Adaptive Memory Programming: A Unified View of Metaheuristics

Éric D. Taillard †, Luca M. Gambardella †,
Michel Gendreau ‡§ and Jean-Yves Potvin ‡§

†Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA),
Corso Elvezia 36,
CH-6900 Lugano, Switzerland

‡Centre de recherche sur les transports
Université de Montréal,
C.P. 6128, succursale Centre-ville,
Montréal, Québec, Canada H3C 3J7

§Département d'informatique et de recherche opérationnelle,
Université de Montréal,
C.P. 6128, succursale Centre-ville,
Montréal, Québec, Canada H3C 3J7

Abstract

The paper analyses recent developments of a number of memory-based metaheuristics such as taboo search, scatter search, genetic algorithms and ant colonies. It shows that the implementations of these general solving methods are more and more similar. So, a unified presentation is proposed under the name of *Adaptive Memory Programming* (AMP). A number of methods recently developed for the quadratic assignment, vehicle routing and graph colouring problems are reviewed and presented under the adaptive memory programming point of view. AMP presents a number of interesting aspects such as a high parallelization potential and the ability of dealing with real and dynamic applications.

1 Introduction

Generic heuristic methods, also called metaheuristics or general local search methods are growing at a very fast, exploding rate: the most important conferences in operational research have one or many sessions entirely devoted to metaheuristics and there are also journals entirely devoted to them. Among the most successful techniques, we can quote genetic algorithms (GA), simulated annealing (SA), taboo search (TS), scatter search (SS) and ant systems (AS). The success of these methods depends on many factors, like their ease of implementation, their ability to consider specific constraints that arise in practical applications and the high quality of the solutions they produce.

From a theoretical point of view, however, the use of these methods has not yet been justified. For example, a few convergence theorems for SA or TS exist [34, 17] but they are useless in practice. These theorems simply state that the search has a very high probability of ending with an optimal solution if a disproportionate computing time is allowed (larger, in fact, than the time needed for a complete enumeration of the solution space). But these metaheuristics are really competitive in practice. In this race for competitiveness, the most efficient methods hybridize two or more metaheuristics. The result is that similar problem-solving methods are known under very different names like genetic hybrids, probabilistic taboo search, adaptive multi-start or MAX-MIN ant system. All these methods have three features in common: first, they memorize solutions or characteristics of solutions generated during the search process; second, they include a procedure that creates an initial solution with the information stored in memory; third, they apply a local search method, like a greedy improvement method, an elementary taboo search or simulated annealing, to improve the initial solution.

Observing the similarities between these methods, a unified presentation is useful and necessary, as well as a more synthetic and general name. The proposed terminology for grouping these metaheuristics under the same roof is *Adaptive Memory Programming* (AMP). This term was already proposed by Glover in connection with taboo search [31, 32]. It is true that taboo search has always been presented as an open technique that can include components from various fields, in particular artificial intelligence. The inclusion of a memory and a learning process within the taboo search framework was thus proposed very early: even the name of the method comes from its short term memory component! However, the evolution of a particular taboo search implementation can very well produce a method without any taboo list (for example, if parameter tuning indicates that a taboo list size of 0 is best). So, the name of the method becomes hard to justify, but for historical reasons.

The same phenomenon is observed with other metaheuristics. For example, most efficient implementations of genetic algorithms do not encode solutions of the problem as binary vectors. Furthermore, the crossover and mutation operators have often little or nothing to do with the standard operators for which some theoretical results have been obtained.

The aim of this paper is thus to propose the name *Adaptive Memory Programming* which is more general and seems to better reflect the current reality. To justify the use of this term, we first review in Section 2 various metaheuristics that can be viewed as AMP methods, at least for their most recent and efficient implementations. Section 3 then presents our views on AMP and shows that many applications for different combinatorial optimization problems perfectly fit this framework.

2 Metaheuristics with memory

The term *memory* was used explicitly for taboo search only, but a number of other metaheuristics use mechanisms that can be considered as memories. For genetic algorithms and scatter search, the memory is constituted by a population of solutions; for ant systems, the pheromone trail is also a form of memory. So, let us first review a number of memory-based metaheuristics.

2.1 Genetic algorithms

Basically, genetic algorithms simulate the evolutionary process of species that sexually reproduce. This evolutionary process can be described as follows. During sexual reproduction, a new individual, different from its two parents, is created through the action of two fundamental mechanisms. The first one is crossover which combines half of the genetic patrimony of each parent to produce the genetic patrimony of the new individual. The second one is mutation by which a spontaneous modification of the genetic patrimony occurs. The new individual so created (child or offspring) will therefore be different from its parents, but will also share a number of their characteristics. If the child inherits good characteristics from its parents, its survival probability will be higher, as compared to individuals that inherit bad characteristics. It will thus have a higher probability to reproduce and disseminate good characteristics to its offspring.

The analogy between this evolutionary process and the GA meta-heuristic proposed by Holland [37] can be established in the following way. An individual is associated with a feasible solution of the problem at hand. The solution is encoded as a binary vector. The crossover operator then exchanges substrings taken from both parents to produce an offspring. Mutation is a secondary operator that flips bit values on the offspring vector, with a small probability at each position. The quality of the newly created vector is finally evaluated according to the objective value of the solution it encodes.

A simple genetic algorithm can be sketched as follows:

1. Generate a population of vectors (individuals).
2. While a stopping criterion is not met do:
   (a) select, with replacement, a pool of parent vectors from the population;

(b) randomly match the parent vectors and apply crossover to produce offspring vectors;
(c) apply mutation to each offspring;
(d) evaluate offspring;
(e) insert offspring in the population;
(f) eventually, remove individuals from the population with a culling operator.

Typically, this procedure stops after a fixed number of iterations (generations) or when the population does not improve anymore. It is worth noting that the selection of parents is probabilistically biased towards the best individuals. Hence, the latter are more likely to disseminate their good characteristics to offspring (like the corresponding natural evolution process). Recent implementations of genetic algorithms do not always use a binary vector for coding a solution but they use a representation that is better adapted to the application domain. For example, permutation of integer values are widely used for the travelling salesman problem (TSP) or the quadratic assignment problem (QAP). This influences directly the cross-over and mutation operators. While the coding scheme was the key point of earlier genetic implementations, the most important elements of recent implementation are the definition of cross-over and mutation operators that are able to extract pertinent (and *a priori* unknown) informations on the structure of good solutions.

2.2 Scatter search

Scatter search has been proposed in 1977 by Glover to solve integer programming problems [27]. The method is rather similar to genetic algorithms, at least if one considers the most recent ways of implementing the genetic metaphor. The scatter search method is based on a population of solutions (integer vectors) that evolves through selection, linear combination, integer vector transformation and culling to produce a new population of solutions. With respect to a standard genetic algorithm, scatter search has the following particularities:

- binary vectors are replaced by integer vectors;
- more than two parents can be matched to produce a new vector;
- crossover is replaced by a convex or non-convex linear combination of two or more vectors;
- mutation is replaced by a procedure that repairs or projects the newly created vector in the space of feasible solutions.

These particularities can also be seen as generalizations of the basic GA procedure. In fact, such generalizations have later been proposed and exploited by various authors (see for example [39, 42]). Let us quote:

- departure from the binary vector scheme;
- use of a various number of parents to produce an offspring;
- development of specialized crossover operators;
- use of local search methods to improve solutions obtained through crossover;
- use of repair operators;
- the solutions that are kept in the population from one iteration to the next are chosen with the help of an elaborated clustering method rather than a simple culling operator.

2.3 Taboo search

Among the metaheuristics with memory presented in this section, taboo search is the only one that has been explicitly developed with a memory, or more exactly, with a set of memories. The term *taboo search* has been proposed for the first time by Glover [28] in 1986. The basic idea of this method is to locally and repeatedly modify a solution while memorizing these modifications to avoid visiting the same solutions a second time or in a cyclic manner. To this end, modifications or characteristics of modifications are stored in lists that forbid their use for a certain number of iterations, thus leading to the names of taboo list and taboo search.

In a sense, this method imitates a human being looking for a good solution of a combinatorial optimization problem. First, an initial solution is looked for, even a bad one. Then, this solution is iteratively improved through local modifications. The latter modifications do not necessarily improve the solution at each iteration but the aim is to direct the search toward a good subset of solutions.

Later, Glover proposed a number of strategies to guide the search and make it more efficient [29, 30]; at the same time, he emphasized that taboo search was open to any strategy well adapted to the problem on which it is applied. In this sense, the metaheuristics with memory presented in this section are implicitly contained in taboo search. We think, however, that the term *taboo search* does not appropriately characterize their most recent implementations.

The most important idea of the first publication on taboo search [28] is the use of a short term memory, known as the taboo list. Numerous applications, even recent ones, still only exploits this mechanism. The use of long term memories spread well after the "complete" description of taboo search by Glover [29, 30]. These are found under different forms, from statistics on modifications made to the current solution [45, 47, 48, 49] to complete solutions of high quality visited during the search. Their exploitation for guiding the search can also vary from a perturbation of the objective function value to the construction of a brand new starting solution.

A basic taboo search can be sketched as follows:

1. Generate an initial solution s_0; initialize the memories; $k \leftarrow 0$; $s^* \leftarrow s_0$.
2. While a stopping criterion is not met do:
   (a) choose s_{k+1}, a neighbour solution of s_k, using data stored in the memories;
   (b) if s_{k+1} is better than s^* then $s^* \leftarrow s_{k+1}$;
   (c) $k \leftarrow k + 1$;
   (d) update the memories.

Typically, the stopping criterion corresponds to a fixed number of iterations or a number of consecutive iterations without improving the best known solution s^*. Each step of the above algorithm can be very simple or very complicated. For example the choice of s_{k+1} in step 2(a) may imply that all neighbours of s_k are examined and the best non taboo solution is chosen; it may also imply the construction, in a complex way, of a solution that is not so close of s_k, through another embedded metaheuristic, as in Rochat and Taillard [44].

2.4 Ant systems

The idea of imitating the behaviour of ants to find solutions to combinatorial optimization problems was initiated by Colorni, Dorigo and Maniezzo in 1991 [8, 9, 10]. The metaphor comes from the way ants search for food and find their way back to the nest. Initially, ants explore the area surrounding their nest in a random manner. As soon as an ant finds a source of food, it evaluates the interest of the source (quantity and quality) and carries some of the food to the nest. During the return trip, the ant leaves on the ground a chemical pheromone trail whose quantity depends on the quality of the source. The rôle of this pheromone trail is to guide other ants toward the source. After a while, the path to a good source of food will be indicated by a large pheromone trail, as the trail grows with the number of ants that reach the source. Since sources that are close to the nest are visited more frequently than those that are far away, pheromone trails leading to the nearest sources grow faster. The final result of this process is that ants are able to optimize their work.

The transposition of this food searching behaviour into an algorithmic framework for solving combinatorial optimization problems is obtained through an analogy between:

- the search area of the real ants and the set of feasible solutions to the combinatorial problem;
- the amount of food associated with a source and the objective function;

- the pheromone trail and an adaptive memory.

A detailed description of how such analogies can be applied in the case of the TSP may be found in Dorigo and Gambardella [15]. A standard ant system is schematically described as follows:

1. Initialize the pheromone trail.
2. While a stopping criterion is not met do:
   (a) for each ant, construct a new solution using the current pheromone trail and an evaluation of the partial solution being constructed;
   (b) update the pheromone trail.

The most important component of an ant system is the management of pheromone trails. In a standard ant system, pheromone trails are used in conjunction with the objective function to guide the construction of new solutions. Once a solution has been produced, a standard ant system updates the pheromone trails as follows: first all trails are weakened to simulate the evaporation of pheromone; then, pheromone trails that correspond to components that were used to construct the resulting solution are reinforced, taking into consideration the quality of this solution.

Based on the previous general scheme different ant systems implementations have been proposed where pheromone updating is performed in different ways: in [9] the pheromone is updated by all ants involved in the optimization process while in [21] only the best ant is allowed to update pheromone informations. MAX-MIN Ant System [46] introduces an updating phase constrained between threshold values. Different ways of modifying pheromone values generate different type of search mechanism: for example, in [21] the result of the updating phase is to drive the search around the neighborhood of the best solutions found so far. All the mentioned ant systems have been applied to different combinatorial optimization problems like symmetric and asymmetric TSPs [9, 15, 21] and quadratic assignment problems [15] with comparable or even better performances than other natural inspired systems. More recently it has been shown that ant systems based algorithms are very powerful in combination with local search procedures. In these situations pheromone information is used to produce solutions (diversification phase) that are optimized by a local search (intensification phase). Optimized solutions are then used to update pheromone information and new solutions are successively generated by the ants. In particular let us mention successful hybrid ant system implementations applied to symmetric and asymmetric TSPs [14, 21, 46], sequential ordering problems [22], quadratic assignment problems [23, 46, 55] and vehicle routing problems [6].

3 Adaptive memory programming

Although the methods presented in the previous section may look very different at first glance, an analysis of a number of implementations, and especially the most efficient and recent ones, shows that they are sometimes extremely close in their working principles. In other terms, one observes a unification of different metaheuristics in such a way that it now becomes difficult to make clear distinctions. Indeed, the population of a genetic algorithm or scatter search, or the pheromone trail of an ant system, can be considered as a special kind of memory, like the memories found in taboo search. Conversely, a taboo search that periodically restarts from new solutions constructed from a memory can be assimilated to a genetic algorithm or an ant system. Hence, a large number of efficient methods for solving combinatorial optimization problems now share common characteristics. More precisely:

1. A set of solutions or a special data structure that aggregates the particularities of the solutions produced by the search is memorized;
2. a provisory solution is constructed using the data in memory;
3. the provisory solution is improved using a greedy algorithm or a more sophisticated metaheuristic;
4. the new solution is included in the memory or is used to update the data structure that memorizes the search history.

These recent methods are thus characterized by the exploitation of a memory to construct a new solution, an improvement procedure to find an even better solution and, finally, a memory update procedure based on pieces of knowledge brought by the improved solution. It thus looks appropriate and natural to speak of *Adaptive Memory Programming* for this type of metaheuristics. From an algorithmic point of view, an AMP may be sketched as follows:

1. Initialize the memory.
2. While a stopping criterion is not met do:
   (a) Generate a new provisory solution s using data stored in the memory.
   (b) Improve s with a local search; let s' be the improved solution.
   (c) Update the memory using the pieces of knowledge brought by s'.

There are other metaheuristics that share a number of characteristics with this AMP scheme. Let us mention memetic algorithms [38], the greedy randomized adaptive search (GRASP) of Feo and Resende [18], the adaptive multi start (AMS) of Boese, Kahng and Muddu [5] and the memory adaptive reasoning of Paterson, Rolland and Pirkul [41].

In this section, we review a number of recent and highly efficient applications, that can be naturally presented as AMP methods, for the quadratic assignment (QAP), vehicle routing (VRP) and graph colouring problems. We briefly sketch all these methods, emphasizing how the four main characteristics of AMP have been used, namely: the memory, the construction of a provisory solution, the improvement procedure and the memory update procedure.

3.1 Quadratic assignment problem

In the quadratic assignment problem, a number of units must be assigned to the same number of locations. Given the distance between each pair of locations and a flow between each pair of units, an assignment of units to locations (i.e., a permutation) is searched for that minimizes the sum of the *distance* $\times$ *flow* products over the assigned unit-location pairs. Therefore, a QAP solution can be viewed as a permutation. Many different AMP methods have been designed for the QAP. They are among the most efficient ones, in particular for real applications, where the flow and distance coefficients exhibit a very high variance. These methods are

- hybrid taboo search-genetic algorithm of Fleurent and Ferland [19];
- scatter search of Cung et al. [13];
- hybrid ant system of Gambardella, Taillard and Dorigo [23];
- fast ant system of Taillard and Gambardella [55].

They are reviewed in the following.

3.1.1 Genetic hybrid of Fleurent and Ferland

(a) memory

The memory is a population made of a number of best solutions found by the search.

(b) provisory solution

The provisory solution is built using a cross-over operator designed by Tate and Smith [57]. First two solutions (permutations) are selected from the population. The elements of the provisory permutation are then chosen in three phases:

1. elements common to both solutions are copied in the new permutation at the same position;

2. if possible, the other elements are randomly chosen from the selected solutions and copied in the new permutation at the same position;

3. the unfilled positions are randomly chosen to complete the permutation.

(c) improvement procedure

The provisory solution is improved with a short taboo search due to Taillard [47].

(d) memory update

The new solution is inserted in the memory and the worst solution is removed from it.

This application illustrates a typical evolution of genetic algorithms: the binary coding scheme is replaced by a natural representation of the solutions, the cross-over operator is replaced by a relatively elaborated procedure and the mutation operator is replaced by an elaborated local search. In [55], it is shown that the use of a simpler and faster local search for improving the provisory solutions may be more efficient than the use of taboo search.

3.1.2 Scatter search of Cung et al.

(a) memory

The memory stores a number of best solutions.

(b) provisory solution

A number of solutions are selected in a probabilistic way from the memory. The choice of the solutions to be combined is done according to a policy that ensures a certain level of diversity in the memory. The selected solutions are manipulated as vectors of integers and a linear combination of these vectors is computed: the provisory solution is then the permutation which is the closest from the previously computed vector. Cung et al. have shown that this closest permutation can be found by solving a linear assignment problem. Instead of solving this assignment problem exactly, Cung et al. use a faster heuristic method.

(c) improvement procedure

The provisory solution is improved with a basic taboo search. The number of iterations performed by this taboo is modulated from one call the the other.

(d) memory update

The new solution is inserted in the memory and the worst solution is removed from it.

3.1.3 Fast ant system of Taillard and Gambardella

(a) memory

The memory is a square matrix that records frequency statistics on the position of each element in the permutations (solutions) produced by the system.

(b) provisory solution

A new permutation is sequentially constructed in a probabilistic way. The probability of inserting a given element at a given position is proportional to its frequency value stored in memory.

(c) improvement procedure

The solution is improved with a very fast greedy local search that does not necessarily reach a local optimum.

(d) memory update

The frequencies are recomputed at each iteration, considering both the solution just produced and the best solution ever produced by the search. The memory is cancelled and re-initialized each time the best known solution is improved in order to intensify the search. A diversification mechanism is also implemented in case the provisory solution generated is the same than the best solution produced so far.

The MAX-MIN ant system of [46] is similar to this method but it uses slightly different statistics as memory. The improvement procedure performs a complete local search reaching the first local optimum and the diversification/intensification mechanism is different: the values contained in the statistics-matrix are lower and upper bounded by two parameters, hence the name of the method.

3.1.4 Hybrid ant system of Gambardella, Taillard and Dorigo

(a) memory

The memory is divided in two parts: the first part contains a matrix that records frequency statistics on solutions produced thus far by the search; the second part contains a small population of solutions.

(b) provisory solution

As opposed to basic ant systems where a new solution is constructed from scratch, this method modifies the solutions found in the population in the spirit of a local search, but using the frequency matrix to choose the modifications to perform.

(c) improvement procedure

Each solution in the population is improved with a fast greedy local search that does not necessarily reach a local optimum.

(d) memory update

The memories are updated as follows. First, each solution in the current population is replaced by the solution obtained after the improvement procedure, unless the latter is worse than the original solution. Second, the frequency matrix is updated: it records a moving average of the frequency (weighted by solution quality) at which a given unit is found at a given position in the best solution of the current population.

A number of these AMPs are compared in [51, 46, 23, 55] and the main conclusions are the following: All AMPs are performing well for instances that exhibit a high variance in the entries of the data matrices. Such instances are typically arising in real applications and often present a large "flow dominance" criterion [61] or a large "ruggedness" value [1]. To obtain the best solutions in the shortest time, the fast ant system of [55] is one of the best method. If solutions of higher quality are desired, it seems that keeping a number of solutions in memory can be recommended. Implementations of simulated annealing, genetic algorithms or variable neighborhood search that do not exploit either memories or local search have been shown to be less efficient than AMPs [55].

3.2 Vehicle routing problem

The aim of the vehicle routing problem (VRP) is to determine optimal collection or delivery tours for transportation vehicles through a set of customer locations, subject to various constraints. Typically, "optimal" refers to a solution of minimum length or a solution that uses the minimum number of vehicles (when this number is not fixed).

Many different vehicle routing problems have been successfully addressed in the past with either taboo search or genetic algorithms. For example, taboo search has been applied to the capacitated vehicle routing problem (CVRP), where a single type of constraint states that the total load on a tour cannot exceed vehicle capacity [24, 40, 48, 62], the VRP with time windows (VRPTW), where customers must be serviced within specified time intervals (in addition to the capacity constraint) [7, 43], and the VRP with backhauls and time windows (VRPBTW) where customers can be either pick-up or delivery points [16]. Applications of genetic algorithms for variants of the VRPTW may also be found in [4, 42, 59, 60, 58]. These implementations, mostly realized before 1995, can hardly be qualified as AMP methods, with the exception of [42] where a genetic algorithm is hybridized with a local search heuristic, in the same spirit as the genetic hybrid of Fleurent and Ferland for the QAP [19]. In 1994, however, Taillard [50] developed a true AMP method, where a taboo search heuristic is used to improve provisory solutions constructed from an adaptive memory. The method was then tested on the VRPTW and good results were published in Rochat and Taillard [44]. Later, Taillard et al. [54] designed another AMP method for a variant of the VRPTW, known as the VRP with soft time windows (VRPSTW), where customer locations can be visited outside of their associated time window (at the expense of a penalty in the objective value). A parallel version of the latter algorithm was implemented on a network of workstations [2] and adapted for a real-time version of the problem, where customer requests occur in an on-going fashion [25, 26]. These articles report numerical results obtained with various methods: Two AMPs that uses either a simple descent method or a taboo search as improvement procedure, and four different implementations of insertion and rebuild methods that are not using an adaptive memory. It is shown that memory-based heuristics are clearly superior to the other methods.

Adaptations of the original AMP method of Taillard have also been applied to other variants of the VRP, namely: the VRP with multiple uses of vehicles (VRPM), where each vehicle is allowed to perform many tours during a work day [56]; the min-max CVRP and min-max Multiple TSP (MTSP) where the goal is to minimize the length of the longest tour [33]; the VRP with heterogeneous, or non identical, vehicles (VRPHE) [52]; and two special cases of the latter problem, the Vehicle Fleet Mix (VFM) and Vehicle Fleet Mix with Variable Routing Cost (VFMVRC) problems, where a fixed cost is incurred for using a given vehicle type, as well as a different cost by unit of length traveled by each vehicle type.

For all these variants, efficient methods were designed using the AMP framework. The basic scheme is the following:

(a) memory

The memory contains a set T of tours belonging to solutions produced by the search.

(b) provisory solution

A partial solution s is constructed by selecting a number of tours contained in the memory, in a probabilistic manner (i.e. tours associated with better solutions have a higher probability of being selected). More precisely, the provisory solution is built as follows:

1. Make a copy T' of T: $T' \leftarrow T$.
2. Set $s = \emptyset$
3. While $T' \neq \emptyset$ repeat:
   (a) Randomly choose a tour $t \in T'$.
   (b) Set $s \leftarrow s \cup \{ t \}$.
   (c) $\forall t' \in T'$ such that $t' \cap t \neq \emptyset$, set $T' \leftarrow T' \setminus \{ t' \}$.

In this procedure, Step 3(c) looks for tours that share one or more customers with the current selected tour t. These tours are removed and are not considered in the remainder of the selection process. Hence, the provisory solution obtained at the end is necessarily made of mutually exclusive tours. However, it is possible that a number of customers remains unserviced. These customers are typically introduced in the provisory solution through some least-cost insertion method to produce a complete solution. The insertion may also be performed by the improvement procedure if the last is able to complete partial solutions.

(c) improvement procedure

The provisory solution s is improved using taboo search.

(d) memory update

The tours of the improved solution are inserted in the memory T and the tours of the worst solution are removed from T.

For a few variants of the VRP, the AMP solves a simpler auxiliary problem. A solution of the original problem is then found by (implicitly) enumerating all solutions that can be constructed with tours contained in the memory, so that all constraints of the original problem are satisfied.

3.3 Graph colouring

Graph colouring is aimed at partitioning the set of vertices into a given number of subsets (colours) such that the number of edges that connect two vertices belonging to the same subset is minimized. In [63], the idea is to use a memory and a procedure for producing provisory solutions similar to that presented just above for the VRP. Indeed, instead of storing tours in T, i.e. subsets of customers, it is possible to store subsets of vertices having received the same colour in a solution. Then, the procedure presented above for the VRP can be applied to built a partial colouring that is improved with a taboo search [36]. So, the method goes along the lines of the AMP of Rochat and Taillard [44].

Costa, Hertz and Dubuis [12] and Fleurent and Ferland [20] have proposed particularly effective AMP methods for graph colouring that are designed in the spirit of a genetic algorithm embedding a local search. They both use the same memory and memory update mechanisms, but differ in the way they construct and improve the provisory solution. We now briefly sketch these two methods.

(a) memory

The memory stores a number of solutions.

(b) provisory solution

Two solutions are selected from the memory and combined to obtain the provisory solution. Then, the colour of each vertex of the provisory solution is copied from the selected solutions. This copy is performed according to a policy that slightly differs in both methods.

(c) improvement procedure

Costa, Hertz and Dubuis use a simple greedy local search for improving the provisory solution while Fleurent and Ferland use a relatively elaborated taboo search [36].

(d) memory update

Both methods insert the new solution in the memory and remove the worst one from it.

3.4 Synthesis

Even if the list of successful applications of AMP methods to solve combinatorial optimization problems, as found in subsections 3.1 to 3.3, is far from exhaustive, it should clearly demonstrate that all these methods use the the same working principles. We must however emphasize that they differ a lot in their implementation details; these details should not be overlooked, as they are responsible for the success or failure of a given AMP method. In fact, there is almost an infinite number of ways of implementing a simple greedy local search, a taboo search, a genetic algorithm, etc.

While promoters of different metaheuristics with memory developed increasingly sophisticated mechanisms to maintain the competitiveness of their approach, their efforts have often converged to the same general principles: use of a memory and procedures for constructing a new solution and for improving it. A notable exception is simulated annealing, as it does not use any memory. The framework of this technique being more rigid, as compared to other metaheuristics, it may explain why it did not evolve in the same way.

Why did metaheuristics with memory converge to the same basic approach? One answer could be that the a natural way of alleviating the weaknesses of a given technique is to borrow components from another technique (which does not exhibit the same weaknesses). Schematically, this evolution could be described as follows for the different approaches:

genetic algorithms

- Coding a solution with a binary vector is not natural and can significantly impact the performance. Hence, binary coding was replaced by a more natural representation of solutions.
- The classical crossover operators do not correspond to logical operations on solutions. Furthermore, the use of other representations than binary vectors naturally led to the design of specialized operators, well adapted to the solution representation and capable of generating new feasible solutions.
- Genetic algorithms can easily identify different solution subspaces with good characteristics, but they lack the "killer instinct" that would allow

them to intensify the search in these area. To alleviate this weakness, the mutation operator was replaced by repair procedures and local search.

scatter search

- This technique initially contained all the ingredients of an AMP. This technique was not exploited for a long time but has recently emerged when its similarity with other modern techniques was noticed. Modern forms of scatter search incorporate the learning capabilities provided by taboo search.

ant system

- Like genetic algorithms, early implementations of ant system converged too slowly toward high quality solutions. Therefore intensification mechanisms were gradually introduced. For example, only the best ant is allowed to update pheromone informations in [21]. The most recent implementations incorporate local search mechanisms to improve the solutions produced by the ants.

taboo search

- Taboo search also contained all elements of the AMP framework, at least in its complete version in [29, 30]. However, a number of implementations, even recent ones, use only the basic ingredients found in Glover's 1986 paper [28]. The main advantage of the basic version is its aggressiveness: the search converges toward a local optimum and examines the neighbourhood of this local optimum very quickly. However, it can easily get trapped in a subspace containing only solutions of poor quality. To diversify the search and force it to visit solutions with different characteristics, one basic idea was to increase the number of forbidden components when performing local modifications to a solution. So, the discussion quickly turned around the optimum taboo list size, since a short list allows a thorough examination of the neighbourhood of a good solution while a long list facilitates the escape from a local optimum to explore new regions of the search space. In fact, the reactive taboo search proposed by Battiti and Tecchiolli [3] was designed to automatically adapt the taboo list size and avoid the fastidious task of explicitly managing the taboo list.

  The main difficulty with taboo search is thus to efficiently incorporate diversification and intensification mechanisms. The use of a memory that stores good solutions visited during the search and the design of a procedure to create provisory solutions from it is a way to achieve this goal. Indeed, solutions contained in memory during the initial search phase present different characteristics, thus leading to a diversified search. Later, solutions contained in memory are mostly representative of one or a few good regions of the solution space. The result is that the search gradually shifts from diversification to intensification [44].

4 Conclusion and perspective

Our goal was to demonstrate that metaheuristics with memory have evolved toward a unified problem-solving approach that encompasses each one of them. Our point of view is that the name *Adaptive Memory Programming* is more general and better characterizes this unified approach than the name of any particular metaheuristic. The simplicity of the general adaptive memory programming scheme could lead to think that one has reached the ultimate goal of meta-heuristics: a unique technique that groups the essence of the good ideas contained in every meta-heuristic. To the contrary, we think on the one hand that the reflections brought by this article should stimulate the creation of new meta-heuristics based on completely different principles. On the other hand, designing an adaptive memory programme requires an intelligence that cannot be embedded in a meta-heuristic.

This intelligence should be brought by the designer that has the knowledge of the problem to solve. First, we can think to the design of the procedure that constructs a provisory solution from solutions generated in the past. Undoubtedly, this construction requires knowledge about the problem. However, the design of the improving procedure could also benefit from future improvements. For example, it is reported in [53] that a simple greedy procedure, using an adequate neighbourhood of reasonable size, could produce solutions that were often of higher quality than previously designed taboo searches, genetic algorithms or variable neighbourhood searches [35] for a class of clustering problems.

Finally, let us point out a few advantages of adaptive memory programming. First, the technique is very well adapted for a parallel implementation on distributed computers [2]. Indeed, the most computationally expensive part of the algorithm is often the local search procedure. Therefore, it is possible to place a local search process on many processors, each of them working on different provisory solutions. Second, AMP is particularly well adapted for solving dynamic problems since its working principle allows the method to adapt itself to slightly modified data [25, 26]. So, it is able to benefit from past computations while other methods like simulated annealing or variable neighbourhood searches “forget” the way they succeeded in finding – sometimes very efficiently – the unique solution they can exhibit. Finally, an adaptive memory programme is often able to exhibit a number of different high quality solutions. This enables the user to choose the practical solution that solves best his problem. Indeed, it is often not possible to include all features of a real application into an optimization model and the possibility of choosing a solution among a number of alternatives is of great importance in practice. For all these reasons, we think that the philosophy of adaptive memory programming will be more and more adopted in the future, especially for real applications.

References

[1] Angel E. and V. Zissimopoulos (1997), "On the landscape ruggedness of the quadratic assignment problem", presented at the *International workshop on combinatorics and computer science LIX–CNRS*, Palaiseau, France.

[2] Badeau P., F. Guertin, M. Gendreau, J.-Y. Potvin and É. D. Taillard (1997), "A parallel tabu search heuristic for the vehicle routing problem with time windows", *Transportation Research 5C*, 109–122.

[3] Battiti R. and G. Tecchiolli (1994), "The reactive tabu search" , *ORSA Journal on Computing 6*, 126–140.

[4] Blanton J. L. and R. L. Wainwright (1993), "Multiple vehicle routing with time and capacity constraints using genetic algorithms", in *Proceedings of the Fifth Int. Conf. on Genetic Algorithms*, S. Forrest ed., Morgan Kaufmann: San Mateo, CA, 452–459.

[5] Boese K. D., A. B. Kahng and S. Muddu (1994), "A new adaptive multi-start technique for combinatorial global optimizations" *Operations Research Letters 16*, 101–113.

[6] Bullnheimer B., R.F. Hartl and C. Strauss (1997), "Applying the Ant System to the Vehicle Routing Problem", Presented at the *2nd Metaheuristic International Conference*, Sophia-Antipolis, France.

[7] Chiang W.-C. and R. A. Russell (1997), "A reactive tabu search metaheuristic for the vehicle routing problem with time windows", *INFORMS Journal on Computing 9*, 417–430.

[8] Colorni A., M. Dorigo and V. Maniezzo (1991), "Positive feedback as a search strategy", Working paper 91-16, Department of Electronics, Politecnico di Milano, Italy.

[9] Colorni A., M. Dorigo and V. Maniezzo (1991), "Distributed optimization by ant colonies", in *Proceedings of the First European Conference on Artificial Life (ECAL-91)*, F. J. Varela and P. Bourgine eds, The MIT Press: Cambridge, MA, 134–142.

[10] Colorni A., M. Dorigo and V. Maniezzo (1992), "An investigation of some properties of an ant algorithm", in *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick eds, North-Holland: Amsterdam, 509–520.

[11] Connolly D. T. (1990), "An improved annealing scheme for the QAP", *European Journal of Operational Research 46*, 93–100.

[12] Costa D., A. Hertz and O. Dubuis (1995), "Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs", *Journal of Heuristics 1*, 105–128.

[13] Cung V.-D., T. Mautor, P. Michelon and A. Tavares (1996), "A scatter search based approach for the quadratic assignment problem", Technical report 96/037, Laboratoire PRISM, Université de Versailles, Saint-Quentin en Yvelines, France.

[14] Dorigo M., L.M. Gambardella (1997), "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation 1*, 53–66.

[15] Dorigo M., V. Maniezzo and A. Colorni (1996), "The ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, an Cybernetics 26B*, 29–41.

[16] Duhamel C., J.-Y. Potvin and J.-M. Rousseau (1997), "A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows", *Transportation Science 31*, 49–59.

[17] Faigle U. and W. Kern (1992), "Some convergence results for probabilistic tabu search", *ORSA Journal on Computing 4*, 32–37.

[18] Feo T. and M. Resende, (1995), "Greedy randomized adaptive search procedures", *Journal of Global Optimization 16*, 109–133.

[19] Fleurent C. and J. A. Ferland (1996), "Genetic hybrids for the quadratic assignment problem", *DIMACS Series in Mathematics and Theoretical Computer Science 16*, 190–206.

[20] Fleurent C. and J. A. Ferland (1996), "Genetic and hybrid algorithms for graph coloring", *Annals of Operations Research 63*, 437–461.

[21] Gambardella L. M. and M. Dorigo (1996), "Solving symmetric and asymmetric TSPs by ant colonies", in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC'96)*, IEEE Press, 622–627.

[22] Gambardella L. M. and M. Dorigo (1997), "HAS-SOP: Ant colony optimization for sequential ordering problems", Technical report IDSIA-11-97, IDSIA, Lugano.

[23] Gambardella L. M. , É. D. Taillard and M. Dorigo, (1997), "Ant colonies for the quadratic assignment problem", Technical report IDSIA-4-97, IDSIA, Lugano.

[24] Gendreau M., A. Hertz and G. Laporte (1994), "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science 40*, 1276–1290.

[25] Gendreau M., P. Badeau, F. Guertin, J.-Y. Potvin and É. D. Taillard (1996), "A solution procedure for real-time routing and dispatching of commercial vehicles", *Proceedings of the 3rd World Congress on Intelligent Transport Systems*, Orlando (Florida).

[26] Gendreau M., F. Guertin, J.-Y. Potvin and É. D. Taillard (1996), "Tabu search for real-time vehicle routing and dispatching", Technical Report CRT-96-47, Centre de recherche sur les transports, Université de Montréal, Canada.

[27] Glover F. (1977), "Heuristics for integer programming using surrogate constraints", *Decision Sciences 8*, 156–166.

[28] Glover F. (1986), "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research 13*, 533–549.

[29] Glover F. (1989), "Tabu search – Part I", *ORSA Journal on Computing 1*, 190–206.

[30] Glover F. (1990), "Tabu search – Part II", *ORSA Journal on Computing 2*, 4–32.

[31] Glover F. (1997), "Tabu search and adaptive memory programming – advances, applications and challenges", in *Advances in Metaheuristics, Optimization and Stochastic Modeling Technologies*, Barr, Helgason and Kennington eds, Kluwer: Boston, MA, 1–75.

[32] Glover F. and M. Laguna (1997), *Tabu Search*, Kluwer: Boston, MA.

[33] Golden B.L., G. Laporte and É. D. Taillard (1997), "An adaptive memory heuristic for a class of vehicle routing problems with minmax objective", *Computers and OR 24*, 445–452.

[34] Hajek B. (1988), "Cooling schedules for optimal annealing", *Mathematics of Operations Research 13*, 311–329.

[35] Hansen P., and N. Mladenovic (1997), "An introduction to Variable Neighborhood Search", Presented at the *2nd Metaheuristic International Conference*, Sophia-Antipolis, France.

[36] Hertz A. and D. de Werra (1987), "Using tabu search techniques for graph coloring", *Computing 39*, 345–351.

[37] Holland J.H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press: Ann Arbor, MI.

[38] Moscato P. (1989), "On Evolution, Search, Optimization Algorithms and Martial Arts: Towards Memetic Algorithms", *Report 826*, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena.

[39] Mühlenbein H., M. Gorges-Schleuter and O. Kräamer (1988), "Evolution algorithms in combinatorial optimization", *Parallel Computing 7*, 65–88.

[40] Osman I. H. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research 41*, 421–451.

[41] Patterson R., E. Rolland and H. Pirkul (1997), "A Memeory Adaptive Reasoning Technique for Solving the Capacitated Minimum Spanning Tree Problem", *Paper WP97-4-007*, Center for Information Technology in Management, Fisher College of Business, The Ohio State University.

[42] Potvin J.-Y. and S. Bengio (1996), "The vehicle routing problem with time windows - Part II: genetic search", *INFORMS Journal on Computing 8*, 165–172.

[43] Potvin J.-Y., T. Kervahut, B.-L. Garcia and J.-M. Rousseau (1996), "The vehicle routing problem with time windows - Part I: tabu search", *INFORMS Journal on Computing 8*, 158–164.

[44] Rochat Y. and É. D. Taillard (1995), "Probabilistic diversification and intensification in local search for vehicle routing", *Journal of Heuristics 1*, 147–167.

[45] Soriano P. and M. Gendreau (1996), "Diversification strategies in tabu search algorithms for the maximum clique problem", *Annals of Operations Research 63*, 189–207.

[46] T. Stützle and H. Hoos (1997), "MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems – Towards Adaptive Tools for Combinatorial Global Optimization", Presented at the *2nd Metaheuristic International Conference*, Sophia-Antipolis, France.

[47] Taillard É. D. (1991), "Robust taboo search for the quadratic assignment problem", *Parallel computing 17*, 443–455.

[48] Taillard É. D. (1993), "Parallel iterative search methods for vehicle routing problems", *Networks 23*, 661–673.

[49] Taillard É. D. (1994), "Parallel taboo search techniques for the job shop scheduling problem", *ORSA journal on Computing 6*, 108–117.

[50] Taillard É. D. (1994), "A diversification/intensification technique for local search applied to vehicle routing problems", Working paper, Centre de recherche sur les transports, Université de Montréal, Canada.

[51] Taillard É. D. (1995), "Comparison of iterative searches for the quadratic assignment problem", *Location Science 3*, 87–105.

[52] Taillard É. D. (1996), "A heuristic column generation method for the heterogeneous VRP", Technical Report CRT-96-03, Centre de recherche sur les transports, Université de Montréal, Canada, forthcoming in *RAIRO-OR*.

[53] Taillard É. D. (1996), "Heuristic methods for large centroid clustering problems", Technical report IDSIA-96-96, IDSIA, Lugano.

[54] Taillard É. D., P. Badeau, M. Gendreau, F. Guertin and J.Y. Potvin (1997), "A tabu search heuristic for the vehicle routing problem with soft time windows", *Transportation Science 31*, 170–186.

[55] Taillard É. D. and L.-M. Gambardella (1997), "Adaptive Memories for the Quadratic Assignment Problem", Technical report IDSIA-87-97, IDSIA, Lugano.

[56] Taillard É. D., G. Laporte and M. Gendreau (1996), "Vehicle routeing with multiple use of vehicles", *Journal of the Operational Research Society 47*, 1065–1070.

[57] Tate D. E. and A. E. Smith (1995), "A genetic approach to the quadratic assignment problem", *Computers & Operations Research 22*, 73–83.

[58] Thangiah S. R. (1995), "An adaptive clustering method using a geometric shape for vehicle routing problems with time windows", in *Proceedings of the Sixth Int. Conf. on Genetic Algorithms*, L. J. Eshelman ed., Morgan Kaufmann: San Mateo, CA, 536–543.

[59] Thangiah S. R., K. E. Nygard and P. L. Juell (1991), "GIDEON: a genetic algorithm system for vehicle routing with time windows", in *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications*, IEEE Computer Society Press, 322–328.

[60] Thangiah S. R., I. H. Osman, R. Vinayagamoorthy and T. Sun (1993), "Algorithms for vehicle routing problems with time deadlines", *American Journal of Mathematical and Management Sciences 13*, 323–355.

[61] Vollmann T. E. and E. S. Buffa (1966), "The facilities layout problem in perspective", *Management Science 12*, 188–204.

[62] Xu J. and J. P. Kelly (1996), "A network-flow based tabu search heuristic for the Vehicle Routing Problem", *Transportation Science 30*, 379–393.

[63] Zufferey N. (1998), "Coloration de graphe à l'aide de méthodes à mémoire adaptative", Diploma thesis, Département de mathématiques, École Polytechnique Fédérale de Lausanne, Lausanne, Suisse.