Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Multiagent Architecture for solving Combinatorial Optimization Problems through Metaheuristics

Filipe Costa Fernandes, Sérgio Ricardo de Souza, Maria Amélia Lopes Silva
Henrique Elias Borges, Fábio Fernandes Ribeiro
PPGMMC, CEFET/MG, Belo Horizonte, Brazil
fcfernandes@gmail.com, sergio@dppg.cefetmg.br, mamelials@gmail.com, hborges@dppg.cefetmg.br, fabiobh@gmail.com

***Abstract*—This article introduces MAM - Multiagent Architecture for Metaheuristics, whose objective is to combine metaheuristics, through the multiagent approach, for solving Combinatorial Optimization Problems. In this architecture, each metaheuristic is developed in the form of an autonomous agent, cooperatively interacting in an Environment. This interaction between one or more agents is carried out through information exchange in the search space of the problem, seeking to improve the same objective. MAM is a flexible architecture, which can be used for solving different optimization problems, without the need to rewrite algorithms. In this paper, the MAM architecture is specialized for Genetic Algorithm (GA), Iterated Local Search (ILS) and Variable Neighborhood Search (VNS) metaheuristics in order to solve the Vehicle Routing Problem with Time Windows (VRPTW). Computational tests were performed and results are presented, showing the effectiveness of the proposed architecture.**

***Index Terms*—Frameworks for Metaheuristics, metaheuristics, autonomous multiagent system, Problem solving.**

I. Introduction

Recently, multiagent approach has been applied to the treatment of combinatorial optimization problems in several works. Strong economic relevance and the impact on real life of many of these problems justify the interest in applying new techniques to solve these problems. Thus, the search for better solutions becomes very important, especially in development of solution techniques that lead to flexibility in incorporating new methods without requiring the effort of reworking its implementation.

In this paper, a multiagent architecture called Multiagent Architecture for Metaheuristics, MAM, is proposed for solving combinatorial optimization problems by means of metaheuristics.

In contrast with mathematical programming methods, metaheuristics do not guarantee that an optimal solution can be found. Instead, they search for a fairly viable solution to the problem in a reduced computational time. For this reason, metaheuristics are able to offer efficient solutions to optimization problems based on the combination of different strategies that allow them to exploit the search space of the problem. An interesting way to use the aforementioned features is through systems composed of several autonomous agents in constant interaction, competing and collaborating with one another to achieve a common objective [1].

Several works have been produced using this approach. Among them, it is important to cite [2], in which the Asynchronous Teams architecture (A-teams) is presented, and [3], in which a specific architecture for agents involved in finding solution to the Vehicle Routing Problem (VRP) is shown. Besides these, the main reference of the present paper is [4], which introduces a hierarchical multiagent architecture called MAGMA. In this architecture, a metaheuristic can be seen as a result of interactions between different kinds of agents. The agents are ordered in four levels of abstraction and there are one or more agents on each of these levels. Therefore, in the MAGMA architecture, a metaheuristic is not an isolated agent acting autonomously, but arising from the action of different agents, acting at several levels of abstraction in this architecture. The environment that the agents perceives in the MAGMA architecture is very simple, just the group of agents with which they communicate. The main characteristic of this framework is its hierarchical structure, which prevents the autonomy of the agents. Two collateral approaches are also related, in some sense, with the one proposed here. The first is the hyperheuristic approach, due to Burke and collegues [5]. Hyperheuristics are "*heuristic to choose heuristics*" and, according to [5], they "*are concerned with the search space of heuristics rather than potential solutions*". Nothing concerning agents and multiagents properties and applications is dealt with this approach. The second collateral approach is cooperative strategies for optimization, proposed by Crainic and others [6]. In this case, a set of metaheuristics are run in parallel, exchanging information through a coordinator, but no autonomy is provided to them.

The multiagent architecture proposed here includes characteristics that distinguish it from others. Agents are autonomous and non-hierarchical; the environment must be considered and modeled as a primary abstraction that provides the conditions for the existence and for the autonomy of the agents; the architecture must be flexible, allowing applications for different combinatorial problems; interaction between the agents (i.e, communication between them) should be asynchronous in order to ensure its autonomy; agents must be cooperative with each other; finally, the architecture must be designed at the high level of abstraction, thus being robust and scalable. The present article is organized as follows: the main aspects of the MAM architecture are presented in Section II, where all types of agents and acting environments are defined, as well as the aspects of the internal structure of the search agents. Section III presents architecture specializations and

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

computational results concerning its application to the solution of a standard problem. Section IV ends the paper, with final considerations and future perspectives for the continuity of the construction of this multiagent architecture.

II. MAM Multiagent Architecture

MAM architecture combines several metaheuristics in order to deal with combinatorial optimization problems with a multiagent approach. Each metaheuristic is developed as an autonomous agent, able to interact with its environment (search space problem) and among themselves in a cooperative and competitive way, in an attempt to find the best solution to combinatorial optimization problems. The framework is built to be flexible, for allowing its use in any class of combinatorial optimization problems, and to be scalable, meaning that various metaheuristics can be added to it as new agents in order to implement and encapsulate specific metaheuristics with a minimum impact on the remaining architecture. In consequence, different optimization problems can be solved without much additional effort.

For this reasons, MAM is an architecture with a high abstraction level. The environment where the metaheuristic agents exist is an integral part of MAM architecture and should be modeled to tackle any particular optimization problem. In this environment, all specific aspects of a particular problem must be specified and the architecture defines and incorporates all communication mechanisms among the agents and its environment. Thus, the modeling effort to build new MAM instances to deal with a particular problem or classes of different problems is minimized.

Figure 1 defines the conceptual model of MAM, formed by agents and environment, as well as the interactions among agents and between agents and environment. Note that capital letters identify components of MAM. The main components of the structure are (1) Environment; (2) Constructor Agent; (3) Local Search Agent; (4) Metaheuristic Agent; (5) Coordinator Agent and (6) Solution Analyzer Agent.

To provide a better understanding of MAM architecture, it is important to define the role of each agent. The Solution Constructor, Local Search and Metaheuristic Agents can be classified according to a general class of Search Agents, since they are consist of heuristics and metaheuristics associated with actions carried out in the search space of the problem [4] [7]. The Coordinator Agent and the Solution Analyzer Agent form the Decision-Making Center. Note that the structure defines the types of agents that could be used, but does not limit how many and which will be implemented to solve a particular problem.

The general description of these components is presented as follow.

A. Environment

The general form of the Environment component, showed on Figure 2, emulates the basic model of a combinatorial optimization problem. It corresponds to the search space of the problem being dealt with. This means that all the relating

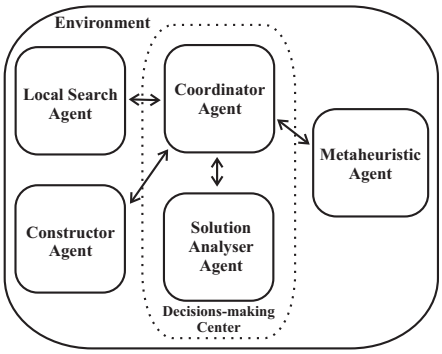

Fig. 1. Conceptual model of MAM.

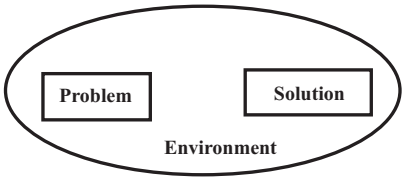

Fig. 2. Environment model.

characteristics of the problem, independent of the way it was modeled, are defined in the Environment. Thus, the change of the Environment component corresponds to changing the problem to be solved by the agents, or, from the other point view, changing the problem to be solved implies changing the Environment component. In consequence, computational effort for solving different problems is reduced to modeling their specific attributes and it is this component that makes MAM architecture flexibe.

The Environment component is made up of the Problem structure and the Solution structure. This form is defined considering that it is necessary to establish the way the problem solution will be represented when the optimization problem is modeled. In addition, all the essential information for searching solutions should be known and included in this component.

The Problem structure contains all specific attributes concerning the problem to be dealt with. The Solution structure contains the specifications of the different neighborhood types as well as how elements will be added by the constructive heuristics. This neighborhood types are some of the possibilities of interactions between the agents acting in MAM and the Environment component, allowing the construction of new solutions starting from the modification of the existent solutions. Since that the neighborhood structures are also specific for each problem model, they need to be defined as attributes of the solution.

The Environment component also mediates interactions between agents, characterizing a form of indirect communication. As an example, exchange of information can be viable by shared memories that allow agents to transfer and recover data through a blackboard-like system. The use of the Environment as the means of interaction between agents encourages flexible,

uncoupled and asynchronous communication between them.

B. Search Agents

The Search Agents are the Solution Constructor Agent, the Local Search Agent and the Metaheuristic Agents, as previously stated.

The Solution Constructor Agent is based on constructive heuristics. These heuristics are responsible for building the initial solutions, which are necessary to initiate the search process of both Local Search Agent and Metaheuristic Agent. For example, a greedy-like heuristic could be implemented as a Solution Constructor Agent. The Local Search Agent is the component responsible for refining the solutions previously found, using the local search algorithms. The Descent Method could be implemented, for example, as a Local Search Agent. The Metaheuristic Agent provides a better exploitation of the search space, which escapes from the local optimum by employing different strategies. The Metaheuristic Agent is composed of algorithms designated to find a "good" solution through by executing each iteration of a subordinate local search heuristic. Any metaheuristic, such as Iteration Local Search, Variable Neighborhood Search or Genetic Algorithm, among others, could be implemented as a Metaheuristic Agent.

C. Internal Structure of the Search Agents

The Search Agents presented above have the same generic internal structure, differing only in their methods of implementation. The existence of this generic structure enables the design of new search agents to be simplified. Figure 3 presents a diagram of the main internal components that are parts of this structure.

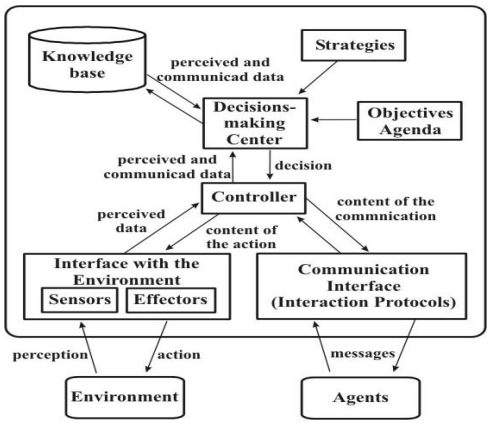

Fig. 3. Internal structure of Local Search Agent.

The Knowledge-base structure is responsible for keeping information about the Environment component, as well as the information necessary for establishing communication between agents and concerning other agents. Moreover, it is also responsible for storing the search history of the agent and describes the representation of the environment. The Communication Interface defines communication protocols between agents, allowing messages to be exchanged among them. Information exchange with the Environment component involves the action of sensors and effectors of the agents. The sensors enable the agent to perceive the Environment component and its changes, for example, the emergence of new solutions that may be useful in the search process of this agent. The effectors provide to the agents the ability to act and modify the Environment. Note that, in MAM architecture, a solution is considered as a stimulus for the agents.

Each Search Agent has its own Decisions-making center. This structure is in charge of providing the agent with the most appropriate way of carrying out the search based on the knowledge acquired by the agent as well as their objectives and strategies. The Objectives Agenda structure is a list of guidelines to be satisfied i.e., objectives and constraints obtained from the Environment component. The Strategies structure encapsulates the agent strategy of searching, that is to say, the algorithm of the implemented heuristic or the implemented metaheuristic. This structure is responsible for finding solutions to the problem represented in the Environment. Therefore, it facilitates the isolation of the problem in relation to the methods used and allows this general structure to be used in defining any search agent. Finally, the Controller structure performs intermediation and coordinates all the internal processes of the agents, such as the exchange of information among their internal components.

The *modus operandi* of a search agent can be summarized as follows: through the sensors, in its interface with the Environment, the agent captures samples of the Environment (information needed to solve the problem), while its effectors devise or modify solutions in the Environment. During this process, an agent can communicate with other agents through its own communication interface. In this way, an agent can obtain collaboration from other agents via information exchange. All perceived and communicated data are used to add to the Decision-Making ability of the agent. During the entire search process, the Decision-Making Center makes choices and decisions in an attempt to minimize (or maximize) the cost function of the problem (objective function).

D. Agents of the Decision-Making Center Component

MAM architecture uses the concept of mediator agent that centralizes and coordinates all the communication among agents under constant interaction.

The Coordinator Agent is the component responsible for being the communication intermediary among other agents, characterizing a management relationship and reducing the communication complexity. The Coordinator Agent coordinates activities and communications between agents and between themselves and the Environment component. It knows all the other agents of the architecture and when it receives a message with some service solicitation, it is responsible for guiding it to the agent that can execute the requested operation. Thus, this agent, besides exchanging information, is responsible for establishing negotiations, agreements and translations of messages among all the different agents, al-

lowing the process of cooperation between agents through an exchange of information, such as promising areas of the search space, unrecommended exploitation regions, guide solutions and search parameters. This information is used in the search process, with the objective of achieving the same purpose in a cooperative way, and characterizing the formation of a society of agents. Therefore, the Coordinator Agent acts as a mediator agent, making it possible to increase the autonomy of other agents and facilitating maintenance of the whole system.

The Solution Analyzer Agent is responsible for the action of analysis and decision-making process with respect to the solutions found by all Metaheuristic Agents, selecting the best solutions provided by them. This analysis can involve the use of different strategies to choose the best solutions.

III. Application of MAM Architecture to the VRPTW

This section shows the application of the MAM Architecture for solving the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW (see [8] for details) is a special class of the classical Vehicle Routing Problem (VRP), in which, besides determining a set of routes with minimum cost for serving a set of customers spread geographically, the attendance of the customers is carried out within a determined interval of time, called time window. An experiment using instantiations of MAM architecture for solving the VRPTW is carried out using the instances for this problem introduced in [9], well-known benchmark data for the VRPTW. The objective of the evaluation function used is to minimize the distance covered. At the same time, it penalizes violation of vehicle capacity constraints and the time window constraints.

A. Instantiations of MAM

Five instantiations of MAM architecture were developed for the computational tests. Each instantiation represents, in fact, a particular implementation of MAM architecture. Three of these instantiations are associated with the use of a particular metaheuristic: Variable Neighborhood Search (VNS) metaheuristic [10], Iterated Local Search (ILS) metaheuristic [11] and Genetic Algorithm [12]. The fourth instantiation includes the simultaneous application of three Metaheuristic Agents, each one associated with one of the previously cited metaheuristics. The fifth instantiation performs a test concerning the scale-effective property of the architecture. In this case, five Metaheuristic Agents, all with ILS metaheuristic, are implemented simultaneously. Due to space limitations, only a few details are provided here about the implementation of these agents.

Before presenting these instantiations, the Environment component must be defined. The Environment component for this case is presented in Figure 4, where the Problem structure has all the information about the customers (location, time windows, demand, among others) and the Solution structure is represented by a set of routes. The first instantiation of MAM architecture is made using Variable Neighborhood

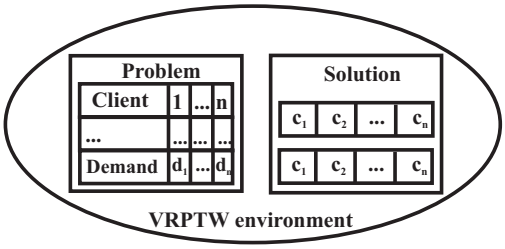

Fig. 4. Instantiation of MAM architecture for solving VRPTW: Environment component.

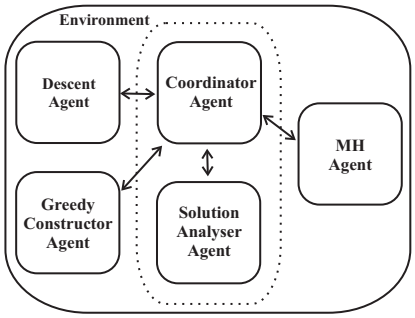

Fig. 5. Instantiation of MAM architecture used for VRPTW application.

Search (VNS) metaheuristic [10]. Figure 5 presents this instantiation, where the MH Agent represents the VNS Metaheuristic Agent. This MAM-VNS instantiation is formed by the VNS Metaheuristic Agent in conjunction with the Greedy Heuristic, based on time windows, implemented as a Constructor Agent, and the Local Search Agent, implementing the Descent Method [7]. The VNS metaheuristic consists of a local search exploitation of the search space of the problem, carrying through systematic exchanges of the neighborhood function as a strategy for escaping of local optimum. At each iteration, a local search is performed in the current solution and the result is compared with the proper current solution. If no improvement is found, the neighborhood function is changed. For this alteration an adaptation of the Push-Forward Insertion Heuristic (PFIH) [9] is used in which a route R_q is randomly chosen and, in the sequel, K clients are removed from this route, for $1 \leq K \leq N$, being N the number of clients belonging to this route and K defined by the neighborhood structure of VNS. The removed clients are then reinserted in the current solution, choosing the best insertion, in this current solution for each client removed in previous step.

The second instantiation uses MAM architecture with the Metaheuristic Agent performing the Iterated Local Search (ILS) metaheuristic [11]. As well as the previous instantiation, Figure 5 shows it, with the ILS Metaheuristic Agent represented by the MH Agent. This MAM-ILS instantiation is also formed, besides the ILS Metaheuristic Agent, by the Constructor Agent (with the Greedy Heuristics based on time windows) and the Local Search Agent, implementing the Descent Method [7]. The ILS metaheuristic builds solutions

iteratively, performing perturbations at existing solutions in order to diversify the search. A local search is applied to the solution found from the perturbation process and then a new solution is obtained, which is a local optimum in the current neighborhood. The process is repeated until a stopping criterion is satisfied. At each iteration, the best local optimum is saved as the best solution of the problem. An adaptation of the PFIH algorithm [9] is used for the process of changing the neighborhood, similar to that used for the VNS.

The third instantiation of the MAM architecture concerns Genetic Algorithm (GA). This metaheuristic consists of applying genetic operators of crossover, mutation and reproduction to an initial population until a stopping criterion for ending the evolution is satisfied. This MAM-GA instantiation is also formed, like the others Metaheuristic Agent implemented, besides the ILS Metaheuristic Agent, by the Constructor Agent (with the Greedy Heuristics based on time windows) and the Local Search Agent, implementing the Descent Method [7]. The initial population generated by the Constructor Agent is evaluated by the Solution Analyzer Agent, which selects the survivor population. Following, at each iteration, new individuals are generated using the operations of crossover and mutation, and again evaluated by the Solution Analyzer Agent, producing a new population, and the process is repeated. The reproduction operator is used for choosing the most suited individuals to pass their characteristics to new population. The GA implemented here uses, for the crossover operator, an adaptation of the Best Coast Route Crossover (BCRC), described in [13]. The mutation operator is an adaptation of the PFIH algorithm [9]. A randomly chosen route is removed from the individual and, next reinserted in the same individual. The objetive of this operator is to increase the diversity of the population.

The fourth and fifth instantiations use the Metaheuristic Agents discussed above, acting jointly. The fourth instantiation joins the three Metaheuristic Agents, now acting simultaneously. The fifth is mounted with up to five replications of the ILS Metaheuristic Agent. All instantiations using more than one Metaheuristic Agent were developed using the communication system called blackboard [14]. With this approach, a type of repository (blackboard), where the agents can read and write messages, is used as a means of interaction. In this case, the repository of solutions is located in the Environment, and the Environment helps in the cooperation among the agents. Direct communication among the agents is analogous to the instantiations presented above. The formulation of this repository, in the present paper, is similar to a tabu list, in which the unrecommended solutions are stored in order to be avoided by the agents. The access of the agents to the repository is constant, carried out each interaction and it is executed before the local search. The similarity among the used methods favors cooperation among the agents.

B. Computational results for the instantiations

MAM architecture is implemented on Java Plataform, version J2SE 5.0, using the IDE Elipse, version 3.4, running on Windows XP SP2. The stopping criterion adopted is the maximum number of iterations, defined as 1.500 iterations for each Metaheuristic Agent. As stated earlier, the instances from [9] are used for these tests with the instantiation of MAM defined above.

TABLE I
RESULTS FOR THE INDIVIDUAL APPLICATION OF GA, ILS AND VNS AGENTS.

Inst.	GA	ILS	VNS	Best known
C101	827.30 (10)	**827.29 (10)**	999,99 (12)	828.94 (10)
C201	**590.10 (3)**	626.99 (4)	668,99 (5)	591.56 (3)
R101	1662.39 (18)	1693.49 (20)	1854,30 (21)	1650.80 (19)
R201	**1198.39 (5)**	1214.39 (5)	1321,39 (6)	1253.26 (4)
RC101	**1656.80 (17)**	1773.89 (16)	2037,70 (19)	1696.94 (14)
RC201	**1303.00 (7)**	1500.99 (5)	1455,99 (7)	1406.94 (4)

The results for the first, second and third instantiations are showed in Table I. For saving space, only the first instance of each classe of Solomon's instances are presented. The number in parentheses represents the number of vehicles. On this table, the first column represents the instance and the column "Best known" shows the best known published results in the literature. This last column is from [13]. It is concluded from this table that the results are competitive with the best ones provided in the literature. The Metaheuristic Agent with best behaviour is GA-Agent, followed by ILS-Agent. An observation must be made concerning the quality of the shown results. For instance R101, the GA-Agent finds a better result for the number of vehicles than the best known result; on the other hand, for instances R201, RC101 e RC201, the results for the distance score obtained by the GA-Agent are better than the best known published.

Table II shows the simultaneous application of three Metaheuristic Agents. This result is important, since it shows that the cooperation between agents with different metaheuristics leads to better results than the isolated action of one agent. In all instances under evaluation, this instantiation obtains best results than the best known result published, when it analyzes only the distance. Finally, Table III refers to the application of up to five replications of ILS Metaheuristic Agents simultaneously. It is shown that the inclusion of a larger number of equal agents allows results to be the improved, but, comparing to the case shown on Table II, in which different Metaheuristic Agents act, the result is worse. Moreover, there is a variation of the results with the number of ILS-Agents. No scale-effective property can be inferred from these data.

TABLE II
RESULTS FOR THE SIMULTANEOUS APPLICATION OF GA, ILS AND VNS AGENTS.

Inst.	GA-ILS-VNS	Best known
C101	**827.29** (10)	828.94 (10)
C201	**589.09** (3)	591.56 (3)
R101	**1646.40**(19)	1650.80 (19)
R201	**1196.40** (5)	1252.26 (4)
RC101	**1684.50** (15)	1696.94 (14)
RC201	**1465.09** (4)	1406.94 (4)

TABLE III
RESULTS FOR SIMULTANEOUS APPLICATION OF UP TO 5 ILS AGENTS.

Inst.	1-ILS	2-ILS	3-ILS	4-ILS	5-ILS	GA-ILS-VNS	Best known
C101	**827.29 (10)**	**827.29 (10)**	**827.29 (10)**	**827.29 (10)**	**827.29 (10)**	**827.29 (10)**	828.94 (10)
C201	626.99 (4)	710.29 (3)	589.10 (3)	589.10 (3)	**589.09 (3)**	**589.09 (3)**	591.56 (3)
R101	1693.49 (20)	1693.30 (19)	1702.89 (20)	1663.50 (19)	1670.79 (18)	**1646.40 (19)**	1650.80 (19)
R201	1214.39 (5)	1280.99 (4)	1198.70 (5)	1272.19 (4)	1206.99 (5)	**1196.40 (5)**	1253.26 (4)
RC101	1773.89 (16)	1729.29 (16)	1698.39 (16)	1733.50 (16)	**1675.39 (16)**	1684.50 (15)	1696.94 (14)
RC201	1500.99 (5)	**1323.50 (5)**	1360.00 (5)	1350.50 (5)	1465.09 (4)	1347.90 (5)	1406.94 (4)

IV. CONCLUSION

In this paper, it is presented MAM architecture. It is a multiagent architecture for the solution of combinatorial optimization problems through the use of metaheuristics. In this proposal, the metaheuristics are defined as agents that act in the exploitation of the search space. Moreover, the agents are able to communicate with one another and with its Environment, facilitating cooperative behavior which tends to achieve a common objective. MAM architecture defines a framework, which allows several metaheuristics to be used to cooperatively and flexibly solve a certain problem. The addition of new agents (i.e, new metaheuristics) is facilitated, since the architecture is defined as a framework for metaheuristics. The experimental evaluation of the architecture was performed using five instantiations for solving the VRPTW, demonstrating how these characteristics are presented. Finally, some possibilities of continuity concerning the development of MAM architecture include the addition of graphic facilities; improvement of the cooperative search capacity; inclusion of intensification and diversification methods in the architecture; and extending it in order to enable parallel computation and/or distributed computation resources.

ACKNOWLEDGMENT

The authors would like to thank CEFET/MG and CAPES for the support in the development of this work.

REFERENCES

[1] M. Wooldridge, *An Introduction to Multiagent Systems*, 1st ed. John Wiley & Sons, 2002.

[2] S. Talukdar, L. Baerentzen, A. Gove, and P. S. de Souza, "Asynchronous teams: Cooperation schemes for autonomous agents," *Journal of Heuristics*, vol. 4, no. 4, pp. 295–321, 1998.

[3] S. Thangiah, O. Shmygelska, and W. Mennell, "An agent architecture for vehicle routing problems," in *Proceedings of the 2001 ACM Symposium on Applied Computing*, Las Vegas, Nevada, EUA, 2001, pp. 517–521.

[4] M. Milano and A. Roli, "Magma: a multiagent architecture for metaheuristics," *IEEE Transaction on Systems Man and Cybernetics*, vol. 34, no. 2, 2004.

[5] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Meta-heuristics*, F. Glover and G. Kochenberger, Eds. Dordrecht, The Netherlands: Kluwer, 2003, pp. 457–474.

[6] A. L. Bouthillier, T. G. Crainic, and P. Kropf, "Towards a guide cooperative search," *MIC2005: The Sixth Metaheuristics International conference, 1227-1*, 2005.

[7] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[8] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Philadelphia: SIAM - Society for Industrial and Applied Mathematics, 2002.

[9] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 2, no. 35, pp. 254–264, 1987.

[10] P. Hansen, N. Mladenovic, and J. A. Moreno, "Variable neighbourhood search," *Revista Iberoamericana de Inteligencia Artificial*, no. 19, pp. 77–92, 2003.

[11] H. R. Lourenço, O. Martin, and T. Stützle, "Iterated local search," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science, F. Glover and G. Kochenberger, Eds. Norwell, MA, EUA: Kluwer Academic Publishers, 2002, vol. 57, pp. 321–353.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison Wesley, 1989.

[13] B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objetive generic algorithm for vehicle routing problem with time windows," *Applied Inteligence*, vol. 24, no. 1, pp. 17–30, 2006.

[14] D. D. Corkill, "Blackboard systems," *AI Expert*, vol. 9, no. 6, pp. 40–47, 1991.