise to first optimize a possibly redundant backbone infrastructure, then design the individual local access network

structures, and finally decide about the concrete cable laying and technical parameters such as the capacities of the individual links.

We remark that in practice such multi-stage approaches will usually not lead to optimal solutions, as the sequentially solved subproblems are typically not independent. However, for many complicated real-world problems of large size, as for example when designing a communication infrastructure for a larger city, a multi-stage approach is the only viable choice. Furthermore, multi-stage approaches are often very meaningful for relatively quickly finding first approximate solutions. Therefore, they are frequently used in practice.

Multi-stage approaches are sometimes even applied when such a problem decomposition is not so obvious but results in algorithmic advantages. Classical preprocessing techniques, where the problem is usually reduced to a hard-to-solve core by applying certain problem specific simplification strategies, are an example.

A more general, systematic approach is based on tools from the field of parameterized complexity. It offers both a framework of complexity analysis and toolkits for algorithm design. One of the tools for algorithm design is known as *problem kernelization*. The idea is to reduce a given problem instance in polynomial time to a so-called problem kernel such that an optimal solution to the problem kernel can, in polynomial time, be transformed into an optimal solution to the original problem instance. In [47], Gilmour and Dras propose several different ways of using the information given by the kernel of a problem instance for making ant colony system more efficient for solving the minimum vertex cover problem. The most intuitive version applies the ant colony system directly to the problem kernel and subsequently transforms the best solution obtained for the kernel into a solution to the original problem instance.

Multi-level refinement strategies [101] can also be considered as a special class of multi-stage approaches. They involve a recursive coarsening to create a hierarchy of approximations to the original problem. An initial solution is identified for the coarsest level and is then iteratively refined at each level—coarsest to finest—typically by using some kind of (meta-)heuristic at each level. *Solution extension* operators transfer the solution from one level to the next. In *iterated multi-level algorithms*, solutions are not just refined but occasionally also re-coarsened, and the refinement process is iterated. These strategies have been successfully applied on several problems including multilevel graph partitioning, graph coloring, very large traveling salesman problems, vehicle routing, and DNA sequencing.

Variable fixing strategies where variables of the original problem are fixed to certain values (according to some, usually heuristic, criterion) to perform the optimization over a restricted search space are also related to the above mentioned strategies. Examples of effective variable fixing strategies are the *core concepts* for knapsack problems [77, 85].

Some approaches determine a set of (complete) initial solutions by a first stage method and apply one (or even more) other technique(s) for further improving upon them. For example, occasionally a metaheuristic is used for finding a pool of diverse high-quality solutions, and *merging* is performed to identify a single final solution

combining the particularly beneficial properties of the intermediate solutions. We will consider merging in more detail in Section 6.

Tamura et al. [96] 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 genetic algorithm (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 constraints on the variable values which must fall within the represented ranges. When the GA terminates, an exhaustive search of the region identified as the most promising is carried out in a second stage.

A special kind of a sequential combination of B&B and a GA is described by Nagar et al [70] for a two-machine flow-shop scheduling problem. Candidate solutions are represented as permutations of jobs. In the first stage, B&B is executed down to a predetermined branching 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 in the second stage, 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 undergoes a mutation that has been specifically designed to change the early part in a favorable way.

In [99], Vasquez and Hao present a two-stage approach for tackling the 0-1 multi-dimensional knapsack problem (MKP). Given n items and m resources, each object has an associated profit c_i and resource consumptions $a_{i,j}$, $\forall i = 1,\ldots,n$, $\forall j = 1,\ldots,m$, and each resource has a capacity b_j. The goal of the MKP is to choose a subset of the n objects such that its total profit is maximized without violating the capacity constraints. In the ILP formulation of the MKP, a binary variable $x_i \in \{0,1\}$ is defined for each object. In the first stage of the proposed hybrid solution method a series of LP relaxations with additional constraints is solved. They are of the form $\sum_{i=1}^{n} x_i = k$ where $k \in \{k_{\min},\ldots,k_{\max}\}$, i.e. the number of items to be selected is fixed to k. Each setting of k defines an LP that is solved to optimality. In the second stage of the process, tabu search is used to search for better solutions around the usually non-integral optimal solutions of the $k_{\max} - k_{\min} + 1$ LPs. The approach has been improved in [100] by additional variable fixing.

A combination of tabu search and CP for the job-shop scheduling problem is presented by Watson and Beck [102]. First an iterated simple tabu search [103] is run for a limited time. The best resulting solutions are used for initializing a *solution-guided multi-point constructive search* (SGMPCS) [13]. SGMPCS is a recent algorithm combining CP tree search with the population concept of metaheuristics. In the CP tree search, a single variable is chosen (variable-ordering heuristic) and assigned a value (value-ordering heuristic). The domains of the remaining variables are reduced accordingly and another variable is chosen, repeating the process as long as unassigned variables exist or until the partial solution becomes infeasible, in which case backtracking to the last choice-point is performed. The main idea of SGMPCS is to perform a standard tree search that is restarted if the number of failures exceeds a certain limit. The value-ordering heuristic is partly determined by an

incumbent solution: If the value of the chosen variable in this solution is allowed by its current domain, the variable is assigned that value; else another value is chosen using another kind of value-ordering heuristic. SGMPCS maintains a small set of solutions which is updated in an elitist way. The tree search is restarted with a randomly chosen solution from the set until a stopping criterion is met.

5 Decoder Based Approaches

The hybrid metaheuristic design template considered in this section is particularly popular for problems where solutions must fulfill certain constraints and a fast construction heuristic yielding feasible solutions exists. In *decoder based* approaches, a candidate solution is represented in an indirect or incomplete way and a problem specific decoding algorithm is applied for transforming the encoded solution into a complete feasible solution. This principle is often applied in evolutionary algorithms, where encoded solutions are denoted as *genotypes* and the decoded counterparts are called *phenotypes* [51].

A prominent, quite generic way of indirectly representing solutions is by means of *permutations* of solution attributes. The decoder is then usually a greedy construction heuristic which composes a solution by trying to add the solution attributes in the order given by the permutation, i.e. the order of an attribute in the permutation is the greedy criterion. Cutting and packing problems are examples where such decoder based methods are frequently used [59]. The overall performance obviously depends strongly on the quality and the speed of the decoder. Such approaches are often straightforward and relatively easy to implement, in particular as standard metaheuristics with traditional neighborhoods for permutations can directly be applied. On the downside, more elaborate metaheuristics based on direct encodings and tuned problem specific operators are often likely to achieve better performance, as they might exploit problem specific features in better ways.

Especially attractive are decoder based approaches where the decoder is a more sophisticated algorithm rather than a simple construction procedure. For example, a mixed integer linear programming problem (MIP) can be approached by splitting the variables into the integral and continuous parts. One can then apply a metaheuristic to optimize the integer part only; for each candidate solution, corresponding optimal fractional variable values are efficiently determined by solving the remaining LP. Such approaches are described in conjunction with GRASP by Neto and Pedroso [71] and in conjunction with tabu search by Pedroso [73].

Glover [49] proposed an alternative *parametric tabu search* for heuristically solving MIPs. A current search point is indirectly represented by the solution to the LP relaxation of the MIP plus additional *goal conditions* that restrict the domains of a subset of the integer variables. Instead of considering the goal conditions directly as hard constraints when applying an LP solver, they 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 solution.

The approach further uses a variety of intensification and diversification strategies based on adaptive tabu memory in order to make the heuristic search more efficient.

Besides problem specific heuristics and LP solvers, other efficient techniques are sometimes used as a decoder to augment incompletely represented solutions. For example, Hu and Raidl [58] consider the generalized traveling salesman problem in which a clustered graph is given and a shortest tour visiting exactly one node from each cluster is requested. Their approach is based on VNS and represents a candidate solution in two different ways: On the one hand, a permutation of clusters is given, representing the order in which the clusters are to be visited. A dynamic programming procedure is used as decoder to derive a corresponding optimal selection of particular nodes. On the other hand, only the unordered set of selected nodes from each cluster is given, and the classical chained Lin-Kernighan heuristic for the traveling salesman problem [66] is uses as a decoder to obtain a corresponding high-quality tour. The VNS uses several types of neighborhood structures defined for each representations.

Last but not least, decoder-based approaches have recently been used in ant colony optimization (ACO). For example, Blum and Blesa [19] present a decoder-based ACO for the general k-cardinality tree problem. Given an undirected graph, this problem involves finding among all trees with exactly k edges a tree such that a certain objective function is minimized. In contrast to a standard ACO algorithm that constructs trees (i.e. solutions) with exactly k edges, the decoder-based approach of [19] builds l-cardinality trees, where $l > k$. Subsequently, an efficient dynamic programming algorithm is applied for finding the best k-cardinality tree that is contained in the l-cardinality tree. Results show that this approach has clear advantages over standard ACO approaches.

6 Solution Merging

The basic idea of *solution merging* is to derive a new, hopefully better solution from the attributes appearing in two or more promising input solutions. The observation that high-quality solutions usually have many attributes in common is exploited.

In the simplest form, this operation corresponds to the classical recombination (crossover) which is considered the primary operator in GAs: Usually two parent solutions are selected and an offspring is constructed by inheriting attributes from both of them based on random decisions. While such an operation is computationally cheap, created offspring are often worse than the respective parents, and many repetitions are usually necessary for achieving strong improvements.

Alternatively, one can put more effort into the determination of such offspring. An established technique is *path relinking* [50]. It traces a path in the search space from one parent to a second by always exchanging only a single attribute (or more generally by performing a move in a simple neighborhood structure towards the target parent). An overall best solution found on this path is finally taken as offspring.

This concept can further be extended by considering not just solutions on an individual path between two parents, but the whole subspace of solutions defined by the joined attributes appearing in a set of two or more input solutions. An *optimal merging* operation returns a best solution from this subspace, i.e. it identifies a best possible combination of the ancestors' features that can be attained without introducing new attributes. Depending on the underlying problem, identifying such an optimal offspring is often 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.

Applegate et al. [8, 9] were among the first to apply more sophisticated merging in practice. For the traveling salesman problem, they derive a set of different tours by a series of runs of the chained Lin-Kernighan iterated local search algorithm. The sets of edges of all these solutions are merged and the traveling salesman problem is finally solved to optimality on this strongly restricted graph. Solutions are achieved that are typically superior to the best ones obtained by the iterated local search.

Besides the one-time application of merging to a set of heuristically determined solutions in a multi-stage way, sophisticated merging can also replace classical recombination in evolutionary and memetic algorithms. Aggarwal et al. [1] originally suggested such an approach for the independent set problem. The subproblem of identifying the largest independent set in the union of two parental independent sets is solved exactly by an efficient algorithm. Ahuja et al. [2] apply this concept to a GA for the quadratic assignment problem. As the optimal recombination problem is more difficult in this case, they use a matching-based heuristic that quickly finds high-quality offspring solutions. Optimal merging is also used by Blum [17] in the context of an evolutionary algorithm for the k-cardinality tree problem. The individuals are trees with k edges. Crossover first combines two parent trees, producing hereby a larger l-cardinality tree. Dynamic programming is then used to reduce this tree to the best feasible subtree with k edges.

Eremeev [38] studies the computational complexity of producing a best possible offspring from two parents for binary representations from a theoretical point of view. He concludes that the optimal recombination problem is polynomially solvable for the maximum weight set packing problem, the minimum weight set partition problem, and linear Boolean programming problems with at most two variables per inequality. On the other hand, determining an optimal offspring is NP-hard for 0/1 integer programming with three or more variables per inequality, like the knapsack, set covering, and p-median problems, among others.

Cotta and Troya [30] discuss merging in the light of a more general framework for hybridizing B&B and evolutionary algorithms. They show the usefulness of applying B&B for identifying optimal offspring on various benchmarks.

For mixed integer programming, Rothberg [91] suggests a tight integration of an evolutionary algorithm in a branch-and-cut based MIP solver. At regular intervals the evolutionary algorithm is applied as a B&B tree node heuristic. Optimal recombination is performed by first fixing all variables that are common in the selected parental solutions and by applying the MIP solver to this reduced subproblem. Mutation selects one parent, fixes a randomly chosen subset of variables, and calls the

MIP solver for determining optimal values for the remaining problem. Since the number of variables to be fixed is a critical parameter, an adaptive scheme is applied to control it. Experimental results indicate that this hybrid is able to find significantly better solutions than other heuristic methods for several very difficult MIPs. This method is integrated in the commercial MIP solver CPLEX[1] since version 10.

7 Strategic Guidance of Metaheuristics by Other Techniques

Many successful hybrid metaheuristics use other optimization techniques for guiding the search process. This may be done by either using information gathered by applying other algorithms such as optimal solutions to problem relaxations; or this may be done by directly enhancing the functionality of a metaheuristic with algorithmic components originating from other techniques. In the following two subsections we give examples for both variants.

7.1 Using Information Gathered by Other Algorithms

Guiding metaheuristics using information gathered by applying other algorithms is often a very successful approach that is commonly used. *Problem relaxations*, where some or all constraints of a problem are loosened or omitted, are often used to efficiently obtain bounds and approximate (not necessarily feasible) solutions to the original problem. The gathered information can be utilized for guiding the search, since an optimal solution to a relaxation often indicates in which parts of the original problem's search space good or even optimal solutions might be found.

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 a promising starting point for a subsequent metaheuristic (or exact) search; see also Section 3. For example, Raidl [86] applies this idea in a GA for the MKP. The MKP's LP relaxation is solved and a randomized rounding procedure derives an initial population of diverse solutions from the LP-optimum. Furthermore, the LP-optimum is also exploited for guiding the repair of infeasible candidate solutions and for local improvement. The variables are sorted according to increasing LP values. The greedy repair procedure considers the variables in this order and removes items from the knapsack until all constraints are fulfilled. In the 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 the biasing of variation operators such as recombination and mutation in evolutionary algorithms—exist.

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

Plateau et al. [78] combine interior point methods and metaheuristics for solving the 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 the initial population for a path relinking/scatter search.

Puchinger and Raidl [83] suggest a new variant of VNS: *relaxation guided variable neighborhood search*. It is based on the general VNS scheme and a new embedded variable neighborhood descent (VND) strategy utilizing different types of neighborhood structures. For a current incumbent solution, the order in which the neighborhoods are searched is determined dynamically by first solving relaxations of them. The objective values of these relaxations are used as indicators for the potential gains of searching the corresponding neighborhoods, and more promising neighborhoods are searched first. The proposed approach has been tested on the MKP but is more generally applicable. Computational experiments involving several types of ILP-based neighborhoods show that the adaptive neighborhood ordering is beneficial for the heuristic search, improving obtained results.

Occasionally, dual variable information of LP solutions is also exploited. Chu and Beasley [25] make use of it 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. For the MKP, these pseudo-utility ratios tend to be better indicators for the likeliness of the corresponding items to be included in an optimal integer solution than the primal variable values and several other measures, see [85].

Other relaxations besides the LP relaxation are occasionally also exploited in conjunction with metaheuristics. A successful example is the hybrid Lagrangian GA for the prize collecting Steiner tree problem from Haouari and Siala [56]. It is based on a Lagrangian decomposition of a minimum spanning tree-like ILP formulation of the problem. The volume algorithm, which is a special variant of subgradient search [11], is used for solving the Lagrangian dual. After its termination, the GA is started and exploits results obtained from the volume algorithm in several ways: (a) 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. (b) 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. (c) Instead of the original objective function, an alternate one, based on the reduced costs that are obtained by the volume algorithm, is used. The idea is to focus the search even stronger on promising regions of the search space, where also better solutions with respect to the original objective function can presumably be found.

Pirkwieser et al. [76] describe a similar combination of Lagrangian decomposition and a GA for the knapsack constrained maximum spanning tree problem. 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 even for large instances. These solutions are polished by the GA, and in most cases proven optimal solutions are finally obtained.

Dowsland et al. [34] propose an approach where bounding information available from partial solutions is used to guide an evolutionary algorithm. An indirect, order-based representation of candidate solutions is applied. Phenotypes are derived by a specific decoding procedure which is a construction heuristic that is also able to calculate upper bounds for intermediate partial solutions (considering a maximization problem). Given a certain target value, which is e.g. the objective value of the so far best solution, a *bound point* is determined for each candidate solution in the population: It is the first position in the genotype for which the corresponding partial solution has a bound that is worse than the target value. A modified one-point crossover is then guided by this bound information: The crossover point must be chosen in the part of the first chromosome before its bound point. In this way, recombinations definitely leading to worse offspring are avoided. The authors successfully tested this concept on a relatively simple pallet loading problem and a more complex two-dimensional packing problem with non-identical pieces.

7.2 Enhancing the Functionality of Metaheuristics

One of the basic ingredients of an optimization technique is a mechanism for exploring the search space. An important class of algorithms tackles an optimization problem by exploring the search space along a so-called *search tree*. This class of algorithms comprises approximate as well as complete techniques. An example of a complete method belonging to this class is B&B. An interesting heuristic derivative of breadth-first B&B is *beam search* [72]. While B&B (implicitly) considers all nodes at a certain level in the search tree, beam search restricts the search to a certain number of nodes based on bounding information.

One relatively recent line of research deals with the incorporation of algorithmic components originating from deterministic B&B derivatives such as beam search into construction-based metaheuristics. Examples are the so-called *Beam-ACO* algorithms [16, 18] and approximate and non-deterministic tree search (ANTS) procedures [62, 63]. Note that Beam-ACO can be seen as a generalization of ANTS. In Beam-ACO, artificial ants perform a probabilistic beam search in which the extension of partial solutions is done in the ACO fashion rather than deterministically. The existence of an accurate—and computationally inexpensive—lower bound for the guidance of the ACO's search process is crucial for the success of Beam-ACO.

Another successful example concerns the use of CP techniques for restricting the search performed by an ACO algorithm to promising regions of the search space.

The motivation for this type of hybridization is as follows: Generally, ACO algorithms are competitive with other optimization techniques when applied to problems that are not overly constrained. However, when highly constrained problems such as scheduling or timetabling are considered, the performance of ACO algorithms generally degrades. Note that this is usually also the case for other metaheuristics. The reason is to be found in the structure of the search space: When a problem is not overly constrained, it is usually not difficult to find feasible solutions. The difficulty rather lies in the optimization part, namely the search for good feasible solutions. On the other side, when a problem is highly constrained the difficulty is rather in finding any feasible solution. This is where CP comes into play, because these problems are the target problems for CP applications. Meyer and Ernst [67] introduced the incorporation of CP into ACO in an application to the single machine job scheduling problem.

8 Strategic Guidance of Other Techniques by Metaheuristics

Many 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: assuming minimization, a node with smallest lower bound is always selected, since it is considered to be most promising for leading to 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" in 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 the one that has been most recently 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 approximate solutions. In some sense, we can consider these strategies to "virtually" execute a metaheuristic.

Danna et al. [31] describe *guided dives*, which are a minor, but effective modification of the already mentioned simple diving by temporarily switching to depth-first search. The branch 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 an incumbent solution. Diving is therefore biased towards the neighborhood of this solution. In-

stead of performing only a single dive at the beginning, guided dives are repeatedly applied at regular intervals during the whole optimization process. This strategy is trivial to implement, and experimental results indicate significant advantages over standard node selection strategies.

Fischetti and Lodi [42] propose *local branching*, an exact approach introducing the spirit of classical k-OPT local search in a generic branch-and-cut based MIP solver. The whole problem is partitioned into a k-OPT neighborhood of an initial solution and the remaining part of the search space by applying a *local branching constraint* and its inverse, respectively. The MIP solver is then forced to completely solve the k-OPT neighborhood before considering the remainder of the problem. If an improved solution has been found in the k-OPT neighborhood, a new subproblem corresponding to the k-OPT neighborhood of this new incumbent is split off and solved in the same way; otherwise, a larger k may be tried. The process is repeated until no further improvement can be achieved. Finally, the remaining problem corresponding to all parts of the search space not yet considered is processed in a standard way.

Hansen et al. [55] present a variant of local branching in which they follow the classical VNS strategy, especially for adapting the neighborhood parameter k. Improved results are reported. Another variant of the original local branching scheme is described by Fischetti et al. in [43]. They consider problems in which the set of variables can be naturally partitioned into two levels and fixing the values of the first-level variables yields substantially easier subproblems; cf. Section 4.

Danna et al. [31] further suggest an approach called *relaxation induced neighborhood search* (RINS) for exploring 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's termination. In the authors' experiments, CPLEX is the 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. CPLEX includes RINS as a standard strategy for quickly obtaining good heuristic solutions since version 10.

The *nested partitioning* method proposed by Shi and Ólafsson [92] is another example where a metaheuristic provides strategic guidance to another technique. At each iteration the search focuses on a part of the search space called the most promising region. The remaining part of the search space is called the surrounding region. The most promising region may, for example, be characterized by a number of fixed variables. At each step, the most promising region is divided into a fixed number of subregions. This may be done, for example, by choosing one of the free variables and creating a subregion for each of the variable's possible domain value. Each of the subregions as well as the surrounding region is then sampled. The best

objective function value obtained for each region is called the promising index. The region with the best index becomes the most promising region of the next iteration. The next most promising region is thus nested within the last one. When the surrounding region is found to be the best, the method backtracks to a larger region. The approach may be divided into four main steps: partitioning, sampling, selecting a promising region, and backtracking. Each of these steps may be implemented in a generic fashion, but can also be defined in a problem specific way. In particular the sampling phase may benefit from the use of metaheuristics instead of performing a naive random sampling. In a sense, metaheuristics can be seen as enhancements for guiding the search process of the method. In [5], for example, ant colony optimization is applied for sampling, whereas in [93] local search is used for this purpose.

A very different paradigm is followed in *constraint-based local search* [98]. It combines the flexibility of CP concepts such as rich modeling, global constraints, and search abstractions with the efficiency of local search. The *Comet* programming language allows the modelling of combinatorial optimization problems in a relatively natural way. It also provides the necessary abstractions for specifying metaheuristics and nondeterministic, local and hybrid search. The concept of differential objects is an important aspect of Comet: Constraints maintain their violation and objective functions their evaluation. They can both return information on the possible consequences of local search moves by automatically propagating changes using so-called invariants. The separation of the model and search allows the specification of generic search procedures such as tabu search, variable neighborhood search, and hybrid evolutionary search [98]. The authors describe applications of various hybrid metaheuristics to problems ranging from car sequencing and graph coloring to scheduling. For example, a tabu search algorithm for the job shop scheduling problem is presented, combining local search with complete enumeration as well as limited backtracking search. Several subsequent publications show the strong research interests in this direction and address issues such as distributed search [68] and visualization [33].

9 Decomposition Approaches

Problem decomposition approaches are another category of powerful techniques for combining different optimization techniques. Usually, a very hard-to-solve problem is decomposed into parts which can be dealt with more effectively. Some of the multi-stage approaches which we discussed in Section 4 already follow this basic idea. Large neighborhood search, heuristic cut and column generation in mixed integer programming, and constraint propagation by means of metaheuristics are three other prominent instances of successful decomposition techniques, which we consider in the following in more detail.

9.1 Exploring Large Neighborhoods

A common approach in more sophisticated local search based metaheuristics is to search neighborhoods not by naive enumeration but by clever, more efficient algorithms. If the neighborhoods are chosen appropriately, they can be quite large and nevertheless an efficient search for a best neighbor is still possible in short time. Such techniques are known as *very large-scale neighborhood* (VLSN) search [3]; see also [24] for a recent survey. Many of today's combinations of local search based metaheuristics with dynamic programming or MIP techniques follow this scheme. In the following, we present some examples.

In *Dynasearch* [27, 28] 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 such simple steps that are executed in parallel in a single local search iteration. The required independence in the context of Dynasearch means that the individual simple moves do not interfere with each other; in this case, dynamic programming can be used to find a best combination. Ergun and Orlin [39] investigated several such neighborhoods in particular for the traveling salesman problem.

Particular types of large neighborhoods that can also be efficiently searched by dynamic programming are *cyclic and path exchange neighborhoods* [3, 4]. They are often applied in the context of problems where items need to be partitioned into disjoint sets. Examples of such problems are vehicle routing, capacitated minimum spanning tree, and parallel machine scheduling. In these neighborhoods, a series of items is exchanged between an arbitrary number of sets in a cyclic or path-like fashion, and a best move is determined by a shortest path-like algorithm.

Pesant and Gendreau [75] describe a generic framework for combining CP and local search. They view and model the original problem as well as the (large) neighborhoods as CP problems. Each of the neighborhoods is solved via a CP-based B&B that preserves solution feasibility. The framework allows for a relatively generic problem modeling while providing the advantages of local search. The authors solve a physician scheduling problem as well as the travelling salesman problem with time windows, and they approach them by tabu search in which large neighborhoods are searched by means of the CP-based B&B.

Puchinger et al. [84] describe a hybrid GA for a real-world glass cutting problem in which large neighborhoods are searched by means of B&B. The GA uses an order-based representation which is decoded using a greedy heuristic. B&B is applied with a certain probability, enhancing the decoding phase by generating locally optimal subpatterns. Reported results indicate that occasionally solving subpatterns to optimality often increases the overall solution quality.

Quite often, large neighborhoods are described in the form of MIPs and a MIP-solver is applied for finding a good—or the best—neighbor. For example, Büdenbender et al. [23] 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 achieved by closing flights frequently occurring in previously devised solutions. Other examples for MIP-based large neighborhood search can be found in Duarte et al. [35], where an iterated local search framework is applied to a real-world referee assignment problem, and in Prandtstetter and Raidl [79] where several different MIP-based neighborhoods are searched within a VNS framework for a car sequencing problem.

Hu et al. [57] propose a VNS for the generalized minimum spanning tree problem. The approach uses two dual types of representations and exponentially large neighborhood structures. 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 structure contributes to the overall excellent performance.

Variable neighborhood decomposition search (VNDS) [54] is a variant of VNS obtained by selecting the neighborhoods so as to obtain a problem decomposition. VNDS follows the usual VNS scheme, but the neighborhood structures and the local search are defined on subproblems rather than on the original problem. Given a solution, all but k attributes (usually variables) are kept fixed. For each k, a neighborhood structure $\mathcal{N}_k$ is defined. Local search only regards changes on the variables belonging to the subproblem it is applied to. Successful applications of VNDS include the edge-weighted k-cardinality tree problem [97] and supply chain management planning problems [61].

9.2 Cut and Column Generation by Metaheuristics

Cutting plane algorithms [105] are a powerful tool for solving complex MIPs. They start with a relaxation of the original problem in which most of the constraints—especially the integrality constraints—are omitted. This relaxation is solved, and then a *separation algorithm* is applied for finding further constraints that are fulfilled by an optimal solution to the original problem but are violated by the current solution to the relaxed problem. If such constraints, called *cuts*, could be identified, they are added to the relaxed LP, which is then solved again. This process is iterated until no further cuts can be found. If the final solution is infeasible, either a heuristic repair procedure may be applied, or the cutting plane algorithm is embedded in a B&B framework yielding an exact *branch-and-cut* algorithm.

Often, the subproblem of separating a cut (i.e. finding a valid inequality violated by the current LP solution) is difficult to solve by itself. In such cases, heuristics are often applied, and also fast metaheuristics have already been successfully used.

One example is the work from Augerat et al. [10], which uses a hierarchy consisting of a simple constructive heuristic, a randomized greedy method, and a tabu search for separating capacity constraints within a branch-and-cut algorithm for a capacitated vehicle routing problem. Another more recent example is the branch-

and-cut algorithm for the diameter bounded minimum spanning tree problem by Gruber and Raidl [52], in which local search and tabu search techniques are used for separating so-called jump cuts.

One more example concerns the acceleration of Benders decomposition by local branching, as described by Rei et al. [89]. The basic principle of Benders decomposition is to project a 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 generated. According to the classical method, an optimal solution to the relaxed master problem (including only the already separated cuts) is needed and an LP involving this solution must be solved in order to separate a single new cut. Rei et al. [89] 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 solved again.

Column generation algorithms can be seen as dual to cutting plane algorithms. Instead of starting with a reduced set of constraints and dynamically extending it, the set of variables (which correspond to columns in the matrix notation of the MIP) is restricted, and further variables are iteratively added. Hereby, the essential subproblem, called *pricing problem*, is to identify variables whose inclusion will yield an improvement. Again, the pricing problem is often difficult by itself, and applying fast (meta-)heuristics is sometimes a meaningful option. If column generation is performed within an exact LP-based B&B framework, the approach is called *branch-and-price*.

Filho and Lorena [41] apply a heuristic column generation approach to graph coloring. A GA is used to generate initial columns and to solve the pricing 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. Puchinger and Raidl [80, 82] describe an exact branch-and-price approach for the three-stage two-dimensional bin packing problem. Fast column generation is performed by applying a hierarchy of four methods: (a) a greedy heuristic, (b) an evolutionary algorithm, (c) solving a restricted form of the pricing problem using CPLEX, and finally (d) solving the complete pricing problem using CPLEX.

9.3 Using Metaheuristics for Constraint Propagation

In CP the mechanism of constraint propagation is used to reduce the domains of the variables at each node of the B&B search tree. Similarly to cut generation in mixed integer programming, the search space is reduced by deducing consequences from the current state of the search. Usually specialized and standard combinatorial algorithms are used [65].

Galinier et al. [46] present a tabu search procedure to speed up filtering for generalized all-different constraints. That is:

$$\mathit{SomeDifferent}(X,D,G) = \{(a_1,\ldots,a_n) \mid a_i \in D_i \wedge a_i \neq a_j \forall (i,j) \in E(G)\}$$

is defined over a set of variables $X = \{x_1,\ldots,x_n\}$ with domains $D = \{D_1,\ldots,D_n\}$ and an underlying graph $G = (X,E)$ [90]. The satisfiability of the constraint can be tested by solving a special graph coloring problem. Tabu search is first applied to see if it can color the graph. If it does not find a solution, an exact method is applied. In a second step a similar tabu search procedure is used to determine a large set of variable/value combinations that are feasible. Finally an exact filtering is applied to the remaining variable/value pairs checking if some of them can be excluded from the variable domains. Computational experiments show that the hybrid approach is comparable to the state-of-the-art on data from a real-world work-force management problem and is significantly faster on random graph instances for the SomeDifferent constraint. The authors suppose that the idea of combining fast metaheuristics with exact procedures can speed up filtering procedures for other NP-hard constraints as well.

10 Summary and Conclusions

We have reviewed a large number of different possibilities for combining traditional metaheuristic strategies with each other or with algorithmic techniques coming from other fields. All these possibilities have their individual pros and cons, but the common underlying motivation is to exploit the advantages of the individual techniques in order to obtain a more effective hybrid system, benefiting from synergy. In fact, history clearly shows that the concentration on a single metaheuristic is rather restrictive for advancing the state-of-the-art when tackling difficult optimization problems. Thus, designing hybrid systems for complex optimization problems is nowadays a natural process.

On the downside, metaheuristic hybrids are usually significantly more complex than classical "pure" strategies. The necessary development and tuning effort may be substantially higher than when using a straightforward out-of-the-box strategy. One should further keep in mind that a more complex hybrid algorithm does not automatically perform better—an adequate design and appropriate tuning is always mandatory, and the effort increases with the system's complexity. Einstein's advice of *"keeping things as simple as possible, but not simpler"* therefore is especially true also for metaheuristic hybrids.

We started by presenting a classification of metaheuristic hybrids in which we pointed out the different basic characteristics. Then we discussed several commonly used design templates. Note that these templates are not meant as a clear categorization of existing hybrid approaches: Many of the referenced examples from the

literature can be argued to follow more than one design template, and occasionally the boundaries are fuzzy.

Finding initial or improved solutions by embedded methods might be the most commonly applied approach. Multi-stage combinations are sometimes straightforward for problems that naturally decompose into multiple levels and are also otherwise popular as they are typically easier to tune than more intertwined hybrids. The concept of decoder-based metaheuristics is also quite popular, as they can often be implemented quickly, once an appropriate construction heuristic is available. Solution merging was the next design template we discussed and for which numerous successful examples exist. Then we considered cases where metaheuristics are strategically guided by other techniques. In particular, solutions to relaxations of the original problem are frequently exploited in various ways. The reverse, strategic guidance of other techniques by metaheuristics, has been particularly successful in the field of mixed integer programming, where such strategies can help to find good approximate solutions early within an exact B&B-based method. Last but not least, there are several different decomposition approaches: Exploring large neighborhoods by specialized algorithms has become particularly popular over the last years, and occasionally metaheuristics are applied to solve separation or pricing problems in more complex MIP approaches and propagation subproblems in CP.

As an important final advice for the development of well-performing metaheuristic hybrids, the authors would like to recommend (1) the careful search of the literature for the most successful optimization approaches for the problem at hand or for similar problems, and (2) the study of clever ways of combining the most interesting features of the identified approaches. We hope this chapter provides a starting point and some useful references for this purpose.

Acknowledgements Günther R. Raidl is supported by the Austrian Science Fund (FWF) under grant 811378 and by the Austrian Exchange Service (Acciones Integradas, grant 13/2006).
NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.
Christian Blum is supported by grants TIN2005-08818 (OPLINK) and TIN2007-66523 (FORMALISM) of the Spanish government, and by the EU project FRONTS (FP7-ICT-2007-1). He also acknowledges support from the *Ramón y Cajal* program of the Spanish Ministry of Science and Technology.

References

1. Aggarwal, C., Orlin, J., Tai, R.: Optimized crossover for the independent set problem. Operations Research **45**, 226–234 (1997)
2. Ahuja, R., Orlin, J., Tiwari, A.: A greedy genetic algorithm for the quadratic assignment problem. Computers & Operations Research **27**, 917–934 (2000)
3. Ahuja, R.K., Ergun, Ö., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics **123**(1-3), 75–102 (2002)

4. Ahuja, R.K., Orlin, J., Sharma, D.: Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. Mathematical Programming **91**(1), 71–97 (2001)
5. Al-Shihabi, S.: Ants for sampling in the nested partition algorithm. In: Blum et al. [22], pp. 11–18
6. Alba, E. (ed.): Parallel Metaheuristics: A New Class of Algorithms. John Wiley (2005)
7. Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez, M.P., Roli, A., Sampels, M. (eds.): Proceedings of HM 2006 – Third International Workshop on Hybrid Metaheuristics, *Lecture Notes in Computer Science*, vol. 4030. Springer (2006)
8. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: On the solution of the traveling salesman problem. Documenta Mathematica **Extra Volume ICM III**, 645–656 (1998)
9. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton Series in Applied Mathematics. Princeton University Press (2007)
10. Augerat, P., Belenguer, J., Benavent, E., Corberan, A., Naddef, D.: Separating capacity constraints in the CVRP using tabu search. European Journal of Operational Research **106**(2), 546–557 (1999)
11. Barahona, F., Anbil, R.: The volume algorithm: Producing primal solutions with a subgradient method. Mathematical Programming, Series A **87**(3), 385–399 (2000)
12. Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.): Proceedings of HM 2007 – Fourth International Workshop on Hybrid Metaheuristics, *Lecture Notes in Computer Science*, vol. 4771. Springer (2007)
13. Beck, J.C.: Solution-guided multi-point constructive search for job shop scheduling. Journal of Artificial Intelligence Research **29**, 49–77 (2007)
14. Binato, S., Hery, W.J., Loewenstern, D., Resende, M.G.C.: A GRASP for job shop scheduling. In: C.C. Ribeiro, P. Hansen (eds.) Essays and Surveys on Metaheuristics, pp. 59–79. Kluwer Academic Publishers (2001)
15. Blesa Aguilera, M.J., Blum, C., Roli, A., Sampels, M. (eds.): Proceedings of HM 2005 – Second International Workshop on Hybrid Metaheuristics, *Lecture Notes in Computer Science*, vol. 3636. Springer (2005)
16. Blum, C.: Beam-ACO: Hybridizing ant colony optimization with beam search: An application to open shop scheduling. Computers and Operations Research **32**(6), 1565–1591 (2005)
17. Blum, C.: A new hybrid evolutionary algorithm for the *k*-cardinality tree problem. In: Proceedings of the Genetic and Evolutionary Computation Conference 2006, pp. 515–522. ACM Press (2006)
18. Blum, C.: Beam-ACO for simple assembly line balancing. INFORMS Journal on Computing **20**(4), 618–627 (2008)
19. Blum, C., Blesa, M.: Combining ant colony optimization with dynamic programming for solving the *k*-cardinality tree problem. In: Proceedings of IWANN 2005 – 8th International Work-Conference on Artificial Neural Networks, Computational Intelligence and Bioinspired Systems, no. 3512 in Lecture Notes in Computer Science, pp. 25–33. Springer (2005)
20. Blum, C., Blesa Aguilera, M.J., Roli, A., Sampels, M. (eds.): Hybrid Metaheuristics – An Emerging Approach to Optimization, *Studies in Computational Intelligence*, vol. 114. Springer (2008)
21. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys **35**(3), 268–308 (2003)
22. Blum, C., Roli, A., Sampels, M. (eds.): Proceedings of HM 2004 – First International Workshop on Hybrid Metaheuristics. Valencia, Spain (2004)
23. Büdenbender, K., Grünert, T., Sebastian, H.J.: A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. Transportation Science **34**(4), 364–380 (2000)
24. Chiarandini, M., Dumitrescu, I., Stützle, T.: Very large-scale neighborhood search: Overview and case studies on coloring problems. In: Blum et al. [20], pp. 117–150
25. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics **4**, 63–86 (1998)

26. Cohoon, J., Hegde, S., Martin, W., Richards, D.: Punctuated equilibria: A parallel genetic algorithm. In: J. Grefenstette (ed.) Proceedings of the Second International Conference on Genetic Algorithms, pp. 148–154. Lawrence Erlbaum Associates (1987)
27. Congram, R.K.: Polynomially searchable exponential neighbourhoods for sequencing problems in combinatorial optimisation. Ph.D. thesis, University of Southampton, Faculty of Mathematical Studies, UK (2000)
28. Congram, R.K., Potts, C.N., van de Velde, S.L.: An iterated Dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. INFORMS Journal on Computing **14**(1), 52–67 (2002)
29. Cotta, C.: A study of hybridisation techniques and their application to the design of evolutionary algorithms. AI Communications **11**(3–4), 223–224 (1998)
30. Cotta, C., Troya, J.M.: Embedding branch and bound within evolutionary algorithms. Applied Intelligence **18**, 137–153 (2003)
31. Danna, E., Rothberg, E., Le Pape, C.: Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, Series A **102**, 71–90 (2005)
32. Denzinger, J., Offermann, T.: On cooperation between evolutionary algorithms and other search paradigms. In: W. Porto, et al. (eds.) Proceedings of the 1999 Congress on Evolutionary Computation (CEC), vol. 3, pp. 2317–2324. IEEE Press (1999)
33. Dooms, G., Van Hentenryck, P., Michel, L.: Model-driven visualizations of constraint-based local search. In: C. Bessiere (ed.) Principles and Practice of Constraint Programming – CP 2007, 13th International Conference, *Lecture Notes in Computer Science*, vol. 4741, pp. 271–285. Springer (2007)
34. Dowsland, K.A., Herbert, E.A., Kendall, G., Burke, E.: Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. European Journal of Operational Research **168**(2), 390–402 (2006)
35. Duarte, A.R., Ribeiro, C.C., Urrutia, S.: A hybrid ILS heuristic to the referee assignment problem with an embedded MIP strategy. In: Bartz-Beielstein et al. [12], pp. 82–95
36. Dumitrescu, I., Stuetzle, T.: Combinations of local search and exact algorithms. In: G.R. Raidl, et al. (eds.) Applications of Evolutionary Computation, *Lecture Notes in Computer Science*, vol. 2611, pp. 211–223. Springer (2003)
37. El-Abd, M., Kamel, M.: A taxonomy of cooperative search algorithms. In: Blesa Aguilera et al. [15], pp. 32–41
38. Eremeev, A.V.: On complexity of optimal recombination for binary representations of solutions. Evolutionary Computation **16**(1), 127–147 (2008)
39. Ergun, O., Orlin, J.B.: A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. Discrete Optimization **3**(1), 78–85 (2006)
40. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**, 109–133 (1995)
41. Filho, G.R., Lorena, L.A.N.: Constructive genetic algorithm and column generation: An application to graph coloring. In: L.P. Chuen (ed.) Proceedings of APORS 2000, the Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS. Singapore (2000)
42. Fischetti, M., Lodi, A.: Local Branching. Mathematical Programming, Series B **98**, 23–47 (2003)
43. Fischetti, M., Polo, C., Scantamburlo, M.: 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)
44. Fleurent, C., Glover, F.: Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS Journal on Computing **11**, 198–204 (1999)
45. Focacci, F., Laburthe, F., Lodi, A.: Local search and constraint programming: LS and CP illustrated on a transportation problem. In: M. Milano (ed.) Constraint and Integer Programming. Towards a Unified Methodology, pp. 293–329. Kluwer Academic Publishers (2004)
46. Galinier, P., Hertz, A., Paroz, S., Pesant, G.: Using local search to speed up filtering algorithms for some NP-hard constraints. In: Perron and Trick [74], pp. 298–302

47. Gilmour, S., Dras, M.: Kernelization as heuristic structure for the vertex cover problem. In: M. Dorigo, et al. (eds.) Proceedings of ANTS 2006 – 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, *Lecture Notes in Computer Science*, vol. 4150, pp. 452–459. Springer (2006)
48. Glover, F.: Surrogate constraints. Operations Research **16**(4), 741–749 (1968)
49. Glover, F.: Parametric tabu-search for mixed integer programming. Computers and Operations Research **33**(9), 2449–2494 (2006)
50. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **39**(3), 653–684 (2000)
51. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Learning. Addison-Wesley (1989)
52. Gruber, M., Raidl, G.R.: (Meta-)heuristic separation of jump cuts for the bounded diameter minimum spanning tree problem. In: Hansen et al. [53]
53. Hansen, P., Maniezzo, V., Fischetti, M., Stuetzle, T. (eds.): Proceedings of Matheuristics 2008: Second International Workshop on Model Based Metaheuristics. Bertinoro, Italy (2008)
54. Hansen, P., Mladenovic, N., Perez-Britos, D.: Variable neighborhood decomposition search. Journal of Heuristics **7**(4), 335–350 (2001)
55. Hansen, P., Mladenović, N., Urosević, D.: Variable neighborhood search and local branching. Computers and Operations Research **33**(10), 3034–3045 (2006)
56. Haouari, M., Siala, J.C.: A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. Computers & Operations Research **33**(5), 1274–1288 (2006)
57. Hu, B., Leitner, M., Raidl, G.R.: Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. Journal of Heuristics **14**(5), 473–479 (2008)
58. Hu, B., Raidl, G.R.: Effective neighborhood structures for the generalized traveling salesman problem. In: J. van Hemert, C. Cotta (eds.) Evolutionary Computation in Combinatorial Optimisation – EvoCOP 2008, *Lecture Notes in Computer Science*, vol. 4972, pp. 36–47. Springer (2008)
59. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)
60. Klau, G.W., Lesh, N., Marks, J., Mitzenmacher, M.: Human-guided search: Survey and recent results. submitted to Journal of Heuristics (2007)
61. Lejeune, M.A.: A variable neighborhood decomposition search method for supply chain management planning problems. European Journal of Operational Research **175**(2), 959–976 (2006)
62. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS Journal on Computing **11**(4), 358–369 (1999)
63. Maniezzo, V., Carbonaro, A.: An ANTS heuristic for the frequency assignment problem. Future Generation Computer Systems **16**, 927–935 (2000)
64. Maniezzo, V., Hansen, P., Voss, S. (eds.): Proceedings of Matheuristics 2006: First International Workshop on Mathematical Contributions to Metaheuristics. Bertinoro, Italy (2006)
65. Marriott, K., Stuckey, P.J.: Introduction to Constraint Logic Programming. MIT Press (1998)
66. Martin, O., Otto, S.W., Felten, E.W.: Large-step Markov chains for the traveling salesman problem. Complex Systems **5**, 299–326 (1991)
67. Meyer, B., Ernst, A.: Integrating ACO and constraint propagation. In: M. Dorigo, et al. (eds.) Proceedings of ANTS 2004 – Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence, *Lecture Notes in Computer Science*, vol. 3172, pp. 166–177. Springer (2004)
68. Michel, L., See, A., Van Hentenryck, P.: Distributed constraint-based local search. In: F. Benhamou (ed.) Principles and Practice of Constraint Programming – CP 2006, 12th International Conference, *Lecture Notes in Computer Science*, vol. 4204, pp. 344–358. Springer (2006)
69. Moscato, P.: Memetic algorithms: A short introduction. In: D. Corne, et al. (eds.) New Ideas in Optimization, pp. 219–234. McGraw Hill (1999)

70. Nagar, A., Heragu, S.S., Haddock, J.: A meta-heuristic algorithm for a bi-criteria scheduling problem. Annals of Operations Research **63**, 397–414 (1995)
71. Neto, T., Pedroso, J.P.: GRASP for linear integer programming. In: J.P. Sousa, M.G.C. Resende (eds.) Metaheuristics: Computer Decision Making, Combinatorial Optimization Book Series, pp. 545–574. Kluwer Academic Publishers (2003)
72. Ow, P.S., Morton, T.E.: Filtered beam search in scheduling. International Journal of Production Research **26**, 297–307 (1988)
73. Pedroso, J.P.: Tabu search for mixed integer programming. In: C. Rego, B. Alidaee (eds.) Metaheuristic Optimization via Memory and Evolution, *Operations Research/Computer Science Interfaces Series*, vol. 30, pp. 247–261. Springer (2005)
74. Perron, L., Trick, M.A. (eds.): Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems – CPAIOR 2008, 5th International Conference, *Lecture Notes in Computer Science*, vol. 5015. Springer (2008)
75. Pesant, G., Gendreau, M.: A constraint programming framework for local search methods. Journal of Heuristics **5**(3), 255–279 (1999)
76. Pirkwieser, S., Raidl, G.R., Puchinger, J.: Combining Lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. In: C. Cotta, J. van Hemert (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2007, *Lecture Notes in Computer Science*, vol. 4446, pp. 176–187. Springer (2007)
77. Pisinger, D.: Core problems in knapsack algorithms. Operations Research **47**, 570–575 (1999)
78. Plateau, A., Tachat, D., Tolla, P.: A hybrid search combining interior point methods and metaheuristics for 0–1 programming. International Transactions in Operational Research **9**, 731–746 (2002)
79. Prandtstetter, M., Raidl, G.R.: An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. European Journal of Operational Research **191**(3) (2008)
80. Puchinger, J., Raidl, G.R.: An evolutionary algorithm for column generation in integer programming: An effective approach for 2D bin packing. In: X. Yao, et al. (eds.) Parallel Problem Solving from Nature – PPSN VIII, *Lecture Notes in Computer Science*, vol. 3242, pp. 642–651. Springer (2004)
81. Puchinger, J., Raidl, G.R.: 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, *Lecture Notes in Computer Science*, vol. 3562, pp. 41–53. Springer (2005)
82. Puchinger, J., Raidl, G.R.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research **183**, 1304–1327 (2007)
83. Puchinger, J., Raidl, G.R.: Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. Journal of Heuristics **14**(5), 457–472 (2008)
84. Puchinger, J., Raidl, G.R., Koller, G.: Solving a real-world glass cutting problem. In: J. Gottlieb, G.R. Raidl (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2004, *Lecture Notes in Computer Science*, vol. 3004, pp. 162–173. Springer (2004)
85. Puchinger, J., Raidl, G.R., Pferschy, U.: The core concept for the multidimensional knapsack problem. In: J. Gottlieb, G.R. Raidl (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2006, *Lecture Notes in Computer Science*, vol. 3906, pp. 195–208. Springer (2006)
86. Raidl, G.R.: An improved genetic algorithm for the multiconstrained 0–1 knapsack problem. In: D.B. Fogel, et al. (eds.) Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, pp. 207–211. IEEE Press (1998)
87. Raidl, G.R.: A unified view on hybrid metaheuristics. In: Almeida et al. [7], pp. 1–12
88. Raidl, G.R., Puchinger, J.: Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In: Blum et al. [20], pp. 31–62
89. Rei, W., Cordeau, J.F., Gendreau, M., Soriano, P.: Accelerating Benders decomposition by local branching. INFORMS Journal on Computing (2008). In press

90. Richter, Y., Freund, A., Naveh, Y.: Generalizing AllDifferent: The SomeDifferent constraint. In: F. Benhamou (ed.) Principles and Practice of Constraint Programming, 12th International Conference, CP 2006, *Lecture Notes in Computer Science*, vol. 4204, pp. 468–483. Springer (2006)
91. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS Journal on Computing **19**(4), 534–541 (2007)
92. Shi, L., Ólafsson, S.: Nested partitions method for global optimization. Operations Research **48**(3), 390–407 (2000)
93. Shi, L., Ólafsson, S., Chen, Q.: An optimization framework for product design. Management Science **47**(12), 1681–1692 (2001)
94. Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics **8**(5), 541–565 (2002)
95. Talukdar, S., Baeretzen, L., Gove, A., de Souza, P.: Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics **4**, 295–321 (1998)
96. Tamura, H., Hirahara, A., Hatono, I., Umano, M.: An approximate solution method for combinatorial optimisation. Transactions of the Society of Instrument and Control Engineers **130**, 329–336 (1994)
97. Urosevic, D., Brimberg, J., Mladenovic, N.: Variable neighborhood decomposition search for the edge weighted *k*-cardinality tree problem. Computers & Operations Research **31**(8), 1205–1213 (2004)
98. Van Hentenryck, P., Michel, L.: Constraint-Based Local Search. The MIT Press (2005)
99. Vasquez, M., Hao, J.K.: A hybrid approach for the 0–1 multidimensional knapsack problem. In: B. Nebel (ed.) Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI 2001, pp. 328–333. Morgan Kaufman, Seattle, Washington (2001)
100. Vasquez, M., Vimont, Y.: Improved results on the 0–1 multidimensional knapsack problem. European Journal of Operational Research **165**(1), 70–81 (2005)
101. Walshaw, C.: Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In: Blum et al. [20], pp. 261–289
102. Watson, J.P., Beck, J.C.: A hybrid constraint programming / local search approach to the job-shop scheduling problem. In: Perron and Trick [74], pp. 263–277
103. Watson, J.P., Howe, A.E., Whitley, L.D.: Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. Computers & Operations Research **33**(9), 2623–2644 (2006)
104. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
105. Wolsey, L.A.: Integer Programming. Wiley-Interscience (1998)