European Journal of Operational Research 296 (2022) 393–422

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art

Maryam Karimi-Mamaghan[a,*], Mehrdad Mohammadi[a], Patrick Meyer[a], Amir Mohammad Karimi-Mamaghan[b], El-Ghazali Talbi[c]

[a] IMT Atlantique, Lab-STICC, UMR CNRS 6285, Brest F-29238, France
[b] Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
[c] Department of Computer Science, University of Lille, CNRS UMR 9189, Centre de Recherche en Informatique Signal et Automatique de Lille (CRIStAL), Lille F-59000, France

ARTICLE INFO

Article history:
Received 18 December 2020
Accepted 17 April 2021
Available online 24 April 2021

Keywords:
Meta-heuristics
Machine learning
Combinatorial optimization problems
State-of-the-art

ABSTRACT

In recent years, there has been a growing research interest in integrating machine learning techniques into meta-heuristics for solving combinatorial optimization problems. This integration aims to lead meta-heuristics toward an efficient, effective, and robust search and improve their performance in terms of solution quality, convergence rate, and robustness.

Since various integration methods with different purposes have been developed, there is a need to review the recent advances in using machine learning techniques to improve meta-heuristics. To the best of our knowledge, the literature is deprived of having a comprehensive yet technical review. To fill this gap, this paper provides such a review on the use of machine learning techniques in the design of different elements of meta-heuristics for different purposes including *algorithm selection*, *fitness evaluation*, *initialization*, *evolution*, *parameter setting*, and *cooperation*. First, we describe the key concepts and preliminaries of each of these ways of integration. Then, the recent advances in each way of integration are reviewed and classified based on a proposed unified taxonomy. Finally, we provide a technical discussion on the advantages, limitations, requirements, and challenges of implementing each of these integration ways, followed by promising future research directions.

1. Introduction

Meta-heuristics (MHs) are computational intelligence paradigms widely used for solving complex optimization problems, particularly Combinatorial Optimization Problems (COPs) (Osman & Laporte, 1996). COPs are a complex class of optimization problems with discrete decision variables and a finite search space. Many COPs belong to the NP-Hard class of optimization problems that require exponential time to be solved to optimality. For these problems, MHs can provide acceptable solutions in reasonable computational time, and as a result are good substitutes for exact algorithms (Hertz & de Werra, 1990; Osman & Laporte, 1996; Talbi, 2009). It is the main reason behind the significant growth of interest in MH domain in the past two decades.

From a technical point of view, MHs are a family of approximate optimization methods that arrange and pilot an interaction between local improvement procedures and higher-level strategies to create an iterative search process capable to escape from local optima and perform a robust search of a search space (Gendreau & Potvin, 2010). During such an iterative search process, a considerable number of solutions are generated, evaluated, and evolved until a promising solution is obtained. Indeed, during the search process, MHs generate a considerable volume of data including good (elite) or bad solutions in terms of their fitness values, the sequence of search operators from beginning to the end, evolution trajectories of different solutions, local optima, etc. These data potentially carry useful knowledge such as the properties of good and bad solutions, the performance of different operators in different stages of the search process, precedence of search operators, etc.; however, classical MHs do not use any form of knowledge hidden in these data.

Machine Learning (ML) techniques can serve MHs by extracting useful knowledge from the generated data throughout the search process. Incorporating such knowledge within the search

* Corresponding author.
E-mail addresses: maryam.karimi@imt-atlantique.fr (M. Karimi-Mamaghan), mehrdad.mohammadi@imt-atlantique.fr (M. Mohammadi), patrick.meyer@imt-atlantique.fr (P. Meyer), amir.karimi@ut.ac.ir (A.M. Karimi-Mamaghan), el-ghazali.talbi@univ-lille.fr (E.-G. Talbi).

https://doi.org/10.1016/j.ejor.2021.04.032

process guides MHs toward making better decisions and consequently makes MHs more intelligent and significantly improves their performance in terms of solution quality, convergence rate, and robustness. ML is a sub-field of artificial intelligence (AI) that involves learning algorithms to infer from data to learn new tasks (Song, Triguero, & Özcan, 2019). The integration of ML techniques and MHs has attracted intense attention in recent years (Song et al., 2019; Talbi, 2016).

There are recent review papers on the use of MHs in ML tasks (Calvet, de Armas, Masip, & Juan, 2017; Dhaenens & Jourdan, 2016; Gambella, Ghaddar, & Naoum-Sawaya, 2020; Song et al., 2019; Sra, Nowozin, & Wright, 2012; Wagner & Affenzeller, 2005; Xue, Zhang, Browne, & Yao, 2015), the use of ML techniques for solving COPs with the focus on exact optimization methods (Bengio, Lodi, & Prouvost, 2021) as well as the integration of ML techniques into MHs (Song et al., 2019; Talbi, 2016; 2020). The latter has received significant attention in recent years. In this paper, the focus is on the integration of ML techniques into MHs. To the best of our knowledge, the literature is deprived of having a comprehensive yet technical review on how ML techniques can serve MHs and for which specific purpose. In the past few years, several review papers have been published on the ways of integrating ML techniques for a particular purpose (e.g., algorithm selection (Kerschke, Hoos, Neumann, & Trautmann, 2019; Kotthoff, 2014), parameter tuning (Aleti & Moser, 2016), etc.). Few papers provide a general review on the integration of ML techniques for different purposes within MHs (Song et al., 2019; Talbi, 2016; 2020).

Different from the literature, this paper provides a comprehensive and technical review on the integration of ML techniques into MHs for different purposes including *algorithm selection*, *fitness evaluation*, *initialization*, *evolution*, *parameter setting*, and *cooperation*. Among different optimization problems, the focus of this paper is on solving COPs. This paper is not only a pedagogical paper that describes the main concepts and preliminaries, but also a technical paper that classifies the literature's papers to identify the research gaps and provides a technical discussion on the advantages/limitations, requirements, and challenges of implementing each way of integration. Furthermore, a taxonomy is also proposed to provide a common terminology and classification. We believe that this paper is indispensable not only for non-experts in the field of MHs desiring to use ML techniques, but also for senior researchers that aim to provide a pedagogical lesson for junior students, particularly Ph.D. students in both Operational Research and Computer Science.

2. Background

2.1. Combinatorial optimization problems

Combinatorial optimization problems (COPs) are a class of optimization problems with discrete decision variables and a finite search space, although still too large for an exhaustive search to be a realistic option (Korte, Vygen, Korte, & Vygen, 2012). The formal representation of a COP is as follows:

$$\begin{array}{ll} \text{minimize} & c(x) \\ \text{subject to:} & g(x) \geq b \\ & x \geq 0, x \in f \end{array} \quad (1)$$

where the inequalities $g(x) \geq b$ and $x \geq 0$ are the constraints that specify a convex polytope over which the objective function $c(x)$ is to be minimized, and f is the finite set of feasible solutions x that satisfy the constraints. There are many real-life problems (e.g., vehicle routing problem, scheduling problem, etc.) that can be formulated as COPs. A large part of COPs belong to the NP-Hard class of optimization problems, which require exponential time to be solved to optimality (Talbi, 2009). Table A.1 in Appendix A provides a list of COPs that are referred to throughout this paper.

2.2. Meta-heuristics

Solving a large number of real-life COPs in an exact manner is intractable within a reasonable amount of computational time. Approximate algorithms are alternatives to solve these problems. Although approximate algorithms do not guarantee the optimality, their goal is to obtain solutions as close as possible to the optimal solution in a reasonable amount of computational time, at most polynomial (Talbi, 2009).

Approximate algorithms are categorized into *problem-dependent heuristics* and *meta-heuristics*. The former, as its name implies, describes a group of algorithms which are designed for and apply to particular optimization problems. However, MHs are more general algorithms applicable to a large variety of optimization problems, particularly COPs, if well tailored.

MHs can be classified in different ways. They can be *nature-inspired* or *non-nature inspired* (Talbi, 2009). Many MHs are inspired by natural phenomena. Evolutionary Algorithms (EAs) such as Genetic Algorithm (GA) (Holland et al., 1992), Memetic Algorithm (MA) (Moscato et al., 1989), and Differential Evolution (DE) (Storn & Price, 1997) are inspired by biology; Artificial Bee Colony (ABC) (Karaboga, 2005), Ant Colony Optimization (ACO) (Dorigo & Blum, 2005), and Particle Swarm Optimization (PSO) (Kennedy, 2006) are inspired by swarm intelligence. There are also MHs inspired by non-natural phenomena; Imperialist Competitive Algorithm (ICA) (Atashpaz-Gargari & Lucas, 2007) by society, Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983) by physics, and Harmony Search (HS) (Geem, Kim, & Loganathan, 2001) by musics. From another perspective, MHs can be *memoryless* or they may use *memory* during the search process (Talbi, 2009). Memoryless MHs (e.g., GA, SA, etc.) do not use the historical information dynamically during the search process. However, MHs with memory, such as Tabu Search (TS) (Glover & Laguna, 1998), memorize historical information during the search process, and this memory helps to avoid making repetitive decisions.

Furthermore, MHs can make *deterministic* or *stochastic* decisions during the search process to solve optimization problems (Talbi, 2009). MHs with deterministic rules (e.g., TS) always obtain the same final solution when starting from the same initial solution, while stochastic MHs (e.g., SA, GA) apply random rules to solve the problem and obtain different final solutions when starting from the same initial solution. Moreover, in terms of their starting point, MHs are divided into *single-solution based* or *population-based* MHs (Talbi, 2009). Single-solution based MHs, also known as trajectory methods, such as Iterated Local Search (ILS) (Lourenço, Martin, & Stützle, 2003), Breakout Local Search (BLS) (Benlic, Epitropakis, & Burke, 2017), Descent-based Local Search (DLS) (Zhou, Hao, & Duval, 2016), Guided Local Search (GLS) (Voudouris & Tsang, 1999), Variable Neighborhood Search (VNS) (Mladenović & Hansen, 1997), Hill Climbing (HC) (Johnson, Papadimitriou, & Yannakakis, 1988), Large Neighborhood Search (LNS) (Shaw, 1998), Great Deluge (GD) (Dueck, 1993), TS, SA, etc., manipulate and transform a single solution to reach the (near-) optimal solution. Population-based MHs such as Water Wave Optimization (WWO) (Zheng, 2015), GA, PSO, ACO, etc., try to find the optimal solution by evolving a population of solutions. Because of this nature, population-based MHs are more exploration search algorithms, and they allow a better diversification in the entire search space. Single-solution based MHs are more exploitation search algorithms, and they have the power to intensify the search in local regions.

Finally, depending on their search mechanism, MHs can be *iterative* or *greedy* (Talbi, 2009). The former (e.g., ILS, GA) starts with a complete solution and manipulates it at each iteration using a

set of search operators. The latter, also called constructive algorithm, starts from an empty solution and constructs the solution step by step until a complete solution is obtained. Classical examples of greedy algorithms are Nearest Neighbor (NN), Greedy Heuristic (GH), Greedy Randomized Heuristic (GRH), and Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995).

2.3. Machine learning

Machine learning (ML) is a sub-field of artificial intelligence that uses algorithmic and statistical approaches to give computers the ability to "learn" from data, i.e., to improve their performance in solving tasks without being explicitly programmed for each one (Bishop, 2006). These systems improve their learning over time autonomously, using extracted knowledge from data and information in the form of observations and real-world interactions. The acquired knowledge allows these systems to correctly generalize to new settings. According to (Bishop, 2006), ML algorithms can be classified into:

- **Supervised learning algorithms** – In supervised learning, the values of input variables and the corresponding values of the output variables (labels) are known a priori. A supervised learning algorithm attempts to automatically figure out the relationship between input variables and output labels and use it to predict the output for new input variables. Depending on the aim of learning, supervised learning algorithms can be classified into *classification* and *regression* algorithms. Classical supervised learning algorithms include Linear Regression (LR), Logistic Regression (LogR), Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), Naive Bayes (NB), Gradient Boosting (GB), Decision Tree (DT), Random Forest (RF), *k*-Nearest Neighbor (*k*-NN), Artificial Neural Network (ANN), etc.
- **Unsupervised learning algorithms** – They are used when the training data is neither classified nor labeled. In unsupervised learning, the values of input variables are known while there are no associated value for the output variables. The learning task is therefore to figure out and describe the patterns hidden in the input data. *Clustering* and *Association Rules* (ARs) are two tasks of unsupervised learning. Classical unsupervised learning algorithms include *k*-means clustering, Shared Nearest Neighbor Clustering (SNNC), Self-Organizing Map (SOM), Principal Component Analysis (PCA), Multiple Correspondence Analysis (MCA), and Apriori algorithms for ARs.
- **Reinforcement learning (RL) algorithms** – In these algorithms, an agent iteratively learns from interactions with its environment to take actions that would maximize the reward or minimize the risk. At each iteration, the agent automatically determines the ideal behavior (action) within a specific context to maximize its performance based on a reward feedback. RL algorithms include Q-Learning (QL), Learning Automata (LA), Opposition-based RL (OPRL), Monte Carlo RL, SARSA, Deep Reinforcement Learning (DRL), etc.

3. Taxonomy and review methodology

This section aims first at elaborating the major contributions that distinguish our paper from the literature. Next, we present a taxonomy to provide a common terminology and classification on the subject of this paper. Finally, the search methodology describing the procedure of searching and obtaining the relevant papers is presented.

3.1. Contributions

To the best of our knowledge, there is no comprehensive review paper on integrating ML techniques into MHs that investigates the integration from a technical point of view. Jourdan, Dhaenens, and Talbi (2006) provided a short survey on how ML techniques can help MHs with no detailed discussion on how such integration occurs. Another survey has been done by Zhang et al. (2011) on how ML techniques can improve the performance of evolutionary computation algorithms. Corne, Dhaenens, and Jourdan (2012) investigated the synergy between operations research and data mining with a focus on multi-objective approaches. With the rapid advances in the use of new ML techniques in MHs for even new purposes, as well as the increasing trend in the number of annually published papers in the area, there is a need to update the outdated review papers. In this regard, Talbi (2016) studied different ways of hybridization between different MHs as well as hybridizing MHs with mathematical programming, constraint programming, and ML. Although the author provides a good overview on how to hybridize MHs, it is less focused on the integration of ML into MHs. Calvet et al. (2017) reviewed the integration of ML and MH for solving optimization problems with dynamic inputs. The authors enumerate different ways of integrating ML into MH and vice versa; however, their work lacks a technical discussion on the requirements, challenges, and future works of each way of integration.

More recently, more general and comprehensive studies have been done by Song et al. (2019) and Talbi (2020) on integrating ML and MH. Song et al. (2019) studied the integration of ML and optimization in general and not particularly MHs. The authors review all four optimization-in-ML, ML-in-optimization, ML-in-ML, and optimization-in-optimization ways of integration. However, Song et al. (2019) provided less details on the integration of ML in MHs compared to Talbi (2020). Merely providing general research directions, their work lacks a comprehensive discussion on the research gaps and future research directions for the ML-in-optimization way of integration. Talbi (2020) provided a more complete and unified taxonomy on the integration of ML into MHs. The author identifies the integration of ML in MHs in three levels: 1) problem level integration, where ML is used, for example, to decompose the solution space or to reformulate the objectives and constraints of an optimization problem, 2) high-level integration between MHs, where ML techniques are used to make a link between different MHs, and 3) low-level integration in a MH, where ML techniques are used in the components of MHs (e.g., initialization, operator selection, population management, etc.). Although the work by Talbi (2020) is a good comprehensive and pedagogical review paper explaining different general ways (i.e., levels) that ML techniques can be integrated into MHs, it does not go into the details on the requirements, challenges, and possible future research directions on the use of ML techniques in each level of integration.

To the best of our knowledge, the literature is deprived of having a comprehensive and technical review on how ML techniques can be integrated into MHs and for which specific purposes. Different from the literature, this paper provides a review on the use of ML techniques in the design of different elements of MHs for different purposes including *algorithm selection*, *fitness evaluation*, *initialization*, *evolution*, *parameter setting*, and *cooperation*. In a pedagogical way, we describe the key concepts and preliminaries of each way of integration. In addition, this paper reviews the recent advances on each topic and classifies the literature's papers to identify the research gaps. We also propose a taxonomy to provide a common terminology and classification. As an important part, we then provide a technical discussion on the advantages/limitations, requirements, and challenges of implementing each way of integration. Finally, promising future research directions are identified

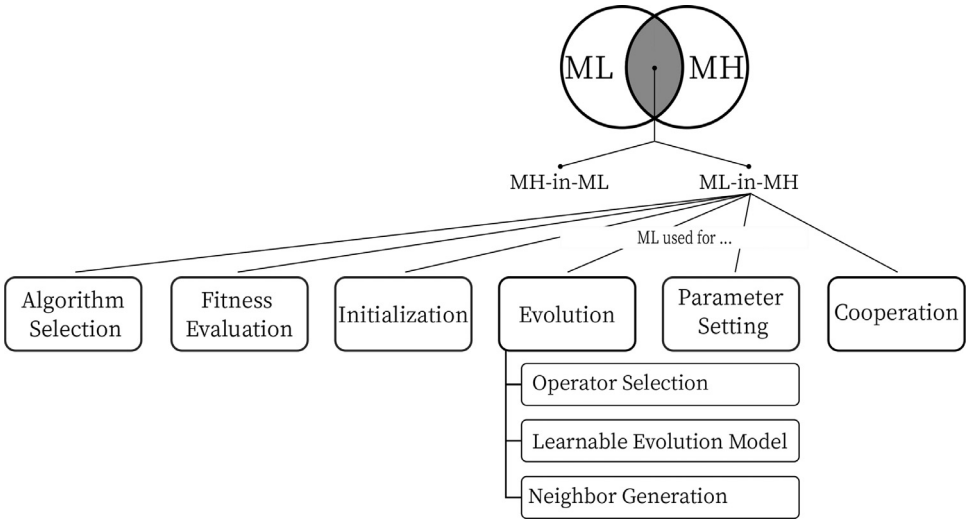

Fig. 1. Taxonomy on the use of ML in MHs (ML-in-MH).

depending on the way of integrating ML into MHs. We believe that this paper is indispensable not only for non-experts in the field of MHs desiring to use ML techniques, but also for senior researchers that aim to provide a pedagogical lesson for junior students, particularly Ph.D. students in both Operations Research and Computer Science.

3.2. Taxonomy

Although MHs and ML techniques have been initially developed for different purposes, they may perform common tasks such as feature selection or solving optimization problems. MHs and ML techniques frequently interact to improve their search and/or learning abilities. MHs have been widely employed in ML tasks (MH-in-ML) for decades (Wagner & Affenzeller, 2005; Xue et al., 2015). For Instance, MHs can be used for feature selection, parameter setting of ML techniques (Oliveira, Braga, Lima, & Cornélio, 2010), or pattern recognition (Kiranyaz, Ince, & Gabbouj, 2014). ML techniques are being extensively integrated into MHs (ML-in-MH) to make the search process intelligent and more autonomous. For instance, RL can help to select the most efficient operators of MHs during the search process (dos Santos, de Melo, Neto, & Aloise, 2014).

The purpose of this paper is to review the studies wherein ML techniques have been integrated into MHs for solving COPs. We propose a taxonomy on integrating ML techniques into MHs, ML-in-MH branch of Fig. 1. Discovering the MH-in-ML branch of Fig. 1 is out of the scope of this paper, and we refer interested readers to the latest literature reviews on the use of MHs in ML tasks and the references cited therein (Calvet et al., 2017; Gambella et al., 2020; Song et al., 2019).

We propose to classify different types of integration according to the taxonomy presented on Fig. 1. According to this classification, ML techniques can be integrated into MHs for the following purposes:

- *Algorithm selection* – When solving an optimization problem with MHs, the first decision is to select one or a set of MHs for solving the problem. ML techniques can predict the performance of MHs in solving optimization problems.
- *Fitness evaluation* – The success of any MH in achieving a specific goal (objective) is evaluated by fitness evaluation of the solutions during the search process. ML techniques can speed up the search process by approximating computationally expensive fitness functions.
- *Initialization* – Any MH starts its search process from an initial solution or a population of solutions. ML techniques can help to generate good initial solutions by using the knowledge of good solutions on similar instances or speeding up the initialization by decomposing the input data space into smaller sub-spaces.
- *Evolution* – It represents the entire search process starting from the initial solution (population) toward the final solution (population). ML techniques can intelligently select the search operators (i.e., *Operator selection*), evolve a population of solutions using the knowledge of good and bad solutions during the search (i.e., *Learnable evolution model*), and to guide the neighbor generation process using the knowledge obtained during the search process (i.e., *Neighbor generation*).
- *Parameter setting* – Any MH, depending on its nature, has a set of parameters which need to be set before the search process starts. ML techniques can help to set or control the values of the parameters before or during the search process.
- *Cooperation* – Several MHs can cooperate with each other to solve optimization problems in parallel or sequentially. ML techniques can improve the performance of cooperative MHs by adjusting their behavior during the search process.

Each type of integration in Fig. 1 can be also classified from another viewpoint into: *problem-level*, *high-level*, and *low-level* integration (Talbi, 2020). *Algorithm selection* and *fitness evaluation* represent a high-level and problem-level integration of ML into MHs, respectively. Depending on the strategy to generate initial solutions (see Section 6), *initialization* belongs to either problem-level or low-level integration. *Evolution* and *parameter setting* fall in the category of low-level integration. Finally, *cooperation* may belong to either high-level or low-level integration depending on the level of cooperation.

3.3. Search methodology

To conduct the literature review, first, well-known scientific databases including Scopus, Google Scholar, IEEE Explore, Science Direct, Springer, ACM Digital Library, and Emerald have been carefully searched to find the relevant papers in both scientific journals and international conferences. To do that, we have identified a set of particular keywords for each type of integration. Then, the search process is conducted using the following search rule:

{`keyword1` **`AND`** `keyword2` **`AND`** `keyword3` **`AND`** `keyword4`}

`keyword1` is an element of a set of keywords related to the integration types of ML techniques into MHs, as explained in

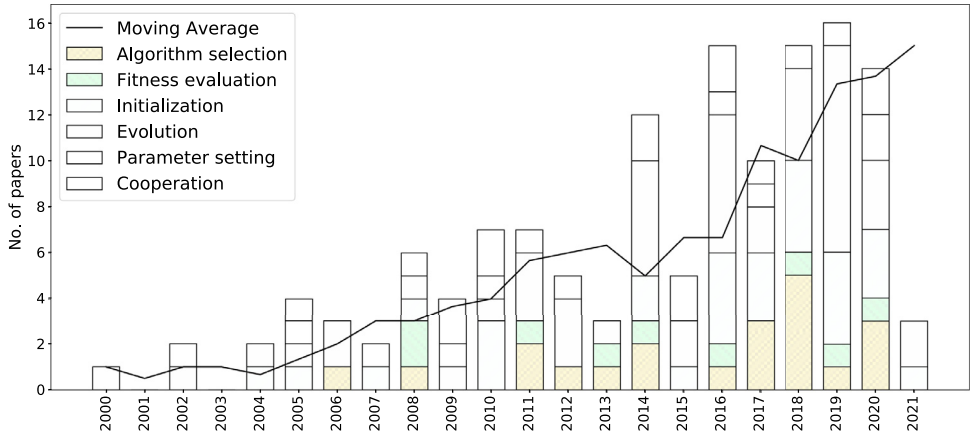

Fig. 2. Number of papers per year and per each type of integration of ML techniques in MHs for solving COPs.

Section 3.2 and Fig. 1. More precisely, `keyword1` belongs to the union of the sets {algorithm selection, algorithm recommendation, autonomous algorithm selection, performance prediction, meta-learning, meta-feature} (for the algorithm selection), {fitness approximation, surrogate model, meta-model, fitness reduction} (for the fitness evaluation), {initialization, initial solution generation} (for the initialization), {adaptive operator selection, autonomous operator selection, learnable evolution model, non-Darwinian evolution, pattern extraction, rule extraction, rule injection} (for the evolution), {parameter setting, parameter tuning, parameter control} (for the parameter setting), and {cooperation, cooperative MHs, parallel MHs, Hybrid MHs, sequential MHs} (for the cooperation of algorithms). `keyword2` stands for the ML techniques extracted from Section 2.3. `keyword3` accounts for different MHs, ranging from single-solution to population-based MHs, extracted from Section 2.2. Finally, `keyword4` is dedicated to the COP under study. We provide a complete list of COPs in Table A.1 in Appendix A.

After filtering the obtained papers, a total number of 136 papers are kept, which are relevant to the scope of this paper. All these papers are reviewed and classified in details in this paper. Fig. 2 shows the number of reviewed papers per year (from 2000 to early 2021) and for each type of integration illustrated in Fig. 1. Looking at the whole number of papers regardless of which type of integration they belong to, Fig. 2 shows a significant increase in the number of papers integrating ML techniques into MHs for different purposes throughout the last two decades, which illustrates a meaningful growth in the knowledge and popularity of the topic. Among all types of integration, studies on *evolution* contribute the most to the total number of papers over time, and an increasing trend can be seen for the last decade. *Algorithm selection* and *initialization* have been at the second place of attention, and they have gained significant attention throughout the last two decades. *Cooperation*, *parameter setting*, and *fitness evaluation* are also the types of integration with semi-constant trend of attention. In summary, Fig. 2 illustrates that *algorithm selection*, *evolution*, and *initialization* are being studied attentively, and *fitness evaluation*, *parameter setting*, and *cooperation* are less-studied directions and they are worthy to be explored more in the future.

The rest of this paper is structured as follows. Each section explains in details each way of integrating ML techniques into MHs and starts with an introduction to the corresponding type of integration. Then, relevant papers are reviewed, classified, and analyzed. Finally, the section ends with a comprehensive discussion on the corresponding guidelines, requirements, challenges, and future research directions. To be more precise, Sections 4–9 are respectively dedicated to algorithm selection, fitness evaluation, initialization, evolution, parameter setting, and cooperation. Finally, conclusions and perspectives are given in Section 10.

4. Algorithm selection

There are many studies in the literature developing high-performance MHs for well-known COPs. However, there is no single MH that dominates all other MHs in solving all problem instances. Instead, different MHs perform well on different problem instances (i.e., performance complementarity phenomena) (Kerschke et al., 2019). Therefore, there is always an unsolved question as "Which algorithm is likely to perform best for a given COP?" (Rice et al., 1976). The ideal way to find the best algorithm to solve a COP, when the computational resources are unlimited, is to exhaustively run all available algorithms and choose the best solution, no matter by which algorithm it has been obtained. However, because of the limited computational resources, it is practically impossible to test all available algorithms on a particular problem instance. In this situation, a major question arises as "Among the existing algorithms, how to select the most appropriate one for solving a particular problem instance?". ML techniques help to answer this question by selecting the most appropriate algorithm(s). This is where the Algorithm Selection Problem (ASP) steps in.

ASP aims at automatically selecting the most appropriate algorithm(s) for solving a problem instance using ML techniques (Kerschke et al., 2019; Kotthoff, 2014). The original framework of ASP was developed by Rice et al. (1976) based on four principal components: 1) the problem space, including a set of problem instances, 2) the feature space, including a set of quantitative characteristics of the problem instances, 3) the algorithm space, including a set of all available algorithms for solving the problem instances, and 4) the performance space that maps each algorithm from the algorithm space to a set of performance metrics such as the Objective Function Value (OFV), CPU Time (CT), etc. The final goal is to find the problem-algorithm mapping with the highest performance.

To find the best problem-algorithm mapping, ASP employs Meta-learning, a sub-field of ML, that learns the problem-algorithm mapping on a set of training instances and creates a meta-model. The meta-model is then used to predict the appropriate mapping for new problem instances (Kotthoff, 2014). In solving COPs, studying ASP has enabled researchers to take advantage of various MHs by systematically selecting the most appropriate algorithm(s) among the existing ones, and has resulted in significant performance improvements (Kotthoff, 2016). In the literature, ASP has been referred to as *algorithm selection* (Kanda, de Carvalho,

Table 1
Classification of papers studying ASP.

Ref.	Prob. space	Alg. space	Portfolio	Perf. space	Learning	Task	ML tech.	Size
Hutter, Hamadi, Hoos, and Leyton-Brown (2006)	SAT	ILS	Static	CT	Offline	Reg.	LR	30,000
Smith-Miles (2008)	QAP	ILS, TS, ACO	Static	OFV	Offline	SLC	ANN	644
Kanda et al. (2011a)	TSP	TS, GRASP, SA, GA	Static	OFV	Offline	MLC	*k*-NN, ANN, NB	2500
Kanda, de Carvalho, Hruschka, and Soares (2011b)	TSP	TS, GRASP, SA, GA	Static	OFV	Offline	LRC	ANN	2000
Kanda, Soares, Hruschka, and De Carvalho (2012)	TSP	TS, SA, GA, ACO	Static	OFV	Offline	LRC	ANN	300
Pitzer, Beham, and Affenzeller (2013)	QAP	TS, VNS	Static	OFV	Offline	SLC	LR, SVM, AR	137
Messelis and De Causmaecker (2014)	PSP	TS, GA	Static	OFV	Offline	SLC	DT	3140
Smith-Miles et al. (2014)	GCP	HC, TS, ACO	Static	OFV	Offline	MLC	SVM, NB	675
Kanda et al. (2016)	TSP	TS, SA, GA, ACO	Static	OFV	Offline	LRC	ANN, *k*-NN, DT	600
Beham et al. (2017)	QAP	TS, VNS, GA, MA	Static	OFV, CT	Offline	SLC	*k*-NN	94
de León, Lalla-Ruiz, Melián-Batista, and Moreno-Vega (2017a)	BAP	LNS	Static	OFV	Offline	LRC	*k*-NN	720
de León, Lalla-Ruiz, Melián-Batista, and Moreno-Vega (2017b)	TSP, VRP	GRASP, SA, LNS	Static	OFV	Offline	LRC	*k*-NN	130
Miranda et al. (2018)	MAX-SAT	GA, PSO	Static	OFV	Offline	SLC	ANN, SVM, DT	555
Pavelski, Kessaci, and Delgado (2018b)	FSP	HC, SA, TS, ILS	Static	OFV	Offline	MLC	GB	27,000
Pavelski, Delgado, and Kessaci (2018a)	FSP	HC, SA, TS, ILS	Static	OFV	Offline	SLC	DT	12,000
Dantas and Pozo (2018)	QAP	BLS, ACO, TS	Static	OFV	Offline	SLC	RF	135
Degroote, González-Velarde, and De Causmaecker (2018)	AP	TS	Static	OFV	Offline	SLC	RF	286
Gutierrez-Rodríguez, Conant-Pablos, Ortiz-Bayliss, and Terashima-Marín (2019)	VRP	EA, GA, PSO	Static	OFV	Offline	SLC	ANN	56
Dantas and Pozo (2020)	QAP	BLS, ACO, MA	Static	OFV	Offline	MLC	RF	5000
Wawrzyniak et al. (2020)	BAP	HC, GRASP, ILS	Static	OFV, CT	Offline	SLC	*k*-NN	2100
Sadeg, Hamdad, Kada, Benatchba, and Habbas (2020)	MAX-SAT	GA, GRASP	Static	OFV, CT	Offline	SLC	*k*-NN, RF, ANN	1534
de la Rosa-Rivera, Nunez-Varela, Ortiz-Bayliss, and Terashima-Marín (2021)	TTP	ILS, SA, VNS	Static	OFV	Offline	Reg.	LR	6000

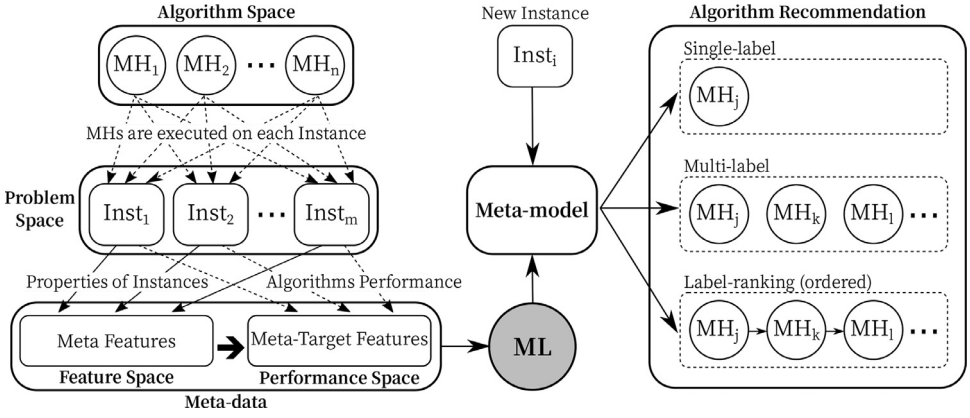

Fig. 3. Procedure of ASP.

Hruschka, Soares, & Brazdil, 2016), *per-instance algorithm selection* (Kerschke et al., 2019), *algorithm recommendation model* (Chu et al., 2019), and *automated algorithm selection* (Dantas & Pozo, 2018). However, they all share the same goal of automatically selecting the most appropriate algorithm(s) for a particular problem instance.

We illustrate the procedure of ASP in Fig. 3. As it can be seen in Fig. 3, ASP involves two main steps: 1) meta-data extraction, and 2) meta-learning and meta-model creation.

- **Meta-data extraction** – Given the problem and algorithm spaces, the goal is to determine the feature and performance spaces called meta-data. Meta-data is classified into two categories: meta-features and meta-target features (Kanda et al., 2016). Meta-features are a set of quantitative features that represent the properties of a problem instance, while meta-target features are a set of performance data that describe the performance of each algorithm on a particular problem instance. Considering the importance of defining appropriate meta-features, there are several works in the literature which aim to identify good meta-features for different COPs, including SAT (Kerschke et al., 2019), TSP (Kerschke et al., 2019; Mersmann et al., 2013; Smith-Miles & Lopes, 2012), AP (Angel & Zissimopoulos, 2002; Smith-Miles & Lopes, 2012), OP (Bossek, Grimme, Meisel, Rudolph, & Trautmann, 2018), KP (Smith-Miles & Lopes, 2012), BPP (Smith-Miles & Lopes, 2012), and GCP (Smith-Miles & Lopes, 2012).
- **Meta-learning and meta-model creation** – Using the meta-data, a meta-model is created which can predict the performance of each algorithm for each problem instance and determine the problem-algorithm mapping. Depending on the type of prediction expected from the meta-model, different ML techniques can be used for meta-learning and creating the meta-

model. Different types of prediction include selecting the best algorithm (Smith-Miles, 2008), selecting a set of most appropriate algorithms (Kanda, Carvalho, Hruschka, & Soares, 2011a), and ranking a set of appropriate algorithms (Kanda et al., 2016) for solving a problem instance. Depending on the type of prediction, the task of meta-learning could be *Single-label Classification* (SLC), *Multi-label Classification* (MLC), and *Label-ranking Classification* (LRC), respectively. Besides classification techniques, regression techniques such as LR can also be used to predict the performance of each algorithm. Using regression techniques, the meta-learning problem is a multiple regression problem wherein one target variable is considered for each algorithm.

Considering the way to create the meta-model, ASP can be either *online* or *offline*. In offline ASP, the meta-model is constructed using a particular set of training instances with the aim to predict the problem-algorithm mapping for new problem instances (Kanda et al., 2016; Miranda, Fabris, Nascimento, Freitas, & Oliveira, 2018; Smith-Miles, 2008). However, in online ASP, the meta-model is constructed and employed dynamically while solving a set of problem instances (Armstrong, Christen, McCreath, & Rendell, 2006; Degroote, Bischl, Kotthoff, & De Causmaecker, 2016; Gagliolo & Schmidhuber, 2010; Kerschke et al., 2019). Furthermore, the algorithm space, commonly known as algorithm portfolio, is classified as *static* or *dynamic*. Static portfolios contain a set of fixed algorithms which are included into the portfolio before solving a problem instance and the composition of the portfolio along with the algorithms within the portfolio do not change during solving an instance, while dynamic portfolios contain a set of algorithms whose composition and configuration may change while solving a problem instance (Kotthoff, 2014).

In the rest of this section, the research papers studying ASP for COPs are reviewed and classified, followed by a detailed discussion on the corresponding guidelines, requirements, challenges, and future research directions.

4.1. Literature classification & analysis

Inspired from Fig. 3, Table 1 classifies the papers studying ASP for COPs based on different characteristics such as *problem space*, *algorithm space*, algorithm *portfolio* type, *performance space*, *learning* mechanism, meta-learning *task*, the employed *ML technique*, and the *size* of the training set. To the best of our knowledge, Table 1 lists all relevant papers, including the most recent papers in the literature that study ASP for selecting MHs to solve COPs using ML techniques. Other papers which miss at least one of these three main components (i.e., MHs, ML techniques, or COPs) are out of the scope of this paper and are not reviewed in this paper.

Regarding the problem space, TSP and QAP are the most studied problems compared to the other COPs. Also, Table 1 shows that a majority of the studied COPs have a common characteristic; they either have permutation-based representation (e.g., TSP, VRP, FSP) or discrete value based representation (e.g., AP, QAP). One of the major reasons for such an observation is the availability of various powerful MHs for these representations, as well as the simplicity of manipulating these types of representations (Abdel-Basset, Manogaran, Rashad, & Zaied, 2018; Arora & Agarwal, 2016; Koç, Bektaş, Jabali, & Laporte, 2016).

Considering the algorithm space, Table 1 reveals that the algorithm portfolio in most of the studies is composed of MHs with different mechanisms, varying from single-solution to population-based, from memory-less MHs (e.g., ILS) to MHs with memory (e.g., TS), from MHs with accepting only better solutions (e.g., ILS) to MHs with accepting worse solutions (e.g., SA), and from MHs with fixed neighborhood size (e.g., TS and SA) to MHs with variable neighborhood size (e.g., VNS). The utilization of such different algorithms in a portfolio highlights the fact that different MHs with different search mechanisms perform differently for different instances of COPs (dos Santos et al., 2014).

Table 1 also shows that all reviewed papers use a static portfolio. Since the algorithms and their configurations do not change in static portfolios, their selection becomes more crucial for the overall success of the resolution process. An efficient way to construct the portfolio is to involve algorithms that complement each other such that good performance can be achieved on a wide range of different problem instances. There has been a debate on the composition and characteristics of the algorithms within the portfolio among the reviewed papers. The first and the most straightforward manner to construct the portfolio is to randomly select algorithms from a large pool of diverse MHs. The second manner is to incorporate MHs with the best overall performance in the portfolio. However, the third and the most promising manner is to construct a portfolio with algorithms of complementary strengths. An ASP with a portfolio composed of MHs with complementary strengths logically seems to be more efficient comparing to an ASP with a portfolio composed of MHs with the best overall performance. However, most research papers construct the portfolio less explicitly using the MHs that have performed well in the literature when solving particular instances of the COP at hand, regardless of their strengths and weaknesses when facing new problem instances.

Considering the *performance space*, we can see that most of the papers evaluate the performance of an algorithm based on the OFV of the obtained solutions. Although considering the quality of solutions in terms of their OFV is the most common criterion to compare algorithms, there are other criteria that play an important role when selecting an algorithm, among which the CT and robustness have a high importance, especially for solving COPs (Choong, Wong, & Lim, 2019; Mosadegh, Ghomi, & Süer, 2020; dos Santos et al., 2014). Therefore, OFV, CT, and robustness are the three most important criteria by which the algorithms could be compared. Although there is a trade-off between these measures, and usually no algorithm performs best in all criteria, taking into account these criteria provides more efficient algorithm selection when solving COPs (Beham, Affenzeller, & Wagner, 2017; Wawrzyniak, Drozdowski, & Sanlaville, 2020). The multiple criteria ASP can be modeled through a multi-objective perspective (Kerschke et al., 2019).

Table 1 shows that all reviewed papers have created the meta-model in an offline manner and none of them studies the online ASP for solving COPs. A big disadvantage of ASP in an offline manner is that in this way, the performance of the selected algorithms is not monitored to confirm whether they satisfy the expectations that led them being selected or not. Accordingly, offline ASP is inherently vulnerable to bad choices of MHs; however, the advantage of an offline ASP is its lower computational effort since the meta-model is created once based on a set of training instances. On the contrary, the major advantage of an online ASP is the more justified decisions that can be made during the algorithm selection process, which also reduces the negative impact of a bad choice. However, adding such flexibility imposes an extra effort, as the meta-model is created and employed dynamically while solving a set of problem instances and thus decisions on algorithm selection need to be made more frequently throughout the resolution of the new problem instances. Broadly speaking, there is no evidence to show the superiority of one method over the other, and both methods have led to performance improvements (Kanda et al., 2016; Wawrzyniak et al., 2020). Hence, the choice of whether to create the meta-model in an offline or an online manner depends highly on the specific application.

Another studied characteristic in Table 1 is the meta-learning task. The most common output of ASP is a single best algorithm

from the portfolio and using it to solve the problem instance (i.e., the result of SLC task). A disadvantage of selecting a single best algorithm is having no way of compensating a wrong selection. Indeed, if a single algorithm is selected and shows unsatisfying performance on a new problem instance, there are no other recommended algorithms to replace such an inefficient algorithm. An alternative approach is selecting multiple algorithms (i.e., the result of MLC and RLC tasks). However, there is no report to show that one of these approaches is superior to another.

4.2. Discussion & future research directions

In this section, first, a guideline is provided for researchers on when to study ASP and which requirements to meet to study ASP. Second, a set of technical challenges of studying ASP are discussed. Finally, several future research directions are provided based on the research gaps extracted from Table 1.

4.2.1. Guideline & requirements

The aim of providing a guideline for studying ASP is to help researchers to understand that although studying ASP may provide the most appropriate MH(s) to solve COPs, it is not always the best choice. In the following, we first describe the situations where ASP is useful; then, the requirements of using ASP are elaborated.

Studying ASP is useful when the computational resources (i.e., available time and the number of available cores) for solving a problem instance are limited. This is the case for optimization problems at an operational level where limited time is available, and the problems should be solved more frequently. On the other hand, for the optimization problems at the strategic level (e.g., FLP), where there is enough time, the best choice is to execute all algorithms and select the most appropriate one, since in the strategic level, finding better solutions outweighs the computational cost of executing all algorithms. Furthermore, another moment when studying ASP becomes indispensable is when there are several efficient competitive algorithms for the problem at hand and none of them could be definitely selected for solving the problem instance. In addition, ASP can help non-experts to select appropriate algorithm(s) for solving optimization problems. In other words, ASP can be replaced by the traditional trial-and-error optimization tasks, especially when the number of candidate algorithms is large and little prior knowledge of the problem is available.

Once the use of ASP is justified, a set of requirements should be fulfilled before applying ASP. The first requirement for using ASP is *affordability* of the selection procedure in terms of computational resources. In fact, if studying ASP for a problem instance is more expensive than solving the problem instance with all algorithms and selecting the best one, there is no need at all to study ASP. The next important requirement that could be also a challenge for ASP is *data availability*. When creating the meta-model, it is necessary to provide a pool of sufficient training instances that well represent new instances. It should be noted that having a pool of sufficient instances does not guarantee the efficiency of ASP, and *instance dissimilarity* and *algorithmic discrimination* are two other requirements that need to be satisfied (Smith-Miles, Baatar, Wreford, & Lewis, 2014). The former denotes the necessity of providing instances which are as diverse as possible and spread out over different regions. The latter denotes the necessity to provide instances that show different behaviors while being solved by different algorithms in the portfolio. Indeed, some instances should be easy for some algorithms and hard for others. *Algorithmic discrimination* requirement helps to learn the strengths and weaknesses of different algorithms when solving different instances with different characteristics.

4.2.2. Challenges & future research directions

Despite the effectiveness of ASP in solving COPs, the implementation of an algorithm selection procedure is not always straightforward, and researchers may face several challenges throughout the ASP procedure, from the design to its implementation.

The first challenge to deal with is called *data generation* challenge. As mentioned earlier, two important requirements of ASP are *instance dissimilarity* and *algorithmic discrimination* that allow generating a rich data set of instances with different characteristics that leads to a more efficient meta-model creation. To generate such a rich data set, the algorithms need to be provided with a wide range of instances. This challenge is twofold: 1) Finding or generating a set of sufficient instances that ensure the *data availability* requirements, particularly *instance dissimilarity* and *algorithmic discrimination*, is a complicated task and 2) Executing all candidate algorithms on the generated instances might be very time consuming if the number of instances is large. This makes the meta-model creation computationally expensive. This first challenge becomes more and more complicated if little knowledge is available for the COPs at hand. On the other hand, it would be less challenging to generate a set of sufficient instances that fulfils the mentioned requirements for classical COPs for which there exists several instance libraries (e.g., TSPLIB for TSP (Reinelt, 1991) and Taillard for FSP (Taillard, 1993)).

Apart from the *data generation* challenge, the second challenge is *instance characterization*. A major issue in creating a meta-model is the characterization of the problem instances through a set of appropriate measures, called meta-features (Smith-Miles & Lopes, 2012). Meta-features must reveal instance properties that affect the performance of the algorithm. More informative and appropriate features lead to a better mapping between the meta-features and algorithm performance and consequently a high-quality meta-model. The *instance characterization* challenge has two aspects; first, the type of meta-features to extract and, second, the computational time associated with the meta-feature extraction. The type of meta-features varies from the most basic ones such as descriptive statistics (e.g., minimum, maximum, mean, and median of input parameters) to more complex ones such as landscape features of COPs. Taking TSP as an example, the basic meta-features include “Edge and vertex measures” such as *number of vertices*, *the lowest/highest vertex cost*, and *the lowest/highest edge cost*, and the more complex meta-features could be “complex network measures” such as *average geodesic distance*, *network vulnerability*, and *target entropy* (Kanda et al., 2016). Depending on the type of meta-features, feature extraction can be a computationally cheap task for basic features and expensive for more complex ones. Therefore, in selecting the meta-features, one should consider both the level of information they provide and their corresponding computational time. The optimal way is to select a set of features that are as informative as possible while computationally affordable. To put the issue into perspective, creating a high-quality meta-model is a complicated interplay between using a set of diverse training instances with different behavior over different algorithms and using a subset of informative meta-features whose extraction is computationally cheap.

There is always an unanswered question on the trade-off between the performance of the algorithm selection and its complexity, particularly on the extraction of meta-features. An important direction for future research is moving from problem-specific features toward more general and simple features which are computationally cheaper to be extracted when studying ASP. There is evidence that shows for particular optimization problems, a small number of simple meta-features suffice for achieving excellent performance of ASP (Hoos, Peitl, Slivovsky, & Szeider, 2018).

In the reviewed studies, the focus has been on heterogeneous portfolios composed of different MHs with different characteristics,

while different configurations of a single MH in a homogeneous portfolio also show different behavior in solving a COP. Another intriguing future research direction could be studying ASP for COPs with a homogeneous portfolio, to select a particular configuration of a single MH among a set of particular configurations, which is even less challenging compared to dealing with a portfolio of different algorithms.

As explained in Section 4.2.1, *instance dissimilarity* and *algorithmic discrimination* are two requirements and also two challenges of ASP. An interesting research direction could be the idea of evolving instances using EAs (Bossek & Trautmann, 2016; van Hemert, 2006; Mersmann et al., 2013; Smith-Miles, van Hemert, & Lim, 2010). Indeed, the idea is to use EAs to evolve instances of COPs as distinct as possible to ensure instance dissimilarity. In this way, a set of diverse instances are obtained, improving the performance of the meta-model.

Developing an online ASP is another promising future research direction. As shown in Section 4.1, all papers have studied offline ASP, where the meta-model is created using a set of training instances. However, it might be possible to get even better results using online ASP, which adapts the algorithm selection mechanism while solving a set of problem instances. Although the online ASP imposes an extra computational overhead, it increases the robustness, as adjustments (if required) can be applied to the algorithm selection mechanism during the resolution of COP instances. It is worth mentioning that the extra overhead can be alleviated by using computationally cheap meta-features. Another way to cope with the extra overhead is using incremental or online active learning techniques where an already trained model is used and improved incrementally during the search process (Lughofer, 2017).

When the output of ASP is multiple selected algorithms, one can study how multiple selected algorithms are *scheduled* to solve a problem instance. The key idea of scheduling is to execute a sequence of algorithms from a given set, one after another, each for a given (maximum) time (Kotthoff, 2014). Most of the research papers in Table 1 proposing multiple algorithms have not studied the algorithm schedule. However, more flexibility is obtained throughout the search process when an algorithm with particular strength (i.e., exploration and exploitation) is employed whenever needed. Accordingly, one future research direction could be scheduling multiple recommended algorithms to solve a problem instance. The schedule of algorithms can be either *static* or *dynamic* (Kadioglu, Malitsky, Sabharwal, Samulowitz, & Sellmann, 2011). In a *static* algorithm schedule, algorithms are executed based on a given order. In a *dynamic* schedule, the sequence of algorithms may change based on their historical performance, and certain algorithms may not be used at all.

Another interesting future research direction could be taking into account multiple performance criteria by which the algorithms are evaluated when studying ASP. The multiple criteria ASP can be modeled through a multi-objective perspective (Kerschke et al., 2019). Last but not least, Table 1 reveals that the focus has been mostly on COPs with permutation-based and discrete value based representations. As a future research direction, ASP can be extended to other COPs such as different types of scheduling problems for which a lot of efficient MHs have been developed in recent years (Allahverdi, 2015).

5. Fitness evaluation

Fitness evaluation is one of the key components of MHs to guide the search process towards the promising regions of the search space. For some optimization problems, there is no analytical fitness function by which the solutions are evaluated, or even if it exists, it is computationally expensive to evaluate. ML techniques can be integrated into MHs to reduce the computational effort for solving such optimization problems either through *fitness approximation* (Díaz-Manríquez, Toscano, & Coello, 2017; Jin, 2005; 2011) or *fitness reduction* (Saxena, Duro, Tiwari, Deb, & Zhang, 2012). Fitness approximation is categorized into *functional approximation* and *evolutionary approximation* (Jin, 2005):

- **Functional approximation** – It is used when evaluating the solutions using the original fitness function is computationally expensive. In this condition, the computationally expensive fitness function is replaced with an approximate model that imitates the behavior of the original fitness function as closely as possible, while being computationally cheaper to evaluate. These approximate models are built using ML techniques such as polynomial regression (Singh, Ray, & Smith, 2010), RF (Zhou, Ong, Nguyen, & Lim, 2005), ANN (Jin & Sendhoff, 2004; Park & Kim, 2017), SVM (González-Juarez & Andrés-Pérez, 2019; Loshchilov, Schoenauer, & Sebag, 2010), Radial Basis Functions (RBFs) (Qasem, Shamsuddin, Hashim, Darus, & Al-Shammari, 2013), and Gaussian process models also referred to as Kriging (Knowles, 2006). These ML techniques are trained using a set of training data, wherein the input variable is a set of features extracted from the solution instances and the output variable is the original fitness value of each solution instance. The aim is then to approximate the fitness value of the new generated solutions. The approximate model can be created either *offline* or *online*. A particular use of functional approximation in MHs is known as surrogate-assisted MHs (Jin, 2011), wherein the approximate (surrogate) model is iteratively refined (i.e., online refining) during the search process. The first surrogate-assisted MHs were developed for continuous optimization problems, and there are numerous efficient surrogate modeling techniques for continuous functions (Pelamatti, Brevault, Balesdent, Talbi, & Guerin, 2020). In recent years, they have also gained attraction for discrete optimization problems (Bartz-Beielstein & Zaefferer, 2017). Surrogate models are categorized into single, multi-fidelity, and ensemble surrogate models (Bartz-Beielstein & Zaefferer, 2017).
- **Evolutionary approximation** – It is specifically developed to deal with EAs. Instead of approximating the fitness function, the evolutionary approximation aims at reducing the computational effort by approximating the elements of the EAs. There are two main sub-categories of evolutionary approximation:
  - *Fitness inheritance* in which the fitness value of an individual is calculated based on the fitness values of its parents. For instance, the fitness value of an offspring can be the (weighted) average of the fitness values of its parents.
  - *Fitness imitation* in which the fitness value of an individual is calculated using the fitness value of its siblings. Using ML techniques such as clustering techniques, the population is divided into several clusters, and only the representative individuals of each cluster are evaluated. Afterward, the fitness values of other individuals are calculated based on their corresponding representatives in the clusters. Clustering techniques have been widely used for fitness imitation in the literature (Xiang, Tian, Xiao, & Zhang, 2020; Yu, Tan, Sun, & Zeng, 2017).

Apart from fitness approximation, *fitness reduction* is another approach for dealing with computationally expensive fitness functions in multi-objective optimization problems (Saxena et al., 2012). Instead of approximating the fitness function, fitness reduction aims at reducing the number of fitness functions using ML techniques such as PCA (Saxena et al., 2012) and Feature Selection (FS) techniques (López Jaimes, Coello Coello, & Chakraborty, 2008) as well as reducing the number of fitness function evalua-

Table 2
Classification of papers studying fitness evaluation.

Ref.	Fitness evaluation	Single/ Multi obj.	ML tech.	MH	COP
Pathak, Srivastava, and Srivastava (2008)	Functional approximation	Multi	ANN	GA	PSP
López Jaimes et al. (2008)	Fitness reduction	Multi	FS	GA	KP
Moraglio, Kim, and Yoon (2011)	Functional approximation	Single	RBF	GA	QAP
Horng et al. (2013)	Functional approximation	Single	ANN	MA	ATOP
Nguyen, Zhang, Johnston, and Tan (2014)	Evolutionary approximation	Single	*k*-NN	GA	JSP
Hao et al. (2016)	Functional approximation	Single	ANN	DE	SMSP
Wang and Jin (2018); Zheng, Fu, and Xuan (2019)	Functional approximation	Multi	RF	GA	KP
Lucas et al. (2020)	Functional approximation	Single	DT	VNS	VRP

tions by using clustering techniques (Sun, Zhang, Zhou, Zhang, & Zhang, 2019; Zhang et al., 2016).

In the following, first, we review, classify, and analyze the relevant papers, and then the corresponding challenges and future research directions are provided.

5.1. Literature classification & analysis

Table 2 classifies the papers using ML techniques for fitness evaluation of COPs based on different characteristics such as the *fitness evaluation* approach, the type of the problem (*single/ multi objective*), the employed *ML technique*, the *MH* algorithm, and the *COP* under study.

Considering Table 2, it can be seen that there are few studies applying fitness approximation/reduction to COPs. The reason is twofold; first, for most COPs, there exists an analytical fitness function whose evaluation is not computationally expensive (Hao & Liu, 2014), and second, constructing surrogate models for COPs is a complicated task and several additional issues should be overcome to create an accurate and reliable surrogate model for COPs (Pelamatti et al., 2020).

It can be seen from Table 2 that fitness approximation is mostly used for single-objective COPs whose original fitness function is calculated by a time-consuming simulation (Hao, Liu, Lin, & Wu, 2016; Horng, Lin, Lee, & Chen, 2013), or an approximate fitness function is used to help the search to escape from the local optima (Lucas, Billot, Sevaux, & Sörensen, 2020). There are also studies that apply fitness approximation to multi-objective COPs. Indeed, it is more computationally expensive for a MH to evaluate solutions using multiple fitness functions compared to a single fitness function, especially when there are too many objective functions.

Table 2 shows that fitness approximation/reduction is mostly used for EAs (e.g., GA and MA). The main reason is that EAs generate and evolve a population of solutions at each iteration and it might be therefore very time-consuming to evaluate every new solution at each iteration of the algorithm. Using fitness approximation especially becomes crucial when the evaluation of each new solution is computationally expensive (e.g., using time-consuming simulation to evaluate a solution (Horng et al., 2013)). This necessity has led to the development of new EAs called surrogate-assisted EAs.

5.2. Discussion & future research directions

As all MHs do an iterative process to reach the (near-) optimal solution, many fitness evaluations are needed to find an acceptable solution. Fitness approximation may help MHs to significantly reduce their computational effort for computing the fitness value (Jin, 2005). However, using fitness approximation in MHs is not as straightforward as one may expect, and it has its own challenges.

One of the major challenges is the accuracy of the approximate function and its functionality over the global search space (Jin, 2005). To replace the original fitness function of a MH with an approximate function, it has to be ensured that the MH with the approximate function converges to the (near-) optimal solution of the original function. However, due to some issues such as few training data and high dimensionality of the search space, it is difficult to construct such an approximate function. Therefore, to overcome this issue, one way is to use both the original and the approximate fitness function during the resolution of the problem. This is addressed as model management or evolution control in the literature (Jin, 2005).

An open question in fitness approximation is choosing the best suited technique for fitness approximation in COPs. The answer mainly depends on the COP under study and the user's preferences; however, due to numerous approximation techniques, selecting the best suited technique a priori is often impossible. In this regard, the first try could be using the simplest technique. If the performance of the approximate function obtained by the simplest technique is unsatisfactory or degrades over time, more sophisticated techniques can be used. A future research direction could provide a comparative study on the performance of different techniques for approximating the fitness function of COPs. These techniques may differ from simple techniques such as fitness inheritance or *k*-NN to more sophisticated ones such as polynomial regression, Kriging, RBF, and clustering/classification techniques (Shi & Rasheed, 2010). Usually more complex methods provide better fitting accuracy but need more construction time. Another way to answer this open question is using *ensemble surrogate modeling* that aggregates several surrogate models.

Apart from fitness approximation, another aspect that deserves to be explored more in the future is online fitness generation, wherein new objectives are targeted depending on the status of the search process. The new fitness function is generated based on some knowledge of the optimization problem at hand, as well as the features extracted from the visited regions during the search process. For instance, a set of representative features of the good solutions for the COP at hand can be extracted during the search process to form a new objective, and once the MH gets trapped in a local optimum, the original fitness function is replaced by the new one. During the search process, the original and the new fitness function can interchange to guide the MH toward promising solutions (Lucas et al., 2020). Another example of online fitness generation in MHs can be found in GLS that modifies the original fitness function when trapped in a local optimum (Voudouris & Tsang, 2003). Such a modification is done by adding a set of penalty terms to the original fitness function. Whenever the GLS gets stuck in a local optimum, the penalties are modified and the search process continues to optimize the transformed fitness function.

Real-time COPs are those COPs that need to be solved regularly (e.g., every hour or every day) under a time limitation. For these problems, the computational time spent even in one iteration of MHs, especially population-based MHs, may be too long for real-

time applications. Therefore, to cope with real-time COPs, especially large-scale COPs, one future research direction is to employ fitness approximation to lower the computational effort of fitness evaluation.

6. Initialization

There are three main strategies for generating initial solutions for MHs: *random*, *greedy*, and *hybrid* strategies (Talbi, 2009). In the random strategy, an initial solution is generated randomly, regardless of the quality of the solution. In the greedy strategy, a solution is initialized with a good-enough quality. Finally, the hybrid strategy combines two random and greedy strategies. There is always a trade-off between the use of these strategies in terms of exploration and exploitation. Indeed, the way of initializing a MH has a profound impact on its exploration and exploitation abilities. If the initial solutions are not well diversified, a premature convergence may occur, and the MH gets stuck in local optima. On the other hand, starting from low quality solutions may take a larger number of iterations to converge. In this regard, ML techniques can be used not only to maintain the diversity of solutions but also to produce initial solutions with good quality. ML techniques can contribute to the initialization through three major strategies:

- **Complete generation** – As a low-level integration, ML techniques can replace the solution generation strategies to construct the initial solution on their own. Indeed, ML techniques are used to construct a complete solution from an empty solution. The main ML techniques used in this category are RL-based techniques such as QL (Khalil, Dai, Zhang, Dilkina, & Song, 2017; dos Santos et al., 2014), ANN (Bengio, Frejinger, Lodi, Patel, & Sankaranarayanan, 2020), and Opposition-based Learning (Rahnamayan, Tizhoosh, & Salama, 2008).
- **Partial generation** – As a low-level integration, ML techniques are used to generate partial initial solutions using the apriori knowledge of good solutions. Then, the remaining part of the solution can be generated using any of the initialization strategies. ML techniques extract knowledge from previous good solutions and inject it into the new initial solutions (Li & Olafsson, 2005; Nasiri, Salesi, Rahbari, Meydani, & Abdollai, 2019). This knowledge is mostly in the form of ARs, which characterize the properties of good solutions. Apriori algorithms are widely used to extract the rules in ARs (Li, Chu, Chen, & Xing, 2016). Another example is case-based initialization strategy (Louis & McDonnell, 2004) derived from the idea of Case-Based Reasoning (CBR), in which the initial solutions are generated based on the solutions of already solved similar instances.
- **Decomposition** – As a problem-level integration, the decomposition is done either in the data space or in the search space. In data space decomposition, ML techniques are used to decompose the data space into several sub-spaces and consequently facilitate generating initial solutions by reducing the required computational effort. In this regard, an initial solution is generated for each sub-space using any of the initialization strategies, and finally a complete solution is constructed from the partial solutions of the sub-spaces (Ali, Essam, & Kasmarik, 2020; Chang, 2017; Min, Jin, & Lu, 2019). In search space decomposition, ML techniques are used to diversify the initial solutions over the search space, where different solutions represent different regions of the search space. For example, the problem of selecting the sub-regions of the search space to explore can be formulated using the Multi-armed Bandit (MAB) technique, wherein each arm represents a region of the search space, and the technique learns which regions worth exploring further and which are not (Catteeuw, Drugan, & Manderick, 2014).

Depending on how ML techniques are employed in generating the initial solutions, learning can occur either *offline* or *online*. In offline learning, knowledge is gathered from the initial solutions generated for a set of training instances with the aim to generate initial solution(s) for a new problem instance. The properties of those good initial solutions that led to a better performance are extracted and used to generate promising initial solution(s) for a new problem instance. Although an offline learning can provide rich knowledge, it might be very time-consuming, and the extracted knowledge might not be useful enough when applied to a new problem instance with completely different properties compared to the training instances. On the contrary, in online learning, knowledge is extracted and employed dynamically while generating the initial solution(s) for a problem instance. Although the extracted knowledge might not be that rich, it completely suites the instance at hand. In the rest of this section, the relevant papers are reviewed and analyzed and the corresponding challenges along with future research directions are provided.

6.1. Literature classification & analysis

Table 3 classifies the papers generating initial solutions for COPs using ML techniques based on different characteristics such as the initialization *strategy*, *learning* mechanism, the used *ML technique*, the *MH* algorithm for which the initialization is performed, the *COP* under study, and the *size* of the training set in case of offline learning.

Considering Table 3, RL, particularly QL is a widely used technique to generate complete initial solutions. QL can be counted as a hybrid initialization strategy that balances exploration and exploitation abilities of a MH through its parameters. QL constructs the solutions successively by exploiting the knowledge of the search space using the reward matrix. Taking TSP as an example, QL starts construction with a random city and proceeds with the cities which bring the maximum reward, where the reward is representative of the problem's objective function (e.g., the reward is inversely related to the distance from the current selected city to the next potential city). The superiority of QL over other typical initialization strategies in terms of the quality of the solutions and convergence rate of the algorithm has been illustrated for TSP by dos Santos et al. (2014).

Table 3 shows that the decomposition strategy is mostly used for routing-based COPs including VRP and TSP. Clustering algorithms such as *k*-means are the widely used ML techniques in these studies, where the cities are clustered into several groups, and an initial solution is obtained by using a greedy strategy for each group. Finally, a complete path connecting different groups is created.

6.2. Discussion & future research directions

In the integration of ML techniques into MHs for generating initial solutions, the simplicity of the classical random and greedy strategies is sacrificed to gain better performance in terms of the trade-off between exploration and exploitation through more advanced initialization strategies (i.e., complete generation, partial generation and decomposition). The integration of ML techniques into initialization of MHs has been reported to lead to an improvement in the convergence rate of MHs when better solutions have been found in less computational efforts compared to classical random and greedy strategies (Ali et al., 2020; Hassanat, Prasath, Abbadi, Abu-Qdari, & Faris, 2018; Louis & McDonnell, 2004; dos Santos et al., 2014). These improvements have been more expressive when solving larger instances of COPs as MHs start with already

Table 3
Classification of papers studying initialization.

Ref.	Strategy	Learning	ML tech.	MH	COP	Size
Louis and McDonnell (2004)	Partial generation	Offline	Apriori	GA	JSP	50
Li and Olafsson (2005)	Partial generation	Offline	DT	MHs	JSP	19,900
de Lima Junior, de Melo, and Neto (2007); De Lima, De Melo, and Neto (2008); dos Santos, de Lima Júnior, Magalhães, de Melo, and Neto (2010)	Complete generation	Online	QL	GRASP, GA	TSP	–
Díaz-Parra, Ruiz-Vanoye, and Zavala-Díaz (2010); Haghighi, Zahedi, and Ghazizadeh (2010)	Decomposition	Online	*k*-means	GA	VRP	–
dos Santos et al. (2014)	Complete generation	Online	QL	VNS	TSP	–
Catteeuw et al. (2014)	Decomposition	Online	MAB	HC	QAP	–
Deng, Liu, and Zhou (2015)	Decomposition	Online	*k*-means	GA	TSP	–
Zhou et al. (2016)	Complete generation	Online	RL	DLS	GCP	–
Li et al. (2016)	Partial generation	Online	Apriori	GA	TSP	–
Xiang and Pan (2016); Zhang (2017)	Decomposition	Online	*k*-means	ACO	VRP	–
Gao, Wang, Cheng, Inazumi, and Tang (2016)	Decomposition	Online	*k*-means	ACO	LRP	–
Khalil et al. (2017)	Complete generation	Online	QL	MHs	TSP	–
Chang (2017)	Decomposition	Online	*k*-means	ACO	TSP	–
Hassanat et al. (2018)	Decomposition	Online	LR	GA	TSP	–
López-Santana, Rodríguez-Vásquez, and Méndez-Giraldo (2018)	Decomposition	Online	*k*-means	ACO	WSRP	–
Alipour, Razavi, Derakhshi, and Balafar (2018)	Complete generation	Online	RL	GA	TSP	–
Miki, Yamamoto, and Ebara (2018)	Partial generation	Offline	DRL	MHs	TSP	2,000,00
Ali, Essam, and Kasmarik (2019)	Decomposition	Online	*k*-means	GA, DE	TSP	–
Gocken and Yaktubay (2019)	Decomposition	Online	*k*-means	GA	VRP	–
Min et al. (2019)	Decomposition	Online	*k*-means	TS	VRP	–
Nasiri et al. (2019)	Partial generation	Offline	Apriori	GA, PSO	JSP	35
Bengio et al. (2020)	Complete generation	Offline	LR, ANN	MHs	FLP	45,000
Lodi, Mossina, and Rachelson (2020)	Partial generation	Offline	LogR, ANN, NB	MHs	FLP	7145
Ali et al. (2020)	Decomposition	Online	*k*-means	DE	TSP	–
Cheng, Pourhejazy, Ying, and Lin (2021)	Decomposition	Online	*k*-means	ABC	JSP	–

a (set of) good initial solution(s) (Hassanat et al., 2018; dos Santos et al., 2014). This phenomenon saves the computational effort toward exploring/exploiting more promising regions in the solution space instead of spending extensive efforts to find primary good (local) solutions during the search process.

These advanced initialization strategies bring their own challenges. An important challenge is the complexity of using such advanced techniques with additional parameters that need to be carefully tuned to get the highest performance. A set of other challenges arises depending on the way solutions are initialized. For instance, a big challenge when using QL is how to define the set of states and actions so that they satisfy the properties of a Markov decision process. One of the main requirements is to define a set of states such that it is sufficient to characterize the system without the need for the history of information achieved so far. Taking TSP as an example, if one considers the state as the currently selected city and the action as the next city to be added to the tour, the state is not sufficient to characterize the system, and an action depends on the information on the history of the states.

As a new concept in ML and inspired from the opposite relationship among entities, Opposition-based Learning can provide efficient strategies to generate the initial population. Using this concept, the initial population is approximated from two opposite sides (Rahnamayan et al., 2008), wherein an initial population is first generated randomly. Next, an opposite population in terms of values of the solution vector is generated to the randomly generated population. The MH then merges the two populations and selects half of the best solutions to form the initial population. The main aim of opposition-based learning is keeping the diversity between the initial solutions to increase the exploration ability of the MHs. In addition to opposition-based learning, an interpolation technique can be also used to generate the initial population. This technique attempts to provide good solutions by interpolating a set of randomly generated solutions. The interpolated solutions are then used to form the initial population. Neither opposition-based learning nor interpolation have been applied to COPs. Therefore, using these techniques and other advanced ML techniques such as ANNs (Yalcinoz & Altun, 2001) to generate the initial solutions of COPs could be a future research direction.

7. Evolution

ML techniques can be integrated into the evolution process in three major ways:

- ML techniques help to use feedback information on the performance of the operators during the search process to select the most appropriate operator (see Section 7.1).
- ML techniques provide a learning mode to generate new populations in EAs (see Section 7.2).
- ML techniques help to extract the properties of good solutions to generate new solutions (see Section 7.3).

7.1. Operator selection

Operator selection has its roots in *hyper-heuristics*. The term *hyper-heuristic* can be defined as a high-level automated search methodology which explores a search space of low-level heuristics (i.e., neighborhood or search operators) or heuristic components, to solve optimization problems (Burke et al., 2013). Regarding the nature of the heuristic search space, hyper-heuristics are classified into *heuristic selection* and *heuristic generation* methodologies (Drake, Kheiri, Özcan, & Burke, 2019). The former aims at selecting among a set of heuristics, while the latter aims at generating new heuristics. Operator selection inherently belongs to *heuristic selection* methodologies in hyper-heuristics; however, it has been also used in designing MHs (Mosadegh et al., 2020; dos Santos et al., 2014). Accordingly, it has led to a certain level of confusion in the literature in distinguishing MHs involving operator selection from hyper-heuristics. Despite the differences between the

design and performance of hyper-heuristics and MHs, operator selection in both methods targets the same goal as selecting and applying (an) appropriate operator(s) during the search process. In this paper, the focus is on MHs involving operator selection. In this regard, a MH called adaptive large neighborhood search, an extension to the classical large neighborhood search, has been developed with the particular aim of selecting operators during its search process. A meta-analysis on adaptive large neighborhood search MHs has been provided by Turkeš, Sörensen, and Hvattum (2020). Apart from adaptive large neighborhood search, this paper focuses on operator selection in all other types of MHs.

The main motivation of operator selection comes from the fact that individual operators may be particularly effective at certain stages of the search process, but perform poorly at others. Indeed, the search space of COPs is a non-stationary environment including different search regions with dissimilar characteristics, in which different operators (or different configurations of the same operator) show different behavior, and only a set of them works well in each region (Fialho, 2010). Therefore, solving COPs using a single operator does not guarantee the highest performance in finding (near-) optimal solutions. Hence, it is reasonable to expect that employing different operators combined in an appropriate way during the search process will produce better solutions.

Through designing hyper-heuristics or MHs with multiple search operators, we can take advantage of several operators with different performances by switching between them during the search process. In addition, an appropriate implementation of different operators significantly affects the exploration and exploitation abilities of a MH and provides an Exploration-Exploitation balance during the search process. In designing such algorithms, there are two important decisions to make: 1) *which* operator (among multiple operators), and 2) *which* settings to select and apply at each stage of the search process. The former is discussed in this section while the latter will be explained in Section 8.

Search operators are divided into four Mutational/Perturbation Operators (MPOs), Ruin-Recreate Operators (RROs), Local Search or Hill-Climbing Operators (LSOs), and Crossover or Recombination Operators (XROs) categories (Ochoa & Burke, 2014; Talbi, 2009).

ML techniques help the operator selection to use feedback information on the performance of the operators. In this situation, operators are selected based on a credit assigned to each operator (i.e., feedback from their historical performance). Considering the nature of the feedback, the learning can be *offline* or *online*. In offline learning, ML techniques such as case-based reasoning (Burke, Petrovic, & Qu, 2006) help to gather knowledge in the form of rules from a set of training instances with the aim to select operators in new problem instances. However, in online learning, knowledge is extracted and employed dynamically while solving a problem instance (Burke et al., 2019; Talbi, 2016). The use of online feedback from the search environment to dynamically select and apply the most appropriate operator during the search process is referred to as *Adaptive Operator Selection* (AOS) (Fialho, 2010). AOS aims at selecting the most appropriate operator at each stage of the search process while solving a problem instance. AOS gives the chance to adapt MHs behavior to the characteristics of the search space by selecting their operators during the search process based on their historical performance. AOS consists of five main steps:

- **Performance criteria identification** – Whenever an operator is selected and applied to a problem instance, a set of feedback information can be collected that represents the performance of the operator. This feedback can be different performance criteria such as OFV, Diversity of Solutions (DOS), CT, and Depth of the Local Optima (DLP). The credit of an operator highly depends on how the performance criteria are identified and assessed. This makes performance criteria identification an important step in AOS. Therefore, in solving COPs, the performance criteria should be efficiently identified and integrated (in case of multiple criteria) to lead the MH toward the optimal solution.
- **Reward computation** – Once the performance criteria are identified, this step computes how much the application of each operator improves/deteriorates the performance criteria.
- **Credit assignment (CA)** – In this step, a credit is assigned to an operator based on the rewards calculated in the reward computation step. There are different credit assignment methods including Score-based CA (SCA) (Peng, Zhang, Gajpal, & Chen, 2019), Q-Learning based CA (QLCA) (Wauters, Verbeeck, De Causmaecker, & Berghe, 2013), Compass CA (CCA) (Maturana, Fialho, Saubion, Schoenauer, & Sebag, 2009), Learning Automata based CA (LACA) (Gunawan, Lau, & Lu, 2018), Average CA (ACA) (Fialho, 2010), and Extreme Value-based CA (EVCA) (Fialho, 2010) presented in Table 4.
- **Selection** – Once a credit has been assigned to each operator, AOS selects the operator to apply in the next iteration. Different selection methods including Random selection (RS), Max-Credit Selection (MCS), Roulette-wheel Selection (RWS) (Gunawan et al., 2018), Probability Matching Selection (PMS) (Fialho, Da Costa, Schoenauer, & Sebag, 2008), Adaptive Pursuit Selection (APS) (Fialho et al., 2008), Soft-Max Selection (SMS) (Gretsista & Burke, 2017), Upper Confidence Bound Multi-Armed Bandit Selection (UCB-MABS) (Fialho et al., 2008), Dynamic Multi-Armed Bandit Selection (D-MABS) (Maturana et al., 2009), Epsilon Greedy Selection (EGS) (dos Santos et al., 2014), and Heuristic-based Selection (HS) (Choong, Wong, & Lim, 2018) are presented in Table 5.
- **Move acceptance** – After the application of an operator, AOS decides whether to accept the move provided by the operator or not. Different move acceptance methods including All Moves Acceptance (AMA), Only Improvement Acceptance (OIA), Naive Acceptance (NA), Threshold Acceptance (TA), Metropolis Acceptance (MA) (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), Probabilistic Worse Acceptance (PWA), Simulated Annealing Acceptance (SAA) (Mosadegh et al., 2020), and Late Acceptance (LTA) are listed in Table 6.

7.1.1. Literature classification & analysis

Table 7 classifies the papers studying AOS for COPs based on different characteristics such as *performance criteria*, *credit assignment* method, *selection* method, *move acceptance* method, the *MH* algorithm, type of the *operators* to be selected, and the *COP* under study.

Considering the performance criteria, Table 7 indicates that all reviewed papers rely on OFV as the criterion used for evaluating the operators. In the meantime, only few papers have incorporated other criteria such as CT and DOS into their evaluation process (Di Tollo, Lardeux, Maturana, & Saubion, 2015; Maturana et al., 2009; Maturana & Saubion, 2008; Sakurai, Takada, Kawabe, & Tsuruta, 2010; Sakurai & Tsuruta, 2012). Although OFV is the most straightforward criterion by which the operators can be evaluated, DOS is also needed to avoid premature convergence (Di Tollo et al., 2015).

By looking at Table 7, it can be seen that among the credit assignment methods, RL-based methods (e.g., simple SCA and QLCA) have been mostly studied. Among the studied credit assignment methods, the ACA method is biased toward conservative strategies. Indeed, using this method, the operators with frequent small improvements are preferred over operators with rare but high improvements. Despite the ACA method, the EVCA method assumes that rare but high improvements are even more important than frequent but moderate ones (Fialho et al., 2008). Using the EVCA method, the operators are credited based on the maximum im-

Credit assignment (CA) – In this step, a credit is assigned to an operator based on the rewards calculated in the reward computation step. There are different credit assignment methods including Score-based CA (SCA) (Peng, Zhang, Gajpal, & Chen, 2019), Q-Learning based CA (QLCA) (Wauters, Verbeeck, De Causmaecker, & Berghe, 2013), Compass CA (CCA) (Maturana, Fialho, Saubion, Schoenauer, & Sebag, 2009), Learning Automata based CA (LACA) (Gunawan, Lau, & Lu, 2018), Average CA (ACA) (Fialho, 2010), and Extreme Value-based CA (EVCA) (Fialho, 2010) presented in Table 4.

Table 4
Credit assignment methods.

Method	
SCA	As a simple version of RL, it assigns an initial score (credit) to each operator and updates their credits based on their performance at each step of the search process. Generally, initial credits are set to a same value, typically zero. $C_{i,t+1} = C_{i,t} + r_{i,t}$ where $C_{i,t}$ and $r_{i,t}$ are the credit and the reward of operator i at time (step) t.
QLCA	It assigns a Q-value (credit) to an operator (action) at each state of the search process based on its previous performance. $Q(s_t, a) = Q(s_t, a) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a)]$ where s_t is the state at time t, s_{t+1} is the new state at time $t+1$, a is the operator selected in state s, a' is a possible operator in state s_{t+1}, r_t is the reward (punishment) received after selecting operator a at time t, α is the learning rate, and γ is the discount factor which indicates the importance of future rewards.
CCA	It integrates three measures at every application of an operator: population diversity variation, mean fitness variation, and execution time. A sliding window stores the last W changes in terms of *diversity* and *fitness*. A compromised value is then calculated between the diversity and fitness measures. Finally, the compromised value is divided by the operator's execution time to obtain the final credit of each operator.
LACA	It assigns a probability value to each operator based on its previous performance. The following formulations show the selection probabilities (credits) of successful and unsuccessful operators, respectively. $p_{i,t+1} = p_{i,t} + \lambda_1 r_{i,t}(1 - p_{i,t}) - \lambda_2(1 - r_{i,t})p_{i,t}$ $p_{i,t+1} = p_{i,t} - \lambda_1 r_{i,t} p_{i,t} + \lambda_2(1 - r_{i,t})[(K-1)^{-1} - p_{i,t}]$ where $p_{i,t}$ is the selection probability of operator i at time t, λ_1 and λ_2 refer to the learning rates used to update the selection probabilities, and K is the number of operators.
ACA	It assigns credit to each operator according to its performance achieved by its last W applications (Instantaneous credit assignment if W=1). Using W as the size of the sliding window for each operator, the performance of an operator is aggregated over a given time period.
EVCA	It follows the principle that infrequent, yet large improvements in the performance criteria are likely to be more effective than frequent but moderate improvements. At each application of an operator, the changes in the performance criteria are added to a sliding window of size W following a FIFO rule and the maximum value within the window is assigned as a credit to that operator. Despite the ACA, this method emphasizes on rewards to the operators with recent large improvements even once throughout their last W applications. $C_{i,t} = \max_{t'=t-W,\ldots,t}\{r_{i,t'}\}$

Selection – Once a credit has been assigned to each operator, AOS selects the operator to apply in the next iteration. Different selection methods including Random selection (RS), Max-Credit Selection (MCS), Roulette-wheel Selection (RWS) (Gunawan et al., 2018), Probability Matching Selection (PMS) (Fialho, Da Costa, Schoenauer, & Sebag, 2008), Adaptive Pursuit Selection (APS) (Fialho et al., 2008), Soft-Max Selection (SMS) (Gretsista & Burke, 2017), Upper Confidence Bound Multi-Armed Bandit Selection (UCB-MABS) (Fialho et al., 2008), Dynamic Multi-Armed Bandit Selection (D-MABS) (Maturana et al., 2009), Epsilon Greedy Selection (EGS) (dos Santos et al., 2014), and Heuristic-based Selection (HS) (Choong, Wong, & Lim, 2018) are presented in Table 5.

Table 5
Selection methods.

Method	Description
RS	It uniformly selects an operator at random, ignoring the credit $p_{i,t} = \frac{1}{K}$ where K is the number of operators.
MCS	It selects an operator with the maximum credit.
RWS	It assigns a selection probability $p_{i,t}$ to operator i at time t bas these probabilities. The more the credit of an operator, the mo $p_{i,t+1} = \frac{C_{i,t}}{\sum_{j=1}^{K} C_{j,t}}$ where $C_{i,t}$ is the assigned credit of operator i at time t, and K
PMS	It assigns a selection probability to each operator based on its all operators to give them a chance to be selected regardless o $p_{i,t+1} = p_{\min} + (1 - K \times p_{\min}) \frac{C_{i,t}}{\sum_{j=1}^{K} C_{j,t}}$ where $p_{\min}$ is the minimum selection probability of each opera operator i at time t.
APS	Instead of proportionally assigning probabilities to all operators, it selects the operator with the maximum credit, increases its probability, and reduces the probabilities of all other operators. Operator i^* is selected as follow: $\begin{cases} p_{i,t+1} = p_{i,t} + \beta(1 - (K-1)p_{\min} - p_{i,t})(\beta > 0) & i^* = arg \max\{C_{i,t}\} \\ p_{i,t+1} = p_{i,t} + \beta(p_{\min} - p_{i,t}) & \text{otherwise} \end{cases}$ where the notations are similar to PMS and β is the learning rate.
SMS	It uses a Boltzmann distribution to transform the credit of each operator to a probability, and involves a temperature parameter τ to amplify or condense the differences between the operator probabilities. It uniformly selects an operator based on the probability values. As long as the temperature decreases, this method becomes more greedy towards selecting the best available operator. $p_{i,t+1} = \frac{e^{(C_{i,t}/\tau)}}{\sum_{j=1}^{K} e^{(C_{j,t}/\tau)}}$
UCB-MABS	It assigns a cumulative credit to each operator and selects the operator with the maximum value of: $C_{i,t} + G\sqrt{\frac{\log \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}$ where $n_{i,t}$ denotes the number of times the ith arm has been played up to time t and G is the *Scaling* factor used to properly balance rewards and application frequency (Exploration-Exploitation balance) while still maintaining a small selection probability of other operators for exploration purposes.
D-MABS	It adapts the classical multi-armed bandit scenario to a dynamic context where the reward probability of each arm is neither independent nor fixed. To address the dynamic context, the classical UCB (Auer, Cesa-Bianchi, & Fischer, 2002) algorithm is combined with a Page Hinkley test (Page, 1954), to identify the change of reward probabilities.
EGS	It selects the operator with the highest credit with probability of $(1 - \epsilon)$; otherwise, it selects an operator randomly. $i^* = \begin{cases} arg \max_j C_{j,t} & \text{with probability } 1 - \epsilon \\ \text{any other operator} & \text{with probability } \epsilon \end{cases}$
HS	It uses either heuristic rules or optimization algorithms to select the operators. A heuristic rule can be a tabu list of operators to exclude them from being selected during a certain number of iterations. On the other hand, the sequence of operators can be defined as a decision variable and optimization algorithms (e.g., MHs) are employed to find the optimal sequence.

Move acceptance – After the application of an operator, AOS decides whether to accept the move provided by the operator or not. Different move acceptance methods including All Moves Acceptance (AMA), Only Improvement Acceptance (OIA), Naive Acceptance (NA), Threshold Acceptance (TA), Metropolis Acceptance (MA) (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), Probabilistic Worse Acceptance (PWA), Simulated Annealing Acceptance (SAA) (Mosadegh et al., 2020), and Late Acceptance (LTA) are listed in Table 6.

Table 6
Move acceptance methods.

Method	Description
AMA	It always accepts the applied move regardless of whether the move improves the solution or not.
OIA	It only accepts the moves that improve the solution.
NA	It always accepts the moves that improve the solution and considers an acceptance probability of 0.5 for moves that deteriorate the solution.
TA	It always accepts the moves that improve the solution and accepts the moves that deteriorate the solution less than a prefixed threshold (in terms of the solution quality).
MA	It accepts each move with a probability of $e^{\frac{\Delta f}{T}}$, where Δf is the difference between the fitness values before and after applying that move, and T denotes the temperature. The higher the value of T, the higher the chance to accept worse moves and vice versa.
PWA	As a variant of Metropolis acceptance, it accepts each move with a probability of $e^{\frac{\Delta f}{T*\mu_{impr}}}$, where μ_{impr} is the average of previous improvements on the solution quality.
SAA	As a variant of Metropolis acceptance, it accepts each move with a probability $e^{(\frac{\Delta f}{T*\mu_{impr}} \times \frac{t_{\max}}{t_{\max}-t_{current}})}$, wherein the temperature T decreases gradually over time. In addition, $t_{\max}$ denotes the maximum time allowed to execute the algorithm, and $t_{current}$ denotes the current elapsed time.
LTA	It accepts a move that provides a solution with better or equal quality compared to the obtained solutions in the last n iterations. During the initial n iterations, any move that provides a better solution compared to the initial solution is accepted.

Table 7
Classification of papers studying AOS.

Search operators are divided into four Mutational/Perturbation Operators (MPOs), Ruin-Recreate Operators (RROs), Local Search or Hill-Climbing Operators (LSOs), and Crossover or Recombination Operators (XROs) categories (Ochoa & Burke, 2014; Talbi, 2009).

Ref.	Perf. criteria	Credit asnt.	Selection	Move acpt.	MH	Operator	COP
Pettinger and Everson (2002)	OFV	SCA	MS	OIA	GA	MPO, XRO	TSP
Maturana and Saubion (2008)	OFV, DOS, CT	CCA	PMS	OIA	GA	MPO, XRO	SAT
Maturana et al. (2009)	OFV, DOS, CT	CCA, EVCA	D-MABS	OIA	GA	MPO, XRO	SAT
Sakurai et al. (2010); Sakurai and Tsuruta (2012)	OFV, CT	QLCA	SMS	OIA	GA	MPO, XRO	TSP
Francesca, Pellegrini, Stützle, and Birattari (2011)	OFV	EVCA	APS, PMS	OIA	MA	XRO	QAP
Burke, Gendreau, Ochoa, and Walker (2011)	OFV,FT	EVCA	APS, RWS	AMA	ILS	MPO, XRO, RRO, LSO	FSP
Walker, Ochoa, Gendreau, and Burke (2012)	OFV	EVCA	APS	AMA-OIA	ILS	MPO, XRO, RRO, LSO	VRP
Handoko, Nguyen, Yuan, and Lau (2014)	OFV	QLCA	PMS, APS	OIA	MA	XRO	QAP
dos Santos et al. (2014)	OFV	QLCA	EGS	OIA	VNS	LSO	TSP
Consoli and Yao (2014)	OFV, DOS	CCA	D-MABS	OIA	MA	XRO	ARP
Buzdalova, Kononov, and Buzdalov (2014)	OFV	QLCA	EGS, SMS	OIA	EA	MPO, XRO	TSP
Yuan, Handoko, Nguyen, and Lau (2014)	OFV	SCA	APS, D-MABS	OIA	MA	XRO	QAP
Di Tollo et al. (2015)	OFV, DOS, CT	CCA	PMS	OIA	EA	XRO	SAT
Li, Pardalos, Sun, Pei, and Zhang (2015)	OFV	ACA	RWS	OIA, AMA	ILS	MPO, LSO	VRP
Li and Tian (2016)	OFV	SCA	RWS	OIA	VNS	LSO	VRP
da Silva, Freire, and Honório (2016)	OFV	QLCA	APS	OIA	EA	MPO, XRO	TEPP
Mohammadi, Tavakkoli-Moghaddam, Siadat, and Dantan (2016)	OFV	SCA	RWS	OIA	ICA	XRO	HLP
Chen et al. (2016)	OFV	SCA	UCB-MABS	OIA	VNS	LSO	VRP
Zhalechian, Tavakkoli-Moghaddam, Zahiri, and Mohammadi (2016)	OFV	SCA	RWS	OIA	GA	MPO, XRO	LRP
Gretsista and Burke (2017)	OFV	ACA	UCB-MABS	SAA	ILS	MPO	NRP
Gunawan et al. (2018)	OFV	LACA	RS, RWS	OIA, SAA	ILS	LSO	OP
Ahmadi et al. (2018)	OFV, DOS	QLCA	EEGS	OIA	GA	MPO, XRO	SSP
Mohammadi, Jula, and Tavakkoli-Moghaddam (2019)	OFV	SCA	RWS	OIA	GA	MPO, XRO	HLP
Peng et al. (2019)	OFV	SCA	RWS	OIA	MA	LSO	VRP
Lu, Zhang, and Yang (2019)	OFV	SCA	RWS	OIA	ILS	MPO, LSO	VRP
Mosadegh et al. (2020)	OFV	QLCA	EGS	SAA	SA	LSO	MASP
Zhao et al. (2021)	OFV	QLCA	EGS	OIA	WWO	MPO	FSP

provement during their last W applications. In order to avoid the premature convergence, the CCA method has incorporated DOS into its evaluation process. This method takes into consideration both the OFV and DOS, which are related to the exploitation and exploration abilities of MHs, respectively. The ACA, EVCA, and CCA methods assign credit to the operators based on their immediate performance (through the last W applications). This may rise the possibility that the optimization is short sighted.

On the other hand, RL-based methods including the SCA, QLCA, and LACA methods assign credit to the operators based on their performance from their very first application. Indeed, they are able to learn a policy to maximize the rewards in a long-term prospect, which makes it possible to gain optimal operator selection policies. In addition to a long-term prospect, some RL-based methods such as QL are model-free that do not require the complete model of the system including the matrix of transition probabilities and the expected value of the reward for each state-action pair. This is especially useful for COPs since generally, a complete model is not available for COPs (Wauters et al., 2013). Furthermore, many RL-based methods are able to converge to the optimal state-action pair under several conditions. Among these methods, QL has proven to converge to the optimal state-action pair under three conditions: the system model is a Markov decision process, each state-action pair is visited many times, and the immediate reward given to each action is not unbounded (i.e., limited to some constant) (Watkins, 1989). Among the RL-based methods in Table 7, the techniques based on Temporal Differences such as QL take advantage of the concept of delayed reward. They are based on the assumption that there might be a delay before the effect of an action appears. Accordingly, they consider a delayed reward in addi-

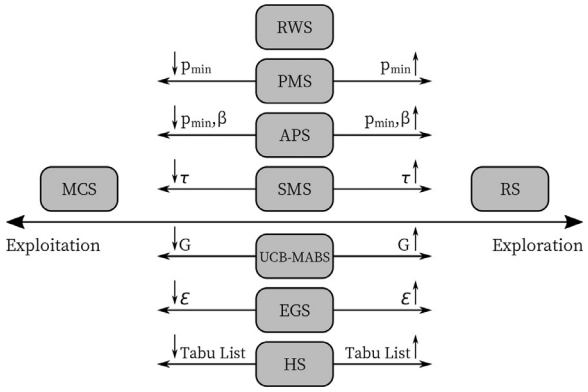

Fig. 4. Exploration vs. Exploitation of selection methods.

tion to an immediate reward. On the other hand, the LACA method (Gunawan et al., 2018) and the SCA method (Chen, Cowling, Polack, & Mourdjis, 2016) work in a single state environment and merely take into account an immediate reward.

The selection step decides which operator to apply in the next iteration. This step can also be seen as an exploration and exploitation dilemma, where there is a need to be a trade-off between selecting the best operator with the best performance so far (i.e., exploitation), and giving a chance to the other operators which may bring the best performance from then (i.e., exploration). Fig. 4 illustrates how different selection methods balance exploration and exploitation abilities of a MH. For each selection method, the responsible parameters and their corresponding effects for making such balance have been also identified. For example, in the APS method, increasing the parameters $p_{\min}$ and β augments the exploration ability of the method, while decreasing $p_{\min}$ and β increases the exploitation ability of the method.

As it can be seen in Fig. 4, the MCS method is a purely exploitation-based method where there is no chance for exploring other operators. In other words, the MCS method selects the operator with the maximum credit at each iteration, giving no chance to other lower quality operators to be selected.

The RWS method selects the operators based on their proportional credit, where there is also a chance for all operators to be selected. However, using the RWS method, operators which do not show good performance during a long time, have very low or even zero chance to be selected, while they might perform well in the latter stages of the search process. To tackle this issue, the PMS method assigns a minimum selection probability to each operator, $p_{\min}$, regardless of its performance. This preserves the balance between exploration and exploitation thorough a minimal level of exploration that is kept fixed during the search process. Indeed, the selection probability of operators with zero credit slowly converges to $p_{\min}$. In this way, the operators with even moderate performance keep being selected, and this degrades the performance of this method (Maturana et al., 2009). To address this drawback, the proposed APS method updates the selection probabilities using the winner-takes-all strategy. Indeed, instead of updating the probabilities based on operators' proportional credit, the APS method increases the selection probability of the best operator while decreasing the selection probability of all other operators. This aims at quickly enhancing the application probability of the current best operator (Fialho, 2010). Considering the trade-off between exploration and exploitation, this method keeps a minimal level of exploration through $p_{\min}$, which is fixed throughout the search process.

In the SMS method, the balance between exploration and exploitation is controlled through a temperature parameter, τ. As the temperature increases, the selection probabilities tend to be equal for all operators. While decreasing the temperature leads to a larger difference between selection probabilities. In an extreme viewpoint, when the temperature goes to zero, SMS becomes a purely exploitation-based method where only the best operator is selected. The UCB-MABS method also makes a trade-off between the exploration and exploitation abilities of a MH by keeping a minimum selection probability for each operator to be selected through $G\sqrt{\frac{\log \sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}$, where G is a scaling factor to balance the trade-off. Similarly, the EGS method controls the trade-off between exploration and exploitation using the predefined parameter ϵ. Increasing ϵ increases the exploration ability of the algorithm by giving a chance to the other operators to be selected, while decreasing ϵ favors the selection of the best operator. Accordingly, when $\epsilon = 1$, the EGS method becomes a purely exploration-based method. Finally, the HS method uses a heuristic rule to select operators. For instance, the rule could be a tabu list of operators that excludes successful operators from being selected during a certain number of iterations. The size of the tabu list makes a balance between exploration and exploitation. As the tabu size increases, the successful operators remain longer in the tabu list, giving a chance to other operators to be selected, which leads the method toward exploration.

Table 7 indicates that AOS is mostly applied to EAs (e.g., GA and MA) to select the mutational and crossover operators. The reason may rely on the popularity of EAs for solving COPs and the availability of a variety of problem-specific mutational and crossover operators, which needs to be carefully selected during the search process since the performance of EAs is highly affected by its operators. In the meanwhile, AOS is also applied to select the local search operators in ILS and VNS.

7.1.2. Discussion & future research directions

In this section, first, a guideline is provided for researchers on when to use AOS, and the fundamental requirements of using AOS are identified. Next, challenges and future research directions are discussed.

There is a rise in the number and variety of problem-specific operators (heuristics) for efficiently solving optimization problems. Selecting and applying these operators within a MH requires much expertise in the domain. That is especially the case for COPs with plenty of proposed problem-specific operators where the classical operators are not as competent as problem-specific ones. Indeed, for COPs with standard representations (e.g., permutation-based representations), users can make the use of classical non-specialized operators, which does not necessarily require much expertise. However, for COPs out of this standard framework, one must have knowledge over the problem-specific operators to efficiently select the operators. This issue highlights the necessity of an automatic approach to select the most appropriate operator(s) based on their performance without having an expertise in the domain. In this way, even inexperienced users are able to select appropriate operators for solving COPs. In addition, AOS is most useful when dealing with several competitive state-of-the-art operators for solving a COP such that none of them can be preferred over the others a priori, and choosing the best operator exhaustively is computationally expensive. In this condition, there is a need for automatic operator selection.

There are a set of specific requirements for the QLCA method as the most common RL-based method used in AOS. Before applying the QLCA method in AOS, one needs to define the set of possible states and actions. In defining the states, the following preconditions should be checked:

- The states should be completely descriptive of the problem status to allow selecting the correct action. There are three ways to define the states. The states could be 1) search-dependent that reflect the properties of the search process such as the number of non-improving iterations, 2) problem-dependent that reflect the properties of the problem through generic features, or 3) instance-dependent that reflect the properties of the problem instance such as the number of bins in a bin packing problem (Wauters et al., 2013).
- The states should be defined in such a way that do not grow exponentially and allow the algorithm to visit each state-action pair many times. This is one of the main conditions of QLCA method convergence. For instance, if the number of states grows exponentially with the size of the problem, the algorithm might need to be executed more times to satisfy the convergence condition.

The integration of ML techniques into AOS has led to an improvement in terms of both solution quality and even computational time. Using advanced ML techniques in AOS such as QLCA (Ahmadi, Goldengorin, Süer, & Mosadegh, 2018; Mosadegh et al., 2020; Zhao, Zhang, Cao, & Tang, 2021) and LACA (Gunawan et al., 2018) has brought significant improvements compared to the non-learning version of MHs. Besides, even new best-known solutions have been obtained for certain COPs (Ahmadi et al., 2018; Gunawan et al., 2018). More interestingly, such integration has been reported to be more efficient as the size of the problem instances increases, and the proposed MHs have shown more stable behavior when solving larger COP instances (dos Santos et al., 2014; Zhao et al., 2021).

Apart from the advantages that AOS brings into application, a set of important challenges arises in this regard. The first challenge is related to the computational overhead of learning in the MHs. Although AOS gives the user the flexibility to adapt the MH's behavior to the characteristics of the search space by selecting its operators during the search process, achieving such flexibility adds an extra computational overhead. Keeping track of the performance of the operators during the search process, assigning credits to them, and updating their selection probabilities all impose an extra computational overhead. This overhead can be compensated by an optimal design of the AOS mechanism wherein the MH converges sooner to the (near-) optimal solution, and consequently the saved computational time compensates the extra overhead.

Another challenge is related to the tuning of the parameters of AOS. Most credit assignment and selection methods introduce new parameters that need to be tuned before applying AOS. Tuning the values of these parameters can significantly affect the performance of AOS; thus they need to be carefully tuned. For example, in the QLCA method, there are two new parameters, the learning rate (α) and the discount factor (γ). The former controls the ratio of accepting the newly learned information while the latter controls the impact of the future reward. Higher levels of α tend to the replacement of the old information by new information. On the other hand, lower values of α emphasize on the existing information. Furthermore, as γ increases, more emphasis is given to future reward compared to the immediate reward.

This challenge becomes more critical when tuning a set of parameters responsible for making a balance between the exploration and exploitation abilities of MHs since they directly control the behavior of MHs and influence the performance of AOS. One way to overcome this challenge is to use the parameter setting methods explained in Section 8, wherein the parameters of the MHs are tuned offline or controlled in an online manner. In almost all papers so far, these parameters are considered fixed during the search process, while they can be dynamically adjusted based on the characteristics of the search space. Accordingly, employing an online parameter control method (see Section 8) within AOS can be a promising future research direction.

Another challenge in AOS is related to the number of operators involved in AOS. Increasing the number of operators may reduce the performance of AOS. As the number of operators increases, more effort is required to perform AOS, and consequently a higher level of overhead is imposed on AOS. In addition, the performance of AOS may degrade if some operators do not perform well, and they will be selected fewer and fewer in the long term. The presence of such operators increases the computational overhead with no significant gain. One way to overcome this challenge is to use Adaptive Operator Management (AOM) that aims to manage operators during the search process by excluding inefficient operators and including other operators in AOS (Maturana et al., 2011). The use of AOM in AOS could be further investigated as a future research direction.

Another promising research direction could be employing AOS in multi-objective COPs where several objectives are evaluated simultaneously. An issue in this regard is how to assign a reward to an operator that improves one objective function but degrades the other objectives. In addition, integrating multiple rewards into a single credit value to be assigned to each operator is another issue to be addressed. One way to cope with these issues could be incorporating the crowding distance as well as the rank of the non-dominated fronts to calculate the reward/credit.

7.2. Learnable evolution model

Darwinian-type EAs (e.g., GA) are inspired from the principles of Darwin's theory of evolution. They apply usual genetic operators like mutation, crossover, and selection to generate new populations. These semi-random operators, which govern the evolution process, do not consider the experiences of individual solutions, the experience of an entire population, or a collection of populations. Therefore, new solutions are generated through a parallel trial-and-error process, so the lessons learned from the past generations are not used in these types of MHs. To overcome some of these inefficiencies, a new class of EAs has been proposed as Learnable Evolution Models (LEMs) (Michalski, 2000). In opposition to Darwinian-type EAs that contain a Darwinian evolution mode, LEM contains a learning mode wherein ML techniques are employed to generate new populations. In the learning mode, a learning system seeks reasons (rules) by particular ML techniques (e.g., AQ18 & AQ21 rule learning, C4.5 decision tree, etc.) on why certain solutions in a population (or a collection of past populations) are superior to others in performing a designated class of tasks.

Specifically, the learning mode of LEM consists of two processes (Michalski, 2000; Wojtusiak, Warden, & Herzog, 2011):

- **Hypothesis generation** – It determines a set of hypotheses that characterizes the differences between high-fitness and low-fitness solutions in recent or previous populations.
- **Hypothesis instantiation** – It generates new solutions on the basis of the learned hypotheses obtained in the hypothesis generation process. The learning mode thus produces new solutions not through semi-random Darwinian-type operations, but through a deliberate reasoning process involving the generation and instantiation of hypotheses about populations of solutions. The new populations are normally generated by injecting the extracted rules (i.e., *rule injection*) into the new solutions.

For the *hypothesis generation* process, two groups (sets) of individuals are selected from the population at each iteration: the *high performance group*, briefly *H-group*, and the *low performance group*, briefly *L-group*, based on the values of the fitness function. The collection of H-group and L-group solutions can be a subset of the population, or they can encompass the whole population

Table 8
Classification of papers studying LEM.

Ref.	Hypothesis gen.	Hypothesis inst.	Group frmt.	uni/duoLEM	MH	COP
Kaufman and Michalski (2000); Domanski et al. (2004)	AQ18	Sequence injection	FBF	duoLEM	EA	HEDP
Jourdan et al. (2005)	C4.5	Rule injection	FBF	uniLEM	EA	WDSDP
Wojtusiak et al. (2011)	AQ21	Rule injection	FBF	duoLEM	GA	VRP
Wojtusiak et al. (2012)	AQ21	Rule injection	FBF	duoLEM	MA	VRP
Wu and Tseng (2017)	Edge-intersection	Rule injection	PBF	duoLEM	GA	TSP
Jung et al. (2018)	C4.5	Rule injection	FBF	duoLEM	HS	WDSDP
Moradi (2019)	AQ18	Sequence injection	FBF	uniLEM	EA	VRP

(Michalski, 2000). The *H-group* and *L-group* can be formed using two methods (Michalski, 2000):

- **Fitness-based formation (FBF)** - In this method, the population is partitioned according to two fitness thresholds, *high fitness threshold* and *low fitness threshold*. These thresholds are presented as a percentage and determine the high and low portions of the total fitness value range in the population. These portions are then used to form the H-group and L-group of solutions. Indeed, the solutions whose fitness value is not worse than the high fitness threshold% of the best fitness value in the population form H-group, and those whose fitness value is better than the low fitness threshold% of the worst fitness value in the population form L-group. When using *fitness-based formation* method, the size of the H-group and L-group will vary from population to population since it depends on the range of solutions' fitness values at each iteration.
- **Population-based formation (PBF)** - In this method, the H-group and L-group are formed according to two thresholds expressed as the percentage of the number of solutions in the population. These thresholds are called *high population threshold* and *low population threshold*. The high population threshold% of the best solutions form H-group and the low population threshold% of the worst solutions form L-group.

The above two methods can be applied to the entire population as a *global* approach or they can be applied to different subsets of the population as a *local* approach. The idea behind a local approach is that different solutions of the population may carry different information, and there would be no global information that can characterize the whole population.

Once H-group and L-group are formed, a ML technique is employed to generate qualitative descriptions that discriminate between these two groups (Michalski, 2000). The description of an H-group represents a hypothesis that the landscape covered by H-group contains the solutions with higher fitness values compared to the landscape of L-group. Therefore, the H-group's qualitative description can be interpreted as the search direction toward the promising areas. LEMs account for such qualitative descriptions to guide the evolution process, rather than relying on semi-blind Darwinian-type evolutionary operators. Such an intelligent evolution in LEMs leads to the detection of the right directions for evolution; hence, making large improvements in the individuals' fitness values.

Two versions of LEM are introduced in the literature (Michalski, 2000): the *uniLEM* and *duoLEM*. In *uniLEM* version, the evolution process is solely conducted through the learning mode, while in *duoLEM* version both Darwinian and learning evolution processes are coupled.

7.2.1. Literature classification & analysis

This section classifies and analyzes the relevant papers wherein LEM has been used to solve COPs. Table 8 classifies the relevant studies based on different characteristics related to the design and implementation of LEM including *hypothesis generation*, *hypothesis instantiation*, *group formation*, *uniLEM/duoLEM* evolution version, the *MH* algorithm and the *COP* under study.

A set of insights can be extracted from Table 8. In terms of *hypothesis generation*, the learning mode of LEM can potentially employ different learning techniques that can generate descriptions discriminating between classes of individuals in a population (i.e., H-group versus L-group). If individuals are described by a vector of values (e.g., a permutation of a number of cities in TSP or a number of jobs in FSP), the rule learning techniques such as AQ learners (Domanski, Yashar, Kaufman, & Michalski, 2004; Kaufman & Michalski, 2000; Moradi, 2019; Wojtusiak et al., 2011; 2012), decision tree learners such as C4.5 (Jourdan, Corne, Savic, & Walters, 2005; Jung, Kang, & Kim, 2018), or ANN can be utilized. It can be stated from Table 8 that AQ learners, in which the learning systems employ some form of AQ algorithms, are particularly suitable for implementing LEM.

Regarding the *hypothesis instantiation*, it can be seen that the majority of the studies have used a *rule injection* mechanism, wherein H-group and L-group of individuals are investigated and their strengths and weaknesses are described in terms of rules. Taking TSP as an example, a partial sequence of cities that frequently appears in H-group individuals could be a rule, and such rules can be used to evolve the population by injecting frequent sequences to create new solutions. However, due to the complexity of the COP at hand in terms of prior knowledge on the structure of the solutions and descriptive characteristics of H-group and L-group of individuals, the learning program uses an abstract, rather than precise, specification of different individuals (Jourdan et al., 2005; Jung et al., 2018). Consequently, the learned rules are also referred to as an abstraction of individuals. The system is then able to instantiate these abstract rules in many ways to generate new solutions in further populations. The rule instantiation process must, however, follow the constraints of the COP at hand.

7.2.2. Discussion & future research directions

This section identifies the challenges of implementing LEM for solving optimization problems, particularly COPs. Throughout the discussion, future research directions are also elaborated.

Considering the group formation in Table 8, there is a requirement for implementing LEM as well as an important challenge when employing fitness-based formation to create H-group and L-group. Indeed, the fundamental assumption underlying LEM is that there is a method for evaluating the performance of individuals in evolving populations. Consequently the ability to determine the fitness value of an individual, or an approximation of this value, is a precondition for the LEM application. Therefore, LEM cannot be implemented for COPs for which defining or even approximating the fitness function is not possible.

A challenge related to the fitness-based formation that may happen in particular COPs is that in some evolutionary processes, the fitness function may not be constant throughout the entire process and change over time. It happens for particular COPs wherein the parameters change in a piece-wise manner depend-

ing on the level of the decision variables. A change in the fitness function may be gradual (i.e., fitness function drift) or abrupt (i.e., fitness function shift) (Michalski, 2000). Indeed, if the fitness function is changing during the evolution process, some high-fitness individuals (i.e., H-group) in a previous population may become low-fitness individuals (L-group) in a future population and vice versa.

One way to overcome this challenge is keeping a record of the L-groups determined in past populations and not only the current population. Therefore, a set of past L-groups plus L-group in the current population become the actual L-group supplied to the learning mode. The number of past L-groups to be taken into consideration is controlled by a parameter. It is worth mentioning that there is no significant need to store past H-groups because the current H-group inherently contains the best individuals until now. Generally, the formation of H-group and L-group from the current population in the evolution process ignores the history of evolution (Michalski, 2000).

An issue that may happen when using population-based formation is the possibility of an intersection between H-group and L-group. This issue should be resolved before hypothesis generation. Simple methods can be used for handling this issue such as ignoring inconsistent solutions, including inconsistent solutions in H-group or L-group, or employing statistical methods to solve the inconsistencies (Michalski, 2000).

Regarding the way of coupling LEM and Darwinian evolution modes, there are research papers (Domanski et al., 2004; Wu & Tseng, 2017) that place the learning mode before Darwinian Evolution mode. On the other hand, LEM can start with Darwinian Evolution mode (Wojtusiak, Warden, & Herzog, 2012), and then the two modes can alternate until the LEM termination criterion is met.

It has been regularly mentioned that involving discriminant descriptions provided by ML techniques in EAs significantly accelerates the search process toward promising solutions. Such accelerations have been shown as frequent quantum leaps of the fitness function that signify the discovery of the correct direction of the evolution process. However, such an evolutionary acceleration imposes a higher computational complexity on the search process. This extra complexity mostly depends on the ML technique used. Therefore, employing efficient ML techniques and a parallel implementation of the AQ algorithm can reduce the complexity. Among them, the latter can reduce the complexity from linear to logarithmic.

Although the initial results from employing LEM to solve COPs are promising, there are still plenty of unsolved questions that require further research. The first attempt for future research direction should be doing numerous systematic theoretical and experimental implementations of LEM to better understand the trade-off between LEM and Darwinian evolution modes.

Among COPs, LEM has been mostly implemented on routing (e.g., TSP, VRP) and design (e.g., HEDP, WDSDP) problems. Therefore, the second important future research direction for the interested researchers is implementing LEM on other COPs to figure out the strengths, weaknesses, and limitations of both LEM and Darwinian evolution modes and to identify the most appropriate areas for their application.

7.3. Neighbor generation

After selecting the most appropriate operator in Section 7.1, it is the time to generate the neighbors from the current solution(s) using the selected operator(s). One naive way to generate neighbors is to generate all possible neighbors and select the ones with the best OFV. Another way is to generate neighbors randomly. However, considering the computational time and the goal to lead the search process towards promising areas of the search space, both strategies might not be very efficient. To be as efficient as possible, ML techniques can be used to leverage the generation of good neighbors by extracting knowledge from the generated good solutions so far.

Indeed, using ML techniques, we can extract the common characteristics that are often present in good solutions during or before the search, and use this knowledge to generate new solutions with the same characteristics by fixing or prohibiting specific solution characteristics. In this way, we can lead the search towards promising areas of the search space and accelerate the process of finding a good solution. Usually, this knowledge is in the form of a set of rules or patterns that are discovered in good solutions (Arnold, Santana, Sörensen, & Vidal, 2021). This process is composed of two phases: *knowledge extraction* and *knowledge injection* (Arnold et al., 2021). In knowledge extraction, the common characteristics found in good solutions are extracted. Then, in knowledge injection, the extracted knowledge is used to generate the neighbors. When the knowledge appears as a set of patterns, the most frequent patterns of good solutions are injected into the new solutions to generate the neighbors.

The knowledge extraction phase can occur either *offline* or *online*. In offline extraction, knowledge is extracted from a set of training instances with the aim to generate good solutions for new instances. However, in online extraction, knowledge is extracted from good solutions obtained during the search process, while solving the problem instance. The most common ML techniques in neighbor generation are Apriori algorithms for ARs, RL, and DT.

7.3.1. Literature classification & analysis

Table 9 classifies the papers using ML techniques for neighbor generation based on different characteristics such as *learning* mechanism, the *ML technique* used to extract knowledge, the *MH* algorithm that conducts the search process, the *operator* to generate new neighborhood, the *COP* under study, and the *size* of the training set for studies that have used *offline* knowledge extraction.

As can be seen in Table 9, almost all reviewed papers have used the online manner to extract knowledge in the form of patterns. Indeed, when dealing with a new instance with unknown characteristics, the most efficient way to extract patterns is online. An advantage of online pattern extraction is avoiding the misleading patterns extracted in an offline manner that may not correctly represent the properties of the good solutions of the new problem instance; however, the computational overhead of online pattern extraction should not be ignored.

7.3.2. Discussion & future research directions

In this section, the requirements and challenges of applying ML techniques for neighbor generation are discussed. Then, some directions for future research are presented.

One of the most important requirements of using ARs is *data availability*. In fact, when extracting patterns, it is indispensable to have a sufficient pool of good solutions, since the accuracy of the extracted patterns directly depends on the availability of sufficient data. The higher the availability of data, the higher the precision and usefulness of the extracted patterns.

The first challenge of neighbor generation is to determine the stage of the search process at which the knowledge should be extracted to generate new neighbors. This challenge is twofold; first, in the beginning of the search process, the improvements are larger, and the set of good solutions frequently change, and no precise pattern can be extracted from the pool of good solutions. Therefore, the knowledge should be extracted after some iterations are passed. Second, if the knowledge is injected into the neighbor generation process in the earlier stages of the search process, it

Table 9
Classification of papers studying neighbor generation.

Ref.	Learning	ML tech.	MH	Operator	COP	Size
Santos, Ochi, Marinho, and Drummond (2006)	Online	Apriori	GA	XRO	VRP	–
Barbalho, Rosseti, Martins, and Plastino (2013); Ribeiro, Plastino, and Martins (2006)	Online	Apriori	GRASP	RRO	SPP	–
Zhou et al. (2016)	Online	RL	LS	RRO	GCP	–
Arnold et al. (2021)	Online	Apriori	GA, GLS	LSO	VRP	–
Thevenin and Zufferey (2019)	Online	Apriori	VNS	LSO	SMSP	–
Sadeg et al. (2019)	Online	Apriori	ABC	LSO	MAX-SAT	–
Arnold and Sörensen (2019)	Offline	DT, SVM, RF	GLS	LSO	VRP	192,000
Fairee, Khompatraporn, Prom-on, and Sirinaovakul (2019)	Online	RL	ABC	LSO	TSP	–
Wang, Pan, Li, and Yin (2020a)	Online	RL	LS	RRO	WIDP	–
Zhou, Hao, and Duval (2020)	Online	Apriori	EAs	MPO	QAP	–
Al-Duoli, Rabadi, Seck, and Handley (2018)	Online	Apriori	GRASP	LSO	VRP	–

prevents the MH to explore different areas of the search space, and it may cause a premature convergence to a local optimum.

The second challenge is related to the frequency at which the extracted pattern should be updated and injected to create new neighbors. Indeed, one may identify the patterns once and use them for neighbor generation throughout the search process or update the patterns frequently based on the characteristics of the newest good solutions. Pattern extraction may be a time-consuming process which increases the computational cost of the search process. Therefore, there is a trade-off between the accuracy of patterns based on the latest information from the search process and the computational overhead of pattern extraction process.

Another challenge arises when several pieces of patterns are available in good solutions and there might be plenty of possibilities to inject them into new solutions (i.e., separate or combined injection of patterns). There is no guarantee that the combination of pieces of patterns also provides good solutions (Arnold et al., 2021). In other words, although several single patterns may appear in a large number of good solutions, their combination does not necessarily generate better or even good solutions. For instance, in permutation-based representations, edges $A-B$ and $C-D$ may separately appear in good solutions, however; there is no guarantee that edges $A-B$ and $C-D$ simultaneously lead to a good solution.

Along with all the previously mentioned challenges, last but not least is making decision about the ratio by which new solutions are generated using the extracted knowledge. The level of such ratio affects the behavior of the MH in terms of its exploration and exploitation abilities. If almost all solutions are generated based on the previous knowledge, the MH tends to mostly exploit the currently observed area. On the other hand, if a small ratio of solutions are generated based on the previous knowledge, the algorithm has an opportunity to explore new areas of the search space. Therefore, the user has to consider the exploration and exploitation abilities of the algorithm while deciding about the ratio of the solutions to generate based on the extracted knowledge.

Considering Table 9, most of the studies that used ARs attempt to identify the characteristics of the good solutions. One research direction could be using ARs to identify the characteristics of bad solutions which have to be removed from the new solutions. Then, this knowledge could be used merely to avoid bad solutions, or it could be used besides the knowledge obtained from good solutions to complement the patterns of good solutions.

In addition, most of the papers in the literature have used the knowledge of good solutions to exploit the most promising areas of the search space, while the exploration aspect of the search process should be also taken into consideration. One way of doing that is extracting the rare patterns from the visited solutions and injecting them into new solutions to generate solutions far from the solutions visited so far.

8. Parameter setting

The success of any MH significantly depends on the values of its parameters (Talbi, 2009). As the parameters control the behavior of the algorithm during the search process, the values of parameters should be properly set to obtain the highest performance. Although there are several suggestions on the values of parameters for a similar group of problem instances in the literature, they are not necessarily the most appropriate settings when solving the problem instances at hand (Wolpert & Macready, 1997). Indeed, parameter setting is not a onetime task, and researchers need to set the algorithm's parameters whenever they solve new problem instances (Huang, Li, & Yao, 2019). Parameter setting, also known as *algorithm configuration* (Hoos, 2011), is divided into two categories, *parameter tuning* and *parameter control* (Eiben, Hinterding, & Michalewicz, 1999).

- **Parameter tuning** – Also known as *offline* parameter setting, identifies appropriate parameter levels before employing the algorithm to solve the problem instances at hand. In this case, the levels of parameters remain unchanged during the execution of the algorithm. Parameter tuning can be done using different methods such as *brute force* experiments (Phan, Ellis, Barca, & Dorin, 2019), *Design of Experiments* (DOE) (Talbi, 2009), *racing* procedures (Huang et al., 2019), and *meta-optimization* (Talbi, 2009).
- **Parameter control** – Also known as *online* parameter setting, focuses on adjusting the levels of parameters of an algorithm during its execution, rather than using initially fine-tuned parameters that remain unchanged during the whole execution. Parameter control methods have been developed based on the observation that tuning the parameters does not necessarily guarantee the optimal performance of a MH, since different settings of a parameter may be appropriate at different stages of the search process (Aleti, 2012). The reason is attributed to the non-stationary search space of optimization problems that results in a dynamic behavior of MHs, which should evolve regularly from a global search mode, requiring parameter values suited for the exploration of the search space, to a local search mode, requiring parameter values suitable for exploiting the neighborhood. Parameter control can be performed in three manners (Karafotias, Hoogendoorn, & Eiben, 2014); *deterministic* manner in which the levels of parameters are adjusted using given schedules (e.g., pre-defined iterations) without no feedback from the search process, *adaptive* manner in which the levels of parameters are adjusted using a feedback from the search process, where a credit is assigned to the parameters levels based on their performance, and *self-adaptive* manner in which the parameters levels are encoded into solution chromosomes and evolved during the search process.

Table 10
Classification of papers studying parameter setting.

Ref.	Tuning/ control	ML tech.	Credit ast.	Selection	MH	Parameter	COP	Size
Hong, Wang, Lin, and Lee (2002)	Control	RL	SCA	MCS	GA	Crossover & mutation rates	KP	–
Ramos et al. (2005)	Tuning	LogR	–	–	EA	Population size	TSP	25
Maturana and Riff (2007)	Control	RL	SCA	MCS	GA	Crossover & mutation rates	JSP	–
Caserta and Rico (2009)	Tuning	LR	–	–	GH	Population size	CLSP	4992
Zennaki and Ech-Cherif (2010)	Tuning	SVM	–	–	TS	Intensification rate	TSP, VRP	25000
Lessmann et al. (2011)	Tuning	SVM, LR	–	–	PSO	Learning rates	WDSDP	5284
Liao and Ting (2012)	Control	RL	SCA	MCS	EA	Mutation rate	PDP	–
Aleti, Moser, Meedeniya, and Grunske (2014)	Control	LR	–	–	EA	Crossover & mutation rates	QAP	–
André and Parpinelli (2014)	Control	RL	SCA	PMS	DE	Perturbation & Mutation rates	KP	–
Segredo et al. (2016)	Control	RL	SCA	MCS, PMS	MA	Mutation rate	AP	–
Benlic et al. (2017)	Control	RL	QLCA	SMS	BLS	Number & probability of perturbation	VSP	–
Chen, Yang, Li, and Wang (2020)	Control	RL	QLCA	EGS	GA	Crossover & mutation rates	FSP	–
Öztop, Tasgetiren, Kandiller, and Pan (2020)	Control	RL	QLCA	EGS	VNS	Acceptance & QL parameters	FSP	–

ML techniques can be employed in both parameter tuning and parameter control. In parameter tuning, ML techniques such as LogR (Ramos, Goldbarg, Goldbarg, & Neto, 2005), LR (Caserta & Rico, 2009), SVM (Lessmann, Caserta, & Arango, 2011), and ANN (Dobslaw, 2010) are used to predict the performance of a given set of parameters based on a set of training instances. In parameter control, ML techniques can be involved in *adaptive parameter control* to help control the parameters levels by using feedback information on the performance of the parameters levels during the search process. The integration of ML techniques into adaptive parameter control is similar to that of AOS (Section 7.1), where a feedback is used to adapt the parameters levels to the search space. It similarly involves four main steps (except Move Acceptance step): 1) performance criteria identification, 2) reward computation, 3) credit assignment, and 4) selection which have been explained in detail in Section 7.1.

8.1. Literature classification & analysis

This section aims at classifying, reviewing, and analyzing the studies on the use of ML techniques in the parameter setting of MHs. In this regard, Table 10 classifies the papers based on different characteristics including parameter *tuning/control*, the employed *ML technique*, *credit assignment* and *selection* methods, the *MH* for which the parameters are set, the *parameters* to set, the *COP* under study, and finally the *size* of the training set for the papers that have studied *parameter tuning*.

As can be seen from Table 10, parameter control has attracted much more attention compared to parameter tuning when solving COPs. Indeed, the main reason is that fixed values of the parameters do not necessarily guarantee the best performance of a MH during the whole search process. The underlying cause is the non-stationarity of the search space as explained at the beginning of this section. From a general point of view, the performance of a MH significantly depends on its capability to explore and exploit the promising areas of the search space, and the exploration and exploitation abilities of a MH depend on the levels of its parameters. Taking the GA as an example, the crossover and mutation rates should change depending on the performance of the algorithm and the properties of the search space. For example, once a promising solution is explored, that solution should be exploited carefully. Therefore, the mutation and crossover rates should be decreased and increased, respectively. On the other hand, when the algorithm gets stuck in local optima, the mutation rate can be increased to help the solution to escape from the local optima.

Based on Table 10, ML techniques based on RL have been mostly employed when controlling the parameters during the search process. The underlying reason is that a RL agent iteratively learns from interactions with its environment to take the actions that would maximize the reward. In the context of parameter setting, a list of different configurations can be defined as a set of actions and each time a configuration set provides better solutions, a reward is assigned to that set of configurations. Indeed, at the beginning of the search process, all possible configuration sets have the same probability to be selected. During the search process, these probabilities can change according to their success in creating better solutions (André & Parpinelli, 2014). It has been reported in the literature that in the earlier stages of the search process, the selection probability of each configuration set changes more often compared to the latter stages of the search process. It has been explained by the diversity loss that occurs during the search process (André & Parpinelli, 2014; Benlic et al., 2017; Liao & Ting, 2012; Segredo, Paechter, Hart, & González-Vila, 2016). In addition, there is evidence that the levels of exploration representative parameters (e.g., mutation rate in GA and DE, size of Tabu list in TS) change more frequently at the earlier stages of the search process when the MH is exploring the search space. On the other hand, the exploitation representative parameters get more attention in the latter stages of the search process when the MH needs to exploit promising solutions found so far (André & Parpinelli, 2014; Benlic et al., 2017; Liao & Ting, 2012; Zennaki & Ech-Cherif, 2010).

Another insight from Table 10 is that most of the papers have done parameter setting for population-based MHs. Population-based MHs possess both exploration and exploitation representative parameters, and a balance between these abilities should be well established. If not, the search process either gets stuck in local optima or performs a random search.

8.2. Discussion & future research directions

The first challenge when performing parameter setting is to decide whether to tune or control the parameters. Each mode of setting has its own advantages/disadvantages. There are experimental evidence revealing that the optimal parameter settings are different not only for different problem instances, but also for different stages of the search process of the same problem instance. In this situation, it is recommended to perform parameter control despite the computational overhead imposed on the search process. A big challenge related to the parameter control is the trade-off between exploration and exploitation to select the current best configuration(s) or search for new good ones. Once the configuration is changed during the search process, the new configuration should work for a certain number of iterations so that its performance can be evaluated. An extra parameter should be then defined to control when the configuration should be changed (i.e., the rate of configuration change). The rate of configuration change itself is another parameter that needs to be tuned or controlled during the search process. Therefore, the parameter control itself introduces a set of

other parameters (i.e., parameters of the parameter control mechanism) that need to be tuned or controlled, which results in the increase of the complexity of the parameter control.

Parameter control faces another challenge when dealing with continuous parameters, where an infinite number of values exist for each parameter. One way to deal with this challenge is considering the parameter setting as a separate optimization problem and the parameter levels as decision variables to be optimized. The other way is developing a self-adaptive parameter control mechanism. Another way studied in the literature is subdividing the levels of parameters into feasible intervals (Aleti, 2012). In this way, the interval borders are normally fixed and not manipulated by the search process. As a result, the number and the size of the intervals have to be determined by the user a priori that may jeopardize the accuracy of the interval as well as the efficiency of MHs. If the levels of a parameter are divided into several narrow intervals, the accuracy of the intervals would be higher, while the computational effort of selecting among the intervals significantly increases. Accordingly, there is a risk to find good intervals late, or there might be some intervals that are not selected at all. If the intervals are wide, different parameter values belonging to a single interval may lead to different performance of the MH, as wide intervals may encompass smaller intervals that behave differently. As a result, no unique performance behavior can be attributed to such wide intervals. To cope with this issue, the adaptive range parameter control method (Aleti, Moser, & Mostaghim, 2012) adapts intervals during the search process, and entropy-based adaptive range parameter control method (Aleti & Moser, 2013) clusters parameter values based on their performance. However, more research is indeed required to understand the impact of small changes in the values of continuous parameters on the performance of MHs.

Considering Table 10, one future research direction is applying parameter setting methods to other MHs such as single-solution based MHs. A first try can be developing a parameter setting mechanism to control the parameters of SA such as the cooling temperature, which is always a challenging problem for practitioners. Additionally, further investigation could be done on controlling parameters that have received little attention so far.

Another research direction that should be put in priority is implementing parameter setting on multi-objective COPs using ML techniques. It should be noted that one of the challenges of adaptive parameter control in multi-objective optimization problems is how to define the feedback as a single value such that it would be representative of the quality of a parameter value over multiple objective functions.

9. Cooperation

Different MHs with particular strengths and weaknesses working on the same problem instance produce different results. In this situation, using a framework enabling the use of different MHs in a cooperative way could result in an improved search process. The main motivation of developing cooperative MHs is to take advantage of the strengths of different MHs in one framework, to balance exploration and exploitation, and to direct the search towards promising regions of the search space (Martin, Ouelhadj, Beullens, & Ozcan, 2011). The interest in using such frameworks for solving COPs has risen due to their successful results (Martin et al., 2016; Talbi, 2002). The cooperation framework can be modeled as a multi-agent system in which the search process is performed by a set of agents that exchange information about states, models, entire sub-problems, solutions, or other search space characteristics (Blum & Roli, 2003).

In multi-agent based cooperative MHs, each agent could be a MH or a MH's component such as an operator, a search strategy, a solution, etc. that tries to solve the problem, while communicating with other agents (Silva, de Souza, Souza, & de Franca Filho, 2018). The cooperation could happen either at the *algorithm level* (between several MHs), wherein different MHs with specific characteristics cooperate to solve a COP, or it can happen at the *operator level* (inside a MH), wherein different operators cooperate when discovering different regions of the search space. The former belongs to the category of high-level integration of ML techniques into MHs while the latter falls in the low-level category of integration.

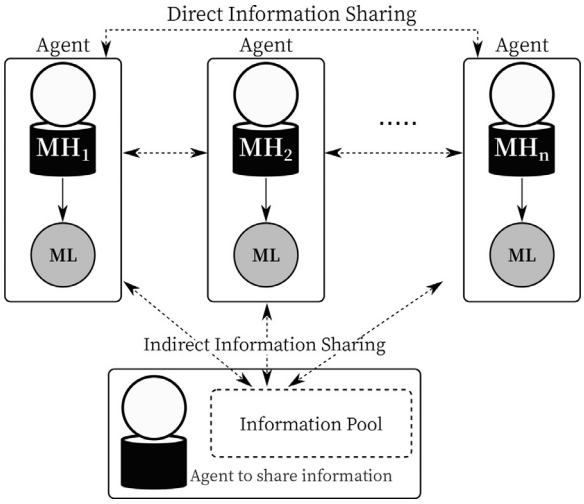

Fig. 5. Cooperation procedure.

ML techniques can help in designing intelligent cooperation frameworks by extracting the knowledge from the resolution of the problem instances by different agents (MHs). This knowledge is then incorporated into the framework that enables the framework to continuously adapt itself to the search space. In this way, ML techniques can improve the overall performance of the cooperation framework. The integration of ML techniques into a cooperation framework can happen in two learning levels: *inter-agent* and *intra-agent* levels. The former is to adapt the behavior of the overall framework to the search space, while the latter is to adapt the behavior of each agent to the search space.

Fig. 5 illustrates the process of cooperation between agents. Agents can cooperate *sequentially* or in *parallel* (Talbi, 2002). The cooperation between the agents can be *synchronous*, where the agents work in a parallel way and none of them waits for the results from other agents, or *asynchronous*, otherwise (Martin et al., 2016). As can be seen in Fig. 5, the exchange of information between the agents is *direct* (many-to-many), where each agent is allowed to communicate with any other agent and *indirect*, where agents are only allowed to use the information provided in a common pool (Martin et al., 2016). While cooperating, the agents can share *partial* or *complete* solutions to proceed the search process.

9.1. Literature classification & analysis

Table 11 classifies the papers studying cooperation for COPs based on different characteristics such as *cooperation level*, *parallel/sequential* mode of cooperation, *learning level*, the employed *ML technique*, *direct/indirect* information sharing, *solution sharing* type between the agents, the *MH* algorithms participated in the cooperation, and finally the *COP* under study. To the best of our knowledge, Table 11 reviews the most relevant papers, including the most recent papers in the literature that study the cooperation between MHs (or MHs' components) to solve COPs with the help of ML techniques.

Table 11
Classification of papers studying cooperation.

Ref.	Coop. level	Parl./ Seq.	Learning	ML tech.	(In)Direct	Sharing	MH	COP
Le Bouthillier, Crainic, and Kropf (2005)	Alg.	Parl.	Inter	Apriori	Ind.	Part.	TS	VRP
Meignan, Créput, and Koukam (2008, 2009); Meignan, Koukam, and Créput (2010)	Alg.	Parl.	Intra	RL	Di.	Comp.	EA	VRP, FLP
Cadenas, Garrido, and Muñoz (2009)	Alg.	Parl.	Inter	DT	Ind.	Comp.	GA, SA, TS	KP
Barbucha (2010a)	Opr.	Parl.	Inter	RL	Ind.	Comp.	LS	VRP
Silva, de Souza, Souza, and de Oliveira (2015)	Alg.	Parl.	Intra	LA	Ind.	Comp.	ILS	VRP
Lotfi and Acan (2016)	Alg.	Parl.	Inter	LA	Ind.	Comp.	GA, DE, SA, ACO, GD, TS	MSP
Martin et al. (2016)	Alg.	Parl.	Inter	Apriori	Di.	Part.	MHs	FSP, VRP
Sghir, Hao, Jaafar, and Ghédira (2015); Sghir, Jaafar, and Ghédira (2018)	Opr.	Seq.	Inter	RL	Ind.	Comp.	GA, LS	QAP, KP, GCP, WDP
Wang and Tang (2017)	Alg.	Seq.	Inter	*k*-means	Di.	Comp.	MA, LS	FSP
Silva, de Souza, Souza, and Bazzan (2019)	Alg.	Parl.	Intra	QL	Ind.	Comp.	ILS	VRP, PMSP
Karimi-Mamaghan et al. (2020a)	Alg.	Seq.	Inter	*k*-means	Di.	Comp.	DE, ILS	IPP
Karimi-Mamaghan et al. (2020b)	Alg.	Seq.	Inter	*k*-means	Di.	Comp.	GA, ILS	HLP

As can be seen in Table 11, the majority of the studies applied cooperation in a *parallel* manner. The main motivation has been an attempt to reduce the computational time of executing several MHs one after another (i.e., sequential cooperation). In Parallelism, the MHs are executed simultaneously, and consequently, it results in the reduction of the search process time. Indeed, the combination of cooperation and parallelism allows self-sufficient algorithms to run simultaneously while exchanging information about the search (Silva et al., 2018). This combination has attracted increasing attention in optimization, especially for solving COPs, since they have shown good results on different COPs (Karimi-Mamaghan, Mohammadi, Jula, Pirayesh, & Ahmadi, 2020a; Martin et al., 2016). Moreover, as mentioned by Talbi (2002) and Cotta, Talbi, and Alba (2005), the best results obtained for many optimization problems are achieved by the cooperative algorithms. In addition, there is greater access to parallel computing resources, which provides new possibilities for developing these techniques (Silva et al., 2018).

Regarding *synchronous* or *asynchronous* cooperation, the majority of the studies focus on the asynchronous way of cooperation. Indeed, in most of the times when MHs have been executed in parallel, their cooperation has been asynchronous, wherein none of MHs wait for the results of the others. Asynchronous cooperation carries fewer operational challenges and gives the opportunity to modify the cooperation framework easily. Using asynchronous cooperation, MHs can be added or dropped easily without any change to the overall framework. On the other hand, a synchronous cooperation faces more challenges. In synchronous cooperation, the agents must coordinate their actions in time. In other words, the agents are activated only when all agents are ready to act (Barbucha, 2010b). Indeed, although the agents work independently, the activation times of the agents depend on each other. Accordingly, there is a need to determine a synchronization point at which the agents announce their readiness and start to act. Hence, the time dependency of the agents may cause some agents to wait and consequently some processors stay idle for a period of time.

According to Table 11, the agents of a cooperative system could be MHs or MHs' components. The cooperation could happen either at an *algorithm level* or *operator level*. The difference between the cooperation at the operator level and AOS (Section 7.1) relies on the fact that in AOS, operators are selected one after another based on their history of performance, while in the cooperation framework, the operators share information while searching for solutions cooperatively.

The main ML techniques used in cooperative MHs are RL and Apriori algorithms for ARs. RL has been used to help the system adapt its behavior based on the experience it gains throughout the search process. RL is used in two levels, within the agents (intra-agent level) to adapt their behavior to the characteristics of the search space during the search process by modifying their components (selecting the operators) and/or in a higher level (inter-agent level) to adapt the application of the agents based on their performance compared to other agents. On the other hand, ARs are used to identify the common characteristics of good solutions. Then, this knowledge is shared among the agents in the form of partial solutions, which allows each agent to generate new solutions based on the identified patterns and guide the search toward promising regions.

9.2. Discussion & future research directions

This section aims at introducing the requirements and potential challenges when designing a cooperation framework of MHs. Next, a set of future research directions are provided.

The design and implementation of efficient cooperative MHs require sufficient apriori knowledge about different MHs. To take advantage of the strengths of different algorithms, which is the main motivation of developing cooperative algorithms, one needs to be aware of a broad spectrum of algorithms and have knowledge on their strengths and weaknesses. For instance, population-based MHs are powerful in exploration. On the other hand, single-solution based MHs are strong in exploitation. As can be seen in Table 11, studies with heterogeneous algorithms have incorporated both population-based and single-solution based MHs into their cooperation framework to take advantage of both exploration and exploitation abilities. As discussed earlier, the information between the agents can be shared in the form of partial solutions, where ARs can be used to generate these partial solutions. In this regard, a set of challenges in front of ARs for partial solution generation, which were elaborated in Section 7.3, also needs to be addressed in the design of cooperation frameworks.

Considering Table 11, in most of the studies, the agents (MHs) attempt to save and share good obtained solutions partially or completely. In this way, each MH would be aware of the promising regions exploited by other MHs. As a future research direction, sharing the bad solutions and their corresponding characteristics could be also useful to prevent MHs to explore non-promising regions. Indeed, the non-promising regions already visited by a MH could be prohibited to be explored and exploited again by other MHs.

Most of the reviewed papers in this section have used cooperative MHs to solve single-objective COPs, and there are only few papers that study cooperation in multi-objective COPs (Karimi-Mamaghan et al., 2020a; Karimi-Mamaghan, Mohammadi, Pirayesh, Karimi-Mamaghan, & Irani, 2020b). These two papers have

used *k*-means to link multi-objective population-based MHs with single-solution based MHs. Once the non-dominated solutions are obtained via the population-based MHs, *k*-means is used to cluster these solutions. Then, the representative of each cluster is given to the single-solution based MH to be more exploited. This cooperation has led to better non-dominated solutions in terms of both the quality and the computational time of the search process. In this regard, another future research direction could be extending the concept of cooperation to multi-objective COPs.

10. Conclusion and perspectives

In recent years, ML techniques have been extensively integrated into MHs for solving COPs, and promising results have been obtained in terms of solution quality, convergence rate, and robustness. This paper provides a comprehensive and technical review on the integration of ML techniques in the design of different elements of MHs for different purposes including algorithm selection, fitness evaluation, initialization, evolution, parameter setting, and cooperation. Throughout the manuscript, particular challenges are elaborated for each way of integrating. Regardless of the way of integration, there are also a set of common challenges in the use of ML techniques in MHs as follows:

- Whenever a ML technique is integrated into a MH, a set of additional parameters are introduced that need to be carefully tuned/controlled to obtain the highest performance. In almost all papers so far, these parameters are considered fixed during the search process. To cope with this challenge, we propose to dynamically adjust the additional parameters based on the characteristics of the integration.
- Scaling to larger problem instances is a challenge when training ML techniques on problem instances up to a particular size. To overcome this issue, one may attempt to use larger instances for training, while this could be very time-consuming and become a computational issue, except for very simple ML techniques and optimization problems.
- The higher the volume of data, the higher the performance of ML techniques. Data availability is another challenge when integrating ML techniques into MHs. Indeed, collecting or even generating enough data is a hard task. If even enough historical data is available, the way of sampling from historical data to appropriately mimic the behavior reflected in such data is another challenge (Bengio et al., 2021). One way of tackling the data availability challenge could be using *Few-Shot Learning* to train a model with a very small amount of training data (Wang, Yao, Kwok, & Ni, 2020b). By using prior knowledge of similar problem instances, few-shot learning can be rapidly generalized to new tasks containing only a few problem instances with supervised information.
- With the rapid development of new technologies, real-world problems are becoming increasingly complex, and with the new advances in digitalization, various real-time data are collected massively that cannot be processed by classical ML techniques. Such big data carries several issues that need to be taken into consideration (Emrouznejad, 2016). To cope with such big data, more advanced ML techniques such as deep learning can be integrated into MHs.

Apart from the above-mentioned ways to cope with common challenges that can be considered as promising future works, a set of common future research directions for the integration of ML techniques into MHs are elaborated as follows:

- Almost all the studies reviewed in this paper only deal with the integration of ML techniques into MHs with a single purpose, while the higher performance of MHs is expected to be achieved when ML techniques serve MHs for multiple purposes. Therefore, an interesting future research direction could be integrating ML techniques into MHs simultaneously for different purposes. For instance, implementing parameter control and adaptive operator selection simultaneously in a MH may increase the overall performance of the MH throughout the search process.
- With the development of supercomputers, it could be an interesting future research direction to explore the parallelism concept in the integration of ML techniques into MHs using GPU (Graphics Processing Units) and TPUs (Tensor Processing Units) accelerators (Alba, 2005; Cahon, Melab, & Talbi, 2004; Van Luong, Melab, & Talbi, 2011).
- The majority of papers in the literature use conventional ML techniques such as *k*-NN, *k*-means, SVMs, LR, etc. With the recent rapid development of ML techniques, more advanced and modern ML techniques such as deep reinforcement learning or transfer learning can be employed as a promising research direction. In this regard, when various ML techniques are available to be integrated into MHs for a particular purpose, the algorithm selection problem can be studied to select the most appropriate ML technique.
- An important issue in real-world optimization problems is the uncertainty of input data and particularly where the input data statistically contains various distributions. In this regard, a promising research direction could be using ML techniques such as clustering methods (e.g., *k*-means, SOM) to cluster the input data with the aim of discriminating the data with different distributions. These classes of data can be then used/integrated to solve the optimization problem at hand.
- The dynamicity of the input data is another issue that can be handled by ML techniques when solving dynamic optimization problems. Indeed, ML techniques can be used to monitor/predict the evolution of the input data, and once a new evolution is detected by ML techniques, the optimization variables are updated correspondingly.
- Last but not least, another future research direction is using the integration of ML techniques into MHs for either new COPs or other complex optimization problems such as multi-objective optimization, bi-level optimization, etc. This direction also opens other research questions that are worthy for further investigations.

Appendix A. List of COPs

In this section, a complete list of COPs that have been used throughout this paper is provided.

Table A.1
Exhaustive list and the abbreviation (Abv.) of the COPs studied by the articles reviewed in this paper.

COP	Abv.	Description
Assignment Problem	AP	Assigning a set of locations to a set of facilities such that the total assignment cost is minimized (Degroote et al., 2018).
Arc Routing Problem	ARP	Finding a set of tours with minimum cost in an undirected graph to serve the positive demand of edges by a set of available vehicles where each tour begins and ends at the depot (Consoli & Yao, 2014).
Assemble-To-Order Problem	ATOP	Determining an order based on which the parts and sub-assemblies are made but the final assembly is delayed until the customer orders are received such that the total production cost is minimized (Horng et al., 2013).
Berth Allocation Problem	BAP	Allocating the berthing position and berthing time to the incoming vessels to perform loading/unloading activities such that the total vessels' waiting time or the early or delayed departures is minimized (Wawrzyniak et al., 2020).
Bin-Packing Problem	BPP	Placing N items in a number of capacitated knapsacks so that the total number of knapsacks used to pack all items is minimized (Burke et al., 2011).
Capacitated Lot Sizing Problem	CLSP	Planning the lot size of a set of different items over a planning horizon under production capacity constraints such that the total production, setup, and inventory cost is minimized (Caserta & Rico, 2009).
Facility Location Problem	FLP	Locating a number of facilities in a set of potential locations to serve a set of customers with predefined locations such that the total opening and transportation cost is minimized (Meignan et al., 2010).
Flowshop Scheduling Problem	FSP	Finding the order of processing N jobs on M machines with the same sequence such that the makespan, total tardiness, or total lag between the jobs is minimized (Pavelski et al., 2018b).
Graph Coloring Problem	GCP	Finding the minimum number of colors for coloring the vertices of a graph such that no two adjacent vertices have the same color or finding the maximum sub-graph of a graph to be colored with k colors such that no two adjacent vertices have the same color (Mostafaie, Khiyabani, & Navimipour, 2020; Zhou et al., 2016).
Heat Exchanger Design Problem	HEDP	Designing the structure of tubes under technical and environmental constraints including the size of the exchanger and air temperature such that the heat transfer rate is maximized (Domanski et al., 2004).
Hub Location Problem	HLP	Locating a set of hubs and allocating a set of origin and destination nodes to the located hubs for transferring the origin-destination flows through the hubs such that the total hub opening and transportation costs is minimized (Mohammadi et al., 2019).
Inspection Planning Problem	IPP	Determining which quality characteristics of a product should be inspected at which stage of the production process such that the total inspection cost is minimized (Karimi-Mamaghan et al., 2020a).
Job-Shop Scheduling	JSP	scheduling the processing of N jobs consists of a sequence of tasks that need to be performed in a given order on specific subsets of M machines such that the makespan or total tardiness is minimized (Nasiri et al., 2019).
Knapsack problem	KP	Placing a number of items with specific values and dimension in M capacitated knapsacks such that the total value of the knapsacks is maximized (Cadenas et al., 2009).
Location-Routing Problem	LRP	Locating a number of facilities in a set of potential locations, assigning customers with predefined demand to the located facilities, and finding the routes from located facilities to customers such that the total cost of opening facilities, cost of vehicles and transportation cost is minimized (Zhalechian et al., 2016).
Maximum Satisfiability Problem	MAX-SAT	Finding an assignment of the truth values to the variables of a Boolean formula such that the number of satisfied clauses is maximized (Miranda et al., 2018).
Mixed-model Assembly Line Sequencing Problem	MASP	Determining the optimal production planning of multiple products along a single assembly line while maintaining the least possible inventories (Mosadegh et al., 2020).
Multiprocessor Scheduling Problem	MSP	Given a directed graph representing a parallel program, where the vertices represent the tasks and the edges represent the communication cost and task dependencies, scheduling the tasks on a network of processors under task precedence constraints such that the makespan is minimized (Lotfi & Acan, 2016).
Nurse Rostering Problem	NRP	Assigning nurses to working shifts under a set of constraints including nurse preferences, time restrictions, labor legislation, and hospital standards such that the total cost of the hospital is minimized or the nurses' preferences are maximized (Gretsista & Burke, 2017).
Orienteering Problem	OP	Selecting a set of nodes from available nodes with specific score and determining the shortest path between the selected nodes such that the total score of the visited nodes is maximized (Gunawan et al., 2018).
Pickup and Delivery Problem	PDP	Designing a set of routes to collect commodities from specific origins and deliver them to their specific destinations using capacitated vehicles such that the total cost is minimized (Liao & Ting, 2012).
Parallel Machine Scheduling Problem	PMSP	Scheduling the processing of N jobs on M identical parallel machines such that the total makespan is minimized (Silva et al., 2019).
Project Scheduling Problem	PSP	Assigning limited resources (employees) to activities with predefined duration and resource requirements under activity precedence relations such that the lateness or the total tardiness is minimized (Pathak et al., 2008).
Personnel Scheduling Problem	PSSP	Assigning personnel to working shifts under a set of constraints (e.g., shift time) such that the total cost is minimized (Burke et al., 2011).
Quadratic Assignment Problem	QAP	A special case of Assignment Problem with quadratic objective function (Pitzer et al., 2013).
Boolean Satisfiability Problem	SAT	Determining if the variables of a Boolean formula can be substituted by *True* and *False* values such that the Boolean formula turns out to be *TRUE* (Satisfiable) or not (Maturana & Saubion, 2008).
Single-Machine Scheduling Problem	SMSP	A special case of Flow-Shop scheduling problem where there is only a single machine (Thevenin & Zufferey, 2019).
Set Packing Problem	SPP	Determining/picking a subset of elements from a bigger set such that the total value of picking is maximized (Ribeiro et al., 2006).
Job Sequencing and Tool Switching Problem	SSP	Sequencing N jobs, each of which requires a predefined set of tools, on a single flexible machine and assigning tools to a capacitated machine such that the number of tool switches is minimized (Ahmadi et al., 2018).
Transmission Expansion Planning Problem	TEPP	Planning new transmission facilities as an expansion to the existing transmission network to satisfy demand without load interruption under technical constraints such that the total investment and operational cost is minimized (da Silva et al., 2016).
Traveling Salesman Problem	TSP	Finding a tour in a complete weighted graph that goes through all vertices only once and returns to the starting vertex such that the tour cost is minimized (Kanda et al., 2016).
Timetabling Problem	TTP	Allocating predefined resources (teachers and rooms) to events (classes) such that there is no conflict between any two events and a set of objectives are satisfied (de la Rosa-Rivera et al., 2021).
Vehicle Routing Problem	VRP	Finding a set of undirected edges in a graph by which the demand of customers located in the vertices is satisfied by a set of vehicles that visit each customer exactly once. Vehicles start and end their route at the depot such that the total transportation cost is minimized (Gutierrez-Rodríguez et al., 2019).
Vertex Separator Problem	VSP	Partitioning the graph into three non-empty subsets A, B, and C such that there is no edge between A and B, and $\|C\|$ is minimized subject to a bound on $max\{\|A\|, \|B\|\}$ (Benlic et al., 2017).
Winner Determination Problem	WDP	Considering a set of bids in a combinatorial auction, assigning items to bidders such that the auctioneer's revenue is maximized (Sghir et al., 2018).
Water Distribution System Design Problem	WDSDP	Determining the location, size, and capacity of water system components including pipes and pumps such that the system's reliability (ability to supply adequate water with acceptable pressure and quality to customers) is maximized (Lessmann et al., 2011).
Weighted Independent Domination Problem	WIDP	Determining a pairwise non-adjacent subset D of V of a graph $G = (V, E)$ such that every vertex not in D is adjacent to at least one vertex in D (Wang et al., 2020a).
Workforce Scheduling and Routing Problem	WSRP	Assigning workforce to the activities needed to be performed at different locations where the workforce need to travel between locations to perform the activities such that the employees travel time or hiring cost is minimized (López-Santana et al., 2018).

References

Abdel-Basset, M., Manogaran, G., Rashad, H., & Zaied, A. N. H. (2018). A comprehensive review of quadratic assignment problem: Variants, hybrids and applications. *Journal of Ambient Intelligence and Humanized Computing*, 1–24.

Ahmadi, E., Goldengorin, B., Süer, G. A., & Mosadegh, H. (2018). A hybrid method of 2-tsp and novel learning-based ga for job sequencing and tool switching problem. *Applied Soft Computing, 65*, 214–229.

Al-Duoli, F., Rabadi, G., Seck, M., & Handley, H. A. (2018). Hybridizing meta-raps with machine learning algorithms. In *2018 ieee technology and engineering management conference (TEMSCON)* (pp. 1–6). IEEE.

Alba, E. (2005). *Parallel metaheuristics: A new class of algorithms*: 47. John Wiley & Sons.

Aleti, A. (2012). An adaptive approach to controlling parameters of evolutionary algorithms. *Swinburne University of Technology*.

Aleti, A., & Moser, I. (2013). Entropy-based adaptive range parameter control for evolutionary algorithms. In *Proceedings of the 15th annual conference on genetic and evolutionary computation* (pp. 1501–1508).

Aleti, A., & Moser, I. (2016). A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Computing Surveys (CSUR), 49*(3), 1–35.

Aleti, A., Moser, I., Meedeniya, I., & Grunske, L. (2014). Choosing the appropriate forecasting model for predictive parameter control. *Evolutionary computation, 22*(2), 319–349.

Aleti, A., Moser, I., & Mostaghim, S. (2012). Adaptive range parameter control. In *2012 IEEE congress on evolutionary computation* (pp. 1–8). IEEE.

Ali, I. M., Essam, D., & Kasmarik, K. (2019). New designs of k-means clustering and crossover operator for solving traveling salesman problems using evolutionary algorithms. In *Proceedings of the 11th international joint conference on computational intelligence* (pp. 123–130).

Ali, I. M., Essam, D., & Kasmarik, K. (2020). A novel design of differential evolution for solving discrete traveling salesman problems. *Swarm and Evolutionary Computation, 52*, 100607.

Alipour, M. M., Razavi, S. N., Derakhshi, M. R. F., & Balafar, M. A. (2018). A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. *Neural Computing and Applications, 30*(9), 2935–2951.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345–378.

André, L., & Parpinelli, R. S. (2014). A binary differential evolution with adaptive parameters applied to the multiple knapsack problem. In *Mexican international conference on artificial intelligence* (pp. 61–71). Springer.

Angel, E., & Zissimopoulos, V. (2002). On the hardness of the quadratic assignment problem with metaheuristics. *Journal of Heuristics, 8*(4), 399–414.

Armstrong, W., Christen, P., McCreath, E., & Rendell, A. P. (2006). Dynamic algorithm selection using reinforcement learning. In *2006 international workshop on integrating ai and data mining* (pp. 18–25). IEEE.

Arnold, F., Santana, Í., Sörensen, K., & Vidal, T. (2021). Pils: Exploring high-order neighborhoods by pattern mining and injection. *Pattern Recognition*, 107957.

Arnold, F., & Sörensen, K. (2019). What makes a VRP solution good? the generation of problem-specific knowledge for heuristics. *Computers & Operations Research, 106*, 280–288.

Arora, D., & Agarwal, G. (2016). Meta-heuristic approaches for flowshop scheduling problems: A review. *International Journal of Advanced Operations Management, 8*(1), 1–16.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In *2007 ieee congress on evolutionary computation* (pp. 4661–4667). Ieee.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*(2–3), 235–256.

Barbalho, H., Rosseti, I., Martins, S. L., & Plastino, A. (2013). A hybrid data mining grasp with path-relinking. *Computers & Operations Research, 40*(12), 3159–3173.

Barbucha, D. (2010a). Cooperative solution to the vehicle routing problem. In *Kes international symposium on agent and multi-agent systems: Technologies and applications* (pp. 180–189). Springer.

Barbucha, D. (2010b). Synchronous vs. asynchronous cooperative approach to solving the vehicle routing problem. In *International conference on computational collective intelligence* (pp. 403–412). Springer.

Bartz-Beielstein, T., & Zaefferer, M. (2017). Model-based methods for continuous and discrete global optimization. *Applied Soft Computing, 55*, 154–167.

Beham, A., Affenzeller, M., & Wagner, S. (2017). Instance-based algorithm selection on quadratic assignment problem landscapes. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 1471–1478).

Bengio, Y., Frejinger, E., Lodi, A., Patel, R., & Sankaranarayanan, S. (2020). A learning-based algorithm to quickly compute good primal solutions for stochastic integer programs. In *International conference on integration of constraint programming, artificial intelligence, and operations research* (pp. 99–111). Springer.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: Amethodological tour d'horizon. *European Journal of Operational Research, 290*(2), 405–421.

Benlic, U., Epitropakis, M. G., & Burke, E. K. (2017). A hybrid breakout local search and reinforcement learning approach to the vertex separator problem. *European Journal of Operational Research, 261*(3), 803–818.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR), 35*(3), 268–308.

Bossek, J., Grimme, C., Meisel, S., Rudolph, G., & Trautmann, H. (2018). Local search effects in bi-objective orienteering. In *Proceedings of the genetic and evolutionary computation conference* (pp. 585–592).

Bossek, J., & Trautmann, H. (2016). Evolving instances for maximizing performance differences of state-of-the-art inexact TSP solvers. In *International conference on learning and intelligent optimization* (pp. 48–59). Springer.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society, 64*(12), 1695–1724.

Burke, E. K., Gendreau, M., Ochoa, G., & Walker, J. D. (2011). Adaptive iterated local search for cross-domain optimisation. In *Proceedings of the 13th annual conference on genetic and evolutionary computation* (pp. 1987–1994).

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2019). A classification of hyper-heuristic approaches: revisited. In *Handbook of metaheuristics* (pp. 453–477). Springer.

Burke, E. K., Petrovic, S., & Qu, R. (2006). Case-based heuristic selection for timetabling problems. *Journal of Scheduling, 9*(2), 115–132.

Buzdalova, A., Kononov, V., & Buzdalov, M. (2014). Selecting evolutionary operators using reinforcement learning: Initial explorations. In *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation* (pp. 1033–1036).

Cadenas, J. M., Garrido, M. C., & Muñoz, E. (2009). Using machine learning in a cooperative hybrid parallel strategy of metaheuristics. *Information Sciences, 179*(19), 3255–3267.

Cahon, S., Melab, N., & Talbi, E.-G. (2004). Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics, 10*(3), 357–380.

Calvet, L., de Armas, J., Masip, D., & Juan, A. A. (2017). Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Mathematics, 15*(1), 261–280.

Caserta, M., & Rico, E. Q. (2009). A cross entropy-lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Computers & Operations Research, 36*(2), 530–548.

Catteeuw, D., Drugan, M., & Manderick, B. (2014). 'guided'restarts hill-climbing. In *In search of synergies between reinforcement learning and evolutionary computation, workshop at the 13th international conference on parallel problem solving from nature. ed. by Madalina M. Drugan and Bernard Manderick. Ljubljana, Slovenia* (pp. 1–4).

Chang, Y.-C. (2017). Using k-means clustering to improve the efficiency of ant colony optimization for the traveling salesman problem. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 379–384). IEEE.

Chen, R., Yang, B., Li, S., & Wang, S. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & Industrial Engineering, 149*, 106778.

Chen, Y., Cowling, P. I., Polack, F. A., & Mourdjis, P. J. (2016). *A multi-arm bandit neighbourhood search for routing and scheduling problems* (pp. 2–32). The University of York.

Cheng, C.-Y., Pourhejazy, P., Ying, K.-C., & Lin, C.-F. (2021). Unsupervised learning-based artificial bee colony for minimizing non-value-adding operations. *Applied Soft Computing*, 107280.

Choong, S. S., Wong, L.-P., & Lim, C. P. (2018). Automatic design of hyper-heuristic based on reinforcement learning. *Information Sciences, 436*, 89–107.

Choong, S. S., Wong, L.-P., & Lim, C. P. (2019). An artificial bee colony algorithm with a modified choice function for the traveling salesman problem. *Swarm and Evolutionary Computation, 44*, 622–635.

Chu, X., Cai, F., Cui, C., Hu, M., Li, L., & Qin, Q. (2019). Adaptive recommendation model using meta-learning for population-based algorithms. *Information Sciences, 476*, 192–210.

Consoli, P., & Yao, X. (2014). Diversity-driven selection of multiple crossover operators for the capacitated arc routing problem. In *European conference on evolutionary computation in combinatorial optimization* (pp. 97–108). Springer.

Corne, D., Dhaenens, C., & Jourdan, L. (2012). Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research, 221*(3), 469–479.

Cotta, C., Talbi, E., & Alba, E. (2005). Parallel hybrid metaheuristics. *Parallel Metaheuristics: A New Class of Algorithms, 47*, 347.

Dantas, A., & Pozo, A. (2020). On the use of fitness landscape features in meta-learning based algorithm selection for the quadratic assignment problem. *Theoretical Computer Science, 805*, 62–75.

Dantas, A. L., & Pozo, A. T. R. (2018). A meta-learning algorithm selection approach for the quadratic assignment problem. In *2018 IEEE congress on evolutionary computation (CEC)* (pp. 1–8). IEEE.

De Lima, F. C., De Melo, J. D., & Neto, A. D. D. (2008). Using the q-learning algorithm in the constructive phase of the grasp and reactive grasp metaheuristics. In *2008 ieee international joint conference on neural networks (IEEE world congress on computational intelligence)* (pp. 4169–4176). IEEE.

Degroote, H., Bischl, B., Kotthoff, L., & De Causmaecker, P. (2016). Reinforcement learning for automatic online algorithm selection-an empirical study. *ITAT 2016 Proceedings, 1649*, 93–101.

Degroote, H., González-Velarde, J. L., & De Causmaecker, P. (2018). Applying algorithm selection–a case study for the generalised assignment problem. *Electronic Notes in Discrete Mathematics, 69*, 205–212.

Deng, Y., Liu, Y., & Zhou, D. (2015). An improved genetic algorithm with initial population strategy for symmetric TSP. *Mathematical Problems in Engineering, 2015*, 1–6.

Dhaenens, C., & Jourdan, L. (2016). *Metaheuristics for big data*. Wiley Online Library.

Di Tollo, G., Lardeux, F., Maturana, J., & Saubion, F. (2015). An experimental study of adaptive control for evolutionary algorithms. *Applied Soft Computing, 35*, 359–372.
Díaz-Manríquez, A., Toscano, G., & Coello, C. A. C. (2017). Comparison of metamodeling techniques in evolutionary algorithms. *Soft Computing, 21*(19), 5647–5663.
Díaz-Parra, O., Ruiz-Vanoye, J. A., & Zavala-Díaz, J. C. (2010). Population pre-selection operators used for generating a non-random initial population to solve vehicle routing problem with time windows. *Scientific Research and Essays, 5*(22), 3529–3537.
Dobslaw, F. (2010). A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In *International conference on computer mathematics and natural computing* (pp. 1–4). WASET.
Domanski, P. A., Yashar, D., Kaufman, K. A., & Michalski, R. S. (2004). An optimized design of finned-tube evaporators using the learnable evolution model. *Hvac&R Research, 10*(2), 201–211.
Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical computer science, 344*(2–3), 243–278.
Drake, J. H., Kheiri, A., Özcan, E., & Burke, E. K. (2019). Recent advances in selection hyper-heuristics. *European Journal of Operational Research, 285*(2), 405–428.
Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics, 104*(1), 86–92.
Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation, 3*(2), 124–141.
Emrouznejad, A. (2016). *Big data optimization: Recent developments and challenges*: 18. Springer.
Fairee, S., Khompatraporn, C., Prom-on, S., & Sirinaovakul, B. (2019). Combinatorial artificial bee colony optimization with reinforcement learning updating for travelling salesman problem. In *2019 16th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ecti-con)* (pp. 93–96). IEEE.
Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization, 6*(2), 109–133.
Fialho, Á. (2010). *Adaptive operator selection for optimization*. Université Paris Sud - Paris XI Ph.D. thesis..
Fialho, Á., Da Costa, L., Schoenauer, M., & Sebag, M. (2008). Extreme value based adaptive operator selection. In *International conference on parallel problem solving from nature* (pp. 175–184). Springer.
Francesca, G., Pellegrini, P., Stützle, T., & Birattari, M. (2011). Off-line and on-line tuning: a study on operator selection for a memetic algorithm applied to the qap. In *European conference on evolutionary computation in combinatorial optimization* (pp. 203–214). Springer.
Gagliolo, M., & Schmidhuber, J. (2010). Algorithm selection as a bandit problem with unbounded losses. In *International conference on learning and intelligent optimization* (pp. 82–96). Springer.
Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2020). Optimization problems for machine learning: A survey. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2020.08.045.
Gao, S., Wang, Y., Cheng, J., Inazumi, Y., & Tang, Z. (2016). Ant colony optimization with clustering for solving the dynamic location routing problem. *Applied Mathematics and Computation, 285*, 149–173.
Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *simulation, 76*(2), 60–68.
(2010). In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of metaheuristics*. Springer US.
Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093–2229). Springer.
Gocken, T., & Yaktubay, M. (2019). Comparison of different clustering algorithms via genetic algorithm for VRPTW. *International Journal of Simulation Modeling, 18*(4), 574–585.
González-Juarez, D., & Andrés-Pérez, E. (2019). Study of the influence of the initial a priori training dataset size in the efficiency and convergence of surrogate-based evolutionary optimization. In *Evolutionary and deterministic methods for design optimization and control with applications to industrial and societal problems* (pp. 181–194). Springer.
Gretsista, A., & Burke, E. K. (2017). An iterated local search framework with adaptive operator selection for nurse rostering. In *International conference on learning and intelligent optimization* (pp. 93–108). Springer.
Gunawan, A., Lau, H. C., & Lu, K. (2018). Adopt: Combining parameter tuning and adaptive operator ordering for solving a class of orienteering problems. *Computers & Industrial Engineering, 121*, 82–96.
Gutierrez-Rodríguez, A. E., Conant-Pablos, S. E., Ortiz-Bayliss, J. C., & Terashima–Marín, H. (2019). Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning. *Expert Systems with Applications, 118*, 470–481.
Haghighi, M. M. S., Zahedi, M. H., & Ghazizadeh, M. (2010). A multi level priority clustering ga based approach for solving heterogeneous vehicle routing problem (pcgvrp). In *Innovations and advances in computer sciences and engineering* (pp. 331–335). Springer.
Handoko, S. D., Nguyen, D. T., Yuan, Z., & Lau, H. C. (2014). Reinforcement learning for adaptive operator selection in memetic search applied to quadratic assignment problem. In *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation* (pp. 193–194).
Hao, J.-h., & Liu, M. (2014). A surrogate modelling approach combined with differential evolution for solving bottleneck stage scheduling problems. In *2014 world automation congress (WAC)* (pp. 120–124). IEEE.
Hao, J.-h., Liu, M., Lin, J.-h., & Wu, C. (2016). A hybrid differential evolution approach based on surrogate modelling for scheduling bottleneck stages. *Computers & Operations Research, 66*, 215–224.
Hassanat, A. B., Prasath, V., Abbadi, M. A., Abu-Qdari, S. A., & Faris, H. (2018). An improved genetic algorithm with a new initialization mechanism based on regression techniques. *Information, 9*(7), 167.
van Hemert, J. I. (2006). Evolving combinatorial problem instances that are difficult to solve. *Evolutionary Computation, 14*(4), 433–462.
Hertz, A., & de Werra, D. (1990). The tabu search metaheuristic: How we used it. *Annals of Mathematics and Artificial Intelligence, 1*(1–4), 111–121.
Holland, J. H., et al. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.
Hong, T.-P., Wang, H.-S., Lin, W.-Y., & Lee, W.-Y. (2002). Evolution of appropriate crossover and mutation operators in a genetic process. *Applied Intelligence, 16*(1), 7–17.
Hoos, H. H. (2011). Automated algorithm configuration and parameter tuning. In *Autonomous search* (pp. 37–71). Springer.
Hoos, H. H., Peitl, T., Slivovsky, F., & Szeider, S. (2018). Portfolio-based algorithm selection for circuit QBFS. In *International conference on principles and practice of constraint programming* (pp. 195–209). Springer.
Horng, S.-C., Lin, S.-Y., Lee, L. H., & Chen, C.-H. (2013). Memetic algorithm for real-time combinatorial stochastic simulation optimization problems with performance analysis. *IEEE Transactions on Cybernetics, 43*(5), 1495–1509.
Huang, C., Li, Y., & Yao, X. (2019). A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation, 24*(2), 201–216.
Hutter, F., Hamadi, Y., Hoos, H. H., & Leyton-Brown, K. (2006). Performance prediction and automated tuning of randomized and parametric algorithms. In *International conference on principles and practice of constraint programming* (pp. 213–228). Springer.
Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing, 9*(1), 3–12.
Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation, 1*(2), 61–70.
Jin, Y., & Sendhoff, B. (2004). Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Genetic and evolutionary computation conference* (pp. 688–699). Springer.
Johnson, D. S., Papadimitriou, C. H., & Yannakakis, M. (1988). How easy is local search? *Journal of Computer and System Sciences, 37*(1), 79–100.
Jourdan, L., Corne, D., Savic, D., & Walters, G. (2005). Preliminary investigation of the 'learnable evolution model'for faster/better multiobjective water systems design. In *International conference on evolutionary multi-criterion optimization* (pp. 841–855). Springer.
Jourdan, L., Dhaenens, C., & Talbi, E.-G. (2006). Using datamining techniques to help metaheuristics: A short survey. In *International workshop on hybrid metaheuristics* (pp. 57–69). Springer.
Jung, D., Kang, D., & Kim, J. H. (2018). Development of a hybrid harmony search for water distribution system design. *KSCE Journal of Civil Engineering, 22*(4), 1506–1514.
Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., & Sellmann, M. (2011). Algorithm selection and scheduling. In *International conference on principles and practice of constraint programming* (pp. 454–469). Springer.
Kanda, J., Carvalho, A., Hruschka, E., & Soares, C. (2011a). Selection of algorithms to solve traveling salesman problems using meta-learning 1. *International Journal of Hybrid Intelligent Systems, 8*(3), 117–128.
Kanda, J., de Carvalho, A., Hruschka, E., Soares, C., & Brazdil, P. (2016). Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features. *Neurocomputing, 205*, 393–406.
Kanda, J., Soares, C., Hruschka, E., & De Carvalho, A. (2012). A meta-learning approach to select meta-heuristics for the traveling salesman problem using MLP-based label ranking. In *International conference on neural information processing* (pp. 488–495). Springer.
Kanda, J. Y., de Carvalho, A. C., Hruschka, E. R., & Soares, C. (2011b). Using meta-learning to recommend meta-heuristics for the traveling salesman problem. In *2011 10th international conference on machine learning and applications and workshops: 1* (pp. 346–351). IEEE.
Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report*. Citeseer.
Karafotias, G., Hoogendoorn, M., & Eiben, Á. E. (2014). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation, 19*(2), 167–187.
Karimi-Mamaghan, M., Mohammadi, M., Jula, P., Pirayesh, A., & Ahmadi, H. (2020a). A learning-based metaheuristic for a multi-objective agile inspection planning model under uncertainty. *European Journal of Operational Research, 285*(2), 513–537.
Karimi-Mamaghan, M., Mohammadi, M., Pirayesh, A., Karimi-Mamaghan, A. M., & Irani, H. (2020b). Hub-and-spoke network design under congestion: A learning based metaheuristic. *Transportation Research Part E: Logistics and Transportation Review, 142*, 102069.
Kaufman, K. A., & Michalski, R. S. (2000). Applying learnable evolution model to heat exchanger design. In *AAAI/IAAI* (pp. 1014–1019).
Kennedy, J. (2006). Swarm intelligence. In *Handbook of nature-inspired and innovative computing* (pp. 187–219). Springer.
Kerschke, P., Hoos, H. H., Neumann, F., & Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary Computation, 27*(1), 3–45.
Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial op-

timization algorithms over graphs. In *Advances in neural information processing systems* (pp. 6348–6358).
Kiranyaz, S., Ince, T., & Gabbouj, M. (2014). *Multidimensional particle swarm optimization for machine learning and pattern recognition*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-37846-1.
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.
Knowles, J. (2006). Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation, 10*(1), 50–66.
Koç, Ç., Bektaş, T., Jabali, O., & Laporte, G. (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research, 249*(1), 1–21.
Korte, B., Vygen, J., Korte, B., & Vygen, J. (2012). *Combinatorial optimization*: 2. Springer.
Kotthoff, L. (2014). Algorithm selection for combinatorial search problems: A survey. *AI Magazine, 35*(3), 48–60.
Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data mining and constraint programming* (pp. 149–190). Springer.
Le Bouthillier, A., Crainic, T. G., & Kropf, P. (2005). A guided cooperative search for the vehicle routing problem with time windows. *IEEE Intelligent Systems, 20*(4), 36–42.
de León, A. D., Lalla-Ruiz, E., Melián-Batista, B., & Moreno-Vega, J. M. (2017a). A machine learning-based system for berth scheduling at bulk terminals. *Expert Systems with Applications, 87*, 170–182.
de León, A. D., Lalla-Ruiz, E., Melián-Batista, B., & Moreno-Vega, J. M. (2017b). Meta-learning-based system for solving logistic optimization problems. In *International conference on computer aided systems theory* (pp. 339–346). Springer.
Lessmann, S., Caserta, M., & Arango, I. M. (2011). Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Systems with Applications, 38*(10), 12826–12838.
Li, C., Chu, X., Chen, Y., & Xing, L. (2016). A knowledge-based technique for initializing a genetic algorithm. *Journal of Intelligent & Fuzzy Systems, 31*(2), 1145–1152.
Li, J., Pardalos, P. M., Sun, H., Pei, J., & Zhang, Y. (2015). Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. *Expert Systems with Applications, 42*(7), 3551–3561.
Li, K., & Tian, H. (2016). A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. *Applied Soft Computing, 43*, 469–479.
Li, X., & Olafsson, S. (2005). Discovering dispatching rules using data mining. *Journal of Scheduling, 8*(6), 515–527.
Liao, X.-L., & Ting, C.-K. (2012). Evolutionary algorithms using adaptive mutation for the selective pickup and delivery problem. In *2012 IEEE congress on evolutionary computation* (pp. 1–8). IEEE.
de Lima Junior, F. C., de Melo, J. D., & Neto, A. D. D. (2007). Using q-learning algorithm for initialization of the grasp metaheuristic and genetic algorithm. In *2007 international joint conference on neural networks* (pp. 1243–1248). IEEE.
Lodi, A., Mossina, L., & Rachelson, E. (2020). Learning to handle parameter perturbations in combinatorial optimization: An application to facility location. *EURO Journal on Transportation and Logistics*, 100023.
López Jaimes, A., Coello Coello, C. A., & Chakraborty, D. (2008). Objective reduction using a feature selection technique. In *Proceedings of the 10th annual conference on genetic and evolutionary computation* (pp. 673–680).
López-Santana, E., Rodríguez-Vásquez, W., & Méndez-Giraldo, G. (2018). A hybrid expert system, clustering and ant colony optimization approach for scheduling and routing problem in courier services. *International Journal of Industrial Engineering Computations, 9*(3), 369–396.
Loshchilov, I., Schoenauer, M., & Sebag, M. (2010). A mono surrogate for multiobjective optimization. In *Proceedings of the 12th annual conference on genetic and evolutionary computation* (pp. 471–478).
Lotfi, N., & Acan, A. (2016). A tournament-based competitive-cooperative multiagent architecture for real parameter optimization. *Soft Computing, 20*(11), 4597–4617.
Louis, S. J., & McDonnell, J. (2004). Learning with case-injected genetic algorithms. *IEEE Transactions on Evolutionary Computation, 8*(4), 316–328.
Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics* (pp. 320–353). Springer.
Lu, H., Zhang, X., & Yang, S. (2019). A learning-based iterative method for solving vehicle routing problems. In *International conference on learning representations*.
Lucas, F., Billot, R., Sevaux, M., & Sörensen, K. (2020). Reducing space search in combinatorial optimization using machine learning tools. In *International conference on learning and intelligent optimization* (pp. 143–150). Springer.
Lughofer, E. (2017). On-line active learning: A new paradigm to improve practical useability of data stream modeling methods. *Information Sciences, 415*, 356–376.
Martin, S., Ouelhadj, D., Beullens, P., & Ozcan, E. (2011). A generic agent-based framework for cooperative search using pattern matching and reinforcement learning. *Technical Report*.
Martin, S., Ouelhadj, D., Beullens, P., Ozcan, E., Juan, A. A., & Burke, E. K. (2016). A multi-agent based cooperative approach to scheduling and routing. *European Journal of Operational Research, 254*(1), 169–178.
Maturana, J., Fialho, A., Saubion, F., Schoenauer, M., Lardeux, F., & Sebag, M. (2011). Adaptive operator selection and management in evolutionary algorithms. In *Autonomous search* (pp. 161–189). Springer.
Maturana, J., Fialho, Á., Saubion, F., Schoenauer, M., & Sebag, M. (2009). Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *2009 ieee congress on evolutionary computation* (pp. 365–372). IEEE.
Maturana, J., & Riff, M.-C. (2007). Solving the short-term electrical generation scheduling problem by an adaptive evolutionary approach. *European Journal of Operational Research, 179*(3), 677–691.
Maturana, J., & Saubion, F. (2008). A compass to guide genetic algorithms. In *International conference on parallel problem solving from nature* (pp. 256–265). Springer.
Meignan, D., Créput, J.-C., & Koukam, A. (2008). A coalition-based metaheuristic for the vehicle routing problem. In *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)* (pp. 1176–1182). IEEE.
Meignan, D., Créput, J.-C., & Koukam, A. (2009). A cooperative and self-adaptive metaheuristic for the facility location problem. In *Proceedings of the 11th annual conference on genetic and evolutionary computation* (pp. 317–324).
Meignan, D., Koukam, A., & Créput, J.-C. (2010). Coalition-based metaheuristic: A self-adaptive metaheuristic using reinforcement learning and mimetism. *Journal of Heuristics, 16*(6), 859–879.
Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., & Neumann, F. (2013). A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence, 69*(2), 151–182.
Messelis, T., & De Causmaecker, P. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research, 233*(3), 511–528.
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics, 21*(6), 1087–1092.
Michalski, R. S. (2000). Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning, 38*(1–2), 9–40.
Miki, S., Yamamoto, D., & Ebara, H. (2018). Applying deep learning and reinforcement learning to traveling salesman problem. In *2018 international conference on computing, electronics & communications engineering (ICCECE)* (pp. 65–70). IEEE.
Min, J., Jin, C., & Lu, L. (2019). Maximum-minimum distance clustering method for split-delivery vehicle-routing problem: Case studies and performance comparisons.. *Advances in Production Engineering & Management, 14*(1), 125–135.
Miranda, E. S., Fabris, F., Nascimento, C. G., Freitas, A. A., & Oliveira, A. C. (2018). Meta-learning for recommending metaheuristics for the maxsat problem. In *2018 7th Brazilian conference on intelligent systems (BRACIS)* (pp. 169–174). IEEE.
Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100.
Mohammadi, M., Jula, P., & Tavakkoli-Moghaddam, R. (2019). Reliable single-allocation hub location problem with disruptions. *Transportation Research Part E: Logistics and Transportation Review, 123*, 90–120.
Mohammadi, M., Tavakkoli-Moghaddam, R., Siadat, A., & Dantan, J.-Y. (2016). Design of a reliable logistics network with hub disruption under uncertainty. *Applied Mathematical Modelling, 40*(9-10), 5621–5642.
Moradi, B. (2019). The new optimization algorithm for the vehicle routing problem with time windows using multi-objective discrete learnable evolution model. *Soft Computing*, 1–29.
Moraglio, A., Kim, Y.-H., & Yoon, Y. (2011). Geometric surrogate-based optimisation for permutation-based problems. In *Proceedings of the 13th annual conference companion on genetic and evolutionary computation* (pp. 133–134).
Mosadegh, H., Ghomi, S. F., & Süer, G. (2020). Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research, 282*(2), 530–544.
Moscato, P., et al. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*.
Mostafaie, T., Khiyabani, F. M., & Navimipour, N. J. (2020). A systematic study on meta-heuristic approaches for solving the graph coloring problem. *Computers & Operations Research, 120*, 104850.
Nasiri, M. M., Salesi, S., Rahbari, A., Meydani, N. S., & Abdollai, M. (2019). A data mining approach for population-based methods to solve the JSSP. *Soft Computing, 23*(21), 11107–11122.
Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2014). Selection schemes in surrogate-assisted genetic programming for job shop scheduling. In *Asia-pacific conference on simulated evolution and learning* (pp. 656–667). Springer.
Ochoa, G., & Burke, E. K. (2014). Hyperils: An effective iterated local search hyper-heuristic for combinatorial optimisation. In *International conference of the practice and theory of automated timetabling,(august)* (pp. 26–29).
Oliveira, A. L., Braga, P. L., Lima, R. M., & Cornélio, M. L. (2010). Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *information and Software Technology, 52*(11), 1155–1166.
Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research, 63*(5), 511–623.
Öztop, H., Tasgetiren, M. F., Kandiller, L., & Pan, Q.-K. (2020). A novel general variable neighborhood search through q-learning for no-idle flowshop scheduling. In *2020 ieee congress on evolutionary computation (CEC)* (pp. 1–8). IEEE.
Page, E. S. (1954). Continuous inspection schemes. *Biometrika, 41*(1/2), 100–115.
Park, J., & Kim, K.-Y. (2017). Meta-modeling using generalized regression neural network and particle swarm optimization. *Applied Soft Computing, 51*, 354–369.
Pathak, B. K., Srivastava, S., & Srivastava, K. (2008). Neural network embedded multiobjective genetic algorithm to solve non-linear time-cost tradeoff problems of project scheduling. *Journal of Scientific and Industrial Research, 67*(2), 124–131.
Pavelski, L., Delgado, M., & Kessaci, M.-E. (2018a). Meta-learning for optimization: A case study on the flowshop problem using decision trees. In *2018 IEEE congress on evolutionary computation (CEC)* (pp. 1–8). IEEE.

Pavelski, L. M., Kessaci, M.-É., & Delgado, M. R. (2018b). Recommending meta-heuristics and configurations for the flowshop problem via meta-learning: analysis and design. In *2018 7th Brazilian conference on intelligent systems (BRACIS)* (pp. 163–168). IEEE.
Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.-G., & Guerin, Y. (2020). Overview and comparison of gaussian process-based surrogate models for mixed continuous and discrete variables: Application on aerospace design problems. In *High--performance simulation-based optimization* (pp. 189–224). Springer.
Peng, B., Zhang, Y., Gajpal, Y., & Chen, X. (2019). A memetic algorithm for the green vehicle routing problem. *Sustainability, 11*(21), 6055.
Pettinger, J. E., & Everson, R. M. (2002). Controlling genetic algorithms with reinforcement learning. In *Proceedings of the 4th annual conference on genetic and evolutionary computation* (p. 692).
Phan, H. D., Ellis, K., Barca, J. C., & Dorin, A. (2019). A survey of dynamic parameter setting methods for nature-inspired swarm intelligence algorithms. *Neural Computing and Applications*, 1–22.
Pitzer, E., Beham, A., & Affenzeller, M. (2013). Automatic algorithm selection for the quadratic assignment problem using fitness landscape analysis. In *European conference on evolutionary computation in combinatorial optimization* (pp. 109–120). Springer.
Qasem, S. N., Shamsuddin, S. M., Hashim, S. Z. M., Darus, M., & Al-Shammari, E. (2013). Memetic multiobjective particle swarm optimization-based radial basis function network for classification problems. *Information Sciences, 239*, 165–190.
Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary computation, 12*(1), 64–79.
Ramos, I. C., Goldbarg, M. C., Goldbarg, E. G., & Neto, A. D. D. (2005). Logistic regression for parameter tuning on an evolutionary algorithm. In *2005 IEEE congress on evolutionary computation: 2* (pp. 1061–1068). IEEE.
Reinelt, G. (1991). Tsplib'a traveling salesman problem library. *ORSA Journal on Computing, 3*(4), 376–384.
Ribeiro, M. H., Plastino, A., & Martins, S. L. (2006). Hybridization of grasp metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms, 5*(1), 23–41.
Rice, J. R., et al. (1976). The algorithm selection problem. *Advances in Computers, 15*(65–118), 5.
de la Rosa-Rivera, F., Nunez-Varela, J. I., Ortiz-Bayliss, J. C., & Terashima–Marín, H. (2021). Algorithm selection for solving educational timetabling problems. *Expert Systems with Applications, 174*, 114694.
Sadeg, S., Hamdad, L., Haouas, M., Abderrahmane, K., Benatchba, K., & Habbas, Z. (2019). Unsupervised learning bee swarm optimization metaheuristic. In *International work-conference on artificial neural networks* (pp. 773–784). Springer.
Sadeg, S., Hamdad, L., Kada, O., Benatchba, K., & Habbas, Z. (2020). Meta-learning to select the best metaheuristic for the maxsat problem. In *International symposium on modelling and implementation of complex systems* (pp. 122–135). Springer.
Sakurai, Y., Takada, K., Kawabe, T., & Tsuruta, S. (2010). A method to control parameters of evolutionary algorithms by using reinforcement learning. In *2010 sixth international conference on signal-image technology and internet based systems* (pp. 74–79). IEEE.
Sakurai, Y., & Tsuruta, S. (2012). A population based rewarding for reinforcement learning to control genetic algorithms. In *2012 eighth international conference on signal image technology and internet based systems* (pp. 686–691). IEEE.
Santos, H. G., Ochi, L. S., Marinho, E. H., & Drummond, L. M. d. A. (2006). Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing, 70*(1–3), 70–77.
dos Santos, J. P. Q., de Lima Júnior, F. C., Magalhães, R. M., de Melo, J. D., & Neto, A. D. D. (2010). A parallel hybrid implementation using genetic algorithms, grasp and reinforcement learning for the salesman traveling problem. In *Computational intelligence in expensive optimization problems* (pp. 345–369). Springer.
dos Santos, J. P. Q., de Melo, J. D., Neto, A. D. D., & Aloise, D. (2014). Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. *Expert Systems with Applications, 41*(10), 4939–4949.
Saxena, D. K., Duro, J. A., Tiwari, A., Deb, K., & Zhang, Q. (2012). Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation, 17*(1), 77–99.
Segredo, E., Paechter, B., Hart, E., & González-Vila, C. I. (2016). Hybrid parameter control approach applied to a diversity-based multi-objective memetic algorithm for frequency assignment problems. In *2016 IEEE congress on evolutionary computation (CEC)* (pp. 1517–1524). IEEE.
Sghir, I., Hao, J.-K., Jaafar, I. B., & Ghédira, K. (2015). A multi-agent based optimization method applied to the quadratic assignment problem. *Expert Systems with Applications, 42*(23), 9252–9262.
Sghir, I., Jaafar, I. B., & Ghédira, K. (2018). A multi-agent based optimization method for combinatorial optimization problems. *International Journal on Artificial Intelligence Tools, 27*(05), 1850021.
Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming* (pp. 417–431). Springer.
Shi, L., & Rasheed, K. (2010). A survey of fitness approximation methods applied in evolutionary algorithms. In *Computational intelligence in expensive optimization problems* (pp. 3–28). Springer.
da Silva, A. M. L., Freire, M. R., & Honório, L. M. (2016). Transmission expansion planning optimization by adaptive multi-operator evolutionary algorithms. *Electric Power Systems Research, 133*, 173–181.
Silva, M. A. L., de Souza, S. R., Souza, M. J. F., & Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert systems with applications, 131*, 148–171.
Silva, M. A. L., de Souza, S. R., Souza, M. J. F., & de Franca Filho, M. F. (2018). Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Applied soft computing, 71*, 433–459.
Silva, M. A. L., de Souza, S. R., Souza, M. J. F., & de Oliveira, S. M. (2015). A multi-agent metaheuristic optimization framework with cooperation. In *2015 Brazilian conference on intelligent systems (BRACIS)* (pp. 104–109). IEEE.
Singh, H. K., Ray, T., & Smith, W. (2010). Surrogate assisted simulated annealing (sasa) for constrained multi-objective optimization. In *IEEE congress on evolutionary computation* (pp. 1–8). IEEE.
Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research, 45*, 12–24.
Smith-Miles, K., van Hemert, J., & Lim, X. Y. (2010). Understanding TSP difficulty by learning from evolved instances. In *International conference on learning and intelligent optimization* (pp. 266–280). Springer.
Smith-Miles, K., & Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research, 39*(5), 875–889.
Smith-Miles, K. A. (2008). Towards insightful algorithm selection for optimisation using meta-learning concepts. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)* (pp. 4118–4124). ieee.
Song, H., Triguero, I., & Özcan, E. (2019). A review on the self and dual interactions between machine learning and optimisation. *Progress in Artificial Intelligence, 8*(2), 143–165.
Sra, S., Nowozin, S., & Wright, S. J. (2012). *Optimization for machine learning*. Mit Press.
Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization, 11*(4), 341–359.
Sun, J., Zhang, H., Zhou, A., Zhang, Q., & Zhang, K. (2019). A new learning-based adaptive multi-objective evolutionary algorithm. *Swarm and evolutionary computation, 44*, 304–319.
Taillard, E. (1993). Benchmarks for basic scheduling problems. *European journal of Operational Research, 64*(2), 278–285.
Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics, 8*(5), 541–564.
Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. John Wiley & Sons, Inc..
Talbi, E.-G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research, 240*(1), 171–215.
Talbi, E.-G. (2020). Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics. *Working paper*.
Thevenin, S., & Zufferey, N. (2019). Learning variable neighborhood search for a scheduling problem with time windows and rejections. *Discrete Applied Mathematics, 261*, 344–353.
Turkeš, R., Sörensen, K., & Hvattum, L. M. (2020). Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2020.10.045.
Van Luong, T., Melab, N., & Talbi, E.-G. (2011). Gpu computing for parallel local search metaheuristic algorithms. *IEEE Transactions on Computers, 62*(1), 173–185.
Voudouris, C., & Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research, 113*(2), 469–499.
Voudouris, C., & Tsang, E. P. (2003). Guided local search. In *Handbook of metaheuristics* (pp. 185–218). Springer.
Wagner, S., & Affenzeller, M. (2005). Heuristiclab: A generic and extensible optimization environment. In *Adaptive and natural computing algorithms* (pp. 538–541). Springer.
Walker, J. D., Ochoa, G., Gendreau, M., & Burke, E. K. (2012). Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In *International conference on learning and intelligent optimization* (pp. 265–276). Springer.
Wang, H., & Jin, Y. (2018). A random forest-assisted evolutionary algorithm for data–driven constrained multiobjective combinatorial optimization of trauma systems. *IEEE Transactions on Cybernetics, 50*(2), 536–549.
Wang, X., & Tang, L. (2017). A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem. *Computers & Operations Research, 79*, 60–77.
Wang, Y., Pan, S., Li, C., & Yin, M. (2020a). A local search algorithm with reinforcement learning based repair procedure for minimum weight independent dominating set. *Information Sciences, 512*, 533–548.
Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020b). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR), 53*(3), 1–34.
Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. King's College, Cambridge Ph.D. thesis..
Wauters, T., Verbeeck, K., De Causmaecker, P., & Berghe, G. V. (2013). Boosting metaheuristic search using reinforcement learning. In *Hybrid metaheuristics* (pp. 433–452). Springer.
Wawrzyniak, J., Drozdowski, M., & Sanlaville, É. (2020). Selecting algorithms for

large berth allocation problems. *European Journal of Operational Research, 283*(3), 844–862.
Wojtusiak, J., Warden, T., & Herzog, O. (2011). Agent-based pickup and delivery planning: the learnable evolution model approach. In *2011 international conference on complex, intelligent, and software intensive systems* (pp. 1–8). IEEE.
Wojtusiak, J., Warden, T., & Herzog, O. (2012). The learnable evolution model in agent-based delivery optimization. *Memetic Computing, 4*(3), 165–181.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation, 1*(1), 67–82.
Wu, W., & Tseng, S.-P. (2017). An improved learnable evolution model for discrete optimization problem. In *Advances in intelligent information hiding and multimedia signal processing* (pp. 333–340). Springer.
Xiang, T., & Pan, D. (2016). Clustering algorithm for split delivery vehicle routing problem. *Journal of Computer Applications, 36*(11), 3141–3145.
Xiang, X., Tian, Y., Xiao, J., & Zhang, X. (2020). A clustering-based surrogate-assisted multi-objective evolutionary algorithm for shelter location under uncertainty of road networks. *IEEE Transactions on Industrial Informatics, 16*(12), 7544–7555.
Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2015). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation, 20*(4), 606–626.
Yalcinoz, T., & Altun, H. (2001). Power economic dispatch using a hybrid genetic algorithm. *IEEE power engineering review, 21*(3), 59–60.
Yu, H., Tan, Y., Sun, C., & Zeng, J. (2017). Clustering-based evolution control for surrogate-assisted particle swarm optimization. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 503–508). IEEE.
Yuan, Z., Handoko, S. D., Nguyen, D. T., & Lau, H. C. (2014). An empirical study of off-line configuration and on-line adaptation in operator selection. In *International conference on learning and intelligent optimization* (pp. 62–76). Springer.
Zennaki, M., & Ech-Cherif, A. (2010). A new machine learning based approach for tuning metaheuristics for the solution of hard combinatorial optimization problems. *Journal of Applied Sciences(Faisalabad), 10*(18), 1991–2000.
Zhalechian, M., Tavakkoli-Moghaddam, R., Zahiri, B., & Mohammadi, M. (2016). Sustainable design of a closed-loop location-routing-inventory supply chain network under mixed uncertainty. *Transportation Research Part E: Logistics and Transportation Review, 89*, 182–214.
Zhang, H., Zhou, A., Song, S., Zhang, Q., Gao, X.-Z., & Zhang, J. (2016). A self-organizing multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation, 20*(5), 792–806.
Zhang, J. (2017). An efficient density-based clustering algorithm for the capacitated vehicle routing problem. In *2017 international conference on computer network, electronic and automation (ICCNEA)* (pp. 465–469). IEEE.
Zhang, J., Zhan, Z.-h., Lin, Y., Chen, N., Gong, Y.-j., Zhong, J.-h., … Shi, Y.-h. (2011). Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine, 6*(4), 68–75.
Zhao, F., Zhang, L., Cao, J., & Tang, J. (2021). A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Computers & Industrial Engineering, 153*, 107082.
Zheng, Y., Fu, X., & Xuan, Y. (2019). Data-driven optimization based on random forest surrogate. In *2019 6th international conference on systems and informatics (ICSAI)* (pp. 487–491). IEEE.
Zheng, Y.-J. (2015). Water wave optimization: A new nature-inspired metaheuristic. *Computers & Operations Research, 55*, 1–11.
Zhou, Y., Hao, J.-K., & Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications, 64*, 412–422.
Zhou, Y., Hao, J.-K., & Duval, B. (2020). Frequent pattern-based search: A case study on the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.
Zhou, Z., Ong, Y. S., Nguyen, M. H., & Lim, D. (2005). A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In *2005 IEEE congress on evolutionary computation: 3* (pp. 2832–2839). IEEE.