An Agent-Based Metaheuristic Approach applied to the Vehicle Routing Problem with Time-Windows

Maria Amélia Lopes Silva*, Sérgio Ricardo de Souza*, Sabrina Moreira de Oliveira†, Marcone Jamilson Freitas Souza‡
*CEFET-MG – Centro Federal de Educação Tecnológica de Minas Gerais, Av. Amazonas, 7675 Nova Gameleira
CEP:30510-000, Belo Horizonte, MG, Brazil, Email:mamelia@dppg.cefetmg.br, sergio@dppg.cefetmg.br
†IBMEC - Instituto Brasileiro de Mercado de Capitais, Rua Rio Grande do Norte, 300 Funcionrios
CEP:30130-130, Belo Horizonte, MG, Brazil, Email: sabrinaoliveira@ibmecmg.com
‡UFOP - Universidade Federal de Ouro Preto, Departamento de Computação, ICEB
35400-000, Campus Universitário (UFOP), Ouro Preto, MG, Brazil, Email: marcone@iceb.ufop.br

***Abstract*—In this paper, we present a framework, based on parallel cooperative approach and concepts of Multi-agent Systems, for solving the Vehicle Routing Problems with Time Windows. We propose an effective way of applying metaheuristics as agents, as autonomous as possible. The framework proposed here creates an environment where interactions between metaheuristics are carried out. As a consequence, these agents present a cooperative behavior and exchange information during the run. Two metaheuristic agents (Iterated Local Search and Genetic Algorithm) were implemented. The results show the cooperative behavior of these metaheuristics and the influence of this behavior on improving the quality of the solutions, once these agents have achieved better performance on the tackled problem, when compared to the execution of each metaheuristic agent separately.**

***Keywords*-Framework, Multi-agent, Metaheuristics, Vehicle Routing Problem.**

I. Introduction

In the latest years, the number of researches applying two or more metaheuristics, at same time, to improve solution quality of algorithms has increased significantly [1], [2]. This strategy is known as hybridization of metaheuristics. Its advantage arises from the fact that strengths of each metaheuristic are applied together. As a consequence, besides obtaining better solution quality in a shorter amount of time, its ability to deal with more complex problems increases.

Many ways of metaheuristic hybridization can be found in the literature. Here, we emphasize the parallel cooperative approach. Its importance lies in the fact that it adds, to the applied metaheuristics, parallel computational resources and possibility of information exchange. Some examples can be cited, like a hybrid cooperative model applied to the Tool Switching Problem [3], parallel metaheuristics for solving Dynamic Optimization Problems [4], the Capacitated Vehicle Routing Problem [5] and the Traveling Salesman Problem [6].

In this paper we apply the parallel cooperative approach, managed as Multi-agent Systems (MAS) [7]. MAS can be used as a liaison between different metaheuristics for solving optimization problems. The advantages of using MAS is to refine and combine the solutions built by the metaheuristics. Each agent is responsible for performing its own task and, at same time, for using the solutions provided by other agents to improve their own solutions. In this approach, agents interact and work together to achieve a pre-defined objective.

A number of papers have been proposed to show the application of cooperative approach and multi-agent systems, each one presenting different ways and advantages to deal with the addressed problem [5], [6], [8]–[10]. Some of them are highlighted below.

The MAGMA architecture [8] proposes a conceptual framework applying a multi-agent perspective, where a metaheuristic can be seen as a consequence of the interaction between different agents. These agents work according to pre-defined hierarchical levels, each one corresponding to a metaheuristic action. A Multi-agent architecture for metaheuristics (AMAM) is proposed in [9]. In this paper, each metaheuristic is implemented as an agent. The problem search space is represented as an environment in which agents play and interact to each other. Every time a different problem has to be tackled, just the architecture environment has to be changed. This cooperative approach is also used in [5] for solving the Capacitated Vehicle Routing Problem. This proposal applies an asynchronous information exchange between multiple cooperative tabu search threads. Another way to coordinate agent activities in multi-agents architectures by means of metaheuristics is through the use of algorithms inspired by collective intelligence [6], [10]. In these papers, the performance of the Particle Swarm Optimization algorithm is measured according to hybridization and cooperation of algorithms.

However, according to the literature review, two questions arise. Which one is the best way to coordinate the metaheuristics when the objective is to explore their potential of hybridization? At the same time, how could be developed a more robust structure able to deal with different problems with minimum changes?

Our objective is to cover this lack in the literature, presenting a framework that uses a multi-agent approach

for combining metaheuristics. Each agent acts as a specific metaheuristic, which works simultaneously, but independently, during the run. In this paper, we have implemented two metaheuristics as agents: Iterated Local Search – ILS [11] and Genetic Algorithm – GA [12]. As they have different ways to explore and exploit the search space, they are able to help each other simultaneously to improve the solution quality of the problem addressed in this paper, which is the Vehicle Routing Problem with Time Windows (VRPTW). This approach guides the search throughout promising areas by the exchange of information during the search space exploration.

The rest of this paper is organized as follows. Section II describes the problem under consideration. Section III introduces the proposed approach. Computational results are presented in Section IV. Finally, some conclusions and discussions are showed in Section V.

II. The Vehicle Routing Problem with Time Windows

The problem to be dealt with the MAS approach is the Vehicle Routing Problem with Time Windows (VRPTW). In this problem there is a depot, a set of costumers and a set of vehicles. Each costumer has a demand to be attended by a vehicle into a time window. Each vehicle should arrive before the opening of the time window and wait until the service becomes available, but it is not permitted to arrive after its ending. The routes must start and end at a depot. The objective of this problem is to achieve a minimum number of routes that are able to attend a set of customers with minimum distance respecting their time windows. Figure 1 illustrates a solution example involving a depot, seven clients and three routes.

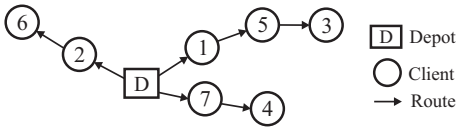

Figure 1. A solution of VRPTW.

A solution for the VRPTW is computationally represented by a vector of positions. Each position represents a route and each route is represented by another vector, with the identification of the customer attendance order. The sequence of customers to be attended is defined by the position that each one has at the route vector. Figure 2 illustrates its representation. In this figure, for example, the Route 2 contains the clients 2 and 6.

The solution's quality is given by (i) the number of vehicles used and, after that, (ii) the total distance travelled.

III. The proposed approach

The framework we have proposed here is based on the multi-agent approach and it is presented in Figure 3. It is inspired by the work presented in [9]. In this latter

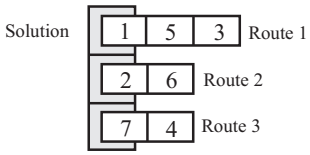

Figure 2. Computational representation of a Solution.

work, a multi-agent conceptual model was proposed by the application of several metaheuristics at one generalized structure. Each agent represents a metaheuristic, responsible to build and improve solutions for a specific problem. In the search solution process, the agents must scroll through the multi-agent system environment. In this case, the multi-agent system environment is defined by the search space of the tackled problem, seeking a predetermined goal. The ability to move the agent in the search space of the problem is defined by the neighborhood structures (movements) that he has. The agents can, then, move through the environment, from one solution to another one.

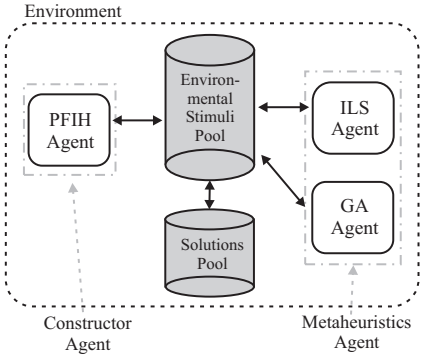

Figure 3. Multi-agent Framework for VRPTW.

In the current proposal, our multi-agent framework is composed by four main elements, as shown in Figure 3: (i) Environmental Stimulus Pool; (ii) Pool of Solutions; (iii) Constructor Agent; and (iv) Metaheuristic Agents.

The Constructor Agent is the element responsible for building the initial solutions for the problem. These solutions are improved by metaheuristic agents separately and simultaneously. Solutions are stored in the Pool of Solutions. One important characteristic of this structure is the autonomy of the agents: one agent should not directly interfere in the decisions of other agents of the structure. This is the main difference between this current proposal and the one presented in [9]. In order to guarantee this autonomy, from the initial proposal we have removed the coordinator agent, as well as the solution analysis agent. Both are responsible for intermediating communication and making decisions inside the framework; however, they limit the autonomy of the agents in the search space. They have been replaced by an Environmental Stimulus Pool, which is responsible for

managing all information exchange between agents, without any interference with concern to their autonomy.

Any information exchange concerning the search space occurs by means of stimulus. A stimulus can be a provision or a request of a solution. The available solutions to be requested are stored in the Pool of Solutions. However, only each metaheuristic agent can decide when is necessary to include or to remove these solutions.

In this paper, we have implemented a construction agent and two metaheuristic agents to solve the VRPTW. Note that each metaheuristic agent plays on its own thread, both executed in parallel. In this way, each metaheuristic agent works independently at the same time. The time spent to improve the solution's quality is, consequently, decreased.

A. Constructor agent

The constructor agent is the element responsible for building the initial solution that is going to be improved by metaheuristic agents. The initial solution is built by a variation of the Push-Forward Insertion Heuristic – PFIH [13]. This heuristic is largely applied to problems that deal with customers/tasks allocation with time windows constraints. More specifically, the PFIH defines to each customer a solution insertion cost related to its final time window, distance and polar angle between it and the depot. This insertion cost defines the attendance order of inserted customers at the solution.

We have added a probabilistic component, which allows the diversification of the PFIH initial solutions. Thus, customers are ordered according to their cost of insertion, as proposed by Solomon [13]. The customers with lowest insertion cost are inserted in a restrict candidate list, as the one used at construct phase of GRASP (Greedy Randomized Adaptive Search Procedures) metaheuristic [14]. Nevertheless, at each iteration, the choice of next customer to be attended is randomly taken from this set. Solutions that have been built here are stored in the Environmental Stimulus Pool, where they are available to be used by other metaheuristic agents.

B. Metaheuristic agents

The ILS and GA metaheuristics work as metaheuristic agents. They are responsible for improving the solution's quality by the use of strategies specifically used to escape from local optimum traps .

1) ILS Agent: The ILS agent is an implementation of ILS metaheuristic [11]. It mainly consists in applying perturbations in the local optimum solutions and refine these perturbed solutions.

The ILS agent has been implemented using different levels of perturbations. These levels consist in different perturbation techniques applied after a number of pre-defined iterations without any solution's quality improvement. The perturbation levels are applied in an ascending order and according to their degree of intensity, that is, the number of modifications in the current solution. Six levels of perturbation have been implemented: (i) the random exchange of two consecutive customers; (ii) the relocation of two random customers; (iii) the relocation of three consecutive customers; (iv) the removal of the shortest route; (v) the removal of a random route; and (vi) the random choice of a solution in the pool.

The interaction between the ILS agent and the environment can happen as follows: (i) an initial solution is received; (ii) a solution from the perturbation method is received; and (iii) a complete solution at the end of an iteration is added, only if it is the best so far solution.

2) GA Agent: The GA agent represents the GA metaheuristic [12]. The solutions are converted to chromosomes, i.e., an individual in the population, as shown in Figure 4. Because a solution for the VRPTW is defined as a set of routes, when GA is applied it is represented by an individual. In addition, each gene represents a customer to be attended.

An Individual

0	1	5	3	0	2	6	0	7	4	0

Figure 4. A GA solution representation to the solution presented at Figure 1.

Similarly to the ILS agent, all initial solutions need to be built using the Constructor Agent. After that, a crossover operator is applied to generate an offspring. Characteristics like the order of customers attendance, number of routes, and number of customers per route are transferred to next individuals. Later, a mutation procedure may be applied following a probabilistic rule. It consists in the elimination of the shortest route from selected individuals from the offspring. Finally, a new population is generated. Part of it is formed by parents and descendants from the old population, which are chosen by applying a binary tournament procedure. The remaining individuals of the new population are taken from the other agents at the framework. They are randomly taken from the pool of solutions.

C. Multi-agent Environment

The multi-agent environment is mainly defined by the search space of the tackled problem. Therefore, it provides all information that is needed for solving the problem eg.: number of customers to be attended, distance between customers, number of vehicles, and so on. Besides that, the multi-agent environment is the element responsible for allowing the construction of new solutions, their modification or combination. It is composed by following elements: (*i*) Problem data: the element responsible for storing specific information of the tackled problem; (*ii*) Stimulus Pool: it stores the stimulus sent by agents, that can be the delivery of solutions that have been built or the request of new ones. It is the way of interaction and information exchange between

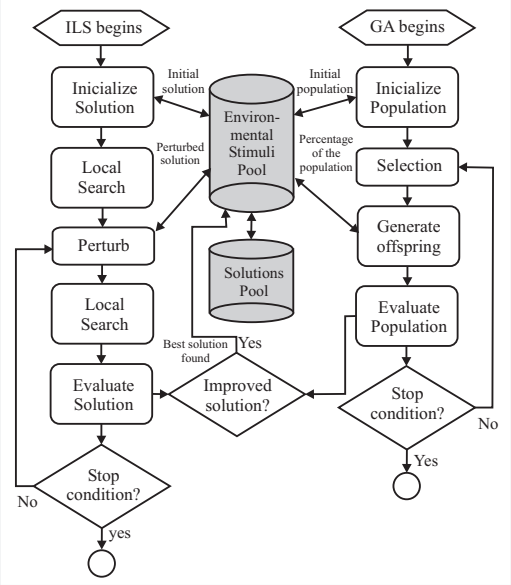

Figure 5. Environment for VRPTW.

agents; (*iii*) Pool of solutions: it is a vector responsible for storing solutions generated by agents.

D. Information Exchange and Cooperation

The cooperation between agents takes place by the exchange of information from the search space. After each iteration the pool of solutions is updated with the solution obtained for each agent.

The main objective of this cooperative structure is to guide agents through the search space, by diversifying the search and, at the same time, guiding the exploitation to possible promising search areas in a shorter computation time.

In order to guarantee the autonomy of each agent, all information exchange is mediated by stimulus exchange, through the environmental Stimulus Pool. Whenever an agent builds a solution or needs to request one (like the requirement of an initial solution), it sends a stimulus to the Environmental Stimulus Pool. Figure 6 shows the interaction procedure between agents in the framework. The maximum

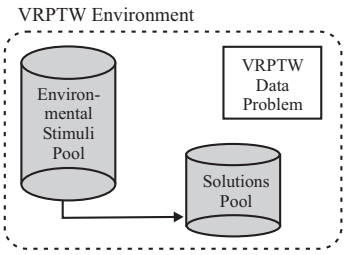

Figure 6. Cooperation between agents.

size of the Pool of Solutions is static. Firstly, every time a new solution is generated, it is inserted in the Pool of Solutions. Whenever the pool is full, the insertion of a new solution is managed by a function based on the "distance" between solutions. The solution with the worst solution's quality is replaced by a new one.

The evaluation function is given by the sum of distances from a solution i to all other solutions at the Pool, as Eq. (2):

$$s_i = \sum_{j=1}^{N} s(d_{ij}) \tag{1}$$

where d_{ij} is the distance between two solutions i and j from the Pool of Solutions. As smaller as the distance between i and j, the biggest is the similarity of their solutions. Thus, we consider a ratio r as the limit of closeness of the solutions to be accepted. This ratio is calculated by:

$$s(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{r} & d_{ij} \leq r \\ 0 & d_{ij} > r \end{cases} \tag{2}$$

The main objective of Eq. (2) is to keep the diversity of solutions in the Environmental Stimulus Pool. At the same time, the best so far solution from the Pool is always stored with a specific environment attribute and updated at each solution's quality insertion. This procedure avoids to remove the best so far solution when the above procedure is applied.

IV. Computational results

In this section we describe all instances, algorithms and computational experiments. The proposed framework here was implemented in Java, with JDK 1.6. Each agent runs in its own thread. The elements of the environment are shared between the agents. In order to guarantee this sharing, the environment is passed through reference to agents.

The results were obtained using an PC with an Intel i7-4500U processor, 1.8 GHz, 16 GB of DDR3 RAM, running Windows 8 operational system. Table I shows the used parameter values in the experiments. All used instances were

Table I
Parameter Values

Parameters	**Setup**
Population size (GA)	20 individuals
Number of generations (GA)	100
Crossover rate (GA)	0.8
Mutation rate (GA)	0.05
Maximum number of iterations (ILS)	500
Perturbation levels (ILS)	6
Maximum number of iterations without improvement (ILS)	5% of the maximum number of iterations
RCL size (PFIH)	20% of the total number of customers
Maximum size of the pool of solutions	Number of clients.

proposed by [13]. They are separated into 3 different sets (C, R and RC) according to geographical distribution of the clients. From the 56 available instances, we have chosen 8 from each group, all of them with 100 clients.

In order to evaluate the multi-agent framework, we have compared our approach with ILS and GA metaheuristics

executed separately. In this way, the evaluation of cooperation between agents and how strong are their cooperation on the solution's quality is also possible. Nevertheless, besides comparing the solution's quality, the number of access of each metaheuristic to the pool of solutions during the run is also measured. As already mentioned, the solution's quality is obtained according to the total distance traveled and to the number of generated routes. Therefore, both of them are taken into account in the following.

The obtained results are shown in Tables II, III and IV. Each table presents achieved results from one instance of each set (C, R and RC). The first column describes the framework scenario. The second column (BKS) presents the best known value in the literature and its respective number of routes/vehicles (V) and traveled distance (D). The column "Results" presents the average values found in 30 executions of each algorithm (AD); their respective number of vehicles (V); the best value found (BS) and the associated number of vehicles (V).

Table II
RESULTS FROM THE THREE TESTED SCENARIOS ON INSTANCE C101

Instance: C101						
Scenario	**BKS**		**Results**			
	D	**V**	**AD**	**V**	**BS**	**V**
ILS	827.29	10	935.16	10.9	827.29	10
AG	827.29	10	977.5	12.3	958.09	11
ILS+AG	827.29	10	869.83	10.2	827.29	10

Table III
RESULTS FROM THE THREE TESTED SCENARIOS ON INSTANCE R101

Instance: R101						
Scenario	**BKS**		**Results**			
	D	**V**	**AD**	**V**	**BS**	**V**
ILS	1650.8	19	1722	20	1674.79	19
AG	1650.8	19	1745.7	21	1717.6	20
ILS+AG	1650.8	19	1675.3	18.6	1662.79	18

Table IV
RESULTS FROM THE THREE TESTED SCENARIOS ON INSTANCE RC101

Instance: RC101						
Scenario	**BKS**		**Results**			
	D	**V**	**AD**	**V**	**BS**	**V**
ILS	1696.94	14	1813.7	17.8	1688.9	17
AG	1696.94	14	1948.6	19	1757	18
ILS+AG	1696.94	14	1720.6	16.1	1674.9	15

According to the tables above, we can see that the best values are achieved by the multi-agent approach (ILS+AG), for all used instances. This better performance of the multi-agent approach, in comparison to GA and ILS working separately, is also confirmed by the execution of the ANOVA test (see the box plots presented in Figures 7 and 12).

For the Vehicle Routing Problem, the cost generated by the number of vehicles is the first objective to be satisfied. This objective is harder to improve than total distance travelled. With concern to our approach, it is also minimized, as shown in Figures 7, 9 and 11.

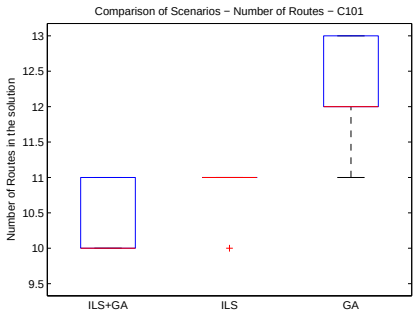

Figure 7. Boxplot regarding the number of used vehicles on instance C101.

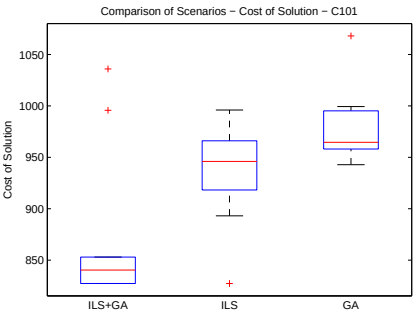

Figure 8. Boxplot regarding the obtained solution values on instance C101.

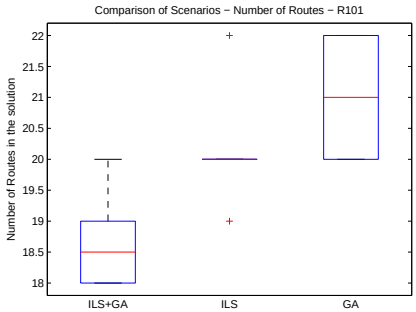

Figure 9. Boxplot regarding the number of used vehicles on instance R101.

V. CONCLUSIONS AND PERSPECTIVES

In this paper, a multi-agent framework for solving the Vehicle Routing Problems with Time Windows was presented. The proposed approach implements each metaheuristic as a parallel cooperative autonomous agent. Each agent is responsible for building solutions to the tackled problem at the same time. In addition, they can communicate to each other by their ability of sending and receiving stimulus to/from the environment, which generates a parallel cooperative behavior.

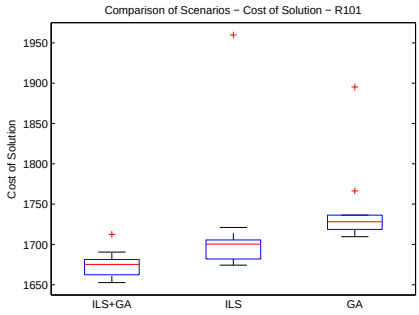

Figure 10. Boxplot regarding the obtained solution values on instance R101.

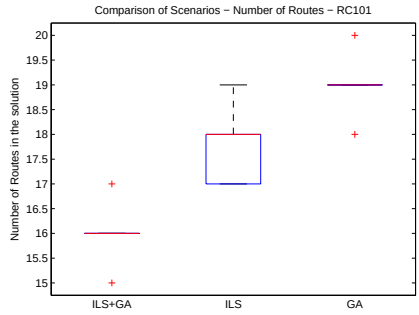

Figure 11. Boxplot regarding the number of used vehicles on instance RC101.

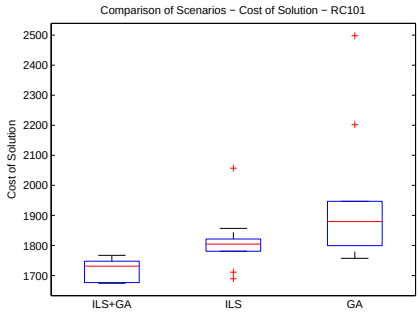

Figure 12. Boxplot regarding the obtained solution values on instance RC101.

The pool of solutions is responsible for storing solutions sent throughout the stimulus. The number of solutions inside the pool is managed by an evaluation function. This function is also responsible for diversifying the solutions inside the pool. From the presented results, the ability of agent cooperation and its influence in the solution's quality is confirmed. The use of two metaheuristic agents (ILS and GA) allowed the reduction of both costs, that is, the number of routes and total traveled distance.

Acknowledgment

The authors would like to thank CEFET/MG, UFV, CAPES, CNPq and FAPEMIG for supporting this research.

References

[1] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Applied Soft Computing*, vol. 11, no. 6, pp. 4135–4151, 2011.

[2] C. Cotta, E. g. Talbi, and E. Alba, "Parallel hybrid metaheuristics," in *Parallel Metaheuristics, a New Class of Algorithms*, E. Alba, Ed. John Wiley, 2005, pp. 347–370.

[3] J. E. Amaya, C. Cotta, and A. J. F. Leiva, "Hybrid cooperation models for the tool switching problem," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, ser. Studies in Computational Intelligence, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds. Springer Berlin Heidelberg, 2010, vol. 284, pp. 39–52.

[4] M. R. Khouadjia, E. g. Talbi Talbi, L. Jourdan, B. Sarasola, and E. Alba, "Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems," *The Journal of Supercomputing*, vol. 63, no. 3, pp. 836–853, 2013.

[5] J. Jin, T. G. Crainic, and A. Lkketangen, "A cooperative parallel metaheuristic for the capacitated vehicle routing problem," *Computers & Operations Research*, vol. 44, pp. 33–41, 2014.

[6] G. R. Souza, E. F. G. Goldbarg, M. C. Goldbarg, and A. M. P. Canuto, "A multiagent approach for metaheuristics hybridization applied to the traveling salesman problem," in *2012 Brazilian Symposium on Neural Networks (SBRN)*, Oct 2012, pp. 208–213.

[7] M. Wooldridge, *An Introduction to Multiagent Systems*, 2nd ed. Chichester, UK: Wiley, 2009.

[8] M. Milano and A. Roli, "MAGMA: a multiagent architecture for metaheuristics," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 925–941, 2004.

[9] F. C. Fernandes, S. R. de Souza, M. A. L. Silva, H. E. Borges, and F. F. Ribeiro, "A multiagent architecture for solving combinatorial optimization problems through metaheuristics," in *Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics – SMC 2009*, 2009, pp. 3071–3076.

[10] M. E. Aydin, "Collaboration of heterogenous metaheuristic agents," in *2010 Fifth International Conference on Digital Information Management (ICDIM)*, July 2010, pp. 540–545.

[11] H. R. Lourenço, O. Martin, and T. Stützle, "A beginner's introduction to iterated local search," in *Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, Porto, Portugal, 2001, pp. 1–11.

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

[13] 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.

[14] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 9, pp. 849–859, 1995.