A Unified View on Hybrid Metaheuristics*

Günther R. Raidl[1]

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{raidl}@ads.tuwien.ac.at

Abstract. Manifold possibilities of hybridizing individual metaheuristics with each other and/or with algorithms from other fields exist. A large number of publications documents the benefits and great success of such hybrids. This article overviews several popular hybridization approaches and classifies them based on various characteristics. In particular with respect to low-level hybrids of different metaheuristics, a unified view based on a common pool template is described. It helps in making similarities and different key components of existing metaheuristics explicit. We then consider these key components as a toolbox for building new, effective hybrid metaheuristics. This approach of thinking seems to be superior to sticking too strongly to the philosophies and historical backgrounds behind the different metaheuristic paradigms. Finally, particularly promising possibilities of combining metaheuristics with constraint programming and integer programming techniques are highlighted.

1 Introduction

Metaheuristics have proven to be highly useful for approximately solving difficult optimization problems in practice. A general overview on this research area can be found e.g. in [1], for more information see also [2, 3]. The term *metaheuristic* was first introduced by Glover [4]. Today, it refers to a broad class of algorithmic concepts for optimization and problem solving, and the boundaries are somewhat fuzzy. Voß [5] gives the following definition:

> A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method.

According to Glover [2],

> ... these methods have over time also come to include any procedure for problem solving that employs a strategy for overcoming the trap of

* This work is supported by the European RTN ADONET under grant 504438.

> local optimality in complex solution spaces, especially those procedures that utilize one or more neighborhood structures as a means of defining admissible moves to transition from one solution to another, or to build or destroy solutions in constructive and destructive processes.

Simulated annealing, tabu search, evolutionary algorithms like genetic algorithms and evolution strategies, ant colony optimization, estimation of distribution algorithms, scatter search, path relinking, the greedy randomized adaptive search procedure (GRASP), multi-start and iterated local search, guided local search, and variable neighborhood search are – among others – often listed as examples of classical metaheuristics, and they have individual historical backgrounds and follow different paradigms and philosophies; see e.g. [2].

Especially over the last years a large number of algorithms were reported that do not purely follow the concepts of one single traditional metaheuristic, but they combine various algorithmic ideas, sometimes also from outside of the traditional metaheuristics field. These approaches are commonly referred to as *hybrid metaheuristics*.

As for metaheuristics in general, there exist various perceptions of what a hybrid metaheuristic actually is. Looking up the meaning of *hybrid* in the current issue (May 2006) of the Merriam Webster dictionary yields

a) something heterogeneous in origin or composition,
b) something (as a power plant, vehicle, or electronic circuit) that has two different types of components performing essentially the same function,

while the current entry in Wiktionary defines this term as

a) offspring resulting from cross-breeding different entities, e.g. different species,
b) something of mixed origin or composition.

The motivation behind such 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. The vastly increasing number of reported applications of hybrid metaheuristics and dedicated scientific events such as the series of *Workshops on Hybrid Metaheuristics* [6, 7] document the popularity, success, and importance of this specific line of research. In fact, today it seems that choosing an adequate hybrid approach is determinant for achieving top performance in solving most difficult problems.

Actually, the idea of hybridizing metaheuristics is not new but dates back to the origins of metaheuristics themselves. At the beginning, however, such hybrids were not so popular since several relatively strongly separated and even competing communities of researchers existed who considered "their" favorite class of metaheuristics "generally best" and followed the specific philosophies in very dogmatic ways. For example, the evolutionary computation community

grew up in relative isolation and followed relatively strictly the biologically oriented thinking. It is mostly due to the no free lunch theorems [8] that this situation fortunately changed and people recognized that there cannot exist a general optimization strategy which is globally better than any other. In fact, to solve a problem at hand most effectively, it almost always requires a specialized algorithm that needs to be compiled of adequate parts.

Several publications exist which give taxonomies for hybrid metaheuristics or particular subcategories [9–14]. The following section tries to merge the most important aspects of these classifications and at some points extends these views. Also, several examples of common hybridization strategies are given. In Section 3, we turn to a unified view on metaheuristics by discussing the pool template. It helps to extract the specific characteristics of the individual classical metaheuristics and to interpret them as a toolbox of key components that can be combined in flexible ways to build an effective composite system. Section 4 refers to a selection of highly promising possibilities for combining metaheuristics with algorithms from two other prominent research fields in combinatorial optimization, namely constraint programming and integer linear programming. Finally, conclusions are drawn in Section 5.

2 Classification of Hybrid Metaheuristics

Figure 1 illustrates the various classes and properties by which we want to categorize hybrids of metaheuristics. Hereby, we combine aspects from the taxonomy introduced by Talbi [10] with the points-of-view from Cotta [9] and Blum et al. [11]. Classifications with particular respect to parallel metaheuristics are partly adopted from El-Abd and Kamel [14] and Cotta et al. [12] and with respect to the hybridization of metaheuristics with exact optimization techniques from Puchinger and Raidl [13].

We start by distinguishing *what* we hybridize, i.e. which kind of algorithms. We might combine (a) different metaheuristic strategies, (b) metaheuristics with certain algorithms specific for the problem we are considering, such as special simulations, or (c) metaheuristics with other more general techniques coming from fields like operations research (OR) and artificial intelligence (AI). Prominent examples for optimization methods from other fields that have been successfully combined with metaheuristics are exact approaches like branch-and-bound, dynamic programming, and various specific integer linear programming techniques on one side and soft computation techniques like neural networks and fuzzy logic on the other side.

Beside this differentiation, previous taxonomies of hybrid metaheuristics [10, 9] primarily distinguish the *level* (or strength) at which the different algorithms are combined: High-level combinations in principle retain the individual identities of the original algorithms and cooperate over a relatively well defined interface; there is no direct, strong relationship of the internal workings of the algorithms. On the contrary, algorithms in low-level combinations strongly de-

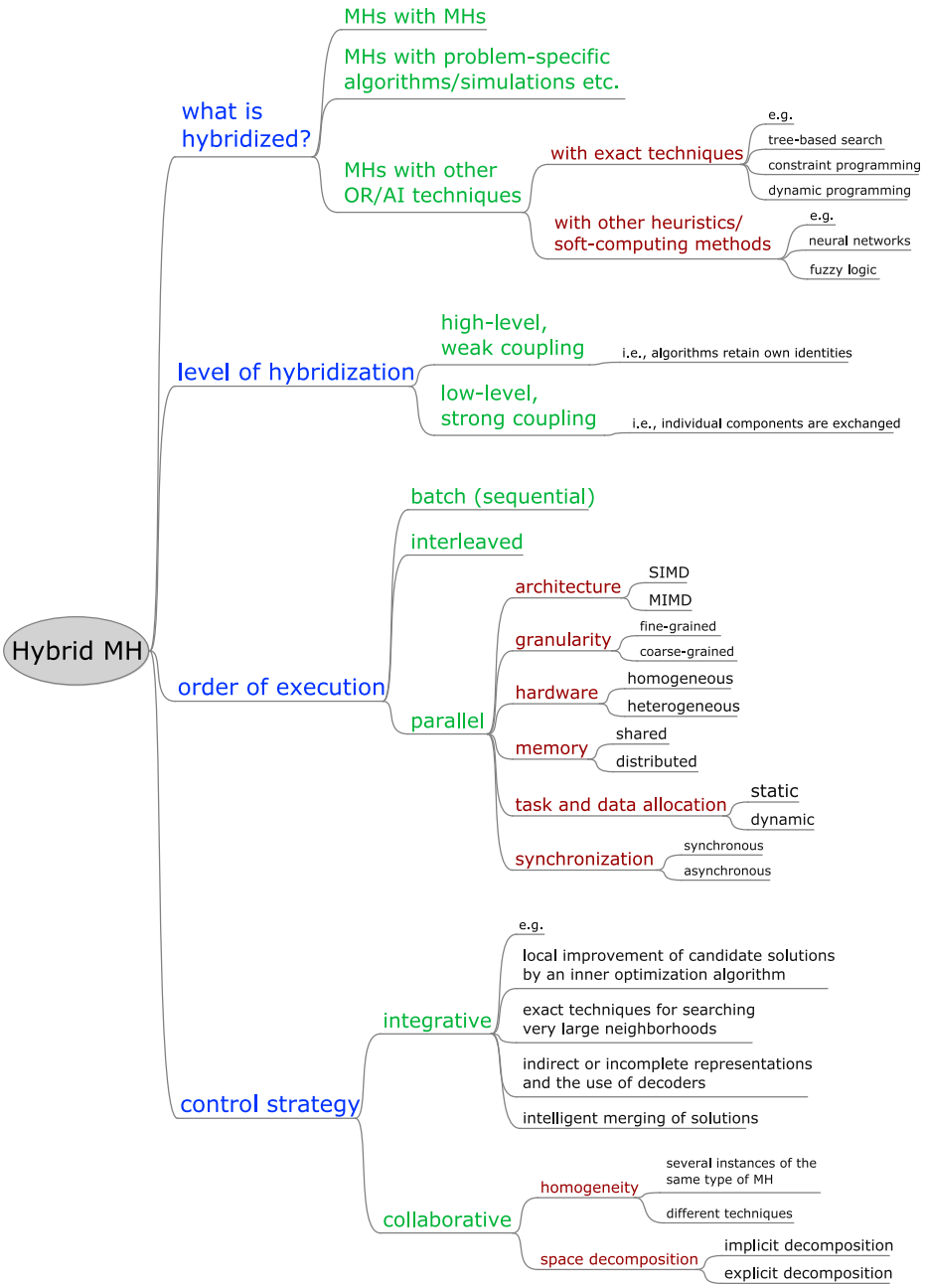

Fig. 1. A summarized classification of hybrid metaheuristics (MHs).

pend on each other – individual components or functions of the algorithms are exchanged.

Another property by which we may distinguish hybrid systems is the *order of execution*. In the batch model, one algorithm is strictly performed after the other, and information is passed only in one direction. An intelligent preprocessing of input data or a postprocessing of the results from another algorithm would fall into this category. Another example are multi-level problems which are solved by considering one level after the other by dedicated optimization algorithms. On the contrary, we have the interleaved and parallel models, in which the algorithms might interact in more sophisticated ways. Parallel metaheuristics are nowadays a large and important research field for their own, see [15]. Detailed classifications of hybrid parallel metaheuristics can be found in [14, 12]. Following general characterizations of parallel algorithms, we can distinguish the architecture (SIMD: single instruction, multiple data streams versus MIMD: multiple instruction, multiple data streams), the granularity of parallelism (fine- versus coarse-grained), the hardware (homogeneous versus heterogeneous), the memory strategy (shared versus distributed memory), the task and data allocation strategy (static versus dynamic), and whether the different tasks are synchronized or run in an asynchronous way.

We can further distinguish hybrid metaheuristics according to their *control strategy*. Following [9, 13], there exist integrative (coercive) and collaborative (cooperative) combinations.

In integrative approaches, one algorithm is considered a subordinate, embedded component of another algorithm. This approach is extremely popular.

- For example, in memetic algorithms [16], various kinds of local search are embedded in an evolutionary algorithm for locally improving candidate solutions obtained from variation operators.
- *Very large scale neighborhood search* (VLSN) approaches are another example [17]. They utilize certain exact techniques such as dynamic programming to efficiently find best solutions in specifically designed large neighborhoods within a local search based metaheuristic.
- Also, any decoder-based metaheuristic, in which a master algorithm acts on an implicit or incomplete representation of candidate solutions and a decoder is used to obtain corresponding actual solutions, falls into this category. Such a decoder can be virtually any kind of algorithm ranging from a simple problem specific heuristic to sophisticated exact optimization techniques or other OR/AI methods. For example in the cutting and packing domain, a common approach is to represent a candidate solution as a permutation of the items that need to be cut out or packed, and an actual solution is derived by considering the items in more or less sophisticated assignment heuristics in the given order, see e.g. [18]. Weight-coding [19] and problem space search [20] are further examples of indirect, relatively generally applicable representations based on decoders.
- Merging solutions: In population based methods such as evolutionary algorithms or scatter search, a traditional variation operator is recombination.

It derives a new solution by combining features of two (or more) parent solutions. Especially in classical genetic algorithms, this operator is based on pure random decisions and therefore works without exploiting any problem specific knowledge. Occasionally, this procedure is replaced by more powerful algorithms like path-relinking [21] or by exact techniques based on branch-and-bound or integer linear programming that identify a best combination of parental features, see e.g. [22, 23].

In collaborative combinations, algorithms exchange information, but are not part of each other. For example, the popular island model [24] for parallelizing evolutionary algorithms falls into this category. We can further classify the traditional island model as a homogeneous approach since several instances of the same metaheuristic are performed. In contrast, Talukdar et al. [25, 26] suggested a heterogeneous framework called *asynchronous teams* (A-Teams). An A-Team is a problem solving architecture consisting of a collection of agents and memories connected into a strongly cyclic directed network. Each of these agents is an optimization algorithm and can work on the target problem, on a relaxation of it, i.e. a superclass, or on a subclass. The basic idea of A-Teams is having these agents work asynchronously and autonomously on a set of shared memories. Denzinger and Offermann [27] presented a similar multi-agent based approach for achieving cooperation between search-systems with different search paradigms, such as evolutionary algorithms and branch-and-bound.

In particular in collaborative combinations, a further question is which search spaces are actually explored by the individual algorithms. According to [14] we can distinguish between an implicit decomposition resulting from different initial solutions, different parameter values etc., and an explicit decomposition in which each algorithm works on an explicitly defined subspace. Effectively decomposing large problems is in practice often an issue of crucial importance. Occasionally, problems can be decomposed in very natural ways, but in most cases finding an ideal decomposition into relatively independent parts is difficult. Therefore, (self-)adaptive schemes are sometimes also used.

3 A Unified View on Hybrid Metaheuristics

The success of all these hybrid metaheuristics tells us that it is usually a bad idea to approach a given (combinatorial) optimization problem with a view that is too restricted to a small (sub-)class of metaheuristics, at least when the primary goal is to solve the problem as well as possible. There is nothing to say against the analogy to real-world phenomena, by which several metaheuristics are explained with or even derived from, for example evolutionary algorithms, ant colony optimization, or simulated annealing. However, one should avoid to focus too strongly on such philosophies, hereby losing the view on particular strengths and benefits of other algorithmic concepts.

Instead of perceiving the various well-known metaheuristics as relatively independent optimization frameworks and occasionally considering hybridization

Algorithm Pool Template
Initialize pool P by an external procedure;
while termination=FALSE **do**
  $S \leftarrow OF(P)$;
  if $|S| > 1$ **then**
    $S' \leftarrow SCM(S)$
  else
    $S' \leftarrow S$;
  $S'' \leftarrow IM(S')$;
  $P \leftarrow IF(S'')$;
Apply a post-optimizing procedure to P.

Fig. 2. The pool template from Voß [30, 31]. P: Pool; IF/OF: Input/Output Function; IM: Improvement Method; SCM: Solution Combination Method.

for achieving certain benefits, it might be advantageous to change the point-of-view towards a unified design concept. All the existing metaheuristics share some ideas and differ among each other by certain characteristic *key components*. Making these key components explicit and collecting them yields a *toolbox* of components from which we can choose in the design of an optimization algorithm as it seems to be most appropriate for the target problem at hand.

In fact, this unified point-of-view is not new. Vaessens et al. [28] already presented a template for representing various kinds of local search based approaches, in particular threshold algorithms, tabu search, variable depth search, and even population based methods such as genetic algorithms. They also addressed multi-level approaches such as genetic local search, where a local search algorithm is applied within a genetic algorithm.

Calégary et al. [29] provided a taxonomy and united view on evolutionary algorithms and exemplarily discussed them with genetic algorithms, ant colony optimization, scatter search, and an emergent colonization algorithm.

Greistorfer and Voß [30, 31] introduced a pool template by which they intend to cover even more different classes of metaheuristics, but especially also population based approaches. It is shown in Figure 2 and follows the definition of metaheuristics as given by Voß in [5] and cited in Section 1. To interpret, for example, simulated annealing in terms of this template, we set $|S| = 1$ and $|P| = 2$. The latter choice seems to be unusual at first glace. However, it covers the fact that we always have a current solution in the pool for which one or more neighbors are evaluated and additionally store the overall so-far best solution. The output function OF always simply returns the current solution. The improvement method IM includes the random choice of a neighboring solution and its evaluation, while the input function IF finally applies the Metropolis criterion (or some other condition) in order to either accept the new solution or to retain the previous one. The temperature update can also be considered to be part of the input function. Obviously, also other derivatives of local search like tabu search, guided local search, iterated local search, variable neighborhood de-

scent/search, but also population-based approaches such as genetic algorithms, evolution strategies, scatter search, and particle swarm optimization can be interpreted as instances of this template in straight-forward ways. Multi-level algorithms like memetic algorithms, where some local search procedure is applied to created candidate solutions are also supported via the improvement method *IM* which might include complete other optimization procedures. The template even matches estimation of distribution algorithms such as an ant colony optimization: The pheromone matrix – or more generally the statistical model – is considered as an additional memory structure, the output function covers the derivation of new solution candidates in dependence of this additional memory, and the input function also includes the memory's update. Examples for solution combination methods (*SCM*) are the classical recombination techniques from genetic algorithms, path relinking, and the merging approaches addressed in the previous section.

Interpreting metaheuristics as instances of such a common template yields a decomposition of the algorithms. In case of the pool template, we obtain individual input and output functions, improvement methods, and eventually solution combination methods. Some of these parts may use auxiliary functions and data structures. From the perspective of functionality, a subset of these parts obtained from the decomposition of the different metaheuristics represents the algorithms' the key components that have been pointed out before. They can be considered to form a joined toolbox from where we can select the most promising parts and combine them in order to build effective (hybrid) optimization approaches tailored to the specific characteristics of the problems at hand. Table 1 summarizes important key components provided by popular metaheuristics.

Some software libraries for metaheuristics, such as HotFrame [32] and EAlib [33], partly support this way of thinking by their object oriented structure and allow flexible combinations of key components when implementing problem-specific optimization algorithms; see also [34].

4 Some Promising Hybridization Possibilities with Other Prominent Combinatorial Optimization Techniques

The previous section mainly addressed low-level hybrids between different types of metaheuristics. Most existing hybrid metaheuristics probably fall into this category. To some degree, the described point-of-view can also be extended towards hybrids of metaheuristics with other OR and AI techniques. In fact, it is often a bad idea to prematurely restrict the possible choices in the design of an optimization algorithm too early to metaheuristic techniques only.

In particular constraint programming (CP) and integer linear programming (ILP) shall be mentioned here as two research fields with long histories and also much success in solving difficult combinatorial optimization problems; see [35] and [36] for introductory textbooks to the two fields, respectively. As metaheuristics, the methods from these fields also have their specific advantages and

Table 1. Some key components of popular metaheuristics.

Ant colony optimization	*OF*: derivation of new solution candidates by considering a pheromone matrix; *SCM*: implicitly via pheromone matrix; *IF*: includes update of pheromone matrix
Genetic algorithms	*OF*, *IF*: selection techniques; *SCM*: crossover operators; *IM*: mutation operators, repair schemes, decoding functions
Guided local search	*IM*, *IF*: augmentation of evaluation function to escape local optima
GRASP	initialization, *OF*: creation of meaningful solutions from scratch by a randomized greedy heuristic
Iterated local search	*IM*: perturbation of a solution for diversification
Multi start approaches	initialization, *OF*: creation of (random) solutions from scratch for diversification
Path relinking	*SCM*: more sophisticated method for combining solutions
Pilot method	*IF*: more sophisticated evaluation and acceptance criterion
Scatter search	*IF*: diversification generation methods, subset generation methods; *IM*: improvement methods; *SCM*: solution combination methods; *OF*: reference set update methods
Simulated annealing	*IF*: acceptance criterion, annealing schedule
Tabu search	*IM*, *IF*: consideration and maintenance of tabu list, aspiration criteria
Variable depth search	*IM*: search of a more sophisticated neighborhood
Variable neighborhood descent	*IM*: search of multiple neighborhoods
Variable neighborhood search	*IM*: shaking in different neighborhoods for diversification
Very large neighborhood search	*IM*: efficient search of a large neighborhood

limits. Especially in the last years, combinations between such techniques and metaheuristics have shown to be often extremely successful and promising.

An overview on hybrids of local search based approaches and constraint programming is given in [37]. Basic concepts include:

- CP can be used as preprocessing for reducing the search space.
- CP techniques can be used to more efficiently search neighborhoods, especially under the existence of difficult problem-specific constraints.
- Special large neighborhoods can sometimes be defined by introducing appropriate artificial constraints, and CP is again used for efficiently searching these neighborhoods.

– In constructive heuristics like GRASP or ant colony optimization, CP can be utilized to make better choices in the selection of the next solution component to be added.

An overview on promising combinations of metaheuristics and integer linear programming (ILP) techniques is given in [13]. Basic concepts include:

– Solving a linear programming or Lagrangian relaxation of the problem often yields valuable information that can be effectively exploited in construction heuristics or variation operators.
– As CP, also ILP has been used to search large neighborhoods.
– ILP can be used for merging solutions.
– Exact ILP techniques are often based on tree search, and good upper and lower bounds are of crucial importance. While for a minimization problem lower bounds are obtained via relaxations, heuristics are important for obtaining upper bounds. Metaheuristics can here be very beneficial.
– In cutting plane techniques such as branch-and-cut inequalities need to be identified which are violated by the current solution to the linear programming (LP) relaxation, but which are valid for the integer optimum. These inequalities are then added to the system and the LP is resolved, yielding an improved bound. The identification of such violated inequalities often is a hard problem for its own, which can be approached by metaheuristics.
– Column generation techniques such as branch-and-price start with a small, restricted set of variables. When having solved this reduced problem, variables being part of the model but currently not included are identified whose insertion enable a further improvement of the current solution; the whole process is repeated. The task of finding such variables is often difficult and metaheuristics have been successfully used for solving it [38].
– Last but not least, some promising concepts such as local branching [39] exist which bring the idea of local search based metaheuristics into linear programming based branch-and-bound: Specific neighborhood-defining inequalities are added to subproblems and branching is controlled in order to perform a "virtual" metaheuristic within tree search.

5 Conclusions

Manifold possibilities of hybridizing individual metaheuristics with each other and/or with algorithms from other fields exist. A large number of publications documents the great success and benefits of such hybrids. Based on several previously suggested taxonomies, a unified classification and characterization of meaningful hybridization approaches has been presented. Especially with respect to low-level hybrids of different metaheuristics, a unified view based on a common pool template can be advantageous. It helps in making different key components of existing metaheuristics explicit. We can then consider these key components as a toolbox and build an effective (hybrid) metaheuristic for a problem at hand by selecting and combining the most appropriate components. This approach of

thinking seems to be superior to sticking too strongly to the philosophies and historical backgrounds behind the different metaheuristic paradigms. Finally, particularly promising possibilities of combining metaheuristics with constraint programming and integer programming techniques were pointed out.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys **35**(3) (2003) 268–308
2. Glover, F., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer (2003)
3. Hoos, H.H., Stützle, T.: Stochastic Local Search. Morgan Kaufmann (2005)
4. Glover, F.: Future paths for integer programming and links to artificial intelligence. Decision Sciences **8** (1977) 156–166
5. Voß S., Martello, S., Osman, I.H., Roucairo, C.: Meta-Heuristics: Andvances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston (1999)
6. Blum, C., Roli, A., Sampels, M., eds.: Proceedings of the First International Workshop on Hybrid Metaheuristics, Valencia, Spain (2004)
7. Blesa, M.J., Blum, C., Roli, A., Sampels, M., eds.: Hybrid Metaheuristics: Second International Workshop. Volume 3636 of LNCS. (2005)
8. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 67–82
9. Cotta, C.: A study of hybridisation techniques and their application to the design of evolutionary algorithms. AI Communications **11**(3–4) (1998) 223–224
10. Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics **8**(5) (2002) 541–565
11. Blum, C., Roli, A., Alba, E.: An introduction to metaheuristic techniques. In: Parallel Metaheuristics, a New Class of Algorithms. John Wiley (2005) 3–42
12. Cotta, C., Talbi, E.G., Alba, E.: Parallel hybrid metaheuristics. In Alba, E., ed.: Parallel Metaheuristics, a New Class of Algorithms. John Wiley (2005) 347–370
13. 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. Volume 3562 of LNCS., Springer (2005) 41–53
14. El-Abd, M., Kamel, M.: A taxonomy of cooperative search algorithms. In Blesa, M.J., Blum, C., Roli, A., Sampels, M., eds.: Hybrid Metaheuristics: Second International Workshop. Volume 3636 of LNCS., Springer (2005) 32–41
15. Alba, E., ed.: Parallel Metaheuristics, a New Class of Algorithms. John Wiley, New Jersey (2005)
16. Moscato, P.: Memetic algorithms: A short introduction. In Corne, D., et al., eds.: New Ideas in Optimization. McGraw Hill (1999) 219–234
17. 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) (2002) 75–102
18. Puchinger, J., Raidl, G.R.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research, Feature Issue on Cutting and Packing (to appear 2006)
19. Julstrom, B.A.: Strings of weights as chromosomes in genetic algorithms for combinatorial problems. In Alander, J.T., ed.: Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications. (1997) 33–48

20. Storer, R.H., Wu, S.D., Vaccari, R.: New search spaces for sequencing problems with application to job-shop scheduling. Management Science **38** (1992) 1495–1509
21. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **39**(3) (2000) 653–684
22. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On the solution of the traveling salesman problem. Documenta Mathematica **Vol. ICM III** (1998) 645–656
23. Cotta, C., Troya, J.M.: Embedding branch and bound within evolutionary algorithms. Applied Intelligence **18** (2003) 137–153
24. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Learning. Addison-Wesley, Reading, MA (1989)
25. Talukdar, S., Baeretzen, L., Gove, A., de Souza, P.: Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics **4** (1998) 295–321
26. Talukdar, S., Murty, S., Akkiraju, R.: Asynchronous teams. In: Handbook of Metaheuristics. Volume 57. Kluwer Academic Publishers (2003) 537–556
27. Denzinger, J., Offermann, T.: On cooperation between evolutionary algorithms and other search paradigms. In: Proceedings of the Congress on Evolutionary Computation 1999, IEEE Press (1999)
28. Vaessens, R., Aarts, E., Lenstra, J.: A local search template. In Manner, R., Manderick, B., eds.: Parallel Problem Solving from Nature, Elsevier (1992) 67–76
29. Calégari, P., Coray, G., Hertz, A., Kobler, D., Kuonen, P.: A taxonomy of evolutionary algorithms in combinatorial optimization. Journal of Heuristics **5**(2) (1999) 145–158
30. Greistorfer, P., Voß, S.: Controlled pool maintenance in combinatorial optimization. In Rego, C., Alidaee, B., eds.: Metaheuristic Optimization via Memory and Evolution – Tabu Search and Scatter Search. Volume 30 of Operations Research/Computer Science Interfaces. Springer (2005) 382–424
31. Voß, S.: Hybridizing metaheuristics: The road to success in problem solving (2006) Slides of an invited talk at the EvoCOP 2006, the 6th European Conference on Evolutionary Computation in Combinatorial Optimization, Budapest, Hungary, http://www.ads.tuwien.ac.at/evocop/Media:Invited-talk-EvoCOP2006-voss.pdf.
32. Fink, A., Voß, S.: HotFrame: A heuristic optimization framework. In Voß S., Woodruff, D.L., eds.: Optimization Software Class Libraries. OR/CS Interfaces Series. Kluwer Academic Publishers (1999)
33. Wagner, D.: Eine generische Bibliothek für Metaheuristiken und ihre Anwendung auf das Quadratic Assignment Problem. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms (2005)
34. Voß S., Woodruff, D.L., eds.: Optimization Software Class Libraries. OR/CS Interfaces Series. Kluwer Academic Publishers (2002)
35. Marriott, K., Stuckey, P.: Programming with Constraints. The MIT Press (1998)
36. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley (1988)
37. Focacci, F., Laburthe, F., Lodi, A.: Local search and constraint programming. In: Handbook of Metaheuristics. Volume 57. Kluwer Academic Publishers (2003) 369–403
38. Puchinger, J., Raidl, G.R., Gruber, M.: Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem. In: Proceedings of MIC2005, the 6th Metaheuristics International Conference, Vienna, Austria (2005) 775–780
39. Fischetti, M., Lodi, A.: Local Branching. Mathematical Programming Series B **98** (2003) 23–47