Evolutionary Computation Meets Multiagent Systems for Better Solving Optimization Problems

Vinicius Renan de Carvalho(✉) and Jaime Simão Sichman

Laboratório de Técnicas Inteligentes (LTI), Escola Politécnica (EP), University of Sao Paulo (USP), São Paulo, Brazil
{vrcarvalho,jaime.sichman}@usp.br

Abstract. In this work, we discuss the synergy between Evolutionary Computation (EC) and Multi-Agent Systems (MAS) when both are used together to enhance the process of solving optimization problems. Evolutionary algorithms are inspired by nature and follow Darwin theory of the fittest. They are usually applied where there is no specific algorithm which can solve optimization problems in a reasonable time. Multi-Agent Systems, in their turn, are collections of autonomous entities, named agents, that sense their environment and execute some actions in the environment to meet their individual or common goals. When these two techniques are applied together, one can create powerful approaches to better solve optimization problems. This paper presents an overview of this combined approach, considering both mono-objective and multi-objective approaches. In particular, we stress the importance of hyper-heuristic approaches, i.e., heuristics that help to choose the best EC algorithm among a candidate set.

Keywords: Hyper-heuristics · Multi-objective Evolutionary Algorithms · Voting methods · Agent cooperation

1 Introduction

An *optimization problem* is the problem of finding the best possible solution among a set of candidate solutions. More precisely, it aims to find a solution in the feasible region that minimizes (or maximizes) the value of a certain objective function [2]. In an optimization problem, solutions are composed of a set of decision variables $s = (x_1, \ldots, x_n)$, whose values belong to a set of domains $D = (D_1, \ldots, D_n)$. These domains can be either continuous ($D = \mathbb{R}$) or discrete ($D = \mathbb{Z}$). Thus, solutions can solve discrete or continuous optimization problems. In order to determine which solution better solves a given optimization problem, it is necessary to define a *fitness function* $f(s)$ that evaluates the quality of each solution.

F. Koch et al. (Eds.): GEAR 2018, CCIS 999, pp. 27–41, 2019.
https://doi.org/10.1007/978-981-13-6936-0_4

However, many real-world problems are multi-criteria, and thus need the specification of multiple fitness functions in order to evaluate their solutions. This is the case, for example, of buying a car considering price and fuel consumption. These problems are called *Multi-Objective Problems* (MOPs), where the solutions should optimize different and often conflicting criteria [16]. Usually, classical exact optimization methods cannot be used to deal with MOPs and more sophisticated techniques are required. In this paper, evolutionary algorithms and hyper-heuristics are addressed to solve both mono-objective and multi-objective optimization problems.

2 Evolutionary Computation

According to Pearl [38], *heuristics* can be defined as criteria, methods or principles for deciding which among several alternatives courses of action promises to be the most effective in order to achieve some goal. Heuristics do not guarantee optimal solutions; in fact, they do not guarantee any solutions at all: all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time [20].

Meta-heuristics, in turn, can be defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, using strategies to structure information in order to find efficiently near-optimal solutions [35]. Usually, heuristics are specialized in solving problems for one particular domain, while meta-heuristics are more generic and adaptive in several domains.

One of the most used meta-heuristics are classified as *Evolutionary Computation* (EC) algorithms. It is the general term for several optimization algorithms that are inspired by the Darwinian principles of nature's capability to evolve living beings well adapted to their environment [7]. These algorithms are also called as *Evolutionary Algorithms* (EA), and they all share a common underlying idea of simulating the evolution of individual (or solution) structures via processes of selection, recombination, and mutation reproduction, thereby producing better solutions [7].

2.1 Evolutionary Algorithms

In the literature, we can find some algorithms that implement the concept of an evolutionary algorithm. That is the case of *Genetic Algorithm* [22] and of *Genetic Programming* [25]. In a genetic algorithm, individuals from the population compete and generate offspring using crossover and mutation. Genetic Programming employs more complex data representation than GA, such as a tree, to represent individuals. Thus, allowing individuals to have different lengths.

Both algorithms are focused on mono-objective optimization, that means, one single value to represent the quality of a given solution. However, several real-world problems considerate more than one fitness value in order to

properly evaluate an individual quality. In this scenario, Multi-objective Evolutionary Algorithms (MOEAs) are able to find good solutions for this kind of problems [16].

Choosing an algorithm to solve a particular optimization problem is not a trivial task. Usually, a tuning method is necessary, if there is no previous knowledge about which algorithm to use and what is the recommended algorithm configuration to solve a given optimization problem. The tuning process consists in solving an optimization problem using different algorithm running instances (where each instance represent a different configuration), taking their results, and finding the best according to a quality indicator. The tuning task has to be executed for several times, since meta-heuristics are not deterministic algorithms. Some research has been proposed in order to reduce this difficult task. That is the case of *hyper-heuristics* [8].

2.2 Hyper-Heuristics

The motivation for proposing hyper-heuristic came from the "No Free Lunch Theorem" which establish that "for any algorithm, any elevated performance over one class of problems is an offset by diminished performance over another class" [45].

Hence, hyper-heuristics are defined as a high-level methodology that—given a particular problem instance or class of instances, and some low-level heuristics (LLH), or components—automatically produces an adequate combination of them to solve the problem efficiently [8].

Most of the research focuses on the selection of online hyper-heuristics, meaning that the process of finding solutions for the optimization problem tries to figure out which LLH should be given more time to execute. These works usually consider heuristics such as differential evolution, crossover, and mutation as low-level heuristics (LLH). However, there are some works which consider the whole algorithms as LLHs. The majority of research in this area has been limited to focus on single-objective optimization problems [27].

Evolutionary algorithms and hyper-heuristics have been developed along the years, most of the times considering centralized implementations. Given their number of complex components, they could be built as a multi-agent system. In the next section, we analyze how these areas have been applied together.

3 Multi-Agent Systems

According to Wooldridge [46], an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives. An agent perceives the environment by sensors. Thus, based on his perceptions and considering his internal knowledge and beliefs, the agent can plan how to act, that means, using his actuators in order to update the environment. A multi-agent system (MAS) is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure [46].

4 Combining Multi-Agent Systems and Evolutionary Computation

Evolutionary algorithms can be used by an agent in order to perform some tasks, such as learning, parameter estimations, or to support coordination of some group (team) activity, e.g. planning [39]. On the other hand, evolutionary algorithms can be built using the MAS background, which can provide the mechanisms for a decentralized search.

4.1 Multi-Agent Systems and Evolutionary Algorithms

Several approaches have been proposed for combining MAS and Evolutionary Algorithms. We can classify them as the following:

- Solutions as passive agents: in this case, solutions may act autonomously for instance, by selecting which partner would be more interesting to generate new offspring. In this category, agents are passive and do not perform any operation, but the algorithm perform operations;
- Solutions as active agents: in this case, solutions are agents which actively can generate new offspring. In this case, agents contain operations such as crossover and mutation;
- Algorithms or algorithm components as agents: in this case, complete algorithms instances are considered to generate hybrid approaches where algorithms agents share their solutions along the search;
- Multi-objective: to identify which approach deals with multi-objective optimization;
- Specially Organized: means if agents are organized in a specific structure, such as lattice and neighborhood.

Solutions as Passive Agents: In most research, agents represent a complete feasible solution (an individual in the population) to a given optimization problem. Thus, agents can cooperate (usually by crossover reproduction), compete for survival, observe and communicate with the environment. In this case, instead of the centralized view of a population of solutions, these works consider a population of agents, each one representing a solution. Using this concept, Eiben et al. [19] solved Travel Salesman Person problem by implementing a blackboard system which allowed the interchange among agents. Thus, each agent could access other agent's current solution by means of the blackboard mechanism.

In [39], solution agents also considered a non-renewable resource named life energy, which is increased or decreased based on how well an agent solve the given problem. This kind of agent representation was also applied in [47] for dynamic optimization problems, where the goal of an algorithm is no longer to find an optimal solution but to track the moving optima in the search. In [15] propose an approach to solve multi-objective problems: in this approach, each

agent is invited to be a mate and can accept or decline the proposal according to its own strategy. These agents can be part of a society of agents, and can only accept agents from the same society as mates. Offspring are assigned to a certain society according to a dominance concept. In [43], the authors present an agent-based memetic algorithm for solving nonlinear optimization problems with equality constraints, where agents can be either cooperative or competitive through the simulated binary crossover or a proposed Life Span Learning Process. Chalupa [13] proposes a Multi-agent evolutionary algorithm (MEA). In this approach, each agent has its lifespan assigned when the solution is created. MEA performs many short-term local search subroutines, instead of long-term local search. After that, a collection of promising solutions are stored in the elite list. Thus, MEA relies on a well-tuned process of highly organized restarts from promising solutions and performing short-term local search subroutines to improve these solutions. In [44] the knowledge-based multi-agent evolutionary algorithm (KMEA) is proposed for solving the semiconductor final testing scheduling problem. KMEA has two phases: mutual-learning and competition. In the mutual-learning, each agent learns from the best one of its neighbors to obtain a better solution. In the competition, each agent competes with its current best neighbor. If it loses the competition, it will be replaced by a new agent generated according to the knowledge base.

Some research defined how solution agents should be set in the agent community. That is the case of [6], where agents reside in a grid, and each cell in the grid contains one agent. A similar approach was employed by Sun and Zhou [40], where the authors found solutions for the multi-objective energy resource scheduling of micro-grid, by considering a micro-grid as agents. In [37], all agents live in a lattice-like environment and die when the energy finishes. They can only interact with their neighbors. The local environments of all the agents are constructed by a social acquaintance net. Zeng et al. [48] also set agents in a lattice in order to find solutions for the Assembly Sequence Planning.

Solutions as Active Agents: Like in mentioned researches, some studies treat solution as agents, but they also consider the operator which generate the solution as a component. That is the case of [30], where agents have additional capacities of decision, learning, and cooperation, by using several operators which are scheduled by an adaptive decision process. The decision rules of the agents are adapted during the optimization process by reinforcement learning and mimetism. In [24] proposed a novel multi-agent multi-objective evolutionary framework based on trust where each solution is represented as an intelligent agent, and evolutionary operators and control parameters are represented as services. Agents select services in each generation based on trust that measures the competency or suitability of the services for solving particular problems. Huang et al. [23] propose an approach to solve the software module clustering problem (SMCP). They designed three evolutionary operators as agents: neighborhood competition operator, the mutation operator, and the self-learning operator. They also employed a similar energy concept used in earlier works. In [26] they

solved the 3-D protein structure prediction using an approach which considers crossover and swap operations as agents. Each agent work with the complete protein (torsional angles) representation, called solutions, like in an evolutionary algorithm [26]. Drezewski and Siwik [18] designed a co-evolution multi-agent system, where an agent is a combination of the solutions and operations it can perform, such as clone, recombination and resources management. This approach was evaluated using ZDT benchmark, a multi-objective optimization benchmark.

Algorithms or Algorithm Components Agents: Some research treats other evolutionary components or even a full algorithm instance as agents. Zheng et al. [49] treat the whole population of solutions as an agent. This approach also keeps a master-agent is responsible for evolving the solutions for the original problem, and it is modeled as a top-level agent that is an asynchronous team, consisting of a population of complete solutions and a set of sub-agents working on the population. The authors evaluated their approach using mono-objective optimization problems such as Rosenbrock and Sphere. In [4] agents are the one which finds solutions for the optimization problem. There are two types of agents: master and slave. A master agent generates the population and divides them into sub-populations. It sends them to slave agents, responsible for executing conventional genetic algorithm steps on their sub-population, and periodically return their best partial results to the master agent. The master stores the partial results in lists. Denzinger and Offerman [17] present TECHS approach (TEams for Cooperative Heterogeneous Search), where Genetic Algorithms and Branch-and-Bound are considered as agents to solve the mono-objective job-shop-scheduling problem. Each agent gets the whole instance of the problem to solve, tries to solve the instance according to its search paradigm, and periodically interrupts its search in order to send information to other agents and to receive information from them. Chatzinikolaou and Roberteson [14] designed agents to contain a standard, canonical GA that acts on a local population of solutions, performing standard crossover and mutation on them. This paper also investigates the effectiveness with which reputation can replace direct fitness observation as the selection bias in an evolutionary multi-agent system. This is performed by implementing a peer-to-peer, self-adaptive genetic algorithm, in which agents act as individual GAs that, in turn, evolve dynamically themselves in real-time (namely the parameters employed by them). The evolution of the agents is implemented in two alternative ways: First, using the traditional approach of direct fitness observation (self-reported by each agent), and second, using a simple reputation model based on the collective past experiences of the agents. The authors validated their approach applying Rastringin, a mono-objective benchmark problem.

In [33], considered a genetic algorithm and a set of search techniques as agents to solve the mono-objective flexible job shop scheduling problem (FJSP). The genetic algorithm is employed for global exploration of the search space, the search set agents guide the research in promising regions of the search space and to improve the quality of the final population.

In [21] proposed a multi-agent hybrid algorithm composed by two agents: MOEA/D with a resource allocation mechanism (an evolutionary algorithm) and a local search method for the two-objective deteriorating scheduling.

Table 1 summaries all agent-based evolutionary algorithms, by classifying them according to how they implement agents in their design, if they implement these concepts on solutions, heuristics operator, algorithms instances. We can see that most papers employs solutions as agents and are designed for mono-objective optimization.

4.2 Multi-Agent Systems and Hyper-Heuristics

Several approaches have been proposed for combining MAS and Hyper-Heuristics. We can classify them as the following:

- Heuristic as agent: approaches which consider operators such as crossover and mutation as agents;
- Algorithm as agent: where a complete algorithm instance is considered as an agent;
- Both algorithms and heuristics as agents: approaches which consider as agents both different algorithm instances and heuristics as agents;
- Multi-objective: to identify which approach deals with multi-objective optimization;
- Cooperative or Competitive: if agents try to cooperate with each other or compete for computational resources.

Heuristic as Agent: Ouelhadj and Petrovic [36] employed search operators, such as Swap, Inversion, Insertion, and Permutation, as LLH agents to solve Permutation Flow Shop. This is a cooperative hyper-heuristic, where the heuristic agents perform a local search through the same solution space starting from the same or different initial solution and using different low-level heuristics. The agents exchanges their best solutions. After a generation, the best solutions are selected from all agents. This approach performs a greedy selection strategy to select an LLH to execute.

Meignan et al. [31] propose a selection hyper-heuristic where agents are responsible for concurrently explore the search space of an optimization problem in a cooperative way, where agents organized in a coalition cooperate by the exchanging of information about the search space and their experiences in order to improve agents behaviors. In order to generate new solutions, an agent uses several heuristics which are scheduled by an adaptive decision process, based on heuristic rules adapted along the optimization process by individual learning and. In this approach, a search agent keeps three solutions: the current, the best-found solution of the agent and the best solution of the entire coalition, and it can employ several operators on its current solution. This approach was applied to solve the Vehicle Routing Problem (VRP).

Table 1. Meta-heuristics papers classification

Paper	Solution as passive agent	Solution as active agent	Algorithms or components as agent	Multi-objective	Specially organized
Eiben et al. [19]	✓				
Socha and Kisiel-Dorohinicki [39]	✓				
Yan et al. [47]	✓				
Chira et al. [15]	✓			✓	✓
Ullah et al. [43]	✓				
Chalupa [13]	✓				
Wang and Wang [44]	✓				✓
Belkhelladi et al. [6]	✓				✓
Sun and Zhou [40]	✓				✓
Pan and Chen [37]	✓				✓
Zeng et al. [48]	✓				✓
Meignan et al. [30]		✓			
Jiang et al. [24]		✓		✓	
Huang et al. [23]		✓			
Corrêa et al. [26]		✓		✓	
Drezewski and Siwik [18]		✓		✓	
Zheng et al. [49]			✓		
Balid and Minz [4]			✓		
Denzinger and Offerman [17]			✓		
Chatzinikolaou and Robertson [14]			✓		
Nouri et al. [33]			✓		
Fu et al. [21]			✓	✓	

Algorithm as Agent: Cadenas et al. [9] introduced a cooperative multi-agent meta-heuristic approach, where agents communicate their best solutions using a common blackboard. This blackboard is monitored by a coordinator agent who is responsible to modify meta-heuristic agents behaviors based on fuzzy rules which take in account algorithms performance in the search. The authors tested their approach using 0/1 knapsack problems.

Malek [28] proposes a multi-agent hyper-heuristic to solve several combinatorial problems by considering GA, TS, SA, PSO, ACO as algorithm agents. In this approach, there is also a Problem agent, a Solution Pool agent (responsible to keep all solutions), and Adviser agent, an agent who provides parameter settings for the algorithms and receives reports from them. All algorithm agents of the same kind are associated with a common Adviser agent.

de Carvalho and Sichman [10,12] propose a Multi-Objective Agent-Based Hyper-Heuristic (MOABHH), an agent-based multi-objective hyper-heuristic focused on selecting the most suitable multi-objective evolutionary algorithm during execution time. MOABHH used the concept of voting to define which algorithm should receive a bigger participation in the generation of solutions. As a voting procedure, they applied the Copeland voting method and employed a set of voter agents responsible for evaluating algorithms (composed by NSGA-II, IBEA, and SPEA2) performance according to different quality indicators (composed by Hypervolume, Spread, RNI, GD, and IGD); these are usually used by the MOP community to compare the performance of MOEAs. After the voting, the most voted candidate received a bigger participation on the next offspring generation. In [11], the authors extended their work to solve four real-world engineering optimization problems. In this work, IGD and GD were replaced by HR and ER due to the fact of these indicators need previous problem knowledge, to make the approach applicable to real-world problems. This paper also set GDE3 as MOEA agent.

Acan and Lotfi [1] propose a collaborative hyper-heuristic architecture designed for multi-objective real-parameter optimization problems. In their approach, the population of solutions is split into sub-populations based on Pareto dominance, and then these sub-populations are assigned each one to a meta-heuristic agent, based on a cyclic or round-robin order, making meta-heuristics agents in this approach each operates on a sub-population in subsequent sessions. Meta-heuristic agents have their own population of non-dominated solutions extracted in a session, while there is also a global population of solutions keeping all non-dominated solutions found in the search. This study set MOGA, NSGAII, SPEA2, MODE, IMOPSO, AMOSA as meta-heuristic agents, and it was evaluated considering the CEC2009 benchmark.

Nugraheni and Abednego [34] propose an approach to select one of three agent Hyper-heuristics based on Genetic Programming (GPHH agent), Genetic Algorithm Hyper-Heuristic (GAHH agent), and Simulated Annealing Hyper-Heuristics (SAHH agent). These HH agents choose some low-level heuristics and work in search space of heuristics rather than a space of solutions directly.

Both Algorithms and Heuristics as Agents: Talukdar et al. [42] propose A-Teams, a synergistic team of problem-solving methods which cooperates by sharing a population of candidate solutions. In this approach, there is no coordination or planning mechanism, solutions are shared, through the central memory mechanism, allowing other agents to use these solutions in order to guide the search through promising search space, thus reducing the chances of being stuck at a local optimum. Aydin and Fogarty [3] extended A-Teams approach for solving Job Shop Scheduling. They employed as problem solving agents: SA (Simulated Annealing), TS (Taboo Search), HC (Hill Climbing), CSA (Simulated Annealing), CTS (Taboo Search), CHC (Hill Climbing), CHC2 (Hill Climbing), GA (Genetic Algorithm), NT (Improved version of CTS), and Damage. Barbucha [5] also extended the A-Team approach in order to create an Agent-Based Cooperative Population Learning Algorithm for the Vehicle Routing Problem with Time Windows. In his approach, the search is treated into stages, and different search procedures are used at each stage. The first stage is organized as an A-Team, where agents are used for improving the individuals stored in the common memory. In the second stage, the individuals in the population (common memory) are divided into subpopulations and allocated to a different set of A-Teams. In this level, each A-Team uses the same heuristics working under the same cooperation scheme. In the third stage, the sub-populations and the team of A-Teams architecture are also being employed. However, the process of communication among the set of A-Teams is used. The author evaluated his approach setting five problem-specific heuristics in the first stage, and a set of four Tabu Search and simulated annealing in the higher level.

Milano and Roli [32] presented the Multi-agent Meta-heuristic Architecture (MAGMA), a four-level architecture, with one or more agents at each level, where each level one or more agents act. The first level contains solution builders agents, responsible for providing feasible solutions for upper levels. The second level contains solution improvers, responsible for providing local search and solution improvements until a termination condition is verified. The third-level agents have a global view of the search space, or, at least, their task is to guide the search towards promising regions trying to avoid local optima. In the last level (Level-3) higher level strategies are described, such as a cooperative search and any other combination of meta-heuristics. The authors showed the three first levels are enough to describe standalone meta-heuristics and then evolutionary algorithms. Besides that, Level-3 can model coordinated cooperative hybrid meta-heuristics.

Talbi and Bachelet [41] propose a hybrid approach to solve the quadratic assignment problem by applying Tabu Search, Genetic Algorithm e KO (kick operator) as cooperative agents. The three heuristic agents run simultaneously and exchange information via an adaptive memory (AM). Each algorithm has a role: the Tabu Search is used as the main search algorithm, the Genetic Algorithm is in charge of the diversification and the Kick Operator is applied to intensify the search.

Martin et al. [29] propose a multi-agent hyper-heuristic where each agent implements a different meta-heuristic/local search combination. These agents

also adapt itself along the search by using a proposed cooperation protocol based on reinforcement learning and pattern matching. Two kinds of agents are employed: launcher and meta-heuristic agents. The launcher is responsible for instantiating and keep the optimization problem, set up algorithms, and create initial solutions. The meta-heuristic agent contains an algorithm responsible for searching collectively for good quality solutions. The authors evaluated their approach using the mono-objective problems: Permutation Flow Shop, CVRP e Nurse Rostering and employing RandCWS, RandNEH as meta-heuristic agents.

Table 2 summaries all agent-based hyper-heuristics, by classifying them according to how they implement agents in their design. These paper are classified according to how they employ agents, if they are designed for multi-objective optimization, and if they are cooperative and competitive. Most of the research deals with mono-objective optimization and are cooperative.

Table 2. Hyper-heuristics papers classification

Paper	Heuristic as agent	Algorithm as agent	Algorithm and heuristic as agent	Multi-objective	Cooperative	Competitive
Talukdar et al. [42]			✓		✓	
Aydin and Fogarty [3]			✓		✓	
Barbucha [5]			✓		✓	
Milano and Roli [32]			✓		✓	
Talbi and Bachelet [41]			✓		✓	
Cadenas et al. [9]		✓			✓	
Malek [28]		✓			✓	
Ouelhadj and Petrovic [36]	✓				✓	✓
Meignan et al. [31]	✓				✓	
Martin et al. [29]			✓		✓	
de Carvalho and Sichman [10,11]		✓		✓	✓	✓
Acan and Lotfi [1]		✓		✓	✓	
Nugraheni and Abednego [34]		✓			✓	

5 Conclusions

This paper presented the synergy between Evolutionary Computation (EC) and Multi-Agent Systems (MAS) when both are used together to enhance the process of solving optimization problems.

We have proposed a criteria to characterize how agent theory is considered in order to design evolutionary algorithms, hybrid algorithms and hyper-heuristics by identifying which component is considered an agent and how this agent acts.

One can notice that this synergy has shown several interesting results, especially concerning the case of hyper-heuristics, due to the fact these approaches combine different heuristics and algorithm in order to better explore the search space. This set of components can naturally be designed as an agent-based system.

Acknowledgements. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Vinicius Renan de Carvalho was also supported by CNPq, Brazil, under grant agreement no. 140974/2016-4.

References

1. Acan, A., Lotfi, N.: A multiagent, dynamic rank-driven multi-deme architecture for real-valued multiobjective optimization. Artif. Intell. Rev. **48**(1), 1–29 (2017)
2. Atallah, M.J.: Algorithms and Theory of Computation Handbook. CRC Press, Boca Raton (1998)
3. Aydin, M.E., Fogarty, T.C.: Teams of autonomous agents for job-shop scheduling problems: an experimental study. J. Intell. Manuf. **15**(4), 455–462 (2004)
4. Balid, A., Minz, S.: Improving multi-agent evolutionary techniques with local search for job shop scheduling problem. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 516–521, December 2008
5. Barbucha, D.: A cooperative population learning algorithm for vehicle routing problem with time windows. Neurocomputing **146**, 210–229 (2014). Bridging Machine learning and Evolutionary Computation (BMLEC) Computational Collective Intelligence
6. Belkhelladi, K., Chauvet, P., Schaal, A.: An agent framework with an efficient information exchange model for distributed genetic algorithms. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 848–853, June 2008
7. Boussaid, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013)
8. Burke, E.K., et al.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
9. Cadenas, J.M., Garrido, M.C., Munoz, E.: A cooperative system of metaheuristics. In: 7th International Conference on Hybrid Intelligent Systems (HIS 2007), pp. 120–125, September 2007
10. de Carvalho, V.R., Sichman, J.S.: Applying copeland voting to design an agent-based hyper-heuristic. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 972–980 (2017)

11. de Carvalho, V.R., Sichman, J.S.: Solving real-world multi-objective engineering optimization problems with an Election-Based Hyper-Heuristic. In: International Workshop on Optimisation in Multi-agent Systems (OPTMAS 2018) (2018)
12. de Carvalho, V.R., Sichman, J.S.: Multi-agent election-based hyper-heuristics. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 5779–5780 (2018)
13. Chalupa, D.: Adaptation of a multiagent evolutionary algorithm to NK landscapes. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2013, Companion, pp. 1391–1398. ACM, New York (2013)
14. Chatzinikolaou, N., Robertson, D.: The use of reputation as noise-resistant selection bias in a co-evolutionary multi-agent system. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO 2012, pp. 983–990. ACM, New York (2012)
15. Chira, C., Gog, A., Dumitrescu, D.: Exploring population geometry and multi-agent systems: a new approach to developing evolutionary techniques. In: Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2008, pp. 1953–1960. ACM, New York (2008)
16. Coello, C.: Evolutionary Algorithms for Solving Multi-objective Problems. Springer, New York (2007). https://doi.org/10.1007/978-0-387-36797-2
17. Denzinger, J., Offermann, T.: On cooperation between evolutionary algorithms and other search paradigms. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 3, p. 2324 (1999)
18. Drezewski, R., Siwik, L.: Agent-based multi-objective evolutionary algorithm with sexual selection. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 3679–3684, June 2008
19. Eiben, E.A., Schoenauer, M., Laredo, J.L.J., Castillo, P.A., Mora, A.M., Merelo, J.J.: Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In: Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2007, pp. 2801–2808. ACM, New York (2007)
20. Feigenbaum, E.A., Feldman, J., et al.: Computers and Thought. ACM, New York (1963)
21. Fu, Y., Wang, H., Tian, G., Li, Z., Hu, H.: Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. J. Intell. Manuf., 1–16 (2018)
22. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. **3**(2), 95–99 (1988)
23. Huang, J., Liu, J., Yao, X.: A multi-agent evolutionary algorithm for software module clustering problems. Soft Comput. **21**(12), 3415–3428 (2017)
24. Jiang, S., Zhang, J., Ong, Y.S.: A multiagent evolutionary framework based on trust for multiobjective optimization. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, vol. 1, pp. 299–306, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2012)
25. Koza, J.R.: Evolution of subsumption using genetic programming. In: Proceedings of the First European Conference on Artificial Life, pp. 110–119 (1992)
26. de Lima Corrêa, L., Inostroza-Ponta, M., Dorn, M.: An evolutionary multi-agent algorithm to explore the high degree of selectivity in three-dimensional protein structures. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1111–1118, June 2017

27. Maashi, M., Özcan, E., Kendall, G.: A multi-objective hyper-heuristic based on choice function. Expert Syst. Appl. **41**(9), 4475–4493 (2014)
28. Malek, R.: An agent-based hyper-heuristic approach to combinatorial optimization problems. In: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 3, pp. 428–434, October 2010
29. Martin, S., Ouelhadj, D., Beullens, P., Ozcan, E., Juan, A.A., Burke, E.K.: A multi-agent based cooperative approach to scheduling and routing. Eur. J. Oper. Res. **254**(1), 169–178 (2016)
30. Meignan, D., Créput, J.C., Koukam, A.: A cooperative and self-adaptive metaheuristic for the facility location problem. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO 2009, pp. 317–324. ACM, New York (2009)
31. Meignan, D., Koukam, A., Créput, J.C.: Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism. J. Heuristics **16**(6), 859–879 (2010)
32. Milano, M., Roli, A.: MAGMA: a multiagent architecture for metaheuristics. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **34**(2), 925–941 (2004)
33. Nouri, H.E., Belkahla Driss, O., Ghédira, K.: Metaheuristics based on clustering in a holonic multiagent model for the flexible job shop problem. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion 2015, pp. 997–1004. ACM, New York (2015)
34. Nugraheni, C.E., Abednego, L.: Multi agent hyper-heuristics based framework for production scheduling problem. In: 2016 International Conference on Informatics and Computing (ICIC), pp. 309–313, October 2016
35. Osman, I.H., Laporte, G.: Metaheuristics: a bibliography. Ann. Oper. Res. **63**, 511–623 (1996)
36. Ouelhadj, D., Petrovic, S.: A cooperative hyper-heuristic search framework. J. Heuristics **16**(6), 835–857 (2010)
37. Pan, X., Chen, H.: A multi-agent social evolutionary algorithm for resource-constrained project scheduling. In: 2010 International Conference on Computational Intelligence and Security, pp. 209–213, December 2010
38. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Longman Publishing Co. Inc., Boston (1984)
39. Socha, K., Kisiel-Dorohinicki, M.: Agent-based evolutionary multiobjective optimisation. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol. 1, pp. 109–114, May 2002
40. Sun, H., Zhou, C.: Context-aware multi-agent model of microgrid optimization using fuzzy preferences evolutionary algorithm. In: 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), pp. 803–808, June 2013
41. Talbi, E., Bachelet, V.: COSEARCH: a parallel cooperative metaheuristic. J. Math. Model. Algorithms **5**(1), 5–22 (2006)
42. Talukdar, S., Baerentzen, L., Gove, A., De Souza, P.: Asynchronous teams: cooperation schemes for autonomous agents. J. Heuristics **4**(4), 295–321 (1998)
43. Ullah, A.S.S.M.B., Sarker, R., Lokan, C.: An agent-based memetic algorithm (AMA) for nonlinear optimization with equality constraints. In: 2009 IEEE Congress on Evolutionary Computation, pp. 70–77, May 2009
44. Wang, S., Wang, L.: A knowledge-based multi-agent evolutionary algorithm for semiconductor final testing scheduling problem. Knowl. Based Syst. **84**, 1–9 (2015)

45. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
46. Wooldridge, M.: An Introduction to Multiagent Systems. Wiley, Chichester (2009)
47. Yan, Y., Wang, H., Wang, D., Yang, S., Wang, D.: A multi-agent based evolutionary algorithm in non-stationary environments. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 2967–2974, June 2008
48. Zeng, C., Gu, T., Zhong, Y., Cai, G.: A multi-agent evolutionary algorithm for connector-based assembly sequence planning. Proc. Eng. **15**, 3689–3693 (2011). cEIS 2011
49. Zheng, Y., Xu, X., Chen, S., Wang, W.: Distributed agent based cooperative differential evolution: a master-slave model. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, vol. 01, pp. 376–380, October 2012