Metaheuristic Hybrids

Günther R. Raidl, Jakob Puchinger, and Christian Blum

Abstract Over the last decades, 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 tree-search, dynamic programming and methods from the mathematical programming, constraint programming, and SAT-solving 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, TU Wien, Vienna, Austria,
e-mail: raidl@ac.tuwien.ac.at

Jakob Puchinger
Laboratoire Genie Industriel, CentraleSupélec, Université Paris-Saclay, France
Institut de Recherche Technologique SystemX, Palaiseau, France
e-mail: jakob.puchinger@centralesupelec.fr

Christian Blum
Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, Bellaterra, Spain
e-mail: christian.blum@iiia.csic.es

1 Introduction

Most of the other chapters of this book illustrate the existence of a large number of different metaheuristics. Simulated annealing, tabu search, iterated local search, variable neighborhood search, the greedy randomized adaptive search procedure, evolutionary algorithms such as genetic and memetic algorithms, ant colony optimization, scatter search, and path relinking 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 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*. For hybrids involving mathematical programming models or techniques, the name *matheuristics* also is frequently used. 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 challenging optimization problems of combinatorial nature. The vastly increasing number of reported applications of metaheuristic hybrids and dedicated scientific events such as the *Workshops on Hybrid Metaheuristics* (see the proceedings of the 2016 edition [15]), the *Workshops on Matheuristics* (see the proceedings of the 2016 edition [78]), and the conferences on the *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (see the proceedings of the 2016 edition [103]) 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 relatively isolated and followed quite strictly the biologically inspired thinking. The situation changed, according to many researchers, with the no free lunch theorems [126] 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 made of an adequate combination of sometimes very problem specific parts often originating from different metaheuristics and other al-

gorithmic 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 in-depth reading and comprehensive reviews on hybrid metaheuristics, we recommend the books by Blum and Raidl [25], Talbi [118], and Blum et al. [22]. Hybrid metaheuristics for multiobjective optimization are specifically treated in [43] and for continuous—i.e., real-parameter—optimization in [82].

2 Classification

Several classifications and taxonomies of hybrid metaheuristics can be found in the literature. Here we primarily follow the classification from Raidl [105] that combines aspects of the taxonomy introduced by Talbi [117] with the points-of-view from Cotta [34] and Blum et al. [26]. Differentiations with regard to parallel metaheuristics and hybridization of metaheuristics with exact optimization techniques are adopted from El-Abd and Kamel [44] and from Puchinger and Raidl [99], 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 may combine (components of) different metaheuristics (MH). Second, highly problem specific algorithms, such as entire simulations for evaluating 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 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, constraint programming (CP), and SAT-solving. For a survey dedicated to combinations of metaheuristics with MIP techniques see [109], for an overview on combinations of local search based methods with CP see [39, 53], and for a review on combinations of local search methods with exact techniques see [42]. Examples of other heuristic and soft-computing techniques include neural networks, fuzzy logic, and diverse statistical techniques. As a fourth class we

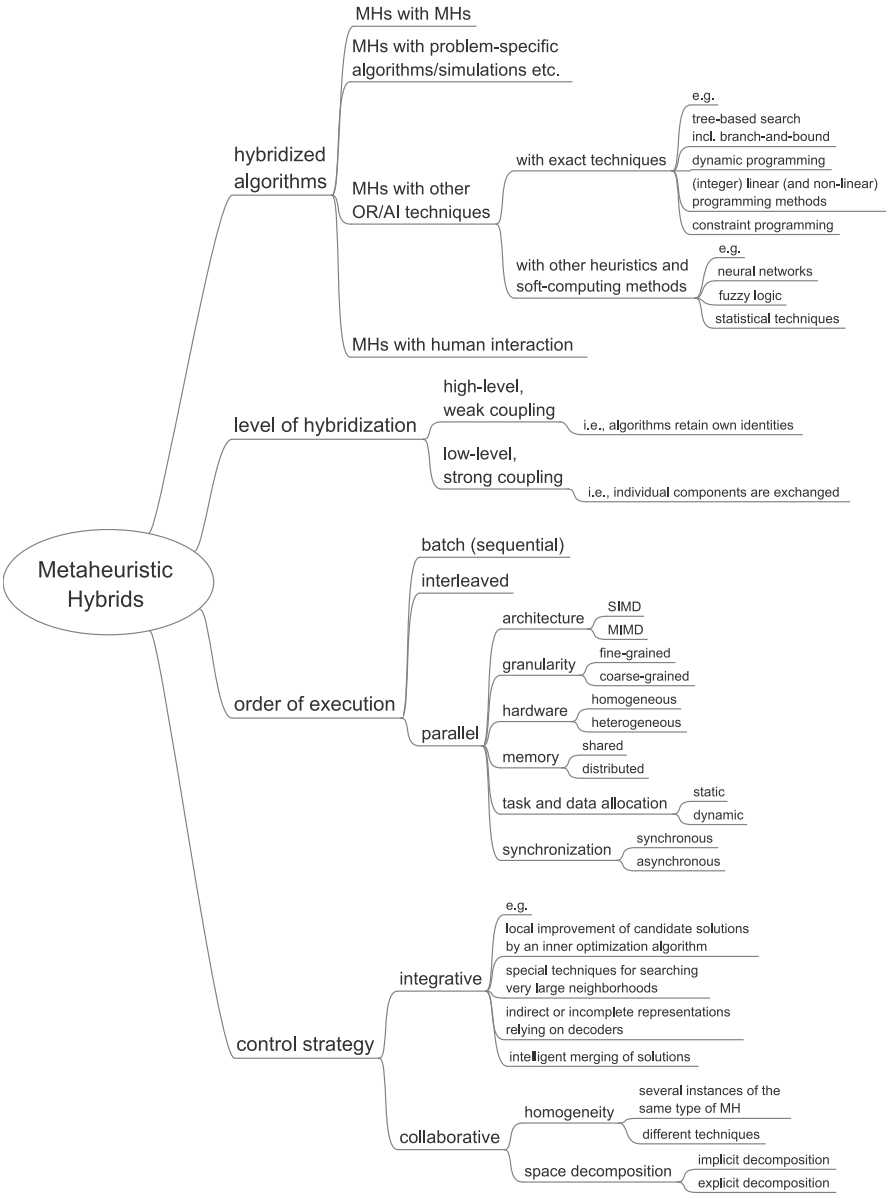

Fig. 1 Classification of metaheuristic (MH) hybrids based on Raidl [105].

want to mention the combination of metaheuristics with human interaction components, 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 well visualized, and where human intuition and intelligence present an advantage, such interactive systems are often highly appreciated in practice [74].

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, substantial 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 mixed.

Order of execution. In the simplest case, the *batch* execution, the individual algorithms are performed in a sequential way and the results of one algorithm are used as input for the next one. 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, 44]. 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 [32] for parallelizing evolutionary algorithms (EAs) 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) [120]: 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 [38] presented a similar multi-agent approach for achieving cooperation between search algorithms following different search paradigms, such as B&B and EAs. Especially in collaborative combinations, a particular 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 for 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, such as the LP relaxation, and repair the obtained solution in some heuristic way, e.g., by rounding. Examples to such approaches can also be found in Section 7.

The *greedy randomized adaptive search procedure* (GRASP) [47] 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 [56]. 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 transferred this principle in [51] 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.

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, *memetic algorithms* [83] typically rely on this principle: The outer metaheuristic is an EA, and intermediate candidate solutions are locally improved. If each intermediate solution is always turned into a local optimum, the EA 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 EAs, because intensification is typically considered a weakness of traditional EAs. 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) [63], where each candidate solution undergoes some kind of local improvement and a sequence 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, which makes use of its own, typically systematically searched sequence of different neighborhood structures.

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 [37, 59, 113] 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 or the whole solution 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. This corresponds to a kind-of top-down solution construction. Different techniques can be applied at the individual levels yielding simple but often very effective hybrid approaches.

For example, for vehicle routing applications where the aim is to deliver goods to customers, it is a meaningful approach to first partition the customers into groups which are then independently treated by finding appropriate delivery tours. Such

approaches are called *cluster-first route-second methods* [50]. In contrast, it is also perfectly reasonable to start by finding a *giant tour* (ordering) over all customers and partition this tour into feasible subtours in the second stage; these methods are called *route-first cluster second* or *order-first split-second* approaches [98]. While such two-staged methods are often relatively fast, they typically only yield solutions of moderate quality due to the neglected dependency of the decision making in the two phases. Such methods, however, can also be further embedded in more rigorous (hybrid) metaheuristic search frameworks and thus combined with other design patterns. In case of vehicle routing, order-first split-second approaches have been very successful recently when applied within a hybrid genetic algorithm following a decoder-based strategy, which we will explain in Section 5.

Another example are job scheduling problems, for which it is often natural to first assign the jobs to machines and then independently optimize the schedules for each machine. For large communication network design problems it may 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 useful for relatively quickly 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 [55], Gilmour and Dras propose 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 [125] can also be considered a special class of multi-stage approaches. They involve a recursive coarsening to create a series 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. *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 whole 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 [94, 102].

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.

In [122], 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 [123] by additional variable fixing.

A general multistage approach is implemented in the context of the so-called generate-and-solve (GS) framework [84], which decomposes the original optimization problem into two conceptually different levels. One of the two levels makes use of a component called *solver of reduced instances* (SRI), in which an exact method is applied to sub-instances of the original problem instance that maintain the conceptual structure of the original instance, that is, any solution to the sub-instance is also a solution to the original instance. At the other level, a metaheuristic component deals with the problem of generating sub-instances that contain high quality solutions. In GS, the metaheuristic component is called *generator of reduced instances* (GRI). Feedback is provided from the SRI component to the GRI component, for example, by means of the objective function value of the best solution found in a sub-instance. This feedback serves for guiding the search process of the GRI component. Application examples in which the GRI component makes use of EAs and simulated annealing can be found in [36, 91, 92].

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 EAs, where encoded solutions are denoted as *genotypes* and the decoded counterparts are called *phenotypes* [58].

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 considering 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 [72]. 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 may 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 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 [85] and in conjunction with tabu search by Pedroso [88].

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 [68] 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 orthogonal 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 is uses as a decoder to obtain a corresponding high-quality tour. The VNS uses several types of neighborhood structures for each representations.

More recently, Biesinger et al. [14] have proposed a related VNS for the generalized vehicle routing problem with stochastic demands. Again, a clustered graph

is given and exactly one node from each cluster needs to be visited. Now, however, stochastic customer demands are additionally given and due to the vehicle's limited capacity, restocking trips back to the depot must also be considered. The VNS employs a permutation of the clusters as incomplete solution representation. For selecting optimal nodes from the clusters to be visited and the expected tour lengths, an exact but time-consuming dynamic programming approach is described. As exact objective values are not always needed in the VNS to possibly recognize and discard inferior solutions, an efficient multi-level evaluation scheme is used.

An impressive example of a quite general decoder-based approach is the *unified hybrid genetic search* (UHGS) framework for solving a large variety of vehicle routing type problems [124]. It is based on the order-first split-second principle as presented in Section 4. The authors propose a component-based method, where problem specifics are treated in the route evaluation component. Route evaluation is the major building block of a problem independent split procedure, allowing to construct a generic hybrid GA. The UHGS framework is applied to 29 vehicle routing variants matching or outperforming most of the state-of-the-art problem specific algorithms. This example shows the strength of decoder-based approaches from an algorithm engineering point of view. Problem specifics are efficiently treated in clearly defined subparts of the metaheuristic allowing to solve a large variety of problem classes without giving away solution quality and algorithmic efficiency.

Besides permutations, *random keys* are another versatile indirect solution representation technique making fundamental use of decoders. Originally, they were proposed by Bean in the context of genetic algorithms [10], and more recently Resende et al. [60] applied refined variants to a larger number of problems. A solution is represented by a vector of real values associated again with the solution elements. For initial solutions, these values are typically independently set to random values of a certain interval, e.g. $[0,1)$—therefore the name "random keys". The decoder sorts all solution elements according to their random keys and then performs as in a permutation-based method in a problem-specific way. The main advantage of this approach is that the metaheuristic's search space is even more basic (i.e., $[0,1)^n$ for n solution elements) and standard operators like uniform crossover and mutation where randomly selected keys are set to new random values can be used, which are entirely independent of the targeted problem.

Decoder-based approaches have also been used in the context of *ant colony optimization* (ACO). For example, Blum and Blesa [20] 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 [20] 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 naive random decisions. While such an operation is computationally cheap, created offspring are often worse than the respective parents, and many repetitions are typically necessary for achieving strong improvements.

Alternatively, one can put more effort into the determination of such offspring. An established technique is *path relinking* [57]. 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 sometimes be solved in reasonable time in practice. Frequently, this underlying problem also is only solved by means of a sub-(meta-)heuristic, yielding a near-optimal merging.

Applegate et al. [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 in the above way, variants of 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 [18] in the context of an EA 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 [45] 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.

Solution merging is also the underlying idea of the general *construct, merge, solve & adapt* (CMSA) algorithm [24] which is suited for problems in which a solution corresponds to a subset of components from a larger base set. CMSA maintains a so-called incumbent sub-instance of the original problem in which only a part of the base set of all components is considered. Initially, this incumbent sub-instance is empty. In each iteration of CMSA, a randomized greedy heuristic is used to generate a set of solutions to the tackled problem. The components of all these solutions are then collected and added to the incumbent sub-instance, and an exact technique like a MIP solver is used to find, if possible within an allotted computation time, a best solution to this sub-instance. The CMSA framework not only adds solution components to the incumbent sub-instance at each iteration, but also contains a mechanism to dispose of seemingly useless solution components. Apart from applications to the minimum common string partition and the minimum weighted arborescence problem in the original paper [24], CMSA has also been successfully applied to problems such as the repetition-free longest common subsequence problem [21] and the MKP [23].

Cotta and Troya [35] discuss merging in the light of a 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 [113] suggests a tight integration of an EA 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 the 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 variables. Since the number of variables to be fixed is a critical parameter, an adaptive scheme is applied to control it. This method is integrated in the commercial MIP solver CPLEX[1] since version 10. See also [76] for further heuristic mechanisms embedded within CPLEX.

Hachemi et al. [61] studied different methods to heuristically integrate (i.e., merge) candidate solutions of a rich multi-depot periodic vehicle routing problem. In particular the authors distinguish between restriction- and incentive-based approaches. In their restriction-based methods they fix critical characteristics of the solutions, while in the incentive-based approaches incentive terms are added to the objective function of the integration-subproblem. The authors conclude that inte-

[1] http://www-01.ibm.com/software/info/ilog

gration operators fixing critical characteristics for which a consensus exist in all input solutions outperform the others when used as stand-alone procedures. In the context of cooperative search (UHGS framework from [124]), however, a mixture also involving the incentive-based methods appears to be more fruitful. Overall, the proposed solution integration procedures exhibit substantial advantages on the performance of the heuristic search.

Parragh and Schmid [87] propose a hybridization of large neighbourhood search and column generation for solving the dial-a-ride problem (DARP). Based on a heuristically obtained initial feasible solution a set covering based column generation scheme is started where new reduced-cost columns are generated using VNS applied on the existing columns (routes). At a certain interval large neighbourhood search is used to improve the current best solution. All routes generated during the LNS phase are added as new columns to the set covering column pool. This approach can be viewed from a solution merging perspective because the routes of multiple solutions are merged into a single DARP solution by combining them optimally using a set covering problem. The obtained computational results are improving the state-of-the-art. In general, heuristic approaches generating multiple solutions for problems that can be decomposed in such a way can strongly benefit from applying a set covering approach for merging them optimally.

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 may 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 [104] 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 operators such as recombination and mutation in EAs—exist.

Plateau et al. [95] 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 [101] 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 [30] 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 heuristic measures (a more detailed analysis is given in [102]).

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 [64]. It is based on a Lagrangian decomposition of a minimum spanning tree-like ILP formulation of the problem. The volume algorithm 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 more on regions of the search space around the results of the Lagrangian heuristic, where also better solutions with respect to the original objective function are likely to be found.

Pirkwieser et al. [93] 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 high quality even for large instances. These solutions are polished by the GA, and, remarkably, in most cases proven optimal solutions are finally obtained.

Dowsland et al. [40] propose an approach where bounding information available from partial solutions is used to guide an EA. 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. That is, 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 tested this concept on a pallet loading and a two-dimensional packing problem.

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. A prominent example of a complete method belonging to this class is B&B. An interesting heuristic derivative of breadth-first B&B is *beam search* [86]. 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 [17, 19] and *approximate and non-deterministic tree search* (ANTS) procedures [77]. 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 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 frequently 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: On the one side, 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 typical target problems for CP applications. Meyer and Ernst [81] introduced the incorporation of CP into ACO in an application to the single machine job scheduling problem.

Raidl and Hu in [108] proposed to enhance a GA by a so-called trie-based complete solution archive, which is in fact a hybridization of a GA with B&B. This relatively general approach is particularly useful for problems with a compact solution representation but expensive solution evaluation, such as methods relying on costly decoders or rigorous simulations. The central idea is to store all created candidate solutions of the GA efficiently in a special trie data structure, which essentially corresponds to an explicitly stored B&B tree. Doing so allows to efficiently check if a created candidate solution has already been considered before. If this is the case, the archive further provides an efficient way to transform the candidate solution into a different but usually similar one, for which it is guaranteed not to have been considered before. Thus, the solution archive can also be seen as an "intelligent mutation" operator effectively avoiding any re-visits. This even implies that the GA is in principle turned into an exact optimization technique that is guaranteed to find an optimal solution in limited time. In practice, however, the approach will typically still be terminated early, i.e., before the whole space of solutions has been covered. This approach was tested in [108] on royal road functions and NK-landscapes with classical binary representations as proof of concept. Later, the usefulness of this hybrid was also shown on a variety of other, practically more relevant problems with different representations, including the generalized minimum spanning tree problem, the discrete $(r|p)$ centroid problem [12], and other competitive facility location problems [13]. The latter works exploit the solution archive also within an embedded local search component turning it into a special kind of tabu search. Furthermore, the principle is extended by also occasionally calculating dual bounds on partial

solutions, effectively including the bounding mechanism of classical B&B [69] in order to prune larger parts of the search space. We refer the reader to [11] for more details on trie-based solution archives.

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 then 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 classical 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 the most promising for reaching 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 used 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 two decades, 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 local search or even a metaheuristic.

Danna et al. [37] 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. Instead 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 [48] proposed *local branching*, an exact approach introducing the classical k-OPT local search idea 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. [63] 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 [49]. 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. [37] 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 is found during this process, it is given 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. Recently, Gomes et al. [59] suggested an extension of RINS that explicitly explores pre-processing techniques. Their method systematically searches for a suitable number of variable fixings to produce subproblems of controlled size, which are explored in a variable-neighborhood-descent fashion.

The *nested partitioning* method proposed by Shi and Ólafsson [114] 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 latter is thus nested within the current most promising region. 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 [115] local search is used for this purpose.

A very different paradigm is followed in *constraint-based local search* [65]. 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 modeling of combinatorial optimization problems in a relatively natural way.

Note that the *construct, merge, solve & adapt* (CMSA) framework [24] previously described in Section 6 in the context of solution merging, can also be seen as a technique in which heuristic elements provide guidance for an exact approach. This is because the reduced sub-instances of the tackled problem instances—which are solved by an exact technique—are generated by iteratively applying a greedy heuristic in a probabilistic way.

Guidance of a complete technique by means of the information gathered by a metaheuristic can also be found in the context of CP. An example is the work of Khichane et al. [73] in which the pheromone information of an ACO algorithm is used for value ordering during CP branching.

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 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 frequently applied 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 (or almost best) neighboring solution is still possible in short time. Such techniques are commonly known as *very large-scale neighborhood search* [3] or just *large neighborhood search* [112]. 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. For further details we refer to the separate chapter in this book specifically dedicated to large neighborhood search.

A frequently found general design principle for large neighborhoods is to select a set of the problem's variables for re-optimization and to fix all remaining variables as they appear in the current incumbent solution. Clearly, the number of variables to be optimized in each iteration and the way to select them have a substantial impact on the performance of the search. Usually variables are selected randomly, but typically in a way to favor the joint selection of strongly related variables that determine certain solution characteristics or to favor variables involved in constraint violations. Different types of large neighborhoods can be obtained through different strategies for variable selection and for re-optimization of the resulting subproblems. Methods for the latter can range from basic greedy heuristics over dynamic programming to MIP or CP solvers.

Numerous applications exist in which large neighborhoods are described in the form of MIPs and a MIP-solver is applied for finding a good—or a best—neighboring solution. Examples of MIP-based large neighborhood search can be found in Duarte et al. [41], where an iterated local search framework is applied to a real-world referee assignment problem, and in Prandtstetter and Raidl [97] where several different MIP-based neighborhoods are searched within a VNS framework for a car sequencing problem. Toledo et al. [121] describe an effective MIP-based "relax-and-fix with fix-and-optimize" heuristic for multi-level lot-sizing problems.

As it is sometimes difficult to decide which large neighborhood is the best to apply at a certain time, adaptive schemes that increase or decrease the probability to apply each neighborhood structure according to its past performance have also been useful. For example Ropke and Pisinger [112] proposed an *adaptive large neighborhood search* (ALNS) for the pickup and delivery problem with time windows, Muller et al. [52] applied an ALNS with MIP neighborhoods to a lot-sizing problem with setup times, and Pereira et al. [89] described such an approach for a probabilistic maximal covering location-allocation problem.

In *Dynasearch* [33] exponentially large neighborhoods are explored by dynamic programming. A neighborhood 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 [46] investigated several such neighborhoods in particular for the traveling salesman problem.

Other 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 to 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 [90] 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 traveling 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. More recent examples of CP-based LNS concern applications to a dial-and-ride problem [70] and a post enrolment-based course timetabling problem [29]. A general modeling language and a hybrid solver specially designed for LNS, called GELATO, was proposed in [31].

Hu et al. [67] describe 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.

A way of defining large neighborhoods in the context of a decomposition approach is proposed in a general framework called POPMUSIC (Partial OPtimization Metaheuristic Under Special Intensification Conditions) [116]. POPMUSIC is thought for the application to generally large-scale optimization problems in which solutions have the property to be composed of parts that can be optimized relatively independently. The basic idea is to identify and to optimize these parts a posteriori once an initial solution to the problem under consideration is obtained. The subproblem solved at each iteration re-shapes at most r parts, where r is a parameter of the algorithm, in the best possible way. This can be seen as a large neighborhood based on *soft fixing* as it is known, for example, from other techniques such as local branching [48], cf. Section 8. Exact solvers often come into play. A recent example is the application of POPMUSIC to a berth allocation problem [75].

Note also that large neighborhood search and POPMUSIC are strongly related to *variable neighborhood decomposition search* [62], where neighborhoods are searched only for selected parts of an incumbent solution.

9.2 Hybrids Based on MIP Decomposition Techniques

Mathematical programming decomposition techniques are methods for solving a large problem by considering a series of smaller problems and appropriately combining the solutions. Lagrangian decomposition, Benders decomposition, and column generation are particularly well known and are used in many state-of-the-art exact and heuristic solution approaches for different combinatorial optimization problems [127]. In fact, these decomposition techniques may also be interpreted as more general metaheuristic frameworks themselves, see [28]. Frequently, these methods can be considerably accelerated with the help of (meta-)heuristics, sometimes even retaining completeness, or combined in fruitful ways with metaheuristics in order

to guide them. See [106] for a survey on such decomposition based hybrids metaheuristics. The possibilities in this context are manifold. Here, we just want to give a few ideas that have already been proven successful in several applications.

9.2.1 Lagrangian Decomposition

With respect to *Lagrangian decomposition* (LD), we have already considered the works from Haouari and Siala [64] and Pirkwieser et al. [93] in Section 7.1, which cleverly exploit the information gathered from LD in various ways within a GA. More generally, LD is in principle only a method for obtaining a lower bound. To get a feasible solution, it typically relies on some further heuristic method that usually exploits the Lagrangian dual.

9.2.2 Column Generation

In *column generation* (CG) one usually aims at solving a MIP model with a huge number of variables. Such models frequently resemble a kind of set covering or set partitioning model and are attractive because they provide a strong LP relaxation. For example, a vehicle routing problem may be modeled in a way where any feasible route corresponds to a variable, and a subset of all routes is sought that covers all customers. Clearly, there are exponentially many routes, and thus variables, and such a model cannot be solved directly in practice. Column generation starts with a reduced model containing only a small set of initial variables, which are for example derived from an initial solution provided by a heuristic. This reduced model is then iteratively solved and augmented by further variables (i.e., columns in the matrix notation of the MIP) that may lead in the next iteration to an improved solution. The subproblem of identifying a new variable whose inclusion will yield an improvement is called the *pricing problem* and is often difficult to solve on its own. Applying fast (meta-)heuristics for this purpose is sometimes a very meaningful option.

For example, Filho and Lorena [111] apply a heuristic CG 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 [100] describe an exact branch-and-price approach for the three-stage two-dimensional bin packing problem. Fast CG is performed by applying a chain 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. Massen et al. [80] use ant colony optimization for heuristic CG to solve a black-box vehicle routing problem.

Alvelos et al [8] describe a general hybrid strategy called *SearchCol* where CG and a metaheuristic are iteratively performed and information is exchanged between them. The metaheuristic works in a problem-independent way trying to find a best

integral solution by searching over combinations of variables identified by CG, while the CG is perturbed in each iteration based on the metaheuristic's result by fixing subproblem variables with special constraints.

9.2.3 Benders Decomposition

Benders decomposition (BD) has been originally suggested for solving large MIPs involving "complicating" integer variables. 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 inequalities on the integer variables. These inequalities, however, are not directly available but are dynamically separated as cuts. According to the classical BD, 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. [110] improved classical BD by introducing phases of local branching on the original problem in order to obtain multiple feasible heuristic solutions quickly. 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. Poojari and Beasley [96] describe such an approach for solving general MIPs in which a GA together with a feasibility pump heuristic are applied to the master problem. The authors argue that a population based metaheuristic like a genetic algorithm is particularly useful as it provides multiple solutions in each iteration giving rise to more Benders cuts. Boland et al. [27] use a proximity search to drive a BD for two-stage mixed-integer linear stochastic programming models.

Extensions of classical BD exist in which the subproblems can contain also integer variables and may be difficult on their own. Especially in these cases, CP and metaheuristics have a great potential for speeding up the overall approach by providing helpful cuts much faster. For example [66] describes a *logic-based BD* in which subproblems are solved by CP. The approach substantially outperforms pure MIP and pure CP approaches on a large class of planning and scheduling problems.

Raidl et al. [107] proposed an exact logic-based BD approach for a bi-level capacitated vehicle routing problem. The authors were able to speed it up considerably by first solving all instances of the master problem as well as all subproblems with a fast variable neighborhood search heuristic. Invalid Benders cuts possibly cutting off feasible solutions may be created. In a second phase, all these heuristically generated Benders cuts undergo a validity check by re-solving exactly the corresponding subproblems with a MIP solver, yielding possibly corrected cuts that replace the invalid ones. When the master problem is solved exactly and no further Benders cuts can be derived, a proven optimal solution is obtained.

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 a tree search. Similarly to cut generation in mixed integer programming, the search space is reduced by propagating constraints from the current state of the search. Usually specialized and standard combinatorial algorithms are used [79] for this purpose. An example of the use of (meta-)heuristic methods in the context of constraint propagation is *local probing* [71].

Galinier et al. [54] presents a tabu search procedure to speed up filtering for generalized all-different constraints. That is:

$$\mathit{SomeDifferent}(X,D,E) = \{(x_1,\ldots,x_n) \in D \mid x_i \neq x_j \; \forall (i,j) \in E\}$$

is defined over variables $X = (x_1,\ldots,x_n)$ with respective domains $D = (D_1,\ldots,D_n)$ and a graph $G = (X,E)$ with edge set E specifying pairs of variables that must be assigned different values. 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 hard constraints as well.

10 Summary and Conclusions

We have reviewed a large number of different approaches for combining traditional metaheuristic strategies with each other or with algorithmic techniques 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 focusing 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 may 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, because they can often be implemented quickly, once an appropriate construction heuristic is available. The next design template that we discussed was solution merging 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 speed up Lagrangian decomposition, column generation, and Benders decomposition.

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 grants P27615 and W1260.

References

1. Aggarwal, C., Orlin, J., Tai, R.: Optimized crossover for the independent set problem. Operations Research **45**(2), 226–234 (1997)
2. Ahuja, R., Orlin, J., Tiwari, A.: A greedy genetic algorithm for the quadratic assignment problem. Computers & Operations Research **27**(10), 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: C. Blum, A. Roli, M. Sampels (eds.) Proceedings of HM 2004 – First International Workshop on Hybrid Metaheuristics, pp. 11–18. Valencia, Spain (2004)
6. Alba, E. (ed.): Parallel Metaheuristics: A New Class of Algorithms. 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, *LNCS*, vol. 4030. Springer (2006)
8. Alvelos, F., de Sousa, A., Santos, D.: Combining column generation and metaheuristics. In: Talbi [119], pp. 285–334
9. 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)
10. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6**(2), 154–160 (1994)
11. Biesinger, B.: Complete solution archives for evolutionary combinatorial optimization: Application to a competitive facility location and stochastic vehicle routing problem. Ph.D. thesis, TU Wien, Institute of Computer Graphics and Algorithms, Vienna, Austria (2016)
12. Biesinger, B., Hu, B., Raidl, G.: A hybrid genetic algorithm with solution archive for the discrete $(r|p)$-centroid problem. Journal of Heuristics **21**(3), 391–431 (2015)
13. Biesinger, B., Hu, B., Raidl, G.: Models and algorithms for competitive facility location problems with different customer behavior. Annals of Mathematics and Artificial Intelligence **76**(1), 93–119 (2015)
14. Biesinger, B., Hu, B., Raidl, G.R.: A variable neighborhood search for the generalized vehicle routing problem with stochastic demands. In: G. Ochoa, F. Chicano (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2015, *LNCS*, vol. 9026, pp. 48–60. Springer (2015)
15. Blesa, M.J., Blum, C., Cangelosi, A., Cutello, V., Di Nuovo, A.G., Pavone, M., Talbi, E. (eds.): Proceedings of HM 2016 – Tenth International Workshop on Hybrid Metaheuristics, *LNCS*, vol. 9668. Springer (2016)
16. Blesa Aguilera, M.J., Blum, C., Roli, A., Sampels, M. (eds.): Proceedings of HM 2005 – Second International Workshop on Hybrid Metaheuristics, *LNCS*, vol. 3636. Springer (2005)
17. Blum, C.: Beam-ACO: Hybridizing ant colony optimization with beam search: An application to open shop scheduling. Computers & Operations Research **32**(6), 1565–1591 (2005)
18. 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)
19. Blum, C.: Beam-ACO for simple assembly line balancing. INFORMS Journal on Computing **20**(4), 618–627 (2008)
20. 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, *LNCS*, vol. 3512, pp. 25–33. Springer (2005)
21. Blum, C., Blesa, M.J.: Construct, merge, solve and adapt: Application to the repetition-free longest common subsequence problem. In: F. Chicano, B. Hu, P. García-Sánchez (eds.)

Proceedings of EvoCOP 2007 – 16th European Conference on Evolutionary Computation in Combinatorial Optimization, no. 9595 in LNCS, pp. 46–57. Springer (2016)
22. 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)
23. Blum, C., Pereira, J.: Extension of the CMSA algorithm: An LP-based way for reducing sub-instances. In: Proceedings of GECCO 2016 – Genetic and Evolutionary Computation Conference, pp. 285–292. ACM (2016)
24. Blum, C., Pinacho, P., López-Ibáñez, M., Lozano, J.A.: Construct, merge, solve & adapt: A new general algorithm for combinatorial optimization. Computers & Operations Research **68**, 75–88 (2016)
25. Blum, C., Raidl, G.R.: Hybrid Metaheuristics – Powerful Tools for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer (2016)
26. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys **35**(3), 268–308 (2003)
27. Boland, N., Fischetti, M., Monaci, M., Savelsbergh, M.: Proximity Benders: a decomposition heuristic for stochastic programs. Journal of Heuristics **22**(2), 181–198 (2015). URL http://link.springer.com/article/10.1007/s10732-015-9306-1
28. Boschetti, M., Maniezzo, V., Roffilli, M.: Decomposition techniques as metaheuristic frameworks. In: V. Maniezzo, T. Stützle, S. Voss (eds.) Matheuristics – Hybridizing Metaheuristics and Mathematical Programming, *Annals of Information Systems*, vol. 10, pp. 135–158. Springer (2009)
29. Cambazard, H., Hebrard, E., O'Sullivan, B., Papadopoulos, A.: Local search and constraint programming for the post enrolment-based course timetabling problem. Annals of Operations Research **194**(1), 111–135 (2012)
30. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics **4**, 63–86 (1998)
31. Cipriano, R., Di Gaspero, L., Dovier, A.: A hybrid solver for large neighborhood search: Mixing gecode and easylocal++. In: M.J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, A. Schaerf (eds.) Proceedings of HM 2009 – 6th International Workshop on Hybrid Metaheuristics, no. 5818 in LNCS, pp. 141–155. Springer (2009)
32. 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)
33. 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)
34. Cotta, C.: A study of hybridisation techniques and their application to the design of evolutionary algorithms. AI Communications **11**(3–4), 223–224 (1998)
35. Cotta, C., Troya, J.M.: Embedding branch and bound within evolutionary algorithms. Applied Intelligence **18**(2), 137–153 (2003)
36. Coudert, D., Nepomuceno, N., Rivano, H.: Power-efficient radio configuration in fixed broadband wireless networks. Computer Communications **33**(8), 898–906 (2010)
37. Danna, E., Rothberg, E., Le Pape, C.: Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, Series A **102**(71), 71–90 (2005)
38. 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)
39. Di Gaspero, L.: Integration of metaheuristics and constraint programming. In: J. Kacprzyk, W. Pedrycz (eds.) Springer Handbook of Computational Intelligence, pp. 1225–1237. Springer (2015)
40. 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)

41. Duarte, A.R., Ribeiro, C.C., Urrutia, S.: A hybrid ils heuristic to the referee assignment problem with an embedded mip strategy. In: T. Bartz-Beielstein, M.J. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, M. Sampels (eds.) Proceedings of HM 2007 – Fourth International Workshop on Hybrid Metaheuristics, *LNCS*, vol. 4771, pp. 82–95. Springer (2007)
42. Dumitrescu, I., Stützle, T.: Combinations of local search and exact algorithms. In: S. Cagnoni, C.G. Johnson, J.J. Romero Cardalda, E. Marchiori, D.W. Corne, J.A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G.R. Raidl, E. Hart (eds.) Applications of Evolutionary Computation, *LNCS*, vol. 2611, pp. 211–223. Springer (2003)
43. Ehrgott, M., Gandibleux, X.: Hybrid metaheuristics for multi-objective combinatorial optimization. In: C. Blum, M.J. Blesa Aguilera, A. Roli, M. Sampels (eds.) Hybrid Metaheuristics – An Emerging Approach to Optimization, *Studies in Computational Intelligence*, vol. 114, pp. 221–259. Springer (2008)
44. El-Abd, M., Kamel, M.: A taxonomy of cooperative search algorithms. In: Blesa Aguilera et al. [16], pp. 32–41
45. Eremeev, A.V.: On complexity of optimal recombination for binary representations of solutions. Evolutionary Computation **16**(1), 127–147 (2008)
46. 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)
47. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**, 109–133 (1995)
48. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming, Series B **98**(1), 23–47 (2003)
49. 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)
50. Fisher, M.L., Jaikumar, R.: A generalized assignment heuristic for vehicle routing. Networks **11**(2), 109–124 (1981)
51. Fleurent, C., Glover, F.: Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS Journal on Computing **11**(2), 198–204 (1999)
52. Flindt Muller, L., Spoorendonk, S., Pisinger, D.: A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. European Journal of Operational Research **218**(3), 614–623 (2012)
53. 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)
54. Galinier, P., Hertz, A., Paroz, S., Pesant, G.: Using local search to speed up filtering algorithms for some np-hard constraints. Annals of Operations Research **184**(1), 121–135 (2011)
55. Gilmour, S., Dras, M.: Kernelization as heuristic structure for the vertex cover problem. In: M. Dorigo, L.M. Gambardella, M. Birattari, A. Martinoli, R. Poli, T. Stützle (eds.) Proceedings of ANTS 2006 – 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, *LNCS*, vol. 4150, pp. 452–459. Springer (2006)
56. Glover, F.: Surrogate constraints. Operations Research **16**(4), 741–749 (1968)
57. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **39**(3), 653–684 (2000)
58. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Learning. Addison-Wesley, Reading, MA (1989)
59. Gomes, T.M., Santos, H.G., Souza, J.F.: A pre-processing aware RINS based MIP heuristic. In: M.J. Blesa, C. Blum, P. Festa, A. Roli, M. Sampels (eds.) Proceedings of HM 2013 – Eighth International Workshop on Hybrid Metaheuristics, *LNCS*, vol. 7919, pp. 1–11. Springer (2013)
60. Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics **17**(5), 487–525 (2011)

61. Hachemi, N.E., Crainic, T.G., Lahrichi, N., Rei, W., Vidal, T.: Solution integration in combinatorial optimization with applications to cooperative search and rich vehicle routing. Journal of Heuristics **21**(5), 663–685 (2015). DOI 10.1007/s10732-015-9296-z. URL http://link.springer.com/article/10.1007/s10732-015-9296-z
62. Hansen, P., Mladenovic, N., Perez-Britos, D.: Variable neighborhood decomposition search. Journal of Heuristics **7**(4), 335–350 (2001)
63. Hansen, P., Mladenović, N., Urosević, D.: Variable neighborhood search and local branching. Computers & Operations Research **33**(10), 3034–3045 (2006)
64. 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)
65. Hentenryck, P.V., Michel, L.: Constraint-Based Local Search. MIT Press, Cambridge, MA (2005)
66. Hooker, J.N.: Planning and scheduling by logic-based Benders decomposition. Operations Research **55**(3), 588–602 (2007)
67. 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), 473–499 (2008)
68. Hu, B., Raidl, G.R.: Effective neighborhood structures for the generalized traveling salesman problem. In: J.I. van Hemert, C. Cotta (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2008, *LNCS*, vol. 4972, pp. 36–47. Springer (2008)
69. Hu, B., Raidl, G.R.: An evolutionary algorithm with solution archives and bounding extension for the generalized minimum spanning tree problem. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 393–400. ACM Press, Philadelphia, PA, USA (2012)
70. Jain, S., Van Hentenryck, P.: Large neighborhood search for dial-a-ride problems. In: J. Lee (ed.) Proceedings of CP 2011 – 17th International Conference Principles and Practice of Constraint Programming, no. 6876 in LNCS, pp. 400–413. Springer (2011)
71. Kamarainen, O., Sakkout, H.E.: Local probing applied to scheduling. In: P. Van Hentenryck (ed.) Proceedings of CP 2002 – 8th International Conference on Principles and Practice of Constraint Programming, no. 2470 in LNCS, pp. 155–171. Springer (2002)
72. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)
73. Khichane, M., Albert, P., Solnon, C.: Strong combination of ant colony optimization with constraint programming optimization. In: A. Lodi, M. Milano, P. Toth (eds.) Proceedings of CPAIOR 2010 – 7th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, no. 6140 in LNCS, pp. 232–245. Springer (2010)
74. Klau, G.W., Lesh, N., Marks, J., Mitzenmacher, M.: Human-guided search. Journal of Heuristics **16**(3), 289–310 (2010)
75. Lalla-Ruiz, E., Voß, S.: POPMUSIC as a matheuristic for the berth allocation problem. Annals of Mathematics and Artificial Intelligence **76**(1), 173–189 (2016)
76. Lodi, A.: The heuristic (dark) side of MIP solvers. In: Talbi [119], pp. 273–284
77. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS Journal on Computing **11**(4), 358–369 (1999)
78. Maniezzo, V., Stützle, T.: Matheuristics 2016 – Proceedings of the Sixth International Workshop on Model-based Metaheuristics. Tech. Rep. TR/IRIDIA/2016-007, IRIDIA, Université libre de Bruxelles, Belgium (2016)
79. Marriott, K., Stuckey, P.J.: Introduction to Constraint Logic Programming. MIT Press, Cambridge, MA (1998)
80. Massen, F., Deville, Y., Hentenryck, P.V.: Pheromone-based heuristic column generation for vehicle routing problems with black box feasibility. In: N. Beldiceanu, N. Jussien, É. Pinson (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems – CPAIOR 2012, *LNCS*, vol. 7298, pp. 260–274. Springer (2012)
81. Meyer, B., Ernst, A.: Integrating ACO and constraint propagation. In: M. Dorigo, M. Birattari, C. Blum, L.M. Gambardella, F. Mondada, T. Stützle (eds.) Proceedings of ANTS 2004 –

Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence, *LNCS*, vol. 3172, pp. 166–177. Springer (2004)
82. Michalewicz, Z., Siarry, P.: Special issue on adaptation of discrete metaheuristics to continuous optimization. European Journal of Operational Research **185**(3), 1060–1273 (2008)
83. Moscato, P.: Memetic algorithms: A short introduction. In: D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, K.V. Price (eds.) New Ideas in Optimization, pp. 219–234. McGraw-Hill (1999)
84. Nepomuceno, N., Pinheiro, P., Coelho, A.L.V.: A hybrid optimization framework for cutting and packing problems. In: C. Cotta, J. van Hemert (eds.) Recent Advances in Evolutionary Computation for Combinatorial Optimization, *Studies in Computational Intelligence*, vol. 153, pp. 87–99. Springer (2008)
85. 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)
86. Ow, P.S., Morton, T.E.: Filtered beam search in scheduling. International Journal of Production Research **26**(1), 297–307 (1988)
87. Parragh, S.N., Schmid, V.: Hybrid column generation and large neighborhood search for the dial-a-ride problem. Computers & Operations Research **40**(1), 490 – 497 (2013)
88. 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)
89. Pereira, M.A., Coelho, L.C., Lorena, L.A.N., de Souza, L.C.: A hybrid method for the probabilistic maximal covering location–allocation problem. Computers & Operations Research **57**, 51–59 (2015)
90. Pesant, G., Gendreau, M.: A constraint programming framework for local search methods. Journal of Heuristics **5**(3), 255–279 (1999)
91. Pinheiro, P.R., Coelho, A.L.V., de Aguiar, A.B., Bonates, T.O.: On the concept of density control and its application to a hybrid optimization framework: Investigation into cutting problems. Computers & Industrial Engineering **61**(3), 463–472 (2011)
92. Pinheiro, P.R., Coelho, A.L.V., de Aguiar, A.B., de Menezes Sobreira Neto, A.: Towards aid by generate and solve methodology: application in the problem of coverage and connectivity in wireless sensor networks. International Journal of Distributed Sensor Networks **2012** (2012). Article ID 790459
93. 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.I. van Hemert (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2007, *LNCS*, vol. 4446, pp. 176–187. Springer (2007)
94. Pisinger, D.: Core problems in knapsack algorithms. Operations Research **47**(4), 570–575 (1999)
95. 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**(6), 731–746 (2002)
96. Poojari, C.A., Beasley, J.E.: Improving Benders decomposition using a genetic algorithm. European Journal of Operational Research **199**(1), 89–97 (2009)
97. 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), 1004–1022 (2008)
98. Prins, C., Lacomme, P., Prodhon, C.: Order-first split-second methods for vehicle routing problems: A review. Transportation Research Part C **40**, 179–200 (2014)
99. 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, *LNCS*, vol. 3562, pp. 41–53. Springer (2005)
100. Puchinger, J., Raidl, G.R.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research **183**(3), 1304–1327 (2007)

101. Puchinger, J., Raidl, G.R.: Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. Journal of Heuristics **14**(5), 457–472 (2008)
102. 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, *LNCS*, vol. 3906, pp. 195–208. Springer (2006)
103. Quimper, C.G. (ed.): Proceedings of CPAIOR 2016 – 13th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, *LNCS*, vol. 9676. Springer (2016)
104. 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)
105. Raidl, G.R.: A unified view on hybrid metaheuristics. In: Almeida et al. [7], pp. 1–12
106. Raidl, G.R.: Decomposition based hybrid metaheuristics. European Journal of Operational Research **244**(1), 66–76 (2015)
107. Raidl, G.R., Baumhauer, T., Hu, B.: Speeding up logic-based Benders' decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem. In: M.J. Blesa, C. Blum, S. Voss (eds.) Proceedings of HM 2014 – Ninth International Workshop on Hybrid Metaheuristics, *LNCS*, vol. 8457, pp. 183–197. Springer (2014)
108. Raidl, G.R., Hu, B.: Enhancing genetic algorithms by a trie-based complete solution archive. In: P. Cowling, P. Merz (eds.) Evolutionary Computation in Combinatorial Optimization – EvoCOP 2010, *LNCS*, vol. 6022, pp. 239–251. Springer (2010)
109. Raidl, G.R., Puchinger, J.: Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In: C. Blum, M.J. Blesa Aguilera, A. Roli, M. Sampels (eds.) Hybrid Metaheuristics – An Emerging Approach to Optimization, *Studies in Computational Intelligence*, vol. 114, pp. 31–62. Springer (2008)
110. Rei, W., Cordeau, J.F., Gendreau, M., Soriano, P.: Accelerating Benders decomposition by local branching. INFORMS Journal on Computing **21**(2), 333–345 (2008)
111. Ribeiro Filho, G., Nogueira Lorena, L.A.: 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 (2000)
112. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4), 455–472 (2006)
113. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS Journal on Computing **19**(4), 534–541 (2007)
114. Shi, L., Ólafsson, S.: Nested partitions method for global optimization. Operations Research **48**(3), 390–407 (2000)
115. Shi, L., Ólafsson, S., Chen, Q.: An optimization framework for product design. Management Science **47**(12), 1681–1692 (2001)
116. Taillard, É.D., Voß, S.: POPMUSIC: Partial optimization metaheuristic under special intensification conditions. In: C.C. Ribeiro, P. Hansen (eds.) Essays and Surveys in Metaheuristics, pp. 613–629. Kluwer Academic Publishers (2001)
117. Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics **8**(5), 541–565 (2002)
118. Talbi, E.G.: Metaheuristics: From Design to Implementation. Wiley & Sons (2009)
119. Talbi, E.G. (ed.): Hybrid Metaheuristics, *Studies in Computational Intelligence*, vol. 434. Springer (2013)
120. Talukdar, S., Baeretzen, L., Gove, A., de Souza, P.: Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics **4**(4), 295–321 (1998)
121. Toledo, C.F.M., Arantes, M.d.S., Hossomi, M.Y.B., França, P.M., Akartunalı, K.: A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. Journal of Heuristics **21**(5), 687–717 (2015). URL http://link.springer.com/article/10.1007/s10732-015-9295-0
122. 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)

123. Vasquez, M., Vimont, Y.: Improved results on the 0–1 multidimensional knapsack problem. European Journal of Operational Research **165**(1), 70–81 (2005)
124. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research **234**(3), 658–673 (2014)
125. Walshaw, C.: Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In: C. Blum, M.J. Blesa Aguilera, A. Roli, M. Sampels (eds.) Hybrid Metaheuristics – An Emerging Approach to Optimization, *Studies in Computational Intelligence*, vol. 114, pp. 261–289. Springer (2008)
126. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
127. Wolsey, L.A.: Integer Programming. Wiley-Interscience (1998)