A Coalition-Based Metaheuristic for the Vehicle Routing Problem

David Meignan Jean-Charles Créput Abderrafiâa Koukam

Abstract— **This paper presents a population based Metaheuristic adopting the metaphor of social autonomous agents. In this context, agents cooperate and self-adapt in order to collectively solve a given optimization problem. From an evolutionary computation point of view, mechanisms driving the search consist of combining intensification operators and diversification operators, such as local search and mutation or recombination. The multiagent paradigm mainly focuses on the adaptive capabilities of individual agents evolving in a context of decentralized control and asynchronous communication. In the proposed metaheuristic, the agent's behavior is guided by a decision process for the operators' choice which is dynamically adapted during the search using reinforcement learning and mimetism learning between agents. The approach is called Coalition-Based Metaheuristic (CBM) to refer to the strong autonomy conferred to the agents. This approach is applied to the Vehicle Routing Problem to emphasize the performance of learning and cooperation mechanisms.**

I. Introduction

This paper introduces the Coalition-Based Metaheuristic (CBM) which combines Evolutionary Algorithm (EA) and Distributed Artificial Intelligence (DAI) principles.

Classical EA approaches are based on a centralized control implicitly stated in the outer loop which synchronizes the sequential application of operators to a population of candidate solutions, e.g. the selection operator may be considered as a centralized supervisor [1]. The CBM adopts a different perspective based on a fully decentralized approach. Indeed, individuals which usually encapsulate a single solution are considered as agents that are organized in a group, called a coalition. In comparison to simple EA individuals, these agents have additional capacities of decision, learning and cooperation. The coalition contains agents with identical capacities which cooperate by the mean of direct peer-to-peer interaction. This type of decentralized structure is intended to support robustness since the removal or addition of any agent do not perturb the global functioning of the system.

DAI, especially multiagent systems, appears as a promising field of research to tackle robustness and modularity in metaheuristics. Indeed, multiagent and metaheuristic approaches are tightly linked because they share the use of social metaphor and self-organization paradigm [2], [3], [4]. However, few approaches such as the Co-Search [5] and MAGMA [6] explicitly use multiagent concepts in the field of metaheuristics. They justify their use by (i) the distribution and robustness inherent to multiagent systems, (ii) the contribution in terms of flexibility and modularity. The aim of this paper is to propose a new metaheuristic combining the advantages of both multiagent and metaheuristic fields. This metaheuristic is applied to the Vehicle Routing Problem (VRP).

The VRP is a well-known problem in the field of transportation and logistics. This problem has been widely studied since five decades. Its common formulation has been proposed by Dantzig and Ramser in [7]. It consists in finding a set of optimal routes that serve a given set of customers.

The VRP is defined on a graph $G(V, E)$ where $V = \{v_0, ..., v_n\}$ 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 δ_i required by a vehicle to unload the quantity q_i at v_i is associated to each vertex $v_i, i \in \{1, ..., n\}$. 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.

The VRP is NP-hard and can rarely be solved exactly for a number of customers exceeding 100. Several heuristics and metaheuristics have been proposed for the VRP. Surveys on these methods can be found in [8], [9].

This paper is organized as follows. Section II presents the CBM and its main components. Section III reports experiments carried out on the capacitated VRP and focuses on the influence of learning and cooperation mechanisms. Finally, the last section is devoted to the conclusion and further research.

II. The coalition-based metaheuristic

A. Method principles

The term "coalition", drawn from [4], [10], refers to an organization where agents have the same capacities and cooperate by the mean of direct interactions. Here, the CBM is the name of a multiagent metaheuristic. The main features of this metaheuristic are a decentralized control and the use of individual and collective learning mechanisms. The CBM is strongly related to Memetic Algorithms (MA) [11], [12], [13] and EA since it is a population-based metaheuristic

David Meignan, Jean-Charles Créput and Abderrafiâa Koukam are with the Systems and Transport Laboratory of the University of Technology of Belfort-Montbéliard, 90010 Belfort cedex, France; Email: david.meignan@utbm.fr, jean-charles.creput@utbm.fr, abder.koukam@utbm.fr.

which uses the metaphor of collective evolution. Each agent or individual manages a solution and performs a search with several intensification and diversification operators. Intensification operators refer to improvement processes such as local search, and diversification operators correspond to generation, mutation or crossover operators.

In CBM the coalition is composed of a fixed number of identical agents. An agent manages a current solution and iteratively applies operators on this solution. The sequential algorithm implementing CBM consists in activating each agent of the coalition. The CBM loop is summarized in the algorithm 1. In this algorithm the activation of all agents is done at each iteration. The only assumption about synchronization between agents is that they must act at a similar rate in order to ensure the collective behavior. No explicit synchronization between agents is needed. An agent can perform a local search while an other one could, at the same time, apply a mutation operator.

Algorithm 1 CBM algorithm

```
1: Initialize the coalition.
2: while stopping criterion is not reached do
3:    for each agent $A$ in the coalition do
4:       Activate agent $A$.
5:    end for
6: end while
```

The main features of an agent in the coalition are a decision process, some learning mechanisms and cooperation means. These elements are reported in figure 1. The choice of an operator is made by a decision process that considers the optimization context. The term "context" is used to describe environmental and agent-related features that can support the decision of an agent. The behavior of an agent is adapted during the optimization by learning mechanisms. In addition, agents can cooperate by two ways. In one hand, an agent shares its best known solution. This solution can be exploited by other agents with crossover operators. This cooperation is intended to guide the search through new promising region of the search space. In another hand, an agent can share its internal decision rules in order to allow mimetism of behavior. This second cooperation mechanism is intended to favor the decision process behaviors that often found new best solutions.

A basic agent iteration, presented in algorithm 2, is composed of four steps: perception, learning, deliberation, and action. The perception allows to characterize the optimization context by computing the current state of the agent and possibly finding an agent to imitate. Then, the learning consists in the adaptation of the agent behavior by reinforcement and mimetism learning. The deliberation corresponds to the choice of an operator to apply. Finally, the action corresponds to the effective application of the operator.

In the next two sections we detail the decision process and the learning mechanisms related to a CBM agent.

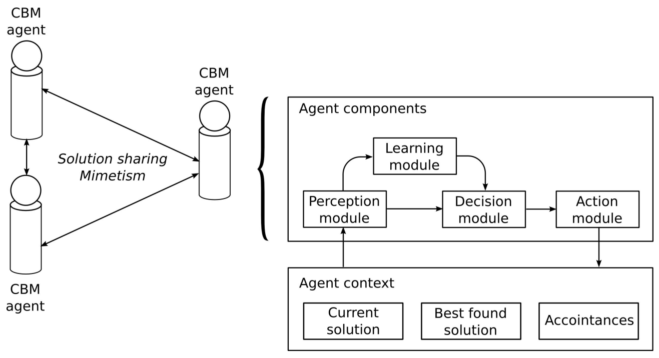

Fig. 1. CBM agent architecture

Algorithm 2 CBM agent iteration

```
1: Compute the current state $s$.
2: Decide of an operator $o$ to apply considering $s$.
3: Learn by reinforcement and mimetism.
4: Apply the operator $o$.
```

B. Decision process

The decision process allows to select an operator according to the optimization context. The different actions of an agent refer to the set of operators, and the optimization context is characterized by a set of states. In order to perform the selection of operators we use the mechanism described in [14], which has some similarity with the Holland Classifier Systems [15], and which is based on a set of condition/action rules. Lets S be the set of states, O the set of operators. For a state s_i a weight $w_{i,j}$ is associated to each operator o_j. The weight $w_{i,j}$ corresponds to the potential of execution of the operator o_j in the state s_i. The effective choice of an operator is performed by a *roulette wheel selection principle*. Thus, the probability $P(o_j|s_i)$ to apply the operator o_j in the state s_i is computed using the following formula.

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

With:

$S : (s_i)_{i=1,\ldots,n}$	Set of states
$O : (o_j)_{j=1,\ldots,m}$	Set of operators
$W : (w_{i,j})_{i=1,\ldots,n;j=1,\ldots,m}$	Weight matrix

Initialization of the weight matrix is made with the parameter α that corresponds to the initial weight. To restrain the behavior of an agent, the weights that correspond to undesirable actions are set to zero. Then, during the optimization, a learning process will adjust the weights according to the past experiences of the agent.

The determination of the set of states is an important step of the design of the algorithm. A state must well characterize the optimization context to take an appropriate decision. Here, the number of states is limited to a number of exclusive conditions characterizing the current step. Considering that an agent has many operators, we have chosen to define the

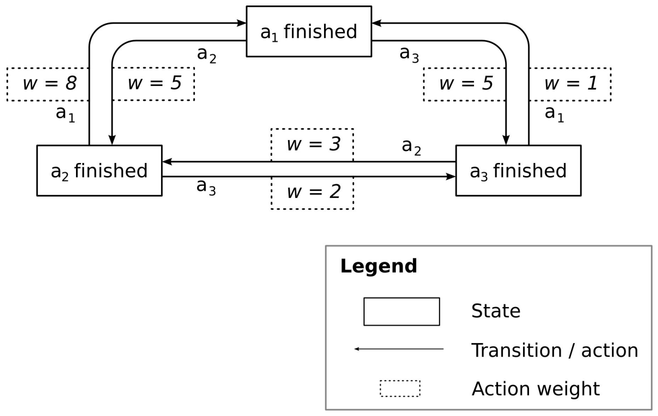

Fig. 2. Example of a decision process

state of an agent by reference to the last operator applied to its current solution. When an agent applies the operator o_i, the state "o_i *finished*" is reached. Thus, the state of an agent represents information about its current solution and about the progress of the optimization process. This definition of states and actions allows an agent to adapt the ordering of the applied operators.

The decision process of an agent can be represented by a finite state probabilistic automaton as reported in figure 2. In this example an agent can perform three actions: a_1, a_2 and a_3. The resulting states are "a_1 *finished*", "a_2 *finished*" and "a_3 *finished*". In a particular state, an agent chooses the next action considering the weights. For instance, in state "a_2 *finished*" the agent can perform actions a_1 and a_3 which have respectively a weight of 8 and 2. Thus, the probability to perform the action a_1 in the state "a_2 *finished*" is of 80%. Moreover, in the state "a_2 *finished*" the agent can not apply action a_2. This restriction corresponds to a null weight: $w_{a_2,a_2 \mathit{finished}} = 0$.

Furthermore, to avoid unnecessary reapplication of a given operator, the last applied rule is inhibited until a new improvement of the current solution occurs. When all operators have successively failed to improve a given solution, a special state "*local optimum*" is reached. This state indicates that the solution has reached a local optimum common to all the intensification operators considered, i.e. no intensification operator can improve the solution. This particular state defines the point where reinforcement learning can take place. The past sequence of the applied rules, since the last diversification step, is then considered for possible reinforcement learning. Then, a new cycle of diversification followed by intensification takes place.

C. Learning mechanisms

The agents use two learning mechanisms to adjust their behaviors, reinforcement learning and mimetism learning. The reinforcement learning is realized during the optimization search in order to favor the sequences of operators which have a positive impact on solution improvement. Since at the beginning of the search, all intensification operators may improve the solution, all the sequences where they appear may be reinforced. Whereas, as the optimization process will progress, it could be the case that only particular order of operator applications will have some positive impact on solution quality. In that case, only such sequences will be reinforced. The mimetism mechanism consists of sharing the decision weight matrix between agents. It can be seen as a recombination operation operating at the meta level of the decision process. A given initiator agent of the interaction modifies its weight matrix considering the best performing agent it founds by a complete examination of the agent coalition.

1) Reinforcement learning: In [16] 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 [17] are: trial-and-error search and delayed reward.

In our case, an agent adjusts its behavior once performed a sequence of actions, starting from a diversification step that has conducted its solution into a common local minimum according to all the different intensification operators. Several sequences of operators are tested during the optimization thanks to the roulette wheel selection principle. A reinforcement learning is performed only when the agent find a solution better than its previous best found solution. In this case, the action plan (sequence of operators) which conducted the agent to find this solution is reinforced. Within the decision model previously presented, an experience is a pair state/operator (s_i; o_j) and the reinforcement corresponds to an augmentation of the related weight value $w_{i,j}$. This mechanism is intended to favor the behaviors that often find new best solutions.

To perform the reinforcement learning, it is necessary to identify the beneficial experiences and assign them 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 an action sequence performed by the agent. A reinforcement is realized when the current solution fitness is better than the one's of the best previously obtained solution of the agent. The experiences from the last diversification operator application to the current state are reinforced.

Figure 3 presents a typical case where the reinforcement learning is applied. The cost of the best found solution and the current solution are plotted. After the application of a diversification operator (o_3) and 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 (*local optim.;* o_3), (o_3 *finished;* o_1) and (o_1 *finished;* o_2).

In order to refine the reinforcement learning, two cases are distinguished, (i) when the agent improves its best found solution, and (ii) when the agent improves the best known solution value it previously obtained during its past interac-

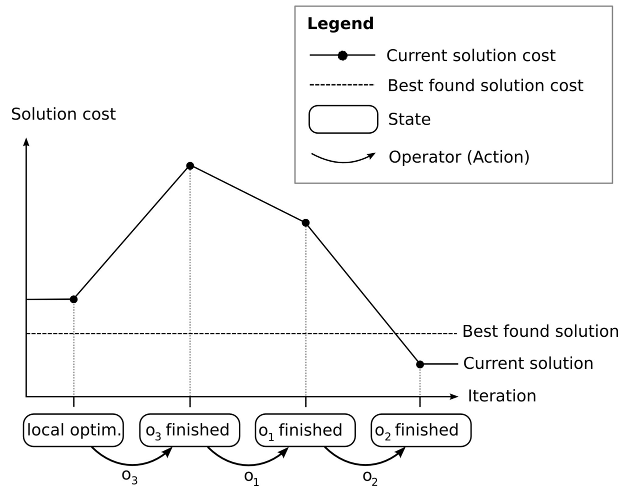

Fig. 3. Case of reinforcement learning.

tions with other agents. The reinforcement factors σ_1 and σ_2 are respectively used for the two types of reinforcement. The reinforcement is performed using the formula (2).

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

With:

$(s_i;\, o_j)$	Experience to reinforce
$w_{i,j}$	Weight related to the experience
$\sigma : \{\sigma_1; \sigma_2\}$	Reinforcement factor

2) Mimetism learning: In the coalition-based metaheuristic, agents perform reinforcement learning individually. The mimetism learning [18] allows cooperation between agents in order to share the behaviors already enhanced by the reinforcement learning.

The mimetism learning works on the assumption that an agent imitates the behaviors of the most efficient agents. At each cycle, the agent examines the fitness value of the best solution found by each other agent of the coalition. When an agent *A* observes that the agent *B* has found the best solution value, the agent *A* imitates the behavior of the agent *B*. Lets W_A be the weight matrix of agent *A* and W_B the weight matrix of 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
ρ	Mimetism rate

The combination of reinforcement learning and mimetism learning allows to introduce adaptation and self-adaptation 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 ensure a better choice of actions in the future. The reinforcement learning allows to improving the local behavior. However, imitation learning permits to exploit the search strategies developed by the other agents.

D. Specializing CBM for solving the Vehicle Routing Problem

The specialization of CBM for a particular optimization problem necessitates to define the diversification and intensification operators. The operators used in our approach are drawn from Operations Research and Evolutionary Algorithms. Generation, crossover and mutation operators perform the diversification task. Several standard local search heuristics are used as intensification operators.

1) Generation operators: Initial solutions are obtained by generation operators. These operators are also used as diversification operators during optimization. Two different 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 in [19]. 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 or the maximal route length is not exceeded.

2) Crossover operators: Two crossover operators are used: *route insertion crossover* and *order crossover*. They are applied in a random way. A given agent randomly chooses a mating agent, and replaces its embedded solution by the new offspring solution generated.

In *route insertion crossover*, an offspring is created by inserting a route from one solution to an other solution. To obtain a valid solution without duplication of customers, each customer in the inserted route is removed from other routes.

The *order crossover* (OX) [20] is a Two-point crossover where the offspring tends to inherit the relative order of the customers on the parent routes.

3) Mutation operators: A simple *Remove-And-Reinsert* (RAR) 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.

4) Intensification operators: Four different local search operators are used for the purpose of intensification: *2-opt*, *3-opt*, *1-move* and *1-swap* heuristics.

The *2-opt* and *3-opt* heuristics are special case of λ-*opt* heuristics [21]. This heuristic, originally proposed for the Traveling Salesman Problem (TSP), is applied to individual routes in the VRP. The λ-*opt* heuristics is a local search, where at each step λ edges of the current route are replaced by λ other ones in such a way that a shorter route is obtained. A route is said λ-*opt* optimal if it is impossible to obtain a shorter one by replacing any λ of its edges by any other set of λ edges.

The *1-move* and *1-swap* heuristics are based on λ-*interchange* mechanisms [22] which involve two vehicle routes. The *1-move* heuristic, also known as *1-String Relocation* heuristic, is a local search where one customer is moved from one route to another route. The *1-swap* heuristic, also known as *1-String Exchange* heuristic, consists of a local search where two customers in different routes are swapped.

III. Computational results

The application of the coalition based metaheuristic to the vehicle routing problem has been tested on the 14 instances described in Christofides et al. [23]. These instances contain between 50 and 199 customers in addition to the depot. Half of the instances have only capacity constraint, and the other ones have in addition a time duration constraint. The CBM has been implemented in Java and tested on a Pentium 4 at 3GHz. with 1Gb. of memory.

The following experiments are performed to assess the improvement of performances resulting from the learning mechanisms proposed, according to different sizes of the population, and to evaluate the performances of the approach against some of the powerful heuristics of Operations Research.

The parameter setting of the CBM is given in table I. The values used for computational experiments are also reported. In addition to these parameters, the behavior of an agent can be restricted thanks to a particular initialization of the decision process. It is possible to disable the undesirable actions of an agent by setting the corresponding weights to zero in the weight matrix *W*. For example, we force an agent to only apply intensification operators until its solution reaches a local optimum. As well, an agent can not apply a diversification operator subsequently to another diversification operator, but must necessarily apply an intensification operator.

TABLE I

CBM Parameters

Parameter	Description	Value
α	Initial operator weight value	5
$\sigma_1;\sigma_2$	Rewards for reinforcement learning	1;2
ρ	Mimetism rate	0.4
C	Number of agents in the coalition	15

A. Performance of reinforcement learning and mimetism

Starting with a referential version of the algorithm with no learning mechanism, we successively introduce the reinforcement learning and the mimetism learning and evaluate the deviation of the average route length to the best known solution values reported in [24].

The CBM has been experimented for different coalition sizes between 1 to 20 agents on the 14 Christofides instances. To make the evaluation fair, the total number of iterations performed by the agents was fixed, and remained constant,

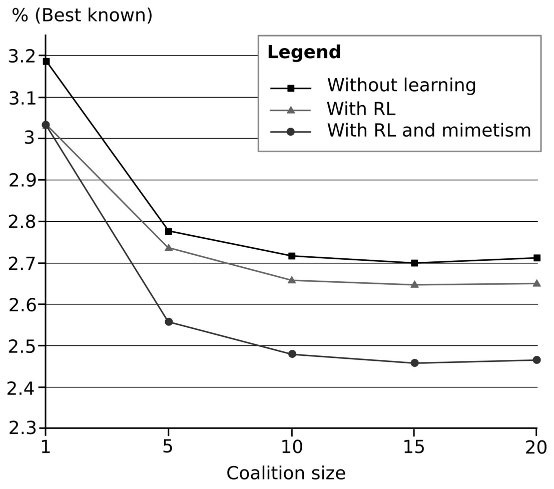

Fig. 4. Impact of the learning and the coalition size on the quality of the solution

for all the tests to $10,000$ iterations. Thus, in a coalition with C agents, a single agent performs $10,000/C$ iterations. The computation time allowed for each configuration of the algorithm was approximately of 30 seconds, the introduction of the learning mechanisms and the augmentation of the population size having a negligible impact on this value.

For each coalition size, CBM is executed 10 times for each of the 14 instances. The average percentage deviations to the best known values are reported in Figure 4. Three different configurations are considered. The first one corresponds to a coalition of agents without Reinforcement Learning (RL) and no mimetism. In this case, the only cooperation mechanism is provided by the standard crossover operators. In the second configuration, the agents have the capacity to individually learn by reinforcement. In the third configuration, both individual and collective learning by mimetism are considered.

It can be observed on the figure that the additional learning capacities improve the quality of the solutions found. In addition, the improvement already carried out by using learning seems to be more pronounced when the population size increases, particularly by using the mimetism learning. Beyond 15 agents, the computational results are slightly deteriorated. This can be explained by the small number of iterations performed by a single agent. The experimentations illustrate the contribution of the cooperation in CBM.

Figure 5 plots the average costs for a population size of 15 agents, according to the number of iterations performed by a single agent $(10,000/15)$. We can see that the benefice of learning appears when the iteration number grows. While at the beginning of the run, the different configurations yield similar and undifferentiated improvements, as the run goes on, a difference appears on the length minimization in favor of the learning configurations. This can be explained by the fact that all operators may have a positive impact at the earlier stages of the search, whereas the system progressively learns the good order of operator applications as the search evolves toward near-optimal solutions, from the middle to the end of the run. As illustrated in the figure 5, these trials cast a light

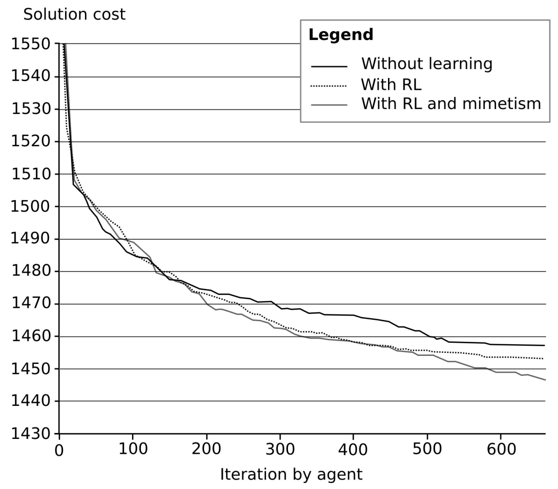

Fig. 5. Progress of the average cost for a population size of 15 agents on the Christofides instance 10

on the improvements carried out by the cooperating learning mechanisms.

B. Evaluation against Operations Research heuristics

Here, we evaluate the CBM approach against two powerful Operations Research heuristics presented in the survey of Cordeau et al. [24]. We compare performances with two tabu search approaches: the Granular Tabu Search (GTS) [25], and the Unified Tabu Search Algorithm (UTSA) [26]. We use the results reported in [24]. These approaches are selected because they are considered as being ones of the most simple and flexible approaches in the literature. We think that they are the better choice for comparison since CBM also addresses simplicity and flexibility by offering three independent levels of modeling, that are the learning level, the population based metaheuristic level with cooperation, and the problem-dependant heuristic level.

The computational results are presented in Table II. For each problem, 10 runs are performed and the average and best solution found are considered. The first four columns respectively give the problem name, the type of constraints, the number of customers and the best known solution value taken from [24]. In the constraints type column, the "C" indicates a capacity constraint, and the "D" indicates a duration constraint. The columns 5 to 8 respectively report the average deviation of the cost, given in pourcentage, to the best known value, the best found value over the 10 runs, the standard deviation, and the computation time per run in minutes. The other columns report the average route length deviation and the computation time in minutes respectively for the two other approaches. The GTS was evaluated on a Pentium (200 MHz), and the UTSA on a Pentium 4 (2 GHz). Our experiments were performed on a Pentium 4 (3 GHz) with a Java program.

The results indicate that our CBM approach is not yet competitive to the powerful Operations Research heuristics. Nevertheless, while considering the different materials used, with an average deviation of 2.47 % and an average computation time of 0.5 minute per run, CBM is not clearly dominated, on both quality and computation time, by the UTSA, which yields 0.56 % of deviation in roughly 25 minutes. On the contrary, to be competitive with the GTS, solution quality produced by the CBM, as well as computation time, would have to be improved both by a factor at least 5. However, it is worth noting that, in a first attempt, we use a naive implementation of the operators. We did not use implementation tricks such as candidate lists, don't look bits, or k-d trees which generally have a great impact on computation times and then on the overall performances. This point is illustrated here by considering the UTSA which is very slow and the GTS which is very fast. This is because the latter uses such implementation tricks, whereas not the former, for example by eliminating edges having an important cost. It is often the case that the most powerful approaches are also the most complicated ones. Such an example is the Active Guided Evolution Strategy (AGES) [27] which is, at the date of writing, the overall winner considering both solution quality and computation time, but which is considered complicated to implement and understand [24].

IV. Conclusion

In this paper we have proposed a multiagent metaheuristic, merging some of the evolutionary algorithm concepts into a context of distributed control and agent-based learning. The metaphor of individuals being evolved and subject to the process of natural selection becomes a metaphor of autonomous agents learning and cooperating, and thus making evolve their embedded candidate solutions in order to intentionally solve a given problem.

In CBM, several agents organized in a coalition treat simultaneously the optimization problem with identical search capacities. Each agent manages a current solution and performs a search by applying intensification and diversification operators. An agent uses a decision process based on a roulette wheel selection principle to determine the operator to apply. The behavior of an agent is adapted during the search by a reinforcement learning mechanism. In addition, the agents can cooperate in two ways: solution sharing and mimetism. Agents have the same status. They are entities with their own local memory and communicating through an asynchronous network. In that way, parallelization of the approach may be facilitated and robustness according to injuries enhanced.

In this paper the efficiency of CBM was illustrated thanks to its application to the Vehicle Routing Problem. The computational results put forward the performances of the reinforcement learning and of the mimetism cooperating mechanism. In further works, performance should be improved by a better implementation of the standard problem-dependant operators. Also, many other learning approaches, for example artificial neural networks, and not yet considered into the context of adaptive search in combinatorial optimization, will merit a thorough examination.

TABLE II

COMPUTATIONAL RESULTS FOR THE CHRISTOFIDES INSTANCES

				CBM				UTSA		GTS	
Instance	Type	Size	Best known	% Average	% Best	St. Dev.	Time (min.)	% Best	Time (min.)	% Best	Time (min.)
1	C	50	524.61	**0.00**	**0.00**	0.00	0.06	**0.00**	2.32	**0.00**	0.81
2	C	75	835.26	1.07	0.06	0.77	0.12	**0.00**	14.78	0.40	2.21
3	C	100	826.14	0.38	0.15	0.19	0.31	**0.00**	11.67	0.29	2.39
4	C	150	1 028.42	1.71	0.73	0.44	0.64	0.41	26.66	0.47	4.51
5	C	199	1 291.26	4.05	2.86	0.80	1.10	1.90	57.68	2.09	7.50
11	C	120	1 042.11	14.56	13.86	0.60	0.40	3.01	11.67	0.07	3.18
12	C	100	819.56	1.53	0.19	1.08	0.30	**0.00**	9.02	**0.00**	1.10
6	C, D	50	555.43	0.02	**0.00**	0.07	0.12	**0.00**	3.03	**0.00**	0.86
7	C, D	75	909.68	0.32	**0.00**	0.33	0.25	**0.00**	7.41	1.21	2.75
8	C, D	100	865.94	0.26	**0.00**	0.31	0.47	**0.00**	10.93	0.41	2.90
9	C, D	150	1 162.55	2.55	1.58	1.03	1.25	0.46	51.66	0.91	5.67
10	C, D	199	1 395.85	3.39	2.23	0.85	1.98	1.50	106.28	2.86	9.11
13	C, D	120	1 541.14	4.65	1.56	2.20	0.80	0.53	21.00	0.28	9.34
14	C, D	100	866.37	0.08	**0.00**	0.21	0.39	**0.00**	10.53	**0.00**	1.41
Average C				3.33	2.55	0.55	0.42	0.76	19.11	0.47	3.10
Average C, D				1.61	0.77	0.72	0.75	0.36	30.12	0.81	4.58
Average				2.47	1.66	0.63	0.58	0.56	24.62	0.64	3.84

REFERENCES

[1] M. Tomassini, "Parallel and Distributed Evolutionnary Algorithms : a Review", *Evolutionary Algorithms in Engineering and Computer Science*, pp. 113–133, 1999.

[2] J. C. Créput, A. Koukam, "Self-Organization in Evolution for the Solving of Distributed Terrestrial Transportation Problems", *Softcomputing applications in industry*, B. Prasad (eds.), pp. 189–205, Springer-Verlag, 2008.

[3] E. H. Durfee, "Distributed Problem Solving and Planning", In Gerhard Weiss (eds.),*Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pp. 121–164, 1999.

[4] J. Ferber, "Multi-agent Systems : An Introduction to Distributed Artificial Intelligence", Addison Wesley, 1999.

[5] E. Talbi and V. Bachelet, "COSEARCH: A Parallel Co-evolutionary Metaheuristic", In proc. *First int. workshop on Hybrid Metaheuritics*, pp. 127–140, 2004.

[6] M. Milano and A. Roli, "MAGMA: a multiagent architecture for metaheuristics", *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 34, pp. 925–941, 2004.

[7] G. B. Dantzig and J. H. Ramser "The Truck Dispatching Problem", *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[8] J. F. Cordeau, M. Gendreau, G. Laporte, J-Y. Potvin and F. Semet, "A guide to vehicle routing heuristics", *Journal of the Operational Research Society*, vol. 53, pp. 512–522, 2002.

[9] M. Gendreau, G. Laporte and J-Y. Potvin, "Metaheuristics for the capacitated VRP", In P. Toth and D. Vigo (eds.), *The Vehicle Routing Problem*, Chapter 6, pp. 129–154, 2002.

[10] H. V. D. Parunak, S. Brueckner, M. Fleischer and J. Odell, "A design taxonomy of multi-agent interactions", AOSE 2003, *Lecture notes in computer science*, vol. 2935, no. 4, pp. 123–137, 2003.

[11] N. Krasnogor and J. Smith, "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues", *IEEE Trans. on Evolutionary Computation*, pp. 474–488, vol. 9, 2005.

[12] P. Merz and B. Freisleben, "Memetic Algorithms for the Traveling Salesman Problem", *Complex Systems*, vol. 13, pp. 297–345, 2001.

[13] P. Moscato and C. Cotta, "A Gentle Introduction to Memetic Algorithms", In F. Glover, G. Kochenberger (eds.), *Handbook of Metaheuristics*, Kluwer Academic Publishers, pp. 105–144, 2003.

[14] S. Ropke and D. Pisinger, "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows", *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006.

[15] J. H. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Goldberg, S. Forrest, R. L. Riolo, R. E. Smith, P. L. Lanzi, W. Stolzmann and S. W. Wilson, "What Is a Learning Classifier System?", *Lecture Notes In Computer Science*, vol. 1813, pp. 3–32, 2000.

[16] L. P. Kaelbling M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research* vol. 4, pp. 237–285, 1996.

[17] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", MIT Press, 1998.

[18] T. Yamaguchi, Y. Tanaka and M. Yachida, "Speed up reinforcement learning between two agents with adaptive mimetism", In proc. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 594–600, 1997.

[19] A. Wren and A. Holliday, "Scheduling of Vehicles from One or More Depots to a Number of Delivery Points", *Operational Research Quarterly*, vol. 23, pp. 333–344, 1972.

[20] I. M. Oliver, D. J. Smith and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem", In proc. Second Int. Conf. on Genetic Algorithms, pp. 224–230, 1987.

[21] S. Lin, "Computer Solutions of the Traveling Salesman Problem", *Bell System Technical Journal*, vol. 44, pp. 2245–2269, 1965.

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

[23] N. Christofides, A. Mingozzi and P. Toth, "The vehicle routing problem", In N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.), *Combinatorial Optimization*, Wiley, pp. 315–338, 1979.

[24] J. F. Cordeau, M. Gendreau, A. Hertz, G. Laporte and J. S. Sormany, "New Heuristics for the Vehicle Routing Problem", In A. Langevin, D. Riopel (eds.), *Logistics Systems: Design and Optimization*, Springer, pp. 279–297, 2005.

[25] P. Toth, D. Vigo, "The granular tabu search and its application to the vehicle routing problem", *INFORMS Journal on Computing*, vol. 15, pp. 333-348, 2003.

[26] J. F. Cordeau, G. Laporte and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows", *Journal of the Operational Research Society*, vol. 52, pp. 928–936, 2001.

[27] D. Mester and O. Bräysy, "Active Guided Evolution Strategies for Large Scale Vehicle Routing Problems with Time Windows" *Computers & Operations Research*, vol. 32, pp.1593–1614, 2005.