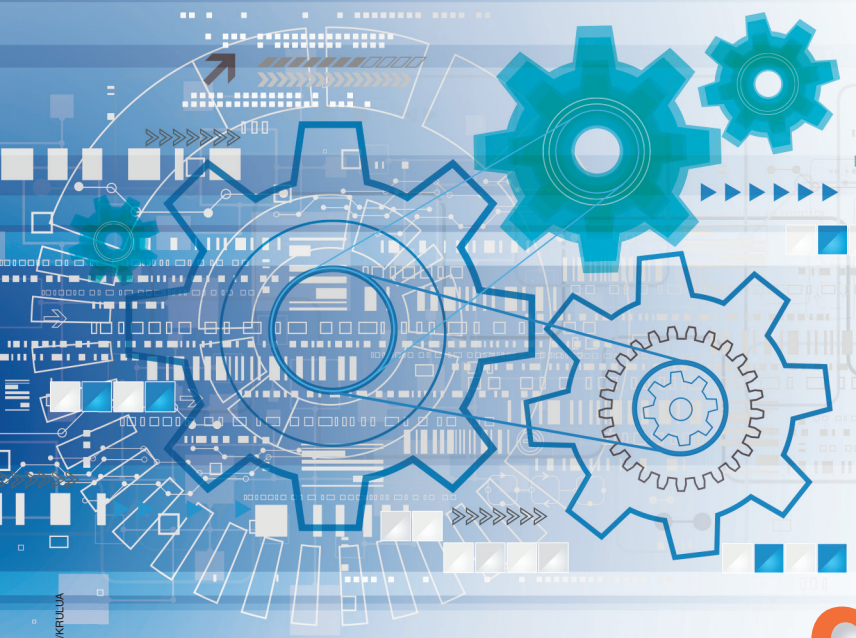

The General Combinatorial Optimization Problem: Towards Automated Algorithm Design

Rong Qu
School of Computer Science, University of Nottingham, Nottingham, UK

Graham Kendall
University of Nottingham, Nottingham, UK; University of Nottingham Malaysia, Selangor, MALAYSIA

Nelishia Pillay
University of Pretoria, SOUTH AFRICA

Abstract—This paper defines a new combinatorial optimization problem, namely General Combinatorial Optimization Problem (GCOP), whose decision variables are a set of parametric algorithmic components, i.e. algorithm design decisions. The solutions of GCOP, i.e. compositions of algorithmic components, thus represent different generic search algorithms. The objective of GCOP is to find the optimal algorithmic compositions for solving the given optimization problems. Solving the GCOP is thus equivalent to automatically designing the best algorithms for optimization problems. Despite recent advances, the evolutionary computation and optimization research communities are yet to embrace formal standards that underpin automated algorithm design. In this position paper, we establish GCOP as a new standard to define different search algorithms within one unified model. We demonstrate the new GCOP model to standardize various search algorithms as well as selection hyperheuristics. A taxonomy is defined to distinguish several widely used terminologies in automated algorithm design, namely automated algorithm composition, configuration and selection. We would like to encourage a new line of exciting research directions addressing several challenging research issues including algorithm generality, algorithm reusability, and automated algorithm design.

Digital Object Identifier 10.1109/MCI.2020.2976182
Date of current version: 10 April 2020

Corresponding Author: Rong Qu (rong.qu@nottingham.ac.uk)

1556-603X/20©2020IEEE

I. Introduction

In this paper, a new model, namely General Combinatorial Optimization Problem (GCOP), is defined to model the problem of algorithm design itself as a COP, solutions of which are new algorithms automatically generated to solve cross-domain COPs.

Along with advances in optimization research, a rich set of Combinatorial Optimization Problems (COPs) has been established. COPs represent a subset of operational research [1]. They consist of, subject to given constraints, assigning discrete domain values to a finite set of decision variables, so as to optimize an objective function which evaluates the solutions. The well-established benchmark COPs (e.g., the OR Library [2] at http://people.brunel.ac.uk/~mastjjb/jeb/info.html) have promoted the design of effective algorithms in evolutionary computation and computational intelligence. Problems addressed include job shop scheduling, knapsack problem, personnel scheduling, timetabling and traveling salesman problem, as well as many others and their extensions with various real-world constraints and features.

In optimization research, computational intelligence and evolutionary computation are two of the recent advances in automated algorithm design. That is, to automatically design search algorithms or solvers which are able to solve COPs or problem instances without extensive domain knowledge from human involvement. In the scientific literature, several terminologies have emerged in different contexts, sometimes being used interchangeably without a clear definition.

In this paper we formally define a taxonomy of automated algorithm design as follows.

- ❏ Automated algorithm configuration: to automatically configure the parameters of pre-defined target algorithm(s) upon a given set of training problem instances.
- ❏ Automated algorithm selection: to automatically select from a portfolio of chosen algorithms with their associated parameters upon a set of training problem instances.
- ❏ Automated algorithm composition: to automatically generate general algorithms by composing heuristics or components of some algorithms to solve problems.

Automated algorithm configuration has been well studied. The most studied algorithms include SAT solvers [3], [4], multiobjective ant colony optimization [5], [6] and stochastic local search [7] for flow shop scheduling problems and traveling salesman problem, and mixed integer programming solvers for traveling salesman problem and vehicle routing problems with time windows [8]. The automated configuration of parameters is usually conducted offline upon a set of training instances. A number of frameworks have been built to search in the configuration space of parameters for the target algorithms, achieving highly promising results which are superior to manually configured algorithms. These include ParamILS [3] which uses iterated local search, F-Race [5] which uses a racing mechanism, and the extended framework irace [9].

In automated algorithm selection, the portfolio of algorithms includes different parametric SAT solvers [10] and evolutionary algorithms [11], [12]. One important research issue concerns the clustering of training instances according to their features, thus to select the best algorithms or solvers for unseen test instances of similar clusters [10]. Frameworks developed include Population-based Algorithm Portfolios (PAP) [11] and Hydra [13] (based on extended ParamILS) for optimization functions [11], Boolean satisfiability problem and traveling salesman problems [10].

In automated algorithm composition, a set of components or heuristics are automatically combined online to generate new generic algorithms. Some research concerns components for a type of target algorithms, e.g., evolutionary algorithms [14], [15]. Another line of research on hyperheuristics [16] decides at a higher level which low-level heuristics to apply [17]. By searching the given low-level heuristics, multiple COPs can be solved online with the same or adaptive heuristic compositions. Frameworks developed include HyFlex [18] and EvoHyp [19], supporting automatic composition of general algorithms across multiple COPs [16].

The fundamental difference between the three lines of research in automated algorithm design is on the decision spaces. Automated algorithm configuration concerns a decision space of parameters within a template of target algorithm(s), rather than freely composing the algorithm components themselves. The resulting algorithms, which are likely variants of the same target algorithms, are best configured for solving the training and unseen testing instances offline. Automated algorithm selection explores a decision space with a family of given target algorithms, which are grouped against some features of training instances for solving similar testing instances offline. Automated algorithm composition explores a decision space of components or heuristics to flexibly compose or combine them. The resulting algorithms generated are new and generic for solving different unseen COPs.

The three lines of research are different ways automating algorithm design. Automated configuration and selection take a top-down approach to configure within given algorithm templates and select from a portfolio of target algorithms, respectively, resulting in variants of the same family of target algorithms. Automated algorithm composition takes a bottom-up approach, working with algorithm components, to flexibly compose and generate new algorithms.

Current automated algorithm design research, however, still requires some human expertise and empirical studies to manually select parameters, target algorithms/solvers, or heuristics/components. Challenges remain to gain the insights on effective algorithms to underpin

With GCOP, the newly generated generic algorithms are likely to be highly different from those manually designed algorithms.

automated algorithm design [20]. Such advances require models and standards to conduct systematic investigations within coherent frameworks.

In this paper, a new model, namely General Combinatorial Optimization Problem (GCOP), is defined to model the problem of algorithm design itself as a COP, solutions of which are new algorithms automatically generated to solve cross-domain COPs. By optimizing the compositions of basic *algorithmic components* as decision variables in GCOP, new generic search algorithms can be automatically generated.

The major aim of GCOP is to establish a standard in algorithm design to model various search algorithms in one framework with the most basic algorithmic components. It is not our intention to model all existing algorithms with GCOP. Other common and user-defined algorithmic components could be added to GCOP to design new algorithms addressing COPs as well as other optimization problems. To our knowledge, there is no existing standard in the literature which formally models the problem of designing search algorithms. Further studies can potentially provide additional insights on how various algorithms work with the new standard, and thus underpin automated algorithm design.

In the rest of the paper, Section II presents the formal definition and search spaces of the new GCOP model. With the most basic algorithmic components, Section III demonstrates the application of GCOP as a standard to define various selection hyperheuristics in the literature for solving two COPs. Section IV discusses research issues and future directions, followed by conclusions in Section V.

II. The General Combinatorial Optimization Problem

A. Definition of the GCOP

Definition 1: The **General Combinatorial Optimization Problem (GCOP)** is a combinatorial optimization problem with decision variables taking 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 given a. The objective function of GCOP, $F(c) \rightarrow R$, $c \in C$, measures the performance of c for solving p, the optimization 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, each obtained by a corresponding algorithmic composition c, i.e., $c \rightarrow s$. The objective function $f(s) \rightarrow R$ evaluates $s \in S$ for p. In this paper we consider COPs as p. This could be extended to other optimization problems as discussed in Section IV.

Let M be a mapping function M: $f(s) \rightarrow F(c)$. 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 optimized, as defined in *Objective* (1). Without loss of generality, we assume p is a minimisation problem in this paper.

$$F(c^*|c^* \rightarrow s^*) \leftarrow f(s^*) = \mathbf{min}(f(s)). \quad (1)$$

The following terminologies are defined in GCOP.

- Problem GCOP: a COP, whose decision variables take discrete values from a finite domain A of algorithmic components a, $a \in A$.
- Domain A for decision variables in GCOP: algorithmic components $a \in A$, including basic operators with heuristics, parameter settings, and acceptance criteria, etc. A can be extended with user-defined components. Examples of the most common basic a in the literature are defined in Table I.
- Solution space C of GCOP: consists of solutions c for GCOP, i.e. algorithmic compositions c upon $a \in A$. Each c is used to produce a solution s for problem p, i.e. $c \rightarrow s$.
- Objective function F for GCOP: $F(c) \rightarrow R$ measures GCOP solutions $c \in C$. The objective is to find the optimal c^* which produces optimal s^* for p, i.e. $c^* \rightarrow s^*$.
- Problem p: the optimization problem(s) under consideration, whose decision variables are problem-specific.
- Solution space S of p: consists of solutions s for p.
- Objective function f for p: $f(s) \rightarrow R$ evaluates solutions s for p, $s \in S$ are obtained by using algorithmic compositions $c \in C$.
- Mapping function M: $F(c) \leftarrow f(s)$: maps each algorithmic composition c for GCOP to a solution s for p, i.e. $c \rightarrow s$, thus the generation of s reflects the performance of c.

In [20], it has been suggested that the optimization research community should adopt a certain standard. The GCOP provides such a standard to define the design of search algorithms with a unified model. In the current research on automated algorithm composition, the search space is upon manually defined heuristics or components in a specific type of algorithm(s). For example, in hyperheuristics [16], the low-level heuristics are often manually determined, and fixed with predefined parameter values. GCOP significantly extends the search space to concern the most basic elementary algorithmic components, thus requires no human involvement. Methods finding the optimal composition of a for GCOP thus automatically design new generic search algorithms.

B. Decision Variables of GCOP

In Table I, we establish *GCOP*1.0 with a domain $A_{1.0}$ of a set of most basic and elementary algorithmic components a. The underlying idea of the GCOP standard is to modularise the existing widely used basic components $a \in A_{1.0}$ which are grouped into two categories, namely operators

$A_{1.0\_o}$ and acceptance criteria $A_{1.0\_a}$, each with their associated heuristic and parametric settings. This categorisation has been widely used in the literature, although quite often manually "hard-wired" into integrated or compound operators or heuristics. Note that acceptance criteria $a \in A_{1.0\_a}$ are general for any problems, and are applicable with any $a \in A_{1.0\_o}$. Different heuristic strategies and parameter settings can be associated with $a \in A_{1.0\_o}$ to define different components.

With the sets of basic generic components $A_{1.0\_o}$ and $A_{1.0\_a}$, GCOP proposes a new standard which defines a large number of different algorithms in one common model. For example, Tabu Search variants can be defined by implementing $o_{\text{xchg}}(k, m, h1^t_w)$ associated with an acceptance criteria $a_{\text{tabu}}(n, l)$, where l (tabu length), n (number of neighbors sampled), and k, m (number of decision variables selected) could be set as fixed or variable values. Associated with $a_{gd}(n, t, r)$ and $a_{oi}(n)$, variants of Great Deluge and greedy search can instead be defined, respectively. Section III demonstrates that with the basic $A_{1.0}$ and some problem-specific components A_p, most selection hyperheuristics for two widely studied COPs in the literature can be defined with the GCOP model.

With GCOP, the newly generated generic algorithms are likely to be highly different from those manually designed algorithms, which are likely subsets of GCOP solutions $c \in C$.

The underlying idea of GCOP is to decompose algorithms into elementary algorithmic components, which can then be composed and optimized in a much more flexible way, and thus design new generic algorithms automatically.

These newly generated c potentially introduce new coherent knowledge in algorithm design with the GCOP model.

The underlying idea of GCOP is to decompose algorithms into elementary algorithmic components, which can then be composed and optimized in a much more flexible way, and thus *design* new generic algorithms automatically. This is different from automated selection from a portfolio of target algorithms and automated configuration of target algorithm(s), where the resulting new algorithms usually belong to the same family or are variants of the target algorithm(s). The decomposition of algorithms into the most basic components allows the most flexibility and provides a much larger scope to design *new* generic algorithms.

The basic a when composed and configured in different ways can define either new or existing search algorithms. *GCOP*1.0 can be seen as a problem instance of the GCOP model, with a small domain of only the basic a. It is not possible, also not intended, to model in GCOP exclusively all algorithmic components in the literature. We aim to establish the

TABLE I Domain $a \in A_{1.0}$ of decision variables for *GCOP*1.0.

Domain $A_{1.0\_o}$	**a WITH HEURISTICS** $h1$, $h2$. $h1^t_w/h1^t_b$ ($h1^r_w/h1^r_b$): *TOURNAMENT* (*ROULETTE WHEEL*) SELECTION OF THE WORST/BEST OF u RANDOMLY CHOSEN DECISION VARIABLES $s_{i,j} \in s, u \in \{0 \dots s\}$; $h2^t_w/h2^t_b$ ($h2^r_w/h2^r_b$): *TOURNAMENT (ROULETTE WHEEL)* SELECTION OF THE WORST/BEST TWO OF v RANDOMLY CHOSEN SOLUTIONS $s \in P, v \in \{0 \dots \lvert P\rvert\}$. P: AN ARCHIVE OF s. $h1/h2$: RANDOM STRATEGIES IF $u, v = 0$.
$o_{\text{asg}}(k, h1_w, h1_b)$	USE $h1^t_b$ OR $h1^r_b$ TO ASSIGN VALUES TO k DECISION VARIABLES SELECTED BY $h1^t_w$ OR $h1^r_w$.
$o_{rm}(k, h1_w)$	REMOVE VALUES OF k DECISION VARIABLES SELECTED BY $h1^t_w$ OR $h1^r_w$.
$o_{\text{chg}}(k, h1_w, h1_b)$	USE $h1^t_b$ OR $h1^r_b$ TO CHANGE THE VALUES OF k DECISION VARIABLES SELECTED BY $h1^t_w$ OR $h1^r_w$.
$o_{\text{xchg}}(k, m, h1_w)$	SWAP k AND m DECISION VARIABLES CHOSEN BY $h1^t_w$ OR $h1^r_w$. $o^{\text{in}}_{\text{xchg}}/o^{bw}_{\text{xchg}}$: DECISION VARIABLES ARE FROM THE SAME/DIFFERENT ROUTES.
$o_{\text{ins}}(k, h1_w, h1_b)$	INSERT K DECISION VARIABLES CHOSEN BY $h1^t_w$ OR $h1^r_w$ TO OTHER POSITIONS SELECTED BY $h1^t_b$ OR $h1^r_b$. $o^{\text{in}}_{\text{ins}}/o^{bw}_{\text{ins}}$: $s_{i,j}$ ARE FROM THE SAME/DIFFERENT i.
$o_{rr}(k, h1_w, h1_b)$	USE $h1^t_b$ OR $h1^r_b$ TO REASSIGN VALUES OF k DECISION VARIABLES SELECTED BY $h1^t_w$ OR $h1^r_w$.
$o_{xo}(k, m, h2_b)$	SWAP k AND m RANDOMLY CHOSEN DECISION VARIABLES BETWEEN TWO SOLUTIONS IN P CHOSEN BY $h2^t_b/h2^r_b$.
Domain $A_{1.0\_a}$	**a WITH PARAMETERS** n, t, r. NEIGHBOR s' OF s IS PRODUCED BY $o \in A_{1.0\_o}$. n: NUMBER NEIGHBORS SAMPLED.
a_{all}	*ACCEPT ALL*, *NAÏVE ACCEPT*: s' IS ALWAYS ACCEPTED, I.E. RANDOM STRATEGY.
$a_{oi}(n)$	*ONLY IMPROVING*: BETTER s' IS ACCEPTED.
$a_{ie}(n)$	*IMPROVE AND EQUAL*: A BETTER OR EQUAL s' IS ACCEPTED.
$a_{\text{late}}(n)$	*LATE ACCEPTANCE*: A s' BETTER THAN THE LAST n VISITED s IS ACCEPTED.
$a_{\text{tabu}}(n, l)$	*TABU*: THE BEST s' NOT IN A TABU LIST OF LENGTH l IS ACCEPTED.
$a_{gd}(n, t, r)$	*GREAT DELUGE*: A WORSE s' IS ACCEPTED BY A PROBABILITY $p = e^{-\lvert f(s') - f(s)\rvert}$. BETTER s' IS ALWAYS ACCEPTED.
$a_{mc}(n, t, r)$	*MONTE CARLO*: A WORSE s' IS ACCEPTED BY A THRESHOLD t, t IS DECREASED BY r. BETTER s' IS ALWAYS ACCEPTED.
$a_{sa}(n, t, r)$	*SIMULATED ANNEALING*: A WORSE s' IS ACCEPTED BY $p = e^{-\lvert f(s') - f(s)\rvert/t}$, t IS DECREASED BY r. BETTER s' IS ALWAYS ACCEPTED.

GCOP standard step by step to explore new effective algorithms that are automatically designed.

Other GCOP instances can be built with extended A consisting of more general or user-defined problem-specific A_p to design new algorithms addressing new COPs. With extended A, the solution space of GCOP increases exponentially, leading to many more new potentially effective algorithms which have been designed automatically. Note that problem-specific features and solution structures are left with users who are familiar with p. The automated algorithm design is handled at a higher level by solving GCOP.

In [21], a unified mathematical formulation for hyperheuristics is proposed as a high-level controller. Elements of heuristic design compete for resources within a shared repository workspace to configure better heuristics. Heuristics interoperate based on information shared from other heuristics. GCOP presents a more general and coherent model and standard with which a set of algorithmic components as the domain of a COP is formally defined and automatically composed.

C. Objective Function of GCOP

With GCOP, new algorithms generated automatically may cater for multiple p with improved level of generality. The newly evolved algorithms c may also reflect a certain level of reusability for other p, saving algorithm development costs.

In GCOP, *Objective* (1) defines the performance measure $F(c)$ on c for solving p. When addressing multiple p, GCOP can be seen as a multiobjective optimization problem where c^* is automatically composed to simultaneously optimize the objective function f_i for each p_i, $i = 1, \ldots, I$, I is the number of p under consideration. Note that the same c^* is used to solve all p, rather than configured individually or manually to solve each p, respectively, i.e., each p acts as a specific problem instance for c^*. *Objective* (2) can be defined to measure performance $F_m(c)$ of c for solving multiple p.

$$F_m(c^*) \leftarrow \mathbf{min}\{f_1, f_2, \ldots, f_I\}^T. \tag{2}$$

In *Objective* (1) and *Objective* (2), F could be the same as f, which reflects direct evaluation of $s \in S$ for p. $F(c)$ could also be different measures of c producing the corresponding s. For example, in hyperheuristics, rewards or aggregated scores have been used to assess the performance of low-level heuristics during the search. Such rewards, rather than direct solution evaluation $f(s)$, can be used in $F(c)$ to provide informed search for GCOP.

In addition to solution quality as the evaluation $F_m(c)$ in *Objective* (2), further extensions and variants could be defined to evaluate different aspects of the automatically designed c, details discussed in Section IV on future research directions.

TABLE II Characteristics of Search Spaces: GCOP versus. *p*.

	***C* of GCOP**	***S* of *P***
ENCODING	COMPOSITIONS c UPON $a \in A$	DIRECT SOLUTIONS s FOR p
OPERATOR	ANY METHODS COMBINING a	SEARCH OPERATORS ON $s \in S$
UPPER BOUND	DEPENDS ON $\|A\|$ AND PARAMETERS OF a	DEPENDS ON THE NO. OF VARIABLES IN s
OBJECTIVE FUNCTION	PERFORMANCE OF c THAT PRODUCES s	QUALITY MEASURE OF s FOR p

D. Search Spaces of GCOP

The search space C for GCOP presents some interesting and unique characteristics compared to that of S for p, see Table II. They can be defined based on the following three factors:

- Solution encoding: represents all solutions based on some finite alphabet for the decision variables. The encoding of $c \in C$ is highly different from that of $s \in S$, leading to different upper bounds of C for GCOP and S for p.
- Successor operator: modifies values of the decision variables thus defines connections of the encoded solutions in the search space. The successor operators in C and S operate upon c and s of different encodings.
- Objective function: evaluates the solutions. In GCOP, $F(c)$ assesses the performance of c, which produces $s \in S$. $F(c)$ thus depends on $f(s)$, however, may potentially be different from $f(s)$, see the examples in Section III.

In solving GCOP, assume c for the decision variables are encoded as one-dimensional strings of $a \in A$ in Table I. With the simple encoding c for GCOP, it is possible and highly useful to analyze the landscape of C, whose spatial structure can be measured using a simple distance metric D on c. It is shown to be very difficult, if not impossible, to analyze the landscape of S for many complex p with d-dimensional solution encodings, $d \geq 2$, see the example COPs in Section III.

In [22], the concept of two search spaces, namely high-level heuristic space and low-level solution space in hyperheuristics, has been introduced into the scientific literature of search algorithms. Fitness landscape analysis on local optimal solutions in the heuristic space revealed interesting findings [22], [23]. The search space of selection constructive hyperheuristics has also been analyzed in [24] to reveal common landscape features of this type of algorithms.

III. Example Methods for GCOP

With the GCOP model, a large number of algorithms could be defined with a subset of $a \in A_{1.0}$ in Table I. Furthermore, GCOP provides a standard toward automated algorithm design. With $a \in A$, different high-level methods, e.g., local search, tools, rules or models could be used to compose a flexibly and design new algorithms automatically in a bottom-up way. Hyperheuristics can be seen as one type of GCOP implementation which combines low-level heuristics, which are defined and integrated based on a subset of basic $a \in A_{1.0}$ in Table I, to design new algorithms automatically. That is, the low-level heuristics are "hard-wired" with the basic a thus requiring a certain level of human knowledge.

This section presents how GCOP with $a \in A_{1.0}$ in Table I defines various selection hyperheuristics. Two representative COPs, namely the Vehicle Routing Problem (VRP) [25] in Section A, and Nurse Rostering Problems (NRP) [26] in Section B are selected as the p in GCOP. The aim is not to provide an exclusive review on all existing selection hyperheuristics, but to demonstrate the modeling of various search algorithms in one GCOP standard toward automated algorithm design.

A. GCOP Methods for Vehicle Routing Problems

As one of the most studied COPs, VRP and its variants have been used to model a range of real-world applications, e.g., transport logistics in supply chain with different constraints. The basic VRP involves constructing a set of closed routes from and to a depot, each route delivering the required demands to an ordered list of customers by a vehicle of limited capacity. The objective is to minimize the total costs (e.g., distance and/or vehicles), while satisfying the capacity constraints. Evolutionary algorithms and computational intelligence techniques have been extensively studied for VRP variants with complex constraints [25], [27]–[29].

In GCOP, c upon the configured a operates on s for VRP. In the most commonly used solution encoding in the VRP literature, customers or tasks are modeled as nodes (i.e., decision variables $s_{i,j}$ for p) in a directed routing network. A large number of operators in VRP algorithms can be modeled using the basic generic $a \in A_{1.0}$, as shown in Table III. For example, *swap*, *interchange* or *k-opt* can be defined by $o^{\text{in}}_{\text{xchg}}(k, m, h1_w)$ or $o^{bw}_{\text{xchg}}(k, m, h1_w)$ in Table I to swap values between k and m decision variables within the same route or between different routes, respectively.

In Table IV, various selection hyperheuristics can be defined using the GCOP standard with subsets of $a \in A_{1.0}$ in Table III for different p, e.g., capacitated VRP (CVRP), distance based VRP (DbVRP), dynamic VRP (DVRP), open VRP (OVRP), and VRP with time windows (VRPTW). Some frameworks, e.g., HyFlex [33], POEMS [34] and ALNS [35], have been adopted to develop the hyperheuristics, i.e., GCOP composition methods. A number of problem-specific components (e.g., *saving*,

TABLE III Operators for VRP modeled as $a \in A_{1.0}$ in GCOP.

	$a \in A_{1.0}$ **IN GCOP FOR VRP** $h1_w, h1_b$: SELECTION CRITERIA/HEURISTICS
$o_{\text{ins}}(k, h1_w, h1_b)$	*GREEDY, INSERTION* [30]: INSERT k NODES CHOSEN BY $h1_w$ TO A ROUTE CHOSEN BY $h1_b$.
$o_{\text{chg}}(k, h1_w, h1_b)$	*SHIFT* [31]: USE $h1_b$ TO CHANGE k NODES SELECTED BY $h1_w$.
$o_{\text{xchg}}(k, m, h1_w)$	k-OPT [31], *INTERCHANGE, VAN BREEDAM* [32]: SWAP k AND m NODES SELECTED BY $h1_w$.
$o_{xo}(k, m, h2_b)$	*CROSSOVER*: EXCHANGE SUB-ROUTES OF k AND m NODES BETWEEN TWO SOLUTIONS CHOSEN BY $h2_b$.
$o_{rr}(k, h1_w, h1_b)$	*DESTROY AND REPAIR*: REMOVE k NODES CHOSEN BY $h1_w$, AND RE-ASSIGN THEM USING $h1_b$.

TABLE IV Selection hyperheuristics defined as GCOP methods for VRP.

$A_{1.0}$	***GCOP METHODS*, F, p**
$o_{rr}(k, h1'_w, h1'_b)$ $a_{sa}(n, t, r)$	ADAPTIVE LARGE NEIGHBORHOOD SEARCH [36]. F: SCORE OF o. p: VRP VARIANTS.
$o_{\text{asg}}(1)$ $o_{\text{ins}}(k, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(k, m, h1_w)$	EVOLUTIONARY APPROACH [32], $h1$: ORDERING HEURISTICS [31]. $F = f(s)$. p: DVRP
$o_{\text{asg}}(1, h1_w, h1_b)$ $o_{rr}(k, h1'_w, h1'_b)$ $o^{bw}_{\text{ins}}(1, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(1, 1, h1_w)$ $o^{bw}_{\text{ins}}(2, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(2, 2, h1_w)$	COALITION-BASED METAHEURISTIC [37] WITH LEARNING MECHANISMS. F: CREDIT/REWARD FROM LEARNING. p: DbVRP, CVRP
$o_{\text{ins}}(k, h1_w, h1_b)$ $o_{\text{asg}}(1, h1_w, h1_b)$ $o_{\text{xchg}}(k, m, h1_w)$	EVOLUTIONARY-BASED SEARCH [38] IN POEMS, $h1$: ORDERING HEURISTICS. $F = f(s)$. p: CVRP
$o_{\text{xchg}}(k, m, h1_w, h1_b)$ $o_{rr}(k, h1'_w, h1'_b)$ $o_{\text{ins}}(k, h1_w, h1_b)$ $o_{\text{chg}}(1, h1_w, h1_b)$ $o_{xo}(k, m, h2_b)$ $a_{oi}(n)$	CLASSIFIER TRAINED BY APPRENTICESHIP LEARNING [39] IN HyFlex. VARIOUS $h1$ USED WITH o. F: change of $f(s)$. p: VRPTW
$a_{mc}(n)$ $o_{\text{chg}}(2, h1_w)$ $o^{\text{in}}_{\text{xchg}}(1, 1, h1_w)$ $o^{bw}_{\text{xchg}}(1, 1, h1_w)$ $a_{mc}(n)$	MULTI-ARMED-BANDIT MECHANISM [40], $h1$: RANDOM SELECTION. F: ACCUMULATED REWARD FOR o. p: VRPTW
$o^{\text{in}}_{\text{xchg}}(1, 1, h1_w)$ $o^{bw}_{\text{ins}}(1, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(1, 1, h1_w)$ $a_{sa}(n, t, r)$	COOPERATION COEVOLUTION APPROACH [41], $h1$: GREEDY STRATEGY. $F = f(s)$. p: REAL VRP
$o_{\text{xchg}}(k, m, h1_w, h1_b)$ $o_{rr}(k, h1'_w, h1'_b)$ $o_{\text{ins}}(k, h1_w, h1_b)$ $o_{\text{chg}}(1, h1_w, h1_b)$ $o_{xo}(k, m, h2_b)$ $a_{oi}(n)$	ADAPTIVE ITERATED LOCAL SEARCH [42] IN HyFlex. F: PERFORMANCE SCORE OF o. p: VRPTW
$o_{\text{xchg}}(k, m, h1_w, h1_b)$ $o_{\text{ins}}(k, h1_w, h1_b)$ $o_{\text{chg}}(1, h1_w, h1_b)$ $o_{rr}(k, h1'_w, h1'_b)$ $o_{xo}(k, m, h2_b)$ a_{all}	TIME DELAY NEURAL NETWORK CLASSIFIER [43] IN HyFlex. F: PERFORMANCE OF o. p: OVRP
$o_{\text{xchg}}(k, m, h1_w, h1_b)$ $o_{\text{ins}}(k, h1_w, h1_b)$ $o_{\text{chg}}(1, h1_w, h1_b)$ a_{all} $o_{xo}(k, m, h2_b)$	ITERATED LOCAL SEARCH WITH DYNAMIC MULTI-ARMED BANDITS [44] IN HyFlex. $h1$: LOCATION/TIME BASED HEURISTICS. F: EXTREME VALUE CREDIT ASSIGNMENT TO o. p: VRPTW
VRP $a \in A_{vrp}$	**VRP a IN GCOP METHODS**
λ-*opt* [45]	EXCHANGE λ EDGES IN A ROUTE [32].
SWEEP [38] [46], [47]	CLUSTER NODES FROM THE DEPOT, EACH SOLVED AS A TSP TO FORM ONE ROUTE [32], [37].
2-*opt** [48]	$o^{bw}_{\text{xchg}}(1, h1_w)$ [40], [42], [43], [44]
SAVING [31]	MERGE TWO ROUTES INTO ONE BASED ON SAVED COSTS [32], [38], [40], [44]
GENI	$o^{bw}_{\text{ins}}(1, h1_w, h1_b)$ FOLLOWED BY A RE-OPTIMIZATION [38], [39], [42], [44].

sweep, *λ-opt*, *2-opt** and *GENI*) in the VRP literature can be defined as a plug-in A_{vrp} in Table IV in the extended GCOP.

In Table IV, the GCOP composition methods in selection hyperheuristics for VRP range from various search methods to classifiers trained using machine learning. Of particular interest is the research on c for different COPs developed in the HyFlex framework with a set of built-in low-level heuristics. With the consistent GCOP standard, further investigations on c composed more flexibly with the automatically selected $a \in A_{1.0}$ could gain useful insights on the effectiveness of complicated as well as simple c for various COPs.

In Table IV, in addition to using the direct evaluation on s for p, i.e., $f(s)$, $F(c)$ for GCOP has also used various assessment metrics to evaluate the effectiveness of a in c. Instead of assessing the final solution, such measurements reflect the short-term performance of a in c during problem solving, thus providing more informed decision making composing a into the generated algorithm c.

Some of the components in Table IV can be seen as compound operators, integrating more than one a in Table I. For example, o_{xchg} can be seen as applying o_{asg} twice, and o_{rr} can be seen as applying o_{rm} followed by o_{asg}. A large number of heuristics $h1$ and $h2$ have been used in the literature. For example, in [36], seven different *removal criteria* have been used to remove requests (nodes) from the routes using different measures (e.g., time-oriented, history-based, cluster, worst, and related removals). Two insertion criteria (i.e., greedy and regret) are also used to reassign requests back to routes.

In the literature, some decisions of algorithm design (e.g., the acceptance criteria $A_{1.0_a}$, or the number of n neighbors explored) are not always provided or take some default settings. This leads to some ambiguity in reproducing the published algorithms. With $A_{1.0}$ defined with consistent parametric and heuristic settings within the GCOP standard, these details can be clearly defined with a, thus supporting consistent research in the literature.

TABLE V Low-level heuristics in hyperheuristics for NRP modeled as basic $a \in A_{1.0}$ in GCOP.

$a \in A_{1.0}$	*A* **IN GCOP FOR SOLVING NRP** $h1_w$: SELECTION CRITERIA SUCH AS THE COST OF CONSTRAINT VIOLATIONS, SHIFT TYPE BALANCE, ETC.
$o_{\text{chg}}(k, h1_w, h1_b)$	*CHANGE SHIFT*: USE $h1_b$ TO CHANGE THE SHIFT TYPE OF k NURSES CHOSEN BY $h1_w$.
$o^{bw}_{\text{xchg}}(k, k, h1_w)$	*SWAP SHIFTS*: SWAP k SHIFTS BETWEEN TWO NURSES CHOSEN BY $h1_w$.
$o_{rr}(k, h1_w, h1_b)$	*RUIN AND RECREATE*: USE $h1_b$ TO REASSIGN ALL k SHIFTS OF A SET OF NURSES CHOSEN BY $h1_w$.

TABLE VI Selection hyper-heuristics defined as GCOP methods for NRP.

$A_{1.0}$	**GCOP METHODS,** F**,** p
$o_{\text{chg}}(1, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(1, 1, h1_w)$ $a_{oi}(1)$	CHOICE FUNCTION [49]. *F*: SCORE OF *o*. *p*: UK DATASET
$o_{\text{chg}}(1, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(1, 1, h1_w)$ $a_{oi}(1)$ a_{all}	TABU SEARCH [50]. $F = f(s)$ MEASURES THE FEASIBILITY AND BALANCED SHIFTS. *p*: UK DATASET
$o_{\text{chg}}(1, h1_w, h1_b)$ $o^{bw}_{\text{xchg}}(1, 1, h1_w)$ a_{all} $a_{oi}(1)$	SIMULATED ANNEALING [51]. *F*: MEASURES CONSTRAINT VIOLATIONS. *P*: UK DATASET
$o^{bw}_{xchg}(1, 1, h1_w)$ $a_{gd}(n)$ $a_{oi}(n)$ $a_{ie}(n)$	RANDOM, CHOICE FUNCTION, DYNAMIC STRATEGY [52]. *F*: SCORES OF *o*. *p*: INRC2010
$o_{rr}(k, h1_w, h1_b)$ $a_{oi}(n)$ $a_{sa}(n)$ $a_{late}(n)$ $o^{bw}_{\text{xchg}}(k, m, h1_w)$ $a_{ie}(n)$ a_{all} $a_{gd}(n)$	ADAPTIVE DYNAMIC METHOD [53]. VARIOUS HEURISTICS WITH *o*. *F*: PERFORMANCE METRIC. *P*: INRC2010 DATASET
$o_{rr}(k, h1_w, h1_b)$ a_{all} $o^{bw}_{\text{xchg}}(k, m, h1_w, h1_b)$ $o_{ins}(k, h1_w, h1_b)$ $o_{chg}(1, h1_w, h1_b)$ $o_{xo}(k, m, h2_b)$ $a_{ie}(n)$	ITERATED LOCAL SEARCH WITHIN A FOUR-STAGE APPROACH BASED ON TENSOR ANALYSIS [54]. $F = f(s)$. *p*: NOTTINGHAM DATASET
NRP $a \in A_{nrp}$	**NRP** a **in GCOP methods**
o upon pre-defined shift patterns for specific *p*	BAYESIAN NETWORK [55] LEARNS TO SELECT A SET OF GOOD SHIFT PATTERNS FOR *p*. *p*: UK DATASET

B. GCOP Methods for Nurse Rostering Problems

The NRP has received extensive research attention in the last five decades [26] due to the high demands of quality healthcare, limited resources, and various legislation around the world. The problem consists of assigning a set of nurses of different skills to a set of different shifts on each day of a scheduling period. A set of hard constraints must be satisfied, including the legislation (e.g., maximum consecutive shifts) and coverage (i.e., all demands must be covered). The objective is to minimize the violations of soft constraints, e.g., personal preferences, free weekend, and preferred shift patterns. Algorithms investigated include exact methods, evolutionary algorithms and hyperheuristics, and many more [26].

With the GCOP standard, most of the low-level heuristics in hyperheuristics and components in other algorithms in the NRP literature can be modeled as $a \in A_{1.0}$ in Table V. They all operate on shifts assigned to selected nurses working on particular days (i.e., decision variables in p). For example, $o^{bw}_{\text{xchg}}(k, k, h1_w)$ swaps k shifts of two nurses selected by $h1_w$. If shifts of one of the nurses are empty (no shift assigned), it defines the *move shift* operator in the NRP literature.

The GCOP composition methods in Table VI employed various local search algorithms and different techniques for NRP. As for VRP, F in GCOP for NRP measures either the quality of the resulting s or the performance of a. Compared to VRP, there are not many problem-specific a in NRP, i.e., most of the a in Table V are in $A_{1.0}$. Also, various acceptance criteria have been studied for NRP, which is not the case for VRP. Note that most widely used acceptance criteria in $A_{1.0}$ are not problem-specific, and can be used across different COPs.

Only three benchmark datasets (i.e., the INRC2010 competition [56], Nottingham and UK [55] datasets) have been widely tested in NRP, the first two with extensive NRP variants. An interesting study on INRC2010 [52] focuses on choosing a compact set of low-level heuristics. The methods developed showed to be highly effective on the benchmark as well as two real-world scheduling problems. Such analysis can also be conducted automatically with the GCOP model. In [22], it is found that simple configuration methods work as effective as complicated algorithms on effective low-level heuristics. Such studies provide useful insights on $a \in A_{1.0}$ for further research in GCOP.

Recent research has made some progress on the generality of algorithms. However, reusability of algorithms remains underexplored.

In this paper, we define the most basic $a \in A_{1.0}$ across COPs, aiming to establish the fundamentals of the GCOP standard. More advanced research will be conducted as discussed in Section IV, and also strongly encouraged from the research communities, to further enhance the GCOP standard toward automated algorithm design. Following the recommended good practice in OR [20], updates of extensions and resources on GCOP will be provided at a dedicated GCOP website at https://sites.google.com/view/general-cop.

IV. Discussions and Future Directions

As a fast emerging topic in computational intelligence, evolutionary computation and optimization research, automated algorithm design has recently attracted increasing research attention. In this paper, GCOP is formally established as a new standard to define various search algorithms in one model, providing the fundamentals and opening a number of potential new research directions in automated algorithm design.

New knowledge toward automated algorithm design: The new GCOP model provides a standard for systematic analysis on the basic a of different behaviors in the optimized c. Some studies in hyperheuristics identified a compact subset of effective low-level heuristics and revealed synergy among them, enabling effective methods to be built [53]. In [57], a runtime analysis on a selection hyperheuristic shows that online reinforcement learning for configuring operators may perform poorer than a fixed distribution of operators. These analyses could also be conducted within the consistent GCOP model. The balanced intensification and diversification may be modeled in c considering synergy among a. New findings on new effective algorithms with different categories of algorithmic components may also lead to new knowledge and deeper understanding in algorithm design and introduce new effective algorithms to the literature.

Generality and reusability of algorithms: In GCOP, the automatically designed new algorithms evolve to perform well for solving different p, thus may cater for similar types of new p with a certain level of generality and reusability. Recent research has made some progress on the generality of algorithms. However, the reusability of algorithms remains underexplored. With the GCOP standard, the optimized components may be analyzed to derive new knowledge potentially transferable to solve unseen p. GCOP may contribute to addressing the challenging research issue of generality and reusability of algorithms.

- Generality: Recent hyperheuristics have shown to be able to address cross-domain COPs [58]. There is to our knowledge, however, not yet a formal definition of algorithm generality in the literature. In [59], a new assessment method has been proposed to evaluate hyperheuristics against four levels of generality, in terms of solving different problem domains, problems, problem instances and benchmarks, respectively. The automatically generated new algorithms with GCOP can be evaluated against these four levels of generality for solving different p. Note that the assessment of algorithm optimality is different from that of algorithm generality. The latter may also measure the robustness and speed, in addition to solution quality for multiple problems/domains.
- Reusability: Recent research has made some progress on reusing algorithms, although the main research focus may not be exactly on reusability. For example, the automatically selected algorithms on training instances [10], [11] could be reused to solve testing instances of certain similar features. In generation hyperheuristics [60], new heuristics can be automatically generated by using genetic programming based on problem state features [61], [62], thus could be potentially reusable for problems of similar features. However, the problem of code bloat may lead to the issues of readability and interpretability [63].

Fundamentals of GCOP: Advanced theoretical investigations are needed to underpin the fundamentals of the new GCOP model in operational research.

- Evaluation of GCOP: In solving GCOP, the objective function can be extended with multiple objectives including generality, reusability and computational time. The new performance measure in [59] can be adopted in the objective function to measure different levels of generality. In GCOP, instead of designing algorithms using human expertise, as happens in most of the research, the time is spent on automatically searching for or composing the optimal c^* for p. The trade-off between solving a number of p and the increased computational time presents another interesting research issue. The c for each p can be further evaluated in F to assess its convergence, the number of operations and number of fitness evaluations used, using different statistical measures as shown in [64].
- No Free Lunch Theorem (NFL): Another interesting research issue is how NFL applies in solving GCOP, that is to explore the scope of generality for the generated new c.

To our knowledge, this is the first standard in the literature defining a large number of search algorithms in one common model.

In [65], the conditions under which the NFL applies to hyperheuristics are discussed. It is concluded that there may be a free lunch developing general methods for a set of problems with fitness functions which are not *closed under permutation*.

- Landscape analysis: Interesting features of the GCOP search space may reveal more findings for automated algorithm design. A theoretical study on selection constructive hyperheuristics for COPs [24] revealed that there were often large plateau and high correlation between local and global optimal heuristic combinations in the heuristic landscapes. In GCOP, each of the generated c leads to a different S for p. Solving GCOP is thus equivalent to exploring multiple S compared to traditional search algorithms employing manually fixed a for solving p. It is interesting to investigate the increased exploration ability and effectiveness of GCOP exploring multiple c.

Extensions of GCOP: GCOP facilitates automated algorithm design, aiming to reduce the development costs and barriers of expertise required for designing algorithms. Researchers and practitioners can focus on establishing and exchanging a better understanding of algorithm developments to address different p.

- Extensions of A: Based on more findings in evolutionary computation, $A_{1.0}$ could be extended with more effective common a. Those a used in the literature (Table III for VRP and Table V for NRP) represent only a subset of $A_{1.0}$ in Table I. The library of general and basic a can be easily extended with problem-specific A_p and is portable to solve a wider range of p. Such efforts are highly valuable and strongly encouraged to promote future advances of automated algorithm design. Resources will be updated at the GCOP web site.
- Other optimization problems: Recent research has developed effective selection hyperheuristics for continuous optimization problems [66], [67]. The GCOP model can also be extended to solve other optimization problems in addition to COPs.

V. Conclusions

This position paper introduces the General Combinatorial Optimization Problem (GCOP) as a new standard for algorithm design. The objective of GCOP is to optimize the compositions of basic algorithmic components, i.e., the decision variables, to automatically design new algorithms for solving different optimization problems. A taxonomy of automated algorithm design, i.e., automated algorithm configuration, automated algorithm selection and automated algorithm composition, has been formally defined. GCOP which standardizes various search algorithms within one model underpins the fundamentals of automated algorithm design.

With the new GCOP model, we define a set $A_{1.0}$ of the mostly used basic elementary algorithmic components for widely studied combinatorial optimization problems in the literature. This set can be extended to include more common components, as well as user-defined problem-specific components A_p for different optimization problems. With the new standard, we also demonstrate the application of the GCOP model with $A_{1.0}$ and A_p to formally define a large number of selection hyperheuristics for solving two benchmark combinatorial optimization problems, namely vehicle routing and nurse rostering problems. This can be seen as the implementation of the GCOP standard, demonstrating its effectiveness for modeling a large number of existing algorithms. To our knowledge, this is the first standard in the literature defining a large number of search algorithms in one common model.

The established new GCOP opens a new line of interesting research directions in optimization research. Further studies will investigate theoretical issues including landscape analysis on the search spaces of GCOP. The objective function can be extended to measure the generality, reusability and computational time of the newly generated algorithms. The new automatically designed algorithms introduced to the literature brings new knowledge which can be used to design new effective algorithms, and reused for solving other optimization problems. In addition to combinatorial optimization problems, continuous optimization problems and multiobjective optimization problems could also be addressed with extended GCOP models.

With the new GCOP standard, this position paper calls for further investigations on the emerging topic of automated algorithm design to stimulate more advances in evolutionary computation and optimization research. We strongly encourage future research in the research communities to adopt and extend the GCOP standard. Resources and latest developments will be continuously updated at the GCOP website.

References

[1] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1982.

[2] J. Beasley, "OR-library: Distributing test problems by electronic mail," *J. Oper. Res. Soc.*, vol. 41, no. 11, pp. 1069–1072, Nov. 1990. doi: 10.2307/2582903.

[3] F. Hutter, H. Hoos, and T. Stützle, "Automatic algorithm configuration based on local search," in *Proc. Nat. Conf. Artificial Intelligence*, Vancouver, July 22–26, 2007, pp. 1152–1157.

[4] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *J. Artificial Intell. Res.*, vol. 36, pp. 267–306, Oct. 2009. doi: 10.1613/jair.2861.

[5] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-race and iterated F-race: An overview," in *Proc. Experimental Methods for the Analysis of Optimization Algorithms*, Oct. 2010, pp. 311–336. doi: 10.1007/978-3-642-02538-9_13.

[6] M. López-Ibáñez and T. Stützle, "The automatic design of multi-objective ant colony optimization algorithms," *IEEE Trans. Evol. Comput.*, vol. 16, no. 6, pp. 861–875, Feb. 2012. doi: 10.1109/TEVC.2011.2182651.

[7] F. Pagnozzi and T. Stützle, "Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems," *Eur. J. Oper. Res.*, vol. 276, pp. 409–421, July 2019. doi: 10.1016/j.ejor.2019.01.018.

[8] T. Adamo, G. Ghiani, A. Grieco, E. Guerriero, and E. Manni, "MIP neighborhood synthesis through semantic feature extraction and automatic algorithm configuration," *Comput. Oper. Res.*, vol. 83, pp. 106–119, July 2017. doi: 10.1016/j.cor.2017.01.021.

[9] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Sept. 2016. doi: 10.1016/j.orp.2016.09.002.
[10] S. Liu, K. Tang, and X. Yao, "Automatic construction of parallel portfolios via explicit instance grouping," in *Proc. AAAI Conf. Artificial Intelligence*, New Orleans, Feb. 2–7, 2018. doi: 10.1609/aaai.v33i01.33011560.
[11] K. Tang, F. Peng, G. Chen, and X. Yao, "Population-based algorithm portfolios with automated constituent algorithms selection," *Inf. Sci.*, vol. 279, pp. 94–104, Sept. 2014. doi: 10.1016/j.ins.2014.03.105.
[12] R. Akay, A. Basturk, A. Kalinli, and X. Yao, "Parallel population-based algorithm portfolios: An empirical study," *Neurocomputing*, vol. 247, pp. 115–125, July 2017. doi: 10.1016/j.neucom.2017.03.061.
[13] L. Xu, H. Hoos, and K. Leyton-Brown, "Hydra: Automatically configuring algorithms for portfolio-based selection," in *Proc. AAAI Conf. Artificial Intelligence*, Atlanta, July 11–15, 2010.
[14] L. Bezerra, M. Lòpez-Ibáñez, and T. Stützle, "Automatic design of evolutionary algorithms for multi-objective combinatorial optimization," in *Proc. Parallel Problem Solving from Nature*, Ljubljana, Sept. 13–17, 2014, pp. 508–517. doi: 10.1007/978-3-319-10762-2_50.
[15] M. Oltean, "Evolving evolutionary algorithms using linear genetic programming," *Evol. Comput.*, vol. 13, no. 3, pp. 387–410, Sept. 2005. doi: 10.1162/1063656054794815.
[16] N. Pillay and R. Qu, Eds., *Hyper-heuristics: Theory and Applications*. Springer-Verlag, 2019.
[17] E. Burke et al., "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, pp. 1695–1724, July 2010. doi: 10.1057/jors.2013.71.
[18] E. Burke et al., "The cross-domain heuristic search challenge: An international research competition," in *Proc. Intelligent Conf. Learning and Intelligent Optimization*, Rome, Jan. 17–21, 2011, pp. 631–634. doi: 10.1007/978-3-642-25566-3_49.
[19] N. Pillay and D. Beckedahl, "EvoHyp: A Java toolkit for evolutionary algorithm hyper-heuristics," in *Proc. IEEE Congr. Evolutionary Computation*, San Sebastian, June 5–8, 2017, pp. 2707–2713. doi: 10.1109/CEC.2017.7969636.
[20] G. Kendall et al., "Good laboratory practice for optimization research," *J. Oper. Res. Soc.*, vol. 67, no. 4, pp. 676–689, Apr. 2016. doi: 10.1057/jors.2015.77.
[21] J. Swan, J. Woodward, E. Özcan, G. Kendall, and E. Burke, "Searching the hyper-heuristic design space," *Cogn. Comput.*, vol. 6, no. 1, pp. 66–73, 2014. doi: 10.1007/s12559-013-9201-8.
[22] R. Qu and E. Burke, "Hybridisations withing a graph based hyper-heuristic framework for university timetabling problems," *J. Oper. Res. Soc.*, vol. 60, pp. 1273–1285, Sept. 2009. doi: 10.1057/jors.2008.102.
[23] G. Ochoa, R. Qu, and E. Burke, "Analyzing the landscape of a graph based hyper-heuristic for timetabling problems," in *Proc. Genetic and Evolutionary Computation Conf.*, Montreal, July 8–12, 2009, pp. 341–348. doi: 10.1145/1569901.1569949.
[24] R. Qu, N. Pillay, and D. Beckedahl, "A fundamental study on selection constructive hyper-heuristics," *IEEE Trans. Evol. Comput.*, to be published.
[25] P. Toth and D. Vigo, "An overview of vehicle routing problems," in *The Vehicle Routing Problem*, 2002, pp. 1–26.
[26] E. Burke, P. D. Causmaecker, G. Berghe, and H. Landeghem, "The state of the art of nurse rostering," *J. Schedul.*, vol. 7, no. 6, pp. 441–499, Nov. 2004. doi: 10.1023/B:JOSH.0000046076.75950.0b.
[27] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part II: Metaheuristics," *Transportation Sci.*, vol. 39, no. 1, pp. 119–139, Feb. 2005. doi: 10.1287/trsc.1030.0057.
[28] M. Gendreau, G. Laporte, and J.-Y. Potvin, "Metaheuristics for the capacitated VRP," in *The Vehicle Routing Problem (SIAM Monographs Discrete Mathematics and Applications)*, 2002, pp. 129–154.
[29] U. Ritzinger, J. Puchinger, and R. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *Int. J. Prod. Res.*, vol. 54, no. 1, pp. 215–231, Jan. 2016. doi: 10.1080/00207543.2015.1043403.
[30] R. Mole and S. Jameson, "A sequential route-building algorithm employing a generalised savings criterion," *Oper. Res. Quart.*, vol. 27, no. 2, pp. 503–511, June 1976. doi: 10.1057/jors.1976.95.
[31] G. Laporte, M. Gendreau, J. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *Int. Trans. Oper. Res.*, vol. 7, nos. 4–5, pp. 285–300, Sept. 2000. doi: 10.1111/j.1475-3995.2000.tb00200.x.
[32] P. Garrido and M. Riff, "DVRP: A hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic," *J. Heuristics*, vol. 16, no. 6, pp. 795–834, Dec. 2010. doi: 10.1007/s10732-010-9126-2.
[33] G. Ochoa et al., "HyFlex: A benchmark framework for cross-domain heuristic search," in *Proc. Evolutionary Computational Combinatorial Optimization*, Málaga, Apr. 11–13, 2012, pp. 136–147. doi: 10.1007/978-3-642-29124-1_12.
[34] J. Kubalik and J. Faigl, "Iterative prototype optimisation with evolved improvement steps," in *Proc. European Conf. Genetic Programming*, Budapest, Apr. 10–12, 2006, p. 154–165. doi: 10.1007/11729976_14.
[35] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transp. Sci.*, vol. 40, no. 4, pp. 455–472, Nov. 2006. doi: 10.1287/trsc.1050.0135.
[36] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2403–2435, Aug. 2007. doi: 10.1016/j.cor.2005.09.012.
[37] D. Meignan, A. Koukam, and J. Créput, "Coalition-based metaheuristic: A self-adaptive metaheuristic using reinforcement learning and mimetism," *J. Heuristics*, vol. 16, no. 6, pp. 859–879, Dec. 2010. doi: 10.1007/s10732-009-9121-7.
[38] J. Mlejnek and J. Kubalik, "Evolutionary hyperheuristic for capacitated vehicle routing problem," in *Proc. Annu. Conf. Genetic and Evolutionary Computation*, Amsterdam, July 6–10, 2013, pp. 219–220. doi: 10.1145/2464576.2464684.
[39] S. Asta and E. Özcan, "An apprenticeship learning hyper-heuristic for vehicle routing in HyFlex," Dec. 9–12, 2014, pp. 1474–1481. doi: 10.1109/EALS.2014.7009505.
[40] N. Sabar, X. Zhang, and A. Song, "A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows," in *Proc. IEEE Congr. Evolutionary Computation*, Sendai, May 25–28, 2015. doi: 10.1109/CEC.2015.7256977.
[41] P.-Y. Yin, S.-R. Lyu, and Y.-L. Chuang, "Cooperative coevolutionary approach for integrated vehicle routing and scheduling using cross-dock buffering," *Eng. Appl. Artif. Intell.*, vol. 52, pp. 40–53, June 2016. doi: 10.1016/j.engappai.2016.02.006.
[42] J. Walker, G. Ochoa, M. Gendreau, and E. Burke, "Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework," in *Proc. Int. Conf. Learning and Intelligent Optimization*, Paris, Jan. 16–20, 2012, pp. 265–276. doi: 10.1007/978-3-642-34413-8_19.
[43] R. Tyasnurita, E. Özcan, and R. John, "Learning heuristic selection using a time delay neural network for open vehicle routing," in *Proc. IEEE Congr. Evolutionary Computation*, San Sebastian, June 5–8, 2017, pp. 1474–1481. doi: 10.1109/CEC.2017.7969477.
[44] J. Soria-Alcaraz, G. Ochoa, M. Sotelo-Figeroa, and E. Burke, "A methodology for determining an effective subset of heuristics in selection hyper-heuristics," *Eur. J. Oper. Res.*, vol. 260, no. 3, pp. 972–983, Aug. 2017. doi: 10.1016/j.ejor.2017.01.042.
[45] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Labs Tech. J.*, vol. 44, no. 10, pp. 2245–2269, Dec. 1965. doi: 10.1002/j.1538-7305.1965.tb04146.x.
[46] B. Gillett and L. Miller, "A heuristic algorithm for the vehicle dispatch problem," *Oper. Res.*, vol. 22, no. 2, pp. 340–349, Apr. 1974. doi: 10.1287/opre.22.2.340.
[47] A. Wren and A. Holliday, "Computer scheduling of vehicles from one or more depots to a number of delivery points," *J. Oper. Res. Soc.*, vol. 23, no. 3, pp. 333–344, Sept. 1972. doi: 10.1057/jors.1972.53.
[48] G. Clarke and J. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Oper. Res.*, vol. 12, no. 4, pp. 568–581, July–Aug. 1964. doi: 10.1287/opre.12.4.568.
[49] P. Cowling, G. Kendall, and E. Soubeiga, "Hyper-heuristics: A robust optimization method applied to nurse scheduling," in *Proc. Parallel Problem Solving from Nature*, Granada, Sept. 7–11, 2002, pp. 851–860. doi: 10.1007/3-540-45712-7_82.
[50] E. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *J. Heuristics*, vol. 9, no. 6, pp. 451–470, Dec. 2003. doi: 10.1023/B:HEUR.0000012446.94732.b6.
[51] R. Bai, E. Burke, G. Kendall, J. Li, and B. McCollum, "A hybrid evolutionary approach to the nurse rostering problem," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 580–590, July 2011. doi: 10.1109/TEVC.2009.2033583.
[52] B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, and G. Berghe, "One hyper-heuristic approach to two timetabling problems in health care," *J. Heuristics*, vol. 18, no. 3, pp. 401–434, June 2012. doi: 10.1007/s10732-011-9192-0.
[53] M. Misir, K. Verbeeck, P. D. Causmaecker, and G. Berghe, "An investigation on the generality level of selection hyper-heuristics under different empirical conditions," *Appl. Soft Comput.*, vol. 13, no. 7, pp. 3335–3353, July 2013. doi: 10.1016/j.asoc.2013.02.006.
[54] A. Shahriar, E. Özcan, and T. Curtois, "A tensor based hyper-heuristic for nurse rostering," *Knowl.-Based Syst.*, vol. 98, no. 1, pp. 185–199, Apr. 2016. doi: 10.1016/j.knosys.2016.01.031.
[55] U. Aickelin and J. Li, "An estimation of distribution algorithm for nurse scheduling," *Ann. Oper. Res.*, vol. 155, pp. 289–309, July 2007. doi: 10.1007/s10479-007-0214-0.
[56] S. Haspeslagh, P. D. Causmaecker, A. Schaerf, and M. Stlevik, "The first international nurse rostering competition 2010," *Ann. Oper. Res.*, vol. 218, no. 1, pp. 221–236, July 2014. doi: 10.1007/s10479-012-1062-0.
[57] P. Lehre and E. Özcan, "A runtime analysis of simple hyper-heuristics: To mix or not to mix operators," in *Proc. Workshop on Foundations of Genetic Algorithms*, Adelaide, Jan. 16–20, 2013, pp. 97–104. doi: 10.1145/2460239.2460249.
[58] G. K. R. Q. N. R. Sabar and M. Ayob, "A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 217–228, Feb. 2015. doi: 10.1109/TCYB.2014.2323936.
[59] N. Pillay and R. Qu, "Assessing hyper-heuristic performance," *European J. Oper. Res.*, to be published. doi: 10.1016/j.ejor.2008.07.023.
[60] E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of Metaheuristics*, 2010, pp. 449–468.
[61] M. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework," *Memet. Comput.*, vol. 1, pp. 205–219, Nov. 2009. doi: 10.1007/s12293-009-0022-y.
[62] E. Burke, M. Hyde, G. Kendall, and J. Woodward, "A genetic programming hyper-heuristic approach for evolving two dimensional strip packing heuristics," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 942–958, June 2010. doi: 10.1109/TEVC.2010.2041061.
[63] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, 1998.
[64] K. Sörensen, "Metaheuristics: The metaphor exposed," *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 3–18, Jan. 2015. doi: 10.1111/itor.12001.
[65] R. Poli and M. Graff, "There is a free lunch for hyper-heuristics, genetic programming and computer scientists," in *Proc. European Conf. Genetic Programming*, Tubingen, Apr. 15–17, 2009, pp. 195–207. doi: 10.1007/978-3-642-01181-8_17.
[66] M. Maashi, G. Kendall, and E. Özcan, "Choice function based hyper-heuristics for multi-objective optimization," *Appl. Soft Comput.*, vol. 28, pp. 312–326, Mar. 2015. doi: 10.1016/j.asoc.2014.12.012.
[67] D. Walker and E. Keedwell, "Multi-objective optimisation with a sequence-based selection hyper-heuristic," in *Proc. Conf. Genetic and Evolutionary Computation*, Denver, July 20–24, 2016, pp. 81–82. doi: 10.1145/2908961.2909016.