Machine Learning into Metaheuristics: A Survey and Taxonomy

EL-GHAZALI TALBI, University of Lille

During the past few years, research in applying machine learning (ML) to design efficient, effective, and robust metaheuristics has become increasingly popular. Many of those machine learning-supported metaheuristics have generated high-quality results and represent state-of-the-art optimization algorithms. Although various appproaches have been proposed, there is a lack of a comprehensive survey and taxonomy on this research topic. In this article, we will investigate different opportunities for using ML into metaheuristics. We define uniformly the various ways synergies that might be achieved. A detailed taxonomy is proposed according to the concerned search component: target optimization problem and low-level and high-level components of metaheuristics. Our goal is also to motivate researchers in optimization to include ideas from ML into metaheuristics. We identify some open research issues in this topic that need further in-depth investigations.

CCS Concepts: • **Computing methodologies → Search methodologies;**

Additional Key Words and Phrases: Metaheuristics, machine learning, optimization, ML-supported metaheuristics

ACM Reference format:
El-Ghazali Talbi. 2021. Machine Learning into Metaheuristics: A Survey and Taxonomy. *ACM Comput. Surv.* 54, 6, Article 129 (July 2021), 32 pages.
https://doi.org/10.1145/3459664

1 INTRODUCTION

Over the past few decades, metaheuristics have shown their efficiency in solving various complex optimization problems in science and industry [234]. In general, metaheuristics do not use explicit knowledge discovered during the search using advanced **machine learning (ML)** models. Metaheuristics generate a lot of data in the search process. The data can be *static* when they concern the target problem and instance features to solve. Moreover, several *dynamic* data are generated during the iterative search process: solutions in the decision and the objective spaces, sequence of solutions or trajectories, successive populations of solutions, moves, recombinations, local optima, elite solutions, bad solutions, and so on. Thus, ML can be helpful in analyzing these data to extract useful knowledge. This knowledge will guide and enhance the search performance of metaheuristics and make them "smarter" and "well informed." Incorporating machine learning into metaheuristics has been proven to be advantageous in both convergence speed, solution quality, and robustness.

Authors' address: E.-G. Talbi, University of Lille, Polytech'Lille, Cité scientifique, Villeneuve d'Ascq 59655, France; email: el-ghazali.talbi@univ-lille.fr.

0360-0300/2021/07-ART129 $15.00
https://doi.org/10.1145/3459664

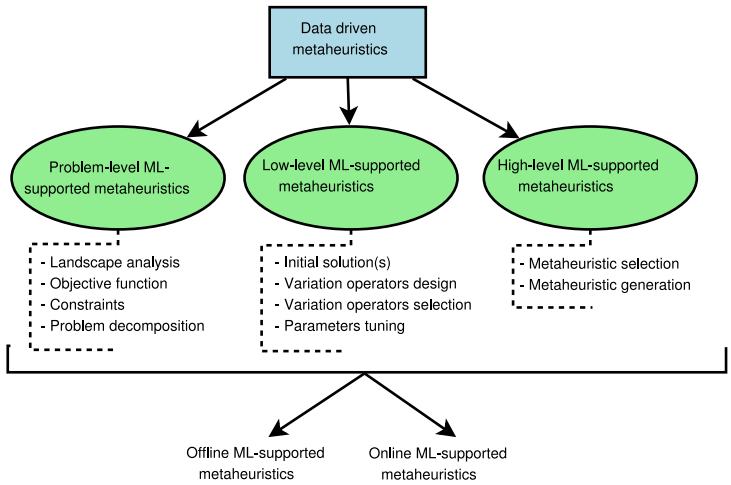

Fig. 1. A general taxonomy of machine learning-supported metaheuristics.

In this article, a survey of machine learning-supported metaheuristics to solve difficult optimization problems is presented. A taxonomy is also proposed in an attempt to provide a common terminology and classification mechanisms. The goal of the general taxonomy given here is to provide a framework to allow comparison of this family of metaheuristics in a qualitative way. In addition, it is hoped that the categories and their relationships to each other have been chosen carefully enough to indicate areas requiring research efforts as well as to help classify future work. We distinguish three hierarchical ways to use ML in metaheuristics (Figure 1):

- **Problem-level ML-supported metaheuristics:** ML can help in modeling the optimization problem to solve (e.g., objective function, constraints). It can also assist landscape analysis and the decomposition of the problem.
- **Low-level ML-metaheuristics:** A metaheuristic is composed of different search components. ML can drive any search component such as the initialization of solution(s) and the search variation operators (e.g., neighborhoods in local search, mutation and crossover in evolutionary algorithms). It may also be used to tune the various parameters of a metaheuristic.
- **High-level ML-supported metaheuristics:** This class of ML-supported metaheuristics concerns the selection and generation of metaheuristics and the design of hybrid and parallel cooperative metaheuristics.

Other flat criteria are used in the taxonomy such as the learning time. In *offline* ML-supported metaheuristics, the ML process occurs *a priori* before starting to solve the problem. In *online* ML-supported metaheuristics, ML gather knowledge during the search while solving the problem.

The synergy between ML and optimization has received increasing attention. Most of the related works basically focus on the use of optimization algorithms in solving ML problems [26, 49, 60, 150, 229]. Indeed, most of the ML problems can be formulated as optimization problems. In the last decade, there was a considerable interest in the use of ML into optimization. Very few papers investigate the role of ML into exact optimization algorithms (e.g., branch and bound, dynamic programming), constraint programming, and mathematical programming [22]. To our knowledge, there is no comprehensive survey that identifies in a unified way how ML can help the design of metaheuristics. In some outdated surveys [49, 120, 279], the authors enumerate some

ML-supported metaheuristics. In Reference [36] the authors focus on dynamic combinatorial optimization problems. In Reference [235], we have proposed a taxonomy of hybrid metaheuristics, in which the combination of metaheuristics with mathematical programming, constraint programming and ML has been addressed. In this article a more complete and general taxonomy of ML-supported metaheuristics is proposed. More than 125 references have been analyzed according to our taxonomy. The unified taxonomy is kept as small as possible by proceeding in a hierarchical way as long as possible; then a flat classification is presented according to other criteria.

The article is structured as follows. In Section 2, the main concepts of metaheuristics and ML are detailed in a general and unified way. Section 3 focuses on problem-level ML-supported metaheuristics. In Section 4, low-level ML-supported metaheuristics are presented, while in Section 5 we describe high-level ML-supported metaheuristics. Finally, the last section presents the main conclusions and identifies some research perspectives.

2 MAIN CONCEPTS

This section presents in a unified way the main concepts used for metaheuristics and machine learning.

2.1 Metaheuristics

An optimization problem consists in searching the optimal solution(s) x^* from a search space X, which maximize (or minimize) an *objective function* $f(x)$ while satisfying a set of *constraints*. The search space X may be composed of discrete variables (e.g., integer, categorical), continuous variables or mixed variables [176]. Metaheuristics represent a class of general-purpose heuristic algorithms that can be applied to any optimization problem. Unlike exact methods, metaheuristics allow to tackle large-scale problems by delivering satisfactory solutions in a reasonable time. There is no guarantee to find global optimal solutions or even bounded solutions. Metaheuristics have received more and more popularity in the past 30 years. Indeed, their use in many applications shows their efficiency and effectiveness to solve large and complex problems.

In the design of a metaheuristic, two contradictory criteria must be taken into account: *exploration* of the search space (*diversification*), and *exploitation* of the best solutions found (*intensification*). Promising regions are determined through the "good" solutions obtained. In intensification, the promising regions are explored more throughly with the hope to find better solutions. In diversification, non-explored regions must be visited to be sure that all regions of the search space are evenly explored and that the search is not only confined to a reduced number of regions.

2.1.1 Single-solution-based Metaheuristics. *Single-solution-based metaheuristics* (S-metaheuristics) improve a single solution. They could be seen as "walks" through neighborhoods or search trajectories through the search space of the target problem [234]. S-metaheuristics iteratively apply the generation and replacement procedures from the current solution. In the generation phase, a set of candidate solutions are generated from the current solution s. This set $C(s)$ is generally obtained by local transformations of the solution. In the replacement phase,[1] a selection is performed from the candidate solution set $C(s)$ to replace the current solution, i.e., a solution $s' \in C(s)$ is selected to be the new solution. This process iterates until a defined stopping criteria. The generation and the replacement phases may be *memoryless*. In this case, the two procedures are based only on the current solution. Otherwise, some history of the search stored in a memory can be used in the generation of the candidate list of solutions and the selection of the new solution. Popular examples of such S-metaheuristics are local search, simulated annealing and tabu search.

[1] Also named transition rule, pivoting rule, and selection strategy.

Algorithm 1 illustrates the high-level template of this family of metaheuristics. Their common search concepts are the definition of the *neighborhood* structure and the generation of the *initial solution*.

ALGORITHM 1: High-level template of single-solution-based metaheuristics

$s = s_0$; /* *Generation of the initial solution* */
$t = 0$;
Repeat
  /* *Generate candidate solutions from* s_t */
  Generate($C(s_t)$);
  /* *Select a solution from* $C(s_t)$ *to replace the current solution* s_t */
  s_{t+1} = Select($C(s_t)$);
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.

2.1.2 Population-based Metaheuristics. *Population-based metaheuristics* (P-metaheuristics) could be viewed as an iterative improvement of a population of solutions. P-metaheuristics start from an initial population of solutions.[2] Then, they iteratively apply the generation of a new population and the replacement of the current population. In the generation phase, a new population of solutions is created. In the replacement phase, a selection is carried out from the current and the new populations. This process iterates until a given stopping criteria. Popular examples of P-metaheuristics are evolutionary algorithms, ant colony optimization, scatter search, differential evolution, particle swarm optimization, bee colony and artificial immune systems. Algorithm 2 illustrates the high-level template of P-metaheuristics.

ALGORITHM 2: High-level template of P-metaheuristics

$P = P_0$; /* *Generation of the initial population* */
$t = 0$;
Repeat
  Generate(P'_t); /* *Generation a new population* */
  P_{t+1} = Replace-Population($P_t \cup P'_t$); /* *Select new population* */
  $t = t+1$;
Until Stopping criteria satisfied
Output: Best solution(s) found.

P-metaheuristics may be classified into two main categories:

- **Evolutionary based:** In this category of P-metaheuristics, the solutions composing the population are selected and reproduced using variation operators (e.g., mutation, recombination[3]) acting *directly* on their representations. A new solution is constructed from the different attributes of solutions belonging to the current population. **Evolutionary algorithms (EAs)**, **Particle Swarm Optimization (PSO)** and scatter search represent well-known examples of this class of P-metaheuristics.

[2]Some P-metaheuristics such as ant colony optimization start from partial or empty solutions.
[3]Also called crossover and merge.

- **Blackboard based**[4]**:** It is based on the idea that the solutions of the population participate in the construction of a shared memory. This shared memory will be the main input in generating the new population of solutions. **Ant colony optimization (ACO)** and **estimation of distribution algorithms (EDA)** belong to this class of P-metaheuristics. For the former, the shared memory is represented by the pheromone matrix while, in the latter strategy, it is represented by a probabilistic learning model. For instance, in ant colonies, the generated solutions by past ants will affect the generation of future ants via the pheromones. Then, the generated solutions participate in updating the pheromones.

Many of the well-known metaheuristics originally tackle continuous spaces such as PSO (respectively, discrete spaces such as ACO) because these have been designed naturally in a continuous space (respectively, discrete space). In the past two decades, many approaches have been proposed to adapt continuous metaheuristics to tackle discrete problems [52] and vice versa [227].

2.2 Machine Learning

ML is one of the most salient research domain in artificial intelligence. ML has experienced an important development in the past decade and has become a powerful analysis tool in a wide range of applications related to big data. ML is the science of extracting useful and hidden patterns from a training dataset considered composed of multiple examples. Various ML tasks can be used depending on the desired outcome of the model. Usually a distinction is made among supervised, semi-supervised, and unsupervised learning. The most common ML tasks used in this article are as follows (Figure 2):

- **Regression and classification:** This is a supervised ML task in which we predict a predefined category or class from a given set of attributes (continuous variables for regression and discrete variables for classification) [8]. The following ML techniques are generally used for solving this family of problems: Gaussian process, linear regression, K-nearest neighbors, **artificial neural networks (ANN)**, **support vector machine (SVM)**, random forests, decision trees, logistic regression, naive Bayes, and deep learning.
- **Clustering:** This unsupervised ML task partitions the input dataset into subsets (clusters), so that data in each subset share common aspects [76]. The partitioning is often indicated by a similarity measure defined by a distance. The main used techniques for clustering are: hierarchical clustering, partitioning methods (e.g., *k*-means, K-medoids, Mean-Shift), grid-based clustering (e.g., CLIQUE, Sting, Wave Cluster), model-based clustering (e.g., EM, COBWEB), and density-based methods (e.g., DBSCAN, Optics, Denclue).
- **Association rules:** Association rule mining is a procedure that is meant to find frequent patterns, correlations, associations, or causal structures from datasets found in various kinds of databases [79]. One of the most efficient used methods are Apriori, FP-growth, and Eclat. Many measures of interestingness for rules have been proposed such as support, confidence, conviction, leverage, and lift (i.e., interest).
- **Feature selection:** The *feature selection* task consists in reducing the number of attributes (i.e., dimensionality of the dataset) [40]. It is an important task, because selecting significant features would help to build simpler models of better accuracy and can reduce overfitting. The traditional techniques used for feature selection are as follows: filter methods (e.g., Pearson's correlation, linear discriminant analysis, analysis of variance, chi-square tests),

[4]A blackboard system is an artificial intelligence application based on the blackboard architectural model, where a shared knowledge, the "blackboard," is iteratively updated by a diverse group of agents [73].

wrapper methods (e.g., forward selection, backward selection, recursive feature elimination), and embedded methods (e.g., mRMR, Greedy).

- **Reinforcement learning: Reinforcement learning (RL)** aims to learn optimal actions from a finite set of available actions through continuously interacting with an unknown environment [232]. The main components are as follows: states, set of actions, and rewards. From the current state, a RL agent takes an action, changes to a new state, and receives a reward. The reward determines the quality of the carried action. RL is also referred to as *approximate dynamic programming* [185]. The goal of the agent is to maximize the accumulation of rewards over time. The main approaches for RL can be classified as follows:
  - **Model-free:** The optimal control policy is learned without first learning an explicit model. Such schemes include **temporal difference (TD)** methods [233], Q-learning [257], State-Action-Reward-State-Action [86], Deep Q Network [11], and Monte Carlo methods (e.g., Monte Carlo Tree Search) [29].
  - **Model based:** Conversely, model-based algorithm uses a reduced number of interactions with the real environment during the learning phase. Its aim is to construct a model based on these interactions and then use this model to simulate further episodes, not in the real environment but by applying them to the constructed model and get the results from it. Model-based RL can be classified as value function methods, policy methods (e.g., evolutionary algorithms, gradient-based, Bayesian optimization), transition models, and return functions [182].

3 PROBLEM-LEVEL ML-SUPPORTED METAHEURISTICS

In this class of ML-supported metaheuristics, ML techniques are carried out on the optimization problem at hand. Knowledge is obtained by the analysis of the landscape of the problem using selected features. Moreover, the problem model can be reformulated (e.g., objective function, constraints) or decomposed for a more efficient and effective solving.

3.1 Landscape Analysis

Exploratory landscape analysis represent a number of techniques used to extract knowledge on the characteristics of metaheuristic search to solve a given optimization problem [175]. The main questions for this scientific challenge are as follows:

- **Which features?** High-level features specified by human experts such as the level of multimodality or separability have been introduced to reflect the problem characteristics [18]. Other properties based on expert knowledge can be used such as: global basin structure, variable scaling, search space homogeneity, basin size homogeneity, global to local optima contrast, and size of plateaus. Low-level features are automatically and numerically computed to characterize the landscape of a given problem. The features can be grouped into five classes related to: the characteristics of the distribution of the objective function values (y-Distribution), the relative position of each objective value compared to the median of all values (Levelset), meta-modeling of the initial dataset (Meta-Model), the estimated degree of convexity of the function (Convexity) as well as the assessment of multimodality by local searches starting from the initial design points (Local Search) [151]. The selection of the features is mostly done offline.
- **Which goal?** Those features are important for the design of efficient metaheuristics and understanding the behavior of the algorithms. This knowledge can be used for the selection of the best suited metaheuristic to solve a given problem by predicting the characteristics of the input instance [25, 57, 123, 264]. It can also be used to tune the parameters of an algorithm

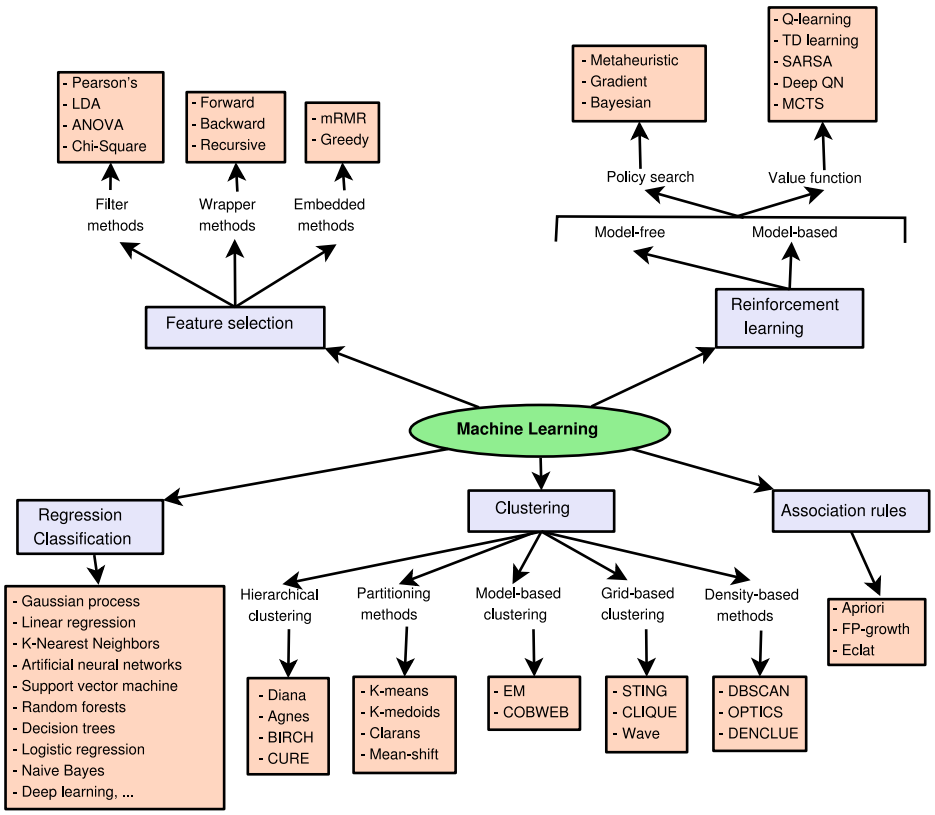

Fig. 2. Main machine learning tasks and associated methods.

[16, 104], predict its runtime for solving a given benchmark instance [109], and generating benchmark instances [80, 133, 138]. Most of the ML-supported landscape analysis use offline strategies [16, 25, 109, 138, 264].

- **Which ML methods?** Many ML methods have been investigated to extract those features such as Bayesian networks [180], support vector regression [25], random forest [57, 109], Gaussian process [109], neural networks [225], regression trees [16], and ridge regression [264].

3.2 Objective Function

Two different goals govern the use of ML in the objective function (Figure 3):

- **Improving the convergence:** ML can help to transform the objective functions to better guide the metaheuristic through the search space. Those learnable objective functions include some knowledge extracted from the problem.
- **Reducing the computational cost:** It consists in approximating the objective function. The approximation is evaluated much faster than the original function for expensive optimization problems.

3.2.1 Learnable Objective Function. The main issue in this family of methods is to generate automatically improved objective functions from exploiting some knowledge of the target optimization

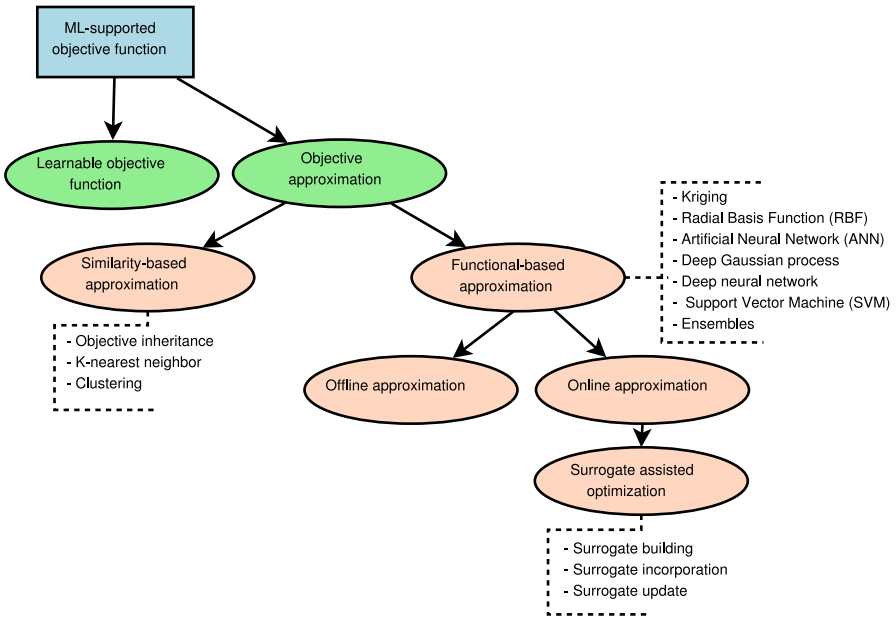

Fig. 3. ML-supported objective function.

problem and features extracted from states visited during the search process. Those online learnable objective functions can help guiding the search to improve solutions. For instance, this is an important issue for problems characterized by multimodality and neutrality. A representative of such an approach is Guided Local Search, which modifies the objective function when trapped on a local optima. The augmented objective function of the problem include a set of penalty terms [250]. Whenever the S-metaheuristic gets caught in a local optima, the penalties are modified and search is iterated to minimize the transformed objective function. The penalty gives the degree up to which the solution features is constrained.

Some approaches to construct learnable objective functions using reinforcement learning have been proposed. In Reference [27], some features characterizing good solutions are defined for the given problem. The objective function is predicted by analyzing search trajectories of local search methods. A TD(λ) family of *temporal-difference reinforcement learning* algorithms is used. The learned evaluation function is then used to bias future search trajectories toward better solutions on the same problem. *Transfer learning* has also been used to transfer previously learned evaluation functions to new, similar optimization problems [78]. In Reference [121], an **inverse reinforcement learning (IRL)** approach is proposed to learn the objective function in robotic problems. The main idea is to predict a suitable objective function given observations of optimal behavior. In many robotics problems, it is significantly easier to provide demonstrations of optimal behavior than it is to design an objective function that defines the required behavior. IRL is used to learn the observed behavior using features of solutions selected by L_1 regularization.

3.2.2 Objective Approximation. Many optimization problems (e.g., engineering design) are concerned by extremely expensive objective functions. Those problems are generally characterized by black-box objective functions whose algebraic definitions are unknown (e.g., simulation-based optimization). However, metaheuristics need to carry out a huge number of function evaluation to

find good solutions. For such expensive optimization problems, the approximation of the objective function is an interesting alternative for designing efficient ML-supported metaheuristics.

ML approaches can be used to build approximate models of the objective function [114]. In this context, previously evaluated solutions are learned by a ML approach to approximate the objective function of new generated solutions. The main issue here is to obtain a "good" approximation in terms of maximizing its quality and minimizing its computing time. Many questions arise in the design of this scheme such as the following: which proportion of the visited solutions are evaluated using the approximation and at what time or in which component of the search algorithm the approximation is used. Objective approximation methods can be classified into two families (Figure 3):

- **Similarity-based approximation:** can be seen as online Lazy learners or memory-based learners. The most popular similarity-based approximation are objective inheritance, K-nearest neighbor, and clustering techniques. Objective inheritance methods[5] are popular in P-metaheuristics. In Reference [224], the objective value of a new solution is computed as a linear weighted combination of the parents, in which the weights depend on similarity with the parents. In Reference [31], a resampling technique is combined with an average combination to solve noisy problems. Other objective inheritance methods based on conditional probabilities tables and decision trees are proposed in Reference [178]. In the k-nearest neighbors method, the objective value of a given solution is computed according to the k-nearest neighbors with known exact values [278]. Similarity-based approximation can also be carried out by clustering algorithms. A clustering algorithm is applied on a population of solutions to be evaluated. Each cluster will have a representative solution. Only the solution that represents the cluster is evaluated [116, 127, 194, 271]. Then, the objective function of other solutions of the cluster is estimated with respect to its associated representative. Different clustering techniques may be used such as *k*-means and fuzzy C-means.
- **Functional-based approximation:** This offline ML strategy consists in constructing a new model of the objective function before the search starts. The model construction strategy is based on previous data obtained from the original objective functions. Many ML algorithms have been investigated [15, 114]: Polynomial models (i.e., Response Surface) [88], **Radial Basis Functions (RBF)**, Kriging (i.e., Gaussian process) [30, 128], SVM [222], Monte Carlo sampling [45], and ANNs [83, 100, 105, 230]. A recent approach consists of using an ensemble of surrogates to improve the performance of the approximation [87]. Multiple criteria have to be used to compare the various models: number of samples provided to build the model, number of parameters of the model, quality of the approximation, cost of the model building, and cost of the model inference.

Some hybrid approaches combining similarity-based and functional-based approximations have been proposed. An example of such a hybrid approache is a clustering approach applied in EAs in which we split the population into several clusters and then construct an approximate model for each cluster. Multiple approximate models are expected to use more local information about the search space and fit the original objective function better than a single global model [39, 179, 264].

3.2.3 Surrogate-assisted Metaheuristics. Surrogate-assisted optimization[6] is a popular approach to deal with the optimization of expensive problems [238]. These algorithms are iterative sampling procedures relying on surrogate models (i.e., metamodels) of the considered objective and constraint functions that are generally characterized by a negligible computational cost, to iteratively

[5]This scheme is also called fitness imitation or fitness inheritance.
[6]Also known as Bayesian Optimization.

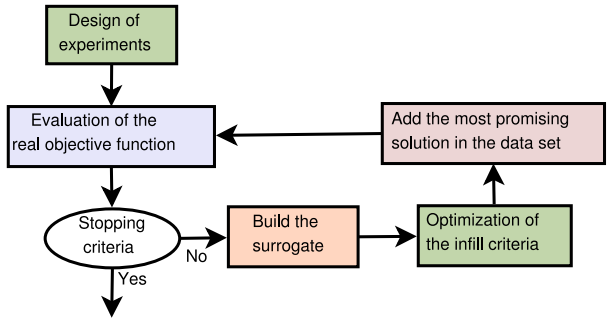

Fig. 4. The EGO Bayesian optimization algorithm.

determine and explore the most promising locations of the design space, thus simultaneously refining online the surrogate model and converging toward the problem optimum.

The main questions in designing surrogate-based metaheuristics are as follows:

- **Surrogate building:** Which ML approach is used to build the surrogate? Different approaches are used in the literature: random forest, polynomial models, Gaussian process [284], neural networks [251], decision tree [163], radial basis functions [62, 196], support vector regression [163], deep Gaussian processes [98], and deep neural networks [237]. A recent trend is to use multiple surrogates (i.e., ensembles of metamodels) to improve the accuracy of the surrogates [35, 77, 262]. Some surrogate models can be classified as global or local surrogate models [42, 141]. The global surrogate model is built based on all samples for global exploration; while, the local surrogate model is constructed with a predefined number of best samples for local exploitation.
- **Surrogate incorporation:** Which solutions should be selected to be evaluated using the real objective function or the surrogate? *Evolution Control* uses jointly surrogates and original objective functions in a metaheuristic. The original objective functions are used to evaluate some/all solutions in some/all iterations, in a fixed or adaptive way [46]. In *direct approximation*, the approximated objective function replaces the original one in the whole search process [213, 224]. In the *indirect approximation*, the metaheuristic use the approximation in some search operators (e.g., neighborhood, population initialization, crossover, mutation) [112, 193].
- **Surrogate update:** When and how the surrogate is updated? The surrogate can be updated in a fixed (e.g., each iteration, given number of iterations) or an adaptive way (e.g., improvement of solutions). Different *infill criteria* have been used for updating the surrogate: lower confidence bound, upper confidence bound, probability of improvement, and expected improvement [275]. They are based on a tradeoff between exploration by searching where predicted variance is high and exploitation by searching where expected value is minimized.

One of the most popular Bayesian Optimization algorithms is the **"Efficient Global Optimization" (EGO)** algorithm [117]. It is based on Gaussian Process regression (also called Kriging). First, a set of solutions are generated using **Design of Experiments (DoE)**. Then, it consists in sampling iteratively, using the prediction and uncertainty by the Gaussian model, the most promising solution based on an *infill sampling criterion*. This point is evaluated using the real objective function and the surrogate is updated using the new training set, and a new solution is sampled, and so on, until a given stopping criterion is satisfied (Figure 4).

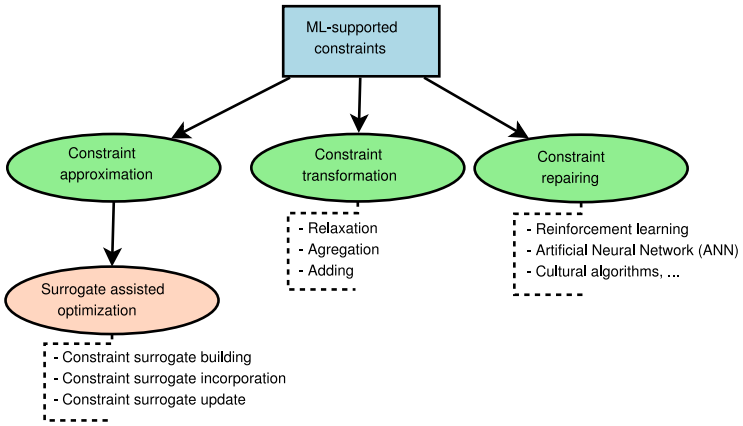

Fig. 5. ML-supported handling of constraints.

Surrogate-assisted optimization has been also extended to multi-objective optimization [94, 189, 255], bi-level optimization [10, 58], and multi-fidelity optimization [97, 267, 269].

3.3 Constraint Handling

For a given problem, there are usually domain specific constraints that can be exploited to construct efficient metaheuristics. The most popular constraint handling procedures in metaheuristics are the penalization and the repairing procedures. Different ML-based constraint handling approaches exist (Figure 5):

- **Constraint approximation:** The approximation strategies detailed in the previous section for the objective functions may also applied to constraints. Online surrogates for constraints have been applied to problems with expensive black-box constraints [157, 204]. Various ML models have been investigated such as ANN [115], Kriging [246], RBF [197], and SVM [143]. Different infill sampling criteria are used to deal with constrained global optimization problems (e.g., expected improvement with Probability of Feasibility [258], Expected Violation, Mean Constraint) [186]. The infill criteria aims to balance between exploitation and exploration of the objective and all the constraint approximations [172].
- **Constraint transformation:** In penalization strategies, online adaptive strategies have been used for switching from lower complexity surrogates to higher-complexity models (e.g., SVM [220]). For problems with many constraints, the search process will be very sensitive to accuracy of the surrogates that are built on the agregation of objective function and constraints. Hence, techniques that aggregate constraint's surrogates to one or a few have been proposed [280].
  Dealing with constraints for many real-world problems requires incorporating domain knowledge into optimization algorithms. In the Adaptive Reasoning Technique, a search memory is used to learn online the behavior of a greedy algorithm [173]. Some constraints are added to the problem. Those constraints are generated from the non-interesting visited solutions according to the values associated to their decision variables. Similarly to the tabu list strategy in tabu search, those constraints are deleted after a given number of iterations. In Reference [161], the authors use constraints to incorporate domain knowledge into Bayesian networks. They have formally proved that adding constraints reduces variance and improves generalization for machine learning problems.

- **Constraint repairing and decoding:** Some ML approaches have been used to design repairing and decoding procedures for online constraint handling. Deep neural networks [208] and reinforcement learning [101, 282] have been applied to design efficient repairing and decoding procedures. In *cultural algorithms*, online knowledge represented by belief cells (i.e., intervals) is used to avoid infeasible regions and for promoting the exploration of feasible regions [113]. Different constraint handling techniques can be effective during different steps of the search process. ML could also be applied to the online selection of the best constraint handling technique.

3.4 Problem Decomposition

ML approaches can be used in breaking large-scale optimization problems into smaller subproblems. Those subproblems are then solved more efficiently by metaheuristics. Problem decomposition approaches can be carried out in both hierarchical and flat ways. *Hierarchical decomposition* is used when the problem can be successively refined. *Flat decomposition* generates separate subproblems that can be solve in a parallel way. Three types of decomposition strategies may be carried out:

- **Data space decomposition:** Partitioning of data input space can be applied to decompose the input space into subspaces. Metaheuristics are then used to solve simpler subproblems associated each partition. Then, a global solution is built using partial final solutions. In general, offline learning is applied to decompose the problem. For instance, in geographical problems such as vehicle routing [74, 167, 198] and covering [59], clustering techniques exploiting the spatial properties can be applied to structure the set of input nodes. Different clustering algorithms using geographical proximity have been used such as k-means [81, 85], density-based clustering [74], and ANNs [184]. This approach has been also applied to other families of problems such as scheduling [19, 140], in which clustering of jobs is carried out [2].
  In stochastic and robust optimization, the clustering approach can also be applied for the set of input scenario [51, 218]. In Reference [142], a hierarchical clustering approach is proposed to select representative scenario clusters for a robust optimization to avoid redundant simulations and improve robustness in uncertain environments. In Reference [51], clustering has been used to enhance a progressive hedging-based metaheuristic for a network design problem that models demand uncertainty with scenario. The metaheuristic solves successive multi-scenario subproblems associated to different clusters of scenario.
- **Decision space decomposition:** In this problem decomposition approach, the set of decision variables is divided into several subsets to remove the inter-relationship between subsets. The subproblems are assumed to be independent or loosely coupled. A popular offline decomposition technique is time-based decomposition in which the time horizon is used as a splitting criterion [131].
  ML can also be used to fix online some variables to certain values and then solve the reduced associated subproblem. The fixed variables are selected using some knowledge (e.g., elite solutions, interdependance between variables). For instance, this approach is used in *coevolutionary algorithms* for global optimization problems [103, 146], and *Benders decomposition* for mixed optimization problems [247].
- **Objective space decomposition:** In multi-objective optimization problems, ML can be used to decompose the objective space [263]. One can also use ML for reducing the number of objectives. Indeed, the difficulties in solving many-objective optimization problems are the inefficiency of dominance relations, important computational cost and complexity in

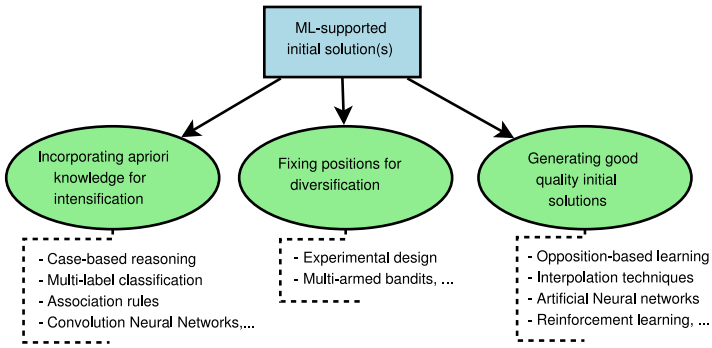

Fig. 6. ML-supported solution(s) initialization.

visualization of the objective space. Reducing the number of objectives using ML approaches represents a popular approach to handle this difficulty: *offline reduction* using *Principle Component Analysis* [214] or *online reduction* based on the clustering of the Pareto front [153, 169]. In Reference [153], an unsupervised clustering algorithm is applied in the objective space at different iterations of the metaheuristic. For each cluster, only representative objectives are selected to guide the search during next iterations.

4 ML IN LOW-LEVEL COMPONENTS OF METAHEURISTICS

Although the effectiveness of metaheuristics has been demonstrated in solving difficult and large-size optimization problems in various areas, the design of the appropriate setting usually requires a deep expert knowledge to obtain competitive results. Here, ML can be used to find good designs for various search components or parameters of metaheuristics such as the initial solutions(s), neighborhoods, variation operators, stopping criteria, and various parameters associated.

4.1 Initial Solution(s)

The most commonly used method in the generation of initial solution(s) is randomness. In general, a population of random solutions is characterized by a bad quality and do not even ensure a good diversification in the search space. ML-assisted initialization techniques can improve the solution quality, reduce the computational costs, and improve the robustness in terms of the variation of the solutions found [122]. Hence, ML has been applied in the initialization of solution(s) following different methodologies (Figure 6):

- **Incorporating *a priori* knowledge for intensification:** Knowledge extracted from previous problems solving can be used to generate solutions that integrate "good" patterns. ML can learn good properties of the obtained solutions and then help to generate good-quality initial solutions in promising regions of the search space. The general learning methodology is shown in Figure 7. First, by selecting a set of training instances of the problem and obtaining a set of elite solutions for this set. Then, some features of the elite set of solutions are found using ML models. Finally, the obtained ML model is used offline to generate initial solutions for a new problem instance. Defining features and similarity metrics for a given problem are the main issues of this methodology. Such approaches have been proposed using case-based reasoning in EAs [144], clustering [183], association rules and attribute-oriented induction [56, 159, 200], multi-label classification using decision trees [139], logistic regression [96, 216], and neural networks [187]. Recently, some deep architectures for neural

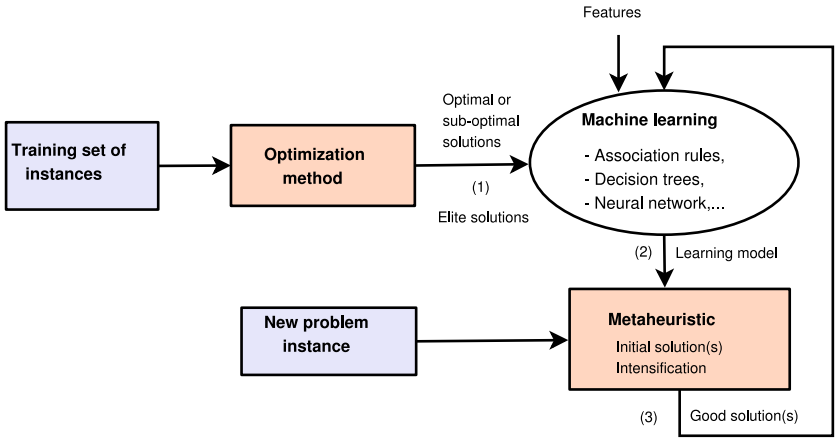

Fig. 7. Extracting knowledge from the history of the search and its use into initial solutions for intensification. (1) Selecting the training problem instances and obtaining the elite solutions (optimal or near-optimal solutions). (2) ML is applied using the elite set of solutions and the selected features of instances. (3) The extracted ML model will be used to generate the initial solutions(s) for solving a new problem instance.

networks have been explored. In Reference [154], Convolution Neural Networks models learn the optimal solution for the traveling salesman problem as an image and extract the patterns (i.e., edges) to be included in a solution. This methodology can also be applied in solving similar problems by applying *transfer learning* approaches. There is an opportunity to exploit the structure of similar problems that have been already solved for which we have a lot of knowledge.

- **Fixing positions for diversification:** Diversification is an important issue in metaheuristics. ML has been applied to improve diversification in P-metaheuristics by using for instance offline *orthogonal experimental design* [136], and in iterative S-metaheuristics by using for instance multi-armed bandits (epsilon-greedy Q-learning) [38]. Indeed, the problem of selecting a region to explore can be defined as a multi-armed bandit: an online reinforcement learning that exemplifies the exploration-exploitation tradeoff dilemma, in which each arm is represented by a region of the search space [248]. This approach can also be used online during the search to guide the underlying metaheuristic toward unvisited regions of the search space. A learning component keeps track of all visited solutions by storing some informations: spheres in the decision space [183], intervals for decision variables [228], **Self-Organizing Maps (SOM)** [9], binary space-partitioning trees [273]. To guide the underlying metaheuristic toward unexplored regions, the learning component generate solutions far away from the visited solutions.
- **Generating good-quality initial solutions:** ML can also help to generate "good"-quality initial solution(s). An example of such offline approaches are:
  - **Opposition-based learning:** using a one-to-one mapping function in the decision space, **opposition-based learning (OBL)** allows to obtain complementary solutions to improve the convergence of the search process and the coverage of the search space. Opposite solutions are generated during the initialization of a population using different types of mapping functions: Type-I opposition, Type-I Quasi-opposition, Type-I Super Opposition, Type-II opposition, Center-based sampling, Generalized OBL, and Quasi-reflection OBL

[203]. This methodology has been carried out in Differential Evolution [4, 20, 191, 242, 254], Evolutionary Algorithms [63, 268], Harmony Search [219, 221], Moth Flame Optimization [211], and PSO [82, 223, 252, 253]. It has also been applied for solving multi-objective optimization problems [135, 145] and noisy problems [95].

— **Artificial neural networks:** different architectures of ANNs have been investigated to generate good-quality solution for optimization problems: Hopfield neural networks [236, 259, 266], Recurrent Neural Networks [21, 160], Pointer networks [249], and SOM [47, 156].

— **Interpolation techniques:** From a given population of solutions, new solutions can be generated using interpolation techniques such as quadratic interpolation [6, 170]. Using a quadratic interpolation of three solutions, a new solution lying at the minimum of the quadratic curve crossing through the three solutions is obtained. It helps to improve the convergence rate of the algorithms.

— **Reinforcement learning:** RL can be seen as a greedy optimization approach in which a complete solution is constructed by successive addition of decisions [124, 125]. Recently, some deep architectures have been investigated such as Deep RL [148], and Deep Q Network [256].

— **Transfer learning:** Transfer learning has been expoited for the population initialization in solving dynamic multi-objective optimization problems [111]. It allows to generate an initial population by reusing past experience to speed up the search process.

4.2 Search Operators Design

This section deals with the application of ML into the design of search operators: constructive (i.e., greedy) operators, unary operators (e.g., neighborhood in local search, mutation in EAs), binary operators (e.g., crossover in EAs), and indirect operators (e.g., probabilistic models in estimation of distribution algorithms) search operators:

- **Constructive operators:** Some metaheuristics such as Ant Colony Optimization and GRASP use greedy operators to generate new solutions. Association rules and clustering (e.g., self-organizing maps) have been used to extract patterns that are incorporated offline into greedy procedures [53, 201]. Reinforcement learning have been integrated into constructive operators, such as Q-learning in ant colony-based optimization [64]. This online approach is very popular in solving control problems such as mobile robots [89]. Indeed reinforcement learning is a natural framework for sequential learning in greedy procedures. Deep RL framework has also been applied to many combinatorial optimization problems such as graph and routing problems [110, 125, 149, 281] and binary problems [43].
- **Unary operators:** One of the key issues in designing efficient unary operators is ensuring that they search in the appropriate neighborhood. Two different issues have been addressed using online learning:
  - **Size of the neighborhood:** The size of the neighborhood can be learned online during search; for instance the step size in Evolution strategies [215].
  - **Sampling of the neighborhood:** ML models can be extracted from a set of good generated solutions to guide the generation of the candidate neighbors. Other than applying a random or complete generation of the neighborhood, learning-based generation is carried out. This methodology has been applied to many metaheuristics such as mutation in EAs [162, 212, 231], and neighborhoods in **Variable Neighborhood Search (VNS)** [239]. In Reference [212], probabilistic models are learned from the population in EAs. New solutions are generated by sampling the probabilistic model. In Reference [241] a linkage tree

genetic algorithm based on hierarchical clustering is proposed. It allows to identify the problem variables that have a high mutual information in a population of good solutions. In the generation of new solutions, these linked variables determine the neighborhood where the metaheuristic searches for better solutions by sampling values for these problem variables from the current population. This neighborhood learning is guided by the linkage hierarchical clustering found so far during the search.

- **Binary operators:** The knowledge extracted online during the search can be incorporated into binary operators such as recombination operators in EAs for the generation of new solutions (Figure 8). From a set of solutions (e.g., current population, elite solutions), some learning models are extracted, that will participate in the following:
  - **Recombination of solutions:** Association rules [210] and decision trees [119] are some examples of ML approaches that have been applied in the crossover operator in EAs. In Reference [210], an elite set of solutions is maintained by an EA. The frequent itemsets are discovered using the Apriori algorithm of association rules. Then those frequent itemsets will guide the crossover operator by a greedy procedure. In Reference [152], a set of decision rules describing the generated solutions are found, using the *AQ learning algorithm*. The extracted rules are incorporated into the crossover operator.
  - **Selection of solutions:** In general, the selection of solutions to be recombined is based only on their qualities (e.g., roulette or tournament selection). The candidates for crossover can be chosen using some distance measures. Hence, hierarchical clustering has been proposed to select solutions in neighboring clusters [5, 171]. Indeed, recombining only neighboring solutions may improve the efficiency of crossover operators [209].
- **Indirect operators:** In blackboard-based metaheuristics (e.g., EDA), online ML models are used to generate new solutions: probabilistic models [177], Bayesian networks [181], incremental learning [13], dependency trees [14], and Markov random fields [217]. Similarly, *cultural algorithms* use good-quality solutions to induce beliefs guiding the generation of solutions by evolutionary operators [199]. Cultural evolution allows populations to learn and adapt faster than biological evolution.

4.3 Search Operators Selection

In many metaheuristics there exists a lot of alternatives in terms of search operators design: neighborhoods and more generally unary operators such as mutation in EAs, and binary operators such as crossover in EAs. ML can be used online for the selection of the best suited search operator. Sequential learning approaches (e.g., reinforcement learning, multi-armed bandit) are well suited for the selection of search operators [67].

Reinforcement learning has been applied to learn the optimal search operator based on the performance of operators. The operator selection problem is formulated as reinforcement learning problem. It is applied to learn optimal policies by maximizing the accumulated rewards. According to the calculated Q function value of each candidate operator, an optimal operator can be selected to maximize the learned Q reward value. This approach has been used for various unary operators such as mutation [276], and neighborhood selection in VNS [41, 65, 130]. VNS integrate a set of different neighborhoods, which are explored in a predefined sequence. ML approaches allows to select the most appropriate neighborhood at a given iteration.

Other sequential learning approaches have been proposed such as *multi-armed bandit* for operator selection in EA [55], and neighborhood selection in VNS [44], and *adaptive pursuit algorithm* [68]. *Ensemble methods* represent alternatives to sequential learning that have been investigated for this task [147].

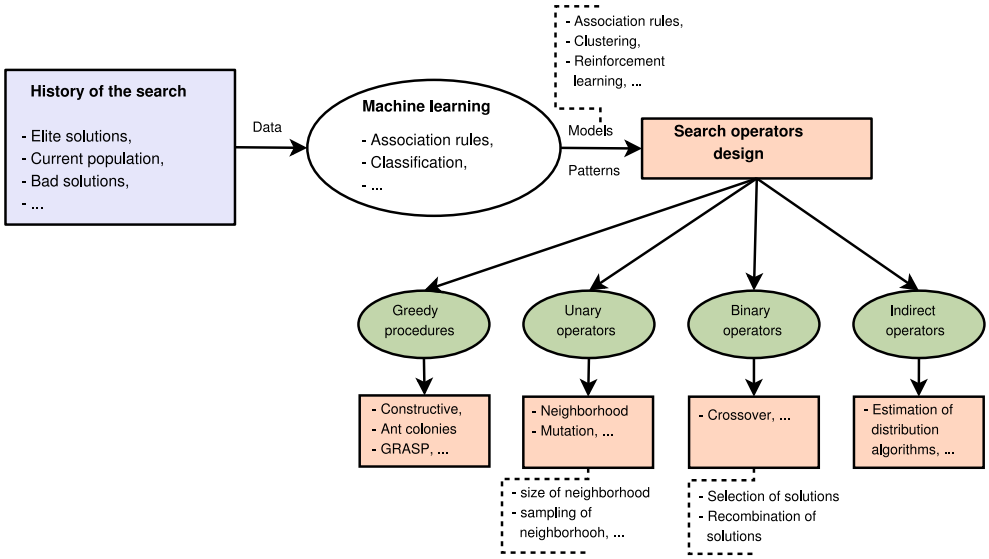

Fig. 8. Extracting knowledge from the search history and its use into search operators.

4.4 Parameter Tuning

Many quantitative parameters compose metaheuristics. Quantitative parameters represent numerical values to be initialized such as the probability of application of variation operators (e.g., mutation, crossover) in EAs, the tabu list size in tabu search, and the velocity in particle swarm optimization. The various parameters of metaheuristics have a significant impact on their efficiency [118]. In general, metaheuristics designers tune the parameters by hand, guided by their experience and some rules of thumb. Finding good parameter setting is a tedious and time-consuming task [72].

There are a lot of similarities between the parameter tuning problem and problems faced in ML [23] and DoE [75]. Offline or static parameter tuning addresses the finding of good parameters before the execution of a metaheuristic [108]. Online or dynamic parameter tuning addresses the dynamic change of parameters during the search. A hybridization of both approaches is generally required for finding satisfactory solutions.

4.4.1 Offline Parameter Tuning. There is no general optimal parameter setting for metaheuristics. For any metaheuristic, an optimal parameter setting can vary considerably depending on the problem, and even between instances of the same problem. Three main ML methodologies can be applied:

- **Unsupervised learning:** Unsupervised learning has been explored to improve the efficiency of factorial experimental design for parameter tuning. The main idea is to reduce the parameter space to reduce the computational complexity: DoE [50], Taguchi fractional experimental design [1], fractional factorial design [90], and correlation graph decomposition [132]. This methodology can be described in an unified way by the following three steps:
  - **Parameter selection:** The main goal of this step is to select the significant parameters that influence the performance of the metaheuristic. Given the input set of parameters, the selection step tries to rank these parameters to determine the importance of parameters.

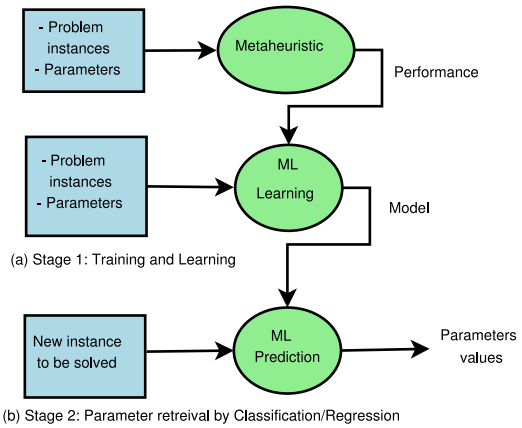

Fig. 9. ML and parameter tuning: training, learning, and prediction.

This step allows to reduce the parameters space to be explored by eliminating some non-significant parameters.

— **Parameter space exploration:** It consists in decomposing the parameter space in different clusters. A model-based approximation (e.g., linear regression [50], response surface [90, 107], logistic regression [192], decision tree [17]) is built to find relationship between the parameters and the objective function value. This step allows to find promising clusters in the parameters space.

— **Parameter space exploitation:** In each cluster, an optimization process is applied to find the best parameters. Any metaheuristic can be used for this step (e.g., local search [50], evolutionary algorithms [132]).

- **Supervised learning:** Optimal parameters setting may differ function of the problem instances of the problem at hand. Supervised learning can be applied to predict the best parameters value for a given instance of the problem. Each instance is characterized by some features [195]. Two different steps can be considered (Figure 9):

— **Training and learning:** For different set of initial parameter values, a metaheuristic is applied to generate solutions for a finite training set of problem instances. Then, the obtained performance data are carried out by ML to build a model.

— **Prediction:** Supervised ML is trained with the data from the first stage to give recommendations for parameters for any new problem instance. ML is therefore used for the recognition of good initial parameter settings for new problem instances. Parameters values for new problem instances can be retrieved in real time using supervised learning, once the training phase is carried out. Different models have been used such as ANN [61], Bayesian networks [174], linear regression [37], and SVM [134].

- **Surrogate-based optimization:** Parameters tuning can be formulated as an expensive optimization problem, in which the decision variables are the parameters, and the objective function is the solution quality obtained by the optimization algorithm. Hence, surrogate-based optimization techniques has been applied to reduce the complexity of the meta-optimization process [260].

4.4.2 Online Parameter Tuning. In online tuning, the parameters are adapted during the search. For instance, the initialization of the probability of application of a given search operator may

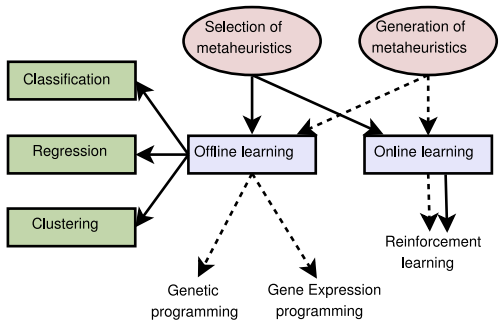

Fig. 10. ML in high-level components of metaheuristics.

be adjusted adaptively by computing the progress of the last applications of the search operator [99, 243]. Hence, it becomes possible to determine the probabilities of application of a given operator in an adaptive manner where the more efficient an operator is, the likelier is its application. Knowledge extracted during the search may serve to dynamically change at run time the values of parameters, using various learning methodologies:

- **Sequential learning approach:** The most popular online approach is sequential learning: adaptive pursuit [240], multi-armed bandit [55], and reinforcement learning [3, 7, 71]. Multi-armed bandit strategies treat each parameter as a separate arm and fix their probabilities by learning the expected rewards [55]. Reinforcement learning uses feedback from the search to define states and actions that represent parameters values [71].
- **Classification/regression approach:** Classifiers (e.g., SVM [274]) or regression models [134] can be used to predict effective parameters settings on the basis of the current status of the search and problem/instance informations.
- **Clustering approach:** Clustering approaches have also been applied. In Reference [277], the distribution of the population in the search space is clustered at each iteration. Using the k-means algorithm, the parameters are adapted according to the knowledge extracted from clusters. Some rules based on the relative size of the cluster containing the best solution and the one containing the worst solution are used to adapt the probabilities of the application of variation operators in EAs.

5 ML IN HIGH-LEVEL COMPONENTS OF METAHEURISTICS

Metaheuristics can be seen as high-level algorithms composed of low-level search operators (e.g., greedy versus iterative). On one hand, many metaheuristic families exist in the literature (e.g., EAs, swarm intelligence, local search) and then various heuristics can be designed. On the other hand, for a given optimization problem, a heuristic can be automatically generated by fixing a configuration from the design space of low-level search operators. According to the nature of the design space of metaheuristics, one can consider the metaheuristic selection problem for selecting existing algorithms, and the metaheuristic generation problem for generating automatically algorithms from the search components of existing ones (Figure 10).

5.1 Selection of Metaheuristics

The design space of metaheuristics is very large. Moreover, the steady development of new metaheuristics makes the selection of the best metaheuristic for a given problem or instance an important challenge [129]. The algorithm selection involves selecting an algorithm from a given portfolio

of algorithms to take into account the varying performance of algorithms over a set of instances and problems [202]. It is well known that a global best metaheuristic for all optimization problems does not exist. We have to accept that no single metaheuristic will produce the best performance on all problems and instances [261]. One is especially interested in giving metaheuristic recommendations for certain families of optimization problems that differ in the kind of exhibited problem features.

The research literature on the algorithm selection problem show the efficiency of using ML for the problem [24, 28]. For the metaheuristic selection problem, one can distinguish between offline and online learning. The advantage of online learning is an adaptive selection of algorithm, whereas the cost for this additional effectiveness is an important overhead:

- **Offline learning:** The idea is to gather knowledge from a set of training instances by using instance features, that will hopefully generalize to solve new instances. One can consider three approaches for the metaheuristic selection problem:
  - **Classification:** A multi-class classifier predict the best performing metaheuristic over the k possible ones. A predictive model is trained on empirical performance data and metaheuristics. Many supervised ML techniques can be used such as ANN [92, 93, 244, 245], Bayesian networks [91], nearest neighbors [84], support vector machines [102], decision trees [48], logistic regression [188], ensembles [226], and deep neural networks [54].
  - **Regression:** The performance of each metaheuristic is predicted via a regression model and then selects the one with the best predicted performance. Several ML regression approaches have been proposed: linear regression [137], support vector regression [25], and Lasso regression [106].
  - **Clustering:** It consists in clustering the problem instances in feature space, then selects the best metaheuristic for each cluster and finally affects to each new instance the metaheuristic associated with the instance's predicted cluster.
- **Online learning:** The learning takes place during the search. For instance, this idea has been widely explored in the *hyper-heuristic* framework [32, 66]. The most used ML strategies are mainly based on sequential learning. In a reinforcement learning formulation of the problem, the metaheuristics represent actions, states correspond to solutions, and the value function (i.e., reward) represents the performance of the metaheuristic, while the environment is represented by the instance problem [205] (Figure 11). Most of the considered RL approaches usually use only a single instance [34, 158, 168, 265]. For a problem domain (i.e., set of instances), the reward can be represented by the average performance over the set of instances. The main issue of different RL algorithms (e.g., temporal difference learning [205]) is to estimate the value functions.

Some hybrid approaches combining offline and online learning have been proposed [155]. By combining metaheuristic portfolios and online selection, the goal is to develop a problem-independent methodology with diverse problem solving capability.

5.2 Generation of Metaheuristics

There are so many various characteristics of problem instances encountered in practice. An automated design of a metaheuristic can be performing well and cost effective for a given problem or instance. The generated metaheuristic can produce better solutions than those obtained by human created metaheuristics. Moreover, the automated process is less demanding on human time and is therefore appealing. One can find offline and online learning strategies for the automatic generation of heuristics:

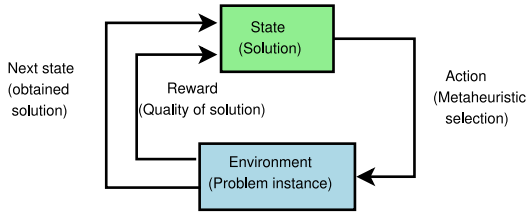

Fig. 11. Online selection of a metaheuristic using a reinforcement learning approach.

- **Offline learning: Genetic programming (GP)** [33, 126, 206] and *Gene Expression Programming* [190, 207, 283] are the most widely used approach for that purpose. The automated design of effective heuristics is carried out via evolution, in which the heuristic is represented by a GP-tree. Other used methodologies for automatically generating a new heuristic for a given problem/instance are based on apprenticeship learning [12], Learning Classifier Systems [165], ANN [166] and logistic regression [164].
- **Online learning:** The most-used online ML approach is RL to generate greedy [69, 70] and iterative metaheuristics [270]. In Reference [270], deep RL using graph neural networks has been used to generate local search heuristics to solve SAT problems. Local search has been formalized as Markov decision process. Generating a good heuristic consists to find an optimal policy π that maximizes the expected accumulated reward obtained by a local search algorithm. This methodology is also popular in swarm of robots, for which at each iteration the best metaheuristic is generated to perform a given number of predefined actions [272].

6 CONCLUSIONS AND PERSPECTIVES

The fields of ML and metaheuristics are increasingly intertwined. In this article, we have investigated the different opportunities for applying ML into metaheuristics. We have defined in a unified way the various synergies that may be achieved. A detailed taxonomy has been proposed according to the concerned search component: target optimization problem and low-level and high-level search components of metaheuristics. Many of those ML-supported metaheuristics have generated high-quality results and represent state-of-the-art optimization algorithms. One has to keep the overhead of learning low. Indeed, the integration of ML techniques in metaheuristics incurs additional costs that have to be considered in the performance evaluation measures.

Research in designing metaheuristics using ML techniques is witnessed to have an important impact in the future. We expect the interplay of metaheuristics and ML will increase. Indeed, ML-supported metaheuristics opens up a wealth of perspectives. From the optimization point of view, investigating the integration of ML into exact optimization techniques (e.g., mathematical programming, branch and bound, dynamic programming, constraint programming) is an important research challenge. Solving more complex optimization problems such as multi-objective optimization, dynamic optimization, optimization under uncertainty, and bi-level optimization, opens also many other research issues.

From the machine learning perspective, the use of more sophisticated and modern ML techniques such as deep learning models will represent an interesting alternative to solve more complex problems. *Transfer learning* can help to reuse the past experience for speeding up the metaheuristic search process for dynamic and cross-domain optimization problems. Indeed, integrating transfer learning in metaheuristics can improve performance and robustness in solving similar and evolving problems and instances.

It could also be interesting to explore the design and implementation of parallel models for ML-supported metaheuristics. High-performance computing is evolving toward Exascale supercomputers composed of millions of cores provided in heterogeneous devices mainly multi-core processors with various architectures, Graphics Processing Units accelerators and Tensor Processing Units and other AI-dedicated Application-Specific integrated Circuits. Finally, the coupling of software frameworks dealing with the two classes of algorithms (i.e., metaheuristics and ML algorithms) is an important issue for the future. This enables to reduce the complexity of developing ML-supported metaheuristics approaches and makes them increasingly popular.

REFERENCES

[1] B. Adenso-Diaz and M. Laguna. 2006. Fine-tuning of algorithms using fractional experimental designs and local search. *Operat. Res.* 54, 1 (2006), 99–114.

[2] M. Adibi and J. Shahrabi. 2014. A clustering-based modified variable neighborhood search algorithm for a dynamic job shop scheduling problem. *Int. J. Adv. Manufact. Technol.* 70, 9 (2014), 1955–1961.

[3] A. Afanasyeva and M. Buzdalov. 2011. Choosing best fitness function with reinforcement learning. In *Proceedings of the 10th International Conference on Machine Learning and Applications*, Vol. 2. 354–357.

[4] M. A. Ahandani. 2016. Opposition-based learning in the shuffled bidirectional differential evolution algorithm. *Swarm Evol. Comput.* 26 (2016), 64–85.

[5] O. Aichholzer, F. Aurenhammer, B. Brandstatter, T. Ebner, H. Krasser, C. Magele, M. Muhlmann, and W. Renhart. 2002. Evolution strategy and hierarchical clustering. *IEEE Trans. Mag.* 38, 2 (2002), 1041–1044.

[6] M. Ali, M. Pant, and A. Abraham. 2013. Unconventional initialization methods for differential evolution. *Appl. Math. Comput.* 219, 9 (2013), 4474–4494.

[7] S. Almahdi and S. Yang. 2019. A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning. *Expert Syst. Appl.* 130 (2019), 145–156.

[8] E. Alpaydin. 2014. *Introduction to Machine Learning*. MIT Press.

[9] H. B. Amor and A. Rettinger. 2005. Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*. 1531–1538.

[10] J. S. Angelo, E. Krempser, and H. Barbosa. 2019. Performance evaluation of local surrogate models in bilevel optimization. In *Proceedings of the International Conference on Machine Learning, Optimization, and Data Science*. 347–359.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Sign. Process. Mag.* 34, 6 (2017), 26–38.

[12] S. Asta, E. Özcan, A. Parkes, and A. Etaner-Uyar. 2013. Generalizing hyper-heuristics via apprenticeship learning. In *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization*. 169–178.

[13] S. Baluja. 1994. *Population-based Incremental Learning. A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Technical Report. Department of Computer Science, Carnegie-Mellon University, Pittsburgh.

[14] S. Baluja and S. Davies. 1997. *Using Optimal Dependency-trees for Combinatorial Optimization: Learning the Structure of the Search Space*. Technical Report. Department of Computer Science, Carnegie-Mellon University, Pittsburgh.

[15] T. Bartz-Beielstein, B. Filipič, P. Korošec, and E-G. Talbi. 2020. *High-Performance Simulation-Based Optimization*. Springer.

[16] T. Bartz-Beielstein and S. Markon. 2004. Tuning search algorithms for real-life applications: Regression tree based approach. In *Proceedings of the Congress on Evolutionary Computation (CEC'2004)*. 1111–1118.

[17] T. Bartz-Beielstein, K. Parsopoulos, and M. N. Vrahatis. 2004. Design and analysis of optimization algorithms using computational statistics. *Appl. Numer. Anal. Comput. Math.* 1, 2 (2004), 413–433.

[18] T. Bartz-Beielstein and M. Preuß. 2014. Experimental analysis of optimization algorithms: Tuning and beyond. In *Theory and Principled Methods for the Design of Metaheuristics*. Springer, 205–245.

[19] M. H. Bassett, J. F. Pekny, and G. V. Reklaitis. 1996. Decomposition techniques for the solution of large-scale scheduling problems. *AIChE J.* 42, 12 (1996), 3373–3387.

[20] M. Basu. 2016. Quasi-oppositional differential evolution for optimal reactive power dispatch. *Int. J. Electr. Power Energy Syst.* 78 (2016), 29–40.

[21] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. 2016. Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940. Retrieved from https://arxiv.org/abs/1611.09940.

[22] K. P. Bennett and E. Parrado-Hernández. 2006. The interplay of optimization and machine learning research. *J. Mach. Learn. Res.* 7 (Jul. 2006), 1265–1281.

[23] M. Birattari. 2009. *Tuning Metaheuristics: A Machine Learning Perspective*. Springer.
[24] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, and K. Tierney. 2016. Aslib: A benchmark library for algorithm selection. *Artif. Intell.* 237 (2016), 41–58.
[25] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß. 2012. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. 313–320.
[26] L. Bottou, F. Curtis, and J. Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Rev.* 60, 2 (2018), 223–311.
[27] J. Boyan and A. W. Moore. 2000. Learning evaluation functions to improve optimization by local search. *J. Mach. Learn. Res.* 1 (Nov. 2000), 77–112.
[28] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. 2008. *Metalearning: Applications to Data Mining*.
[29] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games* 4, 1 (2012), 1–43.
[30] D. Buche, N. Schraudolph, and P. Koumoutsakos. 2005. Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Trans. Syst. Man Cybernet. C* 35, 2 (2005), 183–194.
[31] L. Bui, H. Abbass, and D. Essam. 2005. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*. 779–785.
[32] E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. 2013. Hyper-heuristics: A survey of the state of the art. *J. Operat. Res. Soc.* 64, 12 (2013), 1695–1724.
[33] E. Burke, M. Hyde, G. Kendall, and J. Woodward. 2010. A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics. *IEEE Trans. Evol. Comput.* 14, 6 (2010), 942–958.
[34] E. Burke, G. Kendall, and E. Soubeiga. 2003. A tabu-search hyperheuristic for timetabling and rostering. *J. Heurist.* 9, 6 (2003), 451–470.
[35] X. Cai, L. Gao, X. Li, and H. Qiu. 2019. Surrogate-guided differential evolution algorithm for high dimensional expensive problems. *Swarm Evol. Comput.* 48 (2019), 288–311.
[36] L. Calvet, J. de Armas, D. Masip, and A. Juan. 2017. Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Math.* 15, 1 (2017), 261–280.
[37] M. Caserta and E. Quiñonez. 2009. A cross entropy-Lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Comput. Operat. Res.* 36, 2 (2009), 530–548.
[38] D. Catteeuw, M. Drugan, and B. Manderick. Ljubljana, Slovenia, 2014. Guided Restarts Hill-Climbing. In *Proceedings of the Annual Conference on Parallel Problem Solving from Nature (PPSN'14)*. 313–320.
[39] D. Chafekar, L. Shi, K. Rasheed, and J. Xuan. 2005. Multiobjective GA Optimization Using Reduced Models. *IEEE Trans. Syst. Man Cybernet. C* 35 (06 2005), 261–265.
[40] G. Chandrashekar and F. Sahin. 2014. A survey on feature selection methods. *Comput. Electr. Eng.* 40, 1 (2014), 16–28.
[41] B. Chen, R. Qu, R. Bai, and W. Laesanklang. 2020. A variable neighborhood search algorithm with reinforcement learning for a real-life periodic vehicle routing problem with time windows and open routes. *Rech. Opération.* 54 (2020).
[42] G. Chen et al. 2019. Global and local surrogate-model-assisted differential evolution for waterflooding production optimization. *SPE J.* (2019).
[43] M. Chen, Y. Chen, Y. Du, L. Wei, and Y. Chen. 2020. Heuristic algorithms based on deep reinforcement learning for quadratic unconstrained binary optimization. *Knowl.-Based Syst.* 207 (2020).
[44] Y. Chen, P. Cowling, F. Polack, and P. Mourdjis. 2016. A multi-arm bandit neighbourhood search for routing and scheduling problems. Research Report. University of York.
[45] X. Chou, L. Gambardella, and R. Montemanni. 2019. A metaheuristic algorithm for the probabilistic orienteering problem. In *Proceedings of the 2nd International Conference on Machine Learning and Machine Intelligence*. 30–34.
[46] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. 2019. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Comput.* 23, 9 (2019), 3137–3166.
[47] E. M. Cochrane and J. E. Beasley. 2003. The co-adaptive neural network approach to the Euclidean travelling salesman problem. *Neural Netw.* 16, 10 (2003), 1499–1525.
[48] D. J. Cook and R. C. Varnell. 1997. Maximizing the benefits of parallel search using machine learning. In *Proceedings of the Annual Conferences on Artificial Intelligence and Innovative Applications of Artificial Intelligence (AAAI/IAAI'97)*. 559–564.
[49] D. Corne, C. Dhaenens, and L. Jourdan. 2012. Synergies between operations research and data mining: The emerging use of multi-objective approaches. *Eur. J. Operat. Res.* 221, 3 (2012), 469–479.
[50] S. Coy, B. Golden, G. Runger, and E. Wasil. 2001. Using experimental design to find effective parameter settings for heuristics. *J. Heurist.* 7, 1 (2001), 77–97.

[51] T. G. Crainic, M. Hewitt, and W. Rei. 2014. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Comput. Operat. Res.* 43 (2014), 90–99.
[52] B. Crawford, R. Soto, G. Astorga, J. García, C. Castro, and F. Paredes. 2017. Putting continuous metaheuristics to work in binary search spaces. *Complexity* 2017 (2017), 1–19.
[53] J-C. Créput and A. Koukam. 2008. The memetic self-organizing map approach to the vehicle routing problem. *Soft Comput.* 12, 11 (2008), 1125–1141.
[54] C. Cummins, P. Petoumenos, Z. Wang, and H. Leather. 2017. End-to-end deep learning of optimization heuristics. In *Proceedings of the 26th International Conference on Parallel Architectures and Compilation Techniques (PACT'17)*. 219–232.
[55] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag. 2008. Adaptive operator selection with dynamic multi-armed bandits. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*. 913–920.
[56] F. Dalboni, L . S. Ochi, and L. Drummond. 2003. On improving evolutionary algorithms by using data mining for the oil collector vehicle routing problem. In *Proceedings of the International Network Optimization Conference*. 182–188.
[57] A. Dantas and A. Pozo. 2020. On the use of fitness landscape features in meta-learning based algorithm selection for the quadratic assignment problem. *Theor. Comput. Sci.* 805 (2020), 62–75.
[58] J-A. Mejía de Dios and E. Mezura-Montes. 2020. A surrogate-assisted metaheuristic for bilevel optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'20)*. 629–635.
[59] M. R. de Holanda, A. Plastino, and U. dos Santos Souza. 2020. MineReduce for the minimum weight vertex cover problem. In *Proceedings of the International Conference on Optimization and Learning (OLA'20)*.
[60] C. Dhaenens and L. Jourdan. 2016. *Metaheuristics for Big Data*. John Wiley & Sons.
[61] F. Dobslaw. 2010. A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In *Proceedings of the International Conference on Computer Mathematics and Natural Computing*.
[62] H. Dong and Z. Dong. 2020. Surrogate-assisted Grey wolf optimization for high-dimensional, computationally expensive black-box problems. *Swarm Evol. Comput.* (2020).
[63] X. Dong, S. Yu, Z. Wu, and Z. Chen. 2010. A hybrid parallel evolutionary algorithm based on elite-subspace strategy and space transformation search. In *High Performance Computing and Applications*. Springer, 139–145.
[64] M. Dorigo and L. M. Gambardella. 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* 1, 1 (1997), 53–66.
[65] J. dos Santos, J. D. de Melo, A. Neto, and D. Aloise. 2014. Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. *Expert Syst. Appl.* 41, 10 (2014), 4939–4949.
[66] J. Drake, A. Kheiri, E. Özcan, and E. Burke. 2020. Recent advances in selection hyper-heuristics. *Eur. J. Operat. Res.* 285, 2 (2020), 405–428.
[67] M. Drugan. 2019. Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm Evol. Comput.* 44 (2019), 228–246.
[68] M. Drugan and E-G. Talbi. 2014. Adaptive Multi-operator MetaHeuristics for quadratic assignment problems. In *EVOLVE: A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*. 149–163.
[69] G. Duflo, G. Danoy, E-G. Talbi, and P. Bouvry. 2020. Automated design of efficient swarming behaviours: A Q-learning hyper-heuristic approach. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'20)*. 227–228.
[70] G. Duflo, G. Danoy, E-G. Talbi, and P. Bouvry. 2020. Automating the Design of Efficient Distributed Behaviours for a Swarm of UAVs. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI'20)*.
[71] A. E. Eiben, M. Horvath, W. Kowalczyk, and M. Schut. 2006. Reinforcement learning for online control of evolutionary algorithms. In *Proceedings of the International Workshop on Engineering Self-Organising Applications*. 151–160.
[72] A. E. Eiben and S. K. Smit. 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm Evol. Comput.* 1, 1 (2011), 19–31.
[73] R. S. Engelmore and A. Morgan. 1988. *Blackboard Systems*. Addison-Wesley.
[74] S. Erdoğan and E. Miller-Hooks. 2012. A green vehicle routing problem. *Transport. Res. E: Logist. Transport. Rev.* 48, 1 (2012), 100–114.
[75] L. Eriksson, E. Johansson, N. Kettaneh-Wold, C. Wikström, and S. Wold. 2000. Design of experiments: Principles and Applications. Learn ways AB, Stockholm.
[76] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Zomaya, S. Foufou, and A. Bouras. 2014. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Trans. Emerg. Top. Comput.* 2, 3 (2014), 267–279.
[77] C. Fan, B. Hou, J. Zheng, L. Xiao, and L. Yi. 2020. A surrogate-assisted particle swarm optimization using ensemble learning for expensive problems with small sample datasets. *Appl. Soft Comput.* (2020), 106–142.
[78] L. Feng, Y. Ong, M. Lim, and I. W. Tsang. 2015. Memetic Search With Interdomain Learning: A Realization Between CVRP and CARP. *IEEE Trans. Evol. Comput.* 19, 5 (2015), 644–658.

[79] P. Fournier-Viger, J. Lin, B. Vo, T. Chi, J. Zhang, and H. B. Le. 2017. A survey of itemset mining. *Data Min. Knowl. Discov.* 7, 4 (2017), e1207.
[80] M. Gallagher. 2019. Fitness landscape analysis in data-driven optimization: An investigation of clustering problems. In *Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC'19)*. 2308–2314.
[81] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, and Z. Tang. 2016. Ant colony optimization with clustering for solving the dynamic location routing problem. *Appl. Math. Comput.* 285 (2016), 149–173.
[82] W-F. Gao, S-Y. Liu, and L-L . Huang. 2012. Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Commun. Nonlin. Sci. Numer. Simul.* 17, 11 (2012), 4316–4327.
[83] A. Gaspar-Cunha and A. Vieira. 2004. A hybrid multi-objective evolutionary algorithm using an inverse neural network. In *Hybrid Metaheuristics*. 25–30.
[84] C. Gebruers, A. Guerri, B. Hnich, and M. Milano. 2004. Making choices using structure at the instance level within a case based reasoning framework. In *Proceedings of the International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. 380–386.
[85] S. Geetha, G. Poonthalir, and P. Vanathi. 2009. Improved k-means algorithm for capacitated clustering problem. *INFOCOMP* 8, 4 (2009), 52–59.
[86] A. Geramifard, T. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. How. 2013. A tutorial on linear function approximators for dynamic programming and reinforcement learning. *Found. Trends Mach. Learn.* 6, 4 (2013), 375–451.
[87] T. Goel, R. Haftka, W. Shyy, and N. Queipo. 2007. Ensemble of surrogates. *Struct. Multidiscipl. Optimiz.* 33, 3 (2007), 199–216.
[88] T. Goel, R. Vaidyanathan, R. Haftka, W. Shyy, N. Queipo, and K. Tucker. 2007. Response surface approximation of Pareto optimal front in multi-objective optimization. *Comput. Methods Appl. Mech. Eng.* 196, 4-6 (2007), 879–893.
[89] P. Goyal, H. Malik, and R. Sharma. 2019. Application of evolutionary reinforcement learning (erl) approach in control domain: A review. In *Smart Innovations in Communication and Computational Sciences*. 273–288.
[90] A. Gunawan, H. Lau, and E. Wong. 2013. Real-world parameter tuning using factorial design with parameter decomposition. In *Advances in Metaheuristics*. Springer, 37–59.
[91] H. Guo. 2003. *Algorithm Selection for Sorting and Probabilistic Inference: A Machine Learning-based Approach*. Ph.D. Dissertation. Kensas State University.
[92] J. Gupta, R. Sexton, and E. Tunc. 2000. Selecting scheduling heuristics using neural networks. *INFORMS J. Comput.* 12, 2 (2000), 150–162.
[93] A. Gutierrez-Rodríguez, S. Conant-Pablos, J. Ortiz-Bayliss, and H. Terashima-Marín. 2019. Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning. *Expert Syst. Appl.* 118 (2019), 470–481.
[94] A. Habib, H. Singh, T. Chugh, T. Ray, and K. Miettinen. 2019. A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive multi/many-objective optimization. *IEEE Trans. Evol. Comput.* 23, 6 (2019), 1000–1014.
[95] L. Han and X. He. 2007. A novel opposition-based particle swarm optimization for noisy problems. In *Proceedings of the 3rd IEEE International Conference on Natural Computation (ICNC'07)*, Vol. 3. 624–629.
[96] M. He, P. Kalmbach, A. Blenk, W. Kellerer, and S. Schmid. 2017. Algorithm-data driven optimization of adaptive communication networks. In *Proceedings of the IEEE 25th International Conference on Network Protocols (ICNP'17)*. 1–6.
[97] A. Hebbal, L. Brevault, M. Balesdent, E-G. Talbi, and N. Melab. 2019. Multi-fidelity modeling using DGPs: Improvements and a generalization to varying input space dimensions. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeuIPS'19)*.
[98] A. Hebbal, L. Brevault, M. Balesdent, E-G. Talbi, and N. Melab. 2020. Bayesian Optimization using deep Gaussian processes. *Eng. Optimiz.* (2020), 1–41.
[99] T-P. Hong, H.-S. Wang, and W-C. Chen. 2000. Simultaneous applying multiple mutation operators in genetic algorithm. *J. Heurist.* 6, 4 (2000), 439–455.
[100] Y-S. Hong, H. Lee, and M-J. Tahk. 2003. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Eng. Optimiz.* 35, 1 (2003), 91–102.
[101] A. Hottung and K. Tierney. 2020. Neural large neighborhood search for the capacitated vehicle routing problem. In *Proceedings of the 24th European Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications, Vol. 325. 443–450.
[102] P. D. Hough and P. J. Williams. 2006. *Modern Machine Learning for Automatic Optimization Algorithm Selection*. Technical Report. Sandia National Laboratory (SNL-CA), Livermore, CA.
[103] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang. 2017. Cooperation coevolution with fast interdependency identification for large scale optimization. *Inf. Sci.* 381 (2017), 142–160.
[104] C. Huang, Y. Li, and X. Yao. 2019. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans. Evol. Comput.* 24, 2 (2019), 201–216.

[105] J. Hunger and G. Huttner. 1999. Optimization and analysis of force field parameters by combination of genetic algorithms and neural networks. *J. Comput. Chem.* 20, 4 (1999), 455–471.
[106] F. Hutter, Y. Hamadi, H. Hoos, and K. Leyton-Brown. 2006. Performance prediction and automated tuning of randomized and parametric algorithms. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. Springer, 213–228.
[107] F. Hutter, H. Hoos, and K. Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization*. 507–523.
[108] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. 2009. ParamILS: An automatic algorithm configuration framework. *J. Artif. Intell. Res.* 36 (2009), 267–306.
[109] F. Hutter, L. Xu, H. Hoos, and K. Leyton-Brown. 2014. Algorithm runtime prediction: Methods & evaluation. *Artif. Intell.* 206 (2014), 79–111.
[110] J. James, W. Yu, and J. Gu. 2019. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Trans. Intell. Transport. Syst.* 20, 10 (2019), 3806–3817.
[111] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen. 2018. Transfer learning-based dynamic multiobjective optimization algorithms. *IEEE Trans. Evol. Comput.* 22, 4 (2018), 501–514.
[112] X. Jiang, D. Chafekar, and K. Rasheed. 2003. Constrained multi-objective ga optimization using reduced models. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*. 174–177.
[113] X. Jin and R. Reynolds. 1999. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: A cultural algorithm approach. In *Proceedings of the 1999 Congress on Evolutionary Computation(CEC'99)*, Vol. 3. 1672–1678.
[114] Y. Jin. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.* 9, 1 (2005), 3–12.
[115] Y. Jin, M. Olhofer, and B. Sendhoff. 2002. A framework for evolutionary optimization with approximate fitness functions. *IEEE Trans. Evol. Comput.* 6, 5 (2002), 481–494.
[116] Y. Jin and B. Sendhoff. 2004. Reducing fitness evaluations using clustering techniques and neural network ensembles. In *Proceedings of the Genetic and Evolutionary Computation (GECCO'04)*, Lecture Notes in Computer Science, Vol. 3102. Springer, 688–699.
[117] D. Jones, M. Schonlau, and W. J. Welch. 1998. Efficient global optimization of expensive black-box functions. *J. Global Optimiz.* 13, 4 (1998), 455–492.
[118] K. De Jong. 2007. Parameter setting in EAs: A 30 year perspective. In *Parameter Setting in Evolutionary Algorithms*. 1–18.
[119] L. Jourdan, D. Corne, D. Savic, and G. Walters. 2005. Preliminary investigation of the learnable evolution model for faster/better multiobjective water systems design. In *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*. 841–855.
[120] L. Jourdan, C. Dhaenens, and E-G. Talbi. 2006. Using data mining techniques to help metaheuristics: A short survey. In *Proceedings of the Annual Conference on Hybrid Metaheuristics (HM'06)*,Lecture Notes in Computer Science, Vol. 4030. Gran Canaria, Spain, 57–69.
[121] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal. 2013. Learning objective functions for manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 1331–1336.
[122] B. Kazimipour, X. Li, and A. Qin. 2014. A review of population initialization techniques for evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'14)*. 2585–2592.
[123] P. Kerschke and H. Trautmann. 2019. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evol. Comput.* 27, 1 (2019), 99–127.
[124] H. Khadilkar. 2018. A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Trans. Intell. Transport. Syst.* 99 (2018), 1–11.
[125] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. 2017. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*. 6348–6358.
[126] E. Kieffer, G. Danoy, M. Brust, P. Bouvry, and A. Nagih. 2019. Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic. *IEEE Trans. Evol. Comput.* (2019).
[127] H.-S. Kim and S.-B. Cho. 2001. An efficient genetic algorithm with less fitness evaluation by clustering. In *Proceedings of the Congress on Evolutionary Computation (CEC'01)*. IEEE Press, 887–894.
[128] J. Knowles. 2006. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evol. Comput.* 10, 1 (2006), 50–66.
[129] L. Kotthoff. 2016. Algorithm selection for combinatorial search problems: A survey. In *Data Mining and Constraint Programming*. Springer, 149–190.
[130] P. Laborie and D. Godard. 2007. Self-adapting large neighborhood search: Application to single-mode scheduling problems. *Proceedings of the Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA'07)*.

[131] W. Laesanklang and D. Landa-Silva. 2017. Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. *Ann. Operat. Res.* 256, 1 (2017), 93–127.
[132] H. C. Lau and F. Xiao. 2009. Enhancing the speed and accuracy of automated parameter tuning in heuristic design. (2009).
[133] H. M. Lee, D. Jung, A. Sadollah, and J. H. Kim. 2019. Performance comparison of metaheuristic algorithms using a modified Gaussian fitness landscape generator. *Soft Comput.* (2019), 1–11.
[134] S. Lessmann, M. Caserta, and I. M. Arango. 2011. Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Syst. Appl.* 38, 10 (2011), 12826–12838.
[135] S. W. Leung, X. Zhang, and S. Y. Yuen. 2012. Multiobjective differential evolution algorithm with opposition-based parameter control. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'12).* 1–8.
[136] Y.-W. Leung and Y. Wang. 2001. An orthogonal genetic algorithm with quantization for global numerical optimization. *IEEE Trans. Evol. Comput.* 5, 1 (2001), 41–53.
[137] K. Leyton-Brown, E. Nudelman, and Y. Shoham. 2002. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. 556–572.
[138] K. Leyton-Brown, E. Nudelman, and Y. Shoham. 2009. Empirical hardness models: Methodology and a case study on combinatorial auctions. *J. ACM* 56, 4 (2009), 1–52.
[139] X. Li and S. Olafsson. 2005. Discovering dispatching rules using data mining. *J. Schedul.* 8, 6 (2005), 515–527.
[140] L. Liu and M. Dessouky. 2017. A decomposition based hybrid heuristic algorithm for the joint passenger and freight train scheduling problem. *Comput. Operat. Res.* 87 (2017), 165–182.
[141] N. Liu, J.-S. Pan, C. Sun, and S.-C. Chu. 2020. An efficient surrogate-assisted quasi-affine transformation evolutionary algorithm for expensive optimization problems. *Knowl.-Based Syst.* 209 (2020).
[142] Z. Liu and F. Forouzanfar. 2018. Ensemble clustering for efficient robust optimization of naturally fractured reservoirs. *Comput. Geosci.* 22, 1 (2018), 283–296.
[143] I. Loshchilov, M. Schoenauer, and M. Sebag. 2012. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation.* 321–328.
[144] S. Louis and J. McDonnell. 2004. Learning with case-injected genetic algorithms. *IEEE Trans. Evol. Comput.* 8, 4 (2004), 316–328.
[145] X. Ma, F. Liu, Y. Qi, M. Gong, M. Yin, L. Li, L. Jiao, and J. Wu. 2014. MOEA/D with opposition-based learning for multiobjective optimization problem. *Neurocomputing* 146 (2014), 48–64.
[146] S. Mahdavi, S. Rahnamayan, and M. E. Shiri. 2018. Incremental cooperative coevolution for large-scale global optimization. *Soft Comput.* 22, 6 (2018), 2045–2064.
[147] R. Mallipeddi and P. N. Suganthan. 2010. Ensemble of constraint handling techniques. *IEEE Trans. Evol. Comput.* 14, 4 (2010), 561–579.
[148] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. 2016. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks.* 50–56.
[149] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev. 2020. Reinforcement learning for combinatorial optimization: A survey. arXiv:2003.03600. Retrieved from https://arxiv.org/abs/2003.03600.
[150] S. Meisel and D. C. Mattfeld. 2007. Synergies of data mining and operations research. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07).* 56–56.
[151] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. 2011. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation.* 829–836.
[152] R. S. Michalski. 2000. Learnable evolution model: Evolutionary processes guided by machine learning. *Mach. Learn.* 38, 1 (2000), 9–40.
[153] E. Mihaela and I. Adrian. 2015. Dynamic Objective Sampling in Many-objective Optimization. *Proc. Comput. Sci.* 60 (2015), 178–187.
[154] S. Miki, D. Yamamoto, and H. Ebara. 2018. Applying Deep Learning and Reinforcement Learning to Traveling Salesman Problem. In *Proceedings of the International Conference on Computing, Electronics & Communications Engineering (iCCECE'18).* 65–70.
[155] M. Mısır, S. Handoko, and H. C. Lau. 2015. OSCAR: Online selection of algorithm portfolios with case study on memetic algorithms. In *Proceedings of the International Conference on Learning and Intelligent Optimization.* 59–73.
[156] A. Modares, S. Somhom, and T. Enkawa. 1999. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. *Int. Trans. Operat. Res.* 6, 6 (1999), 591–606.
[157] J. Mueller and J. Woodbury. 2017. GOSAC: Global optimization with surrogate approimation of constraints. *J. Global Optimiz.* 69 (01 2017).
[158] A. Nareyek. 2003. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer Decision-making*. Springer, 523–544.

[159] M. M. Nasiri, S. Salesi, A. Rahbari, N. S. Meydani, and M. Abdollai. 2018. A data mining approach for population-based methods to solve the JSSP. *Soft Comput.* (2018), 1–16.
[160] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. 2018. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*. 9839–9849.
[161] R. S. Niculescu, T. Mitchell, and R. B. Rao. 2006. Bayesian network learning with parameter constraints. *J. Mach. Learn. Res.* 7 (Jul. 2006), 1357–1383.
[162] K. Nitisiri, M. Gen, and H. Ohwada. 2019. A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling. *Comput. Industr. Eng.* 130 (2019), 381–394.
[163] J. Oliveira, M. Almeida, R. Santos, R. de Gusmão, and A. Britto. 2020. New surrogate approaches applied to meta-heuristic algorithms. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*. 400–411.
[164] J. Ortiz-Bayliss, H. Terashima-Marín, and S. Conant-Pablos. 2013. A supervised learning approach to construct hyper-heuristics for constraint satisfaction. In *Proceedings of the Mexican Conference on Pattern Recognition*. 284–293.
[165] J. Ortiz-Bayliss, H. Terashima-Marín, and S. Conant-Pablos. 2013. Using learning classifier systems to design selective hyper-heuristics for constraint satisfaction problems. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC'13)*. 2618–2625.
[166] J. Ortiz-Bayliss, H. Terashima-Marín, P. Ross, and S. Conant-Pablos. 2011. Evolution of neural networks topologies and learning parameters to produce hyper-heuristics for constraint satisfaction problems. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*. 261–262.
[167] A. Ostertag, K. Doerner, and R. Hartl. 2008. A variable neighborhood search integrated in the POPMUSIC framework for solving large scale vehicle routing problems. In *Proceedings of the International Workshop on Hybrid Metaheuristics*. 29–42.
[168] E. Özcan, M. Misir, G. Ochoa, and E. Burke. 2012. A reinforcement learning: Great-deluge hyper-heuristic for examination timetabling. In *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*. 34–55.
[169] M. Pal, S. Saha, and S. Bandyopadhyay. 2018. DECOR: Differential evolution using clustering based objective reduction for many-objective optimization. *Inf. Sci.* 423 (2018), 200–218.
[170] M. Pant, M. Ali, and V. Singh. 2009. Differential evolution using quadratic interpolation for initializing the population. In *Proceedings of the IEEE International Advance Computing Conference*. 375–380.
[171] S.-Y. Park and J.-J. Lee. 2009. Improvement of a multi-objective differential evolution using clustering algorithm. In *Proceedings of the IEEE International Symposium on Industrial Electronics*. 1213–1217.
[172] J. M. Parr, C. M. E. Holden, A. I. J. Forrester, and A. J. Keane. 2010. Review of efficient surrogate infill sampling criteria with constraint handling.
[173] R. Patterson, E. Rolland, and H. Pirkul. 1999. A memory adaptive reasoning technique for solving the capacitated minimum spanning tree problem. *J. Heurist.* 5 (1999), 159–180.
[174] R. Pavón, F. Díaz, R. Laza, and M. V. Luzón. 2009. Automatic parameter tuning with a Bayesian case-based reasoning system. A case of study. *Expert Syst. Appl.* 36, 2 (2009), 3407–3420.
[175] Y. Pei. 2020. Trends on fitness landscape analysis in evolutionary computation and meta-heuristics. In *Frontier Applications of Nature Inspired Computation*. Springer, 78–99.
[176] J. Pelamatti, L. Brevault, M. Balesdent, E-G. Talbi, and Y. Guerin. 2019. Efficient global optimization of constrained mixed variable problems. *J. Global Optimiz.* 73, 3 (2019), 583–613.
[177] M. Pelikan, D. Goldberg, and F. Lobo. 2002. A survey of optimization by building and using probabilistic models. *Comput. Optimiz. Appl.* 21, 1 (2002), 5–20.
[178] M. Pelikan and K. Sastry. 2004. Fitness inheritance in the Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'04)*. 48–59.
[179] M. Pelikan, K. Sastry, and D. Goldberg. 2005. Multiobjective hBOA, clustering, and scalability. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05)*.
[180] J. Pena, J. Lozano, and P. Larranaga. 2005. Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. *Evol. Comput.* 13 (03 2005), 43–66.
[181] M. Pelikan, D. Goldberg, and E. Cantu-Paz. 2000. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation* 8, 3 (2000), 311–340.
[182] A. Polydoros and L. Nalpantidis. 2017. Survey of model-based reinforcement learning: Applications on robotics. *J. Intell. Robot. Syst.* 86, 2 (2017), 153–173.
[183] D. Porumbel, J-K. Hao, and P. Kuntz. 2010. A search space cartography for guiding graph coloring heuristics. *Comput. Operat. Res.* 37, 4 (2010), 769–778.
[184] J.-Y. Potvin and R. S. Thangiah. 2020. Vehicle routing through simulation. *Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications* (2020).

[185] W. B. Powell. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Vol. 703. John Wiley & Sons.
[186] R. Priem, N. Bartoli, and Y. Diouane. 2019. On the use of upper trust bounds in constrained Bayesian optimization infill criteria. In *Proceedings of the AIAA Aviation Forum.* 2986–2999.
[187] P. Priore, D. de la Fuente, J. Puente, and J. Parreño. 2006. A comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems. *Eng. Appl. Artif. Intell.* 19, 3 (2006), 247–255.
[188] L. Pulina and A. Tacchella. 2007. A multi-engine solver for quantified boolean formulas. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming.* 574–589.
[189] S. Qin, C. Sun, Y. Jin, and G. Zhang. 2019. Bayesian approaches to surrogate-assisted evolutionary multi-objective optimization: A comparative study. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI'19).* 2074–2080.
[190] A. Rahati and H. Rakhshani. 2016. A gene expression programming framework for evolutionary design of metaheuristic algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'16).* 1445–1452.
[191] S. Rahnamayan, H. Tizhoosh, and M. Salama. 2008. Opposition-based differential evolution. *IEEE Trans. Evol. Comput.* 12, 1 (2008), 64–79.
[192] I. Ramos, M. C. Goldbarg, E. Goldbarg, and A. D. Neto. 2005. Logistic regression for parameter tuning on an evolutionary algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, Vol. 2. 1061–1068.
[193] K. Rasheed and H. Hirsh. 2000. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO'00).* 628–635.
[194] K. Rasheed, S. Vattam, and X. Ni. 2002. Comparison of methods for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'02).* 390–395.
[195] J. Rasku, T. Kärkkäinen, and N. Musliu. 2016. Feature extractors for describing vehicle routing problem instances. In *Proceedings of the 5th Student Conference on Operational Research (SCOR'16)*, B. Hardy, A. Qazi, and S. Ravizza (Eds.), Vol. 50. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
[196] R. G. Regis. 2014. Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. *IEEE Trans. Evol. Comput.* 18, 3 (2014), 326–347.
[197] R. G. Regis and C. A. Shoemaker. 2004. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Trans. Evol. Comput.* 8, 5 (2004), 490–505.
[198] M. Reimann, K. Doerner, and R. Hartl. 2004. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research* 31 (04 2004), 563–591.
[199] R. G. Reynolds, Z. Michalewicz, and B. Peng. 2005. Cultural algorithms: Computational modeling of how cultures learn to solve problems- an engineering example. *Cybernet. Syst.* 36, 8 (2005), 753–771.
[200] M. H. Ribeiro, A. Plastino, and S. L. Martins. 2006. Hybridization of GRASP metaheuristic with data mining techniques. *J. Math. Model. Algor.* 5, 1 (2006), 23–41.
[201] M. H. Ribeiro, V. Trindade, A. Plastino, and S. L. Martins. 2004. Hybridization of GRASP metaheuristics with data mining techniques. In *Hybrid Metaheuristics.*
[202] J. R. Rice. 1976. The algorithm selection problem. *Adv. Comput.* 15 (1976), 65–118.
[203] N. Rojas-Morales, M.-C. Rojas, and E. M. Ureta. 2017. A survey and classification of opposition-based metaheuristics. *Comput. Industr. Eng.* 110 (2017), 424–435.
[204] R. G. Rommel. 2014. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Eng. Optimiz.* 46, 2 (2014), 218–243.
[205] T. P. Runarsson. 2011. Learning heuristic policies - A reinforcement learning problem. In *Proceedings of the International Conference on Learning and Intelligent Optimization.* 423–432.
[206] N. Sabar, M. Ayob, G. Kendall, and R. Qu. 2015. Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Trans. Evol. Comput.* 19, 3 (2015), 309–325.
[207] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu. 2014. Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Trans. Evol. Comput.* 19, 3 (2014), 309–325.
[208] N. Sakamoto, E. Semmatsu, K. Fukuchi, J. Sakuma, and Y. Akimoto. 2020. Deep generative model for non-convex constraint handling. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'20).* 636–644.
[209] M. Samorani, Y. Wang, Z. Lv, and F. Glover. 2019. Clustering-driven evolutionary algorithms: An application of path relinking to the quadratic unconstrained binary optimization problem. *J. Heurist.* 25, 4-5 (2019), 629–642.
[210] H. G. Santos, L. S. Ochi, E. H. Marinho, and L. M. Drummond. 2006. Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing* 70, 1-3 (2006), 70–77.
[211] S. Sapre and S. Mini. 2019. Opposition-based moth flame optimization with Cauchy mutation and evolutionary boundary constraint handling for global optimization. *Soft Comput.* 23, 15 (2019), 6023–6041.

[212] K. Sastry and D. Goldberg. 2004. Designing competent mutation operators via probabilistic model building of neighborhoods. In *Proceedings of the Genetic and Evolutionary Computation (GECCO'04)*, K. Deb (Ed.). 114–125.
[213] K. Sastry, D. Goldberg, and M. Pelikan. 2001. Don't evaluate, inherit. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation (GECCO'01)*. 551–558.
[214] D. Saxena, J. Duro, A. Tiwari, K. Deb, and Q. Zhang. 2013. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Trans. Evol. Comp* 17, 1 (2013), 77–99.
[215] M. Sebag, M. Schoenauer, and C. Ravise. 1997. Inductive learning of mutation step-size in evolutionary parameter optimization. In *Evolutionary Programming*.
[216] A. Shahzad and N. Mebarki. 2012. Data mining based job dispatching using hybrid simulation-optimization approach for shop scheduling problem. *Eng. Appl. Artif. Intell.* 25, 6 (2012), 1173–1181.
[217] S. Shakya and J. McCall. 2007. Optimization by estimation of distribution with DEUM framework based on Markov random fields. *Int. J. Automat. Comput.* 4, 3 (2007), 262–272.
[218] C. Shang and F. You. 2019. A data-driven robust optimization approach to scenario-based stochastic model predictive control. *J. Process Contr.* 75 (2019), 24–39.
[219] G. Shankar and V. Mukherjee. 2016. Quasi oppositional harmony search algorithm based controller tuning for load frequency control of multi-source multi-area power system. *Int. J. Electr. Power Energy Syst.* 75 (2016), 289–302.
[220] L. Shi and K. Rasheed. 2008. ASAGA: An adaptive surrogate-assisted genetic algorithm. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. 1049–1056.
[221] C. K. Shiva, G. Shankar, and V. Mukherjee. 2015. Automatic generation control of power system using a novel quasi-oppositional harmony search algorithm. *Int. J. Electr. Power Energy Syst.* 73 (2015), 787–804.
[222] Y. Shiyou, Q. Liu, J. Lu, S. L. Ho, G. Ni, P. Ni, and S. Xiong. 2009. Application of support vector machines to accelerate the solution speed of metaheuristic algorithms. *IEEE Trans. Magn.* 45 (04 2009), 1502–1505.
[223] T. Si, A. De, and A. K. Bhattacharjee. 2014. Particle swarm optimization with generalized opposition based learning in particle's pbest position. In *Proceedings of the International Conference on Circuits, Power and Computing Technologies (ICCPCT'14)*. 1662–1667.
[224] R. Smith, B. Dike, and S. A. Stegmann. 1995. Fitness inheritance in genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing*. 345–350.
[225] K. Smith-Miles and J. van Hemert. 2011. Discovering the suitability of optimisation algorithms by learning from evolved instances. *Ann. Math. Artif. Intell.* 61, 2 (2011), 87–104.
[226] K. A. Smith-Miles. 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.* 41, 1 (2009), 1–25.
[227] K. Socha and M. Dorigo. 2008. Ant colony optimization for continuous domains. *Eur. J. Operat. Res.* 185, 3 (2008), 1155–1173.
[228] C. Soza, R. Becerra, M. C. Riff, and C. Coello. 2011. Solving timetabling problems using a cultural algorithm. *Appl. Soft Comput.* 11, 1 (2011), 337–344.
[229] S. Sra, S. Nowozin, and S. Wright. 2012. *Optimization for Machine Learning*. MIT Press.
[230] S. Srivastava, B. Pathak, and K. Srivastava. 2008. Neural network embedded multiobjective genetic algorithm to solve non-linear time-cost tradeoff problems of project scheduling. *J. Sci. Industr. Res.* 67 (2008), 124–131.
[231] J. Sun, H. Zhang, A. Zhou, Q. Zhang, and K. Zhang. 2019. A new learning-based adaptive multi-objective evolutionary algorithm. *Swarm Evol. Comput.* 44 (2019), 304–319.
[232] R. Sutton, A. Barto, et al. 1998. *Introduction to Reinforcement Learning*. Vol. 135.
[233] R. S. Sutton. 1988. Learning to predict by the methods of temporal differences. *Mach. Learn.* 3 (1988), 9–44.
[234] E.-G. Talbi. 2009. *Metaheuristics: From Design to Implementation*. Wiley.
[235] E.-G. Talbi. 2016. Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Ann. Operat. Res.* 240, 1 (2016), 171–215.
[236] K. C. Tan, H. Tang, and S. S. Ge. 2005. On parameter settings of Hopfield networks applied to traveling salesman problems. *IEEE Trans. Circ. Syst. I* 52, 5 (2005), 994–1002.
[237] J. Tao and G. Sun. 2019. Application of deep learning based multi-fidelity surrogate model to robust aerodynamic design optimization. *Aerosp. Sci. Technol.* 92 (2019), 722–737.
[238] Y. Tenne and C.-K. Goh. 2010. *Computational Intelligence in Expensive Optimization Problems*. Vol. 2. Springer Science & Business Media.
[239] S. Thevenin and N. Zufferey. 2019. Learning Variable Neighborhood Search for a scheduling problem with time windows and rejections. *Discr. Appl. Math.* 261 (2019), 344–353.
[240] D. Thierens. 2005. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*. 1539–1546.
[241] D. Thierens and P. Bosman. 2012. Learning the Neighborhood with the Linkage Tree Genetic Algorithm. In *Learning and Intelligent Optimization*, Y. Hamadi and M. Schoenauer (Eds.). 491–496.

[242] H. Tizhoosh, M. Ventresca, and S. Rahnamayan. 2008. Opposition-based computing. In *Oppositional Concepts in Computational Intelligence*. 11–28.
[243] A. Tuson and P. Ross. 1998. Adapting operator settings in genetic algorithms. *Evol. Comput.* 6, 2 (1998), 161–184.
[244] D. Tuzun, M. A. Magent, and L. I. Burke. 1997. Selection of vehicle routing heuristic using neural networks. *Int. Trans. Operat. Res.* 4, 3 (1997), 211–221.
[245] R. Tyasnurita, E. Özcan, and R. John. 2017. Learning heuristic selection using a time delay neural network for open vehicle routing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'17)*. 1474–1481.
[246] H. Ulmer, F. Streichert, and A. Zell. 2003. Evolution strategies assisted by Gaussian processes with improved preselection criterion. In *Proceedings of the Congress on Evolutionary Computation (CEC'03)*. 692–699.
[247] F. Vanderbeck and L. A. Wolsey. 2010. Reformulation and decomposition of integer programs. In *50 Years of Integer Programming 1958-2008*. 431–502.
[248] J. Vermorel and M. Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the European Conference on Machine Learning (ECML'05)*. 437–448.
[249] O. Vinyals, M. Fortunato, and N. Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. 2692–2700.
[250] C. Voudouris and E. Tsang. 2003. Guided local search. In *Handbook of Metaheuristics*. 185–218.
[251] G. Wang and S. Shan. 2007. Review of metamodeling techniques in support of engineering design optimization. *J. Mech. Des.* 129, 4 (2007), 370–380.
[252] H. Wang, H. Li and Y. Liu, C. Li, and S. Zeng. 2007. Opposition-based particle swarm algorithm with Cauchy mutation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'07)*. 4750–4756.
[253] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca. 2011. Enhancing particle swarm optimization using generalized opposition-based learning. *Inf. Sci.* 181, 20 (2011), 4699–4714.
[254] J. Wang. 2015. Enhanced differential evolution with generalised opposition–based learning and orientation neighbourhood mining. *Int. J. Comput. Sci. Math.* 6, 1 (2015), 49–58.
[255] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer. 2020. An adaptive Bayesian approach to surrogate-assisted evolutionary multi-objective optimization. *Inf. Sci.* 519 (2020), 317–331.
[256] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek. 2018. Optimization of global production scheduling with deep reinforcement learning. *Proc. CIRP* 72 (2018), 1264–1269.
[257] C. Watkins and P. Dayan. 1992. Q-learning. *Mach. Learn.* 8, 3-4 (1992), 279–292.
[258] W. J. Welch and M. Schonlau. 1997. Computer experiments and global optimization.
[259] U.-P. Wen, K.-M. Lan, and H.-S. Shih. 2009. A review of Hopfield neural networks for solving mathematical programming problems. *Eur. J. Oper. Res.* 198, 3 (2009), 675–687.
[260] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. 2018. Scalable Gaussian process-based transfer surrogates for hyperparameter optimization. *Mach. Learn.* 107, 1 (2018), 43–78.
[261] D. Wolpert and W. Macready. 1997. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* 1, 1 (1997), 67–82.
[262] G. Wu, R. Mallipeddi, and P. N. Suganthan. 2019. Ensemble strategies for population-based optimization algorithms–A survey. *Swarm Evol. Comput.* 44 (2019), 695–711.
[263] M. Wu, K. Li, S. Kwong, Q. Zhang, and J. Zhang. 2018. Learning to decompose: A paradigm for decomposition-based multiobjective optimization. *IEEE Trans. Evol. Comput.* 23, 3 (2018), 376–390.
[264] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. 2008. SATzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Intell. Res.* 32 (2008), 565–606.
[265] Y. Xu, D. Stern, and H. Samulowitz. 2009. Learning adaptation to solve constraint satisfaction problems. In *Proceedings of the International Conference on Learning and Intelligent Optimization (LION'09)*.
[266] T. Yalcinoz, B. J. Cory, and M. J. Short. 2001. Hopfield neural network approaches to economic dispatch problems. *International Journal of Electrical Power & Energy Systems* 23 (2001), 435–442.
[267] C. Yang, J. Ding, Y. Jin, and T. Chai. 2019. Offline data-driven multiobjective optimization: Knowledge transfer between surrogates and generation of final solutions. *IEEE Trans. Evol. Comput.* 24, 3 (2019), 409–423.
[268] S. Yazdani and J. Shanbehzadeh. 2015. Balanced cartesian genetic programming via migration and opposition-based learning: Application to symbolic regression. *Genet. Program. Evolv. Mach.* 16, 2 (2015), 133–150.
[269] J. Yi, Y. Shen, and C. Shoemaker. 2020. A multi-fidelity RBF surrogate-based optimization framework for computationally expensive multi-modal problems with application to capacity planning of manufacturing systems. *Struct. Multidisc. Optimiz.* (2020), 1–21.
[270] E. Yolcu and B. Poczos. 2019. Learning local search heuristics for boolean satisfiability. In *Advances in Neural Information Processing Systems*. 7990–8001.
[271] S.-H. Yoo and S.-B. Cho. 2004. Partially evaluated genetic algorithm based on fuzzy c-means algorithm. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*. 440–449.

[272] S. Yu, A. Aleti, J. C. Barca, and A. Song. 2018. Hyper-heuristic online learning for self-assembling swarm robots. In *Proceedings of the International Conference on Computational Science*. 167–180.
[273] S. Y. Yuen and C. K. Chow. 2009. A genetic algorithm that adaptively mutates and never revisits. *IEEE Trans. Evol. Comput.* 13, 2 (2009), 454–472.
[274] M. Zennaki and A. Ech-Cherif. 2010. A new machine learning based approach for tuning metaheuristics for the solution of hard combinatorial optimization problems. *J. Appl. Sci.* 10, 18 (2010), 1991–2000.
[275] D. Zhan, Y. Cheng, and J. Liu. 2017. Expected improvement matrix-based infill criteria for expensive multiobjective optimization. *IEEE Trans. Evol. Comput.* 21, 6 (2017), 956–975.
[276] H. Zhang and J. Lu. 2008. Adaptive evolutionary programming based on reinforcement learning. *Inf. Sci.* 178, 4 (2008), 971–984.
[277] J. Zhang, H. Chung, and W.-L. Lo. 2007. Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. *IEEE Trans. Evol. Comput.* 11, 3 (2007), 326–335.
[278] J. Zhang, Y.-S. Yim, and J. Yang. 1997. Intelligent selection of instances for prediction functions in lazy learning algorithms. In *Lazy Learning*. Springer, 175–191.
[279] J. Zhang, Z.-H. Zhan, Y. Lin, N. Chen, Y.-J. Gong, J.-H. Zhong, H. Chung, Y. Li, and Y.-H. Shi. 2011. Evolutionary computation meets machine learning: A survey. *IEEE Comput. Intell. Mag.* 6, 4 (2011), 68–75.
[280] K.-S. Zhang, Z.-H. Han, Z.-J. Gao, and Y. Wang. 2019. Constraint aggregation for large number of constraints in wing surrogate-based optimization. *Struct. Multidisc. Optimiz.* 59, 2 (2019), 421–438.
[281] R. Zhang, A. Prokhorchuk, and J. Dauwels. 2020. Deep reinforcement learning for traveling salesman problem with time windows and rejections. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'20)*. 1–8.
[282] W. Zhang and T. Dietterich. 1995. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, Vol. 95. 1114–1120.
[283] J. Zhong, L. Feng, and Y.-S. Ong. 2017. Gene expression programming: A survey. *IEEE Computat. Intell. Mag.* 12, 3 (2017), 54–72.
[284] Z. Zhou, Y. Ong, N. Hanh, and D. Lim. 2005. A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, 2832–2839.

Received March 2020; revised February 2021; accepted March 2021