Noname manuscript No.
(will be inserted by the editor)

A Review of the Role of Heuristics in Stochastic Optimisation: from metaheuristics to learnheuristics

Angel A. Juan · Peter Keenan · Rafael Martí · Seán McGarraghy · Javier Panadero · Paula Carroll · Diego Oliva

Received: date / Accepted: date

Abstract In the context of simulation-based optimisation, this paper reviews recent work related to the role of metaheuristics, matheuristics (combinations of exact optimisation methods with metaheuristics), simheuristics (hybridisation of simulation with metaheuristics), biased-randomised heuristics for 'agile' optimisation via parallel computing, and learnheuristics (combination of statistical / machine learning with metaheuristics) to deal with *NP-hard* and large-scale optimisation problems in areas such as transport & logistics, manufacturing & production, smart cities, telecommunication networks, finance & insurance, sustainable energy consumption, health care, e-marketing, or bioinformatics. The manuscript provides the main related concepts and updated references that illustrate the applications of these hybrid optimisation-simulation-learning approaches in solving rich and real-life challenges under dynamic and uncertainty scenarios. A numerical analysis is also included to illustrate the benefits that these approaches can offer across different application fields. Finally, this work concludes by highlighting open research lines into the combination of these methodologies to extend the concept of simulation-based optimisation.

Keywords Metaheuristics · simheuristics · learnheuristics · biased-randomised heuristics · stochastic optimisation · dynamic optimisation

Angel A. Juan, Javier Panadero & Diego Oliva
Universitat Oberta de Catalunya & Euncet Business School
Tel.: +34-934-817272
E-mail: {ajuanp, jpanaderom, dolivana}@uoc.edu

Paula Carroll, Peter Keenan & Seán McGarraghy
University College Dublin
E-mail: {peter.keenan, sean.mcgarraghy, paula.carroll}@ucd.ie

Rafael Martí
University of Valencia
E-mail: rafael.marti@uv.es

1 Introduction

Combinatorial optimisation approaches can be used to model many real-world activities in sectors such as transport & logistics, manufacturing & production, finance & banking, bioinformatics, health care, energy systems, and telecommunication networks. Typically, real-life optimisation problems contain rich and soft constraints, as well as non-smooth objective functions, which makes many of them *NP-hard* optimisation problems. In addition, they frequently are of large size, including hundreds or thousands of variables (e.g., customers, products, assets, facilities, etc.). In this scenario, exact optimisation methods are of limited effectiveness, and so metaheuristics provide an excellent alternative to generate near-optimal solutions in reasonable computing times. As explained in Fischetti and Fischetti (2018), it is possible — and frequently convenient — to hybridise metaheuristics with exact methods (matheuristics), since it can help reach solutions of higher quality without incurring prohibitive computational times. However, many real-life systems are challenging not only because of their size but because they also subject to uncertain and dynamic conditions. Dynamic data, for instance: travel times, customers' demands, processing times, investment returns, etc. are better modelled as random variables than as constant values. In addition, real world systems are subject to random events during their operation e.g., random failures of some components, unavailability of some customers or items, changes in market conditions, stockouts due to random demands, etc. Simulation-based optimisation approaches are required (Gosavi et al., 2015; de Sousa Junior et al., 2019) to account for these stochastic uncertainties. Simulation-based optimisation approaches combine optimisation algorithms with simulation methods of any kind, e.g.: Monte Carlo simulation, discrete-event simulation, agent-based simulation, etc. Some simulation-based optimisation approaches rely on the use of heuristics or metaheuristics for their optimisation component. In particular, both biased-randomised heuristics Grasas et al. (2017) and simheuristics Chica et al. (2020) combine simulation with heuristics and/or metaheuristics. The former employs random sampling from a skewed probability distribution to induce a non-symmetric (biased) randomisation effect during the constructive process defined by the logic behind a heuristic procedure. This allows us to transform a deterministic procedure into a probabilistic algorithm, capable of generating alternative solutions of 'good' quality in extremely short computing times (Juan et al., 2009), which can even be real time (less than a second) if parallel computing techniques are utilised. Simheuristics integrate simulation into the metaheuristic framework, so that both components interact and exchange information while searching for an optimal solution in an environment with stochastic uncertainty. Still, other extensions of metaheuristics are possible and even necessary. For instance, some real-life optimisation problems also show dynamic behaviour — e.g., travel times might vary according to the traffic status, customers' demands may depend upon managerial decisions, processing times can vary as servers become less efficient due to intensive use, etc. —, which encourages the combination of metaheuristics

with machine learning methods (learnheuristics) as a way to deal with non-static inputs (Calvet et al., 2017). The main contribution of this paper is to provide an updated overview of all of these extensions of the heuristic concept, which include: metaheuristic, matheuristics, simheuristics, biased-randomised heuristics, and learnheuristics. In doing so, the paper proposes the general terminology *x-heuristics* to cover the aforementioned extensions as well as other possible combinations among them. Thus, for instance, combinations of biased-randomised heuristics with metaheuristics (Ferone et al., 2019), simheuristics (Gonzalez-Martin et al., 2018), and learnheuristics (Bayliss et al., 2020) are already present in the scientific literature. Likewise, it seems quite obvious that some real-life problems might require the hybridisation of simheuristics with learnheuristics to deal with stochastic and dynamic conditions simultaneously. For a similar reason, the combination of metaheuristics, simulation, and fuzzy techniques seems to be a good choice to deal with optimisation problems that consider both stochastic as well as fuzzy uncertainty (Oliva et al., 2020). To the best of our knowledge, this is the first time that such a complete overview on x-heuristics is provided in the scientific literature and therefore we are optimistic that it can be extremely useful for both researchers and practitioners working in the areas of metaheuristic optimisation, simulation-based optimisation, and machine learning. Figure 1 presents the time evolution of these extensions during the last decade, measured by the number of Scopus-indexed articles that mention them in the title, abstract, or keywords.

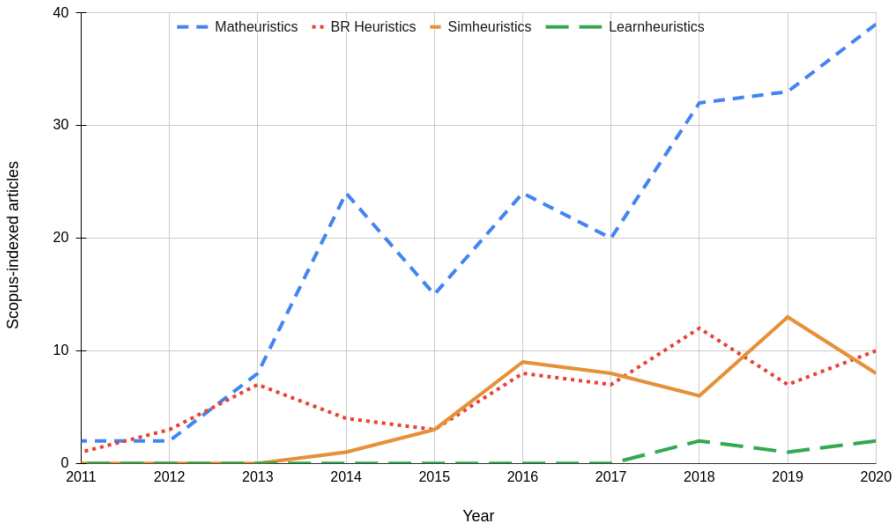

Fig. 1 Evolution of Scopus-indexed articles for each methodology.

Notice that matheuristics are the most popular approach, and the one evolving faster. This reflects the differing strengths of exact methods and metaheuristics and the value of a combination between them. It is also due, at least

in part, to the fact that the combination of exact optimisation methods with metaheuristic optimisation algorithms is a natural process, which is carried out by members of the same academic communities in Operations Research / Management Science (OR / MS) or Computer Science. Other approaches, such as biased-randomised heuristics and simheuristics, are also becoming quite popular, despite the fact that they combine different tools (simulation and optimisation). Finally, learnheuristics is a quite recent concept, but the hybridisation of metaheuristics with machine learning is clearly a very promising area to explore in the future decade.

When searching in Scopus for "(simulation-based) AND optimisation", the first source of documents is the 'Proceedings of the Winter Simulation Conference' (`www.wintersim.org`), with up to 164 indexed documents. If we restrict the search to journals in the period 2006-2020 then four distinct clusters of journals appear. Figure 2 shows the main journals clustered by their citation relations. On the top left are engineering journals such as Structural and Multidisciplinary Optimization. On the bottom left is a small cluster of transport related journals, include Transportation Research Part C: Emerging Technologies and Transport Research Record. On the lower half of the figure is a cluster of OR/MS journals, within this the European Journal of Operational Research and the Annals of Operations Research sit between the transport journals and the production journals, reflecting citation links to both. On the right of the figure is a cluster of energy related journals which have limited citation links with the other fields. This diverse range of applications of simulation in optimisation reflects the value of the techniques discussed in this review.

The rest of this document is distributed as follows: Section 2 provides a short review on metaheuristics, especially in the context of optimisation problems with stochastic components. Section 3 perform a similar analysis in the case of matheuristics. Next, Section 4 discusses the concepts of biased-randomised algorithms and how they can be applied in 'agile' optimisation. Section 5 reviews statistical and machine learning applications in the context of stochastic optimisation. Section 6 offers an overview of the relatively novel concepts of learnheuristics and simheuristics, while commenting on how they are being used to solve stochastic optimisation problems. Section 7 illustrates some numerical examples using several x-heuristics, while Section 8 discusses open challenges and research lines in the field. Finally, Section 9 summarises the main conclusions of the manuscript.

2 Metaheuristics in Stochastic Optimisation

Some well-known decision modelling methods often encounter a great deal of difficulty when faced with the challenge of solving hard optimisation problems that abound in the real world. Vitally important applications in business, engineering, economics, and science cannot be tackled with any reasonable hope of success, within practical time horizons, using exact solution methods that have been the predominant focus of academic research throughout the past three

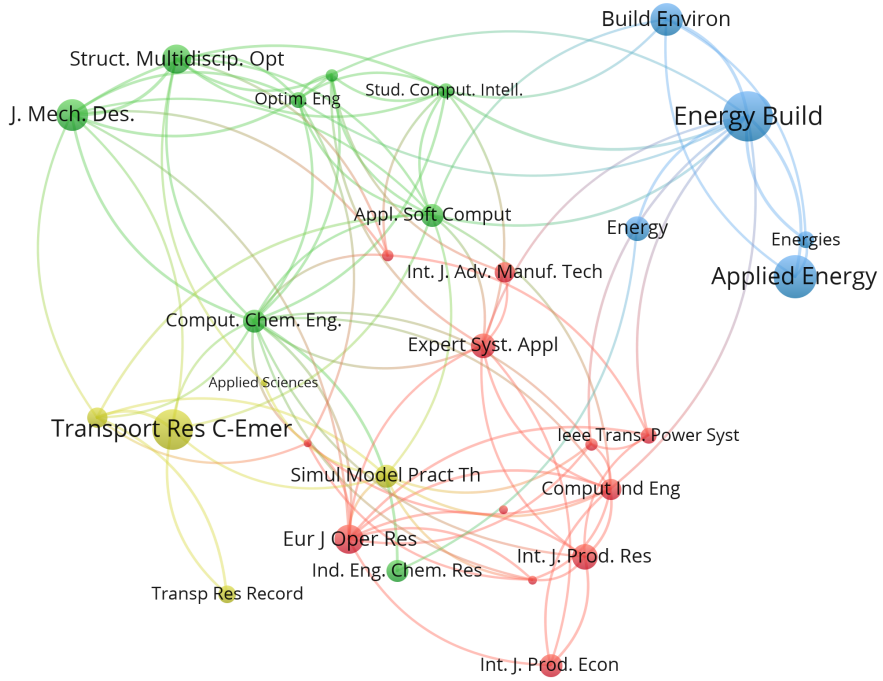

Fig. 2 Main journals in Scopus publishing articles on 'simulation-based optimisation' from 2006-2020 clustered in Vosviewer.

decades. In this context, heuristics have become a very popular family of solution methods for optimisation problems because they are capable of finding acceptable solutions in a reasonable amount of time. In recent decades, algorithmic advances as well as hardware and software improvements have provided an excellent environment on which to build heuristic-based decision support systems based on new and effective methodologies called *metaheuristics*.

Metaheuristic approaches are dramatically changing our ability to solve problems of practical significance and are extending the frontier of problems that can be handled effectively, yielding solutions whose quality often significantly surpasses that obtained by methods previously applied. In this paper, we target a class of optimisation problems that are especially difficult to solve, in which uncertainty adds an extra layer of complexity to problems that are very hard to solve even in their standard deterministic versions, thus resulting in true challenges for optimisation solvers. It is well documented in the scientific literature that it is difficult to obtain good solutions to many families of combinatorial optimisation problems, such as routing, scheduling or production in their deterministic versions. It is therefore expected that when we include uncertainty in the model definition that the resulting stochastic model is especially difficult to solve.

At its core, a metaheuristic algorithm is a search methodology that attempts to find the best (optimal) — or at least a "good" — feasible solution in the solution space defined by the problem representation and the set of constraints in the problem. One metaheuristic methodology differs from another in the way in which it goes about searching the solution space. What they have in common is that they provide the user with a set of rules to guide the design of a heuristic for a specific problem. In this sense, we say that metaheuristics are problem independent, and that it is possible to customise them to solve a particular problem and end up with a heuristic. Recently, Martí et al. (2018), in their three-volume Handbook of Heuristics, compiled 8 search strategies, 4 simple local search based methods, and 14 complex metaheuristics methodologies. Clearly it is not possible to cover all of them in this review, but this study offers an overview of some of them that can be considered representative of a class of methods.

In this paper, we consider a classification of the most commonly applied methodologies in three groups: local search based methods, adaptive memory programming, and evolutionary computation approaches. Local search based methods, as their name indicates, include the methodologies in which one of the main elements is a local search that explores the solution space by successively altering solutions locally. Adaptive memory programming comprises methods that are based on the use of memory, which means that they use past information to make strategic choices in the search process. Finally, evolutionary algorithms, which are possibly now the most popular class of metaheuristics, are adaptive search procedures based on principles derived from the dynamics of natural population genetics. We now highlight three well-known methodologies in each category, although we want to make it clear that this list, far from being exhaustive, only contains some illustrative cases. However, in our opinion, they are the most representative ones.

- **Local Search based**: GRASP, Variable Neighbourhood Search, Iterated Local Search.
- **Adaptive Memory Programming**: Tabu Search, Scatter Search, Ant Colony Optimisation.
- **Evolutionary Methods**: Genetic Algorithms, Evolutionary Strategies, Memetic Algorithms.

In this section, we consider one methodology in each group according to the classification above, to show their implementation in the case of stochastic optimisation. In particular, we consider GRASP, Tabu Search, and Genetic Algorithms. We recommend that the reader further consult the excellent review on metaheuristics for stochastic optimisation by Bianchi et al. (2009) in which four metaheuristic methodologies are considered: Ant colony optimisation, evolutionary computation, simulated annealing, and tabu search. The authors perform an exhaustive review of the previous approaches of each method — see also Brabazon et al. (2015). We limit ourselves to briefly describing the methods chosen, and illustrate them on a classic combinatorial problem, the vehicle routing problem.

2.1 GRASP

The GRASP methodology, *greedy randomised adaptive search procedures*, was developed by Feo and Resende (1995). It was first used to solve computationally difficult set covering problems. Each GRASP iteration consists of constructing a trial solution and then applying a local search procedure to find a local optimum (i.e., the final solution for that iteration). The construction phase is *iterative*, *greedy* and *adaptive*. It is iterative because the initial solution is built considering one element at a time. It is greedy because the addition of each element is guided by a greedy function. It is adaptive because the element chosen at any iteration in a construction is a function of those chosen previously and thus relevant information is updated from one construction step to the next. The local search complements the construction phase by improving the constructed solution performing consecutive moves.

Local search methods, such as GRASP, have been applied to many different stochastic optimisation problems. We consider the vehicle routing problem (VRP) to illustrate how to adapt these methodologies to the stochastic case. In the VRP with stochastic demands a fleet of limited capacity vehicles, located at the depot, has to deliver a product to a set of customers with uncertain demands, which are only known upon the vehicle's arrival. The standard way to approach this problem is called two-stage programming: it first builds a set of routes and then evaluates them, which includes testing their feasibility. If there is a capacity constraint violation (failure), a corrective action (recourse) is taken to recover feasibility, which typically entails adding the extra cost of travelling back to the depot, therefore changing the total duration of the route. In the first stage, a deterministic version of the problem is considered by replacing the random demands by their expected values. In the second stage, many scenarios are simulated (generating values of the random demands) to evaluate the solution in terms of the stochastic data. The approach minimises the cost of the routes including the expected recourse actions.

Mendoza et al. (2016) apply GRASP to the VRP with stochastic demands. The method uses a set of randomised route-first, cluster-second heuristics to generate starting solutions, and a variable neighbourhood descent procedure for the local search phase. For the routing phase, this GRASP implementation applies randomised versions of standard TSP constructive methods. The clustering phase is performed with an adaptation of the classic splitting procedure in routing problems. An interesting implementation detail is the efficient route evaluation using a profile tree that is maintained incrementally.

Ferone et al. (2019) also apply GRASP to the VRP with stochastic demands. The authors implement an advanced design in which the selection of the elements in the construction phase is guided by a probability function instead of the traditional uniform distribution in the restricted candidate list. Their implementation handles the stochastic environment in a two-stage process. In the first stage, the local optima obtained in all GRASP iterations are evaluated by means of short term simulations (to keep the computational effort moderate). Then, only the best local optima according to this fast evaluation

(elite solutions) are submitted to a complete evaluation, which discloses the best solution returned by the method.

2.2 Tabu Search

The term tabu search (TS) was coined in the same paper that introduced the term metaheuristic (Glover, 1986). It is based on artificial intelligence principles, such as memory structures and learning mechanisms. To introduce the methodology, we compare a simple version of TS with a standard descent method for minimising an objective function. Such a method only permits moves to neighbouring solutions that improve (reduce or increase) the current objective function value, and the process ends when no further improvement is possible. The final solution is a local optimum value, since it is at least as good or better than all solutions in its neighbourhood. TS may start in the same way as a descent method, but instead of stopping at the local optimum, the search continues. To perform this process, it has to select moves that cause the objective function value to deteriorate, and therefore it incorporates a mechanism to prevent cycling when alternating between improving and non-improving moves. This is implemented in an efficient way by means of a dynamic neighbourhood definition.

Gendreau et al. (1996), address stochastic demands and customers in a classic paper on TS for vehicle routing. The authors pointed out that in the stochastic VRP, the shape of optimal solutions is quite often counter-intuitive from a geometric standpoint, thus making it particularly hard to solve. One of the major contributions of that paper is the development of an easily-computed proxy for the objective function, to be used in the evaluation of potential moves, and also the elaboration of a series of mechanisms aimed at efficiently managing that proxy. Regarding the exploration, the neighbourhood definition is based on insertions, and the tabu tenure is randomly set in an interval depending on the problem size. The quality of the method can be assessed using empirical results on instances with a known optimum for deterministic values.

Li and Li (2020) propose a tabu search for a stochastic VRP with time windows. Starting from a greedy solution, the method implements three standard neighbourhoods in routing problems: 2-opt, swap, and reallocate. Vehicle travel time, service time, and arrival time are normally-distributed random variables; therefore, the objective function is comprised of three parts: vehicle travel cost, penalty cost for earlier arrival, and penalty cost for later arrival. Their computational experience shows that the greater the variance, the stronger the randomness of vehicle travel time and service time, and the worse the optimal solution and average solution obtained by the algorithm.

2.3 Genetic Algorithms

Holland (1975) introduced genetic algorithms (GAs) and the notion of imitating nature and the "survival of the fittest" paradigm inspired by Darwin's theory. In GAs, a population of candidate solutions, called chromosomes, evolves over successive generations using three genetic operators: selection, crossover, and mutation. First of all, based on some criteria, every chromosome is assigned a fitness value, and then the selection mechanism is invoked to choose relatively fit chromosomes to be part of the reproduction process. Then, new chromosomes are created through the crossover and mutation operators. The crossover generates new individuals by recombining the characteristics of existing ones, whereas the mutation operator is used to maintain population diversity with the goal of avoiding premature convergence. The mutation operation diversifies the search process to explore the candidate solution on the solution space in a random way. Advanced GA designs include the application of local search methods to improve solutions, speeding up the convergence process. GAs coupled with local search are sometimes called Memetic Algorithms.

In the context of the stochastic VRP, Mak and Guo (2004) proposed a variant of the standard genetic algorithm, called Age-GA, in which individuals are not replaced in each generation by their respective offspring. Instead, they continuously may generate new offspring. The survival period of potential individuals therefore becomes longer. Then more useful information and properties can be inherited by their offspring. The number of chromosomes generated by and surviving in each age-group is determined by two sets of parameters, birth rate and survival rate respectively. In this way, the population contains individuals of different ages. The mutation process, as well as the crossover process, may change the number of subroutes in a solution and the customer sequence in a service route.

In terms of the stochastic model considered, Mak and Guo (2004) follow the standard scheme from Gendreau et al. (1996), in which the objective is to design a first stage solution so as to minimise the expected cost of the second stage solution. Eighteen randomly generated problems comprise the benchmark instances used to evaluate the effectiveness of the proposed algorithm. A canonical genetic algorithm is also applied to solve the same group of numerical problems, and it exhibits worse performance than the Age-GA proposed here. The results show that optimal or near-optimal solutions can be obtained by using Age-GA with a reasonable computational time.

This section cannot be complete without briefly describing a very interesting approach presented by Bianchi et al. (2004) on different metaheuristics for the Stochastic VRP, including a GA. As the authors state, they focus on an important aspect of designing metaheuristics for the VRPSD (and for stochastic combinatorial optimisation problems in general): the objective function is computationally demanding, and effective approximations of it should be employed. In particular, they test the impact on the metaheuristic's performance (GAs, ant colony optimisation, or TS) of using the tour length of the a priori tour as a fast approximation of the objective function.

An important contribution of Bianchi et al. (2004) is the speedup due to the use of a fast approximation of the objective in the local search, when many potential moves must be evaluated before one is chosen. The use of this *ad-hoc* approximation permits a straightforward application of metaheuristics originally designed for deterministic problems. As a matter of fact, the literature on metaheuristics for stochastic optimisation can be divided into those approaches using ad-hoc approximations — such as this one —, and those applying Monte-Carlo simulation — such as the ones reviewed above. We may consider these improvements in the implementation of metaheuristics as a first step in the design of hybrid methods, as the simheuristics and learnheuristics covered in the following sections.

3 Exact Methods and Matheuristics in Stochastic Optimisation

As demonstrated in Section 2, heuristics generally offer good solutions in low computational times, but it is difficult to quantify the quality of the solution. Mathematical Programming exact approaches guarantee finding global optimal solutions. Mixed Integer Linear Programming (MILP) solvers have demonstrated massive gains in computational speed becoming 1,250,000 times faster over the period 1990 to 2016. This has been accompanied by significant improvements in hardware making MILP solvers more than 2 trillion times faster than the 1990's (Bertsimas et al., 2016). However, a continuing drawback of using exact methods to solve *NP-complete* problems is that the computational needs increase exponentially with the problem size. Therefore, only small instances of the most challenging real world problems can be solved in a reasonable time.

Some of the limitations of exact approaches such as scalability and determinism point to gaps that can be addressed by augmenting and adapting existing techniques. Matheuristics aim to combine the advantages of exact approaches and heuristics: a combination of heuristics and metaheuristics with exact methods in a matheuristic framework can improve both solution quality and run times (Fischetti and Fischetti, 2018).

3.1 Overview of Exact Mathematical Programming Approach

A brief overview of exact mathematical programming approaches for VRPs appears in Carroll and Keenan (2019). Linear programming (LP) maximises (or minimises) a linear function over real-valued decision variables subject to

a set of linear constraints. The typical mathematical formulation is:

$$\max \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

subject to

$$\sum_{i=1}^{n} a_{ji} x_i \leq b_j \qquad \forall \quad j = 1, \ldots m \tag{2}$$

$$x_i \in \mathbb{R}^+ \qquad \forall \quad i = 1, \ldots n \tag{3}$$

where x_i are the decision variables, c_i are the objective function cost coefficients, b_j are the resource limits or constraint bounds, and a_{ji} are the elements of an $m \times n$ matrix representing the usage of resource j by decision activity i.

An optimal solution is found at a vertex of the LP solution space. Since the feasible solution space is bounded by the linear constraints of the problem, the solution space is convex and the optimal solution is guaranteed to be globally optimal.

VRPs, team orienteering problems (TOPs), and arc routing problems (ARPs) are discrete optimisation problems which fall into the class of *NP-complete* problems. Decisions have to be made on the order in which to visit customers or locations. VRP-like problems are naturally modelled by graphs with nodes representing (warehouse) depots, customers — locations to be visited —, and edges or arcs representing roads or links in transport networks. Graph models are suitable supports for Integer Programming formulations, which opens the opportunity to apply well developed MILP techniques. For example, a route can be defined as a sequence of road network segments. The road network can be modelled as a graph $G = (V, E)$, where each edge ij can then be associated with an integer decision variable, $x_{i,j}$, indicating the number of times the edge should be traversed in the recommended solution.

Unfortunately, the addition of the integrality constraints means that the optimal Integer Programming (IP) solution may no longer be located at a vertex of the LP relaxation (i.e., the LP problem with real valued decision variables). Hence we have to resort to algorithms that methodically search for feasible integer solutions, such as branch-and-bound. This search approach does not scale well, and Integer Programming is itself an *NP-Hard* problem.

Advanced techniques, such as branch-and-cut, are used to improve performance (Naddef and Rinaldi, 2002). This approach identifies additional valid inequalities (called cuts) which are added during the branch-and-bound search to cut off parts of the LP search space that does not contain valid integer solutions. An alternative approach is to consider a subset of the decisions, and only add the decision variables (columns) to the matrix if they add value to the objective function. This type of search is called branch-and-price or column generation.

The branch-and-cut approach adds rows to the matrix. In contrast, the branch-and-price approach adds columns. Finding the best cuts to add to the problem is called the *separation problem*. Some types of cuts can be separated

exactly in polynomial time, but the cut separation problem for some types of cuts is *NP-Hard*. Such cuts are often separated heuristically. Deciding which cuts to separate, and how and when to separate them, is part of the branch-and-cut algorithm design space.

3.2 Overview of Matheuristcs

Heuristics can be used throughout the IP search, starting from simple approaches such as variable fixing or rounding an LP relaxation, i.e., omitting the integer constraints in the model. This can be useful in finding an initial feasible integer solution and so reducing the integrality gap, the gap between the LP relaxation and the best integer solution.

Puchinger and Raidl (2005) discuss the integration of exact and heuristic methods. They suggest that combinations can be classified into loosely coupled collaborative combinations, or more tightly integrated integrative combinations. The simplest form of collaboration is sequential execution, for instance where a heuristic approach is used to establish the value of an upper bound. Parallel or intertwined execution might take advantage of modern multiprocessor computers and allow a heuristic approach to work in parallel with an exact method, again perhaps calculating an upper bound. However, an integrative combination of methods is needed to achieve substantial benefit. This has led to a new class of algorithms known as a matheuristics, which closely combine heuristic and exact approaches. Boschetti et al. (2009) describe matheuristics as "heuristic algorithms made by the interoperation of metaheuristics and mathematical programming (MP) techniques". Matheuristic approaches may exploit exact techniques within metaheuristic frameworks or may use heuristics to improve the performance of exact approaches.

Archetti and Speranza (2014) propose an alternative taxonomy of matheuristics for routing problems with three main approaches: decomposition, improvement heuristics, and branch-and-price/column generation-based approaches. Using a decomposition approach, the problem is decomposed into sub-problems, some or all of which are solved to near optimality using exact approaches. Improvement heuristics apply exact approaches to enhance a sub-optimal solution that has been created using a heuristic. Finally, the Integrated Approach incorporates appropriate heuristics into the branch-and-cut or branch-and-price MILP frameworks.

Fischetti and Fischetti (2018) focus on this last category, and describe matheuristics not as a rigid paradigm but as a concept framework. They describe *local branching* as being in the spirit of local search metaheuristics, where the k-opt neighbourhood of a reference binary solution vector x is searched to find a better nearby solution by flipping (complementing) a small number of the binaries. In what they call a *relaxation induced neighbourhood search*, decision variables that have integer values at specified nodes in the branch-and-bound search are fixed at those integer values. This effectively reduces the number of columns of the matrix, making the remaining MILP easier

to solve. They next describe a *polishing algorithm*, which invokes a genetic algorithm evolutionary heuristic at selected nodes of the branch-and-bound tree. Members of a fixed size population of feasible solutions are combined, and a similar integer fixing approach for good solutions is used to reduce the search space: if the decision variable value in the parents is the same, the decision variable is fixed to that integer value. Lastly, they describe *proximity search*, which addresses the issue of fixing the local branching search to a k neighbourhood. An iterative search of varying values of k is conducted using a tolerance limit, which specifies a minimum improvement required as a stopping criterion.

3.3 Applications of Matheuristics

Matheuristics have been applied in many areas of application, but notably in vehicle routing problems and transport scheduling problems, as there is real difficulty in solving many real world problems in these domains. Complex routing problems are often compound in nature. This is the case, for example, of location-routing problems or inventory-routing problems. The partitioning or clustering of customers would seem to make the problems suitable for divide-and-conquer approaches, so that optimal routes only need to be found for the customer clusters. Unfortunately, this decomposition yields sub-optimal solutions in general. The customer clustering acts to compound the combinatorial nature of VRP-like problems, which makes them some of the most challenging optimisation problems. Their compound nature make them very difficult to solve, but can facilitate their solution by a combination of different methods. Other routing and transportation problems are stochastic in nature, which proves challenging for exact methods, and matheuristics can often make a useful contribution here.

Schermer et al. (2019) and Oliveira et al. (2018) describe matheuristics, which can be classed as Decomposition matheuristics. Schermer et al. (2019) discuss VRP for Drones (VRP-D) and propose a matheuristic that decomposes the problem into an allocation component and a sequencing component. The allocation component is solved using traditional VRP heuristics and drone assignment, while scheduling is subsequently determined by a MILP. For larger problems, they further decompose the assignment and scheduling phase into several smaller problems. The matheuristic was shown to be superior to a purely LP based approach in many cases. This approach approximately fits in the Decomposition class of Archetti and Speranza (2014), with some of the sub-problems solved heuristically and some solved exactly.

Oliveira et al. (2018) discuss the use of a matheuristic approach to solve a challenging stochastic problem in car rental fleet management. They propose a MILP model and decompose the decisions of the original MILP model, with some of the integer decisions made by a genetic algorithm. These partial solutions are fixed and the MILP model is solved for the remaining decisions.

Examples of Improvement Heuristics include those from Archetti et al. (2017), Penna et al. (2019), and Keskin and Çatay (2018). The first authors

use a matheuristic to solve the multi-vehicle inventory routing problem. In this approach, an initial solution is found using a LP relaxation. Then, a tabu search is employed to refine the initial solution, and a further IP formulation is solved using information from the tabu search to fix some variables and focus the search on the most promising part of the solution space. This matheuristic allowed larger problems to be solved to optimality than had been possible in previous work using IP alone. Penna et al. (2019) also use an Improvement Heuristic approach and show that their matheuristic approach is not only effective at finding solutions, but also stable in that good solutions were found in almost all cases. This is an important result, as maximising the worst case performance is sometimes more important than maximising the best case. They combine a multi-start Iterated Local Search with a Set Partitioning IP approach. The heuristic builds a pool of initial routes, which are are then finalised in a restricted IP model. Keskin and Çatay (2018) describe the Electric Vehicle Routing Problem with Time Windows and three different recharging strategies. They couple an Adaptive Large Neighbourhood Search (ALNS) approach with an exact mathematical programming approach to solve large problem instances. They use a two-phase matheuristic approach, which falls in to the Improvement class of Archetti and Speranza (2014). In the first phase, an ALNS is used to find an initial good solution, which is improved in the second MIP phase. Fischetti and Fischetti (2018) also use an Improvement Heuristic to tackle a distance-constrained capacitated vehicle routing problem. As noted, the combinatorial nature of VRPs creates an additional layer of optimisation to find a balanced distribution of the nodes between the routes. An ad-hoc heuristic is used to create and recombine solutions with a MIP deployed to reallocate some solution components.

Fischetti and Fischetti (2018) describe two further applications that can be classed as an Integrated Approach. They address a wind farm layout optimisation problem using the proximity search approach. These authors use a heuristic to fix some of the variables in a multi-start branch-and-bound search, so that they can solve a challenging packing problem that arises in inventory allocation applications. In this problem, the operational cost for packing the bins is relatively high, and even calculating the LP relaxation for this problem turns out to be computationally expensive. They exploit the structure of the MIP model and design a heuristic to iteratively solve a restricted problem — where a subset of variables are fixed — and report good results using the matheuristic. Fischetti and Fischetti (2018) note that designing an effective heuristic is an art that cannot be framed into strict rules. Our brief survey of applications shows the diversity of matheuristic approaches that can be useful to solve challenging combinatorial optimisation problems. The ways in which heuristics are incorporated into the MILP framework impacts the computational speedup and the degree to which the integrality gap can be closed.

3.4 Further Limitations of MILP

A final concern for Integer and Linear Programming is that the parameters $c_i, a_{j,i}$ and b_j from Equations. 1 and 2 must be known with certainty. The focus on the electrification of transport to mitigate climate change and the increased availability of data from Intelligent Transport applications open opportunities to use the new data sources to address advanced vehicle routing and emerging vehicle routing problem variants (Vidal et al., 2020). This requires us to adapt deterministic exact techniques and explore stochastic variants to allow us explore the stochastic nature of the real world, rather than relying on simple expected values of the parameters. Techniques such as Chance Constrained Programming, Stochastic Programming, and Robust Optimisation are used to address uncertainty in the data.

4 Biased-Randomized Heuristics & Agile Optimisation

Biased-randomisation techniques allow us to transform a constructive heuristic, which follows a certain (problem-dependant) logic, into a probabilistic algorithm, which introduces minor deviations from the aforementioned logic. This is achieved by simply introducing a "light" randomness into the heuristic procedure, i.e., a randomness that does not modify the original logic in a significant way. As explained in Grasas et al. (2016), there are many ways to generate such an effect, but one of the most efficient ones — in terms both of computational efficiency and results quality — is by employing skewed (non-symmetric) probability distributions to select the next step, from a candidate list sorted according to a logical criterion, during the solution-building process. In other words, the skewed probability distribution assigns higher probabilities of being selected to building steps located at the top of the list, and lower probabilities to those located near the bottom. Then, Monte Carlo simulation (random sampling) is employed to sequentially select these building steps, which introduces some random but relatively small deviations from the original heuristic (the actual size of these deviations is typically defined by a parameter). As a consequence, this biased-randomised algorithm can be executed multiple times, thus generating alternative solutions of similar quality. As depicted in Figure 3, biased-randomised algorithms fall somewhere in between heuristics and metaheuristics in terms of the quality of the solutions they can generate. The relevant aspect is that they offer the possibility of reaching these intermediate solutions with virtually the same wall-clock time as the one employed by the heuristic — as will be discussed later, this is achieved by executing different threads in parallel.

From a computational point of view, one of the most effective ways to introduce biased randomisation into a heuristic is by employing the Geometric probability distribution. This arises because this distribution has only one parameter, $\alpha \in (0, 1)$, and an analytical expression exists to quickly generate random observations from a Geometric distribution — thus eliminating the

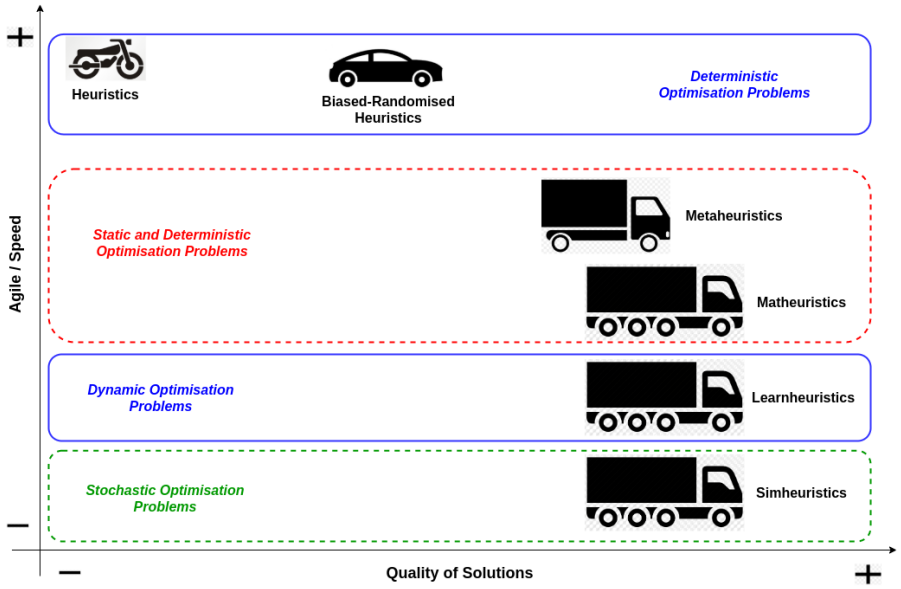

Fig. 3 A classification of x-heuristics based on agility and quality of solutions.

need for time-consuming loops. On the one hand, as the value of α approaches 1, the probabilistic algorithm performs more greedily (i.e., more similarly to the original heuristic). On the other hand, values of α close to 0 emulate the behaviour of a uniform randomisation (i.e., larger deviations from the logic behind the heuristic). It is the use of intermediate values that allows us to explore different regions of the solution space while, at the same time, keeping in mind the logical criterion employed to sort the list of building steps.

Biased-randomisation strategies can be used to extend and enhance the performance of classical metaheuristics (Ferone et al., 2019), and they have been used recently in solving multiple optimisation problems, such as facility location problems (Quintero-Araujo et al., 2017), vehicle routing problems (Fikar et al., 2016; Belloso et al., 2019), or inventory management problems (Quintero-Araujo et al., 2019a). In addition, biased-randomisation strategies can easily be combined with simheuristics to deal with optimisation problems with stochastic components. This form of hybridisation can be found, for instance, in Gruler et al. (2020) for addressing the multi-period inventory routing problem with stochastic demands, or in Guimarans et al. (2018) for solving the two-dimensional vehicle routing problem with stochastic travel times.

One interesting aspect of biased-randomised algorithms is that they can be naturally executed in parallel, i.e., we can use multiple CPU / GPU cores to simultaneously run a large number of threads of the biased-randomised version of the heuristic. Notice that each of these threads will execute in virtually the same time as the original heuristic, which typically means much less than a second on a modern CPU — except maybe for exceptionally large-scale

problems that might require a few seconds of computation. Thus, these parallelised biased-randomised algorithms constitute an efficient tool when performing "agile optimisation", where: *(i)* the best possible solution to a large-scale *NP-hard* optimisation problem is required in real time; and *(ii)* the underlying system or process needs to be re-optimised every few minutes (or even less) as the inputs and constraints are dynamically modified due to the arrival of new data or to changes in environmental conditions. This is the case, for instance, with ride-sharing operations in smart cities (Martins et al., 2021), or when solving two-echelon vehicle routing problems in real time (Martins et al., 2020).

5 Statistical & Machine Learning in Stochastic Optimisation

Statistical learning (SL) and machine learning (ML) can be combined with stochastic optimisation from different points of view. One of them is the algorithmic approach in which the methods help each other for training, or to estimate different internal control parameters. The second point of view is related to the use of SL and ML to handle stochastic problems. Here, the learning tools play an important role, since they permit one to estimate, predict, and generate different variables that directly affect the problem formulation. Figure 4 presents a classification of the approaches that use SL and ML in stochastic optimisation.

This section studies the most recent advances in the two directions and permits an understanding of how the methodologies can be combined.

5.1 Merging Statistical Learning and Machine Learning with Stochastic Optimisation Algorithms

Statistical / machine learning algorithms and stochastic optimisation commonly help each other to solve the drawbacks that each one presents for solving complex problems. The fields of ML and SL are wide, and there exist numerous methods for different tasks like regression, classification, clustering, data exploration, prediction, etc. Such methods have a training phase in which the information is presented to the algorithm and it can be adapted to the data set by the modification of internal parameters. This process is vital for both ML and SL, since the algorithm learns from the sample(s) and a good training is reflected in the final use of the approach. In most learning algorithms, the training process can be seen from an optimisation point of view, and different methodologies of stochastic optimisation can be applied. Some open problems in this direction are presented by Gambella et al. (2020).

An important example is in the supervised training of neural networks, including deep nets, where the parameters to be trained or estimated are hidden layer weights on arcs within the network. A common objective function is the total squared error, a measure of cost, defined as:

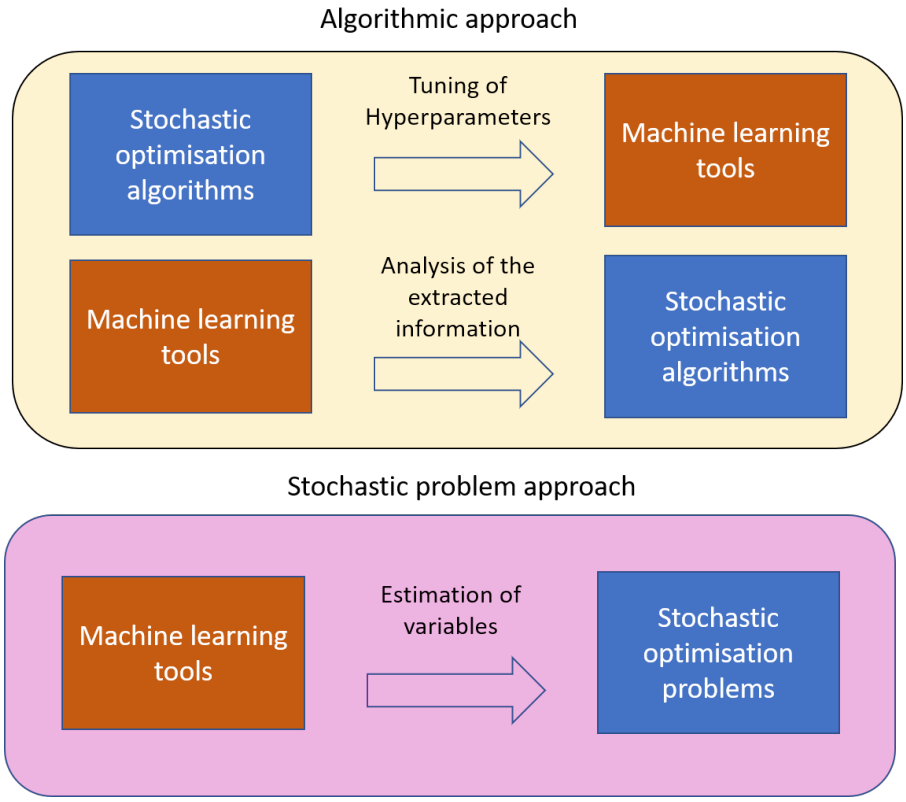

Fig. 4 Different approaches of ML and SL in stochastic optimisation.

$$E = \sum_{k=1}^{N} \sum_{q=1}^{M} (y_q^k - \hat{y}_q^k)^2, \tag{4}$$

where N — the training set size — is the number of input data vectors (and also output vectors), M is the number of output neurons, and y_q^k and $\hat{y}_q^k$ are the target and predicted values of component q of the kth output vector, respectively. Then, the objective is to minimise this "sum-of-squares" (quadratic) error function. If this were a convex objective, a gradient descent approach could be used to find an exact optimum. However, it is in general a multimodal objective surface. Yet, gradient descent methods are still commonly used. For instance, the backpropagation algorithm reduces the total error E by calculating the gradient of the error surface E at its current point (corresponding to the current weight vector for the network) and adjusting the weights in the network to descend the error surface.

In ML, especially when using deep nets, a major issue is the large training sets needed for good generalisation, as these are inevitably more computationally costly. Stochastic gradient descent (SGD, an adaptation of gradient

descent) is used for almost all deep learning training. As above, the error function of a learner is typically written as a sum of per-item error (the loss function) over all training data items, or as a mean of these per-item errors. If there are N training items, then an epoch of backpropagation training with gradient descent requires computing partial derivatives of the loss function for each, at a computational cost of $O(N)$. For a training set of billions of items, each descent step becomes unacceptably long. SGD interprets the gradient as an expectation, which can be estimated using a small set (called a minibatch) of training examples. The minibatch size is typically between one and a few hundred, and is not changed even as the training set size N increases. The minibatch of examples is drawn either systematically or uniformly randomly from the training set. Using SGD, a model can be fitted to a training set of billions of examples, even though using updates computed on only a few hundred examples from the minibatch. Sun et al. (2020) give a useful overview of SGD and related optimisation methods in ML parameter optimisation.

Moreover, ML and SL are able to enhance different aspects of optimisation algorithms (including heuristics). In stochastic optimisation, the methodologies of ML and SL can be used not only to estimate and configure different parameters and variables of the algorithm, but also to find some values of the objective function. The implementation directly depends on the nature of the problem to be solved. In general terms, the optimisation algorithm collects information about the search space by using different operators. Then, using ML / SL, the collected information is analysed. By including ML / SL approaches, it is then possible to improve the search procedure and find more accurate solutions. Corne et al. (2012) present an study of the synergies between data mining algorithms with optimisation algorithms for specific multiobjective problems. This work provides an overview of how ML / SL approaches can be incorporated into optimisation algorithms. In particular, the article explains how data mining can help in the field of OR/MS. Essentially, the authors cover three different ways to hybridise these methods. The first one is to speed up the search process, the second is to improve the quality of the obtained results, and the third is to tune the optimisation algorithm. In the same way, Zhang et al. (2011) present an interesting survey related to evolutionary algorithms and ML. The authors mention different ways of incorporating the two fields. Here, the most important remarks are how ML could help to learn the structure of the problem and also how it is able to predict prominent regions of the search space. It is possible to find several interesting approaches related to the application of hybrid optimisation algorithms. For example, the use of a ML rule called Opposition-Based Learning (OBL) with a stochastic version is considered to improve the differential evolution algorithm in Choi et al. (2019). The idea behind OBL is to evaluate a solution and its opposite at the same time, and then decide which one of them is better. This permits a wide exploration of the solution space with a better chance of finding the optimal solution, while avoiding sub-optimal regions.

Recently, researchers have begun to explicitly combine the fields of SL / ML together with OR / MS. For example, Bertsimas and Kallus (2020) “combine

ideas from ML and OR / MS in developing a framework, along with specific methods, for using data to prescribe optimal decisions in OR / MS problems that leverage auxiliary observations". These authors introduce the concept of a *predictive prescription*: a function $z(x)$ that prescribes a decision in anticipation of the future, given the observation $X = x$. The authors provide constructions for predictive prescriptions, including constructions based on empirical risk minimisation, and show that these are computationally tractable under mild conditions. They further show that asymptotically the prescriptions converge to true full information optimisers, and they extend their work to the case where some decision variables may affect the uncertain variable in unknown ways not captured in the cost function. Analogously to the SL concept of coefficient of determination R^2, they introduce a metric called the *coefficient of prescriptiveness* to measure efficacy of predictive prescriptions. The authors conclude by applying the approach to a real-world inventory management problem. Related work has been carried out on the so-called "Predict-then-Optimise Framework". For example, El Balghiti et al. (2019) derive generalisation bounds on a loss function that considers the cost of the decisions induced by the predicted parameters, rather than the prediction error of the parameters themselves. The fields of SL and stochastic programming are extremely interesting. Their merge allows the use of more robust algorithms for solving complex problems, for example in big data. The Learning Enabled Optimisation (LEO) proposed by Sen and Deng (2018) is part of the suite of hybrid methods in which the SL is combined with stochastic optimisation. The idea is to use any SL to analyse the information of the data set and with the optimisation step it is possible to adopt different paths while the algorithm is running.

5.2 Statistical and Machine Learning for Solving Stochastic Optimisation Problems

SL and ML techniques can help to identify and model different variables of stochastic problems. Since such problems are hard and complex they include values that are constantly updated or that came from different sources. The use of SL / ML methods is then extended to handle the special conditions of stochastic optimisation. We will now analyse the different applications in which ML are used for solving stochastic problems. Donti et al. (2017) discuss learning probabilistic ML models in the context of stochastic programming and examine three real-world applications: an inventory stock problem, an electrical grid scheduling problem, and an energy storage arbitrage task. In energy, Crespo-Vazquez et al. (2018) propose the use of a ML mechanism for multivariate clustering and recurrent neural networks. Such approaches are used to handle the uncertainty in energy price. The goal of this is to find patterns in the daily prices, then extract information out of the sequence in which those patterns appear. In the field of manufacturing, the use of reinforcement learning in stochastic optimisation is proposed by Mosadegh et al.

(2020). The authors incorporate Q-Learning for solving the mixed-model sequencing problem with stochastic processing times (MMSPSP). The goal is to minimise expected total work-overload and idleness (WnI). Since MMSP is an *NP-hard* combinatorial optimisation problem, it is necessary to have powerful algorithms that permit finding the optimal (or near-optimal) solutions. In ML it is possible to find algorithms that work with mathematical and statistical distributions. Gaussian process regression (GPR) is a supervised machine learning method that is able to approximate functions with prominent local features. A combination of GPR with the exploitation of active subspaces (AS) is proposed by Scheidegger and Bilionis (2019) to solve high-dimensional dynamic economic problems. GPR and AS help to estimate the parameters of the value and policy function that are part of the model's problem. The authors report that, by using machine learning techniques, they are able to solve problems of more than 500 dimensions.

Transportation systems are another interesting line of research and several important problems are open to solution with modern methods. Ying et al. (2020) handle the problem of trains scheduling with stochastic passengers demand. Here, the use of deep reinforcement learning is introduced to help minimise the passenger waiting cost and train operating cost over the service runs. The proposal incorporates two artificial neural networks. One is used as a critic, and the second one is the actor. The critic is used to parameterise the state space, while the actor is in charge of the decision space. The critic estimates the states and costs that are used by the actor to generate the schedule decisions. In general terms, the approach is efficient compared to other methods used for comparison.

Statistical learning is another tool that could be applied for solving stochastic problems. SL involves different methodologies such as as logistic regression, Gaussian process regression, quantile regression, Bayesian networks and support vector machines, among others. Regarding such methods, Chen and Wang (2019) propose the inclusion of a Bayesian network into two-stage stochastic programming, and then apply this method for blood bank location-inventory in case of disasters. The authors propose a Bayesian network that uses the interdependence of different uncertain factors to generate practical scenarios and incorporate them in an optimisation model. Regarding electrical power systems, Balata et al. (2019) present the use of SL for stochastic control in microgrid management. The authors employ state-dependent probabilistic constraints represented by an expectation constraint at each system state. SL helps to estimate the admissible set as a function of the system state. In general terms, the authors conclude that using logistic or Gaussian process regression to estimate the admissibility probability outperforms the other options.

6 Learnheuristics and Simheuristics for Dynamic and Uncertain Scenarios

Learnheuristics integrate machine learning techniques with metaheuristic algorithms (Calvet et al., 2017). The main concept here is that some problem inputs (e.g., travel times, customers' demands, processing times, etc.) might depend upon the specific configuration of the solution being built (e.g., assigning a customer to one facility or another might change his / her demand value, choosing a path for a vehicle might modify travel times for other vehicles in the city, selecting an asset to be included in a portfolio might modify the monetary return of other assets due to a substitution effect, etc). In general, even when these dependencies can be identified, they might follow complex patterns that also depend upon many other factors (e.g., the system status at any given time). Hence, these unknown patterns constitute a 'black box' that influences the results of our decisions and so they cannot be ignored by the metaheuristic search. Accordingly, a learning 'white box' mechanism is needed in order to emulate the unknown behaviour and make accurate predictions about the real results of our decisions while building a solution. Learnheuristics have been employed in solving multi-depot vehicle routing problems with market segmentation (Calvet et al., 2016), in vehicle routing problems with dynamic inputs (Arnau et al., 2018), or in team orienteering problems with stochastic rewards (Bayliss et al., 2020).

As already described in Section 1, simheuristics refer to a particular type of simulation-based optimisation that relies on the combination of simulation — of any type — with metaheuristics to solve optimisation problems with stochastic components in their objective function or constraints (Juan et al., 2018). Simheuristic algorithms assume that high-quality solutions to deterministic versions of the optimisation problems are likely to be high-quality solutions to their stochastic counterparts, at least up to a certain degree of variability in the random elements of the problem. In practice, this is usually a reasonable assumption to make, since if the level of variability in the random elements is extraordinarily high, then we might be close to a chaotic scenario in which comparing the quality of two different solutions makes no sense. Observe that the fact that one solution outperforms another under a deterministic scenario does not necessarily imply that the former will be better than the latter under an uncertainty scenario. For instance, a tight vehicle routing plan that performs optimally when travel times and customers' demands are deterministic, might suffer from route failures (and, hence, additional costs) during the execution stage whenever any of the aforementioned elements are modelled as random variables (Gruler et al., 2018). Based on these principles, simheuristic algorithms use the metaheuristic component to generate high-quality solutions for the deterministic version of the problem, and then run a simulation on the most promising deterministic solutions to estimate their real performance in a stochastic scenario. Rabe et al. (2020) provide an example of simheuristic in which a genetic algorithm is combined with a discrete-event simulation. Since simulation — and especially discrete-event simulation — can be time-

demanding, the authors also provide some useful recommendations to speed up the computational times requested to obtain high-quality stochastic solutions. Also, in de Armas et al. (2017) the authors explain how the output provided by the simulation component can be utilised to better guide the search process carried out by the metaheuristic component.

Simheuristic approaches have been compared to other stochastic optimisation methods, such as sample average approximation (Pagès-Bernaus et al., 2019). One of the main advantages of using simheuristics is that the simulation component offers additional random observations on the solution performance. Hence, we are not limited to obtaining just an estimate of the expected cost of the solution but we can also compute many statistics that can be very valuable in performing risk (Gruler et al., 2017) or reliability analysis (Cabrera et al., 2014; Hatami et al., 2018). Recently, simheuristics have been combined with fuzzy techniques in order to address more general optimisation problems in which both stochastic and non-stochastic uncertainty is present (Oliva et al., 2020). This is the case, for instance, for last-mile distribution problems in which some travel times can be modelled as random variables while others show a fuzzy nature. Likewise, combinations of simheuristics with Petri net predictors have been explored to account for possible correlations between random variables (Latorre-Biel et al., 2020). All in all, as stated in Chica et al. (2020) simheuristics constitute a 'first-resort' method when addressing large-scale and *NP-hard* optimisation problems under stochastic scenarios.

7 Cross-Problem Analysis of Computational Results

This section presents a summary of results previously published in different papers. They have been selected to illustrate the effectiveness of the presented approaches. Specifically, this article focuses on the aspects related to the use of such algorithms to solve different well-known *NP-hard* and large-scale real-life optimisation problems. Table 1 presents the obtained average results for a set of optimisation problems with scenarios under uncertainty. The first column identifies the references where the results were obtained, while the second column specifies the exact type of problem to be solved. Notice that seven different problems have been selected in order to cover a wide range of real-life optimisation problems. The next column identifies the type of objective function, i.e., maximisation or minimisation. Subsequently, the next three columns display the obtained results considering the different scenarios. The our-best deterministic (OBD) column shows the best solution found without considering stochasticity. This column refers to the deterministic version of the problem. Notice that this value can be seen as a reference lower bound value in a scenario with perfect information. The next column (OBD-S) shows the expected cost obtained when the best deterministic solution is evaluated in a stochastic scenario, with the corresponding level of uncertainty. A simulation process is applied to the OBS solution to compute the expected cost of this solution, Similarly, the column our-best stochastic (OBS) shows the expected

cost obtained using a simheuristic approach for the stochastic version of the problem.

Figure 5 depicts an overview of the results in Table 1, where the vertical axis represents the gap obtained for the stochastic solutions (OBD-S and OBS) with respect to the deterministic solution (OBD). The results show that the solutions provided by the simheuristic (OBS) clearly outperform the solutions for the deterministic version of the problem when these are simulated (OBD-S) for all the considered problems. On average, an improvement of about 5.8% is observed, in terms of expected cost, for the OBS solutions. Hence, this confirms that near-optimal solutions for the deterministic version of the problem might be sub-optimal solutions for the stochastic version. For instance, the solutions obtained for the stochastic team orienteering problem (STOP) perform optimally when all the elements of the problem are deterministic, but they are sub-optimal when they are applied in scenarios with a degree of variability in the elements of the problem, increasing the expected cost up to 10.8% with respect to the OBS. This is due to route failures, which occur during the execution stage that penalise the entire route. In this case, using the OBS-D solution could lead to inefficient decisions, causing an extra cost to the companies. Hence the importance of integrating simulation methods during the searching process when dealing with stochastic optimisation problems. As a counterpart, when the level of uncertainty does not have too much influence on the elements of the problem, the OBD-S solution could still be competitive, providing results close to the OBS ones. A clear example could be the distributed permutation flow-shop problem (DPFSP), where although the OBS solution outperforms the OBD-S in about 0.45%, the latter still provides competitive results.

Table 1 Summary of results of different optimisation problems with scenarios under uncertainty.

Reference	Problem	Type of Problem	OBD [1]	OBD-S [2]	OBS [3]
Panadero et al. (2018)	SPOP	Max.	3,377.73	3,250.20	3,314.62
Panadero et al. (2020)	STOP	Max.	528.28	359.11	468.80
Guimarans et al. (2018)	2L-VRPST	Min.	1,549.28	1,874.68	1,825.65
Gonzalez-Martin et al. (2018)	ARPSD	Min.	5,412.75	6,223.00	5,669.25
Quintero-Araujo et al. (2019b)	CLRPSD	Min.	98,587.08	111,545.99	111,246.35
Hatami et al. (2018)	DPFSP	Min.	4,822.96	4,964.81	4,944.81
Reyes-Rubiano et al. (2019)	EVRPST	Min.	16,490.13	19,995.73	19,339.89

Table 2 reports the obtained results for a set of problems which have been solved using different optimisation methods. As in the previous table, the first three columns show the following: the reference where the results were obtained, the type of problem to solve, and the type of objective function, respectively. Subsequently, the next four columns correspond to previously published results, which include: *(i)* the best-known solution (BKS) for the problem; *(ii)*

the provided results using a constructive heuristic; *(iii)* the obtained results when the heuristic is turned out into a biased-randomised heuristic (BRA); and *(iv)* the results when a metaheuristic is used to solve the problem. Figure 6 illustrates a summary of the presented results for the different problems, with the vertical axis representing the gap obtained for the different optimisation methods with respect to the BKS. The results show that constructive heuristics give the highest gaps for all the considered problems, because they are simple methods that are intended to be flexible and they are used for quick decisions. On average, heuristic methods give a gap of about 4% with respect to the BKS, varying from about 2.1% for the permutation flow-shop problem (PFSP) up to 7.4% for the arc routing problem (ARP). Notice that the benchmarks commonly used to solve the ARP contain large-scale instances, so the heuristics are not sufficiently powerful methods to explore all of the search space. When these constructive heuristics are turned into a probabilistic ones, applying biased-randomisation techniques, the gap with respect to the BKS is reduced, improving the solutions provided by the constructive heuristic for all the problems. Thus, on average, the obtained gap is about 1.8%, with a highest gap of about 1.9% for the uncapacitated facility location problem (UFLP). In general, it can be observed that the gap is very similar for all the considered problems, demonstrating the efficiency of this type of algorithm — which might provide solutions below 2% of gap with respect to the BKS. Finally, metaheuristic methods provide the best quality solutions for all the problems — at the cost of requiring much higher computing times —, with an average gap of about 0.7% with respect to the BKS. This is because they are the most powerful methods, comprising a set of mechanisms and operators capable of exploring large search space. Notice that, although for some problems — such as the UFLP or the PFSP — the use of a metaheuristic enhances significantly the results provided by the BRA method, in other cases — such as the permutation flow-shop Problem with delivery dates and cumulative payoffs (PFSPDP) —, the BRA method is capable of providing very high-quality solutions, very near to that provided by the metaheuristic. Thus, the use of one or the other will depend on the level of accuracy needed at each moment and, specially, on the computational time available.

8 Insights, Open Challenges & Research Lines

Heuristic and metaheuristic approaches are important tools widely used for solving complex problems. When they are modified or hybridised, they become more powerful and permit the handling of a wide range of problems. X-heuristics comprises the different variants and families of heuristic optimisation algorithms. This field is still growing and open challenges arise in multiple domains. Moreover, they could generate open research lines. One of the most important challenges is to decide when to use a specific methodology. In this case, it is important to analyse the nature of the problem and to identify the different decision variables. Figure 7 shows a classification of x-heuristics based

Table 2 Summary of results of applying different optimisation approaches to the same problem.

Reference	Problem	Type of Problem	BKS [1]	Heuristic [2]	BRA [2]	Metaheuristic [4]
Villarinho et al. (2021)	PFSPDP	Max.	167.83	163.71	166.25	166.93
Ferone et al. (2019) / Juan et al. (2013)	VRP	Min.	966.31	1,012.44	985.33	977.20
Ferone et al. (2019) / Juan et al. (2014)	PFSP	Min.	7,715.80	7,883.63	7,867.30	7,757.87
Ferone et al. (2019) / de Armas et al. (2017)	UFLP	Min.	1,200,420.00	1,248,726.78	1,234,087.33	1,203,558.81
González-Martín et al. (2012) / de Armas et al. (2018)	ARP	Min.	3,653.64	3,924.34	3,705.57	3,692.39

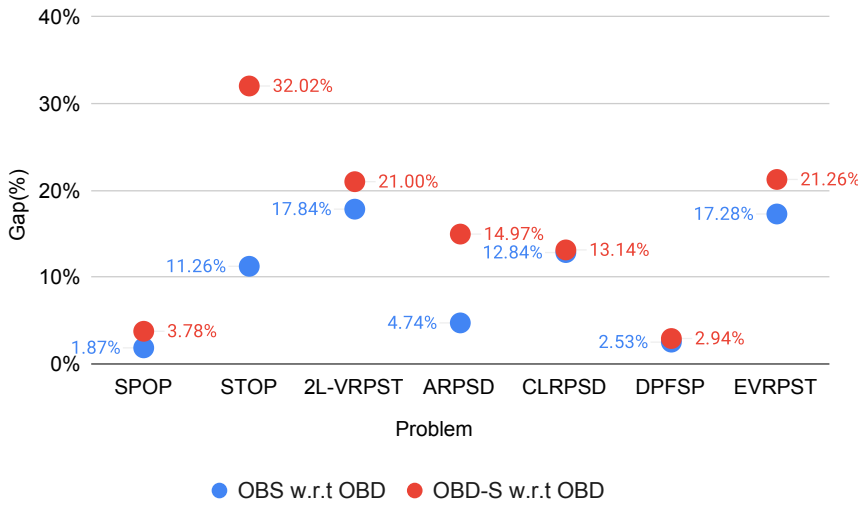

Fig. 5 Gaps of stochastic solutions with respect to deterministic solution (OBD).

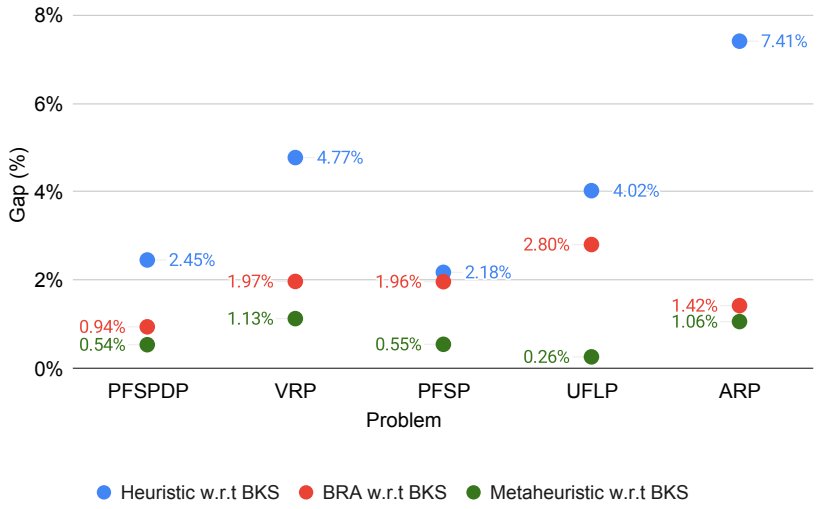

Fig. 6 Gaps of different optimisation methods with respect to the BKS solution

on the type of problem that they can address in a more natural way. In the end, all x-heuristics are extensions of the concepts of heuristics and metaheuristics. Depending on the specific requirements of the problem (e.g., problems with uncertainty, dynamism, real-time requirements, etc.), these extensions include different combinations with exact methods, simulation, machine learning, and even fuzzy sets.

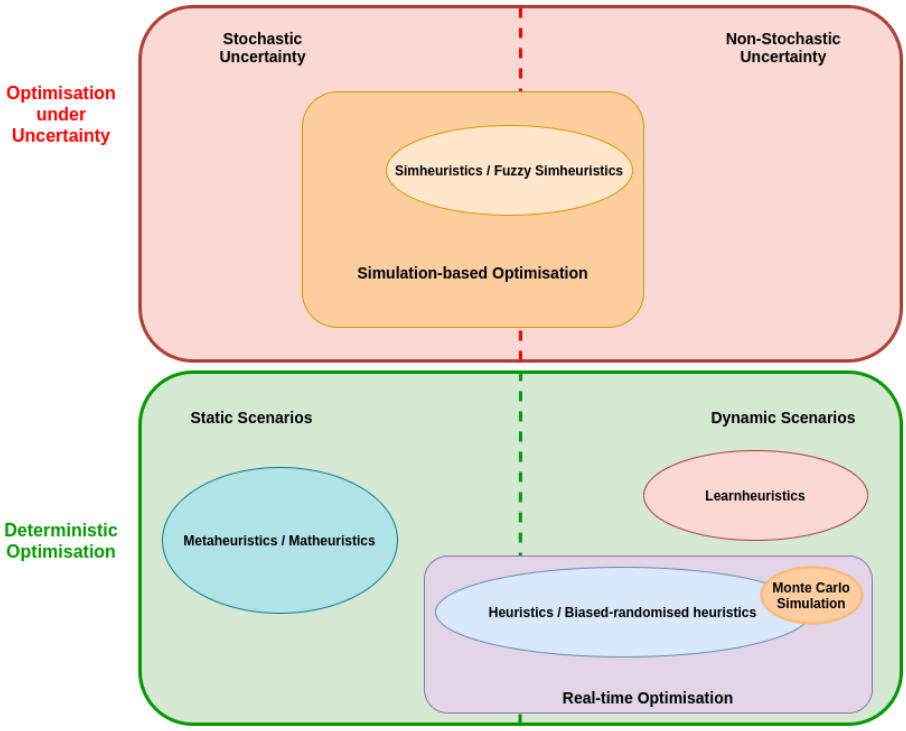

Fig. 7 A classification of x-heuristics based on the type of problem addressed.

Based on Figure 7, it is possible to understand when the "x" in x-heuristic changes its value depending on the problem to be solved. From the previous analysis, one can reach the following conclusions on each methodology and when it can be employed:

- Metaheuristics: The methods in this class are able to find near-optimal solutions in a reasonable computing time. They include the use of different rules that permit applying different operators according to the search process evolution.
- Exact methods and matheuristics: The main feature is that they permit to combine exact approaches with the capabilities of heuristics, in order to speed up the search for near-optimal solutions. While exact methods perform specific tasks — such as enumerating the solutions —, heuristics apply bounds to the search space to carry out the search only in prominent areas.
- Biased-Randomised Heuristics & Agile Optimisation: Biased-randomised techniques allow us to transform constructive heuristics into probabilistic algorithms. This process helps to introduce some typically random external information to the heuristic approach. On the other hand, agile optimisation permits execution of the biased-randomised method on parallel, using hardware as multiple cores or even graphical process units (GPUs). This

allows for real-time decision making even in the case of many large-scale optimisation problems.
- Statistical & Machine Learning: The advantages of this hybridisation give two branches. One is the algorithmic branch, where ML / SL methods help to improve the search process and vice-versa. The second branch is related to employing ML / SL methods in order to estimate different variables and parameters of the optimisation problem. By doing this, it is possible to have more realistic solutions.
- Learnheuristics: These basically combine machine learning techniques with heuristics and metaheuristics. They are based on the fact that problems commonly have variables that depend on the configuration of the solutions. Then the machine learning tools help to predict the states of the search.
- Simheuristics: The merging of simulation methods with heuristics permits handling the stochastic component of the problems by including a simulation stage. Then simheuristics consider that optimal solutions of a deterministic problem are also good solutions of a similar stochastic problem. Some variations of the random elements of the problem can also be included.

An important research line is the exploration of different and novel methodologies that permit the creation of more efficient x-heuristics. For example, in the case of ML / SL methods, the field of fuzzy systems could be extended by using type II fuzzy sets or rough sets (Türkşen, 1999). This allows us not only to enhance the solutions, but also to increase the family of algorithms based on fuzzy systems. In addition, it is also important to explore each of the aforementioned methods separately, since they give rise to many open research lines.

9 Conclusions

This article presents a study of different approaches where heuristics are modified to increase their capabilities. Based on such modifications we define the term x-heuristics, where the variable "x" takes different names such as 'meta', 'sim', 'mat', 'biased-randomised', or 'learn'. The sections introduce a deep analysis of x-heuristics in different stochastic applications, such as transportation, logistic, manufacturing, finance, and energy, among others.

The use of heuristics, independently of the family of algorithms, is growing due to the flexibility of these methods that permit easy adaptation. However, depending on the problem and the requirements of the application, different approaches can be used. Metaheuristics are the most popular group of approaches, that allow exploration of the search space by applying a single heuristic using specific rules at different stages of the search process. Also, the use of exact methods in combination with heuristics permit the creation of robust methods, which provide high-quality solutions. The biased-randomised heuristics are important tools that convert a heuristic in a probabilistic algorithm. When such methods are adapted to run on parallel hardware, they are

called agile optimisation algorithms. Recently, the use of statistical and machine learning has become very important, and its combination with stochastic optimisation permits more powerful search algorithms. However, their use can be extended to estimate stochastic variables in optimisation problems. This introduces the concept of a learnheuristic, where computational learning is used to predict the next steps of the optimisation. Finally, with the merger of heuristics with simulation methods, we define the term simheuristics. In this domain a set of optimal solutions for a deterministic problem is also considered optimal for the stochastic version of the same problem by including random elements to model it.

Considering the above, there are different open research lines as to the proper identification of a methodology for a problem. The presented study tries to clarify this but, since many open problems exist, this task is extremely difficult. However, the information presented can be used as a guide to identify when to use one methodology or another. In addition, we have included a section that permits analysis of the computational results from different references from the state-of-the-art.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science (PID2019-111100RB-C21/ AEI/ 10.13039/501100011033). In addition, we would like to thank the support provided by the Michael Smurfit Graduate Business School at University College Dublin.

References

Archetti, C., Boland, N., and Grazia Speranza, M. (2017). A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29(3):377–387.

Archetti, C. and Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246.

Arnau, Q., Juan, A. A., and Serra, I. (2018). On the use of learnheuristics in vehicle routing optimization problems with dynamic inputs. *Algorithms*, 11(12):208.

Balata, A., Ludkovski, M., Maheshwari, A., and Palczewski, J. (2019). Statistical learning for probability-constrained stochastic optimal control. *arXiv preprint arXiv:1905.00107*.

Bayliss, C., Juan, A. A., Currie, C. S., and Panadero, J. (2020). A learnheuristic approach for the team orienteering problem with aerial drone motion constraints. *Applied Soft Computing*, page 106280.

Belloso, J., Juan, A. A., and Faulin, J. (2019). An iterative biased-randomized heuristic for the fleet size and mix vehicle-routing problem with backhauls. *International Transactions in Operational Research*, 26(1):289–301.

Bertsimas, D. and Kallus, N. (2020). From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044.

Bertsimas, D., King, A., and Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The Annals of statistics*, 44(2):813–852.

Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., and Schiavinotto, T. (2004). Metaheuristics for the vehicle routing problem with stochastic demands. In et al., Y. X., editor, *Proceedings of the 8th international conference PPSN VII*, pages 450–460. Springer.

Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). A survey on metaheuristic for stochastic combinatorial optimization. *Natural Computing*, 8:239 – 287.

Boschetti, M. A., Maniezzo, V., Roffilli, M., and Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control. In Blesa, M.J., B. C., Di Gaspero, L., Roli, A., Sampels, M., and Schaerf, A., editors, *Hybrid Metaheuristics*, volume 5818 of *Lecture Notes in Computer Science*, pages 171–177. Springer.

Brabazon, A., O'Neill, M., and McGarraghy, S. (2015). *Natural Computing Algorithms*. Springer-Verlag, Berlin-Heidelberg-Tokyo-New York, first edition.

Cabrera, G., Juan, A. A., Lázaro, D., Marquès, J. M., and Proskurnia, I. (2014). A simulation-optimization approach to deploy internet services in large-scale systems with user-provided resources. *Simulation*, 90(6):644–659.

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

Calvet, L., Ferrer, A., Gomes, M. I., Juan, A. A., and Masip, D. (2016). Combining statistical learning with metaheuristics for the multi-depot vehicle routing problem with market segmentation. *Computers & Industrial Engineering*, 94:93–104.

Carroll, P. and Keenan, P. (2019). Chapter 10 - decision making using exact optimization methods in sustainable transportation. In Faulin, J., Grasman, S. E., Juan, A. A., and Hirsch, P., editors, *Sustainable Transportation and Smart Logistics*, pages 263 – 283. Elsevier.

Chen, S. and Wang, C. (2019). Incorporating a Bayesian network into two-stage stochastic programming for blood bank location-inventory problem in case of disasters. *Discrete Dynamics in Nature and Society*, 2019.

Chica, M., Juan, A. A., Christopher, B., Oscar, C., and W. David, K. (2020). Why simheuristics? benefits, limitations, and best practices when combining metaheuristics with simulation. *Statistics and Operations Research Transactions*, 44(2):311–334.

Choi, T. J., Togelius, J., and Cheong, Y.-G. (2019). A fast and efficient stochastic opposition-based learning for differential evolution in numerical optimization. *arXiv preprint arXiv:1908.08011*.

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

Crespo-Vazquez, J. L., Carrillo, C., Diaz-Dorado, E., Martinez-Lorenzo, J. A., and Noor-E-Alam, M. (2018). A machine learning based stochastic opti-

mization framework for a wind and storage power plant participating in energy pool market. *Applied Energy*, 232:341–357.

de Armas, J., Ferrer, A., Juan, A. A., and Lalla-Ruiz, E. (2018). Modeling and solving the non-smooth arc routing problem with realistic soft constraints. *Expert systems with applications*, 98:205–220.

de Armas, J., Juan, A. A., Marquès, J. M., and Pedroso, J. P. (2017). Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic. *Journal of the Operational Research Society*, 68(10):1161–1176.

de Sousa Junior, W. T., Montevechi, J. A. B., de Carvalho Miranda, R., and Campos, A. T. (2019). Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. *Computers & Industrial Engineering*, 128:526–540.

Donti, P. L., Amos, B. D., and Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. In *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5490–5500. NIPS.

El Balghiti, O., Elmachtoub, A. N., Grigas, P., and Tewari, A. (2019). Generalization bounds in the predict-then-optimize framework. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 14412–14421. Curran Associates, Inc.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 2:1–27.

Ferone, D., Gruler, A., Festa, P., and Juan, A. A. (2019). Enhancing and extending the classical grasp framework with biased randomisation and simulation. *Journal of the Operational Research Society*, 70(8):1362–1375.

Fikar, C., Juan, A. A., Martinez, E., and Hirsch, P. (2016). A discrete-event driven metaheuristic for dynamic home service routing with synchronised trip sharing. *European Journal of Industrial Engineering*, 10(3):323–340.

Fischetti, M. and Fischetti, M. (2018). Matheuristics. In *Handbook of Heuristics*, pages 121–153. Springer.

Gambella, C., Ghaddar, B., and Naoum-Sawaya, J. (2020). Optimization problems for machine learning: a survey. *European Journal of Operational Research*.

Gendreau, M., Laporte, G., and Seguin, R. (1996). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44 (3):469–477.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–547.

González-Martín, S., Juan, A. A., Riera, D., Castellà, Q., Muñoz, R., and Pérez, A. (2012). Development and assessment of the sharp and randsharp algorithms for the arc routing problem. *AI Communications*, 25(2):173–189.

Gonzalez-Martin, S., Juan, A. A., Riera, D., Elizondo, M. G., and Ramos, J. J. (2018). A simheuristic algorithm for solving the arc routing problem with

stochastic demands. *Journal of Simulation*, 12(1):53–66.

Gosavi, A. et al. (2015). *Simulation-based optimization*. Springer.

Grasas, A., Juan, A. A., Faulin, J., de Armas, J., and Ramalhinho, H. (2017). Biased randomization of heuristics using skewed probability distributions: a survey and some applications. *Computers & Industrial Engineering*, 110:216–228.

Grasas, A., Juan, A. A., and Lourenço, H. R. (2016). SimILS: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, 10(1):69–77.

Gruler, A., Panadero, J., de Armas, J., Moreno, J. A., and Juan, A. A. (2018). Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. *Computers & Industrial Engineering*, 123:278–288.

Gruler, A., Panadero, J., de Armas, J., Moreno, J. A., and Juan, A. A. (2020). A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research*, 27(1):314–335.

Gruler, A., Quintero-Araújo, C. L., Calvet, L., and Juan, A. A. (2017). Waste collection under uncertainty: a simheuristic based on variable neighbourhood search. *European Journal of Industrial Engineering*, 11(2):228–255.

Guimarans, D., Dominguez, O., Panadero, J., and Juan, A. A. (2018). A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times. *Simulation Modelling Practice and Theory*, 89:1–14.

Hatami, S., Calvet, L., Fernández-Viagas, V., Framiñán, J. M., and Juan, A. A. (2018). A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simulation Modelling Practice and Theory*, 86:55–71.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Juan, A. A., Barrios, B. B., Vallada, E., Riera, D., and Jorba, J. (2014). A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 46:101–117.

Juan, A. A., Faulin, J., Jorba, J., Caceres, J., and Marquès, J. (2013). Using parallel & distributed computing for real-time solving of vehicle routing problems with stochastic demands. *Annals of Operations Research*, 207(1):43–65.

Juan, A. A., Faulin, J., Ruiz, R., Barrios, B., Gilibert, M., and Vilajosana, X. (2009). Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem. In *Operations research and cyber-infrastructure*, pages 331–345. Springer.

Juan, A. A., Kelton, W. D., Currie, C. S., and Faulin, J. (2018). Simheuristics applications: dealing with uncertainty in logistics, transportation, and other supply chain areas. In *Proceedings of the Winter Simulation Conference*, pages 3048–3059. IEEE.

Keskin, M. and Çatay, B. (2018). A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172 – 188.

Latorre-Biel, J. I., Ferone, D., Juan, A. A., and Faulin, J. (2020). Combining simheuristics with petri nets for solving the stochastic vehicle routing problem with correlated demands. *Expert Systems with Applications*, page 114240.

Li, G. and Li, J. (2020). An improved tabu search algorithm for the stochastic vehicle routing problem with soft time windows. *IEEE Access*, 8:158115–158124.

Mak, K. and Guo, Z. (2004). A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows. In Jones, M., Patek, S., and Tawney, B., editors, *Proceedings of the 2004 IEEE systems and information symposium*, pages 183–190. IEEE press.

Martí, R., Pardalos, P., and Resende, M. (2018). *Handbook of Heuristics (3 volumes)*. Springer.

Martins, L. d. C., Corlu, de la Torre, R., G, C., Juan, A. A., and Masmoudi, M. A. (2021). Optimizing ride-sharing operations in smart sustainable cities: Challenges and the need for agile algorithms. *Computers & Industrial Engineering*, 153:107080.

Martins, L. d. C., Hirsch, P., and Juan, A. A. (2020). Agile optimization of a two-echelon vehicle routing problem with pickup and delivery. *International Transactions in Operational Research*.

Mendoza, J. E., Rousseau, L.-M., and Villegas, J. G. (2016). A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints. *Journal of Heuristics*, 22:539–566.

Mosadegh, H., Ghomi, S. F., and Süer, G. A. (2020). Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research*, 282(2):530–544.

Naddef, D. and Rinaldi, G. (2002). Branch-and-cut algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 53–84. SIAM.

Oliva, D., Copado, P., Hinojosa, S., Panadero, J., Riera, D., and Juan, A. A. (2020). Fuzzy simheuristics: Solving optimization problems under stochastic and uncertainty scenarios. *Mathematics*, 8(12):2240.

Oliveira, B. B., Carravilla, M. A., and Oliveira, J. F. (2018). Integrating pricing and capacity decisions in car rental: A matheuristic approach. *Operations Research Perspectives*, 5:334 – 356.

Pagès-Bernaus, A., Ramalhinho, H., Juan, A. A., and Calvet, L. (2019). Designing e-commerce supply chains: a stochastic facility–location approach. *International Transactions in Operational Research*, 26(2):507–528.

Panadero, J., Doering, J., Kizys, R., Juan, A. A., and Fito, A. (2018). A variable neighborhood search simheuristic for project portfolio selection under uncertainty. *Journal of Heuristics*, pages 1–23.

Panadero, J., Juan, A. A., Bayliss, C., and Currie, C. (2020). Maximising reward from a team of surveillance drones: a simheuristic approach to

the stochastic team orienteering problem. *European Journal of Industrial Engineering*, 14(4):485–516.

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., and Prins, C. (2019). A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research*, 273(1):5–74.

Puchinger, J. and Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. *Lecture notes in computer science*, 3562:41–53.

Quintero-Araujo, C. L., Caballero-Villalobos, J. P., Juan, A. A., and Montoya-Torres, J. R. (2017). A biased-randomized metaheuristic for the capacitated location routing problem. *International Transactions in Operational Research*, 24(5):1079–1098.

Quintero-Araujo, C. L., Gruler, A., Juan, A. A., and Faulin, J. (2019a). Using horizontal cooperation concepts in integrated routing and facility-location decisions. *International Transactions in Operational Research*, 26(2):551–576.

Quintero-Araujo, C. L., Guimarans, D., and Juan, A. A. (2019b). A simheuristic algorithm for the capacitated location routing problem with stochastic demands. *Journal of Simulation*, pages 1–18.

Rabe, M., Deininger, M., and Juan, A. A. (2020). Speeding up computational times in simheuristics combining genetic algorithms with discrete-event simulation. *Simulation Modelling Practice and Theory*, page 102089.

Reyes-Rubiano, L., Ferone, D., Juan, A. A., and Faulin, J. (2019). A simheuristic for routing electric vehicles with limited driving ranges and stochastic travel times. *Statistics and Operations Research Transactions*, 1(1):3–24.

Scheidegger, S. and Bilionis, I. (2019). Machine learning for high-dimensional dynamic stochastic economies. *Journal of Computational Science*, 33:68–82.

Schermer, D., Moeini, M., and Wendt, O. (2019). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106:166 – 204.

Sen, S. and Deng, Y. (2018). Learning enabled optimization: Towards a fusion of statistical learning and stochastic programming. *INFORMS Journal on Optimization (submitted)*.

Sun, S., Cao, Z., Zhu, H., and Zhao, J. (2020). A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681.

Türkşen, I. B. (1999). Type i and type ii fuzzy system modeling. *Fuzzy Sets and Systems*, 106(1):11–34.

Vidal, T., Laporte, G., and Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286(2):401 – 416.

Villarinho, P. A., Panadero, J., Pessoa, L. S., Juan, A. A., and Oliveira, F. L. C. (2021). A simheuristic algorithm for the stochastic permutation flowshop problem with delivery dates and cumulative payoffs. *International Transactions in Operational Research*, 28(2):716–737.

Ying, C.-s., Chow, A. H., and Chin, K.-S. (2020). An actor-critic deep reinforcement learning approach for metro train scheduling with rolling stock circulation under stochastic demand. *Transportation Research Part B: Methodological*, 140:210–235.

Zhang, J., Zhan, Z.-h., Lin, Y., Chen, N., Gong, Y.-j., Zhong, J.-h., Chung, H. S., Li, Y., and Shi, Y.-h. (2011). Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine*, 6(4):68–75.