METAHEURISTICS[1]

Kenneth Sörensen
University of Antwerp, Belgium

Fred Glover
University of Colorado and OptTek Systems, Inc., USA

1 Definition

A *metaheuristic* is a high-level problem-independent algorithmic *framework* that provides a set of guidelines or strategies to develop heuristic optimization algorithms (Sörensen and Glover, To appear). Notable examples of metaheuristics include genetic/evolutionary algorithms, tabu search, simulated annealing, and ant colony optimization, although many more exist. A problem-specific implementation of a heuristic optimization *algorithm* according to the guidelines expressed in a metaheuristic framework is also referred to as a metaheuristic. The term was coined by Glover (1986) and combines the Greek prefix *meta-* (metá, beyond in the sense of *high-level*) with *heuristic* (from the Greek heuriskein or euriskein, to search).

Metaheuristic algorithms, i.e., optimization methods designed according to the strategies laid out in a metaheuristic framework, are — as the name suggests — always *heuristic* in nature. This fact distinguishes them from *exact* methods, that do come with a proof that the optimal solution will be found in a finite (although often prohibitively large) amount of time. Metaheuristics are therefore developed specifically to find a solution that is "good enough" in a computing time that is "small enough". As a result, they are not subject to combinatorial explosion – the phenomenon where the computing time required to find the optimal solution of NP-hard problems increases as an exponential function of the problem size.

Metaheuristics have been demonstrated by the scientific community to be a viable, and often superior, alternative to more traditional (exact) methods of mixed-integer optimization such as branch and bound and dynamic programming. Especially for complicated problems or large problem instances, metaheuristics are often able to offer a better trade-off between solution quality and computing time. Moreover, metaheuristics are more flexible than exact methods in two important ways. First, because metaheuristic frameworks are defined in general terms, metaheuristic algorithms can be adapted to fit the needs of most real-life optimization problems in terms of expected solution quality and allowed computing time, which can vary greatly accross different problems and different situations. Secondly, metaheuristics do not put any demands on the formulation of the optimization problem (like requiring constraints or objective functions to be expressed as linear functions of the decision variables). However, this flexibility comes at the cost of requiring considerable problem-specific adaptation to achieve good perfomance.

The research field of metaheuristics is not without its critics – most of whom attack the perceived lack of universally applicable design methodology, the lack of scientific rigor in testing and comparing different implementations, and the tendency to create overly intricate methods with many different operators. Yet it is hard to

[1] (in press) In Gass, S.I. and M.C. Fu (eds) *Encyclopedia of Operations Research and Management Science* (3e) Springer, New York.

argue with success. The ability to obtain good solutions where other methods fail has made metaheuristics the method of choice for solving a majority of large real-life optimization problems, both in academic research and in practical applications. As a result, several commercial software vendors have implemented metaheuristics as their primary optimization engines, both in specialized software packages for production scheduling, vehicle routing (Sörensen et al., 2008) and nurse rostering (Burke et al., 2004) as well as in general-purpose optimization and simulation packages (April et al., 2003, Fu, 2002, Glover et al., 1999).

The underlying foundations of different metaheuristics vary significantly. Some model the optimization process by using a metaphor seemingly unrelated to optimization, such as natural evolution (genetic/evolutionary algorithms), the cooling of a crystalline solid (simulated annealing), or the behavior of animal swarms (e.g., ant colony optimization). Others, like tabu search, do not use such an intermediary level of explanation, but rather focus on exploiting the problem structure to improve the search for good solutions. In general, metaheuristics frameworks rely heavily on the use of randomness, although some completely deterministic strategies have also been proposed.

Most metaheuristic frameworks have their origin in the 80's (though in some cases roots can be traced to the mid 60's and 70's) and have enjoyed a steady rise in both use and popularity since the early 80's. The metaheuristics field is currently the subject of a number of dedicated journals and conferences. EU/ME – the metaheuristics community[2] is the EURO-sponsored working group on metaheuristics and, with about 1400 members, the largest platform for communication among metaheuristics researchers worlwide.

2 A taxonomy of metaheuristics

Metaheuristic algorithms attempt to find the best (feasible) solution out of all possible solutions of an optimization problem. To this end, they evaluate potential solutions and perform a series of operations on them in order to find different, better solutions. Metaheuristics operate on a *representation* or *encoding* of a solution, an object that can be stored in computer memory and can be conveniently manipulated by the different operators employed by the metaheuristic. Three fundamental classes of metaheuristics can be distinguished, based on the way in which solutions are manipulated. *Local search* metaheuristics iteratively make small changes to a single solution. *Constructive* metaheuristics construct solutions from their constituting parts. *Population-based* metaheuristics iteratively combine solutions into new ones. However, these classes are not mutually exclusive and many metaheuristic algorithms combine ideas from different classes. Such methods are called *hybrid* metaheuristics.

2.1 Local search metaheuristics

Local search metaheuristics find good solutions by iteratively making changes to a single solution, called the *current solution*. These changes are called *moves* and are typically "small" (so that adjacent solutions are relatively close to each other according to a natural metric), hence the name of this class of metaheuristics. The set of solutions that can be obtained by applying a single move to a given solution is called the *neighborhood* of that solution. Depending on the way the solution is

2 http://metaheuristics.eu

represented, different move *types* can be defined. Each move type gives rise to a *neighborhood structure*.

In each iteration, the current solution is replaced by a solution from its neighborhood. The rule used to select the new current solution is called the *move strategy* or *search strategy*. A common search strategies is the *steepest descent* or *steepest ascent* strategy, in which the best move from the neighborhood is selected. Metaheuristics that use this strategy are often called *hill-climbers*. Other move strategies include the *mildest ascent/descent*, also called *first improving*, strategy, in which the first move is selected that improves the current solution. Selecting a random improving solution is another commonly used move strategy.

A solution that is better than any solution in its neighborhood is called a *local optimum* (as opposed to a *global optimum*, i.e., a best possible solution to the optimization problem). When the current solution is a local optimum, a metaheuristic will use a strategy to "escape" this local optimum. It is this strategy that characterizes a metaheuristic, and usually the name of the metaheuristic is derived from it.

Two simple, but commonly used, strategies are to apply a large random change (called *perturbation*) to the current solution or restart the search from a new random solution altogether. These strategies are called *iterated local search* (ILS) or *multi-start local search* respectively (Lourenco et al., 2003).

A second strategy is motivated by the fact that a local optimum relative to a specific move type can often be improved by using another move type. To exploit this fact, some metaheuristics define different move types and change the move type used once a local optimum has been reached. Such metaheuristics are commonly called *variable neighborhood search* (VNS) algorithms (Mladenović and Hansen, 1997). However, using more than one neighborhood is far more common in the metaheuristics literature and not restricted to algorithms labeled VNS (Sörensen et al., 2008).

A third strategy to find good solutions is to use information on the past progress of the search and record this information in memory structures. Metaheuristics that use this strategy are commonly grouped under the umbrella term *tabu search* (Glover, 1989, 1990, 1996) algorithms (sometimes also called *adaptive memory programming* algorithms). Various types of memory structures are commonly used to remember specific properties of the trajectory through the search space that the algorithm has undertaken. A *tabu list* (from which the name of the metaheuristic framework derives) records the last encountered solutions (or some attributes of them) and forbids these solutions (or solutions containing one of the attributes) from being visited again as long as they are on the list. Alternatively, the tabu list may also record the last moves that have been made for the purpose of preventing them from being reversed. Whereas a tabu list can be viewed as a type of short-term memory, that records information on recently visited solutions, *frequency memory* is used as a type of long-term memory. This memory structure records how often certain attributes have been encountered in solutions on the search trajectory, which allows the search to avoid visiting solutions that display the most often encountered attributes or to visit solutions with attributes seldom encountered. The decision on how to use the frequency memory can be based on the quality of the solutions in which the attributes were found, e.g., favoring attributes found in high-quality solutions. The metaheuristic called *guided local search* (GLS) (Voudouris and Tsang, 1999) introduces a different type of memory, called an *augmented objective function*, that includes a *penalty factor* for each potential element. When a local optimum is reached, the penalty factor for for all elements of the current solution is increased,

which makes other elements (and therefore other moves) more attractive. This in turn allows the search to escape from the local optimum. (Tabu search also sometimes employs penalties as a way of implementing tabu restrictions.)

One of the first metaheuristics developed, *simulated annealing*, mimics the annealing process of a crystalline solid. At each iteration, a random solution x' is selected from the neighborhood of the current solution x. This solution is "accepted" as the new current solution with probability $e^{-[f(x')-f(x)]/T}$ where $f(.)$ is the objective function value (to be maximized) of the solution between brackets and T is an endogenous parameter called the *temperature*. The probability of a solution being accepted is therefore higher if the solution is better, but also if the temperature is higher. The temperature is initially set to a high value, which leads to higher acceptance probabilities, and then gradually lowered as the search progresses (although it may be increased again at certain moments during the search). The function that describes the evolution of T throughout the different iterations is called the *cooling schedule*. Simulated annealing was first described in (Kirkpatrick et al., 1983), based upon an algorithm by Metropolis et al. (1953).

The recently proposed metaheuristic called *relaxation induced local search* (RINS) (Danna et al., 2005) constructs promising neighborhoods using information contained in the continuous relaxation of the mixed integer programming (MIP) model of the optimization problem. RINS has the advantage over other local search metaheuristics that it requires less problem-specific information, but this comes at the price of requiring the problem to be formulated as an MIP (Danna, 2004). This makes it easy to integrate RINS in a general-purpose MIP solver the method is currently available in the latest versions of LINDO/LINGO and CPLEX.

2.2 Constructive metaheuristics

Constructive metaheuristics, as their name suggests, *construct* solutions from their constituting *elements* rather than improving complete solutions. This is done by adding one element at a time to a partial solution, an operation that is also called a *move*. Constructive metaheuristics are often adaptations of *greedy* algorithms that add the best possible element at each iteration. To improve the quality of the final solutions, most constructive metaheuristics include a local search phase after the construction phase.

GRASP, the acronym for *greedy randomized adaptive search procedure* (Feo and Resende, 1995), dampens the greediness of a constructive metaheuristic by using randomization. The most common variant of GRASP uses the following strategy. At each iteration, a *restricted candidate list* is updated, that contains the α best elements that can be added to the partial solution. A random element is selected from this list for addition, after which the list is updated to reflect the new situation. The parameter α determines the "greediness" of the search: if α equals 1, the search is completely greedy whereas if α is equal to the number of elements that can be added, the search is completely random. GRASP algorithms are often combined with a path relinking strategy (discussed later), see e.g., Commander et al. (2008), Nascimento et al. (2010), Resende et al. (2010).

Another way to improve the performance of the construction process, without resorting to randomness, is by using memory. Notable examples of metaheuristics that do this can be found in Fleurent and Glover (1999) and Glover et al. (2000). Similarly, *look-ahead strategies* (Pearl, 1984) evaluate the elements that can be added

by considering the effect not only of the next move, but of several moves into the future. The *pilot method* (Duin and Voß, 1999), for example, is a look-ahead method that uses a constructive heuristic to determine the value of a potential element by generating a complete solution from the current partial solution with this element added.

Ant colony optimization (ACO) (Dorigo et al., 1996, 2006) is an umbrella term for a set of related constructive metaheuristics that build solutions by mimicking the foraging behavior of ants. To this end, an external parameter for each potential element (called the *pheromone* level) is introduced. A pheromone is a chemical factor that triggers a social response to other animals of the same species. Ant colony optimization employs multiple artificial agents (called *ants*) that each construct a solution in parallel. Once each ant has constructed a solution, the pheromone level of each element in this solution is updated to allocate more pheromone to elements that lie in better solutions. This information is then used in the construction process of ACO, which selects elements based on a combination of the value of that element and its pheromone level. The process of ants constructing solutions is repeated, and elements that were present in high quality solutions will receive a larger probability of being selected as a result of their higher pheromone levels. Periodically, the pheromone level of all elements is reduced to reflect evaporation. Ant colony optimization has received and continues to receive widespread attention in the popular press (e.g., Anonymous, 2010), probably as a result of the intuitive appeal of the metaphor.

2.3 Population-based metaheuristics

Population-based metaheuristics find good solutions by iteratively *selecting* and then *combining* existing solutions from a set, usually called the *population*. The most important members of this class are *evolutionary algorithms* because they mimick the principles of natural evolution. We use the term *evolutionary algorithms* as an umbrella term to encompass the wide range of metaheuristics based on evolution. This includes genetic algorithms (Goldberg et al., 1989, Holland, 1975) genetic/evolutionary programming (Koza, 1992), evolutionary computation (Fogel, 2006) evolution strategies (Beyer and Schwefel, 2002), and many others.

Evolutionary algorithms operate on a set or population of solutions and use two mechanisms to search for good solutions: the *selection* of predominantly high-quality solutions from the population and the *recombination* of those solutions into new ones, using specialized operators that combine the attributes of two or more solutions. After recombination, new solutions are *reinserted* into the population, possibly requiring them to satisfy conditions such as feasibility or minimum quality demands, to replace other (usually low-quality) solutions. Operators used in evolutionary algorithms (selection, recombination and reinsertion) almost without exception make heavy use of randomness. A *mutation* operator that randomly changes a solution after it has been recombined, is also frequently applied. Most evolutionary algorithms iterate the selection, recombination, mutation, and reinsertion phases a number of times, and report the best solution in the population.

Deterministic population-based alternatives for evolutionary algorithms are *scatter search* and *path relinking* (Glover et al., 2000, 2003). Scatter search encodes solutions as (rounded) real-valued vectors and finds new solutions by generating convex or concave linear combinations of these vectors. Path relinking introduces the concept of a *path* between high-quality solutions, essentially a generalization of the

concept of linear combination. Paths consist of elementary moves such as the ones used in local search metaheuristics. The moves on a path transform one solution (called the *initiating* solution) into a second solution (called the *guiding* solution) one move at a time. Path relinking can therefore be considered a local search heuristic that uses a move strategy in which the move to execute is chosen based on the fact that this move will bring the solution "closer" to the guiding solution. The selection of initiating and guiding solutions from a population (called the *reference set*), as well as the updating of the reference set once new solutions have been generated, are done according to deterministic rules in both path relinking and scatter search.

2.4 "Hybdrid" metaheuristics

In recent years, there is a tendency to view metaheuristic frameworks as providing general ideas or components that can be used to build optimization methods, rather than as cook book recipes that need to be closely followed (Michalewicz and Fogel, 2004). As a result, most recent metaheuristic algorithms combine ideas from different classes and the term *hybrid metaheuristic* has lost most of its discriminatory power. Many modern metaheuristics use specialized heuristics to efficiently solve subproblems produced by the metaheuristic method (e.g., Gendreau et al., 1994). Similarly, a large number of local search metaheuristics use a construction phase to find an initial solution (or a set of initial solutions) from which to start the neighborhood search. In fact the original description of the GRASP metaheuristic (Feo and Resende, 1995) prescribes a local search phase to follow the greedy randomized construction phase.

Algorithms belonging to the class of *memetic algorithms* (the only type of hybrid metaheuristic that has been given a specific name) (Moscato, 1989) combine recombination operators from the class of evolutionary algorithms with local search (meta)heuristics.

2.5 Metaheuristics and exact methods

Algorithmic developments in both metaheuristics and exact methods have recently drawn the two fields closely together, and combinations of metaheuristic components (usually local search) with exact methods for (mixed integer) linear programming are now common. Sometimes called *matheuristics*, the resulting methods often integrate existing exact procedures to solve subproblems generated by a decomposition strategy, a restriction strategy or a relaxation strategy (see, e.g., Glover and Klingman, 1988, Rego, 2005). The results of solving these subproblems are used to guide a higher-level heuristic (Dumitrescu and Stützle, 2009, Raidl and Puchinger, 2008).

Several additional ways in which exact methods can improve the performance of metaheuristics have been reported. Exact methods can sometimes solve small instances of a complex problem effectively. A metaheuristic may operate by constructing collections of such small instances as a strategy for generating "structured moves" that transition from a given solution to a new one (see, e.g., Glover, 2005). Also, an exact method can be run for a very long time to obtain optimal solutions (at least to some instances of a problem class), and these optimal solutions can be used in the learning approach called target analysis (Glover, 1990, Glover and Laguna, 1997) as a way to produce improved decision rules for both metaheuristics and exact methods.

The result of combining a metaheuristic and an exact method does not

necessarily have to be a heuristic method. Metaheuristics can be integrated with exact methods to improve the performance of the exact methods (Friden et al., 1989, Glover, 1990, Puchinger et al., 2009).

In a similar way, ideas and operators from *constraint programming techniques* have been integrated with metaheuristics, such as in the approach called *constraint-based local search* (Van Hentenryck and Michel, 2009).

3 Metaheuristics for different optimization problems

3.1 Continuous optimization

Metaheuristics are predominantly used for combinatorial optimization, but can be effectively adapted for continuous optimization, although this adaptation process is more involved for some metaheuristics than for others. Scatter search (Glover et al., 2000), particle swarm optimization (Kennedy et al., 1995) and an evolutionary approach called differential evolution (Storn and Price, 1997) are very naturally adapted to continuous problem domains. Most constructive and local search approaches on the other hand, require a considerable adaptation from their original formulation. Nonetheless, algorithms for continuous optimization based on tabu search (Chelouah and Siarry, 2000, Glover, 1994), GRASP (Hirsch et al., 2007), variable neighborhood search (Liberti and Drazič, 2005), and others, have been proposed.

3.2 Multi-objective optimization

Many optimization problems have multiple (conflicting) objectives, essentially rendering the concept of optimality meaningless since the best solution for one objective may not be the best for another. In *multi-objective optimization* the concept of *dominance* is therefore introduced. A solution is said to dominate another solution if its quality is at least as good on every objective and better on at least one. The set of all non-dominated solutions of an optimization problem is called the *Pareto set* and the projection of this set onto the objective function space is called the *Pareto front*. The aim of multi-objective metaheuristics is to approximate the Pareto front as closely as possible (Zitzler et al., 2004) and therefore generate a set of mutually non-dominated solutions called the *Pareto set approximation*. Notwithstanding some exceptions (e.g., Czyżak et al., 1998, Hansen, 1997), most multi-objective metaheuristics belong to the class of evolutionary algorithms (Jones et al., 2002). This can be explained by observing that these algorithms naturally operate on a set of solutions. Examples of evolutionary multi-objective metaheuristics are the vector evaluated genetic algorithm (VEGA) (Schaffer, 1985), the non-dominated sorting algorithm (NDSA) (Srinivas and Deb, 1994), the multi-objective genetic algorithm (MOGA) (Fonseca and Fleming, 1993) and the improved strength pareto evolutionary algorithm (SPEA2) (Zitzler and Thiele, 1999).

3.3 Stochastic optimization

Stochastic (combinatorial) optimization problems include uncertain, stochastic or

dynamic information in their parameters. The objective function value and the violation of constraints of such problems are therefore random variables. Evaluating a solution's objective function value and/or it's feasibility can be done either exactly (if a closed-form expression is available), by approximation or by Monte Carlo simulation. Metaheuristics using each of these possibilities have been proposed to solve different stochastic problems (Bianchi et al., 2009, Ribeiro and Resende, 2010).

4 Research in metaheuristics

4.1 Conferences

MIC, the *metaheuristics international conference* is the premier conference on metaheuristics. A yearly *EU/MEeting* is organized by EU/ME in collaboration with a research group and focuses on a specific (and changing) topic. The *Matheuristics* conference series has recently emerged to discuss combinations of metaheuristics with exact methods. The *Learning and Intelligent Optimization* conferences aim at exploring the boundaries between machine learning, artificial intelligent, mathematical programming and algorithms for optimization.

Many conferences are dedicated exclusively to evolutionary algorithms. These include *Parallel Problem Solving From Nature* (PPSN), the *Genetic and Evolutionary Computation Conference* (GECCO), *EvoStar* (a multi-conference comprising EuroGP, EvoCOP, EvoBIO, and EvoApplications), *Evolutionary Multi-Criterion Optimization* (EMO), and the *IEEE Congress on Evolutionary Computation* (CEC).

The *Ants* conference series focuses on research in swarm intelligence methods.

Besides these specialized conferences, metaheuristics hold a prominent position in general Operations Research conferences such as INFORMS, IFORS, and EURO.

4.2 Journals

Several scientific journals are dedicated to the topic of metaheuristics. The *Journal of Heuristics* is the most important one, and also the oldest journal to exclusively focus on (meta)heuristics. Two relatively young journals, the *International Journal of Metaheuristics* and the *International Journal of Applied Metaheuristic Computing* (IJAMC)), have recently been founded. However, a large majority of articles on metaheuristics are published in general Operations Research journals.

Again, the field of evolutionary algorithms has its own share of journals: *Evolutionary Computation*, *IEEE Transactions on Evolutionary Computation*, *Genetic Programming and Evolvable Machines*, and the *Journal of Artificial Evolution and Applications*.

The swarm intelligence area has a dedicated journal appropriately called *Swarm Intelligence*.

5 Metaheuristics software

Most metaheuristics require considerable problem-specific design and tuning before they achieve world-class performance. Although a great deal of research effort is currently being invested in the development of more robust methods, the need for problem-specific design in order to obtain the best results has not inhibited the use of metaheuristics in general optimization software.

Nevertheless, several vendors of commercial general-purpose optimization software have included metaheuristics in their packages. Frontline Systems' *Risk Solver Platform* and its derivatives, an extension of the Microsoft Excel solver, include a hybrid evolutionary solver. Tomlab/GENO is a package for static or dynamic, single- or multi-objective optimization based on a real-coded genetic algorithm. Both LINDO/LINGO and CPLEX include the relaxation induced neighborhood search (RINS) metaheuristic.

The COIN-OR library has several (open source) metaheuristics software packages: *METSlib*, an object oriented metaheuristics optimization framework, and *Open Tabu Search* (OTS), a framework for constructing tabu search algorithms.

Besides these solvers for combinatorial optimization, most commercial simulation packages today include an optimization tool (Fu, 2002). *Autostat*, included in AutoMod and *Simrunner*, included in ProModel both use evolutionary algorithms. A variety of companies in the simulation industry, as well as general management service and consulting firms like Rockwell Software, Dassault Systemes, Flextronics, Halliburton, HP, Planview and CACI, employ Opttek Systems, Inc. software OptQuest, which uses tabu search and scatter search.

About the Authors:

Kenneth Sorensen (http://antor.ua.ac.be/kenneth.sorensen) is a Research Professor of the Faculty of Applied Economics at Antwerp, and is founder and member of the coordinating team of EU/ME – the metaheuristics community, the largests online forum for metaheuristics researchers.

Fred Glover (http://spot.colorado.edu/~glover) is Distinguished Professor of the University of Colorado System and Chief Technology Officer of OptTek Systems, Inc.

References

Anonymous. Riders on a swarm. The Economist, 12 August 2010.

J. April, F. Glover, J. Kelly, and M. Laguna. Practical introduction to simulation optimization. In S. Chick, T. Sanchez, D. Ferrin, and D. Morrice, editors, *Proceedings of the 2003 Winter Simulation Conference*, 2003.

H.G. Beyer and H.P. Schwefel. Evolution strategies–A comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

L. Bianchi, M. Dorigo, L.M. Gambardella, and W.J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.

E. Burke, P. De Causmaecker, S. Petrovic, and G.V. Berghe. Variable neighborhood search for nurse rostering problems. In M.G.C. Resende, J. Pinho de Sousa, and A.Viana, editors, *Metaheuristics: computer decision-making*, pages 153–172. Kluwer Academic Publishers, 2004.

R. Chelouah and P. Siarry. Tabu search applied to global optimization. *European Journal of Operational Research*, 123 (2):256–270, 2000.

C. Commander, P. Festa, C.A.S. Oliveira, P.M. Pardalos, M.G.C. Resende, and M. Tsitselis. Grasp with path-relinking for the cooperative communication problem on ad hoc networks. In D.A. Grundel, R.A. Murphey, P.M. Pardalos, and O.A. Prokopyev, editors, *Cooperative Networks: Control and Optimization*, pages 187–207. Edward Elgar Publishing, 2008.

P. Czyżak et al. Pareto simulated annealing-a metaheuristic technique for multiple-

objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7 (1):34–47, 1998.
E. Danna. Integrating local search techniques into mixed integer programming. *4OR: A Quarterly Journal of Operations Research*, 2 (4):321–324, 2004.
E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005.
M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, man, and cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.
M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1 (4):28–39, 2006.
C. Duin and S. Voß. The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. *Networks*, 34(3):181–191, 1999.
I. Dumitrescu and T. Stützle. Usage of exact algorithms to enhance stochastic local search algorithms. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*. Springer, 2009.
T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2): 109–133, 1995.
C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11(2): 198–204, 1999.
D.B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. Wiley-IEEE Press, 2006.
C.M. Fonseca and P.J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the fifth international conference on genetic algorithms*, pages 416–423. Citeseer, 1993.
C. Friden, A. Hertz, and D. de Werra. TABARIS: An exact algorithms based on tabu search for finding a maximum independent set in a graph. Working paper, Swiss Federal Institute of Technology, Lausanne, 1989.
M.C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3): 192–215, 2002.
M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.
F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.
F. Glover. Tabu search-part I. *ORSA journal on Computing*, 1(3):190–206, 1989.
F. Glover. Tabu search-part II. *ORSA Journal on computing*, 2(1):4–32, 1990.
F. Glover. Tabu search nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49:231–255, 1994.
F. Glover. Tabu search and adaptive memory programming: Advances, applications and challenges. In Barr, Helgason, and Kennington, editors, *Interfaces in Computer Science and Operations Research*. Kluwer Academic Publishers, 1996.
F. Glover. Adaptive memory projection methods for integer programming. In C. Rego and B. Alidaee, editors, *Metaheuristic optimization via memory and evolution*, pages 425–440. Kluwer Academic Publishers, 2005.
F. Glover and D. Klingman. Layering strategies for creating exploitable structure in linear and integer programs. *Mathematical Programming*, 40(1):165–181, 1988.
F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, Boston, 1997.

F. Glover, J. Kelly, and M. Laguna. New advances wedding simulation and optimization. In D. Kelton, editor, *Proceedings of the 1999 Winter Simulation Conference*, 1999.

F. Glover, M. Laguna, and R. Martì. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–684, 2000.

F. Glover, M. Laguna, and R. Marti. Scatter search and path relinking: Advances and applications. *Handbook of metaheuristics*, pages 1–35, 2003.

D.E. Goldberg et al. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley Reading Menlo Park, 1989.

M.P. Hansen. Tabu search for multiobjective optimization: MOTS. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM'97), Cape Town, South Africa*, pages 574–586. Citeseer, 1997.

M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende. Global optimization by continuous GRASP. *Optimization Letters*, 1(2):201–212, 2007.

J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

DF Jones, SK Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European journal of operational research*, 137 (1):1–9, 2002.

J. Kennedy, R.C. Eberhart, et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948, 1995.

S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671, 1983.

J.R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT press, 1992.

L. Liberti and M. Drazič. Variable neighbourhood search for the global optimization of constrained NLPs. *Proceedings of GO*, pages 1–5, 2005.

H. Lourenco, O. Martin, and T. St
"utzle. Iterated local search. *Handbook of metaheuristics*, pages 320–353, 2003.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6): 1087, 1953.

Z. Michalewicz and D.B. Fogel. *How to solve it: modern heuristics*. Springer-Verlag New York Inc, 2004.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11): 1097–1100, 1997.

P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report*, 826:1989, 1989.

M.C.V. Nascimento, M.G.C. Resende, and F.M.B. Toledo. Grasp heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, 200: 747–754, 2010.

J. Pearl. *Heuristics–intelligent search strategies for computer problem solving*. Addison-Wesley Publishing Co., Reading, MA, 1984.

J. Puchinger, G.R. Raidl, and S. Pirkwieser. Metaboosting: Enhancing integer programming techniques by metaheuristics. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*. Springer, 2009.

G.R. Raidl and J. Puchinger. Combining (integer) linear programming techniques and

metaheuristics for combinatorial optimization. In In C. Blum, M.J. Blesa Aguilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer, 2008.

C. Rego. RAMP: A new metaheuristic framework for combinatorial optimization. In C. Rego and B. Alidaee, editors, *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, pages 441–460. Kluwer Academic Publishers, 2005.

M.G.C. Resende, R. Martí, M. Gallego, and A. Duarte. Grasp and path relinking for the max-min diversity problem. *Computers and Operations Research*, 37:498–508, 2010.

C.C. Ribeiro and M.G.C. Resende. Path-relinking intensification methods for stochastic local search algorithms. Research technical report, AT&T Labs, 2010.

J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985.

N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4): 341–359, 1997.

K. Sörensen and F. Glover. Metaheuristics. In S.I. Gass and M. Fu, editors, *Encyclopedia of Operations Research and Management Science*, New York, To appear. Springer.

K. Sörensen, M. Sevaux, and P. Schittekat. *"Multiple neighbourhood search" in commercial VRP packages: evolving towards self-adaptive methods*, volume 136 of *Lecture Notes in Economics and Mathematical Systems*, chapter Adaptive, self-adaptive and multi-level metaheuristics, pages 239–253. Springer, London, 2008.

P. Van Hentenryck and L. Michel. *Constraint-based local search*. The MIT Press, 2009.

C. Voudouris and E. Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113 (2):469–499, 1999.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3 (4):257, 1999.

E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for multiobjective optimization*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–38, Berlin, 2004. Springer.