



Review Article

Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis



Maria Amélia Lopes Silva, Sérgio Ricardo de Souza ^{*}, Marcone Jamilson Freitas Souza, Moacir Felizardo de França Filho

Federal Center of Technological Education of Minas Gerais (CEFET-MG), 30510-000, Belo Horizonte, MG, Brazil

ARTICLE INFO

Article history:

Received 7 March 2017

Received in revised form 22 June 2018

Accepted 28 June 2018

Available online 7 July 2018

Keywords:

Metaheuristics

Multi-agent systems

Cooperation

Hybridization

Combinatorial optimization

ABSTRACT

This article presents a review and a comparative analysis between frameworks for solving optimization problems using metaheuristics. The aim is to identify both the desirable characteristics as the existing gaps in the current state of the art, with a special focus on the use of multi-agent structures in the development of hybrid metaheuristics. A literature review of existing frameworks is introduced, with emphasis on their characteristics of hybridization, cooperation, and parallelism, particularly focusing on issues related to the use of multi-agents. For the comparative analysis, a set of twenty-two characteristics was listed, according to four categories: basics, advanced, multi-agent approach and support to the optimization process. Strategies used in hybridization, such as parallelism, cooperation, decomposition of the search space, hyper-heuristic and multi-agent systems are assessed in respect to their use in the various analyzed frameworks. Specific features of multi-agent systems, such as learning and interaction between agents, are also analyzed. The comparative analysis shows that the hybridization is not a strong feature in existing frameworks. On the other hand, proposals using multi-agent systems stand out in the implementation of hybrid methods, as they allow the interaction between metaheuristics. It also notes that the concept of hyper-heuristic is little explored by the analyzed frameworks, as well as there is a lack of tools that offer support to the optimization process, such as statistical analysis, self-tuning of parameters and graphical interfaces. Based on the presented analysis, it can be said that there are important gaps to be filled in the development of Frameworks for Optimization using metaheuristics, which open important possibilities for future works, particularly by implementing the approach of multi-agent systems.

© 2018 Elsevier B.V. All rights reserved.

Contents

1. Introduction	434
2. Methodology adopted for the article	436
3. An overview on hybridization of metaheuristics	436
3.1. Cooperative metaheuristics	437
3.2. Parallel metaheuristics	437
3.3. Hyper-heuristics and hybridization of metaheuristics	438
4. Metaheuristic Optimization Frameworks – MOFs	438
4.1. EasyLocal++	439
4.2. ECJ	439
4.3. Evolutionary Algorithms Framework (EvA2)	439

* Corresponding author.

E-mail addresses: mamelia@ufv.br (M.A. Lopes Silva), sergio@dppg.cefetmg.br (S.R. de Souza), marcone@iceb.ufop.br (M.J. Freitas Souza), [\(M.F. de França Filho\).](mailto:franca@des.cefetmg.br)

4.4.	FOM: Framework for Metaheuristic Optimization	439
4.5.	HeuristicLab	439
4.6.	Heuristic OpTimization FRAMEwork (HotFrame)	440
4.7.	HyFlex	440
4.8.	A Java Metaheuristics Search Framework (JAMES)	440
4.9.	Java Class Library for Evolutionary Computation (JCLEC)	440
4.10.	jMetal (Metaheuristic Algorithms in Java)	441
4.11.	MALLBA	441
4.12.	OptFrame	441
4.13.	Opt4j	441
4.14.	Parallel and Distributed Evolving Objects (ParadisEO)	441
5.	Metaheuristics using multi-agent approaches	442
5.1.	Agent Evolution (AgE)	442
5.2.	AMAM	443
5.3.	Agent Metaheuristic Framework (AMF)	444
5.4.	A-Teams conceptual model	444
5.5.	Collaboration of Metaheuristic Algorithms (CMA)	445
5.6.	Distributed Agent Framework for Optimization (DAFO)	445
5.7.	Learning-Based Multi-agent System (LBMAS)	445
5.8.	Multi-agent Cooperative Search (MACS)	446
5.9.	MAGMA	446
5.10.	MultiAgent eNvironment for Global Optimization (MANGO)	446
5.11.	Swarm of Metaheuristic Agents (SMA)	447
6.	A comparative analysis and discussion	447
6.1.	Analyzed characteristics	448
6.2.	General characteristics for frameworks	449
6.3.	Advanced characteristics for framework	449
6.4.	Multi-Agent characteristics	452
6.5.	Support features to optimization process	454
7.	Conclusions and future directions	455
	Compliance and ethical standards	456
	Acknowledgements	456
	References	456

1. Introduction

The search for techniques that lead to flexibility in the incorporation of new methods as well as providing better solutions for solving optimization problems using metaheuristics, without requiring efforts to remake the application, has guided researchers to the development of structures like frameworks. Frameworks provide a structure of generic features for the solution of problems from a specific domain, making the development of new applications in this domain much easier.

This article introduces a deep comparative analysis of a set of metaheuristic optimization frameworks available in the literature, pointing out the strengths and difficulties of each one. The aim is to identify the lacks in the development of frameworks for optimization using metaheuristics and the important possibilities of new works, particularly by analyzing the influence of the hybridization of metaheuristics and implementing new technologies like the approach of multi-agent systems. The objective of this comparative is to comply with two types of specific audiences: (i) specialists or not specialists in methods to solve optimization problems, who are looking for tools that can facilitate the resolution of these problems, as well as offer resources that enhance the performance of the solution process; (ii) professionals involved in the development of frameworks for optimization, who wish to have knowledge of the state of the art in relation to the development of these tools, as well as the existing gaps, especially those associated to the use of multi-agents.

Some issues must be addressed before the presentation of this analysis. In recent years, the joint use of two or more metaheuristics for solving optimization problems has been growing [20,43]. This strategy is known as hybridization of metaheuristics. The main objective is to apply together the best features of

each metaheuristic to solve a problem, allowing, besides getting the best solution quality in a shorter time, to increase the ability to tackle more complex problems. Cotta et al. [43] and Blum et al. [20] claim that hybridizations are responsible for many of the best results found in the literature for various classes of optimization problems, which justifies the rising interest for this approach.

Several authors seek classifying engines for hybrid metaheuristics [116,43,105,22,19,20]. However, there is a struggle in employing a simple hierarchy in this classification, due mainly to the fact that these algorithms possess overlapping characteristics. In addition, many ways of hybridization of metaheuristics can be found in the literature. The demand for software each time more adaptive and intelligent have conducted to the incorporation of new strategies that diversify these hybridization forms. Among these strategies, we highlight, in this article, cooperative metaheuristics, parallel metaheuristics, and hyper-heuristics.

The cooperation between metaheuristics is used as a strategy for the exchange of information between the algorithms involved in finding the solution of the problem. This exchange of information has, as the main purpose, to guide the search for the most promising regions of the solution space. In turn, parallelism allows simultaneous execution of methods and the consequent reduction on search time. The combination between cooperation and parallelism is being intensively developed and is becoming each day more important in Optimization context. Hyper-heuristics, on the other hand, propose the use of a high-level heuristic to select heuristics for solving the problem. Thus, the most suitable heuristic on each search step is applied based on performance measures that are independent of the problem. In this sense, hyper-heuristics can be seen as a way of hybridization of metaheuristics.

However, metaheuristic hybridization has not been the focus of frameworks for optimization. An important alternative presented

on many works to fill this gap is the use of the concept of agents for the development of applications for solving optimization problems. The concept of agents and its generalization, in the form of multi-agent systems, constitute strategies of great importance on this sense, because, as Aydin [10] points, the use of multi-agent systems in the context of Optimization allows the simultaneous exploration of different search space regions, enabling more diversity and better solutions. The approach based on agents is used as a link between the different metaheuristics involved in solving a problem. In this case, each agent is responsible for performing its task and, at the same time, exchange information with other agents in relation to the stage of its search. A specific set of studies that use this approach is shown in this article. The works in question can also be identified as frameworks, as they define generic structures capable of dealing with different optimization problems.

In this sense, from the analysis of the frameworks discussed here, several questions arise: (i) do current frameworks facilitate the development of hybrid metaheuristic applications? (ii) do they allow application development with cooperation (exchange of information) between methods that are run independently and simultaneously? (iii) what is the best way to coordinate metaheuristics when the goal is to explore the potential for hybridization? (iv) at the same time, how could be developed a more robust structure capable of handling different problems with minimal changes?

This article aims to answer these questions. To this objective, a comparative analysis between general-purpose tools used in the optimization problem solving was made. Several authors present similar comparisons between frameworks for optimization. Most of these authors either use performance criteria in the comparison between the evaluated tools [58,130] or propose a specific framework and compare its own tool with other proposed in the literature [63,31].

Among the works used as a reference for this comparison, Parejo et al. [101] have presented a detailed comparative study carried out about metaheuristic optimization frameworks, through the analysis of ten frameworks selected from the literature. The purpose pursued by the authors was to define a set of characteristics of optimization frameworks and indicate future directions for the area. The authors used general research questions to check six areas of interest in the analyzed frameworks: (i) implemented metaheuristic techniques; (ii) adaptation of the framework to the problem and its structure; (iii) advanced features (such as hybridization, hyper-heuristics and parallelism); (iv) global optimization process support; (v) design, implementation and licensing; and (vi) documentation and support. Three points were observed in this analysis and they are highlighted here: (i) the choice of a framework, by a user, is guided by technique that he needs to apply, because the evaluated frameworks do not implement all the available techniques in the literature; (ii) the framework developers have focused more time on the coding of algorithms for the solution of the problems rather than supporting adaptation of these algorithms to the problems; (iii) the hybridization is not a strong characteristic of the existing frameworks, considering that they do not exploit the benefits of combining methods.

The current article proposes to extend the comparison presented in Parejo et al. [101], particularly focusing on issues related to the use of multi-agent structures in hybridization of metaheuristics. In this sense, strategies used in the combination of methods, such as cooperation, parallelism, decomposition of the search space and multi-agent systems are also assessed for their use in the various analyzed frameworks.

Due to the high number of found frameworks, five criteria were used for selecting the analyzed tools, and a set of twenty-two characteristics was listed for this comparison. Twenty-five frameworks for optimization available in the recent literature were selected for

the comparative analysis showed in this current article. The evaluations were carried out from the public information of the analyzed frameworks, available in articles, internet sites and existing manuals in these sites. It should be noted that all frameworks evaluated in Parejo et al. [101] were also evaluated here, except the OAT framework, since it only has a publication in the form of a technical report and, thus, does not meet the requirements exposed in Section 2. However, the nature of the evaluation performed here conceptually differs from the one carried out in this last cited article. Thus, the current article not only updates previous revision articles regarding frameworks for optimization but also differs from these other revisions, since, as explained in Section 2, it presents a deep qualitative analysis of the optimization frameworks shown here and, in addition, it includes multi-agents as the central role of this revision, perspective of analysis not yet addressed in the literature. The comparative tables developed from this analysis show that there are still gaps not covered by the existing works. At the same time, it allows the identification of important directions for future studies involving frameworks for optimization.

The main motivation for the accomplishment of this review article is the analysis of the paths that the academic research involving the development of generic structures for the solution via metaheuristics of optimization problems has followed in the last 15 years, especially those concerning the application of multi-agent systems. The number and diversity of publications involving multi-agent systems for solving optimization problems have grown enormously. Several frameworks have been introduced to this end and a balance of this period is necessary, in order to help the continuity of the research on the theme, while at the same time it is necessary to evaluate the other optimization frameworks that emerged in this period, in order to perform new comparisons and analysis. In this period, the importance of adopting the hybridization between heuristics and metaheuristics in order to obtain better quality results with lower computational cost when solving optimization problems of various class becomes clear. Alongside this, there is greater access to parallel computing resources, opening up new possibilities for developing techniques for solving these problems.

The main contributions of the current paper are to:

- (i) Identify the characteristics necessary for the development of frameworks for optimization using metaheuristics and, from these characteristics, identify gaps in this field;
- (ii) Show that the use of the concepts of multi-agent systems in the design of frameworks for optimization using metaheuristics facilitates and flexibilizes the development of hybrid metaheuristics and allows simultaneous exploration of different regions of the search space.

The remainder of this paper is organized as follows. Section 2 explains the methodology adopted to write this review. Section 3 presents an overview of hybridization of metaheuristics, focusing on cooperative Metaheuristics, parallel metaheuristics, and also discusses hyper-heuristics and their relationship with the hybridization of metaheuristics. In Section 4, a set of frameworks for optimization using metaheuristics, available in the literature, is presented, focusing its general characteristics. Section 5 shows how the multi-agent approach is used in metaheuristic hybridization and, at the same time, identifies framework optimization adopting this approach. In Section 6, we analyze the frameworks for optimization previously showed and carry out a comparison and a wide discussion about them and its characteristics. Finally, Section 7 concludes the article.

2. Methodology adopted for the article

This article presents a comparative analysis of frameworks for the solution of optimization problems using metaheuristics. In particular, it addresses the use of multi-agent systems in frameworks for optimization, considering the reasons presented in Section 1.

The adopted methodology for writing this article reflected these objectives and sought ways of satisfying them. This methodology consisted of five steps: (i) data collection; (ii) definition of analysis indicators; (iii) classification of the frameworks according to these indicators; (iv) selection of frameworks to be analyzed in the article, from the classification made; (v) comparative analysis.

In the data collection step, the search for articles was performed through the Google Scholar tool. Google Scholar was used since it is a free electronic search engine which enables to reach directly all major academic databases of both periodicals and scientific events. Additionally, the provided citation information allowed to determine the impact of the articles in the scientific community. At this step, an extensive search was performed using the specific keywords: “*metaheuristics optimization framework*”, “*multi-agent optimization framework*”, “*MAGMA framework*”, “*cooperation search*”. These four search strings led to new search strings, generated by the results of the initial search. The new search strings were directly related to the most consolidated frameworks, being they: ParaDisEO, MALLBA, EasyLocal++, and JMetal. In addition, all papers that cited Parejo et al. [101] were analyzed, in a forward snowballing search way, in order to find potentially related works. The set of adopted search strings and the snowballing search procedure made possible a non-biased revision that aggregated consolidated frameworks to recently published works, such as MACS, or little-known works, such as JABAT and OptFrame. This step ended with the identification of candidate frameworks to be analyzed in the article.

The definition of indicators that allow (i) the differentiation between the characteristics of the frameworks; (ii) a classification of the frameworks; and (iii) the selection of those frameworks that have real interest for this article were the second step of this analysis. The general indicators adopted to define the frameworks to be analyzed are:

- (i) Framework for solving combinatorial optimization problems: the selected frameworks are exclusive to solving mono and/or multi-objective combinatorial optimization problems;
- (ii) Framework of general purpose: the frameworks must present a general scope structure, allowing the solution of different combinatorial optimization problems, as well as the ease of including new methods. Thus, frameworks directed to specific classes of problems or to a specific metaheuristic were excluded from this study;
- (iii) Framework that supports trajectory and/or population optimization techniques: Frameworks that implement trajectory-based and/or population-based metaheuristics;
- (iv) Frameworks that use the concept of agents and that meet the first three indicators, that is, solve combinatorial optimization problems, are general purpose and support trajectory-based and/or population-based optimization techniques;
- (v) Frameworks that have at least one publication in scientific journals and / or conferences. They will not be considered those that have been published only in the form of technical reports, since they do not have forms of peer evaluation of the work.

The third step is to classify the frameworks according to the indicators. This step requires, for its effectiveness and efficiency, the knowledge of the frameworks and the presentation of minimum information about them. The fourth step is the selection of the frameworks to be exposed in the article, discarding those that

are not adherent to the proposed comparison goals. Based on these criteria, the following works are evaluated: AgE [77,122,30], AMAM [57,112,113], AMF [89–92], Collaboration of Metaheuristic Algorithms [86,87], DAFO [48,49], EasyLocal++ [61–63], ECJ [129], Eva2 [76], FOM [100], HeuristicLab [125–128], HotFrame [59,58], Hyflex [97], JABAT [14,16], JAMES [50,51], JCLEC [123,32], JMetal [53,52], LBMAS [81,82], MACS [88], MAGMA [94], MALLBA [4,3,5], MANGO [75,67,8], OptFrame [36,39], Opt4j [83], ParadisEO [31,79,93,69] and Swarms of Metaheuristic Agents [9–11]. Sections 4 and 5 summarize the analyzed frameworks, showing the necessary elements for the comparison exposed in this article. Finally, the last step consists in carrying out the comparative analysis.

It is important to be clear that this analysis does not have, as an objective, the establishment of a ranking among the analyzed frameworks, as realized, for example, in Parejo et al. [101]. Rankings often carry a high degree of subjectivity in the definition of quantitative metrics of analysis, which may obscure, and not clarify, vital issues both in the knowledge of the current state of the art of the topic and in pointing to future paths of research development. The comparative analysis presented here is therefore strictly qualitative, which does not exempt it from errors, of course, but it provides elements for the reader to differentiate the analyzed frameworks and to be clear about the current limitations and advances in the construction of frameworks for solving optimization problems.

3. An overview on hybridization of metaheuristics

Currently, the hybridization of metaheuristics is present in much of the scientific works focused on solving optimization problems. The main reason for the increasing use of this technique is the good results that have been obtained. According to Cotta et al. [43], Blum et al. [20] and Boussaid et al. [23], the hybridizations are responsible for many of the best results in the literature for several classes of optimization problems.

In great part of the literature reviews related to this topic, the term “*Hybrid Metaheuristics*” is defined as the combination of metaheuristics with other metaheuristics and/or with other methods, usually from the various areas of Computational Intelligence and Operations Research. Raidl [105] adds to this definition the possibility of combining not only complete structures of metaheuristics but also components of metaheuristics. This combination aims to provide a synergy between metaheuristics, from the junction of the characteristics of each algorithm, seeking, thus, to obtain better performance when compared to the use of each technique on its pure form or isolated.

Reviews related to hybrid metaheuristics may be found in Talbi [116], Cotta et al. [43], Raidl [105], Blum and Roli [22], Blum et al. [19] and Blum et al. [20]. Similarly, several publications highlight the combination of metaheuristics with other methods, such as exact methods, as in Puchinger and Raidl [103] and Jourdan et al. [73], or the combination of specific approximate methods, such as evolutionary algorithms, according to El-Mihoub et al. [55] and Rodriguez et al. [107].

Authors like Talbi [116], El-Abd and Kamel [54], Cotta et al. [43], Raidl [105] and Jourdan et al. [73] have proposed different taxonomies for hybrid metaheuristics. These authors define classifications that allow the comparison of hybrid metaheuristics qualitatively, as well as the identification of similarities and key components.

The taxonomy introduced by Talbi [116] is used as basis for several of the proposals presented later regarding new taxonomies, as Cotta et al. [43]; El-Abd and Kamel [54]; Raidl [105], and involves issues related to the project, regarding the structure of the algorithm, and the implementation, regarding hardware, programming language and execution environment. From the project point of

view, two issues are considered, one subordinated to the other: (i) what is the level of hybridization performed? and (ii) what is the order in which metaheuristics are executed? The classification of Raidl [105] proposes to add two more questions to those previously raised: (iii) what is hybridized? and (iv) which control strategy is used?

In relation to the level of hybridization, two types are considered in this taxonomy: (i) Low-Level Hybridization, in which a part of a metaheuristic is replaced by another metaheuristic, or, in view of the extension of this concept introduced by Raidl [105], is the possibility of combination of components of a metaheuristic; and (ii) High-Level Hybridization, in which the metaheuristics involved are self-sufficient. The low-level hybridization closely resembles the design of a new algorithm, since it leads to the application of a single element resulting from the combination.

Regarding the type of hybridized algorithm, three categories are identified: (i) metaheuristics with metaheuristics; (ii) metaheuristics with specific algorithms/simulation; and (iii) metaheuristics with other techniques of Operational Research and Artificial Intelligence, such as mathematical programming methods, neural networks, and fuzzy logic.

The control strategy can be either coercive or cooperative, the first being related to situations in which one algorithm is subordinate to another, and the second related to combinations in which the algorithms exchange information, but one is not part of the other. The metaheuristics based on the cooperative search are currently an important field in Optimization, receiving great prominence in recent years. The use of multi-agent concepts in the hybridization of metaheuristics applies the cooperative control strategy.

The execution order is classified in three ways: (i) block (batch), where each metaheuristic is executed once in sequence and the information is passed in only one direction; (ii) interleaved, in which the metaheuristics are executed one after the other, but, unlike the previous classification, the algorithms must interact in a more elaborate way; and (iii) parallel, in which the metaheuristics are executed simultaneously and periodically exchange information. This classification also presents characteristics that distinguish the proposals related to the order of parallel execution, taking into consideration issues like hardware, architecture, memory and other elements that influence it. Parallel metaheuristics have also received great prominence in recent literature. The parallel execution order has been used to reduce the computational time spent to find the solution mainly in more complex problems. In many recent cases, proposals that fit the high-level hybridization make use of the cooperative control strategy through the parallel execution order, which allows self-sufficient algorithms to run simultaneously and exchange information about the search.

However, these classifications have difficulties in categorizing hybrid metaheuristics developed by employing a simple hierarchy. The difficulties are due to the fact that these implementations do not belong to a single class of metaheuristics, and reside, in particular, in the way the characteristics of algorithms may overlap. Likewise, many authors treat hybrid metaheuristic subclasses as cooperative metaheuristics, parallel metaheuristics and hyper-heuristics as independent classes, proposing specific definitions and classifications [46,2,43,54,47]. These subareas stand out from the others because they are developing intensively and becoming every day more important in Optimization context. This importance is discussed and emphasized in the rest of this section.

Subsection 3.1, in the sequel, reviews issues related to cooperation between metaheuristics. Subsection 3.2 presents a brief analysis about parallel metaheuristics. In addition, Section 3.3 discusses the hyper-heuristics as a proposal for hybridization of metaheuristics.

3.1. Cooperative metaheuristics

Several authors in the literature stress cooperation between metaheuristics. Reviews and classifications related to the topic can be found in Blum and Roli [21] and El-Abd and Kamel [54]. Each of these authors presents its concept for the idea of cooperative search and the context in which it is used. In order to assist in the completeness of this article, some of these concepts are presented below.

Blum and Roli [21] define cooperative search as an optimization problem-solving process performed by various algorithms (or instances of the same algorithm), which share information, as the search stage, solutions and sub-problems, on the search space. El-Abd and Kamel [54] also describe the cooperative search algorithms as a particular theme and present definitions and specific taxonomies. According to these authors, “*the cooperative search is a category of parallel algorithms, in which several search algorithms are run in parallel in order to solve the optimization problem in hand.*”. Crainic and Toulouse [46] consider “cooperation” as a strategy for solving problems and design of algorithms. The authors state that there are two features common to all different forms of cooperation: “*(i) set of highly autonomous programs (APs), each one implementing a particular solution method; and (ii) a cooperation scheme combining these programs into a single solution strategy.*”.

From the above concepts, we can define “cooperation”, in the context of Optimization, as the use of different agents working together by exchanging information to achieve a common goal. Frequently, this goal is to find a solution to a given problem. In the case considered here, agents are understood as elements that implement each one a search method for solving the problem, and, in addition, that act autonomously.

According to the taxonomies introduced by Talbi [116], the cooperative search may be performed either in a sequential manner (relay) or in a parallel manner (teamwork). Some examples of works showing how to accomplish cooperative search are Alba et al. [3], Villaverde et al. [124], Amaya et al. [7] and Jin et al. [70].

In the context of metaheuristic cooperation, it is important to highlight Cooperative Co-evolution. Cooperative Co-evolution is a paradigm for solving high dimensional complex optimization problems in which the problem of interest is partitioned into lower dimensional sub-problems, each being solved by a different algorithm on particular sub-populations [102]. Thus, it is not a matter of simply dividing the total population into sub-populations, but rather breaking down the problem into sub-problems and solving them according to the particular conditions. The immediate issue is coordination between the various sub-problems. Cooperation between sub-populations occurs in the cooperative assessment of each individual of the sub-populations. This interaction between sub-populations happens within a shared domain. In this case, the global goal of the entire system is essentially the local goal of each sub-component. According to Potter and De Jong [102] and from Gong et al. [65], when developing co-evolutionary algorithms, four issues need to be addressed: (i) the decomposition of the problem; (ii) the evolution of interdependent subcomponents; (iii) the evaluation; and (iv) the maintenance of diversity. A recent survey about it is Gong et al. [65] and a good discussion is found in Byrski et al. [29].

3.2. Parallel metaheuristics

Crainic and Toulouse [47] define parallel/distributed computing as “*several processes work simultaneously on several processors solving a given problem instance*”. Still according to the authors, “*parallelism thus follows from a decomposition of the total computational load and the distribution of the resulting tasks to available processors. The decomposition may concern the algorithm, the problem-instance*

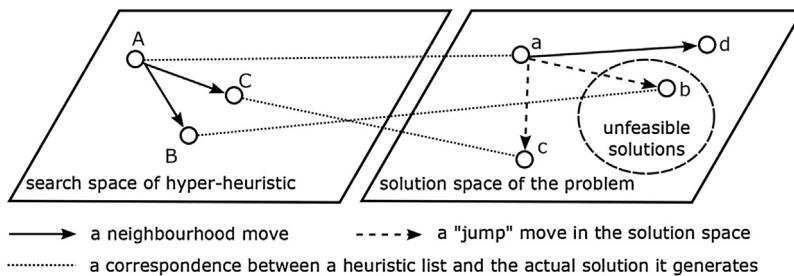


Fig. 1. The relation between the search space of the hyper-heuristic and the solution space of the problem [27].

data, or the problem structure". Thus, the more apt to decomposition an algorithm is, the easier is the process of parallelization of its functions or data. Several articles dedicated to parallel metaheuristics are available in the literature, standing out those found in Trienekens and Bruin [121], Crainic and Gendreau [45], Alba [2] and El-Abd and Kamel [54].

Metaheuristics, however, as pointed out by Crainic and Toulouse [47], belong to the category of algorithms that are hard to parallelize. This difficulty is mainly due to the strong dependency between each iteration of metaheuristics. In most methods based on trajectory, for example, there is a strong dependency between the successive iterations. A similar situation of dependency occurs with the evolutionary methods (that is, the population-based metaheuristics), in relation to the passage from one generation to another.

Hybrid metaheuristics, on the other hand, especially those based on cooperative search, facilitate parallelization process. Different algorithms (or instances of the same algorithm), running independently, certainly explore different regions of the search space and become a source of parallelism easy to be implemented.

In parallel metaheuristics, various methods can be used to run concurrently. It is up to the designer to determine how to control the solution process and how information is exchanged between the methods. The cooperative search techniques are often implemented in parallel. However, it should be clear that the use of metaheuristics parallelization techniques does not necessarily imply the existence of cooperation between metaheuristics. The decomposition of an algorithm in separate processes, without any cooperation, falls into the category of parallel metaheuristics (as well as distributed algorithms), but cannot be seen as cooperative metaheuristics, as there is no exchange of information during the search process. Therefore, hybridizations without cooperation can also be implemented in parallel, providing only the reduction of the execution time.

Some examples of works showing the use of parallel metaheuristics are Toutouh and Alba [120], Alirezai et al. [6] and González-Álvarez and Vega-Rodríguez [66].

3.3. Hyper-heuristics and hybridization of metaheuristics

In the context of Optimization, the term "hyper-heuristic" is used to describe "heuristics to choose heuristics", as said by Cowling et al. [44], or "an automated methodology for selecting or generating heuristics to solve hard computational problems", as found in Chakhlevitch and Cowling [35]. According to Burke et al. [26], "a hyper-heuristic is a high-level approach that, given a particular problem instance and a number of low-level heuristics, can select and apply an appropriate low-level heuristic at each decision point".

Thereby, the hyper-heuristic operates over a set of heuristics, selecting, during the search, a low-level heuristic, most appropriate for finding the solution among the available ones. The hyper-heuristic handles, therefore, two search spaces: (i) the search space of the hyper-heuristic; and (ii) the search space of the problem.

Fig. 1 shows the relationship between the search spaces of the Hyper-heuristic, as point out by Burke et al. [27]. The search space of the hyper-heuristic involves the space of the heuristics, i.e., the available heuristics to solve the problem. The search space of the problem is formed by the feasible solutions of the tackled problem.

As a result, in the light of the concept adopted for hybridization of metaheuristics, hyper-heuristics can be seen as an important category of hybridism, involving associations, combinations, and cooperation (at various hierarchical levels) between metaheuristics for solving optimization problems.

Like metaheuristics, hyper-heuristics are not specific to a problem. Thus, decisions regarding the selection of the used heuristics are performed based on measures that are independent of the problem, such as, for example, the quality of the solution [98].

Hyper-heuristics are good alternatives in cases where a fine-tuning of parameters involved is needed, allowing the implementation and evaluation of different heuristics to a problem. Some examples using hyper-heuristics are showed in Cowling et al. [44], Burke et al. [27], Malek [87] and Özcan and Kheiri [99].

4. Metaheuristic Optimization Frameworks – MOFs

In the search for better solutions using metaheuristics, generic structures, called frameworks, have been used. These structures have flexibility as a fundamental characteristic, and thus allow the addition of new methods without requiring any effort to remake the whole implementation. Moreover, they facilitate the development of metaheuristics, and, at the same time, also incorporate the concept of hybridization in some studies. Frameworks aimed at optimization with metaheuristics are the central subject of analysis in this section.

Examining the meaning of the word "framework" in the Oxford Dictionary of Computer Science [28] yields:

"A template for the development of software applications or components."

In the literature, various authors present definitions for the term "framework", which mainly involve the concept of object orientation:

"A framework is a set of classes that embodies an abstract design for solutions to a family of related problems." [72]

"A framework is a set of cooperating classes that make up a reusable design for a specific class of software." [60]

Therefore, a framework can be understood as a skeleton that provides generic functionality for solving problems in a specific domain by the junction of several common codes. A framework pre-defines the overall structure of how an application should be built from it, defining classes and objects, the collaboration between these classes and control threads. A specific application can be developed with the customization of the framework, based on the definition of subclasses of abstract classes already defined. Thereby,

it is up to the framework's user to develop the specific elements of his application.

Metaheuristic Optimization Framework (MOF), as named in Parejo et al. [101], is a software tool that provides implementations of the main metaheuristic methods through reusable codes that facilitate the development of applications for optimization problems.

The use of frameworks allows, for researchers from Optimization area, numerous advantages such as pre-implemented metaheuristics for test and reuse; support to the evaluation and comparison of different methods; ease of development of a particular metaheuristic and its appropriateness to the problem. In addition, the metaheuristic hybridization can also be facilitated with the use of frameworks. Proposals that have this feature differ on the approach used in the organization of metaheuristics and the way they interact with one another.

Numerous frameworks for problem solving using metaheuristics are available in the literature, most of them with similar characteristics and proposals. The following subsections present fifteen of the most referenced Metaheuristic Optimization Frameworks in the literature, arranged in alphabetical order with respect to its acronym. In addition, the main features of these frameworks are evaluated in Section 6.

4.1. EasyLocal++

In Gaspero and Schaerf [61–63], EasyLocal++ framework is introduced. It was developed in C++ language, for design and analysis of Local Search algorithms aiming at solving Combinatorial Optimization problems. The objective of the framework is to capture the main technical features of Local Search, and their combinations, through an object-oriented design.

Since it is specialized only in local search methods, the EasyLocal ++ framework does not offer the diversity of methods that other frameworks in the literature have. The hybridization supported by EasyLocal ++ is possible only from models of predefined hybrid algorithm and is not simple to adapt new structures. Gaspero and Schaerf [63] is the latest publication related to this framework. However, the associated site and the own framework have been updated in the last years, being the last update released in February 2017. The latest version (version 3.0) of EasyLocal++ framework is available at <https://bitbucket.org/satt/easyllocal-3>.

4.2. ECJ

ECJ (EC system written in Java), is the framework proposed in White [129], that enables researches in Evolutionary Computing. In this sense, ECJ is a generic tool that aims to make available the implementation of all forms of evolutionary computing.

Developed in Java, strategies such as design patterns and inheritance used in the framework definition allow it to support many alternative methods for common functions, such as population initialization and mutation and selection operators without additional user effort.

This framework offers, as additional features, a Graphical Interface and support for parallelism and distributed computing. The ECJ graphical interface allows loading and executing algorithms based on parameter files, checkpoint files, editing parameters, and charting statistics. In relation to parallel and distributed execution functions, the ECJ framework has multi-thread support and execution based on master-slave architecture and island models. On the other hand, this framework does not allow the hybridization of metaheuristics.

Luke [84] is the most recent publication about ECJ. It presents a proposal to expand ECJ, in order to update the Java pattern of ECJ; to perform changes in the framework architecture, incorporation of

new metaheuristics, in order to allow hybridization of metaheuristics, treatment of multi-objective metaheuristics and the analysis of results, as well as increasing the usability of the framework. The last available version of ECJ is the number 25, November 2017, and can be found at the project website <http://cs.gmu.edu/~eclab/projects/ecj/>.

4.3. Evolutionary Algorithms Framework (EvA2)

Kronfeld et al. [76] present a framework called EvA2 (**E**volutionary **A**lgorithms framework). EvA2 is a metaheuristic optimization framework, aimed at Evolutionary Algorithms, derived from the so-called JavaEvA optimization toolbox. The architecture of the EvA2 framework is implemented in Java and has as main characteristic a client-server structure, which allows its search to be distributed over the nodes of a network.

EvA2 provides the implementation of various optimization strategies, preferably population-based. Although its emphasis is on evolutionary methods, the EvA2 framework implements classical and path-trajectory methods, such as Nelder-Mead-Simplex and Simulated Annealing.

In addition to the client-server structure, another feature offered by EvA2 is the graphical interface. This interface allows users who do not know the theory of evolutionary algorithms to use these methods to solve a specific problem. This framework does not offer any alternative for the hybridization of metaheuristics.

The latest version of the EvA2, available at <http://www.ra.cs.uni-tuebingen.de/software/EvA2/>, is 2.2.0, dated December 2015.

4.4. FOM: Framework for Metaheuristic Optimization

In Parejo et al. [100], an object-oriented framework for optimization using metaheuristics, called FOM (Framework for Metaheuristic Optimization), is presented. The main objective of this framework is to propose a general structure that separates, through pre-defined interfaces, the problem to be solved of the solution algorithms involved.

FOM offers, as a differential, the possibility to adjust parameters through an element called “Observer”. These elements follow the parameters evolution of the metaheuristics, at each iteration, and use this information, through elements called “Visualizers”, to generate evolution graphs or to store the values obtained.

Two classical problems of Operations Research (SAT Problem and TSP) are used for the development of applications in Parejo et al. [100]. From the results obtained in these applications, the authors highlight that the use of the FOM framework makes implementations less efficient, even though code reuse reduces development time and induces a correct and robust structure in the implementation of the methods.

FOM also offers graphical user interface, which allows the instantiation of metaheuristics, with the characteristics desired for the solution of the problem selected by the user. As regards the hybridization of metaheuristics, it does not offer any possibility of this kind of combination.

Although the last release related to the framework is dated 2003, its development continued, being the last update of the code realized in 2013. The latest version of FOM (version 0.5, from July 2013) is available at <http://www.isa.us.es/fom> under the GNU GPL license.

4.5. HeuristicLab

Initiated in 2002, the HeuristicLab software environment was developed in the Heuristic and Evolutionary Algorithm Laboratory (HEAL) and was presented in Wagner and Affenzeller [125,126] and Wagner et al. [127,128]. It is an extensible and flexible software

system for Heuristics and Evolutionary Algorithms. Among other features, the system provides ease in the adjustment of the algorithm to a specific problem, enables the implementation of various algorithms and parallel and / or distributed execution from these algorithms.

It stands out for the number and variety of techniques implemented, as well as for the variety of problems addressed, including also multi-objective problems. However, regarding the hybridization of metaheuristics, the HeuristicLab framework does not offer any possibility of implementation.

This framework was extended in Wagner and Affenzeller [125] to a version inspired in grid computing, called HeuristicLab Grid. In this version, each participating computer receives a Client installation (implemented in C#), which searches the tasks in a central database server. The tasks are locally processed, and the obtained results are stored in the central database. Additionally, the HeuristicLab Grid includes Web Services that allow the construction of different types of interface.

A detailed history of the development of HeuristicLab framework is presented in Wagner et al. [128] and Elyasaf and Sipper [56], describing design features of all versions available since 2002. According to the authors, the HeuristicLab has been widely used by the Heuristic and Evolutionary Algorithms Laboratory (HEAL) research group. The authors also claim that the HeuristicLab is a project still in progress. The last downloadable version (3.3.15) is available at <http://dev.heuristiclab.com/>, published in January 2018.

4.6. Heuristic OpTimization FRAMEwork (HotFrame)

In Fink et al. [59] and Fink and Voß [58], the Heuristic Optimization FRAMEwork (HotFrame) is proposed. It is an object-oriented framework, developed in C++ language. Fink and Voß [58] describes in detail its architecture, showing components, implementation and applications. This description exposes how the framework covers metaheuristic concepts, leading to generic and parameterized components, such as solution space, neighborhood structures, and tabu-criteria. In this way, the Hotframe framework can be extended in several directions and allows the addition of new problems to be solved.

Regarding the usability of this framework, the authors affirm that direct use requires basic knowledge of frameworks and metaheuristics. Additionally, a graphical interface was generated, from an interface generator software, allowing the configuration of experiments and making ease its application by non-expert users.

The Hotframe is very similar to the other basic frameworks, not offering additional resources that allow potentiatizing the search of the solution. With respect to the hybridization of metaheuristics, it does not offer any alternative for this type of combination. Although publications related to this framework are available in the literature, no publication about it is known after 2003. Information regarding download of the Hotframe framework is available at <https://www.bwl.uni-hamburg.de/en/iwi/forschung/projekte.html>. The last update was carried out in July 2003.

4.7. HyFlex

HyFlex (Hyper-heuristics Flexible framework), presented in Ochoa et al. [97], is a framework for implementation, testing and comparison of general-purpose iterative heuristic search algorithms, and, in particular, hyper-heuristics. Developed in Java, Hyflex is designed as a modular and flexible class library. According to the authors, modularity is defined using decomposition concepts of search algorithms into two main parts: (i) A general-purpose part: the algorithm or hyper-heuristic; and (ii) The problem-specific part: provided by the HyFlex framework.

Among the additional features evaluated in this study, Hyflex stands out because it offers the possibility of hybridization of metaheuristics. However, this hybridization is limited to the way the hyper-heuristics themselves are defined, not allowing the user to define other forms of combination. The other resources evaluated in this article, such as parallelism and graphical interface, are not described by the documentation associated with the framework.

Several publications [24,25,64,99] use the Hyflex framework in their proposals. Documentation for the latest version of the Hyflex framework (v1.1) can be found at <http://www.asap.cs.nott.ac.uk/external/chesc2011/>, released in February 2012. No new publications were found presenting new versions of this framework.

4.8. A Java Metaheuristics Search Framework (JAMES)

De Beukelaer et al. [50,51] present JAMES (Java Metaheuristics Search) framework, in its version v1.1. JAMES is a framework for discrete optimization, focused on the use of local search algorithms. Its architecture defines a framework that strongly separates the problem specification from the solution algorithms. The search interacts with the problem to obtain, evaluate and validate solutions, being they random or constructed. The search also includes elements called listeners, responsible for informing the occurrence of certain events, for example, when a new best solution has been found.

As the framework is specialized in local search algorithms, the search structure defined in JAMES is based on the concept of neighborhood searches. They move through the search space using one or more neighborhoods, along a certain trajectory toward an optimal one.

As an additional resource, which enhances the search of the solution, the JAMES framework presents only the possibility of using threads in the execution of the algorithms, allowing parallel execution. It does not offer the possibility of combining the implemented methods and, therefore, the hybridization of metaheuristics is not supported.

The code and documentation, as well as the implementations used to evaluate it, are available at <http://www.jamesframework.org/>. Version 1.1 is the last one released, being presented on August 16, 2016.

4.9. Java Class Library for Evolutionary Computation (JCLEC)

Ventura et al. [123] propose a Java-implemented framework for the development of evolutionary computation applications named Java Class Library for Evolutionary Computation (JCLEC). The architecture of the JCLEC framework includes three main modules: (i) JCLEC Core: specifies the data types that define the functionality of the framework; (ii) Experiments Runner: is an execution environment of the algorithms; and (iii) GenLab: is the graphical user interface.

Although it presents a user-friendly interface, offering features that allow rapid prototyping of applications, this framework is focused only on Evolutionary Computing methods and does not offer additional features that enhance the search of the solution, such as hybridization.

An evaluation of this framework is performed in Ventura et al. [123], where a solution of the 0/1 knapsack problem through it is shown. However, the results are not presented. Cano et al. [32] introduce an extension of JCLEC called JCLEC-Classification, focused on genetic programming classification algorithms. Another extension, which adds evolutionary multi-objective algorithms to the framework JCLEC, is showed in Ramírez et al. [106]. However, the latest version of JCLEC is 4.0.0, released in July 2014, and is available at <http://jlec.sourceforge.net/>.

4.10. jMetal (Metaheuristic Algorithms in Java)

jMetal, introduced in Durillo et al. [53] and Durillo and Nebro [52], is an object-oriented framework based on the Java language and specialized in multi-objective optimization problems. In this way, it includes a significant number of classic and modern methods for solving multi-objective optimization problems. Although directed towards multi-objective problems, the jMetal framework also provides implementations of some methods for mono-objective optimization problems.

The main elements and operations of jMetal, which determine the framework structure of jMetal, is defined as follows: an Algorithm solves a Problem using one (possibly more) SolutionSet and a set of Operator objects. The SolutionSet and Solution classes enable the implementation of population-based metaheuristics, representing, respectively, population and individuals. The Algorithm superclass is responsible for representing the optimizers of the framework, allowing the definition of metaheuristics, as well as the parameters and operators involved.

jMetal offers additional features such as Pareto convergence quality indicators, statistical tests and graphical interface. The Quality indicators assessing convergence and uniformity of Pareto Charts implemented are Inverse Generational Distance (IGD), Hypervolume (HV), Epsilon, Spread or D, and Generalized Spread. The graphical interface of jMetal allows the configuration and implementation of experimental studies, as well as determining which statistical tests will be applied in the final results. The following tests are available: Wilcoxon Test (automatically generates script R) and Boxplots (generates a representation of the data in boxplot graphics). jMetal also enables the parallel and distributed execution of the implemented methods. In relation to the hybridization of metaheuristics, there is no possibility of performing this format of implementation.

Some jMetal features are reviewed in Nebro et al. [96]. The main purpose of this review is to simplify the structure and facilitate the use of the framework. This framework is still under development, and its latest version, released in December 2017, called jMetal 5.4, is available at <http://jmetal.github.io/jMetal/>.

4.11. MALLBA

Started in 2000, the MALLBA project, presented in Alba et al. [4], is a skeletons library for Optimization (including exact, heuristics and hybrid methods), which enables parallel execution from sequential implementations. Its three target environments are sequential computers, LANs and WANs workstations. Each optimization method is encapsulated in a skeleton, which is a structure that allows the creation of instances of available methods. The skeletons are implemented in C++ programming language classes.

The MALLBA framework offers features related to parallelism and distributed computing, not requiring the user to have deep knowledge of this subject. Additionally, the MALLBA library also offers tools that allow hybrid skeletons to be built. However, these implementations are closed structures, not offering the user the possibility to define new forms of hybridization. MALLBA LIBRARY v2.0, the latest version of this project, was presented in Alba et al. [3,5]. It is available for download at <http://neo.lcc.uma.es/mallba/easy-mallba/>, released in June 2006.

4.12. OptFrame

The OptFrame framework is introduced in Coelho et al. [37,36] and aims to provide an interface for common elements of metaheuristics based on population and trajectory metaheuristics. Implementations in C++ programming language in simple versions

of these methods are available in OptFrame, allowing the user to develop more efficient versions, according to the specific problem.

An important concept in the OptFrame structure is the Evaluator. It encapsulates the evaluation functions of the solution and enables evaluation of mono or multi-objective functions. OptFrame offers support to parallelism, either with computers with shared memory or distributed memory. It has a parallel generator of the best neighborhood, which allows speed up the process, dividing the search space between the processing nodes.

In the literature, several works show different applications of Optframe for solving optimization problems, among them Coelho et al. [38], Souza et al. [115] and Coelho et al. [39–42]. Optframe is under development and its latest version (version 3.0) from December 2017 is available at <https://github.com/OptFrame/optframe> under the GNU LGPLv3 license.

4.13. Opt4j

A framework metaheuristic optimization of complex optimization tasks, called Opt4j, is presented in Lukasiewycz et al. [83]. Opt4j is a framework for evolutionary computing developed in Java language. The optimization is performed from the decomposition of tasks into sub-tasks that must be designed and developed separately. As these sub-tasks are normally correlated, the framework uses concepts of evolutionary algorithms to separate them from a strict distinction between genotype (genetic representation) and phenotype (representation of a solution). A decoder is used to translate a genotype into a phenotype. In this way, only phenotypes and decoder are problem dependent and need to be specified.

As the framework is focused on methods of evolutionary computing, its basic architecture has, as the main elements, the population and individuals. Operators vary the genotype of the individual to perform the search. In this way, Opt4j does not have the flexibility to develop methods that do not belong to this category and, as a consequence, there are difficulties in the implementation of hybridization.

Opt4j enables the parallel and distributed execution of the algorithms implemented and provides a graphical user interface for ease of configurations and tests. All available modules and methods are automatically listed, and no code is required to view them. The configuration used can be saved or loaded from XML files.

The current version of this framework, named Opt4J 3.1.4, was released in November 2015 and is available at <http://opt4j.sourceforge.net/>.

4.14. Parallel and Distributed Evolving Objects (ParadisEO)

ParadisEO software (Parallel and Distributed Evolving Objects) is a white-box object-oriented framework for reusable design of parallel and distributed metaheuristics, showed in Cahon et al. [31], Liefooghe et al. [79], Melab et al. [93] and Humeau et al. [69]. ParadisEO framework is currently composed of five connected modules:

- ParadisEO-EO: provides a large number of classes for the development of population-based metaheuristics;
- ParadisEO-MO: contains tools for metaheuristics of trajectory or based on a single point;
- ParadisEO-MOEO: is dedicated to the design and implementation of evolutionary techniques for multi-objective optimization;
- ParadisEO-PEO: provides classes for the implementation of parallel and distributed metaheuristics;
- ParadisEO-MO-GPU: presented in Melab et al. [93], the main goal is to explore the power of modern graphics processing unit (GPU), focusing on metaheuristics of trajectory.

The ParadisEO framework stands out for its coverage of both solution techniques and the types of problems and additional resources that exist. Factors such as the GPU module differentiate it from other frameworks in the literature. However, in relation to the possibility of hybridization, despite being reported by the authors as part of the ParadisEO-PEO module, it is restricted to predefined models, limiting, therefore, the definition of new structures.

Melab et al. [93] is the last publication found in the literature regarding the structure definitions of the ParadisEO framework. The latest version of ParadisEO framework (2.0.1) can be found at <http://paradiseo.gforge.inria.fr/>, released in November 2012.

5. Metaheuristics using multi-agent approaches

The demand for even more adaptive and intelligent software has led to the incorporation of new technologies in the treatment of different problems. In this sense, the concept of agents has been widely used for the development of various applications for solving problems, as is the case of optimization area. The approach based on agents is distinguished from others by their power in modeling problems of distributed nature and expressing, through the system entities, the complexity of the involved relationships. This approach is based on the social behavior of human beings or other biological entities, for the design and development of these systems, with emphasis on actions and social interactions.

There is little agreement in the literature regarding the term “agent”, because, for different domains, there are different levels of importance and relevance to each of the properties related to agents. Russell and Norvig [108] define “agent” as:

“An entity that can perceive its environment through sensors (or perceptual systems) and act on it through their mechanisms of action (or actuation systems, actuators). For it, there is a partial representation of this environment and can, in a multi-agent universe, communicate with other agents. The overall behavior of the agent or community of agents is consequence of their perceptions and knowledge, as well as the performed iterations.”

On the other hand, Wooldridge [131] presents the following definition:

“An agent is a computer system that is located in an environment, and is capable of autonomous action in this environment to meet their design objectives.”

In the context of applied methods to optimization, Milano and Roli [94] present the following definition:

“An agent can be understood as a system able to build a solution, move in a search space, communicate with other agents, be active and possibly adaptive.”

A Multi-Agent System (MAS), as Wooldridge [131] states, is characterized by a set of autonomous agents in an environment. Also according to this same author, a Multi-Agent System is composed of agents, each one with a specific function, and a set of goals to be achieved. The sum of the skills of each of the agents, in this system, allows the solution to the problem that is being treated. The constant interaction between agents themselves and between them and the environment define the overall behavior of the system.

Agerbeck and Hansen [1] identify two ways in which the Multi-agent Systems are often applied to the solution of optimization problems: (i) by dividing the constraints and variables of the problem between agents; (ii) as a link between different metaheuristics for solving optimization problems.

However, these two ways do not cover all the different possibilities associated with the application of Multi-agent Systems

for solving optimization problems. For example, one important formulation does not approach by Agerbeck and Hansen [1] is concerning hyper-heuristics, previously discussed in Section 3.3. Another important formulation did not address by this proposal is the MAGMA architecture, defined by Milano and Roli [94] and treated in the current article in Section 5.9. In both cited cases, a metaheuristic is not defined as a single agent, but it comes from the action of different agents distributed in subordinate levels.

From these questions, we propose a classification for Multi-Agent Systems applied to optimization in order to include several related works. Three categories of works are identified in this case:

- (i) Multi-agent Systems with search space decomposition: by dividing the problem constraints and variables between agents, each of them is responsible for optimizing their local perspective and, in the end, the partial solutions are combined for a global solution. This approach is used in works such as Liu and Sycara [80], Jin and Liu [71] and Zheng et al. [133];
- (ii) Multi-agent Systems with metaheuristic decomposition: the elements that compose the metaheuristics are defined as agents. These elements can be low-level heuristics (e.g. constructive and improving heuristics), search strategies (e.g. solution disorder, diversification strategies), solutions (e.g. population members), population or part of populations and/or particles (e.g. ants, birds). In this approach, metaheuristics, hybrid or not, can be seen as the result of the interaction between different types of agents. Aydin [11] and Milano and Roli [94] are examples of works that use these concepts in solving optimization problems;
- (iii) Multi-agent Systems with Metaheuristic Agents: considers each metaheuristic as an autonomous agent and proposes to explore the advantages of using Multi-agent Systems to refine and combine the solutions of these metaheuristics. In this approach, each agent is responsible for performing its own task and, at the same time, use the solutions provided by other agents to improve its result. These agents interact and work together to achieve a predefined goal. Some examples of studies that use this approach are Talukdar et al. [118], Fernandes et al. [57] and Silva et al. [111–113].

According to Aydin [10] and Milano and Roli [94], the use of multi-agent systems in optimization problems solution allows different regions of the search space be treated simultaneously, providing greater diversity and better solutions. These papers also claim that this use allows greater simplicity in implementation, reduces computational time and the involved complexity.

The proposals available in the literature that use the concept of multi-agent systems to metaheuristics are several, regarding the approach used in the organization and form of cooperation between the metaheuristics. The works presented in Sections 5.1–5.11 in the sequel exemplify and demonstrate how this organization can be diverse. Section 6 consolidates a comparison between these works.

5.1. Agent Evolution (AgE)

The AgE framework (Agent Evolution), developed as an open-source project, is a general-purpose platform for running different parallel metaheuristics, oriented to agent-based components. Evolutionary Multi-Agent System (EMAS) is an algorithm for building multi-agent systems, that can be run on AgE (and other platforms), initially presented in Cetnarowicz et al. [34] and later used in several applications, including the solution of combinatorial optimization problems. According to the authors, “*Evolutionary Multi-Agent Systems are a hybrid meta-heuristic which combines multi-agent systems with evolutionary algorithms*” [77,122]. It consists of the formulation of multi-agent systems as evolutionary

processes, in which the agent population can dynamically evolve. In this way, in addition to the typical forms of interaction in a MAS, agents can reproduce (generate new agents), may die (be eliminated from the system) or migrate (in a multi-population case).

The development of this proposal has occurred particularly in works carried out by researchers of the Intelligent Information Systems Group of the Department of Computer Science of AGH University of Science and Technology, Kraków, Poland, under the project Paraphrase AGH (<http://www.paraphrase.agh.edu.pl/>). This project has as its main focus on the development of tools that employ the concept of EMAS, in addition to a framework (AgE) that supports competition and distribution. Revisions regarding this proposal and their implications are in Byrski et al. [29] and Byrski and Kisiel-Dorohinicki [30].

In its development cycle, AgE has been presented in several implementations. Dedicated AgE implementations are building using Java (<http://age.agh.edu.pl> and <https://gitlab.com/age-agh/age3>), Python (<https://github.com/maciek123/pyage>), Scala (<https://github.com/eleaar/scala-mas>), as well as Erlang (<https://github.com/ParaPhraseAGH/erlang-emas>) technologies, associated with different phases of the project, still in progress. The current cycle shows the use of functional languages, such as Erlang (<http://www.erlang.org/>) and Scala (<http://www.scala-lang.org/>). The authors argue for the need to change paradigms of development of computing platforms, in order to use the availability of new hardware possibilities in terms of multi- and many-core computers. According to Turek et al. [122], “*Efficient development of software for many-core architectures will require high level functional languages with immutable variables, no shared state and message-passing concurrency.*”. In Turek et al. [122] the dedicated Erlang AgE implementation is showed, while Krzywicki et al. [77] showed a comparison between the AgE implementations with Erlang and Scala.

The description in the sequel is concerned the dedicated Erlang AgE implementation. The AgE framework is divided into 4 elements: MAS application, execution environment, migrations broker and gathering statistics. The multi-agent application is directed to the interactions, defining: agent types, possible states and actions, state/action mapping, and functions for updating sub-populations. The execution environment implements the computational logic of these functions and links the results to the simulations. The migrations broker module is responsible for the migration of agents between environments. The last module is responsible for collecting the statistics and metrics.

An agent, in the EMAS implementation, is defined by a vector of real values representing a solution of the problem to be solved. Each agent receives from his parents an initial amount of energy. Energy is here a non-renewable resource and will allow the selection of agents. Agents that lose all their energy are removed from the system. Although the number of agents may vary, the energy of the system remains constant. The population of solutions is divided into sub-populations, seeking to preserve diversity. Each sub-population is called island. The interaction between the agents in this context is accomplished through meeting arenas. Each agent enters an arena according to his amount of energy. In this way, the dynamics of the system is defined as follows: the agents enter the arenas according to the energy and behavior of each one; in sequence, the meeting operation is applied in each arena. According to Krzywicki et al. [77], the pattern used is “*very flexible, as it can be implemented in both a centralized and synchronous way or a decentralized and asynchronous one*”.

Four different forms of interaction between agents are implemented in order to define the best model to achieve efficient parallelism in the multi-agent system:

- (i) Sequential: in this version, all populations are involved in a recursive loop within a single Erlang process. Functions are repeatedly applied to update the agent population and, at each step, agents can migrate between populations with a low probability. The Erlang process runs over a collection of islands.
- (ii) Hybrid: in this version, each agent population is contained in a separate Erlang process and runs in a separate loop. Agent migration occurs through message passing. A single, separate process acts as a migration intermediary. The probability of migration is very low.
- (iii) Skeletons: in this version, the SKEL-based implementation is the result of the introduction of SKEL library skeletons in the sequential version.
- (iv) Concurrent: in this version, each agent executes in an autonomous Erlang process and interacts only through the mediating arenas. In this model are present the arenas responsible for each type of interaction: fight, reproduction, death and migration. Each agent population has a separate set of arenas.

According to the URL addresses above, the last update for the Java AgE3 was released in June 2017; for the Python AgE, the last update was released in May 2016; for the Erlang AgE, the last update was released in April 2015; and for the Scala AgE, the last update was released in August 2017. A new stage in development of this project presents the proposal of Memetic agent-based paradigm, which, according to Żurek et al. [134], “*combines evolutionary computation and local search techniques*”.

5.2. AMAM

AMAM framework, whose conceptual model was proposed in Silva [110], is a Multi-agent Architecture for Metaheuristic, in which each agent implements a metaheuristic. The environment, where agents exist and act, corresponds to the search space of the optimization problem to be solved. Thus, by changing the problem to be solved, it is only necessary to make a simple change of the architecture environment. The movement ability of the agent through the search space of the problem is defined by the neighborhood structures (movements) that it has.

The scalability of the AMAM architecture is guaranteed by the ease of adding new agents, with minimal impact on the rest of the architecture. These agents interact with the environment and with others agents cooperatively, exchanging and sharing information about their condition and about the environment. The software developed from the conceptual model is presented in Fernandes et al. [57] and allows the creation of an instance of the environment and multiple agents for the search of the solution. Design patterns are used to ensure that the AMAM architecture is flexible and extensible.

In this initial conceptual model, six major elements make up the proposed Multi-agent System: (i) Environment; (ii) Constructor Agent; (iii) Local Search Agent; (iv) Metaheuristic Agent; (v) Coordinator Agent; and (vi) Solution Analyzer Agent.

The cooperative structure of AMAM architecture is reviewed and improved by Fernandes et al. [57] and Silva et al. [112]. An adaptive memory strategy called Pool of Solution is used for sharing information. The available solutions are stored in this Pool, located in the Multi-Agent System environment.

In the latest proposal, presented in Silva et al. [112,113], the Coordinator and Solution Analyzer agents were removed from the architecture, in relation to the original proposal. The main objective of this change in the structure is to meet the need to increase the autonomy of the agent, preventing other agents interfere in its activities. The new structure for AMAM framework is composed of three main elements:

- (i) Environmental Stimulus Pool: responsible for storing the stimuli sent by the agents, which can be the delivery of solutions that have been generated or the request of solutions. It allows the interaction between agents;
- (ii) Pool of Solutions: responsible for maintaining the solutions available to all agents;
- (iii) Metaheuristic Agents: responsible for guiding the search for the solution.

In Silva et al. [113], self-adaptive skills based on reinforcement learning are assigned to framework agents. These skills allow the agent to modify his actions based on his own experience of interacting with the environment and with the other agents involved in the solution of the problem.

The hybridization of metaheuristics is also guaranteed in the AMAM framework, through iteration between the different heuristic/metaheuristic agents. Additionally, AMAM offers the possibility of parallel execution, in which each agent runs on a separate thread.

AMAM framework applications to optimization problems have been shown in Fernandes et al. [57], Silva et al. [112] and Silva et al. [113]. This framework is under development and its most recent update was presented in Silva et al. [113]. The latest version of the AMAM framework, released in December 2017, is available at <https://github.com/mamelials/AMAM-Multiagente-Architecture-for-Metaheuristics>, under the GNU LGPLv3 license.

5.3. Agent Metaheuristic Framework (AMF)

Meignan et al. [89–92] propose the Agent Metaheuristic Framework (AMF). The main objective is to use the agent-oriented approach to support the design and hybridization of metaheuristics. The AMF is based on an organizational model that describes a metaheuristic in terms of roles. The organizational model used as the reference for the AMF proposal is based on the RIO meta-model. The RIO meta-model involves three basic concepts:

- (i) Role: abstraction of a behavior or status in an organization;
- (ii) Interaction: links between two roles, so that an action on the first role produces a reaction on the second;
- (iii) Organization: set of roles and their interactions.

The roles represent the main elements that compose metaheuristics, such as intensification, diversification, memory, and adaptation or self-adaptation. According to the authors, “*to obtain a metaheuristic from the AMF organizational model, it is necessary to refine the different roles and determine the multiagent structure of the optimization system*”.

To illustrate the use of the AMF model, a metaheuristic called Coalition-Based Metaheuristic (CBM) and its application to the Vehicle Routing Problem are presented. CBM is a metaheuristic based on the metaphor of a coalition. According to the authors, “*the coalition is composed of several agents which have the capacity to individually treat the optimization problem but cooperate to coordinate and improve the search*”.

The CBM system is composed of a number of identical agents. The CBM agent handles a single solution and uses multiple operators to move in the search space. These operators are related to the tasks of intensification and diversification.

An adaptive strategy is used by the CBM agent, through a learning mechanism, to select the most appropriate operator, based on the context. Cooperation between these agents occurs in two ways: (i) an agent shares its best-known solution; (ii) an agent shares its internal decision rules in order to allow mimicry of behavior. Consequently, the hybridization of metaheuristics comes from this cooperation among agents.

The latest publication found in the literature on the AMF framework is Meignan et al. [92]. The versions of this framework, presented in the literature, are not available for download.

5.4. A-Teams conceptual model

In Talukdar and Souza [119] and Talukdar et al. [117,118], the conceptual model of a multi-agent architecture for the solution of problems, called Asynchronous Teams (A-Teams), is introduced. In this architecture, a set of autonomous agents and interconnected memories form a cyclic data flow network in which the found solutions continually circulate through it.

Two types of agents modify the solutions stored in the memories:

- (i) Builders: add new solutions to the memories;
- (ii) Destructors: eliminate solutions of memories.

In this architecture, independent agents work in parallel and cooperate, modifying, each one, the solutions of the others. Each agent implements a problem-solving algorithm and each memory is dedicated to a problem.

A-Teams architecture has been used as basis for several works, such as in Lukin et al. [85], Rabak and Sichman [104], Landa-Silva and Burke [78], Carle et al. [33], Barbucha et al. [14], Barbucha et al. [16] and Barbucha [13], among others. Executable versions of this architecture, in its original version, have been implemented only for specific case studies, as shown in Talukdar et al. [118], that presents many applications for scheduling production problems. In addition, this structure was used as the basis for setting new metaheuristics based on agents, such as the case of the papers cited above. Barbucha et al. [16] presented a critical review of the application over the years of A-teams architecture.

Barbucha et al. [14,16] introduce a tool for the construction of A-Teams architectures in order to solve different optimization problems, named JADE-based A-Team environment (JABAT). This framework produces solutions to combinatorial optimization problems using a set of optimization agents, each implementing a solution algorithm. JABAT is based on JADE [18], one of the most important middlewares for building multi-agent systems using JAVA, providing fundamental communication infrastructure and visualization tools.

The JABAT framework shows a good set of additional features to improve the performance of the search process, such as parallel and distributed execution; extension accessible via the web; and use of reinforcement learning. In distributed execution, the optimization process can be performed on many computers. The user can add or delete computers involved in the optimization process. In these cases, JABAT adapts to the changes, commanding the optimization agents. In addition, the hybridization of metaheuristics is possible in the JABAT from the interaction between the agents defined by the conceptual model of A-Teams. However, this interaction is regulated by the JADE middleware.

An extension of the JABAT framework, named ejABAT, is proposed in Barbucha et al. [15]. The ejABAT extension is designed to be fully accessible via the internet, allowing, through a simple interface, access to troubleshooting and adding new computers to the system. The last extension of this framework, named Cooperative JADE-based A-Team (Cooperative JABAT), is also proposed in Barbucha [12], introducing a Reinforcement Learning mechanism. In this case, the reward is assigned to the optimization agents each time they find a better solution than the one previously found. A negative reward can also be awarded if the agent fails to improve a solution. The weight of each agent in this context is taken into account when solutions are requested in the common memory. In this way, the requested solutions are not immediately sent after the

request of each agent, but only after all agents make such requests, and taking into account the weights.

It is important to note that none of the JABAT versions described here are available for download.

5.5. Collaboration of Metaheuristic Algorithms (CMA)

Malek [86] also used the agent-based approach on a framework, named Collaboration of Metaheuristic Algorithms (CMA), for the execution of various metaheuristics, simultaneously, in order to solve combinatorial optimization problems. Each metaheuristic is implemented as a black box, with inputs and outputs, wherein the input (solutions) is converted to a specific representation of the implemented algorithm.

This proposal uses a framework for multi-agent systems, called A-Globe [114] as a basis for its development. A-Globe offers multi-agent features like skeleton agents, messaging, chat protocols, etc. According to the concept presented by the author, “*the system can be immediately decomposed into blocks such as particular metaheuristics, (global) solution pool and others, each block is handled by its agent*”. This collaborative concept used in this framework is similar to MAGMA architecture, already presented in Section 5.9. The result of the search for a particular metaheuristic is also shared by the global populations of candidate solutions stored in a pool of solutions.

Four types of agents make up the system of this framework:

- (i) Problem Agent: responsible for (i) receiving a request to start the optimization, sent by a user or an agent; (ii) reading the problem data from a configuration file; (iii) initializing the other agents; (iv) measuring the total optimization time; and (v) communicating to agents the end of the search;
- (ii) SolutionPool Agent: responsible for managing the population of candidate solutions and providing solutions to other agents;
- (iii) Algorithm Agent: responsible for implementing and executing a specific metaheuristic;
- (iv) Adviser Agent: responsible for providing configuration parameters to metaheuristics and receiving reports after their execution. Each Algorithm Agents class has an Adviser Agent associated with it.

In his latest article, Malek [87] uses the concepts of this framework in defining a structure for hyper-heuristics. In this proposal, a feature responsible for setting the parameters is added via an object so-called `Policy`.

Malek [87] uses the concepts of this framework in defining a structure for hyper-heuristics. In this proposal, a feature responsible for setting the parameters is added via an object called `Policy`. The goal is to provide new dynamic parameters, from the feedback of the algorithm itself. Thereby, the reports sent to the Adviser Agent are evaluated and used to determine the new algorithm input parameters. The possibility of parallel and distributed execution of the algorithms is also included in this proposal.

The last publication found regarding CMA is Malek [87]. The release of an updated version was also not found.

5.6. Distributed Agent Framework for Optimization (DAFO)

Danoy et al. [48,49] propose a multi-agent system, directed towards evolutionary optimization, called DAFO (Distributed Agent Framework for Optimization). According to the authors, “*a key element of this framework resides in its multi-agent organization model, MAS4EVO (Multi-Agent System for EVolutionary Optimization), which is an extension of the Moise+ model [68]*”. The main goal of the MAS4EVO model is to overcome the limitations in agent-based frameworks by defining a structure and interactions based

on Coevolutionary Genetic Algorithms (CGAs). Unlike Genetic Algorithms (GAs), where a population of similar individuals evolves to represent a global solution, in the CGAs subpopulations of individuals coevolve, representing specific parts of the global solution that evolve independently.

The DAFO multi-agent environment is defined and represents the optimization problem to be solved. Three types of agents comprise the structure:

- (i) Problem Solving Agents (PSA): responsible for seeking the solution to the problem using a metaheuristic;
- (ii) Fabric Agents (FA): responsible for instantiating and configuring the executed application;
- (iii) Observation Agents (OA): responsible for observing Problem Solving Agents and providing outputs to users.

The interaction between agents is defined by communication protocols (FIPA ACL and FIPA-SL propositions). These protocols structure and express the message sequence according to the solution exchange strategy used (best, random, etc.) among the GAs implemented by the agents. The MAS4EVO organization model specifies the global patterns of cooperation on agent behavior, defining a multi-agent organization based on coevolutionary strategies. This form of organization enables the agent to change and reorganize the overall functioning of the system.

This interaction between the agents enables the hybridization of metaheuristics. However, the other additional features of interest in this analysis that allow better search performance, such as parallel and distributed execution, are not covered by the DAFO framework.

The latest version of this framework shown in the literature is described in Danoy et al. [49] and is not available for download.

5.7. Learning-Based Multi-agent System (LBMAS)

In Lotfi and Acan [81,82], Learning-Based Multi-Agent System (LBMAS) for solving combinatorial optimization problems is presented. This system allows collaboration between metaheuristic agents on a common solution population and a two-stage file also common to agents. In this way, different regions of the search space are exploited using the most effective agent at the moment.

The LBMAS architecture includes seven Metaheuristic Agents and four System Agents (Problem Agent, Solution-Pool Agent, Manager Agent and Archive Agent), which are described below.

- Metaheuristic agents: each metaheuristic agent acts with its own search strategy and provides the solutions found. The seven metaheuristics implemented by metaheuristic agents are Genetic Algorithms (GAs), Differential Evolution (DE), Simulated Annealing (SA), Ant Colony Optimization (ACO), Great Deluge Algorithm (GDA), Tabu Search (TS), and the Cross Entropy (EC) method;
- Problem agent: responsible for initializing the input parameters of the problem;
- Solution Pool agent: responsible for handling all transactions with the common solution population;
- Archive agent: responsible for the operations of recovery and manipulation of the common two-stage archive;
- Manager agent: responsible for passing the solutions to the Solution-Pool and Archive agents, which, in turn, update the populations of shared solutions.

At each iteration of the search, the metaheuristic to be used in the next iteration is chosen. This selection is carried out using the concepts of the Russian roulette, in which the evaluation values of metaheuristics are obtained based on the improvement levels of the objective function of each of the metaheuristics. All metaheuristics

initially have the same probability of being chosen and this probability changes (grows or decreases) according to the individual performance of the agents.

In LBMAS, agents cooperate by sharing their individual experiences through a two-stage external memory archive: the first stage stores promising solutions based on their evaluation value; and the second stage maintains the solutions according to their spatial distribution, based on a defined dissimilarity measure. However, although the agents share their experiences, at each iteration only one metaheuristic is performed, and thus the hybridization can occur only sequentially, with information being passed from one iteration to the next.

The latest publication about the LBMAS framework is Lotfi and Acan [82] and a downloadable version was not available.

5.8. Multi-agent Cooperative Search (MACS)

Martin et al. [88] introduce a distributed agent-based framework called Multi-agent Cooperative Search (MACS). This framework is also implemented using the JADE platform [18]. In this proposal, each agent performs a different combination of local search metaheuristics/heuristics and adapts continuously throughout the search process using a cooperation protocol. This cooperation protocol allows communication between the agents involved in the solution search.

MACS has two types of agents: (i) launcher agent: responsible for preparing the problem to be solved by the other agents, converting the problem to the proposed protocol of messages agent; (ii) metaheuristic agent: performs a combination of available local search metaheuristics/heuristics, so that each agent can use a different set of configuration parameters. The launcher agent defines the combination of local search metaheuristics/heuristics and the configuration parameters used by metaheuristics agents.

The main elements of the basic structure of the MACS framework state the basis for the framework communication protocol and, according to the authors, “*it defines a set of general representational primitives that are used to model a number of scheduling and routing problems*”. These elements are:

- SolutionElements: abstract objects that represent specific elements of the problem, such as a client in the Vehicle Routing Problem;
- Edge: contains two SolutionElements objects. These pairs are used in the permutation performed in the cooperation protocol to identify good standards;
- Constraints: interface between the framework and the specific constraints to the problem;
- NodeList: list of SolutionElements objects or of Edges;
- SolutionData: list of NodeList objects.

An iteration of the communication protocol is here called a conversation. Then, the communication between agents takes place as follows: good patterns that compose improvement solutions are identified, according to the frequency that they occur in the conversation, and are then shared among the agents. To do this, each metaheuristic agent breaks the current solution into edge objects and sends them to the launcher agent, which brings together all the edge objects and punctuates them according to their frequency. These elements are shared with the other agents and will be part of the next solutions. Therefore, the hybridization of metaheuristics may occur in the MACS structure in a different way than in other multi-agent structures, since in this proposal only parts of the solution are shared, not the solution as a whole.

The concept of learning is also used in the construction of the MACS framework. The implemented learning mechanism allows each agent to maintain a short-term memory of good edges, which

are used at the beginning of each conversation to influence how new solutions are constructed. The edges identified by the learning are used to reorder the lists of elements to be inserted into the solutions.

The MACS framework is available as an open source project at <http://simonpmartin.github.io/mac/>, and its last update was released in April 2016.

5.9. MAGMA

MAGMA architecture, presented by Milano and Roli [94], uses the multi-agent perspective in defining a structure in which a metaheuristic can be seen as the result of the interaction between the various agents of the structure itself. The agents in this architecture work at subordinate levels, each level corresponding to an action of the metaheuristic:

- (i) Level 0: Solution Builder Agents implement constructive heuristics;
- (ii) Level 1: Solution Enhancer Agents implement local search heuristics;
- (iii) Level 2: the Strategy Agents implement the strategies used to escape from local optima;
- (iv) Level 3: the Coordination Agents coordinate the search process and the lower level agents.

The communication between two levels, done through the interaction between the agents, is divided into two types:

- (i) horizontal: involves the interaction between same level agents;
- (ii) vertical: involves the interaction between different level agents.

An example of an application of MAGMA architecture in which there is cooperation between the Memetic algorithms and GRASP (Greedy Randomized Adaptive Search Procedures) metaheuristics is presented in Milano and Roli [94]. The purpose of this example is to show how the cooperative search is achieved by the addition of different perspectives and the exchange of information. The GRASP metaheuristic is set on three levels: the first one generates initial solutions using the Greedy Random method of construction; the second one runs a local search algorithm; the third one stores the best solution found. The Memetic Algorithm metaheuristic is also defined on three levels: at level 0, several agents generate solutions to the initial population; at level 1, agents improve the initial solutions using local search; and at level 2, the recombination and mutation operators are applied, generating new solutions. The cooperation takes place through the exchange of information of states, solutions and other search space characteristics.

It is important to note that, in the proposal of MAGMA architecture, a metaheuristic is not a single agent operating independently, but arises from the cooperative action of different agents performing at various levels of the architecture abstraction. MAGMA architecture was not available in executable version, remaining, so far, in the concept version shown in Milano and Roli [94].

5.10. MultiAgent eNvironment for Global Optimization (MANGO)

An environment for optimization that uses the concept of multi-agent systems, called MANGO (MultiAgent eNvironment for Global Optimization), is presented by Kerçelli et al. [75] and Aydemir et al. [8]. MANGO system is a project in development and provides an environment in which each agent performs a search algorithm.

Cooperation between agents is one of the main elements of the framework. The agents communicate and cooperate with each other through a communication language. The system provides

protocols of cooperation and organizational models to accomplish communication, as well as the possibility of registration of new agents.

In previous versions of this framework, presented in Kerçelli et al. [75] and Günay et al. [67], the JADE platform (a framework for development and implementation of multi-agent applications, introduced in Bellifemine et al. [17]) was used for the implementation of multi-agent system MANGO framework. Currently, as shown in Aydemir et al. [8], a specific multi-agent environment, also based on JADE, was developed.

In the MANGO environment, the agents are defined from MANGO API, which uses Java language in their implementation. These agents can be run either in a distributed way, in a distributed network platform, or on a single computer. Cooperation is carried out based on the concept of Service Oriented Architecture (SOA). Communication is implemented at two levels: (i) one level, made under Java Messaging Service (JMS), is responsible for networks structures and protocols; (ii) the other level, built on the first, is responsible for the exchange of messages between agents.

The exchange of messages among the framework agents can occur in two ways: interrupts and mailboxes. In the exchange of messages based on interrupts, when a message is sent to an agent, this will immediately stop the search process. In the exchange of mailboxes based messages, the message is stored, and the agent will check its mailbox when is more appropriate. Additionally, the directory agent (DA) is defined, which is mainly responsible for the management of communication resources.

The communication between agents as described above allows the hybridization of metaheuristics in the MANGO framework. The latest version of MANGO environment was released in October 2013 and is available at <http://algopt.sabanciuniv.edu/mango/>.

5.11. Swarm of Metaheuristic Agents (SMA)

Evolutionary algorithms are also widely used in the definition of applications involving the concept of agents. This is a common sense use of the concept of agents, associating it directly with their own structures of evolutionary algorithms, like genes, memes, particles or ants, or other bio-inspired structures.

Along this line, three work groups can be identified: (i) the first one considers that the elements of the evolutionary algorithm (e.g., individuals of the population or solutions) are defined as agents; (ii) the second approach considers that the elements of the evolutionary algorithm (individuals of the population), non-agents, act cooperatively in the search for solution of the problem; (iii) the third group of proposals uses evolutionary algorithms to manage a group of metaheuristics agents, in such a way that each agent implements a method for finding solutions. This last work group relates directly to hyper-heuristics formulations presented in Section 3.3.

Aydin [9–11] present a framework proposal involving agents that use swarm intelligence for coordination of search agents. In this proposal, each metaheuristic is defined as an agent and maintains its own intelligence and employs it in the solution of the problem. The Particle Swarm Optimization (PSO) algorithm is used to coordinate the agents and give support to the diversification of the search.

In order to exemplify, as shown in the latest experiment described in Aydin [11], a population of agents is defined, so that each agent has search skills, based on Simulated Annealing (SA) metaheuristic, as well as communication capabilities. These agents are organized through a swarm that cooperates to solve the problem. In each swarm generation of Simulated Annealing agents, the agents execute their search algorithms. A cooperation algorithm is applied, after each generation, exchanging information related to the obtained solutions.

This form of multi-agent organization restricts the possibilities of interaction between agents to the used population algorithm format. Therefore, the hybridization of metaheuristics is also limited to these interactions. The framework code is not available for download, and new versions of the proposal have not been found in the literature.

6. A comparative analysis and discussion

In this section, a comparative analysis of the metaheuristic optimization frameworks described in Sections 4 and 5 is presented. Table 1 summarizes these descriptions, presents the list of analyzed frameworks, as well as relevant information regarding the publications, versions, and sites where they can be found. In this table, the symbol “–” means that the associated framework does not have an internet site for its publication.

The goal in the current section is to identify which of the desirable characteristics in a framework for optimization are available in the proposals reported in the previous sections. In particular, we seek to discern which ones make possible, to its users, the development of hybrid metaheuristics, focusing on cooperatives and parallel approaches. A deep discussion concerning the comparative analysis is introduced in Tables 2–5 and Table 7. Every one of these tables is in the sequel dissected, in order to understand the information details contained therein.

For the evaluation to be carried out, the following aspects were analyzed: (i) basic and advanced characteristics of a metaheuristic optimization framework; (ii) characteristics related to the support of the optimization Process; and (iii) hybridization-based on multi-agent systems. These aspects were developed according to Talukdar et al. [118], Milano and Roli [94], Silva [110], Parejo et al. [101] and Aydin [11].

In the case of a Multi-Agent Architecture for Combinatorial Optimization, some desirable aspects are identified, which are also used as reference in the evaluation carried out in the present comparison:

- (i) agents must be autonomous;
- (ii) the environment must be considered, and modeled, as a primary abstraction, which provides the conditions of existence of the agents, in the sense of its autonomy [108,109];
- (iii) the architecture must be flexible, allowing to develop applications for different problems of combinatorial optimization;
- (iv) the interaction of agents (communication) must be asynchronous, to ensure its autonomy;
- (v) the architecture should be based on individual and collective mechanisms of action of the agents, contemplating mechanisms of cooperation, *ad hoc* or not, and of competition and selection of the fittest agents;
- (vi) the application of the architecture must be performed in a distributed computer system and, eventually, heterogeneous. Additionally, this application, if using a single processor, might have each agent defined in a separate thread, so as to ensure their independence and asynchronicity;
- (vii) the addition/removal of new agents should be possible at runtime;
- (viii) should be desirable that new heuristics and metaheuristics could arise from the cooperation between agents, without interference or pre-programming of the designer/software operator;
- (ix) the agents (or the system) must have a dynamic and persistent memory of the environments, to enable the start of its activities from “good solutions”;

Table 1

Summary of analyzed frameworks.

	Framework	Type	Main references	Latest release/publication and date	Site
1	AgE	Multi-agent Framework for Evolutionary Optimization	Turek et al. [122]	v.23, October, 2015	https://github.com/ParaPhraseAGH/labs-emas
2	AMAM	Multi-agent Framework for Optimization	Fernandes et al. [57], Silva et al. [112,113]	Silva et al. [113], November 2015	–
3	AMF	Multi-agent Framework for Evolutionary Optimization	Meignan et al. [89–92]	Meignan et al. [92], 2010	http://www.lalea.fr/?page_id=45
4	CMA	Multi-agent Framework for Optimization	Malek [86,87]	Malek [87], 2010	–
5	DAFO	Multi-agent Framework for Evolutionary Optimization	Danoy et al. [48,49]	Danoy et al. [49], 2010	–
6	EasyLocal++	Framework for Local Search	Gaspero and Schaefer [61–63]	v3.0, February 2018	https://bitbucket.org/satt/easylocal-3
7	ECJ	Framework for Optimization Evolutionary	White [129]	v.25, November 2017	http://cs.gmu.edu/~eclab/projects/ecj/
8	EvA2	Framework for Optimization Evolutionary	Kronfeld et al. [76]	v2.2.0, December 2015	http://www.ra.cs.uni-tuebingen.de/software/EvA2/
9	FOM	Framework for Optimization	Parejo et al. [100]	v0.5, July 2013	http://www.isa.us.es/fom
10	HeuristicLab	Framework for Optimization	Wagner and Affenzeller [125,126], Wagner et al. [127,128]	v3.3.15, January 2018	http://dev.heuristiclab.com/
11	HotFrame	Framework for Optimization	Fink et al. [59], Fink and Voß [58]	Fink and Voß [58], July 2003	–
12	HyFlex	Framework of Hyper-heuristics for Optimization	Ochoa et al. [97]	v1.1, September 2011	http://www.asap.cs.nott.ac.uk/external/chesc2011/
13	JABAT	Multi-agent Framework for Optimization	Barbucha et al. [14,16]	Barbucha et al. [16], 2010	–
14	JAMES	Framework for Local Search	De Beukelaer et al [50,51]	v1.1, August 2016	http://www.jamesframework.org/
15	JCLEC	Framework for Optimization Evolutionary	Ventura et al. [123], Cano et al. [32]	v4.0.0, July 2014	http://jclec.sourceforge.net/
16	jMetal	Framework for Multi-Objective Problem	Durillo et al. [53], Durillo and Nebro [52]	v5.4, December 2017	http://jmetal.github.io/jMetal/
17	LBMAS	Multi-agent Framework for Optimization	Lotfi and Acan [81,82]	Lotfi and Acan [82], June 2015	–
18	MACS	Multi-agent Framework for Optimization	Martin et al. [88]	v1.0, April 2016	http://simonpmartin.github.io/mac/
19	MAGMA	Multi-agent Framework for Optimization	Milano and Roli [94]	Milano and Roli [94], August 2002	–
20	MALLBA	Skeletons Library for Optimization	Alba et al. [4,3,5]	v2.0, June 2006	http://neo.lcc.uma.es/mallba/easy-mallba/
21	MANGO	Multi-agent Framework for Optimization	Kerçelli et al. [75], Günay et al. [67], Aydemir et al. [8]	v1.0, October 2013	http://alopt.sabanciuniv.edu/mango/
22	OptFrame	Framework for Optimization	Coelho et al. [36,39]	v3.0, December 2017	https://github.com/OptFrame/optframe
23	Opt4j	Framework for Optimization Evolutionary	Lukasiewycz et al. [83]	v3.1.4, November 2015	http://opt4j.sourceforge.net
24	ParadisEO	Framework for Optimization	Cahon et al. [31], Liefooghe et al. [79], Melab et al. [93], Humeau et al. [69]	v2.0.1, November 2012	http://paradiseo.gforge.inria.fr/
25	SMA	Multi-agent Framework for Optimization	Aydin [9–11]	Aydin [11], September 2013	–

- (x) the architecture should be able to deal with agents or companies of agents (groups), as well as punctual or population methods;
- (xi) the architecture should be designed at a high level of abstraction, therefore, being robust, scalable and, moreover, allowing its specific applications be easily developed.

Based on the benchmarks introduced above, a set of characteristics were identified and used as guidelines for the comparative analysis of the frameworks described in Sections 4 and 5. This set of characteristics is shown in Section 6.1 in the following.

6.1. Analyzed characteristics

The comparison between the selected frameworks was performed considering characteristics put into evidence from Parejo et al. [101] and Silva [110], as well as characteristics used by Talbi [116] and Raidl [105] to classify the hybrid metaheuristics. Based on

these characteristics, twenty-two aspects, grouped in four different categories, were analyzed in each selected framework:

- (a) General characteristics for Frameworks;
- (b) Advanced characteristics of Frameworks;
- (c) Multi-Agent characteristics;
- (d) Support features to Optimization Process.

Sections 6.2–6.5 present the characteristics analyzed in each of the four categories, as well as the obtained results of the evaluation of the frameworks described in Sections 4 and 5. In addition, a deep discussion concerning the comparative analysis is presented, focusing on its meaning for the better understanding as to how to undertake the development of a framework using the concept of multi-agent.

In the tables presented in the following sections, for better comprehension, CMA stands for Collaboration of Metaheuristic Algorithms, and SMA stands for Swarm of Metaheuristic Agents. The frameworks are presented in these tables in alphabetical order.

6.2. General characteristics for frameworks

The general characteristics of Metaheuristic Optimization Frameworks are listed below. These characteristics define the general issues of frameworks, like the class of problems they treat, the class of metaheuristics they support and which metaheuristics are previously implemented. These general characteristics are defined as:

- (i) Problem Class: if the framework only supports problems with a single objective function – SOOP (Single Object Optimization Problem) – or if it supports problems with multiple concurrent objective functions – MOOP (Multi-Objective Optimization Problem);
- (ii) Implemented Metaheuristics: list of metaheuristics which have already been implemented in the framework;
- (iii) Population-based Heuristics: if the framework implements population-based heuristics;
- (iv) Single Solution-based Heuristics: if the framework implements single solution-based heuristics.

Table 2 presents the results concerning the analysis of the general characteristics for the Metaheuristic Optimization Frameworks described in Sections 4 and 5. As can be concluded from this table, among the twenty-five evaluated frameworks, only eight (ECJ, EvA2, HeuristicLab, JCLEC, jMetal, OptFrame, Opt4j, ParadisEO) implement the solution of mono-objective problems as well as multi-objective problems. The other frameworks are only addressed for mono-objective problems.

Regarding implemented methods, we can say, also from **Table 2**, that eleven frameworks are focused on specific optimization techniques. EasyLocal++, JABAT, JAMES, MACS and MANGO frameworks implement single solution-based techniques and AgE, DAFO, ECJ, JCLEC, jMetal and SMA frameworks are population-based techniques. In this context, these frameworks can be viewed as specialized only for these techniques, not be possible to extend them beyond these specified methods. EvA2 and Opt4j frameworks are also specialized for population-based techniques, although some single-solution techniques are also implemented. The Hyflex framework is specialized for the development of hyper-heuristics but also does not offer flexibility in the extension for different methods of the initial proposal for which it was developed.

The frameworks that implement larger number and a greater variety of methods are HeuristicLab and ParadisEO. These frameworks, in this sense, can be considered the most complete among those evaluated in this analysis. The ParadisEO framework is the one that presents the greatest number and diversity of methods implemented, including both evolutionary algorithms and trajectory methods, and, furthermore, presenting the possibility of handling both mono and multi-objective problems. On the other hand, the MAGMA framework remains only as a conceptual model, not being known, at least by the authors of this revision, executable versions of this framework until this moment.

6.3. Advanced characteristics for framework

The advanced characteristics of the Metaheuristic Optimization Frameworks are listed in the sequel. These characteristics define the main issues concerning the implementation of frameworks, and, therefore, their ability to adapt to new challenges, especially changes in technological paradigms that may arise over time. These advanced characteristics are defined as:

- (i) Implementation language: which is the language used in the implementation of the framework;

- (ii) Portability: if the framework can be compiled/run on different platforms;
- (iii) Licensing Model: type of software license used;
- (iv) Hybridization: ability of the framework to enable hybridization of metaheuristics;
- (v) Support for parallel and distributed computing: in a parallel application, subtasks are executed simultaneously on one or more processors. For this purpose, parallel and distributed programming resources must be available. In this analysis, we evaluated in the frameworks the availability of threads and resources for distributed computing.
 - a. Threads: if the framework provides resources for application development using threads. Threads are the subtasks of a parallel program that allow multiple runs to occur in the same application environment on a computer;
 - b. Distributed Computing: if the framework provides distributed programming resources for application development. Distributed programming consists of running competing applications on different machines;
- (vi) Hyper-Heuristic: ability of the framework to enable the development of hyper-heuristics;
- (vii) Design in high level of abstraction: the used project resources should enable the development of flexible and reusable components.

Table 3 presents the results concerning the analysis of the advanced characteristics for the Metaheuristic Optimization Frameworks described in Sections 4 and 5. From this table, it is clear that the Java language is used in the development of most of the analyzed frameworks. This is mainly due to the portability of the language, that is, using Java it is possible to develop applications for various devices, such as cellular, televisions, among others. Likewise, it is possible to develop an application on any operating system that can be executed by another operating system. In addition, **Table 3** shows that eleven frameworks analyzed are available under the free software license. HeuristicLab, JCLEC, MACS frameworks are under the GNU General Public License (GPL) software license, which defines that any work derived from the original must be made available under the same terms as the original license. EvA2, FOM, jMetal, OptFrame, Opt4j and ParadisEO frameworks are under the GNU Lesser General Public License (LGPL), which is similar to the GPL license, but adds some license permissions, such as permission to join to the software that are not under the GPL or LGPL licenses, including the possibility of association with proprietary software. The AgE and JAMES frameworks are available under Apache License, which allows the use and distribution of the source code and is compatible with the GPL license.

Hybridization is evaluated, in this analysis, by two ways: (i) the ease and flexibility that the framework provides to develop hybrid methods, shown in **Table 3**; and (ii) if it is possible to hybridize metaheuristics from the interaction between the existing agents in the framework, mainly through cooperation, presented in **Table 4**.

Table 3 highlights three different situations in relation to hybridization. The first situation refers to the frameworks marked “Yes” in the respective column of **Table 3**. This set of frameworks (AgE, AMAM, AMF, CMA, DAFO, JABAT, jMetal, LBMAS, MACS, MANGO, and Optframe) are the proposals that effectively allow the hybridization of heuristic and metaheuristic methods in a broader context, that is, in these cases, the hybridization technique can be used without restrictions. It is important to emphasize that, except for the JMetal and Optframe frameworks, all these frameworks (AgE, AMAM, AMF, CMA, DAFO, JABAT, LBMAS, MACS, and MANGO) use the concept of agents in their structuring, as shown in **Table 4**. The second situation refers to the frameworks marked “*predefined hybridization*” in the hybridization column of **Table 3**. This is the situation in which hybridization can only be

Table 2

General characteristics of metaheuristic optimization frameworks.

Framework		Characteristics			
		Problem class	Implemented methods	Population	Trajectory
1	AgE	Mono-Objective	Genetic Algorithms	Yes	No
2	AMAM	Mono-Objective	Iterated Local Search, Variable Neighborhood Search, GRASP, Genetic Algorithms	Yes	Yes
3	AMF	Mono-Objective	Evolutionary Algorithm, VNS	Yes	Yes
4	CMA	Mono-Objective	Genetic Algorithm, Tabu Search	Yes	Yes
5	DAFO	Mono-Objective	Panmictic GAs (genGA and ssgA); CGAs (Cooperative Coevolutionary Genetic Algorithm - CCGA, Loosely Coupled Genetic Algorithm - LCGA and hybrid LCGA - hLCGA); LCGA Hybridized with the Next Ascent Hill Climbing (NAHC)	Yes	No
6	EasyLocal++	Mono-Objective	Exclusive to Local Search Algorithm	No	Yes
7	ECJ	Mono-Objective, Multi-Objective	Genetic Algorithms, Evolutionary Strategies, Particle Swarm Optimization and Differential Evolution, NSGA-II e SPEA-II	Yes	No
8	EvA2	Mono-Objective, Multi-Objective	Evolution Strategies (ES), Differential Evolution (DE), Genetic Algorithms (GA), Genetic Programming, Memetic Algorithms, Order-based GA, (IPOP-)CMA-ES, CHC Adaptive Search, Scatter Search, Nelder-Mead-Simplex, Multi-objective EA, NSGA II, PESA II, SPEA II, Cluster-based niching EA, Island-model EA, Simulated Annealing, Multi-start Hill Climbing, PBIL, Particle Swarm Optimization (PSO), Tribes, Clustering-based Niching PSO, Population-based Incremental Learning (PBIL), Bayesian Optimization Algorithm	Yes	Yes
9	FOM	Mono-Objective	Steepest Descent/Hill Climbing, Iterative Steepest Descent, Tabu Search, Simulated Annealing, GRASP, Variable Neighborhood Search, Evolutionary Algorithms (Genetic Algorithms, ES), Ant Systems (Classical Ant System, Ant Colony Optimization, Min-Max Ant System)	Yes	Yes
10	HeuristicLab	Mono-Objective, Multi-Objective	Population-based Algorithms, Trajectory based Algorithms, Multi-Objective Algorithm (NSGA-II), Optimal Algorithms, Extensible Algorithm	Yes	Yes
11	HotFrame	Mono-Objective	Local Search, Simulated Annealing, Tabu Search, Evolutionary algorithms, Candidate Lists, Neighborhood Depth and Pilot Method	Yes	Yes
12	HyFlex	Mono-Objective	Hyper-Heuristics (Iterated local search hyper-heuristic, AdapHH, VNS-TW, MLPHUNTER), Mutational or perturbation heuristics, Ruin-recreate (destruction-construction) heuristics, Hill-climbing or local search heuristics, Crossover heuristics.	Yes	Yes
13	JABAT	Mono-Objective	Local search algorithm, crossing algorithm, precedence tree algorithm, tabu search algorithm, MCS-based algorithm, path relinking algorithm, gene expression programming algorithm, tabu search algorithm, Lin-Kernighan heuristics, 2-opt and 3-opt heuristics, random local search, hill-climbing local search, simulated annealing	No	Yes
14	JAMES	Mono-Objective	Hill climbing, tabu search, variable neighborhood search and parallel tempering	No	Yes
15	JCLEC	Mono-Objective, Multi-Objective	Classic algorithms: simple generational, steady state and CHC, Multi-objective algorithms: NSGA-II and SPEA2, Memetic algorithms: generational and steady state, Scatter search algorithm, Niching algorithms: clearing, sequential and fitness sharing	Yes	No
16	jMetal	Mono-Objective, Multi-Objective	Evolutionary Techniques	Yes	No
17	LBMAS	Mono-Objective	Genetic Algorithms (GAs), Differential Evolution (DE), Simulated Annealing (SA), Ant Colony Optimization (ACO), Great Deluge Algorithm (GDA), Tabu Search (TS), and the Cross Entropy (CE) method	Yes	Yes
18	MACS	Mono-Objective	RandNEH and RandCWS	No	Yes
19	MAGMA	Mono-Objective	-	-	-
20	MALLBA	Mono-Objective	Exact Techniques, Hill Climbing, Metropolis, Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA) and hybrid techniques (GA+TS, GS+SA,Bnb+SA)	Yes	Yes
21	MANGO	Mono-Objective	BFBS quasi-Newton Algorithm, Trust-region Algorithm, Random Search	No	Yes
22	OptFrame	Mono-Objective, Multi-Objective	First Improvement, Best Improvement, Hill Climbing, Iterated Local Search, Simulated Annealing, Tabu Search, Variable Neighborhood Search, the basic versions of Genetic and Memetic Algorithms	Yes	Yes
23	Opt4j	Mono-Objective, Multi-Objective	SPEA2 and NSGA2, differential evolution, particle swarm optimization, and simulated annealing	Yes	Yes
24	ParadisEO	Mono-Objective, Multi-Objective	Modules: EO - Evolutionary Algorithms; MO - Trajectory Methods; MOEO: Evolutionary Methods	Yes	Yes
25	SMA	Mono-Objective	Bee colonies Optimization (BCO), Particle swarm optimisation (PSO), evolutionary simulated annealing (ESA)	Yes	No

Table 3

Advanced characteristics of metaheuristic optimization frameworks.

Framework	Characteristics							
	Implementation language	Portability	Licensing model	Hybridization	Support for parallel and distributed computing		Hyper-heuristic	Design in high-level abstraction
					Threads	Distributed computing		
1 AgE	Erlang	No	Apache License 2.0	Yes	Yes	Yes	Yes	Yes
2 AMAM	Java	Yes	–	Yes	Yes	No	No	Yes
3 AMF	Java	Yes	–	Yes	–	–	No	Yes
4 CMA	A-Globe Platform	–	–	Yes	Yes	Yes	No	–
5 DAFO	–	–	–	Yes	–	–	No	–
6 EasyLocal++	C++	Yes	–	Yes, Predefined	–	–	No	Yes
7 ECJ	Java	Yes	–	No	Yes	Yes	No	Yes
8 EvA2	Java	Yes	GPLv3	No	Yes	Yes	No	Yes
9 FOM	Java	Yes	GNU LGPL	No	No	No	No	Yes
10 HeuristicLab	.NET	No	GPLv3	No	Yes, HeuristicLab Grid Version	Yes, HeuristicLab Grid Version	No	Yes
11 HotFrame	C++	Yes	–	No	No	No	No	Yes
12 HyFlex	Java	Yes	–	Yes, Limited	–	–	Yes	Yes
13 JABAT	Java	Yes	–	Yes	Yes	Yes	No	Yes
14 JAMES	Java	Yes	Apache License 2.0	No	Yes	No	No	Yes
15 JCLEC	Java	Yes	GNU GPLv3	No	No	No	No	Yes
16 jMetal	Java	Yes	GNU LGPL	Yes	Yes	Yes	No	Yes
17 LBMAS	–	–	–	Yes	–	–	No	–
18 MACS	Java (JADE Platform)	Yes	GPLv3	Yes	Yes	Yes	No	Yes
19 MAGMA	Conceptual Model	–	–	Yes, Limited	–	–	No	–
20 MALLBA	C++	Yes	–	No	Yes	Yes	No	Yes
21 MANGO	Java	Yes	–	Yes	Yes	Yes	No	Yes
22 OptFrame	C++	Yes	GNU LGPLv3	Yes	Yes	Yes	No	Yes
23 Opt4j	Java	Yes	LGPL v2	No	Yes	Yes	No	Yes
24 ParadisEO	C++	Yes	GNU LGPL	Yes, Limited	Yes, Specific module (SMP e PEO)	Yes, Specific module (SMP e PEO)	No	Yes
25 SMA	Pop C++	Yes	–	Yes, Limited	–	–	Yes	No

Table 4

Multi-agent characteristics of metaheuristic optimization frameworks.

Framework	Characteristics				
	Agents	Environment	Autonomy of the agents	Learning	Hybridization (from the interaction between agents)
1 AgE	Yes	Yes. The environment of the multi-agent system is the search space of the treated problem.	Yes	No	Yes
2 AMAM	Yes	Yes. The environment of the multi-agent system is the search space of the treated problem.	Yes	Yes. Learning Automata.	Yes
3 AMF	Yes	–	Yes	Yes. Reinforcement and Mimetism Learning.	Yes
4 CMA	Yes	–	Yes	No	Yes
5 DAFO	Yes	Yes. Optimization problem which is provided and specified by the user.	Yes	No	Yes
6 JABAT	Yes	–	Yes	No	Yes
7 LBMAS	Yes	–	Yes, Limited	Yes. Learning automata.	Yes
8 MACS	Yes	–	Yes	Yes. Reinforcement Learning.	Yes
9 MAGMA	Yes	Yes, Fitness Landscapes.	No	No	Yes
10 MANGO	Yes	–	Yes	No	Yes
11 SMA	Yes	–	No	No	Yes

performed from the use of structures already predefined by the own framework. The EasyLocal++ framework presents only predefined hybrid algorithms. There are no facilities for the development of new hybrid algorithms. The ParadisEO-PEO module of the ParadisEO framework presents, in the same way, predefined models of metaheuristic hybridization. The third situation refers to the frameworks marked “*limited hybridization*” in the hybridization column of Table 3. This is the situation where hybridization is limited by the structures of the framework itself. Hybridization in the Hyflex framework is limited to how hyper-heuristics are defined by the own framework. In the SMA (Swarm of Metaheuristic Agent) framework, hybridization is limited to the interaction forms of population algorithms. In the MAGMA structure, the hybridization is limited to the forms of interaction between the levels of the architecture.

Regarding the possibility of using parallel and distributed computing, three frameworks (FOM, HotFrame, JCLEC) do not offer these features. The MAGMA framework, as presented in conceptual form only, does not provide enough elements to state any conclusions about the use of these resources. AMF, DAFO, Easy-Local++, HyFlex, LBMAS and SMA frameworks do not report the availability of these resources. All other analyzed frameworks offer parallel and distributed computing resources. Of all these, only the AMAM and JAMES frameworks offer the possibility of thread insertion; however, it does not provide distributed programming resources. In the case of HeuristicLab and ParadisEO frameworks, this possibility is associated with the use of specific modules for this purpose. It should be noted that the ParadisEO framework has a specific module for parallel computing using GPU architecture (Graphics Processing Units), implementing only local search (trajectory) metaheuristics [93]. The AgE framework stands out for offering features that facilitate distributed deployment, especially in multi-core architectures.

Concerning the possibility of implementing hyper-heuristics, we can consider that, besides Hyflex, i.e., the specific framework built for this formulation, the only framework that has this capability is SMA (Swarm of Metaheuristic Agents), which uses algorithms based on collective intelligence to coordinate metaheuristics and to enable the implementation of hyper-heuristics. The remaining works do not directly implement hyper-heuristics. The frameworks based on agents, such as AMAM, AMF, CMA, DAFO, JABAT, LBMAS, MACS, MAGMA, and MANGO may also allow the possibility of implementing hyper-heuristics, in an indirect way, by inserting control and coordination structures and based on

the agents themselves. Nevertheless, it is important to remark that hyper-heuristics can compromise the autonomy of the agents involved in the framework, by its inherent action in its own format of choosing, controlling and coordinating the performance of heuristics and metaheuristics. This is especially true in cases where the autonomy of the agents in the solution space of the problem is a requirement for the functioning of the framework. In these cases, the existence of structures of coordination and control, external to the agent itself, withdraw from the agent his decision-making capacity regarding his relationship with this solution space and with the other agents involved.

In relation to the used design and development techniques, most of them use good programming practices, which allows increase the reuse and facilitate the use of offered tools. The MAGMA framework does not allow any assessment with regard to this issue since they are only conceptual proposals. The CMA framework is developed based on the A-Globe architecture [114], not be possible, therefore, to infer any analysis regarding its own project characteristics. It is worth emphasizing that most of the analyzed frameworks are implemented in non-proprietary languages and based on the object-oriented paradigm.

6.4. Multi-Agent characteristics

In this section, we concentrate our attention only on the eleven frameworks described in Section 5, i.e., the frameworks built using multi-agent concepts. The objective is to analyze and to sort these frameworks. In order to do this, six characteristics are evaluated, highlighting how each of these frameworks incorporates properties inherent to multi-agent systems. The evaluated multi-agent characteristics are defined as:

- (i) Agents: if the framework uses the agent concept in its definition;
- (ii) Environment: how the used Multi-agent system is described;
- (iii) Autonomy of the agents: the agents are autonomous, that is, act without interference from other entities of the systems;
- (iv) Cooperation: when there is information sharing among agents that effectively influence their actions;
 - a. Communication Type;
 - b. Synchronous: if the cooperation is carried out from synchronous communication, i.e., if the processes, in some

Table 5

Multi-agent cooperation characteristics of metaheuristic optimization frameworks.

Framework	Cooperation characteristics		
	Communication type		Description of cooperation
	Synchronous	Asynchronous	
1 AgE	No	Yes	Erlang processes communicate using asynchronous messages, which removes the possibility of creating synchronization bottlenecks.
2 AMAM	No	Yes	The cooperation takes place through the solutions sharing available in Solutions Pool.
3 AMF	No	Yes	Cooperation between these agents occurs in two ways: (i) an agent shares its best known solution (This solution can be exploited by other agents with crossover operators); (ii) an agent shares its internal decision rules in order to allow mimicry of behavior (favor the search behaviors that often found new best solutions).
4 CMA	No	Yes	The exchange of information between agents is defined by the multi-agent platform A-Globe, which offers multi-agent features like skeleton agents, messaging, chat protocols, etc. Cooperation takes place from the sharing of global populations of candidate solutions stored in a pool of solutions.
5 DAFO	Yes	No	The interaction between agents is defined by communication protocols (FIPA ACL and FIPA-SL propositions). The MAS4EVO organization model specifies the global patterns of cooperation on agents behavior, defining a multi-agent organization based on coevolutionary strategies.
6 JABAT	No	Yes	Cooperation takes place through interconnected memories, in which solutions are constantly circulating.
7 LBMAS	No	Yes	The agents cooperate by sharing their individual experiences through a two-stage external memory archive.
8 MACS	Yes	No	Cooperation takes place through the conversation which is a cooperation protocol based on the retention of partial solutions deemed as possible constituents of future good solutions. There is an agent that takes on the role of an initiator and the others are responders.
9 MAGMA	No	Yes	Information sharing can happen in two ways: (i) horizontally, involves the interaction between the same level agents; And (ii) vertically, involves the interaction between different level agents. Communication between agents can be implemented through any kind of mechanism and protocol.
10 MANGO	No	Yes	The multi-agent environment used in the development of Mango (JADE) provides protocols of cooperation and organizational models to accomplish communication. The exchange of messages between the framework agents can happen in two ways: interrupts and mailboxes. Interrupts: when a message is sent to an agent, this will immediately stop the search process. Mailboxes: the message is stored and the agent will check its mailbox when is more appropriate.
11 SMA	Yes	No	A cooperation algorithm is applied, after each generation of the evolutionary algorithms, exchanging information related to the obtained solutions

- instant of time, are interrupted and become involved in some form of communication and information exchange;
- c. Asynchronous: if the cooperation is carried out from asynchronous communication. i.e., if each process is responsible for its own search and for conducting communications with other processes. The global search ends only if all individual search processes terminate;
 - d. Description: describes how cooperation occurs in the framework;
 - (v) Learning: if some kind of learning is implemented in agents or as a consequence of their actions. This feature enables the agent to assimilate knowledge about the environment and/or regarding other agents and use it to improve their action capacity;
 - (vi) Hybridization: the ability to hybridize metaheuristics, obtained from the interaction between the existing agents in the framework, mainly through cooperation. Note that the ability of interaction between the various agents employed in the optimization problem solution facilitates the emergence of hybrid methods.

Tables 4–6 present the results of the evaluation performed in the eleven frameworks described in Section 5 regarding the characteristics shown above, concerning the concept of multi-agents. They are discussed in the sequel.

Table 4 evaluates whether or not the framework uses the concepts of autonomous agents. Eleven proposals make use of this concept: AgE, AMAM, AMF, CMA, DAFO, JABAT, LBMAS, MACS, MAGMA, MANGO, and SMA. The use of this concept allows the hybridization of metaheuristics, either in the form of a combination of methods or, in particular, using the concept of cooperation, that is inherent in Multi-agent Systems.

The Multi-agent features were evaluated only in relation to frameworks that implement the concept of agents in its definition. The first evaluated feature is the Environment of the multi-agent structure defined in each proposal. The environment in a multi-

Table 6

Multi-agent characteristics of metaheuristic optimization frameworks: relation with search space.

Framework	Characteristics		
	Search space decomposition	Metaheuristic decomposition	Metaheuristic agents
1 AgE	X	X	
2 AMAM		X	X
3 AMF		X	
4 CMA			X
5 DAFO	X	X	X
6 JABAT		X	X
7 LBMAS			X
8 MACS			X
9 MAGMA		X	
10 MANGO			X
11 SMA			X

agent system is one where the agent is located and involves passive entities of the system, i.e., those without the capacity to act. The agent may have a global or local view of its environment, depending on how it has been set.

Most of the evaluated multi-agent frameworks (AMF, CMA, JABAT, LBMAS, MACS, MANGO, and SMA) does not make clear how is defined the environment in which the agents act. Only AgE, AMAM, DAFO and MAGMA frameworks describe their multi-agent environments. In MAGMA architecture, the environment is defined by the triple (S, N, F) , where S is the set of solutions, N is the function defining the neighborhood structure and F is the objective function. On AMAM framework, the environment is the element that allows the flexibility of the architecture, once it corresponds to the search space of the treated problem. Thus, the environment must be modeled for each specific problem and the change of environment instantiated in the application corresponds to the exchange of the problem to be solved. In DAFO framework, the environment is also defined by the optimization problem, which is provided and specified by the user. Table 6 also evidences this relation between

the environment in which the agents act and the solution space of the problem to be solved.

In a multi-agent system, autonomy guarantees to the agent freedom in its choices and in the execution of its tasks. The freedom of action mainly involves communication between agents in the multi-agent environment, allowing them to choose who they want to interact with. In this sense, part of the agent's autonomy is directly related to how the interaction is set. LBMAS, MAGMA, and Swarm of Metaheuristic Agents (SMA) frameworks define structures that limit, in part, the autonomy of agents and cooperation between them. In LBMAS framework, the autonomy of the agents is compromised by the presence of intermediate agents (Manager agent, Archive agent, Solution Pool agent), responsible for managing the exchange of information between agents.

In MAGMA architecture, this is due to the agents be allocated to subordinate levels, depending on each other. The agent, in MAGMA architecture, is autonomous in carrying out its specific activity. However, it depends on other agents and is coordinated by the higher level players (level 3). Cooperation, in MAGMA architecture, is also limited by this levels structure. The Swarm of Metaheuristic Agents (SMA) framework uses swarm intelligence to coordinate the search agents. Thus, agents are autonomous in their solution search process. However, a cooperation algorithm governs the cooperation between them. Cooperation, in Swarm of Metaheuristic Agents (SMA) proposal, is limited to the exchange of information between agents at the end of each generation of the used evolutionary algorithm.

The AgE, AMAM, AMF, CMA, DAFO, JABAT, MACS and MANGO frameworks, also based on multi-agent systems, have autonomous agents and cooperative process that do not interfere in this autonomy. These characteristics are essential for future developments and expansions of those frameworks.

The cooperation used by agent-based frameworks is discussed in Table 5. Two issues are assessed in relation to cooperation: types and functioning. The cooperation types used by the frameworks are evaluated according to the communication type adopted, namely: Synchronous and Asynchronous. As observed in Table 5, most agent-based frameworks use asynchronous communication, which is that in which each agent is responsible for its own search and for establishing communication with other processes, regardless of a specific moment or other agents.

Table 4 allows us to identify an important gap: the use of learning by agents in the solution search. According to Narendra and Thathachar [95], “learning is defined as any relatively permanent change in behavior resulting from past experience, and a learning system is characterized by its ability to improve its behavior with time, in some sense tending towards an ultimate goal”. This feature enables the agent to assimilate knowledge about the environment and/or regarding other agents and use it to improve their action capacity. The AMAM, AMF, LBMAS, MACS frameworks implement different kinds of machine learning. The AMAM and LBMAS frameworks use Learning Automata concepts [95] in decision-making agent. In the learning model based on Learning Automata, agent actions are selected according to a specific probability distribution, which is updated based on the environment responses. In the AMAM framework, the learning is used to define the application order of the neighborhood structures applied in the local searches. In the LBMAS framework, at each iteration, the learning is used in the choice of the next metaheuristic to be used. The AMF framework uses two learning mechanisms: Reinforcement Learning [74] and Mimetic Learning [132]. The Reinforcement Learning, used in the individual learning of the agent, allows selecting the most appropriate operator according to their experiences. The Mimetic Learning allows cooperation between agents, by sharing the weight matrix of the agent considered more efficient. The concept of learning is used, in the MACS framework, from the short-term memory of good arcs,

Table 7
Support to the optimization process.

Framework	Characteristics		
	Statistical analysis	Self-tuning	GUI (User Interface)
1 AgE	Yes	No	No
2 AMAM	No	No	No
3 AMF	No	No	No
4 CMA	No	Yes	No
5 DAFO	No	No	No
6 EasyLocal++	No	No	No
7 ECJ	No	No	Yes
8 EvoA2	No	No	Yes
9 FOM	Yes	Yes, Limited	Yes
10 HeuristicLab	Yes	No	Yes
11 HotFrame	No	No	No
12 HyFlex	No	No	No
13 JABAT	No	No	No
14 JAMES	No	No	No
15 JCLEC	No	No	Yes
16 jMetal	Yes	No	No
17 LBMAS	No	No	No
18 MACS	No	No	No
19 MAGMA	No	No	No
20 MALLBA	No	No	No
21 MANGO	No	No	No
22 OptFrame	No	No	No
23 Opt4j	No	No	Yes
24 ParadisEO	No	No	No
25 SMA	No	No	No

defined for each agent. This memory is used to modify the performance of each metaheuristic, allowing them, therefore, to find better solutions. In these four frameworks, the learning directly influences the decision making of the agent, seeking to improve the quality of the solutions obtained based on the experience acquired. It is important to note that the CMA framework has a learning structure, used, in this case, to perform auto-tuning of the parameters implemented in the methods.

6.5. Support features to optimization process

The support features of the optimization process present in the Metaheuristic Optimization Frameworks described in Sections 4 and 5 are presented below. These features represent their own usability by the user, as well as the possibility of auto-tuning of parameters in the optimization process. It is important to emphasize that the presence or absence of these characteristics evidences the degree of maturity gained by the framework proposals in its development process. These support features are defined as:

- (i) Statistical Analysis: it enables statistical analysis of the results;
- (ii) Self-tuning: if it has the ability of auto adjustment of the implemented metaheuristic parameters;
- (iii) GUI: the user interface.

Table 7 shows the results concerning the evaluation of the frameworks listed in Table 1 about the issues related to the support features to the optimization process. From this table, it is clear that only a little bit of these features are covered by the evaluated frameworks. Only AgE, FOM, HeuristicLab, JCLEC, jMetal and ParadisEO frameworks have statistical analysis capabilities. The self-tuning of parameters is present in only two frameworks (CMA and FOM). The CMA proposal, by the Adviser Agent, allows automatically configuring the value of the parameters from the received reports about the executed algorithms. The FOM framework, although asserting the existence of automatic parameter adjustment, does not specify how the data observed in the execution of metaheuristics are used in the self-tuning of the parameters for the next executions or how they are used to improve the next results. The framework only

specifies the possibility of monitoring the parameters and generating graphs that demonstrate them. Regarding the availability of graphic interface, ECJ, Eva2, FOM, HeuristicLab, Hotframe, JCLEC, jMetal and Opt4j frameworks claim to have such facilities.

7. Conclusions and future directions

The overall goal of this article was to distinguish the desirable characteristics of a framework for metaheuristic optimization and identify existing gaps, especially those related to the hybridization of metaheuristics. Therefore, an effort to investigate the state of the art related to these tools was carried out. Initially, an overview of metaheuristic hybridization, with a special focus on cooperative metaheuristics and parallel metaheuristics, was presented. The combination of cooperation and parallelism, widely used in the context of optimization, was also analyzed. General-purpose structures for optimization, called Frameworks, were presented, which are intended to assist the researchers in the development of metaheuristics while minimizing the computational effort expended. Additionally, a specific set of works that use the concepts of Multi-Agent Systems in the organization of relations between the methods implemented here, also seen as frameworks, is described.

Aiming the proposed objective, a comparison between the presented frameworks was performed. To this end, the analyzed frameworks were selected using five pre-defined criteria. The comparison was made from a set of twenty-two characteristics, which were defined based on the works of Talbi [116], Parejo et al. [101], Raidl [105] and Silva [110]. These characteristics were divided into four categories: (i) general; (ii) advanced; (ii) multi-agent; (iv) and support for the optimization process.

Among the several works listed in this article, it is observed that most of the frameworks have similar features and proposals.

The comparative analysis presented here evaluated all the frameworks considered in Parejo et al. [101] as well as those that arose after the publication of this article. This analysis showed that hybridization is still not a strong characteristic of existing frameworks, as stated by Parejo et al. [101]. Thus, it is paramount to conclude that the frameworks remain without displaying, as a property, the ease and the flexibility for the development of hybrid metaheuristics. The hybridization of metaheuristics has proved to be a very important methodology for solving optimization problems, once, as highlighted in Blum et al. [20], the best results obtained in the solution of combinatorial optimization problems result from hybridizations.

However, the development of frameworks has not moved in that direction. Developing general structures that allow the user to use one optimization technique at a time is noticeably simpler than offering the possibility of combining these techniques, enabling the emergence of hybrid techniques. On the other hand, considering the good results obtained with hybrid metaheuristics, we can say that there is a need for tools that offer the possibility of combining methods for solving optimization problems. In this sense, frameworks that use the multi-agent approaches are, as the presented analysis show, a good option in the process of hybridization, allowing the development of generic structures, that enable interaction between metaheuristics, regardless of which of them are used, and even independent of the problem to be treated. These approaches propose forms of interaction that facilitate hybridization, allowing, with considerable ease, the use of cooperative techniques and, in the vast majority, parallel computing resources.

The comparison confirmed this affirmation, showing that, among the proposals that actually enable hybridization methods, frameworks that use the concept of multi-agent systems in its composition are highlighted. Cooperation and autonomy of action of

the agents, present in proposals that use multi-agent approaches, flexibilize the implementation of hybrid methods. Additionally, the concept of multi-agent systems allows different regions of the search space be operated simultaneously by the agents.

In this context, it is important to stand out the clear differences between the group of optimization frameworks based on the multi-agent approach (Section 5) and the group of optimization frameworks that do not use this concept (Section 4). The frameworks that do not use the multi-agent concept present a good number of advanced features and resources for support to the optimization process; On the other hand, frameworks based on the multi-agent approach have characteristics that benefit the hybridization of metaheuristics.

The impact of new parallel and distributed computing support technologies points to new research possibilities that have yet to be explored by framework developers. The ParadisEO and AgE frameworks stand out from the others discussed here because they present implementations that facilitate distributed development and allow the use of multi- and many-core structures. In addition, it is necessary to explore new programming languages that facilitate the development using these new resources. According to Krzywicki et al. [77] and Turek et al. [122], functional languages, such as Erlang and Scala, are highlighted in this regard.

Hyper-heuristic is another concept very little explored by the analyzed frameworks. In addition to the Hyflex framework, which specializes in hyper-heuristics, only the Swarm of Metaheuristic Agents (SMA) framework, which uses algorithms based on collective intelligence to coordinate metaheuristics, implements hyper-heuristics. On the other hand, hyper-heuristics may restrict the autonomy of decision of multi-agent systems. Thus, it does not seem advisable to use these coordination structures in conjunction with multi-agent systems.

Among the common features in multi-agent systems, learning is not implemented even by frameworks that use this approach in their design. This feature can help in the search process, allowing the agent to adapt and possibly improve their capacity for action, without, however, constrain their autonomy, but rather, adding information to the agent can better decide when solving an optimization problem. Another important open question is the use of multi-agent systems for solving multi-objective optimization problems. Tables 2, 4 and Table 5 show that the AMAM framework is an important candidate to supply this lack in the development of frameworks for optimization using metaheuristics.

Additionally, there is a lack of tools that support the optimization process, such as statistical analysis, self-tuning of parameters and graphical interfaces, features already assessed in the presented comparative.

Finally, we believe that this article is advancing strongly in the answers to the four questions posed in the Introduction section. Thus, regarding the first question, the current frameworks do not facilitate the development of hybrid metaheuristic applications. Regarding the second question, the current frameworks have difficulties in allowing application development with cooperation (exchange of information) between methods that are run independently and simultaneously. As for the third question, the answer to the best way to coordinate metaheuristics when the goal is to explore the potential for hybridization is the use of frameworks with multi-agent systems. At the end of this review article, the answer to the fourth question is, again, to develop the use of frameworks with multi-agent systems once it allows a more robust structure, capable of handling different problems with minimal changes. It can be stated that the gaps identified by this analysis provide important opportunities for future works on the development of frameworks for optimization using metaheuristics, which can be elucidated particularly by the implementation of multi-agent systems-based approaches.

Compliance and ethical standards

This study was funded by Brazilian agencies National Council of Technological and Scientific Development - CNPq (Grants 307915/2016-6, 306694/2013-1 and 552289/2011-6) and Minas Gerais State Research Foundation - FAPEMIG (Grants PPM CEX 772/15 and 676/17). Authors Maria Amélia Lopes Silva, Sérgio Ricardo de Souza, Marcone Jamilson Freitas Souza and Moacir Felizardo de França Filho declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

Acknowledgements

The authors would like to thank Marcus Vinicius Silva e Souza and Prof. Rogério Gomes Martins for their valuable suggestions and commentaries about the article. The authors would also like to thank the Minas Gerais State Research Foundation (FAPEMIG), the National Council of Technological and Scientific Development (CNPq), the Federal Center of Technological Education of Minas Gerais (CEFET-MG), the Federal University of Ouro Preto (UFOP) and the Federal University of Viçosa (UFV) for supporting the development of the present study. Finally, we would like to thank the anonymous reviewers for their constructive comments and suggestions that significantly improved the content and exposition of this work.

References

- [1] C. Agerbeck, M.O. Hansen, A multi-agent approach to solving NP-complete problems. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark (DTU), Denmark, 2008.
- [2] E. Alba, Parallel Metaheuristics: A New Class of Algorithms, Wiley-Interscience, 2005.
- [3] E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, G. Luque, J. Petit, C. Rodríguez, A. Rojas, F. Xhafa, Efficient parallel LAN/WAN algorithms for optimization. The MALLBA Project, *Parallel Computing* 32 (5–6) (2006) 415–440.
- [4] E. Alba, F. Almeida, M.J. Blesa, J. Cabeza, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, J. Luna, L.M. Moreno, C. Pablos, J. Petit, A. Rojas, F. Xhafa, MALLBA: a library of skeletons for combinatorial optimisation (research note), in: B. Monien, R. Feldmann (Eds.), Proceedings of the 8th International Euro-Par Conference on Parallel Processing (Euro-Par '02), Vol. 2400 of Lecture Notes in Computer Science, Springer-Verlag, 2002, pp. 927–932.
- [5] E. Alba, G. Luque, J. García-Nieto, G. Ordóñez, G. Leguizamón, MALLBA: a software library to design efficient optimisation algorithms, *International Journal of Innovative Computing and Applications* 1 (1) (2007) 74–85.
- [6] E. Alirezai, Z. Vahedi, M. Ghaznavi-Ghoushchi, Parallel hybrid meta heuristic algorithm for university course timetabling problem (PHACT), Proceedings of the 20th Iranian Conference on Electrical Engineering (2012 ICEE) (2012 May) 673–678.
- [7] J.E. Amaya, C. Cotta, A.J.F. Leiva, Hybrid cooperation models for the tool switching problem, in: J.R. González, D.A. Pelta, C. Cruz, G. Terrazas, N. Krasnogor (Eds.), *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Vol. 284 of Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2010, pp. 39–52.
- [8] F.B. Aydemir, A. Günay, F. Öztoprak, Ş.I. Birbil, P. Yolum, Multiagent cooperation for solving global optimization problems: an extensible framework with example cooperation strategies, *Journal of Global Optimization* 57 (2) (2012) 499–519.
- [9] M.E. Aydin, Collaboration of heterogenous metaheuristic agents, 2010 Fifth International Conference on Digital Information Management (ICDIM) (July 2010) 540–545.
- [10] M.E. Aydin, Coordinating metaheuristic agents with swarm intelligence, *Journal of Intelligent Manufacturing* 23 (4) (2012) 991–999.
- [11] M.E. Aydin, Agentification of individuals: a multi-agent approach to metaheuristics, *Journal of Computer Science & Systems Biology* 6 (5) (2013).
- [12] D. Barbucha, Cooperative solution to the vehicle routing problem, in: P. Jedrzejowicz, N.T. Nguyen, R.J. Howlett, L.C. Jain (Eds.), *Agent and Multi-Agent Systems: Technologies and Applications: 4th KES International Symposium, KES-AMSTA 2010, Gdynia, Poland, June 23–25, 2010, Proceedings. Part II*, Springer, Berlin, Heidelberg, 2010, pp. 180–189.
- [13] D. Barbucha, Team of A-Teams approach for vehicle routing problem with time windows, in: G. Terrazas, F.E.B. Otero, A.D. Masegosa (Eds.), *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, Vol. 512 of Studies in Computational Intelligence, Springer International Publishing, 2014, pp. 273–286.
- [14] D. Barbucha, I. Czarnowski, P. Jedrzejowicz, E. Ratajczak, I. Wierzbowska, JABAT – an implementation of the A-Team concept, in: Proc. Int. Multiconference Computer Science and Information Technology, Wisła. Vol. 1, 2006, pp. 235–241.
- [15] D. Barbucha, I. Czarnowski, P. Jedrzejowicz, E. Ratajczak-Ropel, I. Wierzbowska, e-JABAT – an implementation of the web-based A-Team, in: N.T. Nguyen, L.C. Jain (Eds.), *Intelligent Agents in the Evolution of Web and Applications*, Springer, Berlin, Heidelberg, 2009, pp. 57–86.
- [16] D. Barbucha, I. Czarnowski, P. Jedrzejowicz, E. Ratajczak-Ropel, I. Wierzbowska, JABAT middleware as a tool for solving optimization problems, in: N.T. Nguyen, R. Kowalczyk (Eds.), *Transactions on Computational Collective Intelligence II*, Springer, Berlin, Heidelberg, 2010, pp. 181–195.
- [17] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi-agent systems with JADE, in: C. Castelfranchi, Y. Lespérance (Eds.), *Intelligent Agents VII Agent Theories Architectures and Languages*, Vol. 1986 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2001, pp. 89–103.
- [18] F. Bellifemine, A. Poggi, G. Rimassa, *Developing Multi-agent Systems with JADE*, John Wiley, 2007.
- [19] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, A brief survey on hybrid metaheuristics, in: B. Filipič, J. Šilc (Eds.), *Proceedings of the 4th International Conference on Bio-inspired Optimization Methods and their Applications (BIOMA 2010)*, Jozef Stefan Institute, Ljubljana, Slovenia, 2010, pp. 3–18.
- [20] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, *Applied Soft Computing* 11 (6) (2011) 4135–4151.
- [21] C. Blum, Metaheuristics in combinatorial optimization: overview and conceptual comparison., *ACM Computing Surveys* 35 (3) (2003) 268–308.
- [22] C. Blum, A. Roli, Hybrid metaheuristics: an introduction, in: C. Blum, M.J.B. Aguilera, A. Roli, M. Sampels (Eds.), *Hybrid Metaheuristics: An Emerging Approach to Optimization*, Vol. 114 of Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2008, pp. 1–30.
- [23] I. Boussaid, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, *Information Sciences* 237 (July 2013) 82–117.
- [24] E. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J.A. Vázquez-Rodríguez, M. Gendreau, Iterated local search vs. hyper-heuristics: towards general-purpose search algorithms, *IEEE Congress on Evolutionary Computation* (July 2010) 1–8.
- [25] E.K. Burke, M. Gendreau, G. Ochoa, J.D. Walker, Adaptive iterated local search for cross-domain optimisation, in: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11*, New York, NY, USA, 2011, pp. 1987–1994.
- [26] E.K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, S. Schulenburg, Hyper-heuristics: an emerging direction in modern search technology, in: F. Glover, G.A. Kochenberger (Eds.), *Handbook of Metaheuristics*, Vol. 57 of International Series in Operations Research & Management Science, Springer, US, 2003, pp. 457–474.
- [27] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for educational timetabling problems, *European Journal of Operational Research* 176 (1) (2007) 177–192.
- [28] A. Butterfield, G.E. Ngondi, *Oxford Dictionary of Computer Science*, 7th ed., Oxford University Press, 2016.
- [29] A. Byrski, R. Dreżewski, L. Siwik, M. Kisiel-Dorohinicki, Evolutionary multi-agent systems, *The Knowledge Engineering Review* 30 (2) (2015) 171–186.
- [30] A. Byrski, M. Kisiel-Dorohinicki, *Evolutionary Multi-agent Systems: From Inspirations to Applications*, Vol. 680 of *Studies in Computational Intelligence*, Springer, 2017.
- [31] S. Cahon, N. Melab, E.-G. Talbi, ParadisEO: a framework for the reusable design of parallel and distributed metaheuristics, *Journal of Heuristics* 10 (3) (2004) 357–380.
- [32] A. Cano, J.M. Luna, A. Zafra, S. Ventura, A classification module for genetic programming algorithms in JCLEC, *J. Mach. Learn. Res.* 16 (1) (Jan 2015) 491–494.
- [33] M. Carle, A. Martel, N. Zufferey, Collaborative Agent Teams (CAT) for Distributed Multi-Dimensional Optimization, *Tech. Rep. CIRREL-T-2012-43* (2012), CIRRELT, Montréal, Canada.
- [34] K. Cetnarowicz, M. Kisiel-Dorohinicki, E. Nawarecki, The application of evolution process in multi-agent world (MAW) to the prediction system, in: M. Tokoro (Ed.), *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS'96)*, AAAI Press, 1996, pp. 26–32.
- [35] K. Chakhlevitch, P. Cowling, Hyperheuristics: recent developments, in: *Adaptive and Multilevel Metaheuristics*, Springer, 2008, pp. 3–29.
- [36] I.M. Coelho, P.L.A. Munhoz, M.N. Haddad, V.N. Coelho, M.M. Silva, M.J.F. Souza, L.S. Ochi, OptFrame: a computational framework for combinatorial optimization problems, in: *Proc. of the VII ALIO/EURO Workshop on Applied Combinatorial Optimization, ALIO/EURO 2011*, Porto, Portugal, May, 2011, pp. 51–54.
- [37] I.M. Coelho, S. Ribas, M.H.P. Perché, P.L.A. Munhoz, M.J.F. Souza, L.S. Ochi, OptFrame: a computational framework for combinatorial problems, in: *Proceedings of the XLII Brazilian Symposium of Operations Research, Bento Gonçalves, Brazil, Sept. 2010*, pp. 1887–1898.
- [38] I.M. Coelho, S. Ribas, M.J.F. Souza, V.N. Coelho, L.S. Ochi, A hybrid heuristic algorithm based on GRASP, VND, ILS and Path Relinking for the open-pit-mining operational planning problem, in: *Proceedings of the XXX*

- Iberian-Latin-American Congress on Computational Methods in Engineering-CILAMCE, Búzios, Brazil, 2009.
- [39] V.N. Coelho, I.M. Coelho, B.N. Coelho, M.W. Cohen, A.J.R. Reis, S.M. Silva, M.J.F. Souza, P.J. Fleming, F.G. Guimarães, Multi-objective energy storage power dispatching using plug-in vehicles in a smart-microgrid, *Renewable Energy* 89 (2016) 730–742.
- [40] V.N. Coelho, I.M. Coelho, B.N. Coelho, A.J.R. Reis, R. Enayatifar, M.J.F. Souza, F.G. Guimarães, A self-adaptive evolutionary fuzzy model for load forecasting problems on smart grid environment, *Applied Energy* 169 (2016) 567–584.
- [41] V.N. Coelho, A. Gratas, H. Ramalhinho, I.M. Coelho, M.J.F. Souza, R.C. Cruz, An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints, *European Journal of Operational Research* 250 (2) (2016) 367–376.
- [42] V.N. Coelho, T.A. Oliveira, I.M. Coelho, B.N. Coelho, P.J. Fleming, F.G. Guimaraes, H. Ramalhinho, M.J.F. Souza, E.-G. Talbi, T. Lust, Generic Pareto local search metaheuristic for optimization of targeted offers in a bi-objective direct marketing campaign, *Computers & Operations Research* 78 (2017) 578–587.
- [43] C. Cotta, E.-G. Talbi, E. Alba, Parallel hybrid metaheuristics, in: E. Alba (Ed.), *Parallel Metaheuristics: A New Class of Algorithms*, John Wiley & Sons, 2005, pp. 347–370.
- [44] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: *Practice and Theory of Automated Timetabling III*, Springer, 2001, pp. 176–190.
- [45] T.G. Crainic, M. Gendreau, Cooperative parallel tabu search for capacitated network design, *Journal of Heuristics* 8 (6) (Nov 2002) 601–627.
- [46] T.G. Crainic, M. Toulouse, Parallel strategies for meta-heuristics, in: F. Glover, G.A. Kochenberger (Eds.), *Handbook of Metaheuristics*. Vol. 57 of International Series in Operations Research & Management Science, Springer, US, 2003, pp. 475–513.
- [47] T.G. Crainic, M. Toulouse, Parallel meta-heuristics, in: M. Gendreau, J.-Y. Potvin (Eds.), *Handbook of Metaheuristics*, Springer US, Boston, MA, 2010, pp. 497–541, URL https://doi.org/10.1007/978-1-4419-1665-5_17.
- [48] G. Danoy, P. Bouvry, O. Boissier, Dafo, a multi-agent framework for decomposable functions optimization, in: R. Khosla, R.J. Howlett, L.C. Jain (Eds.), *Knowledge-Based Intelligent Information and Engineering Systems: 9th International Conference, KES 2005, Melbourne, Australia, September 14–16, 2005, Proceedings, Part IV*, Springer, Berlin, Heidelberg, 2005, pp. 626–632.
- [49] G. Danoy, P. Bouvry, O. Boissier, A multi-agent organizational framework for coevolutionary optimization, in: K. Jensen, S. Donatelli, M. Koutny (Eds.), *Transactions on Petri Nets and Other Models of Concurrency IV*, Springer, Berlin, Heidelberg, 2010, pp. 199–224.
- [50] H. De Beukelaer, G.F. Davenport, G. De Meyer, V. Fack, JAMES: a modern object-oriented java framework for discrete optimization using local search metaheuristics, Proc. 4th International Symposium and 26th National Conference on Operational Research: Hellenic Operational Research Society (2015) 134–138.
- [51] H. De Beukelaer, G.F. Davenport, G. De Meyer, V. Fack, JAMES: an object-oriented java framework for discrete optimization using local search metaheuristics, *Software: Practice and Experience*, n/a-n/a. (2016).
- [52] J.J. Durillo, A.J. Nebro, jMetal: a java framework for multi-objective optimization, *Advances in Engineering Software* 42 (10) (2011) 760–771.
- [53] J.J. Durillo, A.J. Nebro, E. Alba, The jMetal framework for multi-objective optimization: design and architecture, *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC)* (2010) 1–8.
- [54] M. El-Abd, M. Kamel, A taxonomy of cooperative search algorithms, in: M.J. Blesa, C. Blum, A. Roli, M. Sampels (Eds.), *Hybrid Metaheuristics*. Vol. 3636 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005, pp. 32–41.
- [55] T.A. El-Mihoub, A.A. Hopgood, L. Nolle, A. Battersby, Hybrid genetic algorithms: a review, *Eng. Lett.* 13 (2) (2006) 124–137.
- [56] A. Elyasaf, M. Sipper, Software review: the heuristiclab framework, *Genetic Programming and Evolvable Machines* 15 (2) (2014) 215–218.
- [57] F.C. Fernandes, S.R. de Souza, M.A.L. Silva, H.E. Borges, F.F. Ribeiro, A multiagent architecture for solving combinatorial optimization problems through metaheuristics, *Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics (SMC 2009)* (2009) 3071–3076.
- [58] A. Fink, S. Voß, Hotframe: a heuristic optimization framework, in: S. Voß, D.L. Woodruff (Eds.), *Optimization Software Class Libraries*, Springer US, Boston, MA, 2002, pp. 81–154, URL http://dx.doi.org/10.1007/0-306-48126-X_4.
- [59] A. Fink, S. Voß, D.L. Woodruff, Building reusable software components for heuristic search, in: P. Kall, H.-J. Lüthi (Eds.), *Operations Research Proceedings 1998: Selected Papers of the International Conference on Operations Research Zurich, August 31–September 3, 1998*. Vol. 1998 of *Operations Research Proceedings 1998*, Springer, Berlin, Heidelberg, 1999, pp. 210–219.
- [60] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Object-Oriented Software*, Addison Wesley, 1995.
- [61] L.D. Gaspero, A. Schaerf, A case-study for EasyLocal++: the course timetabling problem, *Tech. Rep. Technical Report UDMI/13/2001/RR*, Dipartimento di Matematica e Informatica - Università di Udine, Italy, 2001.
- [62] L.D. Gaspero, A. Schaerf, Writing local search algorithms using EasyLocal++, in: S. Voß, D.L. Woodruff (Eds.), *Optimization Software Class Libraries*, Springer US, Boston, MA, 2002, pp. 155–175.
- [63] L.D. Gaspero, A. Schaerf, EASYLOCAL++: an object-oriented framework for the flexible design of local-search algorithms, *Software: Practice and Experience* 33 (8) (2003) 733–765.
- [64] L.D. Gaspero, T. Urli, A reinforcement learning approach for the cross-domain heuristic search challenge, in: *Proceedings of the 9th Metaheuristics International Conference (MIC 2011)*, Udine, Italy, July 2011.
- [65] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, J.-J. Li, Distributed evolutionary algorithms and their models: a survey of the state-of-the-art, *Applied Soft Computing* 34 (2015) 286–300.
- [66] D. González-Álvarez, M. Vega-Rodríguez, A parallel cooperative team of multiobjective evolutionary algorithms for motif discovery, *The Journal of Supercomputing* 66 (3) (2013) 1576–1612.
- [67] A. Güney, F. Öztoprak, Ş. İlker Birbil, P. Yolum, Solving global optimization problems using MANGO, in: A. Häkansson, N.T. Nguyen, R.L. Hartung, R.J. Howlett, L.C. Jain (Eds.), *Agent and Multi-Agent Systems: Technologies and Applications*. Vol. 5559 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2009, pp. 783–792.
- [68] J.F. Hubner, J.S. Sichman, O. Boissier, Developing organised multiagent systems using the MOISE+ Model: programming issues at the system and agent levels, *International Journal of Agent-Oriented Software Engineering* 1 (3/4) (Dec 2007) 370–395.
- [69] J. Humeau, A. Leflooghe, E.-G. Talbi, S. Verel, ParadisEO-MO: from fitness landscape analysis to efficient local search algorithms, *Journal of Heuristics* 19 (6) (2013) 881–915.
- [70] J. Jin, T.G. Crainic, A. Løkketangen, A cooperative parallel metaheuristic for the capacitated vehicle routing problem, *Computers & Operations Research* 44 (0) (2014) 33–41.
- [71] X. Jin, J. Liu, Multiagent SAT (MASSAT): autonomous pattern search in constrained domains, in: H. Yin, N. Allinson, R. Freeman, J. Keane, S. Hubbard (Eds.), *Proceedings of the Third International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'02)*, Springer, Berlin, London, UK, 2002, pp. 318–328.
- [72] R. Johnson, B. Foote, Designing reusable classes, *Journal of Object-Oriented Programming* 1 (2) (1988) 22–35.
- [73] L. Jourdan, M. Basseur, E.-G. Talbi, Hybridizing exact methods and metaheuristics: a taxonomy, *European Journal of Operational Research* 199 (3) (2009) 620–629.
- [74] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, *EJ. Artif. Intell. Res* 4 (1996) 237–285.
- [75] L. Kerçelli, A. Sezer, F. Öztoprak, P. Yolum, Ş.I. Birbil, MANGO: a multiagent environment for global optimization, in: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'08)*, Estoril, Portugal, 2008, pp. 86–91.
- [76] M. Kronfeld, H. Planatscher, A. Zell, The Eva2 optimization framework, in: C. Blum, R. Battini (Eds.), *Learning and Intelligent Optimization: 4th International Conference, LION 4, Venice, Italy, January 18–22, 2010. Selected Papers*, Springer, Berlin, Heidelberg, 2010, pp. 247–250.
- [77] D. Krzywicki, W. Turek, A. Byrski, M. Kisiel-Dorohinicki, Massively concurrent agent-based evolutionary computing, *Journal of Computational Science* 11 (2015) 153–162.
- [78] D. Landa-Silva, E.K. Burke, Asynchronous cooperative local search for the office-space-allocation problem, *INFORMS J. Comput.* 19 (4) (Nov 2007).
- [79] A. Leflooghe, L. Jourdan, E. Talbi, A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEIO, *European Journal of Operational Research* 209 (2) (2011) 104–112.
- [80] J. Liu, K. Sycara, Distributed problem solving through coordination in a society of agents, *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence* (1994) 169–185.
- [81] N. Lotfi, A. Acan, Learning-based multi-agent system for solving combinatorial optimization problems: a new architecture, in: E. Onieva, I. Santos, E. Osaba, H. Quintián, E. Corchado (Eds.), –24, 2015, *Proceedings*, S, 2015, pp. 319–332.
- [82] N. Lotfi, A. Acan, A tournament-based competitive-cooperative multiagent architecture for real parameter optimization, *Soft Computing* 20 (11) (2016) 4597–4617.
- [83] M. Lukasiewycz, M. Glaß, F. Reimann, J. Teich, Opt4J: a modular framework for meta-heuristic optimization, in: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11*, New York, NY, USA, 2011, pp. 1723–1730.
- [84] S. Luke, ECj then and now, in: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17*, ACM, New York, NY, USA, 2017, pp. 1223–1230.
- [85] J.A. Lukin, A.P. Gove, S.N. Talukdar, C. Ho, Automated probabilistic method for assigning backbone resonances of (13C,15N)-labeled proteins, *Journal of Biomolecular NMR* 9 (2) (1997) 151–166.
- [86] R. Malek, Collaboration of metaheuristic algorithms through a multi-agent system, in: V. Mařík, T. Strasser, A. Zoitl (Eds.), *Holonic and Multi-Agent Systems for Manufacturing*. Vol. 5696 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 72–81.
- [87] R. Malek, An agent-based hyper-heuristic approach to combinatorial optimization problems, *Proceedings of the 2010 IEEE International*

- Conference on Intelligent Computing and Intelligent Systems (ICIS). Vol. 3 (Oct 2010) 428–434.
- [88] S. Martin, D. Ouelhadj, P. Beullens, E. Ozcan, A.A. Juan, E.K. Burke, A multi-agent based cooperative approach to scheduling and routing, *European Journal of Operational Research* 254 (1) (2016) 169–178.
- [89] D. Meignan, J.-C. Créput, A. Koukam, A coalition-based metaheuristic for the vehicle routing problem, *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)* (2008) 1176–1182.
- [90] D. Meignan, J.-C. Créput, A. Koukam, An organizational view of metaheuristics, in: N. Jennings, A. Rogers, A. Petcu, S.D. Ramchurn (Eds.), *First International Workshop on Optimisation in Multi-Agent Systems, AAMAS'08*, 2008, pp. 77–85.
- [91] D. Meignan, J.-C. Créput, A. Koukam, A cooperative and self-adaptive metaheuristic for the facility location problem, in: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO'09*, ACM, New York, NY, USA, 2009, pp. 317–324.
- [92] D. Meignan, A. Koukam, J.-C. Créput, Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism, *Journal of Heuristics* 16 (6) (2010) 859–879.
- [93] N. Melab, T.V. Luong, K. Boufaras, E. Talbi, ParadisEO-MO-GPU: a framework for parallel GPU-based local search metaheuristics, *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, ACM* (2013) 1189–1196.
- [94] M. Milano, A. Roli, MAGMA: a multiagent architecture for metaheuristics, *IEEE Trans. Syst. Man Cybern. Part B: Cybern.* 34 (2) (2004) 925–941.
- [95] K.S. Narendra, M.A.L. Thathachar, Learning automata – a survey, *IEEE Trans. Syst. Man Cybern. SMC-4* (4) (July 1974) 323–334.
- [96] A.J. Nebro, J.J. Durillo, M. Vergne, Redesigning the jmetal multi-objective optimization framework, in: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15*, ACM, New York, NY, USA, 2015, pp. 1093–1100, URL: <http://doi.acm.org/10.1145/2739482.2768462>.
- [97] G. Ochoa, M. Hyde, T. Curtois, J.A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A.J. Parkes, S. Petrovic, E.K. Burke, HyFlex: a benchmark framework for cross-domain heuristic search, in: J.-K. Hao, M. Middendorf (Eds.), –13, 2012. *Proceedings*, Springer, Berlin, Heidelberg, 2012, p. 1.
- [98] E. Özcan, B. Bilgin, E.E. Korkmaz, A comprehensive analysis of hyper-heuristics, *Intelligent Data Analysis* 12 (1) (2008) 3–23.
- [99] E. Özcan, A. Kheiri, A hyper-heuristic based on random gradient, greedy and dominance, in: E. Gelenbe, R. Lent, G. Sakellari (Eds.), *Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences, Springer London*, London, 2012, pp. 557–563.
- [100] J.A. Parejo, J. Racero, F. Guerrero, T. Kwok, K.A. Smith, FOM: a framework for metaheuristic optimization, in: P.M.A. Sloot, D. Abramson, A.V. Bogdanov, Y.E. Gorbachev, J.J. Dongarra, A.Y. Zomaya (Eds.), –4, 2003 *Proceedings, Part IV*, Springer, Berlin, Heidelberg, 2003, p. 8.
- [101] J.A. Parejo, A. Ruiz-Cortés, S. Lozano, P. Fernandez, Metaheuristic optimization frameworks: a survey and benchmarking, *Soft Computing* 16 (3) (2012) 527–561.
- [102] M.A. Potter, K.A. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, *Evolutionary Computation* 8 (1) (2000) 1–29.
- [103] J. Puchinger, G.R. Raidl, Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification, in: J. Mira, J.R. Álvarez (Eds.), *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*. Vol. 3562 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2005, pp. 41–53.
- [104] C.S. Rabak, J.S. Sichman, Using A-Teams to optimize automatic insertion of electronic components, *Advanced Engineering Informatics* 17 (2) (2003) 95–106.
- [105] G.R. Raidl, A unified view on hybrid metaheuristics, in: F. Almeida, M.J.B. Aguilera, C. Blum, J.M.M. Vega, M.P. Pérez, A. Roli, M. Sampels (Eds.), *Hybrid Metaheuristics*. Vol. 4030 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2006, pp. 1–12.
- [106] A. Ramírez, J.R. Romero, S. Ventura, An extensible JCLEC-based solution for the implementation of multi-objective evolutionary algorithms, in: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15*, ACM, New York, NY, USA, 2015, pp. 1085–1092.
- [107] F.J. Rodríguez, C. García-Martínez, M. Lozano, Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison and synergy test, *IEEE Transactions on Evolutionary Computation* 16 (6) (2012) 787–800.
- [108] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1995.
- [109] B.A. Santos, Aspects conceituais e arquiteturais para a criação de linhagens de agentes de software cognitivos situados, Master's thesis, Federal Center of Technological Education of Minas Gerais (CEFET-MG), Belo Horizonte, Brasil, 2003.
- [110] M.A.L. Silva, Modelagem de uma arquitetura multiagente para a criação , via metaheurísticas, de problemas de criação combinatória, Master's thesis, Federal Center of Technological Education of Minas Gerais (CEFET-MG), Belo Horizonte, Brazil, 2007.
- [111] M.A.L. Silva, S.R. de Souza, H.E. Borges, S.M. de Oliveira, E.C.C. Temponi, AMAM: Arquitetura multiagente para a criação , via metaheurísticas, de problemas de otimização,, in: *Proceedings of the 8th Brazilian Symposium on Intelligent Automation (Simpósio Brasileiro de Automaç ao Inteligente - SBAI)*, Florianópolis, Brazil, 2007.
- [112] M.A.L. Silva, S.R. de Souza, S.M. de Oliveira, M.J.F. Souza, An agent-based metaheuristic approach applied to the vehicle routing problem with time-windows, in: *Proceedings of the 2014 Brazilian Conference on Intelligent Systems - Enc. Nac. de Inteligência Artificial e Computacional (BRACIS-ENIAC 2014)*, São Carlos, SP, Brazil, 2014.
- [113] M.A.L. Silva, S.R. de Souza, M.J.F. Souza, S.M. de Oliveira, A multi-agent metaheuristic optimization framework with cooperation, in: *Proceedings of the 2015 Brazilian Conference on Intelligent Systems (BRACIS)*, Natal, Brazil, Nov 2015, pp. 104–109.
- [114] D. Sislak, M. Rehak, M. Pechoucek, A-globe: multi-agent platform with advanced simulation and visualization support, *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (Sept 2005)* 805–808.
- [115] M.J.F. Souza, I.M. Coelho, S. Ribas, H.G. Santos, L.H.C. Merschmann, A hybrid heuristic algorithm for the open-pit-mining operational planning problem, *European Journal of Operational Research* 207 (2) (2010) 1041–1051.
- [116] E.-G. Talbi, A taxonomy of hybrid metaheuristics, *Journal of Heuristics* 8 (5) (2002) 541–564.
- [117] S. Talukdar, L. Baerentzen, A. Gove, P. de Souza, Asynchronous teams: cooperation schemes for autonomous agents, *Journal of Heuristics* 4 (4) (1998) 295–321.
- [118] S. Talukdar, S. Murthy, R. Akkiraju, Asynchronous teams, in: F. Glover, G.A. Kochenberger (Eds.), *Handbook of Metaheuristics*. Vol. 57 of *International Series in Operations Research & Management Science*, Springer US, 2003, pp. 537–556.
- [119] S. Talukdar, P.S. Souza, Asynchronous Teams, in: *Proceedings of the Second SIAM Conference on Linear Algebra: Signals, System and Control*, San Francisco, USA, 1990.
- [120] J. Toutouh, E. Alba, Parallel swarm intelligence for VANETs optimization, *Proceedings of the 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (2012) 285–290.
- [121] H.W.J.M. Triesenekens, A. Bruin, Towards a taxonomy of parallel branch and bound algorithms, *Tech. Rep. Report EUR-CS-92-01*, Erasmus University Rotterdam, Department of Computer Science, 1992.
- [122] W. Turek, J. Stypka, D. Krzywicki, P. Anielski, K. Pietak, A. Byrski, M. Kisiel-Dorohinicki, Highly scalable Erlang framework for agent-based metaheuristic computing, *Journal of Computational Science* 17 (Part 1) (2016) 234–248.
- [123] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás, JCLEC: a java framework for evolutionary computation, *Soft Computing* 12 (4) (2008) 381–392.
- [124] A. Villaverde, J. Egea, J. Banga, A cooperative strategy for parameter estimation in large scale systems biology models, *BMC Systems Biology* 6 (1) (2012) 75.
- [125] S. Wagner, M. Affenzeller, HeuristicLab Grid: a flexible and extensible environment for parallel heuristic optimization, *Proceedings of the 15th International Conference on Systems Science*. Vol. 1 (2004) 289–296.
- [126] S. Wagner, M. Affenzeller, HeuristicLab: a generic and extensible optimization environment, in: B. Ribeiro, R.F. Albrecht, A. Dobnikar, W. Pearson, D.N.C. Steele (Eds.), *Proceedings of the International Conference in Adaptive and Natural Computing Algorithms*, Coimbra, Portugal, 2005, Springer, Vienna, 2005, pp. 538–541.
- [127] S. Wagner, A. Beham, G.K. Kronberger, M. Kommenda, E. Pitzer, M. Kofler, S. Vonolfen, S.M. Winkler, V. Dorfer, M. Affenzeller, HeuristicLab 3.3: a unified approach to metaheuristic optimization, in: *Proceedings of the VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2010)*, Valencia, Spain, 2010.
- [128] S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, M. Affenzeller, Architecture and design of the HeuristicLab optimization environment, in: R. Klempous, J. Nikodem, W. Jacka, Z. Chaczko (Eds.), *Advanced Methods and Applications in Computational Intelligence*. Vol. 6 of *Topics in Intelligent Engineering and Informatics*, Springer, 2014, pp. 197–261.
- [129] D.R. White, Software review: the ECJ toolkit, *Genetic Programming and Evolvable Machines* 13 (1) (2012) 65–67.
- [130] G.C. Wilson, A.M. Intyre, M. Heywood, Resource review: three open source systems for evolving programs – LiLgp, ECJ and Grammatical Evolution, *Genetic Programming and Evolvable Machines* 5 (1) (2004) 103–105.
- [131] M. Wooldridge, *An Introduction to Multiagent Systems*, 2nd ed., John Wiley & Sons, 2009.
- [132] T. Yamaguchi, Y. Tanaka, M. Yachida, Speed up reinforcement learning between two agents with adaptive mimetism, *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1997 (IROS'97). Vol. 2 (1997) 594–600.
- [133] Y. Zheng, X. Xu, S. Chen, W. Wang, Distributed agent based cooperative differential evolution: a master-slave model, *Proceedings of the 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS)*. Vol. 1 (Oct 2012) 376–380.
- [134] D. Żurek, K. Pjetak, M. Pietron, M. Kisiel-Dorohinicki, Toward hybrid platform for evolutionary computations of hard discrete problems, *Procedia Computer Science* 108 (2017) 877–886, international Conference on

Computational Science, ICCS 2017, 12–14 June 2017, Zurich,
Switzerland.

Maria Amélia Lopes Silva. B.Sc. in Computational Science (2002) at Formiga University Center. M.Sc. in Mathematical and Computational Modeling (2007) at Federal Center of Technological Education of Minas Gerais. Ph.D. candidate in Mathematical and Computational Modeling at Federal Center of Technological Education of Minas Gerais. Her main research interests include multi-agents systems, metaheuristics, optimization frameworks and vehicle routing. She has published, by the end of 2017, 17 full papers in conferences. More information is available at <http://lattes.cnpq.br/1584173805850799>.

Sérgio Ricardo de Souza. Full Professor at Computer Science Department of Federal Center of Technological Education of Minas Gerais. B.Sc. in Electrical Engineering from PUC-MG. M.Sc. and Ph.D. in Electrical Engineering from UNICAMP in 1991 and 1994, respectively. His main research interests include metaheuristics, production scheduling, vehicle routing, timetabling and multi-agent systems for optimization. He has published, by the end of 2017, 16 full papers in journals and 189 full papers in conferences. More information is available at <http://lattes.cnpq.br/3677015295211434>.

Marcone Jamilson Freitas Souza. Full Professor at Computer Science Department of Federal University of Ouro Preto (UFOP). B.Sc. in Metallurgical Engineering from UFOP. M.Sc. and Ph.D. in Systems Engineering and Computing from Federal University of Rio de Janeiro (UFRJ) in 1989 and 2000, respectively. His main areas of research are: combinatorial optimization, metaheuristics, scheduling, timetabling, open-pit mine planning, vehicle routing, public transport and multi-agent systems for optimization. He has published, by the end of 2017, 48 full papers in journals, 09 book chapters and 201 full papers in conferences. More information is available at <http://lattes.cnpq.br/6078945717558464>.

Moacir Felizardo de França Filho. Assistant Professor at Mechanical Engineering Department of Federal Center of Technological Education of Minas Gerais. B.Sc. in Mechanical Engineering from PUC-MG. M.Sc. in Mechanical Engineering from Federal University of Santa Catarina (1993) and Ph.D. in Electrical Engineering from UNICAMP in 2007. His main areas of research are metaheuristics and scheduling problems on parallel machines. He has published, by the end of 2017, 03 full papers in journals and 28 full papers in conferences. More information is available at <http://lattes.cnpq.br/1302666258264971>.