Chapter 1
A General Model for Automated Algorithm Design

Rong Qu

Abstract
This chapter presents a recently defined novel combinatorial optimisation problem, namely General Combinatorial Optimisation Problem (GCOP), whose decision variables are a set of elementary algorithmic components. The solutions of GCOP, i.e. compositions of algorithmic components, thus represent different search algorithms. The objective of GCOP is to find the optimal algorithmic compositions for solving the given optimisation problems. Solving the GCOP is thus equivalent to automatically designing the best search algorithms for optimisation problems.

The definition of the GCOP is presented with a new taxonomy which categorises relevant literature on automated algorithm design into three lines of research, namely automated algorithm configuration, selection, and composition. Based on the decision space under consideration, the algorithm design itself is defined and transferred to an optimisation problem. Relevant literature is briefly reviewed, motivating a new line of challenging research directions on the emerging topic of automated algorithm design.

1.1 Introduction

Combinatorial Optimisation Problems (COPs) represent an important topic in operational research [1]. Subject to given constraints, a COP consists of assigning discrete domain values to a finite set of decision variables. The aim is to optimise an objective function which evaluates the solutions, i.e. different assignments of domain values to decision variables. In the literature, the mostly studied COPs include job shop scheduling, knapsack problem, personnel scheduling, timetabling and traveling salesman problem, etc. Well-established benchmark COPs (e.g. the OR Library [2] at http://people.brunel.ac.uk/∼mastjjb/jeb/info.html) have motivated research advances on designing effective search algorithms.

Effectively addressing the extensions of the benchmark COPs with various real-world constraints and features has a direct impact on improving the operations efficiency across business and industry sectors. Due to their complex constraints and non-convex problem structure, most of the COPs are NP-hard [3]. Exhaustive search in mathematical optimisation is often not applicable due to the exponential search spaces of COPs. In the last few decades, evolutionary computation and meta-heuristic search algorithms have been extensively investigated and successfully applied to real-world COP applications.

The increasing demand of real-world applications requires fast developments of search algorithms with less involvement of human expertise. This leads to one of the fast emerging recent advances on automated algorithm design in optimisation research and evolutionary computation. That is, to automatically design search algorithms or solvers which are applicable to solve unseen COPs or problem instances without the extensive knowledge from human experts. This echoes the recent fast developments of AI which automatically models extensive human intelligence to address problems in various domains.

In [4], a new taxonomy of automated algorithm design has categorised the latest research into automated algorithm configuration, automated algorithm selection and automated algorithm composition. Based on this taxonomy, this chapter further defines automated algorithm design as a COP with decision variables of parameters, algorithms and components for automated algorithm configuration, selection and composition, respectively. Solving a classic COP usually concerns a search space of problem solutions. Automated algorithm design can be seen as a COP concerning a search space of algorithms.

In [4], a new model, General Combinatorial Optimisation Problem (GCOP), is defined with the new taxonomy to formally model automated algorithm composition as a COP. Algorithm design decisions have been defined as elementary algorithmic components and modeled as decision variables in the search space, optimisation of which automatically composes and designs new generic search algorithms. The GCOP thus also presents a general model capable of defining a wide range of search algorithms as shown in [4].

In this chapter, a brief overview of the three lines of research in automated algorithm design is presented with the new taxonomy in Section 1.2. With this context the fundamentals of the GCOP model are discussed in Section 1.3. Section 1.4 presents some examples of search algorithm defined with the GCOP model, followed by discussions on research issues and future directions in Section 1.5. Section 1.6 concludes the chapter.

1.2 Automated Algorithm Design

To design evolutionary algorithms or meta-heuristics for solving optimisation problems, different decisions need to be made. These include, at a lower level for the chosen target algorithms, how to fine-tune their parameters (e.g. temperatures in simulated annealing, tabu tenure in tabu search, population size in genetic algorithms), how to design operators (e.g. neighborhood operators in local search algorithms, genetic operators in evolutionary algorithms); and at a higher level, which are the most appropriate algorithms, and how to integrate or hybridise different heuristics for solving the problems / instances in hand.

In the existing scientific literature, the majority of search algorithms have been designed manually. Some of the decisions designing effective algorithms have been made online while solving the problems by adaptively adjusting parameter settings, choosing appropriate operators using some learning mechanisms. Many other decisions have been made offline with domain knowledge, human expertise or based on experimental results on testing instances. Nevertheless, making these different decisions is highly dependant on human expertise, and presents a challenge for researchers of different skills.

Automated algorithm design aims to make these above mentioned decisions with less or no human involvement. In [4], recent research advances in automated algorithm design have been categorised with the following different focuses and objectives:

- Automated algorithm configuration: given some pre-defined target algorithm(s), to automatically configure their parameters on a given set of training problem instances offline so as to solve unseen instances.
- Automated algorithm selection: from a portfolio of chosen algorithms with their associated parameters, to select the most appropriate one(s) based on a set of training instances for solving unseen instances.
- Automated algorithm composition: given heuristics or components of some algorithms, to automatically compose them into new general algorithms while solving the problem instances online.

Although not specifically defined in the literature, these above three lines of research on automated algorithm design concern usually discrete decisions in a *search space* of possible *algorithms*, i.e.

solutions for the problem of algorithm design. Automated algorithm design itself thus can be transferred to an optimisation problem, with different decision variables as follows:

- Automated algorithm configuration: explores a search space of different parameter settings of some target algorithm(s), i.e. decision variables are parameters of certain domain within a template of the target algorithm(s). The target algorithm(s) with the best parameter configurations are applied to solve unseen problem instances offline.
- Automated algorithm selection: explores a search space of a family or portfolio of target algorithms, i.e. decision variables are different algorithms or solvers themselves, in some cases associated with their parameters. These algorithms are usually selected against a set of features identified to categorise typical training problem instances into clusters, thus to solve unseen similar instances offline.
- Automated algorithm composition: explores a search space of different heuristics or components, i.e. decision variables are heuristics or algorithmic components of some chosen target algorithm(s) or general algorithms. The best composition(s) of these heuristics or components are explored to solve the problem instances online.

With these fundamental differences on the decision / search spaces, the above defined taxonomy aims to categorise the existing research on automated algorithm design, where terminologies have been used interchangeably at places. The first two lines of research take a top-down approach, with pre-defined target algorithm(s) or pool of algorithms. The resulting configured or selected algorithms are likely to be new variants of the same target algorithm(s) or of the same family of algorithms. The third line of research on automated algorithm composition takes a bottom-up approach, with the given heuristics or components, to freely compose usually new general algorithms.

The following sections briefly review some selected literature on automated design of methheuristics for COPs with the above taxonomy. As an emerging topic potentially across different disciplines, it is challenging to review exclusively all relevant literature in automated algorithm design. For example, in automated algorithm configuration, tree search solvers [5] have been extensively investigated, resulting into highly strong solvers winning SAT competitions. Heuristic search in constraint programming [6] represents another interesting topic. In genetic programming [7], operations upon problem attributes are considered as decisions designing new heuristics. One stream of research focuses on solving SAT [8], where stochastic local search and tree search in a large number of solvers have been investigated across the three lines of research in automated configuration, selection and composition. This broad literature is not the focus in this chapter, however, some of the techniques reviewed could be adopted or transferred to design exact search techniques addressing other optimisation problems.

1.2.1 Automated Algorithm Configuration

Among the three lines of research on automated algorithm design, automated algorithm configuration has been the mostly studied in the last two decades. The target meta-heuristic search algorithms cover heuristics, local search, evolutionary algorithms and swarm intelligence algorithms.

Compared to evolutionary algorithms, heuristics and local search algorithms attracted relatively less attention in automated algorithm configuration. Algorithms considered include Tabu Search [9], Simulated Annealing [10] and Variable Neighborhood Descent [11]. Early research focused on tuning numerical values [12, 13, 9, 14, 10]. Neighborhood operators have also been automatically configured in recent research [15, 16]. COPs considered include flowshop scheduling [10, 9], VRP [12, 15, 16] and TSP [16].

In evolutionary algorithms, different types of parameters have been automatically configured. Most research concern configuring numerical parameters [17, 18, 19], including mutation/crossover rate, population size, chromosome length and number of iterations / evaluations, etc. Other types include categorical [20], symbolic [21], conditional [22], and mixed of these parameters [23, 21, 22, 24, 25, 26]. Types of algorithms cover genetic algorithms [24], evolutionary algorithms [27, 23, 18, 21], memetic algorithms [19, 26], as well as continuous evolutionary algorithms [25]. The problem domains concern a wide range of applications, from function optimisation [27, 21, 25], NK landscape [24, 26], CSP [20], TSP [23, 28] to multiple COPs [19].

Automated configuration has also addressed swarm intelligence, including (multi-objective) ant colony optimisation mainly for TSP [14, 22, 28] and particle swarm optimisation for scheduling and function optimisation [29, 30]. Within the template of ant colony optimisation, automated configurations have been made upon numerical, categorical and conditional parameters. Compared to other meta-heuristic algorithms, the search space of parameters is usually much larger, requiring efficient automated configuration methods.

Different platforms, frameworks and methods have been developed to automatically configure the large number of parameters in the target algorithms using statistical techniques [12, 27, 17, 9, 18, 19], sequential model-based optimisation [15], and evolutionary computation methods [20, 23, 21, 29]. The mostly adopted platforms include ISAC [31], F-Race [22], ParamILS [5], and their extensions I/F-race [22] and irace [28]. In F-Race, the automated parameter tuning is defined as a machine learning problem [32], for which the racing technique is used to statistically eliminate poor configurations of ant colony optimisation [22] and evolutionary algorithms [25] upon training instances offline. In ParamILS [5], iterative local search is employed to effectively explore the configuration space of search algorithms. Although it is mainly used to configure local search algorithms in tree search and SAT solvers [5], it can also be used to configure any parameterised algorithms including heuristics or meta-heuristics [24]. Based on GGA [33] and stochastic offline programming, ISAC [31], which configures mainly SAT solvers can also be used to configure meta-heuristics.

The majority of automated configuration is conducted offline, usually upon a set of training instances, aiming to solve unseen similar testing instances. This opens a new line of research, with regard to algorithm performance, on the choice of the instance features [30], landscape [34, 26], and the choice of the benchmark instances themselves [25]. It has been shown that some of the existing ant colony optimisation algorithms could be replicated by automated configuration of parameters in the defined framework [35]. The new algorithms automatically generated, even of the same family of the target algorithms, are superior to those manually designed, producing highly promising results [9, 21] especially for SAT [8]. Note that manually designed algorithms often require and highly depend on extensive human expertise in algorithm design.

1.2.2 Automated Algorithm Selection

Compared to automated algorithm configuration, automated algorithm selection has started recently, receiving relatively less research attention. The portfolios of algorithms cover a diverse range of approaches and techniques, from different parametric SAT solvers [8, 36], specific heuristics [37, 38], meta-heuristics [39, 40, 41], evolutionary algorithms [42, 43] and swarm intelligence [44].

One main research topic in automated algorithm selection focuses on the analysis and clustering of training instances according to their features, thus to select the best algorithms or solvers accordingly for unseen test instances of the same cluster [36]. This follows the same idea as that of automated algorithm configurations in Section 1.2.1 for unseen instances of similar features. Techniques analysing and clustering problem instances include mainly machine learning [40, 38] and statistical analysis [39, 44, 37, 45].

Several frameworks have been built in automated algorithm selection. Population-based Algorithm Portfolios (PAP) [42] and Hydra [8] have been developed for optimisation functions [42], Boolean satisfiability problem and traveling salesman problem [36]. During the algorithm selection from the portfolios, configuration of the target algorithms has been considered at the same time, using existing platforms as discussed in Section 1.2.1. In Hydra [8], a large number of parameters are configured using stochastic local search in ParamILS, and configurations are iteratively combined into a portfolio, from which the best solvers based on training instances are selected for solving new

SAT instances. In [46], algorithm configuration using GGA [33] integrated with algorithm selection using SATzilla [47] obtain highly promising results solving SAT instances.

1.2.3 Automated Algorithm Composition

In automated algorithm composition, a set of components or heuristics are automatically combined online to produce new generic algorithms. A main line of research is on hyper-heuristics [48, 49], which aim to decide "at a higher abstraction level which low-level heuristics to apply" [50]. Constructive or perturbative low-level heuristics are selected or generated to solve optimisation problems. By searching the given low-level heuristics, high-level methods can thus solve multiple COPs with the same or adaptive algorithms online. With the taxonomy defined above, this can be seen as to automatically design new algorithms by composing the low-level heuristics.

Beside hyper-heuristics, research in automated algorithm composition is under-developed. Some research in the literature concerns components (also called building blocks) for a type of target algorithms, e.g. evolutionary algorithms [51, 23]. Compositions of these building blocks thus can be seen as designing new variants of evolutionary algorithms in a more flexible way. In [4], elementary components of general search algorithms rather than building blocks in evolutionary algorithms have been defined within a new GCOP model. As a result new generic algorithms rather than specific type of algorithms can be automatically designed. Details of the GCOP model are discussed in Section 1.3 in this chapter.

Frameworks have been developed in hyper-heuristics, including HyFlex [52] and EvoHyp [53], supporting automatic composition of low-level heuristics across multiple COPs [49] including those in Cross-Domain Heuristic Search Challenge [52].

1.3 The General Combinatorial Optimisation Problem

In [4], a new problem model, namely General Combinatorial Optimisation Problem (GCOP), is formally defined to model algorithm design as a COP. In the new GCOP model, the most *basic elementary algorithmic components* associated with different heuristics and parameters are defined as decision variables. Different techniques applied to explore the search space of these decisions in algorithm design can thus be seen as to automatically designing new generic algorithms for solving the problems under consideration online. The resulting optimised compositions of basic algorithmic components thus represent new generic search algorithms automatically designed.

Definition of the GCOP [4]

The **General Combinatorial Optimisation Problem** (GCOP) is a combinatorial optimisation problem, where decision variables take domain values from a finite set A of algorithmic components $a \in A$. The solution space of GCOP, C, consists of algorithmic compositions c upon the decision variables a. The objective function of GCOP, $F(c) \rightarrow R$, $c \in C$, measures the performance of c for solving p, the optimisation problem(s) under consideration.

In problem p, the decision variables take values from a finite set of problem-specific values. The solution space S of p consists of the direct solutions s obtained by corresponding algorithmic compositions c, i.e. $c \rightarrow s$. The objective function $f(s) \rightarrow R$ evaluates $s \in S$ for p. The problem p could be extended to other optimisation problems including continuous optimisation, multi-objective optimisation problems, etc.

Objective Functions of the GCOP

Let M be a mapping function M: $f(s) \rightarrow F(c)$, i.e. each algorithmic composition c for GCOP maps to solutions s for p, i.e. $c \rightarrow s$, thus the generation of s reflects the performance of c. The objective of GCOP is to search for the optimal $c^* \in C$ which produces the optimal $s^* \in S$ for p, so that $F(c^*)$ is optimised, as defined in *Objective* (1.1).

$$F(c^* | c^* \rightarrow s^*) \leftarrow f(s^*) = \mathbf{min}(f(s)) \tag{1.1}$$

Objective (1.1) can be extended to measure various aspects of c, e.g. the performance of c for solving multiple p, computational time of c and short-short gain of c, etc. to provide informed search for the GCOP.

Domains of Decision Variables in the GCOP

In [4], a domain $A_{1.0}$ has been established with a set of most basic and elementary algorithmic components a of two categories, namely operators $A_{1.0\_o}$ and acceptance criteria $A_{1.0\_a}$, each with their associated heuristic and parameters. This provides a general algorithm design model for automatically designing a large number of existing meta-heuristics in the literature, i.e. local search algorithms and selection hyper-heuristics. The GCOP can be extended in different ways to sustain advanced research in automated algorithm design. For experienced users, the domain A can be extended with user defined components for different problems under consideration as shown in [4]. Research is ongoing to extend the elementary components to define and automatically design other main types of meta-heuristics, i.e. population-based algorithms.

The basic idea of the GCOP is to breakdown search algorithms into a set of elementary components, which are modularised into general operators associated with different heuristics, as the decision variables. In the existing literature, these elementary components are quite often "hard-wired" manually into integrated or compound operators or heuristics using human expertise. The idea with the GCOP is that the most basic elementary components can thus be freely optimised to automatically generate new generic algorithms which significantly expand the scope that human experts may not be able to explore. In [4] it has been demonstrated that many of the selection hyper-heuristics can be defined using the unified GCOP model. Section 1.4 presents more examples of existing search algorithms in the literature defined using $A_{1.0}$ in the GCOP model.

Search Spaces in the GCOP

The search space C for GCOP consists of all possible compositions $c \in C$. It is fundamentally different from that of S for p, which consists of all possible problem solutions $s \in S$. The encodings of c and s are different, upon which different techniques or methods could be applied to analyse and explore the search spaces of different characteristics. The search space of c is usually of a much lower dimension compared to s which depends on the specific p, and has a different upper bound. Furthermore, the objective function of the GCOP, i.e. performance of c, can be more than just the direct evaluation of s for p.

The concept of two search spaces has been firstly analysed in [54] in the context of hyper-heuristics [49], where high level methods explore the search space of general or problem-specific low-level heuristics, which are then applied to solve the problems online. This concept is generalised in the GCOP to model algorithm design. The search space on decisions of algorithm design is explored at a higher level, leading to the best compositions of algorithmic components addressing the problems in hand. In other words, the GCOP can be seen as modelling the problem of algorithm design as a COP at a higher level.

The underlying theory of the GCOP model is fundamentally different from that of hyper-heuristics, where the low-level heuristics are designed and selected manually by human experts. In another word, many of the low-level heuristics can be seen as specific or compound heuristics integrating subsets of the elementary components as modelled in the GCOP. The idea of hyper-heuristics which search upon these specific low-level heuristics to solve COPs is fundamentally different from that of GCOP, which is upon elementary algorithmic components to support evolving fundamentally new generic algorithms modelled in a search space.

Along with the advances in evolutionary computation, different frameworks, platforms and toolkits have been established, supporting fast development of algorithms with certain consistency. However, there is a lack of certain standard in the optimisation research community [55]. The GCOP provides a new standard which formulates various search algorithms in a unified model with the most basic elementary algorithmic components. Such standard can thus support systematic investigations within coherent frameworks to gain insights in automated algorithm design. To sustain such research advances, the latest developments and resources have been made available at a dedicated GCOP web site at https://sites.google.com/view/general-cop.

1.4 Search Algorithms Defined with the GCOP Model

In [4], with the basic elementary operators $A_{1.0}\_o$ and general acceptance criteria $A_{1.0}\_a$ in $A_{1.0}$, various selection hyper-heuristics for solving vehicle routing problems (VRP) and nurse rostering problems (NRP) have been defined with the GCOP model. The unified GCOP model supports insightful analysis across different search algorithms for the two COPs. For example, it was observed that a subset of the basic $a \in A_{1.0}$ is enough to design a large number of selection hyper-heuristics for solving VRP variants of different constraints. Among them the mostly used operators include *swap*, *interchange* or *k-opt*. Some of the operators used can be seen as compound operators, combining more than one $a \in A_{1.0}$.

With the GCOP model, analysis on $a \in A_{1.0}$ adopted in selection hyper-heuristics for NRP and VRP leads to interesting findings [4]. On the one hand, various acceptance criteria $a \in A_{1.0}\_a$ have been studied for NRP, which is not the case for VRP. This may be due to the highly constrained nature of NRP, requiring complicated constraint handling techniques. On the other hand, most of the NRP algorithms can be formulated using operators $a \in A_{1.0}\_o$, while for VRP problem-specific operators often needed to extend $A_{1.0}\_o$. This may be due to the fixed solution structure in NRP, where shifts can be easily exchanged on the same days; while the variable solution structure in VRP, i.e. different lengths of the routes for each vehicle, calls for effective problem-specific operators addressing different problem features in the wide range of variants.

In this chapter, Table 1.1 presents more selected examples of various local search algorithms defined using subsets of $A_{1.0}$ in the GCOP. In the literature, a large amount of advanced algorithms have been developed with different mechanisms, enhancing the basic variants as shown in Table 1.1. For example, learning has been recently integrated in evolutionary computation and meta-heuristics, greatly improving algorithm performance. Of course, domain knowledge usually significantly improves the efficiency of search algorithms, however, usually at the cost of computational expenses and or human expertise, thus reduces the generality of the algorithms in real-world applications.

Table 1.1 Examples of Local Search Algorithms Defined in GCOP

Algorithm c	$o \in A_{1.0}$ and $a \in A_{1.0}$ used to define algorithm c
Greedy local search (GLS)	Different operators with $a_{oi}(n)$: only the best out of n neighbors is accepted. n can be set as a fixed or variable value.
GLS variant 1	With $a_{oi}(n)$, $o_{xchg}(k,m,h1_w)$, where values of k and m decision variables selected by heuristic $h1_w$ in two parts of a solution s are swapped.
GLS variant 2	With $a_{oi}(n)$, $o_{chg}(k,h1_w,h1_b)$, k decision variables selected by $h1_w$ are changed using heuristic $h1_b$.
Tabu Search (TS)	Different operators with $a_{tabu}(n,l)$: the best out of n neighbors and not in the tabu list is accepted,. l (tabu length) and n are set as a fixed or variable value.
TS variant 1	With $a_{tabu}(n,l)$, $o_{xchg}(k,m,h1_w)$ or $o_{chg}(k,h1_w,h1_b)$ as defined above.
TS variant 2	With $a_{tabu}(n,l)$, $o_{ins}(k,h1_w,h1_b)$, where k decision variables selected by $h1_w$ are inserted to another position in solution s selection by $h1_b$.
Simulated Annealing (SA)	Different operators with $a_{gd}(n,t,r)$: worse solutions sampled from n neighbors are accepted by a probability subject to a temperature t decreased at rate r. Better solutions are always accepted.
SA variant 1	With $a_{gd}(n,t,r)$, $o_{xchg}(k,m,h1_w)$ or $o_{chg}(k,h1_w,h1_b)$ as defined above.
SA variant 2	With $a_{gd}(n,t,r)$, $o_{rr}(k,h1_w,h1_b)$, where values of a section of k decision variables in solution s selected by $h1_w$ are removed, and then reassigned using $h1_b$.
Variable Neighborhood Search (VNS)	$a_{oi}(n)$ with more than one operator as above

In the literature, it is not always clear how some decisions in algorithm design have been made. For example, the acceptance criteria $A_{1.0\_a}$ or the number of n neighbors explored in local search algorithms is not always clearly specified. This led to some ambiguity in reproducing the published algorithms. With the consistent modular components with parametric and heuristic settings in the GCOP standard, these details can be clearly defined with $a \in A$, sustaining consistent research findings reported in the literature.

1.5 Challenges in Automated Algorithm Design with the GCOP

The search space of the GCOP as a COP increases exponentially with A. Identifying a single best $a \in A_{1.0}$, i.e. using a single algorithmic component with the best parametric settings, to design algorithms as shown in Table 1.1 might be manageable using advanced computational techniques. However, the exponential search space of algorithmic compositions freely composed upon different $a \in A$ is a challenging research issue as that for COPs in optimisation research. In other words, those manually designed algorithms in Table 1.1, i.e. component repeatedly called in a search algorithm, represent only a small subset of C, i.e. combinations of different components called in a search algorithm. This presents a new challenge to evolutionary computation and computational intelligence.

With the advances in automated algorithm design, new algorithms generated are likely to be highly different from those manually designed. In other words, many of the existing algorithms designed by human experts are subsets of the search space of algorithm design. This newly identified algorithms provide an interesting subject of research and new challenges across different disciplines including artificial intelligence, evolutionary computation, and machine learning.

The GCOP model provides a new standard to formulate various search algorithms, and also serves a consistent data structure to collect, store and process data of algorithm design for potential knowledge discovery using machine learning. Statistical methods have been used to identify compact subset of low-level heuristics in hyper-heuristics towards building general effective methods [56]. Further systematic analysis using machine learning with standards like the GCOP may reveal new knowledge such as behaviours and speed of algorithmic components and synergy between the basic components, etc. The new standard bridging search algorithms in optimisation and machine learning in artificial intelligence potentially contributes to a significantly improved understanding on effective search algorithms.

Heuristic search algorithms have been criticised with lacking theoretical fundamentals, replying on experience to solve problems where exhaustive search is not realistic. Most established heuristic search algorithms and frameworks in the literature are defined descriptively, lacking a fundamentally consistent structure and standard. As a result, although numerous algorithms have been manually designed in optimisation research, the rich but scattered experience is difficult to model and reuse. The novel GCOP model supports the establishment of a new standard to retain such knowledge into structured optimised general components as the training data for machine learning to sustain data-driven automated design of new algorithms.

Recent research advances in automated algorithm design made some progress on the generality [48, 22] and reusability of algorithms, however, much more needs to be done to significantly reduce human expertise and development barriers. In automated algorithm configuration and selection, existing research has aimed to reuse the configured or selected algorithms upon training instances for solving unseen problems or instances categorised by similar problem features identified [36, 42]. In automated algorithm composition, hyper-heuristics have been developed to address cross-domain COPs [57]. Assessment method [58] has also been proposed to evaluate the generality, rather than the optimality, of search algorithms. The scope of generality and reusability, subject to the No Free Lunch Theorems [59], presents a challenging and interesting theoretical issue to the optimisation research community. Collaborative research across disciplines is needed, especially with recent advances in machine learning, to reveal new knowledge transferable to solve different optimisation algorithms thus to enhance the reusability and generality of search algorithms.

1.6 Conclusions

With the advanced research in evolutionary computation and optimisation research, a large number of meta-heuristics and evolutionary algorithms have been designed in scientific literature for solving benchmark and real-world combinatorial optimisation problems (COPs). The extensive knowledge on algorithm design led to the emerging new challenges on automated algorithm design.

This chapter discusses a recently proposed new taxonomy on automated algorithm design, which categorises research into automated algorithm configuration, selection and composition, concerning a search space of algorithm parameters, a portfolio of algorithms, and algorithmic components or heuristics, respectively. A recently defined new model, the General Combinatorial Optimisation Problem (GCOP), is discussed, where the design of search algorithms itself is transferred and defined as a COP upon elementary components, automated compositions of which evolve new generic algorithms which may be difficult to design manually.

The novel GCOP opens a new line of interesting research directions in optimisation research. In addition to extending the GCOP domain with more algorithmic components to define population-based algorithms, the objective function can be also extended to measure the generality, reusability and computational time of the newly evolved algorithms. The new automatically designed algorithms may introduce new knowledge on algorithm design in the literature for solving other optimisation problems including continuous optimisation problems and multi-objective optimisation problems. Further investigations on the emerging topic of automated algorithm design will stimulate more advances in evolutionary computation and optimisation research.

References

1. C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications Inc., 1982.
2. J. Beasley, "OR-library: Distributing test problems by electronic mail," *Journal of Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, Nov. 1990.
3. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman, 1979.
4. R. Qu, G. Kendall, and N. Pillay, "The general combinatorial optimisation problem - towards automated algorithm design," *IEEE Computational Intelligence Magazine*, To appear, 2020.
5. F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, Oct. 2009.
6. S. Minton, "Automatically configuring constraint satisfaction programs: A case study," *Constraints*, vol. 1-2, no. 1, pp. 7–43, Jan. 1996.
7. J. MacLachlan, Y. Mei, J. Branke, and M. Zhang, "Genetic programming hyper-heuristics with vehicle collaboration for uncertain capacitated arc routing problems," *Evolutionary computation*, Accepted, 2019.
8. L. Xu, H. Hoos, and K. Leyton-Brown, "Hydra: Automatically configuring algorithms for portfolio-based selection," in *Proc. of AAAI Conf. on Artificial Intelligence*, Atlanta, July 11–15, 2010.
9. B. Adenso-Díaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," *Operations Research*, vol. 54, no. 1, p. 99–114, Jan.-Feb. 2006.
10. M.-W. Park and Y.-D. Kim, "A systematic procedure for setting parameters in simulated annealing algorithms," *Computers & Operations Research*, vol. 25, no. 3, pp. 207–217, Mar. 1998.
11. T. Adamo, G. Ghiani, E. Guerriero, and E. Manni, "Automatic instantiation of a variable neighborhood descent from a mixed integer programming model," *Operations Research Perspectives*, vol. 4, pp. 123–135, Sept. 2017.
12. S. Coy, B. Golden, G. Runger, and E. Wasil, "Using experimental design to find effective parameter settings for heuristics," *Journal of Heuristics*, vol. 1, no. 7, pp. 77–97, Jan. 2001.
13. M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation (GECCO'02)*, Jul. 2002, pp. 11–18.
14. P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the f-race algorithm: Sampling design and iterative refinement," in *HM 2007: Hybrid Metaheuristics*, Dortmund, Germany, October 8-9, 2007, pp. 108–122.
15. N. Dang and P. D. Causmaecker, "Characterization of neighborhood behaviours in a multi-neighborhood local search algorithm," in *LION 2016: Learning and Intelligent Optimization.* Lecture Notes in Computer Science 10079), May-Jun., 2016, pp. 234–239.
16. T. Adamo, G. Ghiani, A. Grieco, E. Guerriero, and E. Manni, "MIP neighborhood synthesis through semantic feature extraction and automatic algorithm configuration," *Computers & Operations Research*, vol. 83, pp. 106–119, Jul. 2017.

17. I. Ramos, M. Goldbarg, E. Goldbarg, and A. Neto, “Logistic regression for parameter tuning on an evolutionary algorithm,” in *2005 IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, 2-5 Sept. 2005.
18. M. Preuss and T. Bartz-Beielstein, “Sequential parameter optimization applied to self-adaptation for binary-coded evolutionary algorithms,” in *Parameter Setting in Evolutionary Algorithms*, 2007, pp. 91–120.
19. D. Gümüs, E. Özcan, and J. Atkin, “An analysis of the taguchi method for tuning a memetic algorithm with reduced computational time budget,” in *ISCIS 2016: Computer and Information Sciences*, Poland, Sept, 2016, pp. 12–20.
20. H. Terashima-Marín, P. Ross, and M. Valenzuela-Rendón, “Evolution of constraint satisfaction strategies in examination timetabling,” in *GECCO’99: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, Jul. 1999, p. 635–642.
21. S. Smit and A. Eiben, “Comparing parameter tuning methods for evolutionary algorithms,” in *2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May, 2009.
22. M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, “F-race and iterated F-race: An overview,” in *Experimental Methods for the Analysis of Optimization Algorithms*, Oct. 2010, pp. 311–336.
23. M. Oltean, “Evolving evolutionary algorithms using linear genetic programming,” *Evolutionary Computation*, vol. 13, no. 3, pp. 387–410, Sept. 2005.
24. M.-C. Riff and E. Montero, “A new algorithm for reducing metaheuristic design effort,” in *2013 IEEE Congress on Evolutionary Computation*, Mexico, June 2013.
25. T. Liao, D. Molina, and T. Stützle, “Performance evaluation of automatically tuned continuous optimizers on different benchmark sets,” *Applied Soft Computing*, vol. 27, pp. 490–503, Feb, 2015.
26. A. Liefooghe, B. Derbel, S. Verel, H. Aguirre, and K. Tanaka, “Towards landscape-aware automatic algorithm configuration: preliminary experiments on neutral and rugged landscapes,” in *EvoCOP 2017: Evolutionary Computation in Combinatorial Optimization*, Mar, 2017, pp. 215–232.
27. O. François and C. Lavergne, “Design of evolutionary algorithms - a statistical perspective,” *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 2, pp. 129–148, Apr, 2001.
28. M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, T. Stützle, and M. Birattari, “The irace package: Iterated racing for automatic algorithm configuration,” *Operations Research Perspectives*, vol. 3, pp. 43–58, Sept. 2016.
29. A. Visheratin, M. Melnik, and D. Nasonov, “Automatic workflow scheduling tuning for distributed processing systems,” *Procedia Computer Science*, vol. 101, pp. 388–397, Dec, 2016.
30. T.Agasiev and A.Karpenko, “The program system for automated parameter tuning of optimization algorithms,” *Procedia Computer Science*, vol. 103, pp. 347–354, Jan, 2017.
31. S. Kadioglu, Y. Malitsky, M. Sellmann, K. Tierney, and K. Tierney, “Isac - instance-specific algorithm configuration,” in *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, Lisbon, Portugal, Aug, 2010, pp. 751–756.
32. M. Birattari, *The Problem of Tuning Metaheuristics As Seen From a Machine Learning Perspective.* IOS Press, US, 2005.
33. C. Ansótegui, M. Sellmann, and K. Tierney, “Gga: A gender-based genetic algorithm for the automatic configuration of algorithms,” in *Proceedings of 2009 15th International Conference on Principles and Practice of Constraint Programming*, Lisbon, Portugal, Sept, 2009, pp. 142–157.
34. F. Hutter, H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *LION 2011: Learning and Intelligent Optimization*, Rome, Italy, Jan, 2011, pp. 507–523.
35. M. López-Ibáñez and T. Stützle, “The automatic design of multi-objective ant colony optimization algorithms,” *IEEE Trans. on Evolutionary Computation*, vol. 16, no. 6, pp. 861–875, Feb. 2012.
36. S. Liu, K. Tang, and X. Yao, “Automatic construction of parallel portfolios via explicit instance grouping,” in *Proc. of AAAI Conf. on Artificial Intelligence*, New Orleans, February 2–7, 2018.
37. B. Bischl, O. Mersmann, H. Trautmann, M. Preuss, and M. Preuß, “Algorithm selection based on exploratory landscape analysis and cost-sensitive learning,” in *GECCO ’12: Proceedings of the 14th annual conference on Genetic and evolutionary computation*, Philly, Jul, 2012, pp. 313–320.
38. J. Pihera and N. Musliu, “Application of machine learning to algorithm selection for tsp,” in *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, Limassol, Cyprus, December 2014, pp. 47–54.
39. B. Huberman, R. Lukose, and T. Hogg, “An economics approach to hard computational problems,” *Science*, vol. 275, no. 5296, pp. 51–54, Jan 1997.
40. T. Carchrae and J. Beck, “Applying machine learning to low-knowledge ccontrol of optimisztion algorithms,” *Computational Intelligence*, vol. 4, no. 21, pp. 372–387, Nov, 2005.
41. T. Messelis and P. D. Causmaecker, “An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem,” *European Journal of Operational Research*, vol. 233, no. 3, pp. 511–528, March 2014.
42. K. Tang, F. Peng, G. Chen, and X. Yao, “Population-based algorithm portfolios with automated constituent algorithms selection,” *Information Sciences*, vol. 279, pp. 94–104, Sept. 2014.
43. R. Akay, A. Basturk, A. Kalinli, and X. Yao, “Parallel population-based algorithm portfolios: An empirical study,” *Neurocomputing*, vol. 247, pp. 115–125, Jul. 2017.
44. J. Pérez, R. Pazos, J. Frausto, G. Rodríguez, D. Romero, and L. Cruz, “A statistical approach for algorithm selection,” in *WEA 2004: Experimental and Efficient Algorithms*, Angra dos Reis, Brazil, May, 2004, pp. 417–431.
45. Y. He, S. Yuen, Y. Lou, and X. Zhang, “A sequential algorithm portfolio approach for black box optimization,” *Swarm and Evolutionary Computation*, vol. 44, pp. 559–570, February 2019.
46. Y. Malitsky and M. Sellmann, “Instance-specific algorithm configuration as a method for non-model-based portfolio generation,” pp. 244–259, May-June, 2012.
47. L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown, “Satzilla: Portfolio-based algorithm selection for sat,” *Journal of Artificial Intelligence Research*, vol. 32, pp. 565–606, June, 2008.

48. E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, and E. Özcan, “Hyper-heuristics: A survey of the state of the art,” *Journal of Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013.
49. N. Pillay and R. Qu, *Hyper-heuristics: Theory and Applications.* Springer Nature, 2019.
50. P. Cowling, G. Kendall, and E. Soubeiga, “A hyperheuristic approach to scheduling a sales summit,” in *Proc. of Practice and Theory of Automated Timetabling*, Konstanz, August 16–18, 2000, pp. 176–190.
51. L. Bezerra, M. Lòpez-Ibáñez, and T. Stützle, “Automatic design of evolutionary algorithms for multi-objective combinatorial optimization,” in *Proc. of Parallel Problem Solving from Nature*, Ljubljana, September 13-17, 2014, pp. 508–517.
52. E. Burke, M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. J. Parkes, and S. Petrovic, “The cross-domain heuristic search challenge - an international research competition,” in *Proc. of Intelligent Conf. Learning and Intelligent Optimization*, Rome, January 17-21, 2011, pp. 631–634.
53. N. Pillay and D. Beckedahl, “EvoHyp - a Java toolkit for evolutionary algorithm hyper-heuristics,” in *Proc. of IEEE Congress on Evolutionary Computation*, San Sebastian, June 5-8, 2017, pp. 2707–2713.
54. R. Qu and E. Burke, “Hybridisations withing a graph based hyper-heuristic framework for university timetabling problems,” *Journal of Operational Research Society*, vol. 60, pp. 1273–1285, Sept. 2009.
55. G. Kendall, R. Bai, J. Blazewicz, P. D. Causmaecker, M. Gendreau, R. John, J. Li, B. McCollum, E. Pesch, R. Qu, N. Sabar, G. V. Berghe, and A. Yee, “Good laboratory practice for optimization research,” *Journal of Operational Research Society*, vol. 67, no. 4, pp. 676–689, Apr. 2016.
56. M. Misir, K. Verbeeck, P. D. Causmaecker, and G. Berghe, “An investigation on the generality level of selection hyper-heuristics under different empirical conditions,” *Applied Soft Computing*, vol. 13, no. 7, pp. 3335–3353, Jul. 2013.
57. G. K. R. Q. N.R. Sabar, M. Ayob, “A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems,” *IEEE Trans. on Cybernetics*, vol. 45, no. 2, pp. 217–228, Feb. 2015.
58. N. Pillay and R. Qu, “Assessing hyper-heuristic performance,” *European Journal of Operational Research*, under review, 2019.
59. D. H. Wolpert and W. G. McReady, “No free lunch theorems for optimisation,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April, 1997.