J Heuristics (2010) 16: 859–879
DOI 10.1007/s10732-009-9121-7

# Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism

**David Meignan · Abderrafiaa Koukam · Jean-Charles Créput**

Received: 28 February 2009 / Revised: 14 November 2009 / Accepted: 27 November 2009 /
Published online: 9 December 2009


**Abstract** We present a self-adaptive and distributed metaheuristic called Coalition-Based Metaheuristic (CBM). This method is based on the Agent Metaheuristic Framework (AMF) and hyper-heuristic approach. In CBM, several agents, grouped in a coalition, concurrently explore the search space of a given problem instance. Each agent modifies a solution with a set of operators. The selection of these operators is determined by heuristic rules dynamically adapted by individual and collective learning mechanisms. The intention of this study is to exploit AMF and hyper-heuristic approaches to conceive an efficient, flexible and modular metaheuristic. AMF provides a generic model of metaheuristic that encourages modularity, and hyper-heuristic approach gives some guidelines to design flexible search methods. The performance of CBM is assessed by computational experiments on the vehicle routing problem.

**Keywords** Combinatorial optimization · Metaheuristic · Multiagent system · Hyper-heuristic

## 1 Introduction

Several recent frameworks of metaheuristics such as I&D Frame (Blum and Roli 2003), Adaptive Memory Programming (AMP) (Taillard et al. 2001) and MAGMA

D. Meignan (✉) · A. Koukam · J.-C. Créput
Laboratoire Systèmes et Transports, Université de Technologie de Belfort-Montbéliard, Belfort, France
e-mail: david.meignan@utbm.fr

A. Koukam
e-mail: abder.koukam@utbm.fr

J.-C. Créput
e-mail: jean-charles.creput@utbm.fr

(Milano and Roli 2004) tend to put the emphasis on simplicity, flexibility and modularity of metaheuristics. These features constitute important criteria for an effective use of metaheuristics, and have been put forward in several articles and surveys (Voss 2001; Blum and Roli 2003).

In Cordeau et al. (2005), the authors defined the simplicity and flexibility criteria in these terms: "Simplicity relates to ease of understanding and coding of an algorithm" and "Flexibility measures the capacity of adapting an algorithm to effectively deal with additional constraints". In addition, robustness can be viewed as the ability to solve different instances of a same problem while maintaining computational performance. Finally, modularity is the capacity of an algorithm to be reused, hybridized or parallelized.

Distributed Artificial Intelligence (DAI), particularly multiagent systems, seems to be a promising field of research to tackle these new issues. Multiagent approach is tightly linked to metaheuristics considering that both approaches can exploit the social metaphor and self-organization paradigm. Thus, multiagent concepts are widely used in metaheuristics, particularly for population-based, hybrid and distributed metaheuristics. For instance, the concept of agent is explicitly used in Co-search metaheuristic (Talbi and Bachelet 2004) or MAGMA's metaheuristics architecture (Milano and Roli 2004). The advantages of using multiagent approach for metaheuristics may be justified by the distribution and robustness inherent to multiagent systems and the need of flexibility and modularity.

The aim of this work is to explore how DAI methods and tools might be exploited to conceive efficient, flexible and modular metaheuristics. To do so we present in this article a Coalition-Based Metaheuristic (CBM). This method, introduced in Meignan et al. (2008a), is based on the Agent Metaheuristic Framework (AMF) and hyperheuristic approach. In CBM, several agents organized in a coalition concurrently explore the search space of an optimization problem. These agents cooperate to perform a better search of solutions. The cooperation consists in exchanging information about the search space and sharing experiences to improve the agents' behavior. The main features of this approach are the use of a heuristic decision process, the introduction of unsupervised learning mechanisms and the exploitation of cooperation between agents. This metaheuristic is then applied to solve the Vehicle Routing Problem (VRP). Computational results are reported to confirm the effectiveness of learning mechanisms and to compare our approach with existing metaheuristics.

This paper is organized as follows. Section 2 introduces the Agent Metaheuristic Framework. Section 3 presents the Coalition-Based Metaheuristic. Then, Sect. 4 is devoted to the application of the metaheuristic to the VRP. The last section gives some conclusions and perspectives.

## 2 The agent metaheuristics framework

The Coalition-Based Metaheuristic presented in the next section is build from Agent Metaheuristic Framework (AMF) (Meignan et al. 2008b). This framework aims at analyzing existing algorithms, and facilitating the design of hybrid or new metaheuristics. It proposes an organizational model of metaheuristics that can be used

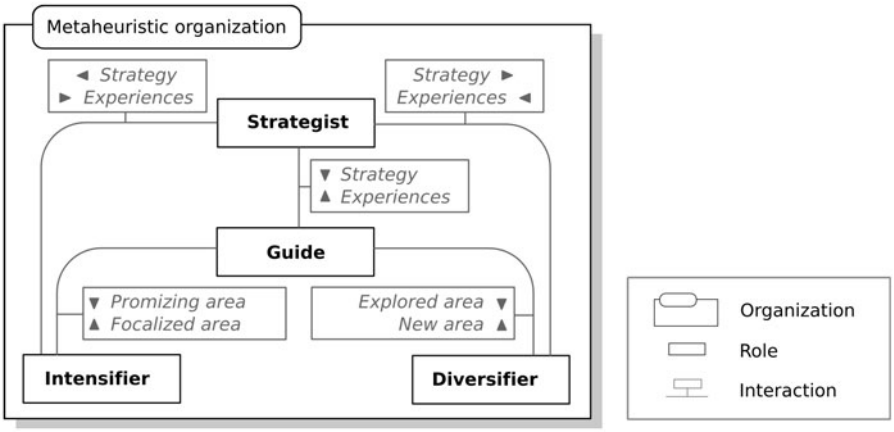


**Fig. 1** AMF organizational model of metaheuristics

to describe both population-based metaheuristics and trajectory methods. In fact, a metaheuristic is viewed as an organization composed of a set of roles which interact in order to find an optimal solution. In this section, we first describe the AMF model of metaheuristic, and then we present a particular instantiation of the model following the hyper-heuristic approach.

## 2.1 Organizational model of metaheuristics

The organizational model of AMF uses the concepts of role, interaction and organization (Gruer et al. 2002) to describe metaheuristics. A role is an abstraction of a behavior or a status defined in an organization. It is associated to an objective to accomplish. In the organizational approach, the concept of role does not match to a particular entity or agent, it can be played by several agents and an agent can play several roles. An interaction links two roles in such a way that an action in the first role produces a reaction in the second. An organization is defined by a set of roles and their interactions associated to the satisfaction of a goal or the execution of a global task.

From these concepts, a metaheuristic is defined as an organization. The goal of this organization is to efficiently explore the search space in order to find high quality solutions in reduced amount of time. This exploration combines intensification and diversification tendencies. To guide the exploration and balance these two tendencies, structured information about the search space is used by subordinate procedures as heuristics. In addition, the strategies used to guide, intensify and diversify may be adapted according to the search experiences. Four roles stems from this definition: Intensifier, Diversifier, Guide and Strategist. The resulting metaheuristic organizational model is described in Fig. 1. The definitions of the four roles composing the metaheuristic model are given below.

*Intensifier and Diversifier*—The Intensifier and Diversifier roles respectively represent the intensification and diversification procedures or tendencies. Thus, the goal of the Intensifier role is to concentrate the search in promising areas of the search

space. On the contrary, the goal of Diversifier role is to move the search to unexplored areas. A comprehensive study of the concepts of intensification and diversification in metaheuristics can be found in Blum and Roli (2003). In a metaheuristic, these two roles can refer to a single procedure or two distinct ones. For instance, in the iterated local search metaheuristic (Lourenço et al. 2003), intensification is performed by a local descent procedure and diversification corresponds to a perturbation procedure. In the ant colony optimization metaheuristic, intensification and diversification tendencies can be identified (Dorigo and Stützle 2000) but they are combined in the decision process of the ants. Thanks to the concept of role, the AMF organizational model manages these two cases.

*Guide*—The goal of the Guide role is to balance and coordinate diversification and intensification. The main element of the Guide role is the memory that stores and provides information for the intensification and diversification. The term "memory" draws from Adaptive Memory Programming (AMP) scheme (Taillard et al. 2001). Memory can take several forms. For instance, in tabu search, the memory is composed of a tabu list; in evolutionary algorithms, the memory is constituted by a population of solutions; in ant colony algorithms, the pheromone trail may be considered as a kind of memory.

*Strategist*—In opposition to the first three roles, the Strategist role is not implemented in all metaheuristics. This role corresponds to the adaptation or self-adaptation mechanisms in metaheuristics. The goal of the Strategist role is to improve the performance of the search process and possibly to reduce parameter setting. The concept of adaptation in AMF is close to the definition given by Hinterding et al. for evolutionary computation (Hinterding et al. 1997). In AMF, adaptation is characterized by the modification or adjustment of the search strategy resulting from the observation of experiences. Thus, adaptation mechanisms use some kind of feedback to determine the nature or amplitude of the change.

In the AMF model, the interactions that link the roles correspond to different information exchanges according to the metaheuristic. The terms "area", "experiences" and "strategy" are generic concepts to designate this information. For instance, in iterated local search, an area corresponds to a solution. More precisely, focalized and explored areas refer to local optimum solutions.

The organizational model of AMF can be considered as a pattern for metaheuristics and several metaheuristics can be analyzed from the roles introduced in the model. In addition, some guidelines presented in Meignan et al. (2008b) are associated to the AMF model in order to help the design of new metaheuristics. This approach helps the modularity and encourages the design of distributed and adaptive metaheuristics. In the next section, this model is used to define the architecture of CBM.

### 2.2 The hyper-heuristic approach

By the identification of common components, the AMF model encourages the design of modular metaheuristics. However, this model does not ensure the possibility to reuse the components to tackle different problems. Indeed, the instantiation of the organizational model can be strongly problem dependent and consequently not

reusable. We call this capacity of adapting an algorithm to effectively deal with additional constraints or other problems the flexibility. In CBM, the flexibility is obtained thanks to the hyper-heuristic approach.

The term hyper-heuristic has been initially used to describe a heuristic selection process used to choose heuristics (Burke et al. 2003). This definition has been recently generalized to designate a search method or learning mechanism for selecting or generating heuristics to solve hard computational search problems (Burke et al. 2009). The resulting metaheuristic model represented in Fig. 2a is composed of two levels. The low level corresponds to a set of heuristics or components of existing heuristics used to solve the optimization problem. These problem dependent components can be considered as black-box systems (Özcan et al. 2008). The upper level corresponds to the hyper-heuristic which schedules, selects or composes low-level heuristics. In most cases, the hyper-heuristic uses the feedback from low-level components to determine the search strategy. The main point in this architecture is that the hyper-heuristic is problem independent, i.e. it has no knowledge of the domain under which it is operating. Interactions that are allowed to cross the domain barrier correspond to selections or evaluations of low-level components.

In Burke et al. (2009), hyper-heuristics are classified according to two criteria, the nature of low-level components and the source of feedback used by the hyper-heuristic.

The first criterion differentiates hyper-heuristics that select heuristics and the ones that generate heuristics from basic components. In the first case, the low-level is composed of heuristics. For generation hyper-heuristics, the low-level corresponds to a set of basic components (building blocks). For instance, the hyper-heuristic of generation presented in Burke et al. (2006) uses a genetic programming system to determine a criterion for incremental solution construction. The building blocks of this criterion are attributes of solution's components and arithmetical functions.

The second criterion in the classification corresponds to the distinction between hyper-heuristics which use online learning, offline learning and no learning. Here, learning refers to the concept of adaptation mechanism in AMF. This mechanism modifies the search strategy from observation of experiences. In online learning hyper-heuristics, the modification takes place while the algorithm is solving the problem instance. For offline learning approaches, the search strategy is defined during a training phase before solving the problem instance.

### 2.3 Combining AMF and hyper-heuristic approach

To combine hyper-heuristic and AMF approaches, we consider hyper-heuristic approach as constraints or guidelines to design metaheuristics. These constraints can be expressed by a refinement of the AMF organizational model. The resulting model is depicted in Fig. 2b. In this model, the components of the hyper-heuristic framework are associated to roles and interactions of the AMF model.

The low-level heuristics or components of heuristics correspond to Intensifier and Diversifier roles. This level works on the problem search space and produces solutions. Note that low-level components can implement both Intensifier and Diversifier roles. The hyper-heuristic level is associated to Guide and Strategist roles. Here the

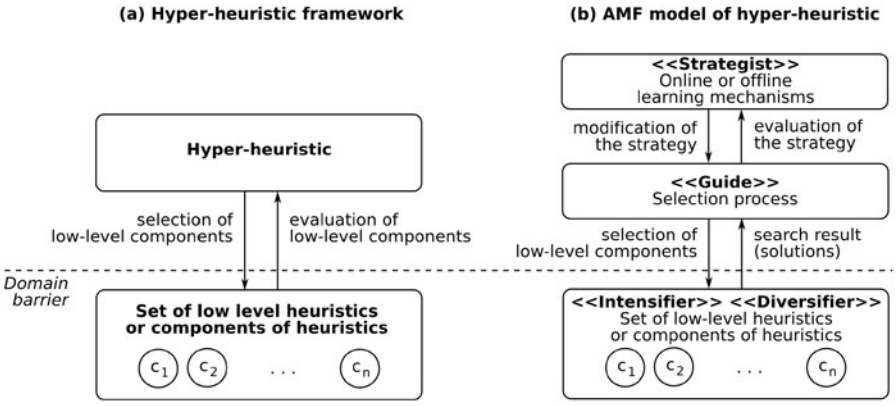


**Fig. 2** Hyper-heuristic framework (**a**) and AMF model of hyper-heuristic (**b**)

Guide role schedules, selects or composes low-level components. Strategist role corresponds to learning mechanisms that modify the strategy of the Guide role.

The CBM (Coalition-Based Metaheuristic) presented in the next section implements this model and combines the advantages of both hyper-heuristic and AMF approaches. Originality of CBM consists, first, in the clear distinction between the heuristics' selection process and learning mechanisms. Second, the metaheuristic is distributed among a coalition of agents that cooperate to improve the search strategy. Finally, CBM combines individual and collective learning mechanisms. Considering the classification of hyper-heuristic in Burke et al. (2009), CBM is a hyper-heuristic of selection with online learning mechanisms. Indeed, the low-level corresponds to a set of heuristics and learning takes place while the algorithm is solving the problem instance.

## 3 The coalition-based metaheuristic

### 3.1 Strategies of cooperation

In Crainic and Toulouse (2003) the authors distinguish three strategies for parallel and distributed implementation of metaheuristics. The first type of distribution concerns the parallel evaluation of solutions or neighborhood moves. This strategy aims solely to speed up computations without achieving a better exploration. The second approach consists in partitioning the set of decision variables or the solution space. It is generally implemented in a master–slave framework without direct interactions between the search processes. The last distribution strategy corresponds to a concurrent and cooperative exploration of the solution space. The distribution strategy in CBM falls into this third category of cooperative metaheuristics. These metaheuristics combine two advantages of parallelism. First, the computational power can be increased by running several tasks simultaneously. Second, cooperation and interaction between processes or agents, can improve the robustness or efficiency of the

search. Different agent-based approaches, such as Co-Search, A-Team and MAGMA, have been proposed in this sense.

In Co-Search (Talbi and Bachelet 2004), the authors propose a multi-agent architecture to better balancing diversification and intensification. The system is composed of three agents with complementary behaviors: a search agent, a diversifying agent and an intensifying agent. The three agents exchange information via an adaptive memory.

In the A-Team metaheuristic architecture (Aydin and Fogarty 2004; Jedrzejowicz and Wierzbowska 2006), each agent corresponds to a particular heuristic or metaheuristic. They cooperate by sharing solutions through a common memory. The main objective of this architecture is to improve the robustness since the performances of the heuristics are context-dependent (Aydin 2007).

The MAGMA architecture proposed in Milano and Roli (2004) is composed of four levels. Each one corresponds to a specific task performed by specialized agents. The first level corresponds to solutions builders, the second one to solution improvers, the third level refers to strategic agents, and the last one includes coordinating agents. This architecture exploits the complementarity of behaviors between levels, and the possible concurrency between agents in a same level.

In addition to the idea of a concurrent exploration of the search space, our approach introduces a collective learning mechanism. Moreover, the distribution principle of CBM is based on the metaphor of the coalition. In the multi-agent field, a coalition is a flat structure where agents have the same capacities and cooperate by means of direct interactions. Agent's cooperation has to contribute to the realization of a common task (Parunak et al. 2003). The concept of coalition also refers to the strong autonomy conferred to the agents (Horling and Lesser 2005). Contrary to the organizations of multi-agent systems in Co-Search, A-Team and MAGMA, where agents behaviors may be different and mainly correspond to a functional decomposition of the optimization process, in our approach the agents have the same initial capacities and behaviors. We essentially investigate a cooperative learning mechanism in the context of a multi-agent hyper-heuristic. The objective is to exploit cooperation in order to dynamically improve the strategy of agents. In addition, the coalition structure is intended to support robustness and facilitate the distribution since control is decentralized, communications between agents are asynchronous, and consequently, the removal or addition of any agent should not perturb the global functioning of the system.

### 3.2 Method principles

In CBM, the coalition is composed of several agents as described in Fig. 3. They concurrently explore the search space and cooperate to improve their search abilities. Thus, each agent implements the model presented in the previous section and plays the four roles of the AMF organizational model.

To perform the search, an agent manages three solutions as in particle swarm optimization (Kennedy and Eberhart 1995): a current solution, the best found solution of the agent and the best solution of the entire coalition. An agent uses several operators which are applied on its current solution. These operators constitute the low level of

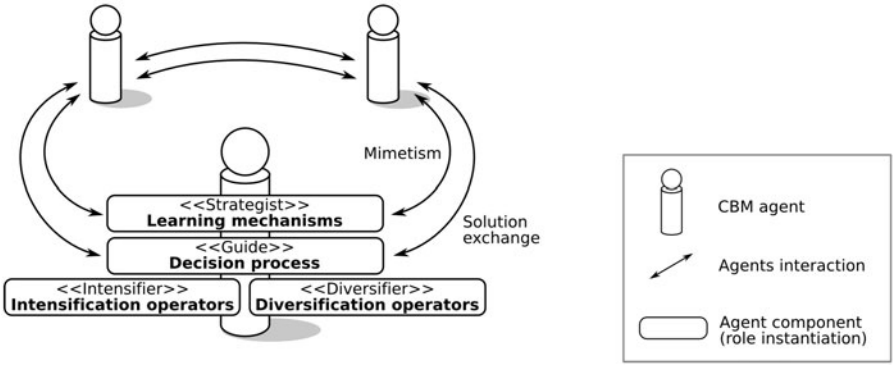


**Fig. 3** Architecture of CBM

the agents and correspond to the Intensifier and Diversifier roles in the AMF model. Intensification operators refer to improvement process such as local search procedure, and diversification operators correspond to generation, mutation or crossover procedures. The schedule of the operators is determined by a decision process. It corresponds to the Guide role in the AMF model. The objectives of this component consist in selecting the most appropriate operators and coordinating intensification and diversification procedures. The selection of operators, which is discussed in more detail later, is based on heuristic rules. In addition, the search behavior of an agent is adapted during the optimization process by an individual reinforcement learning mechanism and mimetism learning. These mechanisms modify the rules of the decision process according to the agents' experiences. In the coalition all agents have the same set of operators but learning mechanisms can lead to different strategies.

The agents have two means of interaction. On the one hand, an agent can inform the rest of the coalition when it improve the best coalition solution. This solution is broadcasted in order to be used for crossover operations and also to compare agents' search abilities. On the other hand, an agent can share its internal decision rules in order to enable mimetism of behavior. This cooperation mechanism is intended to favor the search behaviors that often found better solutions.

### 3.3 Decision process

To perform the selection of operators the agents use a decision process which is close to the ALNS (Adaptive Large Neighborhood Search) heuristic selection principle (Ropke and Pisinger 2006). It is based on a set of rules in form of ⟨condition, action⟩, where the actions correspond to the intensification and diversification operators.

Let $C$ be the set of conditions, $O$ the set of operators. For a condition $c_i$, a weight $w_{i,j}$ is associated to each operator $o_j$. The weight $w_{i,j}$ corresponds to the likelihood of selection of the operator $o_j$ in the condition $c_i$. The effective choice of an operator is performed by a *roulette wheel selection principle*. Thus, the probability $P(o_j|c_i)$ to apply the operator $o_j$ in the condition $c_i$ is computed using the following formula.

$$P(o_j|c_i) = \frac{w_{i,j}}{\sum_{k=1}^{m} w_{i,k}} \quad (1)$$

with:
$C$: $(c_i)_{i=1,\ldots,n}$; Set of conditions
$O$: $(o_j)_{j=1,\ldots,m}$; Set of operators
$W$: $(w_{i,j})_{i=1,\ldots,n;\ j=1,\ldots,m}$; Weight matrix

This simple decision process enables to restrain the choice of operators in a given condition by setting the corresponding weight value to zero. In addition, the augmentation or diminution of a weight value produce respectively an advantage or a restriction of an operator in a given condition. Thus the task of learning mechanisms is to modify the weight values according to the past experiences of the agent.

The set of conditions has been chosen to allow an alternation between intensification and diversification operators and to manage the order and frequency of operators. Thus, each condition is determined by the type of operator previously applied. The first condition corresponds to the previous application of one of the diversification operator. The next conditions are associated to the previous application of each intensification operator. The last condition is activated only when all intensification operators have been applied without modifying the current solution. Since all intensification operators correspond to local descent procedures, this state characterizes a local optimum on all used neighborhood structures. For instance, if two intensification operators $o_1$, $o_2$ and two diversification operators $o_3$, $o_4$ are managed by the decision process, the resulting conditions are:

$c_1$: Diversification operator $o_3$ or $o_4$ have been applied.
$c_2$: Intensification operator $o_1$ has been applied.
$c_3$: Intensification operator $o_2$ has been applied.
$c_4$: Intensification operators $o_1$ and $o_2$ have been applied successively without modifying the current solution.

The weight values in matrix $W$ is initialized with parameter $\alpha$ and several values are set to 0. This initialization determines a cycle in the application of intensification and diversification operators. The Diversification-Intensification cycle (D-I cycle) alternates the application of one diversification operator, then the application of several intensification operators until the last condition is reached. Even if the operators' choice is restricted by this initialization, it is still possible to modify the order and frequency of operators' selection in the D-I cycle. This modification is performed by learning mechanisms.

### 3.4 Learning mechanisms

The agents jointly use two learning mechanisms to adjust their behaviors, reinforcement and mimetism learning mechanisms. The learning is performed during the optimization in order to improve the search strategy of agents. This section describes each of these two learning mechanisms.

In Kaelbling et al. (1996), the authors define reinforcement learning as the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. The two major features of reinforcement learning reported in Sutton and Barto (1998) are trial-and-error search and delayed reward.

In CBM, the problem of selecting the most appropriate operators is viewed as a reinforcement learning problem. During the optimization process, an agent tries

several sequences of operators and it must learn from these experiences. Within the decision model previously presented, an experience is defined as a triplet $\langle$condition $c_i$; operator $o_j$; gain $g\rangle$ where the gain is the fitness difference obtained by the operator application. When an experience has been identified as beneficial (the gain is not necessarily negative), the learning procedure consists in an augmentation by a factor $\sigma$ of the weight value $w_{i,j}$ related to the experience. This mechanism is intended to favor the behaviors that often find new best solutions.

To perform the learning, it is necessary to identify the beneficial experiences and determine a reward. This problem is known as the *credit assignment problem*. It is difficult to evaluate the efficiency of a given operator immediately after its application since it may depend on the order of application of other operators. Thus, beneficial experiences are identified from the observation of a D-I cycle. A reinforcement is realized at the end of a cycle and when a new best found solution has been reached. In this case, the experiences with a non null gain from the last diversification operator application to the current state are reinforced. If the best found solution is not modified in the D-I cycle, then the agent does not modify its weight matrix.

Figure 4 presents a typical case where reinforcement learning procedure is applied. The costs of the best found solution and the current solution of an agent are plotted at the top of Fig. 4. After the application of a diversification operator ($o_3$) and of several intensification operators ($o_1$, $o_2$), the agent improves the cost of its best found solution. Then, a reinforcement is applied on the experiences $\langle c_4; o_3; 10\rangle$, $\langle c_1; o_1; -8\rangle$ and $\langle c_2; o_2; -4\rangle$. Thus, the weights $w_{4,3}$, $w_{1,1}$ and $w_{2,2}$ are augmented to favor the selection of the operators in the same conditions. Figure 4b represents the initial weight matrix ($\alpha = 1$) and the matrix obtained after a reinforcement with $\sigma = 1$. The reinforcement procedure clearly affects the next selections of operators. For instance, if condition $c_1$ is activated after the reinforcement, then the operator $o_1$ has a probability of 66% to be selected against 50% before the reinforcement.

In order to refine the reinforcement learning process, two cases are distinguished, (i) when the agent improves its best found solution, and (ii) when the agent improves the best coalition solution in addition to the best found solution. The learning factors $\sigma_1$ and $\sigma_2$ are respectively used for these two cases. The reinforcement is performed using the formula (2).

$$w_{i,j} = w_{i,j} + \sigma \tag{2}$$

with:
$\langle c_i; o_j; g\rangle$; Experience to reinforce
$w_{i,j}$; Weight related to the experience
$\sigma : \{\sigma_1; \sigma_2\}$; Learning factors

In CBM, agents use a reinforcement procedure to learn individually. The mimetism learning (Yamaguchi et al. 1997) allows cooperation between agents in order to share the behaviors already enhanced by the individual learning. The mimetism learning works on the assumption that an agent tends to behave as the most efficient agents. An agent is considered as more efficient than another one if it improves the best coalition solution in a D-I cycle. When this condition is satisfied, the agent broadcasts its weight matrix to the other agents of the coalition. Then, agents that receive the weight matrix apply a procedure of mimetism learning and resume the

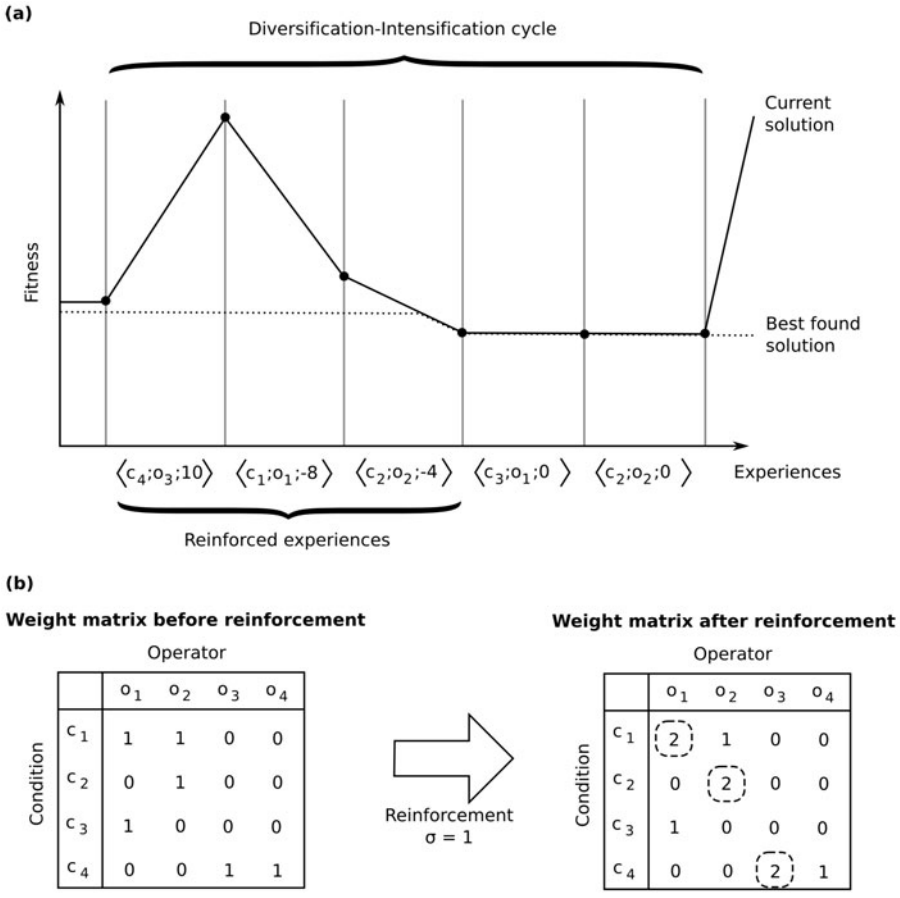


**Fig. 4** Case of reinforcement for individual learning, (**a**) evolution of the current solution cost, (**b**) reinforcement of the weight matrix

search with a modified weight matrix. Let $W_a$ be the weight matrix of the imitator agent $A$ and $W_b$ the weight matrix of the imitated agent $B$. The imitation corresponds to the adoption by agent $A$ of a weight matrix equal to the weighted mean of $W_a$ and $W_b$. The imitation is computed as follow:

$$W_a = (1 - \rho).W_a + \rho.W_b \tag{3}$$

with:
$W_a$; Weight matrix of the imitator agent
$W_b$; Weight matrix of the imitated agent
$\rho$; Mimetism rate

The combination of individual learning and mimetism learning allows to introduce adaptiveness into the population based search, and then to enhance individual and global behavior. An agent exploits its past experiences in order to improve its capacity to find new best solutions, but it also shares its experiences in order to collectively

**Algorithm 1**: CBM agent algorithm

```
   /* Initialization                                                    */
1  s_current ← generate_solution( )
2  W ← init_weight_matrix( )
3  H ← init_experience_memory( )
4  while stopping criterion is not reached do
       /* Choose and apply an operator                                 */
5      c ← compute_condition( H )
6      o ← choose_operator( W , c )
7      s_new ← apply_operator( o , s_current [, s_best coalition] )
       /* Update the experience history                                */
8      gain ← f(s_current) − f(s_new)
9      update_history( H , c , o , gain )
       /* Update solutions                                             */
10     update_solutions( s_new , s_current , s_best found , s_best coalition )
11     if s_best coalition improved by operator application then
12         broadcast_solution( s_best coalition )
13     end
14     if new best coalition solution received from another agent then
15         s_best coalition ← s_received
16     end
       /* Learning mechanisms                                          */
17     if end of Diversification-Intensification cycle then
18         if s_best found improved by the agent in the D-I cycle then
19             individual_learning( W , H )
20         end
21         if s_best coalition improved by the agent in the D-I cycle then
22             broadcast_weight_matrix( W )
23         end
24     end
25     if weight matrix received from another agent then
26         mimetism_learning( W , W_received )
27     end
28 end
```

ensure a better choice of actions in the future. The reinforcement process enables to improve the local behavior. However, mimetism learning allows to exploit the search strategies developed by the other agents.

### 3.5 Behavior of CBM agents

Briefly speaking, the behavior of CBM agents is based on three components: operators, decision process and learning mechanisms. The operators are related to intensification or diversification tasks. Intensification operators refer to improvement

process based on local search procedures, and diversification operators correspond to generation, mutation and crossover procedures. The decision process determines the sequence of operators while maintaining a Diversification-Intensification cycle (D-I cycle). For each application of an operator to the current solution, the agent's set of solutions is updated and the experience is stored. Based on the experiences accumulated during D-I cycles, learning mechanisms modify the rules of the decision process.

The behavior of CBM agents is described in Algorithm 1. In this algorithm, $s_{current}$, $s_{best\ found}$ and $s_{best\ coalition}$ represent the set of solutions managed by the agent. When a solution is received (line 14), this last one is noted $s_{received}$. The attribute $W$ is the decision matrix and $H$ denotes the experience memory. This memory stores the values of $c$ (condition), $o$ (operator index) and *gain* for each operator application in a D-I cycle. In Algorithm 1, an agent iteration starts by choosing and applying an operator. Then, the experience is stored and the set of solutions is updated. Finally, if the agent have reached the end of a D-I cycle, then learning mechanisms are executed. It is important to note that agents have no synchronization point and doesn't necessitate shared memory.

Some advantages of AMF and hyper-heuristic approaches can be observed in the algorithm. First, it is possible to distinguish the realization of AMF roles. Realization of Intensifier and Diversifier roles corresponds to the application of operators (line 7). Guide role performs the choice of operators (lines 5–6) and updates the set of solutions (lines 10–16). Strategist role observes the experiences (lines 8–9) and modifies the weight matrix using individual and mimetism learning (lines 17–27). Thus, the algorithm can be viewed as a particular schedule of roles. Even if the realization of AMF roles does not correspond to distinct processes, CBM stays modular. Identification of AMF roles can facilitate the comprehension and supports the modularity of CBM. Second, all procedures in Algorithm 1, except the operator application, are problem independent. This point, that results from hyper-heuristic approach, ensures the flexibility of CBM.

## 4 CBM for solving the vehicle routing problem

In this section we present the specialization of CBM to solve the Vehicle Routing Problem (VRP). Computational results are also reported to confirm the improvement of performances resulting from the learning mechanisms. Our approach is then compared with several existing metaheuristics.

### 4.1 The vehicle routing problem

The VRP is a well-known problem in the field of transportation and logistics. It has been widely studied since its formulation in Dantzig and Ramser (1959). It consists in finding a set of optimal routes that serve a given set of customers. We use the formulation depicted in Cordeau et al. (2005).

The VRP is defined on a graph $G(V, E)$ where $V = \{v_0, \ldots, v_n\}$ is a set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V; i \neq j\}$ represents a set of edges. The vertex $v_0$

corresponds to the depot while remaining vertices are customers. A quantity $q_i$ of some goods to be delivered by a vehicle and a service time $\delta_i$ required by a vehicle to unload the quantity $q_i$ at $v_i$ is associated to each vertex $v_i, i \in \{1, \ldots, n\}$. A cost or length $c_{i,j}$ is associated to each edge $(v_i, v_j)$. A feasible solution corresponds to a set $R$ of $m$ vehicle routes such that, (i) each route starts and ends at the depot, (ii) each customer is visited exactly once, (iii) the total demand of any route does not exceed the vehicle capacity $Q$ and (iv) the duration of any route does not exceed a bound $D$. The objective is to minimize the total travel time. Here we consider VRP instances with duration constraint, called DVRP (Distance constrained VRP), as instances without duration constraint called CVRP (Capacitated VRP).

## 4.2 CBM specialization

The specialization of CBM for a particular optimization problem requires the definition of diversification and intensification operators. The operators used in our approach partially draw from evolutionary algorithms. Generation, crossover and mutation operators perform the diversification task. Several local descent heuristics are used for intensification. To solve the VRP, six diversification operators and six intensification operators are used by the agents.

### *4.2.1 Diversification operators*

The set of diversification operators is composed of two generation, two crossover and two mutation procedures.

Initial solutions are obtained by generation operators. These operators are also used as diversification operators during optimization. Two generation operators are used: *greedy insertion algorithm* and *sweep algorithm*. The *greedy insertion algorithm* gradually builds the routes by selecting randomly an unserved customer and by inserting it at minimum cost in existing routes. Insertion of a customer is performed by considering the capacity and the duration constraints. The *sweep algorithm* has been introduced by Wren and Holliday (1972). It consists in constructing sequentially the routes from an ordered set of customers. The order of customers is obtained by rotating a ray centered at the depot. Each customer is then inserted in the current route while the capacity and the maximal route length are not exceeded.

Because of their random nature, crossover procedures are considered as diversification operators in CBM. The two crossover operators implemented for the VRP are *route insertion crossover* and *order crossover*. Initial solutions used for these operators are the current solution and the best coalition solution. The *route insertion crossover* creates an offspring solution by inserting a route from the first solution in the other one. To obtain a valid solution without duplication of customers, each customer in the inserted route is removed from other routes. The *order crossover* (Oliver et al. 1987) is a two-point crossover where the offspring tends to inherit the relative order of the customers on the parent routes.

A simple *remove-and-reinsert* procedure is used as a mutation operator. It consists in randomly removing then reinserting one or several customers in such a way that the capacity and the duration constraints are satisfied. Two diversifications operators are defined from this procedure. They differ in the perturbation amplitude (number of random moves applied).

#### 4.2.2 *Intensification operators*

The six intensification operators are based on different neighborhood structures: *2-opt*, *3-opt*, *1-move*, *1-swap*, *edge-move* and *edge-swap*. These neighborhood structures are used in a local descent procedure that consists in performing a sequence of moves towards a local optimum solution. It uses a first improvement policy to select a solution in a randomly ordered neighborhood (Hansen and Mladenović 2003).

The *2-opt* and *3-opt* operators are special cases of $\lambda$-*opt* heuristic (Lin 1965). This mechanism, originally proposed for the traveling salesman problem, is applied to individual routes in the VRP. The $\lambda$-*opt* heuristic is a local search which consists in iteratively replace $\lambda$ edges of the current route by $\lambda$ other ones in such a way that a shorter tour is obtained. A route is said $\lambda$-*opt* optimal if it is impossible to obtain a shorter tour by replacing any $\lambda$ of its edges by any other set of $\lambda$ edges.

The *1-move*, *1-swap*, *edge-move* and *edge-swap* operators are based on the $\lambda$-*interchange* mechanism (Osman 1993) which involves two vehicle routes. The *1-move* neighborhood of a solution corresponds to all possible solutions obtained by moving one customer from a route to another one. A *1-swap* move is obtained by swapping two customers in different routes. The *edge-move* and *edge-swap* neighborhoods are respectively based on moving and swapping two successive customers.

### 4.3 Performance of reinforcement learning mechanism and mimetism

To evaluate reinforcement learning mechanism and mimetism in CBM, three configurations which differ on activated learning mechanisms are considered. The first one corresponds to a coalition of agents without reinforcement learning mechanism and no mimetism. In the second configuration, the agents have the capacity to individually learn by reinforcement. In the last configuration, both individual and collective learning by mimetism are considered. The two first configurations have been obtained by modifying the conditions for learning. To remove reinforcement learning mechanism, we consider that reinforcement condition (Algorithm 1, line 18) is never reached. To remove mimetism, the condition to broadcast the weight matrix (Algorithm 1, line 21) is always considered to be false.

The parameter setting is given in Table 1. These parameters were determined experimentally over a set of combinations, choosing the one that yielded the best average output.

These three configurations of CBM have been tested on the fourteen instances of the Christofides benchmark (Christofides et al. 1979). The CBM has been implemented in Java, on a Pentium 4 clocked at 3 GHz with 1 Gb of memory. For each instance, CBM was run 50 times with 10 agents in the coalition and 1000 iterations

**Table 1** Parameter setting of CBM for the VRP

| Parameter | Description | Value |
|---|---|---|
| $\alpha$ | Initial operator weight value | 1.0 |
| $\sigma_1; \sigma_2$ | Reinforcement factors | 0.5; 1.0 |
| $\rho$ | Mimetism rate | 0.3 |

**Table 2** Comparison of CBM with and without learning mechanisms on Christofides benchmark

| Instance | | | | Without learning | | With RL | | With RL and mimetism | |
|---|---|---|---|---|---|---|---|---|---|
| n° | Cons. | Size | Best known | Dev. % | CPU (min.) | Dev. % | CPU (min.) | Dev. % | CPU (min.) |
| 01 | C | 50 | 524.61 | 0.00 | 0.04 | 0.00 | 0.04 | 0.00 | 0.04 |
| 02 | C | 75 | 835.26 | 1.05 | 0.10 | 1.00 | 0.09 | 0.91 | 0.09 |
| 03 | C | 100 | 826.14 | 0.52 | 0.25 | 0.43 | 0.25 | 0.40 | 0.24 |
| 04 | C | 150 | 1028.42 | 1.68 | 0.53 | 1.41 | 0.50 | 1.35 | 0.46 |
| 05 | C | 199 | 1291.29 | 4.06 | 0.90 | 3.56 | 0.87 | 3.55 | 0.86 |
| 06 | C, D | 50 | 555.43 | 0.12 | 0.09 | 0.10 | 0.09 | 0.10 | 0.09 |
| 07 | C, D | 75 | 909.68 | 0.65 | 0.18 | 0.54 | 0.18 | 0.48 | 0.17 |
| 08 | C, D | 100 | 865.94 | 0.31 | 0.43 | 0.21 | 0.41 | 0.15 | 0.40 |
| 09 | C, D | 150 | 1162.55 | 1.97 | 1.13 | 1.82 | 1.07 | 1.76 | 1.03 |
| 10 | C, D | 199 | 1395.85 | 2.94 | 1.83 | 2.60 | 1.75 | 2.18 | 1.66 |
| 11 | C | 120 | 1042.11 | 12.22 | 0.30 | 11.38 | 0.31 | 11.28 | 0.31 |
| 12 | C | 100 | 819.56 | 1.72 | 0.24 | 1.61 | 0.24 | 1.53 | 0.24 |
| 13 | C, D | 120 | 1541.14 | 1.95 | 0.78 | 1.71 | 0.75 | 1.59 | 0.72 |
| 14 | C, D | 100 | 866.37 | 0.32 | 0.36 | 0.17 | 0.35 | 0.11 | 0.33 |
| Average | | | | 2.11 | 0.51 | 1.90 | 0.49 | 1.81 | 0.47 |

by agent. Table 2 reports the results for each configuration. The first four columns respectively give the instance number, the type of constraint (C for capacity constraint and D for duration constraint), the instance size and the best known value reported in Mester and Bräysy (2007). Then, for each configuration of CBM, the average deviations in % to the best known values and the computation times in minutes are reported.

The results indicate that the addition of Reinforcement Learning (RL) mechanism and the addition of mimetism improve the quality of the solutions found by the coalition with approximately the same computation times. This observation is verified for all instances of the Christofides benchmark. This experimentation illustrates the positive impact of learning mechanisms on the behavior of the agents. During the optimization process, the agents modify the rules of their decision processes and improve their capacities to choose the most appropriate operators. It results an improvement of the final solutions values.

### 4.4 Evaluation against other metaheuristics

CBM have been compared with four powerful heuristics presented in the survey of Cordeau et al. (2005): Granular Tabu Search (GTS) (Toth and Vigo 2003), Unified Tabu Search Algorithm (UTSA) (Cordeau et al. 2001), Active Guided Evolutionary Strategies (AGES) (Mester and Bräysy 2005) and Memetic Algorithm (MA) (Prins 2004). Computational results are reported in Table 3. The deviations and CPU times

**Table 3** Computational results for the Christofides instances

| Instance | | | | CBM | | GTS | | UTSA | | AGES | | MA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n° | Cons. | Size | Best known | Dev. % | CPU (min.)[a] | Dev. % | CPU (min.)[b] | Dev. % | CPU (min.)[c] | Dev. % | CPU (min.)[c] | Dev. % | CPU (min.)[d] |
| 1 | C | 50 | 524.61 | 0.00 | 0.06 | 0.00 | 0.81 | 0.00 | 2.32 | 0.00 | 0.01 | 0.00 | 0.00 |
| 2 | C | 75 | 835.26 | 0.05 | 0.60 | 0.40 | 2.22 | 0.00 | 14.78 | 0.00 | 0.26 | 0.00 | 0.77 |
| 3 | C | 100 | 826.14 | 0.21 | 0.95 | 0.29 | 2.39 | 0.00 | 11.67 | 0.00 | 0.05 | 0.00 | 0.46 |
| 4 | C | 150 | 1028.42 | 0.52 | 4.02 | 0.47 | 4.51 | 0.41 | 26.66 | 0.00 | 0.47 | 0.31 | 5.50 |
| 5 | C | 199 | 1291.29 | 2.05 | 9.37 | 2.09 | 7.50 | 1.90 | 57.68 | 0.00 | 101.93 | 0.69 | 19.10 |
| 6 | C, D | 50 | 555.43 | 0.00 | 0.28 | 0.00 | 0.86 | 0.00 | 3.03 | 0.00 | 0.02 | 0.00 | 0.01 |
| 7 | C, D | 75 | 909.68 | 0.09 | 0.99 | 1.21 | 2.75 | 0.00 | 7.41 | 0.00 | 0.43 | 0.29 | 1.42 |
| 8 | C, D | 100 | 865.94 | 0.00 | 1.77 | 0.41 | 2.90 | 0.00 | 10.93 | 0.00 | 0.44 | 0.00 | 0.37 |
| 9 | C, D | 150 | 1162.55 | 0.70 | 9.47 | 0.91 | 5.67 | 0.46 | 51.66 | 0.00 | 1.22 | 0.15 | 7.25 |
| 10 | C, D | 199 | 1395.85 | 1.24 | 5.92 | 2.86 | 9.11 | 1.50 | 106.28 | 0.38 | 2.45 | 1.74 | 26.83 |
| 11 | C | 120 | 1042.11 | 0.88 | 2.77 | 0.07 | 3.18 | 3.01 | 11.67 | 0.00 | 0.05 | 0.00 | 0.30 |
| 12 | C | 100 | 819.56 | 0.00 | 1.32 | 0.00 | 1.10 | 0.00 | 9.02 | 0.00 | 0.01 | 0.00 | 0.05 |
| 13 | C, D | 120 | 1541.14 | 1.00 | 5.97 | 0.28 | 9.34 | 0.53 | 21.00 | 0.00 | 0.63 | 0.12 | 10.44 |
| 14 | C, D | 100 | 866.37 | 0.00 | 1.13 | 0.00 | 1.41 | 0.00 | 10.53 | 0.00 | 0.08 | 0.00 | 0.09 |
| Average | | | | 0.48% | 3.19 | 0.64% | 3.84 | 0.56% | 24.62 | 0.03% | 7.72 | 0.24% | 5.19 |
| Average normalized CPU time | | | | | 3.19 | | 0.09 | | 12.24 | | 3.84 | | 0.77 |

[a]CPU times on a Pentium IV 3 GHz

[b]CPU times on a Pentium 200 MHz

[c]CPU times on a Pentium IV 2 GHz

[d]CPU times on a Pentium III 1 GHz

**Table 4** Estimated computers performances

| Computer | Approach | Estimated performance (Mflop/s) | Normalization factor |
|---|---|---|---|
| Pentium IV 3 GHz | CBM | 1571 | 1 |
| Pentium 200 MHz | GTS | 38 | 1/41.34 |
| Pentium IV 2 GHz | UTSA | 781 | 1/2.01 |
| Pentium IV 2 GHz | AGES | 781 | 1/2.01 |
| Pentium III 1 GHz | MA | 234 | 1/6.71 |

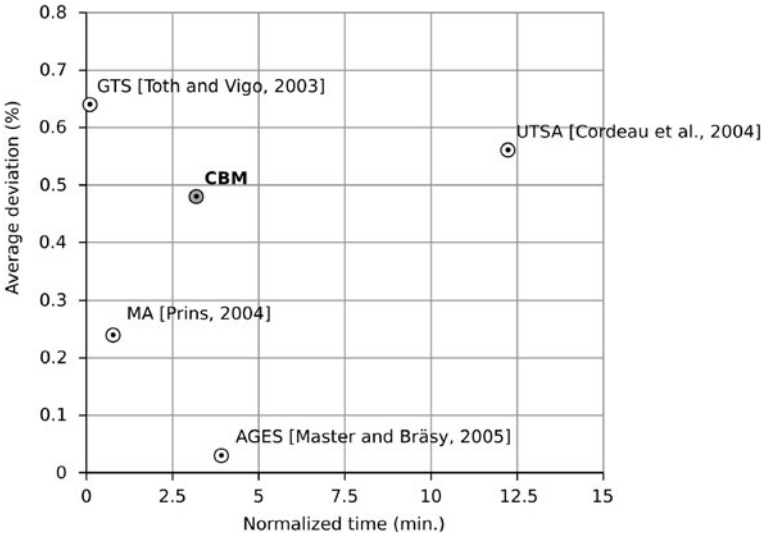


**Fig. 5** Average computational results on Christofides benchmark

for CBM correspond to average values on 10 runs with the parameter setting presented in Table 1 and a coalition size of 20 agents. The stopping criterion is based on the stability of the best coalition solution. After 400 iterations by agents without improving the best coalition solution, all solutions are reinitialized and the coalition is restarted. This restart procedure is stopped when the stability is obtained on the same value that the previous best stable value, or after four restarts.

To compare computation times coming from different computers, we have normalized the average CPU times with the factors presented in Table 4. These factors derive from the performance of computers measured in Mflops/s reported in Dongarra (2006). The normalized average CPU times are given in the last line of the Table 3. To facilitate the comparison of the different heuristics, the average performances are also plotted in Fig. 5.

According to the average results on Christofides benchmark, our method is competitive with some others presented heuristics. Considering the average deviation from the best-known solutions, CBM is better than GTS and UTSA. Our method

also appears to be faster than AGES and UTSA. However, CBM is not yet competitive with solution values of AGES and computation times of MA. It is worth to note that, in a first attempt, we use a simple implementation of the operators. The results of CBM can be improved for instance by a better implementation of the structures used to evaluate the solutions costs, or by reducing the neighborhood size used by intensification operators.

Some additional criteria such as flexibility and modularity have to be considered to evaluate the different approaches since CBM addresses these issues. Flexibility can be defined as the capacity of adapting an algorithm to effectively deal with additional constraints. By extension, an algorithm which is highly problem dependent cannot be considered as flexible. On this criterion, AGES is probably the most complicated and problem dependent of all algorithms used in the comparison (Cordeau et al. 2005). GTS and UTSA exploit the problem structure to define short term and long term memories of the tabu search. Thus, these two heuristics seems to be more problem dependent than CBM and MA that are population based approaches.

The modularity is the capacity of an algorithm to be reused, hybridized or paralleled. Considering this criterion, CBM has several advantages. First of all, new intensification and diversification operators can be easily introduced without modifying the architecture of the agents. These operators are automatically managed thanks to the decision process and learning mechanisms. Then, by using the AMF model, others decision or learning procedures can be considering. Finally, the decentralization in CBM and the asynchronous nature of agents' interactions make CBM a good candidate for a parallel execution.

## 5 Conclusion

In this paper we have introduced CBM, a new multiagent metaheuristic to solve the VRP. This metaheuristic have been designed using the Agent Metaheuristic Framework (AMF) that provides a generic model of metaheuristics.

CBM adopts the metaphor of the coalition. A coalition is composed of several agents which concurrently explore the search space but cooperate to coordinate the search and improve their behaviors. To perform the search, an agent uses several operators which are scheduled by an adaptive decision process. This decision process is based on heuristic rules and follows the hyper-heuristic approach in the sense that it is problem independent. In addition, the decision rules of the agents are adapted during the optimization process by individual learning and mimetism.

CBM exploits several aspects of multiagent systems. The CBM agent architecture is based on the Agent Metaheuristic Framework (AMF) which encourages modularity and reusability. Then, the coalition structure is intended to support robustness and facilitate the distribution, since the control is decentralized and the agents' interactions are asynchronous. Finally, cooperation and learning mechanisms contribute to the effectiveness of the optimization.

The metaheuristic has been applied to the vehicle routing problem. Experiments have been performed to confirm the efficiency of learning mechanisms. Our approach has been also compared with several existing metaheuristics. Computational results

indicate that our approach is competitive with some of the most powerful heuristics. In addition, CBM has several advantages considering modularity and flexibility criteria.

In further works, computational times should be improved by a better implementation of the problem-dependent operators. These works will also compare decision and learning mechanisms in CBM with hyper-heuristics ones. Regarding AMF, future works will study methodological aspects and provide a set of tools for the implementation of metaheuristics. This work might be integrated with existing agent-oriented methodologies and software platforms.

## References

Aydin, M.E.: Metaheuristic agent teams for job shop scheduling problems. In: 3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent Systems for Manufacturing, pp. 185–194 (2007)

Aydin, M.E., Fogarty, T.C.: Teams of autonomous agents for job-shop scheduling problems: an experimental study. J. Intell. Manuf. **15**, 455–462 (2004)

Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)

Burke, E., Hart, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. In: Handbook of Meta-Heuristics, pp. 457–474. Kluwer, Dordrecht (2003)

Burke, E.K., Hyde, M.R., Kendall, G.: Evolving bin packing heuristics with genetic programming. In: Runarsson, T.P., Beyer, H.G., Burke, E., Merelo-Guervos, J.J., Whitley, L.D., Yao, X. (eds.) Parallel Problem Solving from Nature—PPSN IX. Lecture Notes in Computer Science, vol. 4193, pp. 860–869. Springer, Berlin (2006)

Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: A classification of hyper-heuristics approaches. Tech. Rep. Computer Science Technical Report No. NOTTCS-TR-SUB-0907061259-5808, School of Computer Science and Information Technology, University of Nottingham (2009)

Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Combinatorial Optimization pp. 315–338. Wiley, New York (1979)

Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. **52**, 928–936 (2001)

Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., Sormany, J.S.: New heuristics for the vehicle routing problem. In: Logistics Systems: Design and Optimization, pp. 279–297. Springer, Berlin (2005)

Crainic, T., Toulouse, M.: Parallel strategies for meta-heuristics. In: State-of-the-Art Handbook in Metaheuristics, pp. 475–513. Kluwer, Dordrecht (2003)

Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Manag. Sci. **6**(1), 80–91 (1959)

Dongarra, J.J.: Performance of various computers using standard linear equations software. Tech. Rep. CS-89-85, Computer Science Department, University of Tennessee and Computer Science and Mathematics Division, Oak Ridge National Laboratory (2006)

Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: algorithms, applications and advances. Tech. Rep. IRIDIA-2000-32, IRIDIA (2000)

Gruer, P., Hilaire, V., Koukam, A., Cetnarowicz, K.: A formal framework for multi-agent systems analysis and design. Expert Syst. Appl. **23**(4), 349–355 (2002)

Hansen, P., Mladenović, N.: Variable neighborhood search. In: Handbook of Metaheuristics, pp. 145–184. Kluwer, Dordrecht (2003)

Hinterding, R., Michalewicz, Z., Eiben, A.E.: Adaptation in evolutionary computation: a survey. In: IEEE International Conference on Evolutionary Computation, pp. 65–69 (1997)

Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. Knowl. Eng. Rev. **19**, 281–316 (2005)

Jedrzejowicz, P., Wierzbowska, I.: Jade-based a-team environment. In: 6th International Conference on Computational Science, pp. 28–31 (2006)

Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks pp. 1942–1948 (1995). http://www.engr.iupui.edu/~shi/Coference/psopap4.html
Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Tech. J. **44**, 2245–2269 (1965)
Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: Handbook of Metaheuristics, pp. 321–353. Kluwer, Dordrecht (2003)
Meignan, D., Créput, J.C., Koukam, A.: A coalition-based metaheuristic for the vehicle routing problem. In: IEEE Congress on Evolutionary Computation, pp. 1176–1182, (2008a)
Meignan, D., Créput, J.C., Koukam, A.: An organizational view of metaheuristics. In: Jennings, N.R., Rogers, A., Petcu, A., Ramchurn, S.D. (eds.) First International Workshop on Optimisation in Multi-Agent Systems, AAMAS'08, pp. 77–85 (2008b)
Mester, D., Bräysy, O.: Active guided evolution strategies for large scale vehicle routing problems with time windows. Comput. Oper. Res. **32**, 1593–1314 (2005)
Mester, D., Bräysy, O.: Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Comput. Oper. Res. **34**(10), 2964–2975 (2007)
Milano, M., Roli, A.: Magma: a multiagent architecture for metaheuristics. IEEE Trans. Syst. Man Cybern., Part B **34**(2), 925–941 (2004)
Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the traveling salesman problem. In: Grefenstette, J.J. (ed.) International Conference on Genetic Algorithms, pp. 224–230, (1987)
Osman, I.H.: Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann. Oper. Res. **41**(4), 421–451 (1993)
Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intell. Data Analysis **12**(1), 3–23 (2008)
Parunak, H.V.D., Brueckner, S., Fleischer, M., Odell, J.: A design taxonomy of multi-agent interactions. Lect. Not. Comput. Sci. **2935**(4), 123–137 (2003)
Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**, 1985–2002 (2004)
Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Trans. Sci. **40**, 421–438 (2006)
Sutton, R.S., Barto, A.G.: Reinforcement learning: Introduction. Tech. rep., Cognitive Science Research Group (1998)
Taillard, E.D., Gambardella, L.M., Gendreau, M., Potvin, J.Y.: Adaptive memory programming: A unified view of metaheuristics. Eur. J. Oper. Res. **135**, 1–16 (2001)
Talbi, E.G., Bachelet, V.: Cosearch: a parallel co-evolutionary metaheuristic. In: Int. Workshop on Hybrid Metaheuritics, pp. 127–140, (2004)
Toth, P., Vigo, D.: The granular tabu search and its application to the vehicle routing problem. INFORMS J. Comput. **15**, 333–348 (2003)
Voss, S.: Meta-heuristics: the state of the art. In: Local Search for Planning and Scheduling. LNCS, vol. 2148, pp. 1–23 (2001)
Wren, A., Holliday, A.: Computer scheduling of vehicles from one or more depots to a number of delivery points. Oper. Res. Q **23**(3), 333–344 (1972)
Yamaguchi, T., Tanaka, Y., Yachida, M.: Speed up reinforcement learning between two agents with adaptive mimetism. In: IEEE International Conference on Intelligent Robots and Systems, vol. 2, pp. 594–600 (1997)