A Unified Framework for Metaheuristics

Jürgen Branke, Michael Stein, and Hartmut Schmeck

Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{branke,stein,schmeck}@aifb.uni-karlsruhe.de

1 The Framework

Over the past decades, a multitude of new search heuristics, often called "metaheuristics" have been proposed, many of them inspired by principles observed in nature. What distinguishes them from random search is primarily that they maintain some sort of memory of the information gathered during the search so far, and that they use this information to select the location where the search space should be tested next. Based on this observation, we propose a general unified framework which is depicted in Fig. 1: A memory is used to construct one or more new solutions which are then evaluated and used to update the memory, after which the cycle repeats.

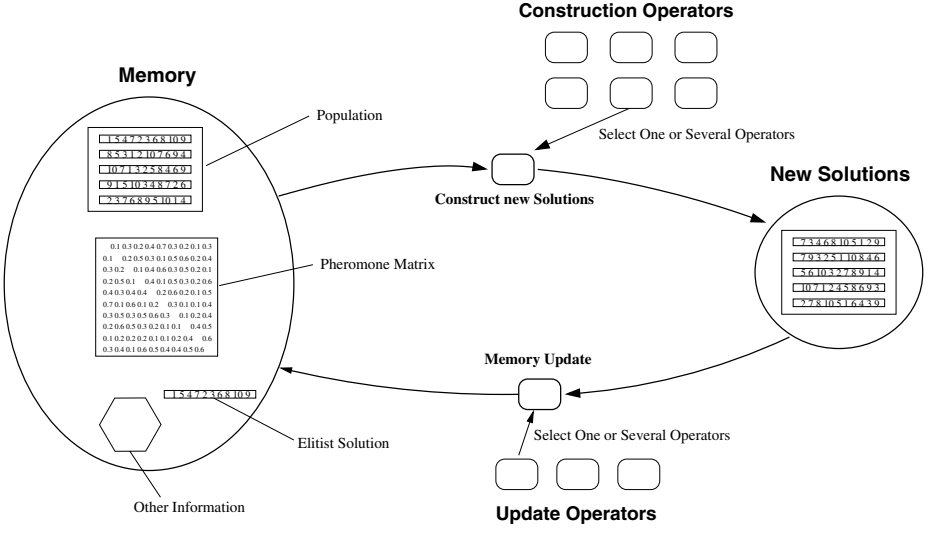

Fig. 1. Unified framework for iterative search algorithms

In the following, we will show in more detail how this general framework can be used to describe some of the aforementioned metaheuristics.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1568–1569, 2003.

2 All Metaheuristics are Just One

EAs store information about the previous search in form of a set of solutions (population). New solutions are constructed by selecting two solutions (parents), combining them in some way (crossover) and performing some local modifications (mutation). Then, the memory is updated by inserting the new solutions into the population. Although there exists a variety of EA variants, they all fit into this general framework. For example, evolution strategies with self-adaptive mutation can be specified by extending the memory to also maintain information about the strategy parameters. Steady state genetic algorithms update the population after every newly generated solution, while genetic algorithms with generational reproduction generate a whole new population of individuals before updating the memory.

Simulated annealing only maintains a single solution in the memory. In addition to that, it keeps track of time by a temperature variable. New solutions are created by local modifications more or less equivalent to the mutation operator in EAs. The new solution replaces the current memory solution depending on the quality difference and the temperature.

Tabu search, just as simulated annealing, also maintains a single solution from which new solutions are constructed, but additionally maintains a tabu list of recently visited solutions that should not be re-visited. New solutions are created by local modifications, but taking into account the tabu list.

Ant colony optimization has a completely different way to store information about the search conducted so far. Instead of storing complete solutions, information about which partial decisions have been successful when constructing solutions from scratch is accumulated in a (pheromone) matrix. This matrix, is used to construct new solutions, giving a higher probability to decisions which have yielded successful solutions in the past. Usually, several new solutions are generated that way, and then the best solution found is used to update the matrix such that the decisions made to construct that particular solution become more preferable. An elitist ant (best solution found so far) can be modeled by an additional (complete) solution stored in the memory.

3 Benefits

Given a description of the different metaheuristics in this general form has many benefits. First of all, it creates a common language, which allows researchers from different fields to understand each other's approaches easily. Second, it moves the focus from the complete algorithms to the components. And third, it provides the interfaces for the different components to work together. It suggests a natural way for hybridization, basically turning the variety of metaheuristics into one large toolbox from which an algorithm designer can choose those parts that seem most appropriate for the application at hand.