# A Multi-agent Metaheuristic Optimization Framework with Cooperation

Maria Amélia Lopes Silva*, Sérgio Ricardo de Souza†, Marcone Jamilson Freitas Souza‡, Sabrina Moreira de Oliveira§
* UFV – Universidade Federal de Viçosa - campus Florestal
Rodovia LMG 818, km 6, 35690-000, Florestal, MG, Brazil Email: mamelia@ufv.br
† CEFET-MG – Centro Federal de Educação Tecnológica de Minas Gerais, Av. Amazonas, 7675 Nova Gameleira
30510-000, Belo Horizonte, MG, Brazil, Email:sergio@dppg.cefetmg.br
‡ UFOP - Universidade Federal de Ouro Preto, Departamento de Computação, ICEB
35400-000, Campus Universitário (UFOP), Ouro Preto, MG, Brazil, Email: marcone@iceb.ufop.br
§ Ibmec Minas Gerais R. Rio Grande do Norte, 300
30130-130, Belo Horizonte, MG, Brazil Email: sabrina.oliveira@ibmecmg.br

***Abstract*—This article address a Multiagent Metaheuristic Optimization Framework. In this proposal, each agent acts independently in the search space of a combinatorial optimization problem. The Framework allows the simultaneous execution of various agents, in a cooperative way. The coalition concept of cooperation is adopted. The agents have auto-learning abilities, based on reinforcement learning. The ability of cooperation and its influence on the quality of solutions provided by the agents are confirmed by performed experiments. In addition, experiments show that influence is greater when the number of agents is increased.**

***Keywords*—Multi-agent Systems; Frameworks; Combinatorial Optimization; Metaheuristics.**

## I. Introduction

Multi-agent approach has been applied to the processing of combinatorial optimization problems in several works. The strong economic relevance and impact in real life generated by many of these issues justifies the growing interest in the application of new techniques for their solution. Thus, the search for better ways of solution becomes important, especially in developing techniques that lead to flexibility in incorporating new methods without requiring the effort to remake its implementation.

This article aims to introduce a Multi-Agent Framework for Optimization using Metaheuristics. This proposed framework encapsulates metaheuristics in agents which cooperate in order to solve optimization problems.

Several frameworks for metaheuristics can be found in the literature [1]–[6],most of them with similar characteristics and proposals. OptFrame is a framework proposed in [4]. Its main characteristic is an interface for common elements of population-based metaheuristics and of trajectory-based metaheuristics. jMetal is a object-oriented framework based on the Java language [5]. It includes a significant number of classic and modern methods for multi-objective optimization problems and a wide range of issues instances. ParadisEO is a global framework composed by four connected modules [6]–[8]. These modules treat population and trajectory metaheuristics, multi-objective evolutionary techniques as well as parallel and distributed implementations. In [9], [10] are used Asynchronous Teams (A-Teams). In this structure there is a set of autonomous agents that communicate through shared memories. A bibliographical review and a comparative study with major available frameworks can be found in [11].

According to [11], the existing frameworks do not exploit the benefits of combining metaheuristic methods. In consequence, hybridization or any kind of interaction between the involved metaheuristics are not important characteristics of these existing frameworks. Thus, we propose in this article a structure that will favor the interaction between metaheuristics in a framework, in order to facilitate hybridization and to allow the development of generic structures, regardless of which metaheuristics are used as well as the problem to be treated.

Like other frameworks available in the literature, this proposal presents common features such as: (i) metaheuristics are pre-implemented to test and reuse; (ii) support the evaluation and comparison of different methods; (iii) ease in the development of a particular metaheuristic and their suitability to the treated problem. In addition to these features, the framework presented here has the strength of hybridization of metaheuristics through the parallel cooperative approach managed by Multi-Agent Systems (MAS). MAS are used here as a liaison between different metaheuristics for solving optimization problems. 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.

The agents have self-adaptive capabilities based on reinforcement learning, from which modify their actions based on their experience of interaction with the environment.

The performed experiments use the concept of cooperation by coalition, in which identical agents are instantiated and cooperate to resolve the addressed issue. As an example, in this article the framework is applied for solving the Vehicle Routing Problem with Time Windows (VRPTW).

The remainder of this article is organized as follows. Section II shows concepts related to metaheuristic hybridization,

978-1-5090-0016-6/15 $31.00 © 2015 IEEE
DOI 10.1109/BRACIS.2015.64

focusing on cooperatives and parallel approaches. Section III describes the proposed framework and its main components. Section IV reports experiments carried out on the VRPTW. Finally, the last section is devoted to the conclusions.

## II. An Overview of Hybridization

Currently, the hybridization of metaheuristics is present in much of the work done for the solution of Optimization Problems. The main reason for the increased use of this technique is the good results that have been obtained [12]-[13].

In much of the literature, a hybrid metaheuristic is defined as the combination of metaheuristics with other metaheuristics and/or with other methods, usually from the various areas of computational intelligence and operational research. This combination of methods aims synergy benefit from combining the features of each algorithm, obtaining therefore a better performance. However, how these methods are combined varies greatly and may involve advanced strategies, such as cooperation, parallelism, multi-agent systems, decomposition of the search space.

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. The cooperative metaheuristics are highlighted by several authors in the literature. [14] defines cooperative search as an optimization problem solving process, performed by various algorithms (or instances of the same algorithm), who share information about the search space, such as their status, solutions and sub-problems.

The cooperative metaheuristics facilitate parallelization process. In this case, different algorithms (or instances of a same algorithm) are executed independently exploring different regions of the search space concurrently. From there, it is up to the designer to determine how to control the solution process and how information is exchanged between the methods.

## III. Multi-Agent Metaheuristic Optimization Framework

A framework is a computational structure that provides generic functionalities for solving problems in a particular area by the junction of several common codes. A Metaheuristic Optimization Frameworks (MOF), as named in [11], is a software tool that provides implementations of the main metaheuristics, via reusable codes, facilitating the development of applications for solving optimization problems.

This paper presents a Multi-Agent MOF, based on the model proposed in [15], [16]. In this proposal, each agent encapsulates a heuristic/metaheuristic and has the function of seeking the solution for a given Combinatorial Optimization problem. The strength of the proposed framework is the hybridization capacity of metaheuristics through multi-agent approach, using concepts of cooperation and parallelism.

During the search process of the solution, the agents in this framework should go through the multi-agent system environment. In this case, the multi-agent environment is defined by the search space of the addressed problem. The search space of an optimization problem is the set generated by all their feasible solutions, i.e., those that complies with

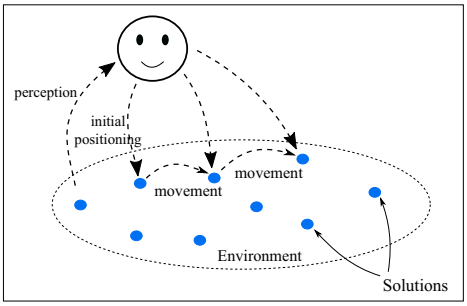


Fig. 1. Interaction between Agent and Environment.

the objective and satisfy the set of predefined constraints. The perception capabilities and action of the agent are defined in this environment as (see Figure 1):

- Perception of the environment: access to information concerning the problem and required for solving it;
- Positioning: the agent defines its position in the environment, either by building a new solution or the choice of an already available solution;
- Movement: mobility of the Agent by the environment, from one solution to another solution. The movement here comprises all types of modifications of a solution (neighborhood structure, operators) that allow the agent to move from one solution to another.
- Cooperation: to provide solutions for the other agents of the system (see Section III-A).

Each agent acts with mechanisms of action available to it, having thus a vision of the environment region which these mechanisms give it access. Therefore, its representation is partial in relation the environment. The goal is to apply, at the same time, the strengths of each metaheuristic through the cooperative work of the agents.

The cooperation takes place through the exchange of search space information between those involved in problem solving. The shared information are stored in a Pool Solutions. After each iteration of the search process, this Pool is updated.

Agents still have individual capacities of learning [17], [18]. Each agent uses a reinforcement learning technique (based on its previous experience) in selecting the neighboring structure to be used to improve a solution.

The Object Oriented Programming paradigm is used to facilitate the development of the framework, allowing reduce the effort used in the implementation of the methods and in adapting these to a specific problem. Therefore, a generic structure that allows the definition of the problem characteristics is used. This structure consists of three main components:

- Problem: component that provides the characteristics of the problem to be solved;
- Solution: defines the representation of the solution to be used;
- Movement: provides the interface between metaheuristics and solutions, allowing the construction, modification and combination of solutions.

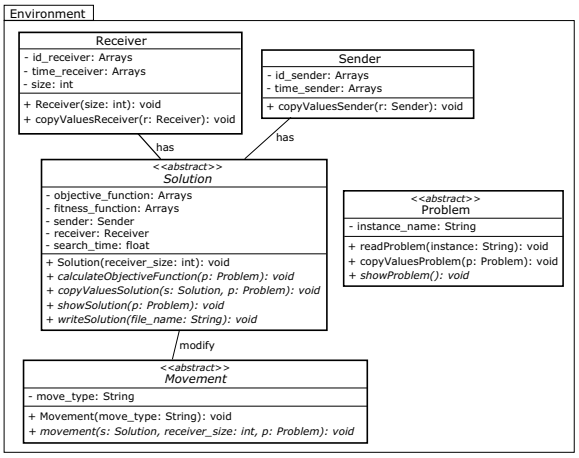


Fig. 2. Structure of the Environment of a Multi-agent System.

This structure is implemented based on the Factory Design Pattern that encapsulates the creation of objects, letting the subclasses decide which objects will be created.

A specific application can be developed with the customization of the framework, based on the definition of subclasses of the pre-defined classes. Thus, the framework user shall develop the specific elements of the problem you want to solve.

### A. Adaptive Capabilities of Individual Agents

As mentioned above, the agents have individual adaptive capabilities [17], [18]. The concept of Reinforcement Learning is used to define the application order of the neighborhood structures of local search. The Variable Neighborhood Descent (VND) [19] is the local search method used by the agent. VND is a refinement method which explores the search space using several neighborhood structures. In the conventional format, VND uses a predetermined sequence of application of neighborhood structures. When a better solution is found, the method returns to the first neighborhood structure. The VND ends when it encounters a local optimum with respect to all neighborhood structures.

In the present article, the order of the neighborhoods is chosen by applying an operator similar to the "roulette wheel" selection operator of Genetic Algorithms. For each possible pair of neighborhood structures $(m1, m2)$, a probability of choice is assigned. Initially, all pairs of sequences have the same probability value. The probability of choice of the sequence $(m1, m2)$ is updated by a reinforcement factor $w$ if a movement of the neighborhood structure $m2$ applied after another movement of the neighborhood structure $m1$ improves the current solution. This agent learning process is described in Algorithm 1.

### B. Cooperation

The cooperation between agents occurs through the exchange of information in the search space of the problem. The available solutions are stored in a Pool of Solutions in the multi-agent system environment. All agents have access to the solutions available in the Pool. The agents share, at the end of the iteration, the best solutions found so far.

**Algorithm 1** Adaptation of the agent behavior by Reinforcement Learning [20]

1: **procedure** REINFORCEMENTLEARNING(perception)
2: *perception* ▷ action (movement) which resulted in a better solution than the previously obtained
3: *memory* ▷ is the memory that the agent has already applied movements
4: $r_f$ ▷ reinforcement factor
5: $Memory \leftarrow upgrade\_Memory(memory, perception, r_f)$
6: $Action \leftarrow choose\_Best\_Action(memory)$ ▷ next move to be applied
7: $Memory \leftarrow upgrade\_Memory(memory, action)$
8: **return** *action*
9: **end procedure**

The purpose of this cooperative structure is to guide agents in the solutions space toward the most promising areas, and thus, improve the final result and reduce the time needed to solve the problem.

The maximum size of the Pool of Solutions is predefined and the insertion of new solutions is regulated by the function $s_i$. This function estimates the solution density in the neighborhood of the $i$ solution by means of the distance between the solutions contained in the pool. The distance between two solutions $i$ and $j$, given by $d_{ij}$, is evaluated considering how much they are similar or not and depends fundamentally on the problem being treated. As an example, considering the case of VRPTW described in Section IV, the distance between two solutions is calculated in relation to the number of common arcs to these solutions.

The evaluation function $s_i$ is given by the sum of distances of a solution $i$ for all the other pool solutions:

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

where $s(d_{ij})$ is defined as:

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

The factor $r$ is the pool radius, and controls the dispersion degree of the solutions. It should be calculated according to the treated instance.

When a solution needs to be inserted into the pool and this has no available space, the existing solutions are evaluated according to the function $s_i$. As a consequence, the solution with the worst evaluation is excluded for inserting the new solution in the pool. The main objective of this evaluation function is to maintain the diversity of the pool, avoiding to keep very similar solutions, or even equal. At the same time, the best existing solution in the pool is always stored in a specific attribute of the environment and updated at every insertion, thus preventing the found best solution be eliminated.

For evaluating the effectiveness of cooperation performed by the agents, two structures linked to the solution were defined

(see Figure 2): (i) Sender: stores the identifier of the agent that sent the solution to the Pool; (ii) Receiver: stores the identifiers of the agents who used the solution in the Pool. For each access solution, the time instant is also recorded.

For the evaluation of the Multi-agent Framework introduced here, the experiments presented in the following have used the concept of coalition according to [17]. This author describes "*coalition*" as "*an organization where agents have the same capacities and cooperate by the mean of direct interactions*". The implementation details are given in the next section.

## IV. Computational Experiments

The Multi-Agent MOF introduced here has flexibility both in solving many combinatorial optimization problems as in including new forms of cooperation and communication among agents. The experiment presented in the sequel, as already mentioned in Section III-B, is based on the concept of coalition proposed in [17]. The main objective is to improve and evaluate the cooperation between the agents of the Framework.

The agent used in this experiment implements a variation of the Iterated Local Search (ILS) metaheuristic [21]. This method is shown in Algorithm 2. In this algorithm, the perturbation of the solution, performed from changes in the current solution, is implemented at levels, i.e., at each iteration the perturbation function is changed if there is no improvement in the solution (line 12), and returns to its first level (line 10), if a better solution is found.

**Algorithm 2** ILS

```
1:  procedure ILS_VND_RL
2:      s_0 ← generateInitialSolution_GreedyRandom()
3:      s ← localSearchVND_RL(s_0, agent_memory)
4:      level_perturbation ← 1
5:      while stopping_criterion_is_not_reached do
6:          s' ← perturbation(s, level_perturbation)
7:          s" ← localSearchVND_RL(s', agent_memory)   ▷ uses Reinforcement Learning (RL)
8:          if f(s") < f(s) then
9:              s ← s"
10:             level_perturbation ← 1
11:         else
12:             level_perturbation ← level_perturbation + 1
13:         end if
14:     end while
15: end procedure
```

Identical ILS agents are used to solve the chosen problem. In this context, three test scenarios are proposed to evaluate the Framework: (i) a single agent ILS; (ii) two identical ILS agents; and (iii) four identical ILS agents. The scenarios with more than one agent make use of the cooperation environment during the search process.

For the tests presented here, the Framework has been customized to solve the Vehicle Routing Problem with Time Windows (VRPTW) [22]. In this problem a set of vehicles is located at a single depot and must serve a set of customers. Each vehicle has a given capacity. Each costumer has a given demand and must be served within in a specified time window. The objective is to determine a set of routes in order to minimize the involved total cost with this operation. The routes must start and end in the depot. In our case, the cost is calculated hierarchically. The priority is to minimize the number of vehicles (or routes). In case of a tie in the number of vehicles, it tries to minimize the total distance traveled.

In order to test the proposed framework, the 56 instances of VRPTW with 100 customers proposed by [23] were used. These instances are formed by three different sets of customers (C-Cluster; R-Random; and RC-Random-Cluster) in accordance with the geographic distribution considered. The proposed framework was implemented in Java with JDK 1.6, and the results were obtained using a personal computer with Intel i7- 4500U with 1.8 GHz, 16 GB of DDR3 RAM and Windows 7 Home Premium environment. It is worthy mentioning that competing with the best literature results for these instances of VRPTW is out the scope of this experiment. The data acquired from experiments are analyzed in two ways:

(a) Comparison between the distances obtained in the three presented scenarios: 1 ILS agent, 2 ILS agents and 4 ILS agents. This comparison aims to verify if the use of agents in a cooperative environment has effect on the results. These experiments demonstrate the scalability of the framework, that is, its ability to add new agents without computational effort.

(b) Analysis of the trajectory of the solutions in the process of cooperation among agents. This evaluation is performed from the solutions stored in the Pool Solutions. For each solution in the pool, the following information are known: (i) the agent who sent it to the pool and (ii) the agents who used it in their search process. For each one these accesses, the time instant is also recorded, in order to determine the trajectory of the solutions.

The comparison between the three presented scenarios is performed using the ANOVA test and boxplot graphics. The ANOVA test confirms the improvement of the performance with the addition of agents in 80 % of cases. In some instances, for example, the C101 instance, there is no difference in the values obtained in three scenarios. This occurs in instances where the optimum value, or the best known value, is reached or by a single agent or by the agents working together.

The boxplot graphics are used to evaluate the empirical distribution of found values in the results, its variability, and especially for the visual comparison between the data groups. Two points are evaluated in Figures 3 to 5 for each proposed scenario:

(i) number of vehicles: for the instances, the number of vehicles is not the same in all executions;

(ii) cost: distance traveled.

Figures 3, 4 and 5 present the values found in executions for instances C203, R201 and RC101, respectively. Three scenarios are shown: One Agent, Two Agents and Four Agents. In these three figures, we observe that the executions performed with two and four agents have outperformed executions with a single agent.

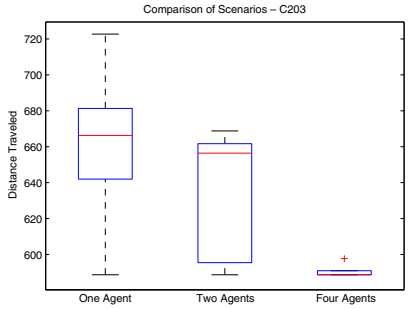


Fig. 3. Comparison scenarios in relation to the distance traveled - C203 instance

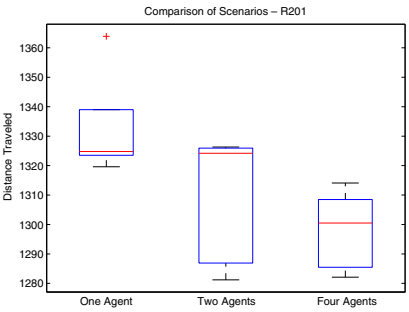


Fig. 4. Comparison scenarios in relation to the distance traveled - R201 instance

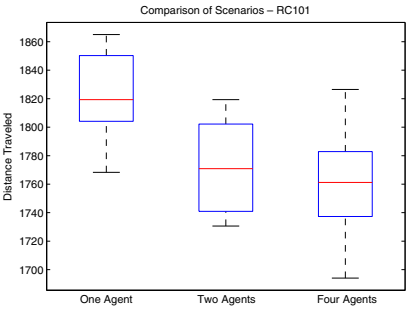


Fig. 5. Comparison scenarios in relation to the distance traveled - RC201 instance

Figures 6 and 7 present the vehicle numbers obtained in the execution of instances RC201 and R101, respectively. We observed that the number of used vehicles is also reduced with the use of resources of the cooperative environment. The scenario with four agents has obtained smaller and with less variability values.

Cooperation is evaluated using the trajectory of the solutions in the Pool Solutions as basis. Figure 8 shows the solutions submitted by two agents in Solution Pool for instance C102. In this figure, we observe that each agent perform, at the same time, its path to solve the problem, acting independently. We

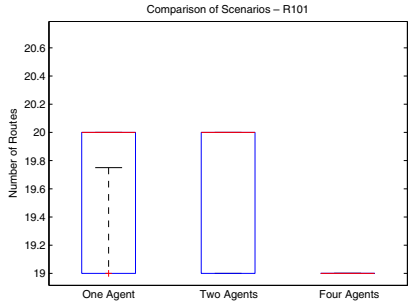


Fig. 6. Comparison scenarios in relation to the number of vehicles - RC101 instance.

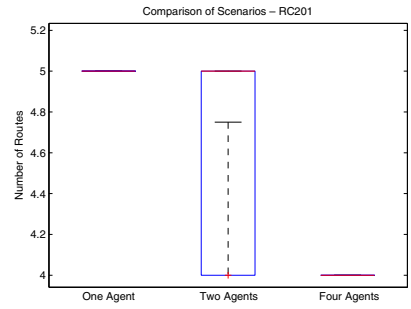


Fig. 7. Comparison scenarios in relation to the number of vehicles - RC201 instance.

highlighted that the solutions that have been shared by the agents in the pool and that really influenced the search are identified with red circles. A good example is the solution $A2$ with total distance 1034.00, which was found by the Agent 2 and inserted into the pool (at 31 seconds). Then, this solution, named $A1$, is accessed by Agent 1 (at 34 seconds) and used to pursue its search. From this cooperation between the agents, Agent 1 can achieve, as a final solution, the value of the best solution for this instance.

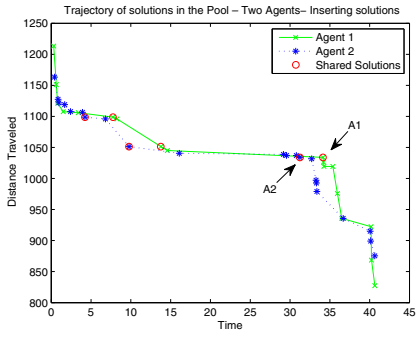


Fig. 8. Trajectory of solutions in the pool with two agents - inserting solutions - instance C102.

Figure 9 shows the same analysis for instance C207. The trajectory generated by the solutions inserted in the pool for

each agent is presented. In this run, as in the previous, we notice that each agent performs its search independently, by tracing different paths. Red circles indicate the points where cooperation has occurred. The second solution found by the Agent 3, named *A*3, is an example of cooperative action. It was initially obtained in the pool, after having been shared by the Agent 2 (as the *A*2 solution). Another example of the cooperative effect is the solution with total distance 606.50 (inserted by Agent 4 in the pool at the instant 1.42 seconds and named the *A*4 solution). This solution is, at time 2.58 seconds, used by Agent 2, as the *A*2 solution, which then will find the final solution (the best solution among the agents).

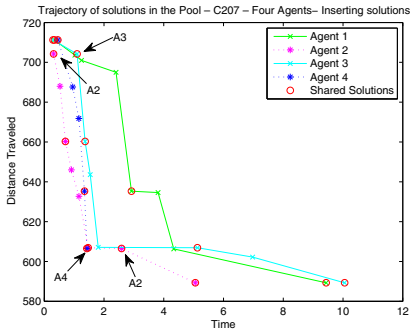


Fig. 9. Trajectory of solutions in the pool with two agents - inserting solutions - instance C207.

## V. Conclusion

A new Multiagent Metaheuristic Optimization Framework is presented in this paper. In this approach, each agent acts independently in the search space of a combinatorial optimization problem. The agents have shared information and have collaborated with other agents through the multi-agent environment. This cooperation and influence ability on the quality of the solutions of the agents involved is confirmed by the realized experiments. The results demonstrate a reduction in costs with the use of cooperating agents. This reduction is even better when increasing the number of agents.

## Acknowledgment

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

## References

[1] A. Fink and S. Voß, "Hotframe: A heuristic optimization framework," in *Optimization Software Class Libraries*, ser. Operations Research/Computer Science Interfaces Series, S. Voß and D. L. Woodruff, Eds., 2002, vol. 18, pp. 81–154.

[2] E. Alba, G. Luque, J. Garcia-Nieto, G. Ordonez, and G. Leguizamon, "MALLBA: a software library to design efficient optimisation algorithms," *Int. J. Innov. Comput. Appl.*, vol. 1, no. 1, pp. 74–85, 2007.

[3] L. D. Gaspero and A. Schaerf, "EASYLOCAL++: an object-oriented framework for the flexible design of local-search algorithms," *Software: Practice and Experience*, vol. 33, no. 8, pp. 733–765, 2003.

[4] I. M. Coelho, P. L. A. Munhoz, M. N. Haddad, V. N. Coelho, M. M. Silva, M. J. F. Souza, and L. S. Ochi, "OptFrame: A computational framework for combinatorial optimization problems," in *Proc. of the VII ALIOEURO Workshop on Applied Combinatorial Optimization*. ALIO/EURO 2011, MAY 2011, pp. 51–54.

[5] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[6] S. Cahon, N. Melab, and E.-G. Talbi, "ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics," *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, 2004.

[7] A. Liefooghe, L. Jourdan, and E. Talbi, "A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO," *European Journal of Operational Research*, vol. 209, no. 2, pp. 104–112, 2011.

[8] N. Melab, T. V. Luong, K. Boufaras, and E. Talbi, "ParadisEO-MO-GPU: A framework for parallel GPU-based local search metaheuristics," in *Proc. of the 15th Annual Conf. on Genetic and Evolutionary Computation*. ACM, 2013, pp. 1189–1196.

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

[10] S. Talukdar, S. Murthy, and R. Akkiraju, "Asynchronous teams," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science, F. Glover and G. A. Kochenberger, Eds. Springer US, 2003, vol. 57, pp. 537–556.

[11] J. A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez, "Metaheuristic optimization frameworks: a survey and benchmarking," *Soft Computing*, vol. 16, no. 3, pp. 527–561, 2012.

[12] 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.

[13] I. Boussaid, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, Jul 2013.

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

[15] 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 *Proc. of the 2009 IEEE International Conf. on Systems, Man and Cybernetics (SMC 2009)*, 2009, pp. 3071–3076.

[16] M. A. L. Silva, S. R. de Souza, S. M. de Oliveira, and M. J. F. Souza, "An agent-based metaheuristic approach applied to the vehicle routing problem with time-windows," in *Proc. of the 2014 Brazilian Conference on Intelligent Systems - Enc. Nac. de Inteligência Artificial e Computacional (BRACIS-ENIAC 2014)*, São Carlos, SP, Brazil, 2014.

[17] D. Meignan, J.-C. Creput, and A. Koukam, "A coalition-based metaheuristic for the vehicle routing problem," in *Proc. of the 2008 IEEE Cong. on Evolutionary Computation (CEC 2008)*, 2008, pp. 1176–1182.

[18] I. Bachiri, J. Gaudreault, B. Chaib-draa, and C. Quimper, "RLBS: An adaptive backtracking strategy based on reinforcement learning for combinatorial optimization," *Cirrelt*, vol. 07, Feb 2015.

[19] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[20] S. Russell and P. Norvig, *Artificial Intelligence: a Modern Approach*. Prentice-Hall, 1995.

[21] H. R. Lourenço, O. Martin, and T. Stützle, "A beginner's introduction to Iterated Local Search," in *Proc. of the 4th Metaheuristics International Conf. (MIC 2001)*, Porto, Portugal, 2001, pp. 1–11.

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

[23] 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.