University of Nottingham
UK | CHINA | MALAYSIA

University of Nottingham

Automated Design of Local Search Algorithms for Vehicle Routing Problems with Time Windows

Weiyao Meng

14342160

Supervised by

First Supervisor: Rong Qu
Second Supervisor: Dario Landa-Silva

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

June 2023

Abstract

Designing effective search algorithms for solving combinatorial optimisation problems presents a challenge for researchers due to the time-consuming experiments and experience required in decision-making. Automated algorithm design removes the heavy reliance on human experts and allows the exploration of new algorithm designs. This thesis systematically investigates machine learning for the automated design of new and generic local search algorithms, taking the vehicle routing problem with time windows as the testbed.

The research starts by building AutoGCOP, a new general framework for the automated design of local search algorithms to optimise the composition of basic algorithmic components. Within the consistent AutoGCOP framework, the basic algorithmic components show satisfying performance for solving the VRPTW. Based on AutoGCOP, the thesis investigates the use of machine learning for automated algorithm composition by modelling the algorithm design task as different machine learning tasks, thus investigating different perspectives of learning in automated algorithm design.

Based on AutoGCOP, the thesis first investigates online learning in automated algorithm design. Two learning models based on reinforcement learning and Markov chain are investigated to learn and enhance the compositions of algorithmic components towards automated algorithm design. The Markov chain model presents a superior performance in learning the compositions of algorithmic components during the search, demonstrating its effectiveness in designing new algorithms automatically.

The thesis then investigates offline learning to learn the hidden knowledge of effective algorithmic compositions within AutoGCOP for automated algorithm design. The forecast of algorithmic components in the automated composition is defined as a sequence classification task. This new machine learning task is then solved by a Long Short-term Memory (LSTM) neural

network which outperforms various conventional classifiers. Further analysis reveals that a Transformer network surpasses LSTM at learning from longer algorithmic compositions. The systematical analysis of algorithmic compositions reveals some key features for improving the prediction.

To discover valuable knowledge in algorithm designs, the thesis applies sequential rule mining to effective algorithmic compositions collected based on AutoGCOP. Sequential rules of composing basic components are extracted and further analysed, presenting a superior performance of automatically composed local search algorithms for solving VRPTW. The extracted sequential rules also suggest the importance of considering the impact of algorithmic components on optimisation performance during automated composition, which provides new insights into algorithm design.

The thesis gains valuable insights from various learning perspectives, enhancing the understanding towards automated algorithm design. Some directions for future work are present.

Acknowledgements

This thesis would not have happened without many people who accompanied me during the whole or part of the work in my private and academic life. Therefore, I would like to take this moment to sincerely thank these people and acknowledge their invaluable help.

I am immensely grateful to my supervisors, Professor Rong Qu and Professor Dario Landa-Silva, for their unwavering support and patience throughout my PhD study and research. Their guidance and dedication have played a pivotal role in shaping me into an independent researcher, and have equipped me with essential skills beyond the scope of my studies. I am truly grateful for their encouragement of my personal and professional development.

In addition, I extend my gratitude to Dr Jason Atkin and Professor Christian Wagner for their insightful feedback and constructive comments on the progress of my project. I also would like to express my special thanks to Dr Xingxing Hao from Northwest University for his valuable advice and support during the initial stages of my PhD. Their guidance has been invaluable to my research, and I am truly thankful for their help.

I would like to thank all members of the COL Lab at the University of Nottingham, for contributing to such a vibrant and delightful research environment that has enriched my experience. In particular, I extend special thanks to Dr Salman Arif, Dr Yun He, Dr Warren G. Jackson, Salim Maaji, Han Meng, Rebecca Tickle, and Dr Wenjie Yi from the COL Lab, as well as Dr Feng Chen from the Computer Vision Lab, and Dr Elif E Firat, Zeyang Liu, Dr Yuan Tian, and Wen Zheng from the VisTAG Lab at the University of Nottingham. Their support, insightful discussion and friendship have been invaluable to me. I would also like to acknowledge the constant encouragement and support I received from Dr Jeremie Clos, Dr Peer-Olaf Siebers and Adam Walker.

I would like to extend my heartfelt appreciation to my dear friends back in China, with a special mention to Qing Huang, Yiqing Liu, Yuanwei Li, Xinyi Zhang, and Liyi Pan. Their unwavering support has been a constant source of strength throughout my PhD journey. To my friends, thank you for always being there for me, believing in me, being ready to lend a helping hand and lifting my spirits whenever I lose faith in myself. I am truly grateful for your enduring friendship and support.

Also, I would like to thank my partner Dr Yuzheng Chen from the PEMC group at the University of Nottingham, for bringing me tremendous happiness, for being there with me through every joyful and difficult moment of my PhD journey, and for being my rock and a constant source of support every step of the way.

Last but not least, I am deeply grateful to my parents, Fanyi Meng and Peiyun Miao, as well as all my family back in China. I would not start the journey pursuing my PhD in the UK without their unwavering support. This thesis would not have happened without their endless love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and motivations

An optimisation problem is the problem of searching for the best configuration of a set of variables with the aim of maximising or minimising some objectives (Blum and Roli, 2003). Generally, optimisation problems can be categorised into two groups, i.e., one has solutions encoded with real-valued variables, and the other with solutions encoded with discrete variables (Blum and Roli, 2003). Combinatorial optimisation problems (COPs) fall into the latter category. They aim to find the optimal solution among a finite set of solutions. COPs are well-known for their numerous applications in domains such as scheduling, logistics, network design and operations research (Vesselinova et al., 2020).

Heuristic algorithms, ranging from classic heuristics to meta-heuristics, have been extensively studied in solving complex COPs (Blum and Roli, 2003). These algorithms do not guarantee optimal solutions but often provide a high-quality solution in a reasonable time, thus have been rapidly

developed in recent years in different domains (Talbi, 2009). Extensive research has been dedicated to designing effective meta-heuristics for a variety of practical applications (Boussaïd et al., 2013).

The design of effective heuristic algorithms usually follows an iterative and manual process where a large number of decisions need to be made (Hoos, 2008). Some of the basic and common decisions include representation (solution encoding), objective function, and constraint handling. These decisions are essential steps in algorithm design, however, do not directly determine the specific algorithms that are produced (Talbi, 2009). Other decisions, such as the search scheme (single-solution-based or population-based), the set of algorithmic components (such as operators, acceptance criteria, etc.) and the parameter settings, are directly related to algorithm designs. Given the growing complexity of COPs, the design and development of effective search algorithms present a challenge for researchers due to the extensive expertise and efforts required in decision-making (Pillay et al., 2018b).

In recent studies, a great amount of research effort has been made in automating the algorithm design process. Releasing human experts from the tedious design process is a primary motivation, i.e., to produce effective algorithms with less human involvement in the algorithm design process. In addition, the performance of manually designed algorithms highly relies on the experience and effort of the human experts, who may only consider a limited number of designs, leaving a significant number of potential algorithms unexplored (Hoos, 2008). Automation in algorithm design helps to explore a larger scope of candidate algorithms, some of which may never be considered by manual designs. Therefore, automated algorithm design is becoming an important field of research in different research domains, such as machine learning (He et al., 2021), software engineering (Woodward

et al., 2016), (Petke et al., 2017), optimisation research and evolutionary computation (Pillay et al., 2018b).

Based on a new taxonomy defined in (Qu et al., 2020), the current research in automated algorithm design has been categorised into three themes, namely automated configuration, automated selection and automated composition. Automated configuration aims to automatically select parameter values for specific target algorithm(s) to solve a collection of problem instances (Hutter et al., 2007). Automated selection aims to automatically select the most suitable algorithm from a portfolio of candidate algorithms for a set of training instances thus solving new testing instances. Automated composition aims to automatically compose heuristics or components of arbitrary algorithms to solve the problem at hand (Qu et al., 2020). Automated configuration and selection take a top-down method in algorithm design, to consider parameters and algorithms themselves in the decision space, thus the resulting algorithms are variants of the target algorithms. Automated composition takes a bottom-top method to compose flexibly a set of algorithmic components, thus generating new algorithms (Qu et al., 2020).

The automated algorithm design is fast emerging along with the successful research development in machine learning (Qu, 2021b). Machine learning (ML) algorithms can "learn" knowledge from data to make predictions or decisions without being explicitly programmed to do so (Bishop and Nasrabadi, 2006). From the aspect of machine learning, the search process of heuristic algorithms generates a considerable volume of data (Karimi-Mamaghan et al., 2022), which has been discarded in the literature. Various machine learning techniques, ranging from classification to reinforcement learning, have been applied to build learning models for automated algorithm design. In the context of automated algorithm design, the learning

process can be online or offline based on data availability. Online learning continuously learns from newly arriving data gathered during solving the optimisation problem at hand to update the learned knowledge in algorithm design. Offline learning learns from the complete dataset which is available before the problem-solving begins, without incremental updates in the algorithm design process.

In (Qu et al., 2020), a new model named General COP (GCOP) formally defines the problem of algorithm design itself as a COP. The decision variables of GCOP consist of elementary algorithmic components, thus the design of various algorithms can be defined as flexible compositions of basic components. The objective of GCOP is to optimise the composition of these components. Solving the GCOP thus automates the design of the best algorithms for solving the problem at hand. GCOP provides a standard to support automated algorithm design (Qu et al., 2020) by formulating various search algorithms in one model.

Recent advances in automated algorithm design have evidenced that automatically designed algorithms might outperform some manually designed algorithms. This indicates that automated algorithm design can greatly benefit from machine learning due to the generalisation and predictive power of machine learning. However, there are many types of machine learning models, each with its own assumptions, algorithms, and performance characteristics. An essential question is how machine learning algorithms can be leveraged to learn from the large amount of complex data generated in the search process. In other words, how machine learning can be applied effectively in learning from the search data to produce effective algorithm designs. This question is the primary motivation behind the work in this thesis.

In light of recent developments, it is now becoming more evident that useful information is hidden in the data generated during the search. These new findings encourage further analysis of the data of effective algorithm designs for producing new effective algorithms to the literature (Qu et al., 2020). Machine learning can capture both explicit and implicit knowledge from the data. Explicit knowledge is directly observable or explicitly described, such as rules and patterns. Implicit knowledge refers to information that is not explicitly stated or easily represented, which can be embedded in learning models such as neural networks. However, most of the research in the field of automated algorithm design focuses on building learning models. The knowledge from effective algorithm designs has been rarely studied.

In addition, with more machine learning models investigated in automated algorithm design, the rich and new knowledge acquired from the data is captured in the learning models. This hidden knowledge is, however, implicit to interpret. Few studies in automated algorithm design have focused on the interpretability of machine learning, which makes it difficult to understand why certain decisions are being made for producing effective algorithms. Enhancing the interpretability of learning is important for acquiring valuable insights into decision-making processes and identifying the key factors that contribute to the development of effective algorithms.

By proposing the most basic algorithmic components, GCOP defines a large space of algorithm designs. Most of the studies of automated algorithm design however focused on the design of specific heuristics or investigating hand-picked procedures, thus only a smaller number of algorithm designs are considered (Qu et al., 2020). In principle, larger algorithm design spaces can be expected to contain better-performing algorithms (Hoos, 2008). The new GCOP standard calls for further investigations of automated algorithm design (Qu et al., 2020), however, requires coherent frameworks to explore

the insights on designing effective algorithms with these components.

1.2 Research aim and objectives

This thesis gathers the most recent advances in machine learning and automated algorithm design to fill the aforementioned knowledge gap. The research aim of the thesis is to investigate how to leverage machine learning to automatically design effective local search algorithms based on the new GCOP standard, thus enhancing the understanding of algorithm design and introducing new effective algorithms to the literature. In the context of three lines of research in automated algorithm design, this thesis focuses on automated composition.

To achieve this aim, the following research questions (RQs) are formulated to support the investigations in the thesis:

- RQ1: How to conduct investigations based on the GCOP model?
- RQ2: How to use machine learning in automated algorithm design?

To answer RQ1, two research objectives (ROs) have been identified:

- RO1: to develop a coherent framework to support the automated algorithm design based on the general GCOP model. This objective is to support the automated design of local search algorithms and systematic investigations on effective algorithm designs.
- RO2: to examine the performance of the elementary algorithmic components of the GCOP model in automated algorithm design. This objective is to justify the effectiveness of the basic GCOP components in automated algorithm design.

The outcomes of RO1 and RO2 set the base for the subsequent research for answering RQ2. RQ2 can be divided into two sub-questions:

- RQ2.a: What to learn? This question emphasises the knowledge that can be learned by machine learning in the algorithm design process.
- RQ2.b: How to learn? This question focuses on various learning methods for automated algorithm design.

To address RQ2, it is crucial to align the analysis of investigations with the specific learning tasks at hand. The definition of machine learning tasks determines the learning target, the type of knowledge that can be acquired from the data, and the choice of learning methods. To answer RQ2, several ROs have been identified with a focus on different perspectives of learning, including the form of knowledge (i.e., explicit, implicit), the way of extracting knowledge (i.e., online learning, offline learning), and the way of using knowledge (i.e., different ways of modelling algorithm design tasks into machine learning tasks).

- RO3: to investigate different online learning behaviour. This objective is to identify effective learning behaviour that can be used to design effective learning models for automated algorithm design. Effective algorithm designs can be produced and collected for subsequent investigations of useful knowledge to support algorithm design.
- RO4: to learn implicit knowledge within the data of effective algorithm designs with offline learning. This objective is to build machine learning models which can capture useful knowledge from the data to support automated algorithm design.
- RO5: to learn explicit knowledge within the data of effective algorithm designs with offline learning. This objective focuses on the

interpretability of learning to gain insights into the decision-making involved in the design of effective algorithms.

A good understanding of the data is essential for clearly defining the machine learning task. Based on the defined GCOP model, an algorithm can be seen as a composition of algorithmic components (Meng and Qu, 2021), which suggests an algorithmic composition can be represented by a sequence of algorithmic components. In automated composition, the decision-making regarding algorithmic components may vary over time. Thus, a hypothesis can be formed regarding the sequential and temporal features of the algorithmic compositions. The investigations for answering RQ2 are mainly based on this hypothesis.

Based on the above ROs, this thesis investigates automated algorithm design using Vehicle Routing Problems with Time Windows (VRPTW) as the domain example. As a widely investigated optimisation problem in operational research (Wong, 1983), the basic vehicle routing problem (VRP) (Fisher and Fisher, 1995) consists of ordering and assigning customer delivery demands to a set of vehicles. The objective is to minimise the total travel costs serving all the customers. Variants of VRP have been investigated with complex constraints (Braekers et al., 2016) to address different real-world scenarios. In the most widely studied VRPTW variant, customers must be served within specified time intervals (Cordeau et al., 2007). The design of effective heuristic algorithms to solve VRPTW remains a challenge for researchers, providing an ideal testbed to investigate of machine learning in automated algorithm design.

1.3 Main works and contributions

In the thesis, a series of studies have been conducted based on the identified RQs and ROs. The main contributions of the thesis can be summarised as follows:

1. This thesis presents an extended taxonomy of automated algorithm design, along with a categorisation of the decisions involved in the algorithm design process. The proposed categorisation and taxonomy contribute to capturing recent advances in automated algorithm design, thus establishing a more structured way of analysing the existing literature.

2. This thesis presents a new general AutoGCOP framework to support the automated composition of elementary algorithmic components, thus supporting the automated design of local search algorithms.

3. Within the consistent AutoGCOP framework, this thesis confirms the satisfying performance of the elementary algorithmic components for the VRPTW. This serves as the baseline to conduct further investigations on effective algorithm compositions.

4. Within AutoGCOP, the thesis evaluates reinforcement learning and Markov chain as a means of online learning to compose algorithmic components, investigating the effectiveness of learning on the individual performance of algorithmic components and the transition performance of algorithmic components. Results within the general AutoGCOP framework confirm the superior performance of the Markov chain model (which observes transition performance) in the automated design of new local search algorithms, thus suggesting the benefits of learning the transitions between algorithmic components.

5. Within AutoGCOP, the thesis investigates the implicit knowledge with machine learning models which can be acquired from effective algorithmic compositions offline.

   (a) Firstly, the prediction of algorithmic components in automated algorithm compositions is formally defined as a sequence prediction task for machine learning, supported by the underlying GCOP model theoretically. With the collected data upon the basic GCOP components, the newly defined machine learning task introduces new challenges to the machine learning community and encourages cross-disciplinary collaborations between evolutionary computation and machine learning.

   (b) Secondly, the thesis confirms the superior performance of Long Short-term Memory (LSTM) in the defined new machine learning task on automated algorithm design. To the best of our knowledge, it is the first attempt to propose an LSTM model in the automated design of search algorithms.

   (c) Thirdly, the analysis of different features of the collected data confirms the effectiveness and contributions of problem instance features and search stage in algorithmic compositions. These identified two types of features offer new insights and inform further effective algorithm design.

6. Within AutoGCOP, the thesis investigates explicit knowledge in effective algorithm designs with sequential rule mining.

   (a) With the collected data on the basic GCOP components, the thesis investigates the sequential rules of composing basic components. To the best of our knowledge, it is the first research work on sequential rule mining, a classical mining technique,

in learning from effective algorithm designs towards automated algorithm design.

(b) Secondly, this thesis confirms the satisfying effectiveness and generality of the sequential rules on basic components in automated algorithm design.

(c) The investigation of the sequential rules suggests a novel way of grouping the basic operators based on their impact on optimisation objectives, which was previously not considered in the literature. The analysis of operator groups with GCOP methods confirms their effectiveness in the automated composition of groups of operators. Furthermore, grouping operators effectively reduces the search space of algorithm design. This allows automated algorithm design to focus on a subset of algorithm designs in the search space. The automated design process could find good algorithms more quickly if the reduced search space is promising. However, there is a possibility of neglecting certain regions of the search space that may contain the potentially better or the best algorithms.

1.4 Thesis outlines

This section introduced the structure of the thesis to offer a clear roadmap of what lies ahead. Figure 1.1 presents an overview of the main chapters and their correlations to the identified RQs and RQs.

First, Chapter 2 "Related works" introduces the recent advances in related research areas to provide the context for the studies of the thesis. The ROs of this thesis are further explained based on the related research

background.

Among the proposed ROs, this thesis first focuses on RO1 and RO2 by building a coherent algorithm design framework and assessing the performance of the basic GCOP components. In Chapter 3, a general AutoGCOP framework is built to support the automatic composition of elementary algorithmic components based on the general GCOP model, thus designing local search algorithms automatically. Based on the AutoGCOP framework, this chapter analyses the performance of the most basic algorithmic components and justifies their effectiveness in designing search algorithms capable of solving VRPTW.

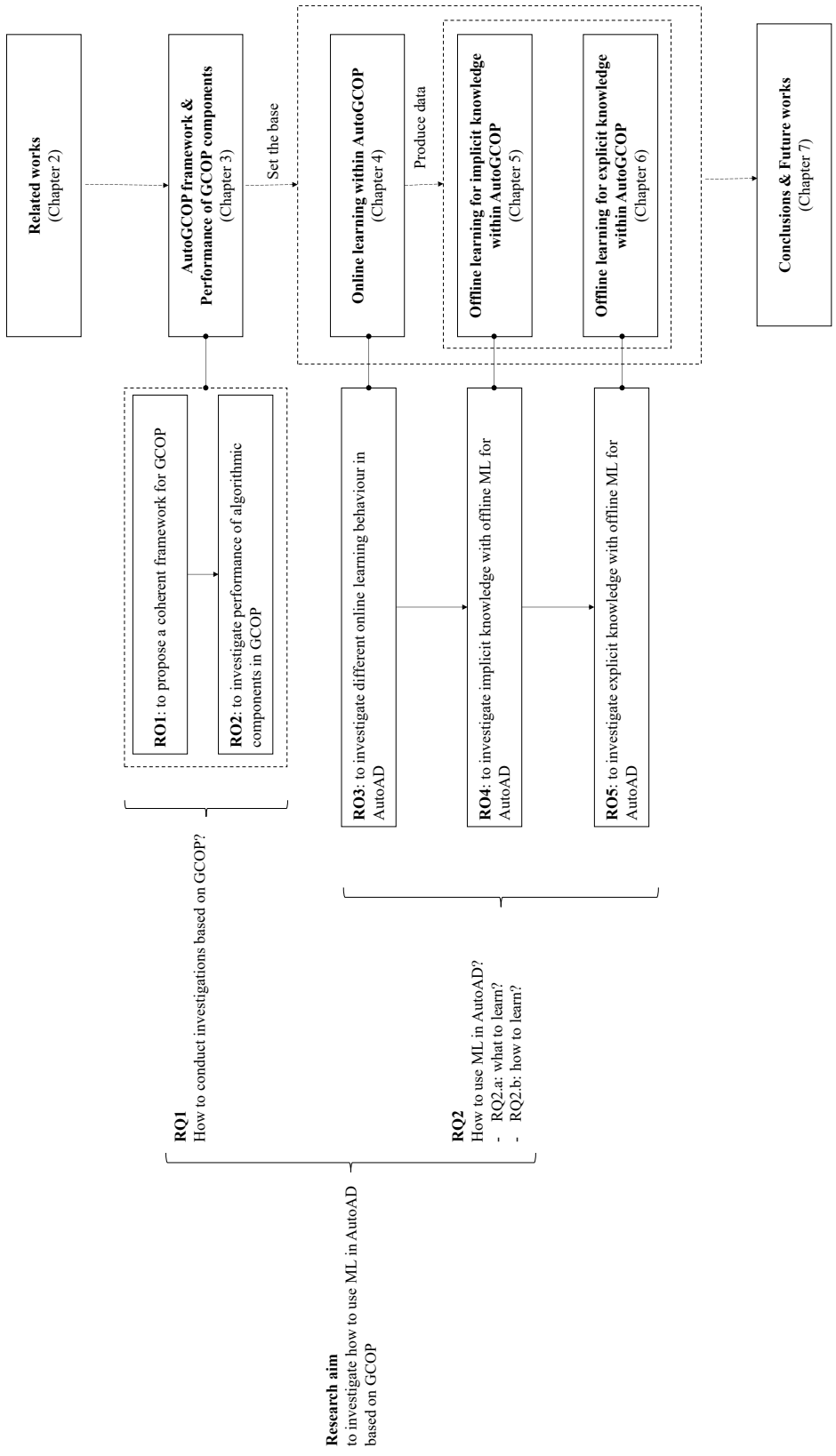


Figure 1.1: Chapter structure and correlations with research questions and objectives of the thesis. ML: machine learning. The abbreviations used in the figure are as follows: ML (Machine Learning), AutoAD (Automated Algorithm Design), RQ (Research Question), and RO (Research Objective).

Chapter 3 can be seen as the base for the development of automated composition methods and the systematic analysis of effective algorithmic compositions in the following chapters. Based on the proposed AutoGCOP framework in Chapter 3 and the ROs for answering RQ2, Chapter 4, Chapter 5, and Chapter 6 investigate learning in automated compositions with different perspectives in the form of knowledge (i.e., explicit, implicit), the way of extracting knowledge (i.e., online learning, offline learning), and the way of using knowledge (i.e., different ways of modelling algorithm design tasks into machine learning tasks). The learning targets in these chapters mainly focus on the knowledge of the sequential relations between elementary algorithmic components in GCOP towards automated algorithm design.

Based on RO3, Chapter 4 investigates online learning to compose algorithmic components within AutoGCOP. Reinforcement learning and Markov chain are evaluated as a means of learning the individual performance of algorithmic components and learning the transition performance of algorithmic components.

In Chapter 5, the prediction of algorithmic components is formally defined as a sequence prediction task for machine learning for RO4. This chapter evaluates LSTM and Transformers as a means of learning from the collected data on the basic GCOP components, predicting the operators for automated composition. Different types of information in the search process are investigated as features to enhance the prediction.

Based on RO5, Chapter 6 focuses on investigating explicit knowledge in the form of rules between algorithmic components. The collected data on the basic GCOP components are analysed with sequential rule mining. Experimental studies evaluate the effectiveness and generality of the sequential

rules of basic components in automated algorithm design.

Finally, Chapter 7 concludes this thesis, discusses the obtained results, and proposes promising research directions, especially on (1) generalisation of the proposed framework to other COPs, (2) extension of the proposed framework to more general search frameworks, (3) and investigations on other categories of algorithmic components.

1.5 List of publications

The following publications have arisen during the study for this thesis and are related to the work herein. They are listed below in chronological order.

1. Meng, W. and Qu, R. (2021). Automated design of search algorithms: Learning on algorithmic components. *Expert Systems with Applications*, 185:115493.

   The content of this paper is covered in Chapter 3 and Chapter 4.

2. Meng, W. and Qu, R. (2023a). Automated design of local search algorithms: Predicting algorithmic components with lstm. *Expert Systems with Applications*, page 121431.

   The content of this paper is covered in Chapter 5.

3. Meng, W. and Qu, R. (2023b). Sequential rule mining for automated design of meta-heuristics. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 1727–1735.

   The content of this paper is covered in Chapter 6.

Chapter 2

Related Works

Contents

This chapter provides an overview of the background and techniques used in this thesis. Section 2.1 introduces the definitions of the classic Vehicle Routing Problems (VRPs), followed by a more detailed mathematical formulation of the VRP with Time Windows (VRPTW) which is the case study of this thesis. Section 2.2 provides an overview of the existing heuristic algorithms for VRPs, with a particular focus on meta-heuristics which have shown powerful performance in solving real-life large-scale problems. Section 2.3 presents a comprehensive review of automated algorithm design, with a specific focus on machine learning techniques that have been used to automate the search algorithm design process. Section 2.4 introduces the background of machine learning techniques used in this study. Finally, Section 2.5 provides a summary of the previous studies discussed in this chapter.

2.1 The vehicle routing problem

The Vehicle Routing Problem (VRP) is a well-studied combinatorial optimisation problem in operational research (Wong, 1983). It was first introduced as the truck dispatching problem in (Dantzig and Ramser, 1959) for determining the most efficient routes for a fleet of vehicles to visit a set of locations with specific demands while minimising the overall travel distance or cost. The problem was later extended in (Clarke and Wright, 1964) to include trucks with varying capacities that need to service several delivery depots from and back to a central depot. Since then, variants of VRPs have been developed serving different needs of a variety of transportation and logistics scenarios (Braekers et al., 2016).

The basic VRP concerns routing for a fleet of vehicles to serve a number

of customers with delivery demand from and back to a depot (Fisher and Fisher, 1995). The most commonly used objectives in VRP research are minimising the total travel distance (TD) and the number of vehicles required (NV) (Osman, 1993). Numerous VRP variants have been proposed, reflecting practical constraints and objectives specific to different real-world scenarios (Eksioglu et al., 2009). The Capacitated VRP (CVRP) and the VRP with Time Window constraints (VRPTW) are two variants that received the most attention from the research community (Toth and Vigo, 2002). In CVRP (Toth and Vigo, 2002), each vehicle has a certain demand and should be satisfied when serving each customer's demand, concerning the issue that delivery vehicles have limited capacity. VRPTW concerns the constraint that customers must be served within specified time intervals (Cordeau et al., 2007). Recent surveys on VRPs and their variants can be found in (Eksioglu et al., 2009), (Kumar and Panneerselvam, 2012), (Braekers et al., 2016) and (Tan and Yeh, 2021).

2.1.1 Vehicle routing problem with time windows

VRPTW is one of the most studied VRP variants since it represents a series of common applications in the real world. The VRPTW can be modelled as follows (Chen et al., 2016). A routing network is represented by a graph $G = (V, E)$ where $V = \{v_0, v_1, ..., v_n\}$ represents a depot (v_0) and n customers $(v_1, ..., v_n)$. Each v_i is associated with a non-negative demand q_i and service time s_i, while $q_0 = 0$ and $s_o = 0$. Each edge $e \in E = \{(v_i, v_j) : v_i, v_j \in V, v_i \neq v_j\}$ represents the link between customers v_i and v_j associating with a travel distance d_{ij}.

A fleet of K vehicles is used to serve all customer demands. To customer v_i, the service start time b_i must be within the time window $[e_i, f_i]$, where e_i

and f_i denote the earliest and latest time to serve q_i. If a vehicle arrives at v_i at time $a_i < e_i$, a waiting time $w_i = max\{0, e_i - a_i\}$ required. Consequently, the service start time $b_i = max\{e_i, a_i\}$. Each vehicle of a capacity Q travels on a route connecting a subset of customers starting from v_0 and ending within the scheduling horizon $[e_0, f_0]$.

Decision variables:

The decision variable $X_{ij}^k = 1$ if the edge from v_i to v_j is assigned in the route of vehicle k ($k \in K$); otherwise $X_{ij}^k = 0$.

Objective function:

$$\text{Minimise} \quad K \tag{2.1}$$

$$\text{Minimise} \quad \sum_{k \in K} \sum_{v_i \in V} \sum_{v_j \in V} X_{ij}^k \cdot d_{ij} \tag{2.2}$$

Constraint:

$$\sum_{k \in K} \sum_{v_i \in V} X_{ij}^k = 1, \forall v_i \in V \setminus \{v_o\} \tag{2.3}$$

$$\sum_{k \in K} \sum_{v_j \in V} X_{ij}^k = 1, \forall v_j \in V \setminus \{v_o\} \tag{2.4}$$

$$\sum_{k \in K} \sum_{v_i \in V} \sum_{v_j \in \setminus \{v_0\}} X_{ij}^k = n \tag{2.5}$$

$$\sum_{v_j \in V} X_{oj}^k = 1, \forall k \in K \tag{2.6}$$

$$\sum_{v_i \in V} X_{ij}^k - \sum_{v_j \in V} X_{ji}^k = 0, \forall k \in K, v_j \in V \setminus \{v_o\} \tag{2.7}$$

$$\sum_{v_i} X_{io}^k = 1, \forall k \in K \tag{2.8}$$

$$e_i \leq b_i \leq f_i, \forall v_i \in V \tag{2.9}$$

$$\sum_{v_i \in V} \sum_{v_j \in V} X_{ij}^k \cdot q_i \leq Q, \forall k \in K \tag{2.10}$$

$$X_{ij}^{k} \in \{0,1\}, \forall v_i, v_j \in V, k \in K \tag{2.11}$$

Objective (2.1) is to minimise the number of vehicles, while objective (2.2) is to minimise the total travel distance. Constraints (2.3-2.5) limit every customer to be visited exactly once and all customers are served. Constraints (2.6-2.8) define the route of vehicle k. Constraints (2.9) and (2.10) guarantee feasibility in terms of the time window constraints and vehicle capacity constraints, respectively. Constraint (2.11) defines the domain of the decision variable X_{ij}^{k}.

VRPTW is a more practical variant of the CVRP as it introduces the time window constraints that customers must be served within specified time intervals (Cordeau et al., 2007). However, this additional constraint makes VRPTW more complex than CVRP as it introduces an additional dimension of time during routing. The design of effective search algorithms for VRPTW can be a time-consuming and iterative process, requiring significant domain expertise. Therefore, the algorithm design for solving VRPTW remains a challenging problem for the research community.

The focus of this thesis is to investigate strategies for automated algorithm design for solving VRPTW. Specifically, this study considers the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD) by a weighted sum objective function used in the literature (Walker et al., 2012) as shown in Equation (2.12), where C denotes a sufficiently large number to ensure that K is always the primary objective. This thesis seeks to contribute to the development of automated algorithm design for effectively solving complex optimisation problems by solving the VRPTW with dual objectives.

$$f = \sum_{k \in K} \sum_{v_i \in V}^{k} \sum_{v_j \in V} X_{ij}^{k} d_{ij} + C \times K \tag{2.12}$$

2.1.2 Problem instances for VRPTW

The Solomon (Solomon, 1987) and Hombeger-Gehring (Homberger and Gehring, 1999) instances are widely tested in the VRPTW literature. Both data sets consist of six types of instances, i.e., C1, C2, R1, R2, RC1, and RC2. These instances vary with respect to the customer distribution type (i.e., random, clustered, and mixed), vehicle capacity (i.e., short and long), and the density and tightness of the time windows. The usage of these instances allows for a systematic comparison of the performance of different algorithms in solving the VRPTW with different problem characteristics.

Customer distribution types and scheduling horizons are the key problem instance features investigated in many studies. Type-C instances are characterised by customers being located in several clusters. Type-R instances have customers randomly distributed geographically, while Type-RC instances are a combination of both. Short scheduling horizons and lower vehicle capacities are typical for Type-1 instances, whereas Type-2 instances have longer scheduling horizons and larger vehicle capacities.

2.2 Search algorithms for Vehicle Routing Problems

Different search algorithms, including exact methods and approximation approaches, have been proposed to address VRPs (Laporte et al., 1992). Among various algorithms, exact methods can guarantee optimal solutions.

However, as VRP is an NP-hard problem (Lenstra and Kan, 1981), exact methods are often limited to small problem instances (Kumar and Panneerselvam, 2012). On the other hand, approximation approaches focus on finding an acceptable solution within a reasonable time frame, which are often more suitable for practical applications of VRPs (Braekers et al., 2016).

Approximation approaches for VRPs can be broadly classified into classical heuristics and meta-heuristics in the literature (Laporte et al., 2000), (Tan et al., 2001), (Lin et al., 2009). The heuristics developed mostly between 1960 and 1990 are known as classical heuristics, while meta-heuristics have occurred and developed in the last decade (Laporte et al., 2000).

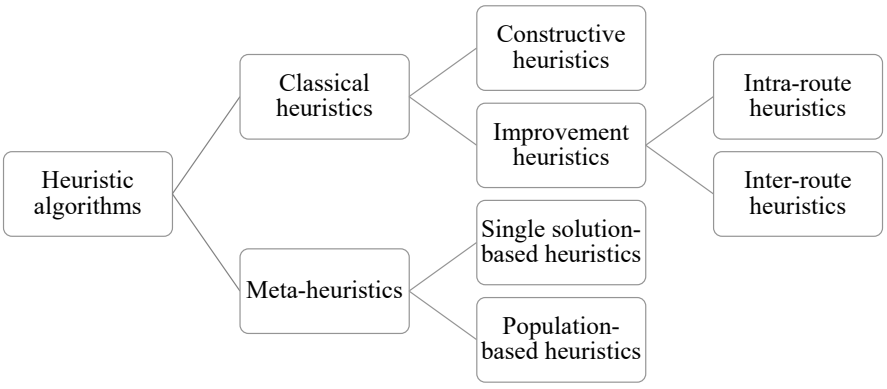
Classical heuristics for VRPs are usually problem-specific (Talbi, 2009) and explore a limited region of the search space to yield good solutions faster than an exact method. Construction heuristics and improvement heuristics are two main types of classical heuristics for constructing and improving routes, respectively (Laporte et al., 2000). Construction heuristics build a set of routes from zero following some fixed empirical heuristic procedures, while improvement heuristics tries to produce an improved solution on the basis of an incumbent solution (Liu et al., 2023). Classical heuristics have limited performance in solving large and complex VRPs (Kumar and Panneerselvam, 2012). In contrast, meta-heuristics can handle more complex problems by exploring a larger search space and using higher-level strategies (Talbi, 2009).

A meta-heuristic is an intelligent strategy combining classical heuristics to perform a deep exploration and exploitation of the promising area of the search space (Desale et al., 2015). Compared to classic heuristics, meta-heuristics conduct a more extensive exploration of the solution space

(Cordeau et al., 2002), thus have been extensively studied and proven to achieve superior performance for solving VRPs (Laporte et al., 2000), (Elshaer and Awad, 2020), and (Tan and Yeh, 2021). The optimisation performance by meta-heuristics is usually better than classical heuristics (Bräysy and Gendreau, 2005a). Many meta-heuristics have been designed to solve VRPTWs to near-optimal solutions (Tan et al., 2001).

Meta-heuristics can be classified based on various criteria (Talbi, 2009). Based on the preference for exploration versus exploitation during the search, meta-heuristics can be divided into two categories, i.e., single solution-based and population-based. Single solution-based meta-heuristics operate on a single solution during the search, while population-based meta-heuristics manipulate a whole population of solutions. In general, basic single solution-based meta-heuristics are designed to intensify the search in local regions, thus more exploitation-oriented, whereas basic population-based meta-heuristics allow for more exploration of the whole search space (Talbi, 2009).

Figure 2.1: A classification of vehicle routing heuristics (Laporte et al., 2000), (Liu et al., 2023).



There are numerous studies that provide systematic reviews of well-established research fields, from classical heuristics to meta-heuristics (Laporte et al.,

2000), (Elshaer and Awad, 2020), (Tan and Yeh, 2021), and (Liu et al., 2023). This section aims to provide a contextual understanding of various heuristic methods for addressing VRPs, thus providing an overview of representative studies in the VRP literature grouped based on the classification as shown in Figure 2.1. The focus of this thesis is to investigate the automated design of single solution-based meta-heuristics for solving the VRPTW. Therefore, the following overview emphasises the recent advances in this line of heuristic methods and highlights important algorithmic components for automated algorithm design.

2.2.1 Classical heuristics

The VRP literature is extensive and includes several review works and book chapters that discuss classical heuristics for solving VRPTW (Tan et al., 2001), (Bräysy and Gendreau, 2005a), (El-Sherbeny, 2010) and many other variants of VRPs (Laporte et al., 1992), (Laporte et al., 2000), (Cordeau et al., 2005), (Laporte et al., 2014), (Liu et al., 2023). Classical heuristics for solving VRPs can generally be classified as constructive heuristics and improvement heuristics (Bräysy and Gendreau, 2005a). Moreover, these standard classical heuristics are important components of meta-heuristics (Bräysy and Gendreau, 2005a) for constructing routes in VRP and further improving the generated solutions. This subsection presents an overview of the widely used classical heuristics in the VRP literature, to provide a contextual understanding of the important algorithmic components of meta-heuristics.

Constructive heuristics

Constructive heuristics for VRPs are used to construct routes of a solution by systematically adding customers into the partial routes to minimise cost iteratively. The most commonly used constructive heuristics for VRPs in the literature (Laporte et al., 1992), (Laporte et al., 2000), (Cordeau et al., 2005) are listed as follows:

- Saving algorithm (Clarke and Wright, 1964): it starts with an initial solution and uses a saving formula to evaluate potential cost savings by merging short routes into longer routes.
- Insertion algorithms: unrouted customers are inserted sequentially (Mole and Jameson, 1976) or in parallel (Christofides, 1979) to the solution until all customers are visited. When there is no feasible route to accept the customer, a new route would be created.
- Sweep algorithm (Gillett and Miller, 1974): it sets the depot as the origin and divides the customers into angular clusters by rotating a ray through customers, then process the customers in each cluster to build routes that return to the depot.
- Cluster-first, route-second algorithm (Fisher and Jaikumar, 1981): it separates the customers into clusters by modelling the clustering step as a Generalised Assignment Problem (GAP), such that the feasible cluster is generated by solving the GAP for the minimal cost, then construct routes for each cluster.

Extensions of saving algorithms mainly focus on modifying the saving formula and the selection of route pairs. These variations can be considered as different instances of saving algorithms that employ distinct parameters

in the saving formula and various strategies to choose route pairs. A general form of the saving formula was proposed in (Gaskell, 1967), (Yellow, 1970), which parameterised the original formula in (Clarke and Wright, 1964) to adjust the relative importance of different factors. Some studies proposed more sophisticated saving formulas that consider multiple criteria (Caccetta et al., 2013), (Cengiz Toklu, 2022). In terms of the selection of route pairs, different strategies have been proposed, such as greedy selection which selects the pairs with the largest savings from the saving list, the matching method (Altinkemer and Gavish, 1991), (Wark and Holt, 1994) and the probabilistic function (Juan et al., 2010).

Extensions to insertion algorithms mainly focus on the customer selection criteria and the route insertion criteria. The customer selection criteria determine which new customers to insert, while the route insertion criteria determine where to insert them. The most basic greedy insertion algorithm selects the unrouted customers and the insertion locations with the minimum insertion cost. The regret criterion, proposed in (Potvin and Rousseau, 1993), measures the difference between the cost of the best position and the near-best positions as regret for insertion. Another example criterion, proposed in (Ioannou et al., 2001), takes into account the impact on both routed and unrouted customers in route insertion.

In the VRP literature, the sweep algorithm (Gillett and Miller, 1974) and the cluster-first, route-second algorithm (Fisher and Jaikumar, 1981), are generally grouped as two-phase methods (Chen, 2018). These methods decompose the VRP solution process into clustering and routing, thus can be further categorised into two groups depending on whether clustering first or routing first. Cluster-first algorithms, such as the classical sweep algorithm (Gillett and Miller, 1974) and the cluster-first, route-second algorithm (Fisher and Jaikumar, 1981), first partition customers into different

clusters and then route customers in each cluster. Route-first algorithms, such as the route-first cluster-second algorithm proposed in (Beasley, 1983), construct a giant tour with all customers first, and then split it into many feasible routes. Extension to two-phase algorithms mainly focuses on improving the clustering and customer selection criteria, such as the use of the polar-coordinate angle (Gillett and Miller, 1974), (Na et al., 2011) and distance metrics (Peya et al., 2019).

Improvement heuristics

Improvement heuristics are methods used to improve an initial solution by making specific modifications to the incumbent solution. The main concept behind most improvement heuristics is the notion of a neighbourhood (Laporte et al., 2000), (Funke et al., 2005). The neighbourhood of a solution $s \in S$ in the solution space S is a set of solutions $N(s) \subseteq S$ that can be generated with a single modification of s. The process of moving from s to another solution $s' \in N(s)$ is called a neighbourhood move (Funke et al., 2005). In the context of meta-heuristics, an operator is a recipe to modify a solution to obtain another solution (Rousseau et al., 2002). An operator defines a neighbourhood structure $N(s)$ that can be reached by applying that operator on s. Therefore, the design of an operator is important in designing effective improvement heuristics.

Local search algorithms are widely used improvement heuristics in the VRP literature, which apply operators to the current solution to find potentially better solutions with respect to the objective function (El-Sherbeny, 2010). The local search algorithm uses operators to conduct modification operations on VRP solutions, defining the local searching area. The algorithm searches for candidate solutions in this neighbourhood iteratively. In each

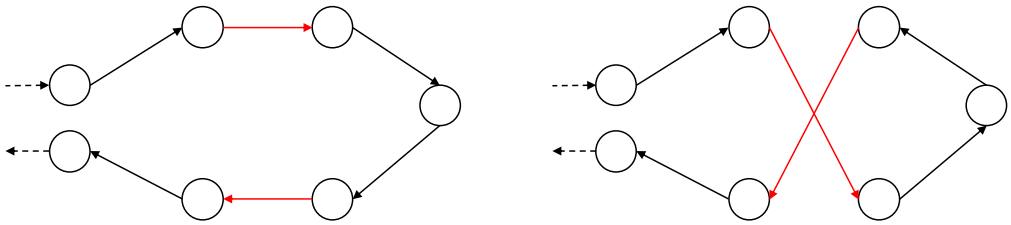
iteration, the quality of the current solution and that of the candidate solution are compared to determine a new search point. Two common strategies for determining the new search point are the first-improving search and best-improving methods. The former selects the first improvement found in the neighbourhood, while the latter identifies the most significant improvement among all neighbour solutions (Funke et al., 2005). Once no better solution can be found in the neighbourhood, the search process terminates, indicating that an optimum solution has been achieved which may only be a local optimum or even a global optimum (El-Sherbeny, 2010). To prevent exhaustive search, the d-best search is a commonly used method to halt the search process once d-improving neighbour solutions have been discovered (Funke et al., 2005).

The local search heuristics for solving VRPs mainly rely on a set of essential operators that apply modification operations on each vehicle route individually (known as intra-route heuristics), or on multiple routes simultaneously (known as inter-route heuristics) (Laporte et al., 2000). The most commonly used intra-route heuristics are listed as follows:

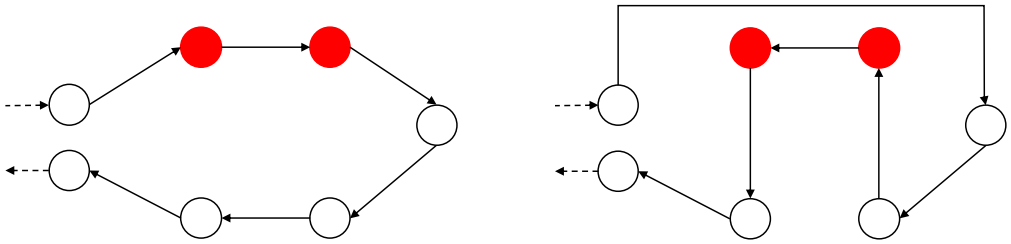
- λ-opt (Lin, 1965): this operator conducts the edge-exchange operation within one route, i.e., λ edges are removed from a route and the λ remaining segments are reconnected in all possible ways. In the literature, λ has been studied at most 3 in practical problems, i.e., 2-opt and 3-opt.

- Or-opt: (Or, 1976): this operator can be seen as a special case of λ-opt. Instead of edge-exchange, Or-opt is a node-exchange operation, displacing consecutive (no more than three) nodes of one route to another location within the same route.

Figure 2.2: Illustration of intra-route heuristics. The nodes are customers and the lines are routes. The red lines and red nodes are the segments of the routes removed/added by the improvement heuristics.



(a) 2-opt



(b) Or-opt

The operators of the most commonly used inter-route heuristics are summarised as follows:

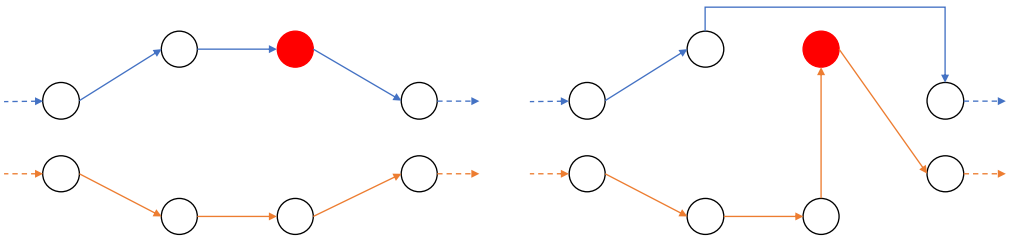
- Relocate operator (El-Sherbeny, 2010): also known as shift operator (Pillay et al., 2018a). It moves one customer from one route to another.
- GENI-exchange (Gendreau et al., 1992): this can be seen as a special case of the relocate operator. The operation is to move one customer from one route to another and the customer will be positioned between the two customers which are closest to it.
- Swap operator: replace one customer from one route with the customer from another route.
- λ-interchange (Osman, 1993): it can be seen as a generalised operation of the basic swap operator. The operator replaces no more than λ customer of a route with the customer from another route (not

more than λ as well) to generate a new solution. The customer from a route can be either consecutive or non-consecutive.

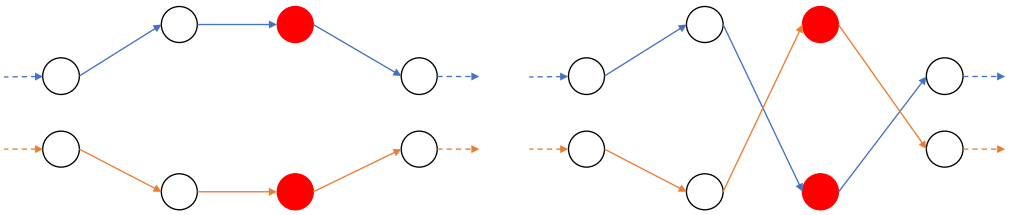
- *CROSS*-exchange (Taillard et al., 1997): this operator can be seen as a different generalisation of the basic swap operator, which swaps two strings of consecutive customers from different routes.

- 2-opt* (Potvin and Rousseau, 1995): A special case of *CROSS*-exchange. It removes two edges from two independent routes and reconnects the routes with no inversion of the orientation of route segments. 2-opt* is a subset of the 2-opt neighbourhood for inter-route operation.

Figure 2.3: Illustration of inter-route heuristics. The nodes are customers and the lines in different colours are different routes. The red nodes are the customers of the routes changed by the improvement heuristics.



(a) Relocate



(b) Swap

To illustrate the basic operations in improvement heuristics, Figure 2.2 shows 2-opt and Or-opt operators, and Figure 2.3 describes relocate operator and swap operator. In the literature, extensions to the commonly used improvement heuristics mainly focus on modifying the number of

edges/vertices to exchange/relocate, as well as the selection policies and the insertion policies of these operations. Some operators, such as λ-opt (Lin, 1965) and λ-interchange (Osman, 1993), can be seen as the parameterised versions of basic operations. Other operators, such as 2-opt* (Potvin and Rousseau, 1995), can be seen as special cases of the parameterised operations that use specific settings. For example, the shift operator in (Schulze and Fahle, 1999) can be seen as the relocate operator with a specific insertion criterion that evaluates all insertion positions before inserting a customer.

Remarks

The operations performed by classical VRP heuristics upon routing solutions are usually subject to the constraints in the VRPs (Pillay et al., 2018a). For solving VRPTW, the operations of constructive and improvement heuristics need to carefully consider the time window constraints (Bräysy and Gendreau, 2005a), particularly in the selection of edge or vertice and insertion locations. The study in (Potvin and Rousseau, 1995) identified that the λ-opt neighbourhood can cause time constraint violations by reversing the tour orientation. Operations without any inversion of the tour orientation, such as 2-opt* (Potvin and Rousseau, 1995), have been shown to improve the efficiency of local search in VRPTW (Chen, 2018).

Many studies have used effective constructive heuristics to provide a good solution before route improvement. The study in (Van Breedam, 1994) has shown that a good starting point is essential for designing effective improvement heuristics. This supports many studies that investigated the combination of constructive heuristics with improvement heuristics and

meta-heuristics for solving variants of VRPs.

The improvement heuristics play a crucial role in meta-heuristics for solving VRPs. These heuristics can be seen as being extended from the most basic operations to VRP solutions. The specific operators can be seen as compound operators of the most basic neighbourhood moves and can explore a limited search space of the basic operators (Meng and Qu, 2021). The operators used in VRP algorithms can be modelled as a collection of basic algorithmic components, as demonstrated in (Qu et al., 2020). Modelling various search algorithms in one model provides a standard for the design of new heuristic algorithms.

2.2.2 Meta-heuristics

Meta-heuristics is a class of methodologies that use high-level strategies within a heuristic search framework to navigate the search in the solution space of optimisation problems (Boussaïd et al., 2013). Due to the complexity of VRPs and their real-life applications, meta-heuristics have been extensively studied for practical applications (Elshaer and Awad, 2020), especially for solving VRPTW (Bräysy and Gendreau, 2005b). Several taxonomic reviews of meta-heuristics in the VRP literature have been conducted in recent years, including those by (Laporte et al., 2000), (Gendreau et al., 2008), (El-Sherbeny, 2010), (Elshaer and Awad, 2020).

Based on the population management strategy, meta-heuristics are classified as single solution-based methods and population-based methods (Blum and Roli, 2003). Single solution-based meta-heuristics tend to focus more on the exploitation of solution space, while population-based meta-heuristics tend to emphasise exploration (Boussaïd et al., 2013). This section presents

an overview of the representative studies of meta-heuristics in the VRP literature (Elshaer and Awad, 2020), to provide the context for the automated design of local search-based algorithms in the case study of VRPTW.

Population-based meta-heuristics

Population-based meta-heuristics operate on a population of solutions to explore the search space. The commonly studied population-based meta-heuristics generally include two types, i.e., Evolutionary Computation (EC) and Swarm Intelligence (SI) (Boussaïd et al., 2013). These algorithms are both inspired by natural phenomena but in different ways (Boussaïd et al., 2013). EC algorithms are based on the principles of natural selection, while SI is inspired by the collective behaviour of social organisms (such as ants, bees, etc) (Boussaïd et al., 2013). Both types of algorithms achieve high diversification while exploring the search space by evolving a population of solutions towards better regions of the search space (Beheshti and Shamsuddin, 2013).

EC algorithms (also known as Evolutionary Algorithms (EAs) (Boussaïd et al., 2013)) start with an initial population of solutions. Each iteration of the algorithm represents a generation, where the population of candidate solutions are modified by mutation and crossover operators to generate new offspring solutions. These offspring solutions are evaluated and the better ones have a greater chance of being selected to create the next generation. This process is repeated until the termination criterion is met.

SI algorithms involve the coordination and cooperation of a group of agents (or particles) to solve a problem. The group of agents in SI move through the search space, and their positions and movements are influenced by the positions of other agents in the swarm. The agents communicate with each

other and adjust their behaviour based on the information they receive, leading to emergent behaviour that can help to guide the algorithm towards better solutions over successive generations.

The widely used population-based meta-heuristics in the VRP literature (Elshaer and Awad, 2020) are reviewed in this section, including Genetic Algorithms (GA) and Memetic algorithms (MA) from the EC group, and Ant Colony algorithms (ACO) and Particle Swarm Optimisation (PSO) from the SI group.

Genetic Algorithms

Genetic Algorithms (GAs) (Holland, 1975) are the most well-known evolutionary algorithms (Back, 1996). In the original GA (Holland, 1975), a candidate solution is represented by a "chromosome" that consists of "genes" (e.g., bits). In each generation, two parent solutions are selected to generate two offspring by crossover and mutation operations. The crossover operation (De Jong and Spears, 1991) exchanges a part of the parent solutions, while the mutation operation applies classic move operators to change some locations in the chromosome. The new offspring are then evaluated and compared with the solutions in the population. A replacement strategy is applied to selectively replace individuals in the current generation with offspring to produce the next generation.

Applications of GAs have been reported for several applications to VRPs, including VRPTW (Thangiah et al., 1991), (Blanton Jr and Wainwright, 1993), (Thangiah, 1993), (Potvin and Bengio, 1996), (Chen et al., 2001), (Baker and Ayechew, 2003), VRP with backhauls (Potvin et al., 1996), multi-depot VRP (Ho et al., 2008) and dynamic VRP (Prins, 2004). In variants of GA, solutions may be encoded as different types and combined with different operators and solution selection mechanisms.

Classical VRP heuristics, such as the sweep algorithms (Baker and Ayechew, 2003), the saving algorithm (Ho et al., 2008) and λ-interchange (Ghoseiri and Ghannadpour, 2010), have been used in GAs for generating the initial solution population. Local search algorithms have been incorporated into GAs as mutation operators in (Chen et al., 2001) and (Prins, 2004). GAs show powerful performance in handling multiple complicated objectives, especially for solving multi-objective VRPs (MOVRPs). As one of the most studied VRP variants, the VRPTW concerns two objectives, i.e., minimising the number of vehicles and the total travel distance (Bräysy and Gendreau, 2001). The study in (Berger et al., 2003) proposed to use two populations of solutions respectively targeting distinct objectives for VRPTW in GA. More recently, (Vidal et al., 2013) proposed a new mechanism for population management and a new evaluation method of penalising infeasibility. The proposed GA outperforms the state-of-the-art algorithms in many VRPTW instances.

(2) Memetic algorithms

Memetic algorithms (MA), also known as Hybrid Evolutionary Algorithms (Hybrid EAs)(Gendreau et al., 2008), combine local search algorithms with population-based algorithms to enhance the exploitation of the search space (Moscato et al., 1989). MAs have been successfully applied to various problem domains by leveraging the strengths of different search approaches (Gendreau et al., 2008).

In the VRP literature, MAs usually combine GAs with local search heuristics (Liu et al., 2023), such as the combination of GA and classical improvement heuristics in (Cattaruzza et al., 2014), (Mendoza et al., 2010), and (Nagata et al., 2010). Various MAs have been developed to solve different VRP variants, including CVRP (Ngueveu et al., 2010), VRPTW

(Cattaruzza et al., 2014), (Qi et al., 2015), VRP with stochastic demands (Mendoza et al., 2010), and multi-trip VRP(Cattaruzza et al., 2014).

(3) Ant Colony algorithms

Ant Colony Optimisation algorithms (ACO) (Dorigo and Di Caro, 1999) are inspired by the behaviour of real ant colonies which use pheromone trails to communicate and navigate for finding food (Blum, 2005). In the ACO algorithms, an artificial ant is a simple computational agent which can explore the search space for good solutions. An ACO algorithm starts with a set of candidate solutions to the optimisation problem and a group of artificial ants. The ants then construct solutions by selecting one of the available options at each decision point, guided by the strength of the pheromone trail left by previous ants. Updating the pheromone trail is a key element of ACO (Bell and McMullen, 2004). After constructing a solution, the ants update the pheromone trail based on the quality of the solution found.

In the VRP literature, various ACO algorithms have been developed for variants of VRPs, such as CVRP (Bell and McMullen, 2004), (Mazzeo and Loiseau, 2004), (Doerner et al., 2005), large scale VRP (Reimann et al., 2004), open VRP (Li and Tian, 2006) and others. These algorithms incorporate different constructive heuristics (such as saving algorithms (Reimann et al., 2004)) and local search heuristics (such as swap, 2-opt (Bell and McMullen, 2004), (Reimann et al., 2004), interchange heuristics (Mazzeo and Loiseau, 2004)) for route construction and improvement. To tackle the multiple objectives of VRPTW, the study in (Gambardella et al., 1999) proposed to use two colonies to search for good solutions with respect to distinct objectives.

(4) Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) (Eberhart and Kennedy, 1995) follows the social behaviour of bird flocking. In PSO, each candidate solution is represented by a particle in the swarm, associated with a position and a velocity in the search space. The position of a particle represents the fitness value of the solution, and its velocity controls the direction and speed of its movement in the search space (Poli et al., 2007). During the search process, each particle adjusts its position and velocity based on its own experience and that of its neighbours in the swarm, including its current fitness value, the best position it found so far, and the best position the swarm found so far. Each particle tries to move towards the best solution it found so far and the best solution found by the swarm (i.e., the global best).

The application of PSO for solving VRPs starts from solving CVRP (Kachitvichyanukul et al., 2007). More PSO algorithms have been proposed by incorporating different route construction and improvement heuristics for dealing with different constraints of VRPs, such as the time windows (Zhao et al., 2004), stochastic demands (Marinakis et al., 2013), and simultaneous pickup and delivery (Ai and Kachitvichyanukul, 2009). The study in (Marinakis et al., 2013) combined local search-based meta-heuristics with PSO, which produces superior results compared to other EAs for solving VRPs.

Single Solution-based meta-heuristics

Single solution-based meta-heuristics are also known as local search-based meta-heuristics (Funke et al., 2005). These algorithms systematically explore the solution space by starting from an initial solution and moving at each step from the current solution to another promising solution in its neighbourhood (Cordeau et al., 2002), following a trajectory in the search

space (Boussaïd et al., 2013).

Single solution-based meta-heuristics are developed from the classical local search algorithms. The effectiveness of single solution-based meta-heuristics is due to the "intelligent" modifications of classic local search which strategically balance exploration and exploitation of the search space. Two key aspects of local search algorithms are the neighbourhood structures (i.e., operators) and the search technique for exploring the neighbourhood (Funke et al., 2005). In terms of the neighbourhood structure, smaller neighbourhood regions offer a more limited search and larger neighbourhood regions allow the exploration of a wider range of solutions. (Rousseau et al., 2002). However, larger neighbourhoods require more time to evaluate the larger set of solutions thoroughly. Acceptance criteria direct the search process to escape from local optimum and maintain a certain degree of exploration within the search space (Liu et al., 2023).

This section provides an overview of the most widely used single solution-based meta-heuristics in the VRP literature (Elshaer and Awad, 2020).The neighbourhood structures of those algorithms are mostly from the classic heuristics for solving VRPs. The major difference among these algorithms lies in how they guide the search process to jump out of the local optimum.

(1) Tabu search

Tabu Search (TS) (Glover, 1986) utilises a tabu list to forbid or penalise recently visited solutions. The tabu list acts as a short-term memory, recording the solutions explored in recent iterations. In each iteration, TS moves to the non-tabu positions to cycle moves and explore new solutions, thus moving away from the local optimum. To avoid over-restriction of the tabu list and potentially missing good solutions, aspiration criteria are

proposed to accept some tabu solutions under specific conditions. (Glover and Laguna, 1998).

In the recent VRP literature, TS stands out as the best (Cordeau et al., 2002) and the most used local search-based meta-heuristics (Elshaer and Awad, 2020). Various neighbourhood structures have been applied in the TS search framework, such as 3-opt (Willard, 1989), Or-opt (Garcia et al., 1994), 2-opt* (Garcia et al., 1994), (Jin et al., 2012), GENI (Gendreau et al., 1994), λ-exchange (Taillard, 1993) etc. Using multiple basic operators in TS show great effectiveness for solving variants of VRPs, such as CVRP (Jin et al., 2012), (Xu and Kelly, 1996), VRPTW (Rochat and Taillard, 1995), (Badeau et al., 1997) and VRPTW with split delivery constraints (Ho and Haugland, 2004) etc.

To solve VRPs, the tabu list can store objects such as nodes, edges (subroutes) or routes of the recently visited or best-visited solutions. Most of the proposed TS use specialised diversification and intensification strategies to guide the search (Bräysy and Gendreau, 2005b). The Taburoute scheme, proposed in (Gendreau et al., 1994), forbids a vertex to be reinserted into its original route in a random number of iterations. Adaptive memory, which allows random tabu duration, has been used to enhance TS for solving VRPs (Taillard, 1993), (Rochat and Taillard, 1995), (Taillard et al., 1997). Another frequently used strategy is to implement TS in parallel (Bräysy and Gendreau, 2005b), such as the implementation in CVRP (Jin et al., 2012) and VRPTW (Garcia et al., 1994), (Badeau et al., 1997). More variants of tabu search for solving VRP are developed by introducing different meta-heuristics, such as simulated annealing (Osman, 1993), (Küçükoğlu and Öztürk, 2015) and variable neighbourhood search (Polacek et al., 2004), (Paraskevopoulos et al., 2008).

(2) Variable neighborhood search

Variable Neighborhood Search (VNS) (Hansen and Mladenović, 1999) systematically changes neighbourhood structures to escape from the local optimum (Blum and Roli, 2003). The main idea behind VNS is that different neighbourhood structures can lead to different local optimums, many of which may be close to each other. Furthermore, the global optimum is a local optimum for all possible neighbourhood structures. Generally, VNS contains a *Local Search* phase to find the local optimum in the neighbourhood of the current solution, and a *Shaking* phase to jump out of the local optimum. The combination of the two steps enhances the intensification and diversification of the search process (Hansen and Mladenović, 2003).

Variable Neighbourhood Descent (VND) is the simplest and an effective variant in the family of VNS (Hansen et al., 2019). It changes the neighbourhood in a deterministic way (Hansen et al., 2010) and moves through several neighbourhood structures following a sequential order (usually manually specified from smaller to larger ones). Reduced VNS (RVNS) is another simple variant that randomly selects the neighbourhood structure rather than in a deterministic way. Basic VNS (BVNS) involves both types of changing mechanisms during the search process (Mladenović and Hansen, 1997). Several other variants of BVNS include the General VNS (GVNS) which applies VND in the *Local Search* step and the Skewed VNS (SVNS) which enhances the exploration of the region far from the incumbent solution (Hansen et al., 2010).

The main strength of VNS is its ability to adopt different classic improvement heuristics as neighbourhood structures for solving variants of VRPs, such as those used in VRPTW (Bräysy, 2003), multi-depot VRPTW (Polacek et al., 2004), Open VRP (Fleszar et al., 2009), and other variants. In-

stead of using classic operators, the study in (Chen et al., 2016) reviews the widely used neighbourhood operators and proposes the compound neighbourhood operators by composing several independent neighbourhood operators in VNS for VRPTW, resulting in a more effective exploration of the search space.

To balance exploration and exploitation during the search process, some algorithms combine VNS with other search strategies. For instance, the study in (Paraskevopoulos et al., 2008) incorporated TS into the *Local Search* phase of VNS and applies a solution reformation mechanism in the *Shaking* step for solving a realistic VRP. In another study (Goksal et al., 2013), VNS is combined with PSO to preserve the swarm diversity and make exploitation in the search space.

(3) LNS

Large Neighborhood Search (LNS) (Shaw, 1997) explores a larger neighbourhood to avoid getting trapped in a local optimum. It ruins the current solution by removing parts of the solution with a heuristic algorithm and then recreating it with another heuristic algorithm. The key components of LNS are the heuristics for ruining and recreating the solution.

In the context of VRPs, the criteria for selecting customers to remove and the locations of routes to insert are important in LNS. For solving VRPs (Shaw, 1997), (Shaw, 1998), side constraints have been efficiently incorporated into the search process. The LNS in (Shaw, 1998) for solving VRPTW concerns the relevance of customer visits using criteria like geographical metrics and time windows when selecting customers. Adaptive LNS (ALNS) (Ropke and Pisinger, 2006) is an extension of LNS that dynamically changes the neighbourhood structures during the search. It has

been tested on CVRP, VRPTW, the multi-depot VRP, and many other variants (Pisinger and Ropke, 2007), (Chen et al., 2021). A ruin-recreate operator is extracted from LNS as a basic operator for solving VRPs (Qu et al., 2020).

(4) Simulated annealing

Simulated annealing (SA) (Kirkpatrick et al., 1983) escapes from the local optimum by adaptively accepting a worse move. It accepts every better position during the search process and accepts a worse move based on a dynamically decreased probability, which mimics the physical annealing process of the temperature changes. The probability is determined by the metropolis criterion (Metropolis et al., 1953), i.e., $e^{-\delta/T}$, where $\delta = \mid f(s') - f(s) \mid$ represents the absolute value of the difference of the objective function values between the current solution s and candidate solution s', and T denotes the system temperature which decreases following a cooling scheme. Various cooling schedules have been developed (Ben-Ameur, 2004), such as linear cooling, geometric cooling, and Ludy and Mees cooling (Lundy and Mees, 1986). The traditional annealing process is extended in (Connolly, 1992) with a reheating mechanism to periodically increase the temperature.

SA was first used for solving VRP combining with the classic 3-opt heuristic in (Alfa et al., 1991). Later studies involve more neighbourhood operators to reach further search space. Various inter-route heuristics have been used in SA for solving VRPTW (Chiang and Russell, 1996), (Van Breedam, 1995). Parallel SA has been investigated for VRPs. The study for solving VRPTW in (Czech and Czarnas, 2002) adopts a set of processors each with its annealing process to search for the local optimum. A central processor determines whether to update the global optimum based on the best local

optimum from all processors.

Variants of SA share the same objective, which is to accept worse moves based on a decreasing probability. This probabilistic acceptance scheme can be seen as a relaxation mechanism based on a dynamic threshold. The similar mechanisms which accept worse moves based on a threshold value also include the Threshold Acceptance with a static threshold and the Great Deluge with an increasing probability as a threshold (Dueck, 1993). The study in (Han and Cho, 2002) combines Threshold Accepting with Great Deluge for VRP, leading to encouraging results.

(5) Iterated Local Search

Iterated Local Search (ILS) (Lourenço et al., 2003) escapes from local optimum by perturbing a current solution and restarting local search from the resulting solution (Blum and Roli, 2003). Variants of ILS differ in four aspects, i.e., the initial solution generation, perturbation, local search and acceptance criterion. The perturbation in ILS is important as it should be strong enough to escape from the local optimum but not like a random restart. The acceptance criterion determines whether to accept the perturbed solution and start the next iteration or reject it and keep searching from the current solution (Blum and Roli, 2003).

In the VRP literature, ILS has been applied by using multiple perturbative and local search heuristics. For example, in (Walker et al., 2012), multiple perturbative and local search heuristics were used to solve the VRPTW. ILS is a more general framework than other meta-heuristics, such as VNS (Blum and Roli, 2003), and thus can be easily extended by using other meta-heuristics as sub-components. For instance, the study in (Penna et al., 2013) incorporates ILS with a random neighbourhood ordering VND for solving Heterogeneous Fleet VRP. Another extension to ILS is by using

the history of the search with memory (Blum and Roli, 2003). The study in (Cuervo et al., 2014) developed the oscillating local search heuristic which can both explore a broad neighbourhood and store information on the neighbour solutions with a data structure.

(6) Greedy Randomised Adaptive Search Procedure

Greedy Randomised Adaptive Search Procedure (GRASP) (Feo and Resende, 1995) avoids getting stuck in a local optimum by using a multi-start strategy. Each iteration of GRASP consists of two steps, i.e., construction and local search (Boussaïd et al., 2013): in the first step, a feasible solution is generated with constructive heuristics; the built solution is then used as the starting point for local search in the second step. GRASP conducts these two steps iteratively, until reaching a specific number of iterations. Therefore, the search process uses multiple initial solutions to diversify the search (Blum and Roli, 2003).

GRASP has been applied for solving variants of VRPs, such as CVRP (Layeb et al., 2013), (Marinakis, 2012), and VRPTW (Kontoravdis and Bard, 1995), (Chaovalitwongse et al., 2003), (Layeb et al., 2013). Usually, the construction phase generates a feasible solution by a random greedy method which iteratively selects the best solution component from a set of randomised choices. Therefore, the local search phase could start with a solution likely to be close to the optimal solution and potentially reduce the time in the local search phase (Chaovalitwongse et al., 2003). To improve the effectiveness of the local search procedure, GRASP has been developed with multiple operators (Chaovalitwongse et al., 2003) and meta-heuristics (Marinakis, 2012), (Layeb et al., 2013).

2.2.3 Discussions

In summary, classical heuristics are problem-specific and consider the constraints of the problem domain. In contrast, meta-heuristics are more general-purpose as they use probabilistic and adaptive search strategies to explore a broader search space with multiple heuristics, and thus can be applied to a wide range of problems. However, both classical heuristics and meta-heuristics are problem-specific because the search space of both lines of algorithms concerns the direct solutions of the optimisation problem.

Hyper-heuristics (Cowling et al., 2000) are a class of search methodologies with a two-level framework to solve a wide range of optimisation problems. This class of algorithms is more general-purpose than classical heuristics and meta-heuristics, as they determine "at a higher abstraction level which low-level heuristics to apply" (Cowling et al., 2000). The low-level heuristics, e.g. algorithms or operators, are called to generate heuristic algorithms on the fly. The search space of hyper-heuristics is the space of low-level heuristics, while low-level heuristics operate in a space of problem solutions. One type of hyper-heuristics, selection hyper-heuristics, automatically combines low-level heuristics by iterative selection. Nowadays, more and more selection hyper-heuristics are developed for solving VRPs (Chen, 2018). Low-level heuristics are mostly the widely used classic heuristics and meta-heuristics.

Hyper-heuristics (Pillay and Qu, 2018) can be seen as one of the main streams in automated algorithm design (Qu et al., 2020), (Meng and Qu, 2021). The recent advances in hyper-heuristics (Drake et al., 2020) indicate the need for automated algorithm design for solving complex optimisation problems. It also aligns with the development direction of hybridisation for developing effective search algorithms. High-performance

algorithms can often benefit from combining the ideas of different methodologies (Chen, 2018), (Talbi, 2009). Combining multiple heuristics and different search frameworks could combine their strengths and allow the flexible exploration and exploitation of many neighbourhood structures. Intelligent strategies are required to control this hybridisation and efficiently switch between different neighbourhood structures. A more detailed review of hyper-heuristics is presented in the next section, i.e., a review of automated algorithm design.

In terms of different search frameworks in meta-heuristics, single solution-based meta-heuristics focus more on improving the quality of a single solution iteratively, while population-based meta-heuristics explore the search space by evolving a diverse set of candidate solutions and exploiting the best ones. Population-based meta-heuristics generally achieve higher diversification by maintaining many solutions, however, may require longer computation time to converge. Single solution-based meta-heuristics tend to converge faster to a single optimal solution due to their focused search. However, the meta-heuristics of each type of algorithm follow similar search frameworks, thus common procedures can be extracted to model more generalised search frameworks.

2.3 A review of learning in automated design of search algorithms

The design of effective algorithms for solving complex COPs involves making a large number of decisions, thus is a challenge for algorithm designers (Pillay et al., 2018b). The performance of manually designed algorithms highly depends on the expertise and effort of human experts, who may only

consider limited designs, leaving a large number of potential algorithms unexplored (Hoos, 2008). Automated algorithm design aims to make decisions in the algorithm design process with less or no human involvement (Qu, 2021a). Automated algorithm design can remove the burden of human experts from the tedious design process and enable the exploration of a larger scope of candidate algorithms, some of which may never be considered by manual designs (Meng and Qu, 2021).

Recent advances in artificial intelligence, especially evolutionary computation and machine learning, have led to a rapid development in the automated design of search algorithms (Qu, 2021b). However, the search process generates a considerable volume of data (Karimi-Mamaghan et al., 2022), which has been discarded in most meta-heuristic literature. From the perspective of machine learning, the data contains useful knowledge that can be used to automate the algorithm design process. Machine learning can be used to automate the design process of search algorithms mainly by building effective models to make decisions automatically. A variety of learning methods, ranging from regression to reinforcement learning, have been applied to utilise the information gathered during problem-solving.

Given recent progress and interest in automated algorithm design, it is crucial to gain a better understanding of how machine learning can be used in automated algorithm design based on the main findings in recent years. This section presents a review of the use of machine learning in the automated design of search algorithms for solving various COPs, including VRPs (i.e., the case study of the thesis). The representative studies in automated algorithm design are categorised based on an extended taxonomy. Each line of research is reviewed with a focus on how machine learning can be applied to automated algorithm design. The issues and research opportunities in each line of research are discussed, supporting the main

works and contributions of this thesis.

Section 2.3.1 presents the taxonomy extended by an existing taxonomy in (Qu et al., 2020) that guides the review process. The following sections, Section 2.3.2, Section 2.3.3, Section 2.3.4 and Section 2.3.5, provide a detailed review of the methodologies and recent progress in each area. Finally, Section 2.3.6 concludes the review and discusses future research directions.

2.3.1 An extended taxonomy of automated design of search algorithms

An existing taxonomy

With the increasing attention to automated algorithm design, fundamental issues on models and standards have been addressed. A taxonomy of automated algorithm design is proposed in (Qu et al., 2020), categorising three lines of research on automated algorithm design based on the search space under consideration, explained as follows.

- Automated algorithm configuration: aims at automating the parameter control of target algorithms.
- Automated algorithm selection: aims at automatically choosing the most appropriate algorithm(s) from a portfolio of algorithms with their associated parameters.
- Automated algorithm composition: aims at automating the composition of basic components of arbitrary algorithms.

Automated algorithm configuration as one theme of automated algorithm design aims at automating parameter settings or configurations of specific

target algorithm(s) for solving a collection of problem instances (Hutter et al., 2007). It searches in a space of parameter configurations of the target algorithms with fixed templates (Qu et al., 2020). The idea of using specialised methods for algorithm configuration can be traced back to several "configurable systems", consider for instance the DITOPS transportation scheduling system (Smith et al., 1996). This field of research has received a lot of attention, investigating various search algorithms, such as simulated annealing (Hutter et al., 2010a), tabu search (Battiti and Tecchiolli, 1994), (Nanry and Barnes, 2000), stochastic local search (KhudaBukhsh, 2009), evolutionary algorithms (Goldman and Tauritz, 2011), (Araya and Riff, 2014), (Liefooghe et al., 2011) and ant colony algorithms (Lopez-Ibanez and Stutzle, 2012). Great progress has been made in automated configuration, leading to the development of the widely adopted automated configuration frameworks F-Race (Birattari et al., 2010), ParamILS (Hutter et al., 2009), GGA (Ansótegui et al., 2009) and ISAC (Kadioglu et al., 2010).

Automated algorithm selection aims to automatically select the most appropriate algorithm(s) from a portfolio of candidate algorithms for solving a set of problem instances, concerning a search space of algorithms or algorithm portfolios (Qu et al., 2020). It has achieved remarkable results in several research areas (Kerschke et al., 2019), especially in the automated selection of SAT solvers (Xu et al., 2008) and evolutionary algorithms (Peng et al., 2010), (Tang et al., 2014), leading to several useful frameworks such as Aslib (Bischl et al., 2016) and Alors (Mısır and Sebag, 2017).

Automated composition aims to automatically compose or combine heuristics or components of arbitrary algorithms to solve the problem at hand online, producing new and generic algorithms (Qu et al., 2020). A main line of research in automated algorithm composition is that of hyper-heuristics (Qu et al., 2020), where a set of low-level heuristics as algo-

rithmic components are combined for problem-solving. Within a two-level framework, hyper-heuristics determine "at a higher abstraction level which low-level heuristics to apply" (Cowling et al., 2000). The low-level heuristics, e.g. algorithmic operators, are integrated or composed to design heuristic algorithms automatically. Different frameworks, including HyFlex (Ochoa et al., 2012), Hyperion (Swan et al., 2011), and EvoHyp (Pillay and Beckedahl, 2017), are developed to support the automated composition within the hyper-heuristic framework.

These three lines of research are fundamentally different in terms of the decision spaces. In the theme of research in automated composition, the algorithm design process is automated by composing algorithmic components. It takes a bottom-top method to compose flexibly a set of algorithmic components, thus generating new algorithms (Qu et al., 2020). The other two themes of research in automated configuration and selection take a top-down method, to consider parameters and algorithms themselves in the decision space.

In recent work by (Qu et al., 2020), the problem of algorithm design has been defined as a COP, namely the General Combinatorial Optimisation Problem (GCOP), which aims to search for the composition of algorithmic components for solving a given problem (Qu et al., 2020). Solving the GCOP is thus equivalent to automatically designing the best algorithms for optimisation problems. The GCOP model provides a standard of algorithm design by treating various algorithms as compositions of basic components, thus underpinning automated algorithm design.

Learning heuristics: an emerging topic in automated algorithm design

The current taxonomy of automated algorithm design mainly focuses on studies that concern a given set of decision choices, including existing algorithms, and existing algorithmic components, each with their associated heuristic and parametric settings. These decision choices are manually defined, thus the algorithm design process still needs explicit knowledge from experts.

Learning heuristics is emerging in solving many complex COPs, such as travelling salesman problems (TSPs) and VRPs (Li et al., 2022). These studies build learning models to act as important components of traditional heuristic algorithms, ranging from classic heuristics to meta-heuristics. The resulting heuristics are known as "learning heuristics". One example of learning heuristics is the work in (Wu et al., 2021), where the node selection policy of the classic 2-opt algorithm is replaced by a reinforcement learning model. The learning model is trained to learn the node selection process and propose a node pair to be selected for the 2-opt operation, given a solution to the optimisation problem. It's important to note that the learned node selection policy is embedded within the learning model and not explicitly defined as a handcrafted selection policy. However, the learned selection policy exists within the space of all possible policies that the learning model can learn.

From the perspective of reducing human involvement in the algorithm design process, learning heuristics fall within the scope of automated algorithm design by using learning models as automatically designed algorithmic components to replace manually designed algorithmic components. In other words, learning heuristics can be seen as heuristics incorporating

automatically designed algorithmic components. Using automatically designed algorithmic components can reduce the reliance on manually designed algorithmic components in automated algorithm design.

However, the existing taxonomy of automated algorithm design does not adequately include learning heuristics. Therefore, it is important to update the taxonomy to account for the recent advancements in learning heuristics and gain insights into different tasks and methodologies within automated algorithm design.

The proposed extended taxonomy

In this review, we distinguish the decisions involved in algorithm design into two levels, i.e., algorithm level and problem level, as illustrated in Figure 2.4. The problem-level decisions refer to the common and basic decisions of the problem domain for designing heuristics. The algorithm-level decisions, on the other hand, include the elements that are independent of the optimisation problem. This novel distinction between the two levels is important for distinguishing different research themes in automated algorithm design and their underlying decision spaces.

Figure 2.5 provides an overview of the general scheme of automated algorithm design methods. The current studies mainly focus on making algorithm-level decisions. Each automated algorithm design task utilises automation methods that operate within a specific decision space, while the resulting algorithms operate within the solution space of the optimisation problem being addressed. Feedback can be obtained from the algorithm design process and the problem-solving process associated with the optimisation problem.

Figure 2.4: A categorisation of the algorithm design decisions.

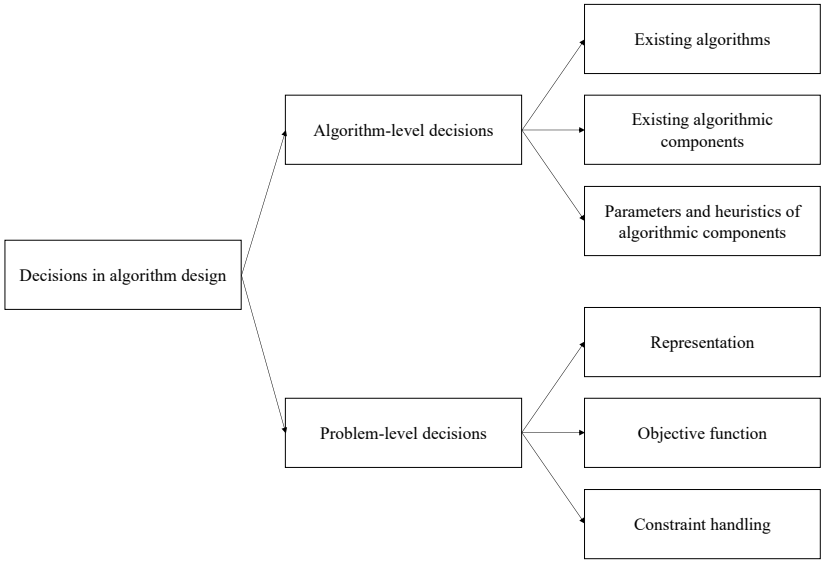
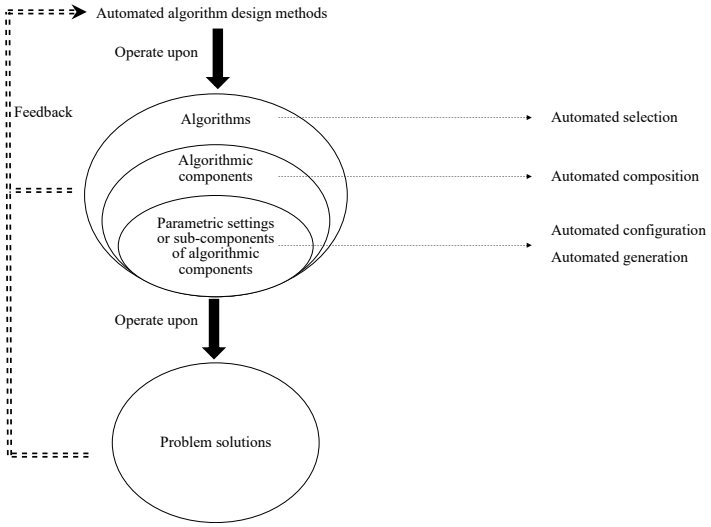


Figure 2.5: General scheme of how automated algorithm design works.



Based on the decision choices, this review extends the taxonomy of automated algorithm design in (Qu et al., 2020) into four lines as shown in Figure 2.6. The newly defined automated generation aims to automatically generate algorithmic components or heuristics of the target algorithm, such as the selection policy of 2-opt heuristics in (Wu et al., 2021). The designed algorithmic components are mostly embedded in learning models and thus are not as explicit as manually designed ones. This automated design process can lead to the creation of variant algorithmic components or heuristics that may have remained unexplored by human experts.

Figure 2.6: An extended classification of automated algorithm design.

Feedback		Tasks	Decision choices	Algorithm design space
Online learning	Automated Algorithm Design	Automated Configuration	Parameters within a template of target algorithm(s)	Variants of the target algorithms
		Automated Selection	A family of given target algorithms	A family of given target algorithms
Offline learning		Automated Composition	Algorithmic components or heuristics	Algorithmic compositions
		Automated Generation	Algorithmic components or heuristics of target algorithm(s)	Variants of target algorithmic components or heuristics
No learning				

While the other three lines of research require explicitly specifying the possible range of decision choices, automated generation can automatically produce algorithmic components without explicitly specifying the decision choices. Moreover, automated generation focuses on creating algorithmic components within target algorithms, instead of directly generating complete algorithms.

Based on whether the methods learn from the feedback of algorithm design and problem-solving, the methods of automated algorithm design can be categorised into learning-based and non-learning-based. Non-learning-based methods, such as random methods and search methods (Adenso-Diaz and Laguna, 2006), do not explicitly learn from feedback for decision-

making. On the other hand, learning-based methods acquire knowledge from the data for automated algorithm design. The learning-based methods can be further classified into online learning and offline learning: online learning occurs during the algorithm design process, while offline learning takes place prior to the algorithm design process.

The extended taxonomy in the thesis captures the recent advances in automated algorithm design. It provides a more comprehensive structure for discussing the findings of each line of research in automated algorithm design.

An overview of machine learning into automated algorithm design

In the research of automated algorithm design, machine learning plays an important role in the automated design process. The underlying rationale comes from two aspects. Firstly, a class of effective algorithms may share similar structures and contain useful knowledge. Secondly, machine learning models can discover the underlying knowledge of the data generated during problem-solving, which could be used to make decisions and generate alternative algorithms automatically. Effective learning in automated algorithm design can improve search effectiveness and establish intelligence from a larger scope of candidate designs which may not be explored in manual designs.

Various learning methods, ranging from regression to reinforcement learning, have been applied to make use of the information from problem-solving, thus automating the algorithm design tasks. With such recent progress and interest in automated algorithm design, it is crucial to gain a better understanding of how learning has been conducted in the decision-making of the algorithm design process. This review focuses on machine learning in the

four lines of research in the automated design of search algorithms, reviewing the recent developments, overviewing the widely investigated learning techniques and highlighting issues and research opportunities.

2.3.2 Learning in automated algorithm configuration

Automated configuration refers to the process of automatically determining the values of algorithmic parameters for a given target algorithm or set of algorithms (Qu et al., 2020). The research in algorithm configuration has been divided into two main categories (Karimi-Mamaghan et al., 2022):

- Parameter control, which involves adjusting the parameter settings of an algorithm during problem-solving.
- Parameter tuning, which involves identifying appropriate parameter settings before employing the algorithm. Once the parameters have been fine-tuned, they remain unchanged during problem-solving.

The application of machine learning techniques in automated configuration research has been explored in both the online and offline categories (Karimi-Mamaghan et al., 2022). In this review, we will focus on how feedback from the search process is utilised and categorise studies accordingly into online learning and offline learning approaches. Specifically, online learning corresponds to parameter control, while offline learning corresponds to parameter tuning in automated configuration.

Online learning

Considering the dynamic behaviour of the optimisation process, different parameter settings may be optimal at different stages of the optimisation

process (Goldman and Tauritz, 2011), (Aleti and Moser, 2016). The online learning methods for automated configuration are mainly based on the idea of reinforcement learning by trial and error (dos Santos et al., 2014). The candidate configurations are ranked or assigned with credits based on the historical performance online, thus a learning mechanism can learn to make appropriate decisions on parameter settings based on the feedback from the problem-solving process.

The online learning mechanisms are utilised to control critical parameters in various meta-heuristics, such as the crossover and mutation ratios of Genetic Algorithms for Knapsack Problems (Hong et al., 2002), prohibition parameters in Tabu Search for the MAX-SAT problem (Battiti and Protasi, 1997), the Maximum Clique problem (Battiti and Protasi, 2001), and the vehicle routing problem (Nanry and Barnes, 2000), (Osman and Wassan, 2002), (Wassan et al., 2008), (Battiti and Campigotto, 2011), and the ordering of neighbourhoods in Variable Neighbourhood Search for the network design problem (Hu and Raidl, 2006) and the TSP (dos Santos et al., 2014). Most research applies reinforcement signals in the search algorithms to adapt parameter settings.

In addition to the algorithmic parameters, online learning methods can also be applied to adapt the parameters of the objective functions. Due to a lack of complete knowledge of the problem, the objective function may not be fully defined and needs to be refined during the search for a satisfying solution (Battiti and Brunato, 2010), especially in multi-objective optimisation problems. An online learning mechanism has been developed in (Battiti and Brunato, 2010) to learn a weighted aggregation of objectives.

Offline learning

Offline learning for automated configuration is usually upon a set of training instances, aiming to train a learning model to configure target algorithms for solving unseen similar instances. Most studies aim to learn the mappings from instance features to the performance of target algorithms with specific parameter configurations. Based on a set of representative problem instances, an automatic configurator tunes the algorithm by selecting the parameters that yield the best performance (Kadioglu et al., 2010).

One promising approach in the automated configuration is using regression models to predict continuous parameters (Hutter et al., 2010b). Various regression models are employed to learn the relationship between the target algorithm performance and parameter settings (Hutter et al., 2010b), considering for instance the use of linear regression (Coy et al., 2001), (Hutter et al., 2006) and logistic regression (Ramos et al., 2005). These applications of regression models are based on domain-dependent features to approximate the performance of parametric settings.

Some research in hyper-heuristics applies regression models to determine the parameters of low-level heuristics (Asta and Özcan, 2014), (Tyasnurita et al., 2015). Instead of instance features, these studies use problem-independent features to represent the current search states, raising the generality of the automated configuration methods to the applications of different domains. A major limitation of using regression models is the limited number of parameters that can be dealt with.

Clustering shows to be effective for parameter tuning (Kadioglu et al., 2010) by improving the prediction of the learning models, particularly in the choice of instance features (Malitsky and Sellmann, 2010). ISAC (Ka-

dioglu et al., 2010) uses k-means clustering to cluster instances based on the feature vectors, thus, choosing the best parameter settings for each cluster of instances.

Racing algorithms have been widely applied in automated configuration (Birattari et al., 2010) by iteratively evaluating the algorithm configurations on a collection of benchmark instances and using statistical hypothesis tests to drop the worse-performing candidates (Hoos, 2011). F-Race algorithm (Birattari et al., 2002) is developed based on the racing mechanism to automate the configuration of ant colony algorithms for solving travelling salesman problems (TSP). Later in (Balaprakash et al., 2007), sampling F-Race and iterative F-Race are developed to effectively handle a larger configuration space. F-Race has been applied to various applications (Birattari et al., 2010), including ant colony algorithms (Birattari et al., 2002) and estimation-based local search algorithms (Balaprakash et al., 2010) for TSP, simulated annealing algorithms for vehicle routing problems (Pellegrini and Birattari, 2006), and metaheuristics for university timetabling problems (Rossi-Doria et al., 2002). In a recent study in (Gümüş et al., 2023), the F-Race algorithm shows significant performance in the automated configuration of Memetic Algorithms for cross-domain optimisation.

2.3.3 Learning in automated algorithm selection

Automated algorithm selection concerns selecting suitable algorithms or algorithm portfolios from a given set of algorithms for solving problems. A class of studies in automated selection use reinforcement learning techniques to learn suitable algorithms and the runtime allocation for iteratively constructing the algorithm portfolios during problem-solving. Another line

of research uses learning models built offline by learning from collected performance data to predict the performance and runtime of an algorithm for automated algorithm selection.

Meta-learning is a type of method originally proposed to support algorithm selection for classification and regression problems (Cruz-Reyes et al., 2012). In meta-learning, machine learning algorithms are introduced to acquire meta-knowledge that can be used to guide machine learning tasks. While machine learning algorithms are still the most studied task in meta-learning (Pappa et al., 2014), different domains of application have been explored using meta-learning (Cruz-Reyes et al., 2012). Most meta-learning studies for the design of meta-heuristics can be seen in the line of research in automated algorithm selection (Karimi-Mamaghan et al., 2022).

Online learning

The online learning methods in automated algorithm selection mainly employ the idea of reinforcement learning in automatically constructing algorithm portfolios. The main concept is to allocate more runtime to the algorithm with a better performance in a portfolio. Different application domains include job shop scheduling (Carchrae and Beck, 2005), SAT (Battiti and Protasi, 1997), (Gagliolo and Schmidhuber, 2006), ONE-MAX and TSP (Gagliolo et al., 2004).

Some studies combine online learning and offline learning techniques in automated selection. The study in (Carchrae and Beck, 2005) developed reinforcement learning formulas to learn the algorithm improvement per second for allocating more runtime to algorithms that perform well, combined with an offline trained Bayesian classifier predicting the best algorithm on new problem instances. In (Gagliolo et al., 2004), a general framework is

proposed for the automated portfolio of Genetic Algorithms with different settings of population size. A probability update scheme learns performance improvement of the variants of the algorithms to allocate runtime. A simple linear best-fit model is trained to predict useful pairs of algorithms and corresponding time allocations. In (Gagliolo and Schmidhuber, 2006), a bandit learning scheme is adopted for the runtime distributions of algorithms by learning their performance improvement.

Instead of predicting a suitable runtime for the algorithms, some studies in automated selection learn to reward the algorithm with better performance and select the algorithm with more credits, thus iteratively constructing the best portfolio, such as the application in planning systems (Roberts and Howe, 2006) and the development of Hydra (Xu et al., 2010). Q-learning algorithm is adapted in (Lagoudakis and Littman, 2000) to select algorithms for each instance based on the instance features.

Offline learning

Offline learning in automated algorithm selection mainly concerns predicting the performance of an algorithm and predicting the runtime of an algorithm for automated selection. Mainly four classes of machine learning approaches have been used in automated algorithm selection to conduct offline learning, including case-based reasoning (Pomerol, 1997), classification, regression, and statistic relational learning (Kotthoff et al., 2011).

Case-based reasoning (CBR) uses experience from solving similar training instances to solve new and unseen cases. The training instances and their solutions are stored in a case base, representing the knowledge or experience. CBR first identifies the similarity of the stored training instances to the new instance. The solution stored for the most similar case is used.

If no cases of enough similarity are available, the new case may either be solved from scratch (Schirmer, 2000) or by modifying the solutions of the most similar cases (Pomerol, 1997). Retaining the newly solved cases in the case base provides a learning mechanism to extend and refine the knowledge. Cases in the CBR system are usually represented by instance features against the best corresponding algorithm. CBR has been applied to solve project scheduling problems (Schirmer, 2000), and constraint solving problems (O'Mahony et al., 2008).

Classification is another widely applied method in automated algorithm selection based on instance features. Bayesian classification is applied in (Horvitz et al., 2001) to predict the runtime of an algorithm for a target instance on whether the runtime is short or long for solving the CSP and SAT problems. In addition, classification approaches have been applied more to predict the performance of an algorithm based on the features of specific instances. Some classifiers in automated selection include Decision Tree for solving bid evaluation problem (Guerri and Milano, 2004), Association Classification on adaptive planning system (Vrakas et al., 2003), and the K-Nearest Neighbour on SAT (Lindauer et al., 2015), (Malitsky et al., 2011) and black-box optimisation problem (Yuen et al., 2019).

Regression models in automated algorithm selection have also been used mainly to estimate an algorithm's runtime based on instance features. This has been mostly applied to SAT, including the use of ridge regression in the SATzilla system (Xu et al., 2007) and linear regression (Haim and Walsh, 2009), (Hutter et al., 2006). In (Hutter et al., 2014), the performance of random forests of regression trees has been evaluated to predict the algorithm runtime in automated algorithm selection for solving SAT, TSP and MIP. In addition, regression models can be used to predict the performance of an algorithm rather than the runtime based on instance features.

In (Messelis and De Causmaecker, 2014), M5-based regression models are applied to predict the performance scores of a set of algorithms based on instance features for solving project scheduling problems. Support vector machine is also used to assign labels to objects (Noble, 2006) by training a set of instances and the algorithm performance, thus learning to rank the algorithms. The application of a support vector machine in automated selection involves the black-box optimisation problem (He et al., 2019) and SAT problems (Kotthoff et al., 2011).

The offline learning methods in automated selection depend on the instance features to make accurate predictions not only on runtime but also on the algorithm performance. Thus, reasonable and accurate instance features present be the key issue for training the model. Some studies investigate feature engineering for identifying the key instance features by using feature selection (Xu et al., 2007), and feature transformation (Pihera and Musliu, 2014).

2.3.4 Learning in automated algorithm composition

In the theme of research on automated composition, the algorithm design process is automated by composing a set of algorithmic components to generate new algorithms. Hyper-heuristics is a class of methodologies for selecting or generating heuristics to solve optimisation problems (Burke et al., 2013). A line of research in hyper-heuristics, i.e., selection hyper-heuristics, falls into the line of research in automated composition (Qu et al., 2020). The automated composition in selection hyper-heuristics is conducted within a two-level hyper-heuristic framework, where a set of low-level heuristics which can be seen as algorithmic components, are integrated or composed to design heuristic algorithms automatically.

The performance of a hyper-heuristic highly depends on not only the design of the low-level heuristics but also the way they are composed for problem-solving. A line of research in selection hyper-heuristics focuses on focuses on determining an elite set of algorithmic components (i.e., low-level heuristics) for composition. This task is typically accomplished using various online and offline learning methods. However, more attention has been given to using learning methods to automatically compose algorithmic components for problem-solving.

This section reviews the learning techniques used in automated composition, particularly based on the hyper-heuristic framework. Section 2.3.4 focuses on automatically determining the set of algorithmic components, while Section 2.3.4 discusses automatically composing algorithmic components.

Learning to determine the algorithmic components

Several studies in selection hyper-heuristics investigate how the choice of different sets of low-level heuristics impacts the effectiveness of hyper-heuristics in personnel scheduling problem (Cowling and Chakhlevitch, 2003) (Chakhlevitch and Cowling, 2005), nurse rostering and home care scheduling problems (Mısır et al., 2013), workforce scheduling problems (Remde et al., 2012) and course timetabling and VRPs (Soria-Alcaraz et al., 2017). There is evidence suggesting the importance of selecting an appropriate and compact set of low-level heuristics in hyper-heuristics, as an unnecessarily large number of low-level heuristics can impair their performance (Soria-Alcaraz et al., 2017); while using a smaller subset could obtain similar performance within a shorter time (Chakhlevitch and Cowling, 2005). Some of the low-level heuristics do not make valuable contributions

to the search but only slow down the search, while some are more suitable for specific instances than others (Mısır et al., 2013). Therefore, determining an appropriate set of components with effective learning is essential to achieve high performance of the automatically composed algorithms.

In the research of automatically determining a set of low-level heuristics for composition, online learning methods evaluate the performance of the components and exclude the worst one iteratively (Misir et al., 2010). Two learning strategies are investigated in (Chakhlevitch and Cowling, 2005) for determining an elite subset of low-level heuristics to effectively solve the trainer scheduling problem, namely the warming up approach which learns only in the early iterations, and a step-by-step reduction approach which continues to learn until a satisfying size of the elite set is reached. A simple learning mechanism is applied in (Misir et al., 2010) to maintain a subset of elite heuristics for different phases of the search by excluding some heuristics for a given number of phases. To evaluate each heuristic, a snapshot performance during a phase (i.e., several iterations), rather than the accumulated performance during the search, is considered. In (Remde et al., 2012), the step-by-step reduction learning approach in (Chakhlevitch and Cowling, 2005) shows limited generality across different domains. Some low-level heuristics which are discarded early seem to be important later in the search, leading to poor overall performance.

The offline learning methods determine the elite set of low-level heuristics for automated composition mainly based on the evaluation of the collected search data (Soria-Alcaraz et al., 2017). In (Soria-Alcaraz et al., 2017), the performance of each individual low-level heuristic is evaluated offline by using statistical tests to determine an elite subset. Two fitness landscape probing techniques, i.e., evolvability and landmarking, are used. Landmarking is more informative than evolvability in the evaluation. Evolv-

ability seems to be more reliable in quickly distinguishing good from bad heuristics.

Several issues should be considered while learning to determine an elite set:

- Size of the set. A small number of low-level heuristics may obtain similar performance within a shorter computation time; while many low-level heuristics will require more time to explore the larger search space and obtain similar performance without intelligent decision-making with learning.
- Evaluation metrics. Suitable evaluation metrics need to be chosen for short-term and long-term learning during the search, as each low-level heuristic may be effective at different stages of the whole search process (Cowling et al., 2002).
- Learning fashion. The behaviour of low-level heuristics is different for different problems (Kendall et al., 2002). The most effective heuristics may be different for different scenarios (Cowling et al., 2002). Online learning is therefore applied more often than offline learning in the literature.

Learning to compose algorithmic components

In the line of research in automated composition, some studies focus on combining algorithmic components of specific types of algorithms or target algorithms. The most studied algorithms include SAT solver (KhudaBukhsh, 2009), SAs (Franzin and Stützle, 2019), ILS (Mascia et al., 2013), and MOEAs (Bezerra et al., 2015), (Lopez-Ibanez and Stutzle, 2012).

Selection hyper-heuristics can be seen as a main line of research in auto-

mated composition. In selection hyper-heuristics, the automated composition task is to determine the most suitable low-level heuristics to compose for the design of heuristic algorithms for solving the optimisation problem at hand. Some studies also concern the decision-making of suitable acceptance criteria in automated composition. The online learning models use the information gathered during problem-solving to combine algorithmic components, while offline learning often learns from effective algorithmic compositions on a set of training instances to generalise the decision-making on new instances.

(1) Online learning

In selection hyper-heuristics, the online learning methods for automated composition mainly follow the concept of reinforcement learning. Learning models are built to estimate the future performance of an algorithmic component given feedback during problem-solving, thus making predictions on the suitable heuristics or components to be combined next (Khamassi, 2011), (Di Gaspero and Urli, 2011), (Mısır et al., 2012), (Turky et al., 2020). The learning methods can focus on the individual performance of each low-level heuristic and the transition performance between pairs and sequences of low-level heuristics, such as the studies in (McClymont and Keedwell, 2011a) with Markov chain (Kemeny and Snell, 1976) and (Kheiri and Keedwell, 2015) with Hidden Markov Model (Baum and Petrie, 1966). The choice function proposed in (Cowling et al., 2000) can be seen as learning from both individual performance and the transition performance of heuristics.

The learning process can be sensitive to memory length. A few studies investigate the length of the learning period for example on the Pseudo-boolean optimisation (Lissovoi et al., 2017), LeadingOnes (Doerr et al.,

2018), (Lissovoi et al., 2020b), and OneMax problems (Lissovoi et al., 2020a). The learning uses a random gradient method which continues to exploit the currently selected heuristic as long as it is successful. It can be seen as embedded with a reinforcement learning mechanism with the shortest possible memory (Lissovoi et al., 2020b). This simple learning method shows to learn to select the optimal low-level heuristics for standard benchmark functions, leading to optimal asymptotic runtimes for LeadingOnes (Lissovoi et al., 2020b) and OneMax (Lissovoi et al., 2020a). However, the best choices for memory length can be different in various situations (Lissovoi et al., 2020b).

(2) Offline learning

In the research of offline learning in selection hyper-heuristics, various classification methods are explored to learn a mapping between certain search states and the suitable low-level heuristics (Burke et al., 2013), thus predicting the suitable low-level heuristics to compose. The predictive power of the learning models greatly depends on the features describing a search state (Guyon and Elisseeff, 2003). Some problem-dependent and solution features have been identified to characterise search states. A few studies investigate problem-independent information, such as the performance of the applied algorithmic components and search stage information (Asta and Özcan, 2014), (Meng and Qu, 2021), (Yi et al., 2022).

Learning classifier systems are rule-based systems driven by genetic algorithms and reinforcement learning to learn appropriate actions subject to certain conditions (Urbanowicz and Moore, 2009). This class of methodologies shows to be effective for automatically composing algorithms by learning "a solution process". This process transforms the problem from the initial state to a solved goal state by iteratively choosing one of the var-

ious heuristics at each state (Ross et al., 2002). Some of the investigated optimisation problems include the constraint satisfaction problems (Ortiz-Bayliss et al., 2013b), modularized fleet mix problem (Shafi et al., 2012), bin packing problem (Ross et al., 2002), (Marín-Blázquez and Schulenburg, 2003), (Marín-Blázquez and Schulenburg, 2006), and cutting stock problem (Terashima-Marín et al., 2005). The search states in these problems are represented by problem-specific features, such as the constraint density and constraint tightness for constraint satisfaction problems (Ortiz-Bayliss et al., 2013a), and the number of remaining unpacked items for the bin packing problems (Marín-Blázquez and Schulenburg, 2003).

Classification techniques, such as logistic regression (Ortiz-Bayliss et al., 2013a) and neural networks (Ortiz-Bayliss et al., 2011), have been used to compose low-level heuristics in hyper-heuristics by learning a mapping between problem states and heuristics for solving constraint satisfaction problems. Apprenticeship learning learns from the observations while an expert is in action (Abbeel and Ng, 2004). It has been employed by combining with classification techniques in selection hyper-heuristics for solving bin packing problems (Asta et al., 2013) and VRPs (Asta and Özcan, 2014), (Tyasnurita et al., 2015), (Tyasnurita et al., 2017). The data generated by an expert hyper-heuristic is used to train the classifier for predicting the low-level heuristics and acceptance criteria for solving new instances. Various classification or regression methods are investigated, such as linear regression (Asta and Özcan, 2014), decision tree (Asta and Özcan, 2014), k-means classification (Asta et al., 2013), neural networks (Tyasnurita et al., 2015), (Tyasnurita et al., 2017), (Yates and Keedwell, 2017), associative classification (Thabtah and Cowling, 2008).

The apprenticeship learning-based approaches show to be able to generalise the knowledge extracted from small to larger problem instances (Asta

et al., 2013), from one type to different types of instances (Tyasnurita et al., 2017), and achieving better performance than the manually designed algorithm which is seen as an expert to be learned. Particularly, using problem-independent features to represent search states can lead to promising performance with a higher level of generality of the extracted knowledge (Asta and Özcan, 2014). Additional distance metrics used in (Tyasnurita et al., 2017) for representing search states contribute to better learning, indicating that enriching the information with appropriate key features in the learning environment can potentially lead to highly effective problem-solving.

2.3.5 Learning in automated algorithm generation

In the theme of research on automated generation, the automation in the algorithm design process lies in the automated design of algorithmic components within target algorithms. Instead of using manually designed algorithmic components, automated generation uses machine learning techniques to conduct the operation of algorithmic components within the target algorithms. The automatically designed algorithmic components are embedded in the learning techniques.

Learning heuristics fall into this line of research. Learning heuristics are the methods that utilise machine learning techniques within heuristic algorithms to perform specific operations of algorithmic components (Wu et al., 2021). The ability of autonomous and offline learning results in learning heuristics not requiring any explicit operation rules and performing the desired operation to replace specific algorithmic components.

Table 2.1 overviews some representative studies in learning heuristics, cate-

Table 2.1: Classification of papers studying automated algorithm generation.

Reinforcement learning: algorithm design task modelled as reinforcement learning task				
Literature	Target algorithms	Algorithmic components	Optimisation problems	Learning techniques
(Liu and Zeng, 2009)	Genetic algorithm	Mutation operator	TSP	Q-learning
(Khalil et al., 2017)	Greedy algorithm	Node selection	TSP	Q-learning
(d O Costa et al., 2020)	2-opt	Node pair selection	TSP	Policy gradient
(Wu et al., 2021)	2-opt, node swap, and relocation	Search policy	TSP, CVRP	Reinforcement Learning with self-attention-based policy network
(Zheng et al., 2021)	Lin-Kernighan-Helsgaun	Edge selection policy	TSP	Q-learning, Sarsa and Monte Carlo

Supervised learning: algorithm design task modelled as sequence-to-sequence task				
Literature	Target algorithms	Algorithmic components	Optimisation problems	Learning techniques
(Hottung and Tierney, 2019)	Large Neighbourhood Search	Repair operator	CVRP, SDVRP	Attention-based Deep Neural Networks
(Gao et al., 2020)	Large Neighbourhood Search	Reinsertion operator	CVRP, VRPTW	Recurrent Neural Network
(Sui et al., 2021)	3-opt	Link selection and reinsertion location	TSP	Pointer network and feature-wise linear modulation network
(Zhou et al., 2023)	Large Neighbourhood Search	Decision variable to remove	Real world VRP in Airport	Graph Convolutional Network

gorised based on the machine learning tasks. The heuristics with automatically generated algorithmic components (embedded in learning models) have demonstrated superior performance compared to the classic heuristics with the handcrafted algorithmic components on new instances (Deudon et al., 2018). Various machine learning techniques, including reinforcement learning and sequence-to-sequence learning, have been applied as automatically designed selection policies or operators of heuristics. In terms of the target algorithms, classic heuristics have been the main algorithm frameworks of interest, while other studies explore the application in metaheuristics. Specifically, Large Neighbourhood Search has attracted more research interest in these studies.

End-to-end approaches are an emerging and promising area of research that apply machine learning techniques for solving complex COPs, such as TSP and VRPs (Vesselinova et al., 2020). These learning techniques learn the mapping between the optimisation problem and its solution (Li et al., 2022), enabling the learning techniques to solve COPs from the beginning until the final solution (Bogyrbayeva et al., 2022). By using

machine learning techniques, end-to-end approaches automate the process of solving COPs without relying on manually designed search algorithms. In terms of the main strategy for solving COPs, end-to-end approaches fundamentally differ from traditional search algorithms. Specifically, search algorithms solve COPs by systematically exploring the space of possible solutions step-by-step, whereas end-to-end approaches learn to solve COPs automatically without handcrafted search techniques. A number of studies have systematically reviewed end-to-end approaches for solving COPs, such as (Vesselinova et al., 2020), (Kotary et al., 2021), as well as the review in the VRP literature (Bogyrbayeva et al., 2022).

Learning heuristics can be seen as incorporating the principles of end-to-end learning within the step-by-step problem-solving process of search algorithms. The effectiveness of learning heuristics indicates the benefits of leveraging the power of machine learning to capture the underlying patterns of the problem and solutions and incorporate them into search algorithms. Incorporating end-to-end approaches with other search frameworks such as hyper-heuristics might enable the learning of more complex search policies for advanced search schemes (Wu et al., 2021), which is an important future direction in the line of automated generation.

2.3.6 Summary

This review provides an overview of the machine learning techniques applied to the automated design of search algorithms for solving combinatorial optimisation problems. We discuss the role of machine learning in automated algorithm design, based on a new taxonomy that categorises related research into different lines of study, providing insights and guidance on how machine learning can be implemented for different automated

algorithm design tasks.

This review aims to promote the collaboration between machine learning and optimisation research and attract more attention to the automated design of search algorithms. By automating the design process, algorithm designers can focus on solving more complex optimisation scenarios with less human involvement. Meanwhile, evidence suggests that automatically designed search algorithms with effective learning models can outperform manually designed algorithms. Therefore, we believe that further research in this area has the potential to lead to significant advances in optimisation and machine learning.

Despite recent advances in applying machine learning in automated algorithm design, several challenges remain that require attention from multiple disciplines. Addressing these challenges could open up new research directions in the field of automated algorithm design.

One major challenge is the need to establish standard algorithm design frameworks. The recently developed GCOP model (Qu et al., 2020) provides a new standard for algorithm design. However, current frameworks in automated composition only cover a subset of algorithmic components in GCOP, and therefore do not provide adequate support for automated algorithm design based on GCOP. Hyper-heuristics, for instance, only consider a limited search space consisting of problem-specific low-level heuristics, rather than elementary algorithmic components. In principle, larger design spaces can be expected to include better solutions (better algorithm designs in this case) (Hoos, 2008). To design more effective algorithms, it is essential to assess the performance of elementary algorithmic components and explore insights into designing effective algorithm compositions with these components. Therefore, the newly established GCOP model

requires coherent frameworks to assess the performance of the elementary algorithmic components and explore insights on designing effective algorithm compositions with these components.

Another challenge is to systematically compare different learning approaches. With an increasing number of learning models developed for automated algorithm design, especially in automated composition (Thabtah and Cowling, 2008) and selection (Kotthoff et al., 2011), (Pihera and Musliu, 2014), there is a need to understand how to choose suitable learning models for specific algorithm design scenarios. The choice of learning models matters to the algorithm performance (Kotthoff et al., 2012), (Hutter et al., 2014) and depends on the specific automation tasks. Conducting a systematic comparison of learning models based on commonly adapted frameworks would provide guidance and ensure consistency in the literature.

Furthermore, there is potential for more advanced learning models to improve different aspects of effective algorithm design, including Conditional Random Fields (Lafferty et al., 2001), Sequential Pattern Mining (Fournier-Viger et al., 2017), and Deep Reinforcement Learning (Mnih et al., 2013). These techniques possess specific properties that could be beneficial for automating algorithm design.

Last but not least, interpreting learning models can be challenging. Although more machine learning models have been proposed for automated algorithm design, the rich knowledge generated during the search is often implicit and difficult to interpret. The hidden knowledge in machine learning models is rarely investigated in the literature. There is evidence that online learning models along with classical mining techniques could reveal interesting patterns, reaching a new level of generality as well as effectiveness (Carchrae and Beck, 2005), (Asta and Özcan, 2014), (Tyasnurita et al.,

2015). Using data mining techniques to gain insightful and interpretable knowledge from effective algorithm designs could also aid in understanding the performance of algorithmic components.

The current studies in the field focus on algorithm-level decisions. However, future investigations could concentrate on automating decision-making processes at the problem level, further advancing the capabilities of automated algorithm design.

2.4 Machine learning

Machine learning is a subfield of artificial intelligence that allow machines to learn from data without explicitly programmed (Bishop and Nasrabadi, 2006), (Alpaydin, 2020). Machine learning tasks can be generally classified into supervised, unsupervised and reinforcement learning (Bishop and Nasrabadi, 2006), explained as follow:

- Supervised learning: this type of machine learning technique is trained on a labelled dataset where each input example in the training dataset is associated with a known output or target variable. It aims to learn a mapping from the input to the output based on the labelled examples, thus the model can generalise to new and unseen examples and predict the output for those examples. Classific supervised learning tasks include classification and regression. Some widely used algorithms for supervised learning include decision trees, random forests, and neural networks.

- Unsupervised learning: this type of technique is trained on an unlabeled dataset. The aim of unsupervised learning is to extract the un-

derlying structure or patterns in the data without any explicit labels. Some popular unsupervised learning tasks are clustering, dimensionality reduction, and anomaly detection. Another great interest in unsupervised learning is unsupervised pattern mining, i.e., finding interesting associations (relationships, dependencies) in large sets of data items without any prior knowledge of the group labels or target variable (Hastie et al., 2009).

- Reinforcement learning: it aims to make the most suitable action at a specific state by interacting with and getting feedback from an environment, thus maximising the cumulative reward over time.

Different machine learning models are designed to model different types of data. It is important to choose the right model for the specific problem at hand. Some common types of data that machine learning models are designed to handle include numerical data, categorical data, text data, image data and sequential data. In many real-life applications, such as in bioinformatics (Wang et al., 2007), webpage click-stream analysis (Fournier-Viger et al., 2012b) and market basket analysis (Srikant and Agrawal, 1996), the studies are interested in valuable patterns in sequential data. Sequential data consists of sequences of items that are associated with time-related attributes (Zhao and Bhowmick, 2003), thus having high complexity. Machine learning techniques used for modelling sequential data are designed to take into account the temporal or spatial dependencies between data points (Dietterich, 2002).

This thesis aims to utilise machine learning to learn from the effective algorithm compositions of elementary algorithmic components to support automated algorithm design. The investigations of the thesis are based on the hypothesis that algorithmic compositions might exhibit time-series charac-

teristics and dependencies among the algorithmic components. Therefore, the data can be seen as sequential data, presenting an opportunity to explore sequential modelling techniques in machine learning. In what follows, a description of machine learning methods used in this thesis is provided. These techniques have been widely used in modelling sequential data (Dietterich, 2002).

2.4.1 Markov chain

Markov chain is one of the most important and fundamental algorithms in machine learning (Tian et al., 2018). Markov chain is a widely used model for modelling sequential data. It has been widely applied for modelling queueing systems, remanufacturing systems, inventory systems, financial risk management, natural language processing (Manning and Schutze, 1999) and many other practical systems (Ching and Ng, 2006).

A Markov chain is a statistical model describing a sequence of states with certain probabilities to transfer between each other (Kemeny and Snell, 1976). It describes a stochastic process based on the Markov assumption, i.e., the probability of transitioning from one state to another depends only on the current state and not on the previous states (Ching and Ng, 2006).

A Markov chain can be represented as a directed graph, where each state is a node in the graph, and each transition from one state to another is represented by a directed edge. The probabilities associated with each edge represent the likelihood of transitioning from one state to another. A transition probability matrix describes the transition probabilities between states.

In the context of algorithm design, Markov chains provide a way to model

the composition of algorithmic components as stochastic transitions between states. In selection hyper-heuristics (McClymont and Keedwell, 2011b), the Markov chain has been combined with reinforcement learning to model and support decision-making in the heuristic selection strategy. The states refer to low-level heuristics. The transition probability between each pair of low-level heuristics is learned according to the optimisation performance during problem-solving. Low-level heuristics with better performance are more likely to be chosen.

2.4.2 Neural networks

Neural networks are highly effective learning models for a wide range of machine learning tasks. In learning sequential data, recurrent neural networks (RNNs) (Goodfellow et al., 2016) show great promise by considering the context information of sequences (Rao et al., 2018). The most well-known model in recurrent networks is Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) which can capture the long and short-term dependencies or temporal differences in a sequence.

Transformer networks, the attention-based models, have received great attention and outperformed LSTMs in various applications of Natual Language Processing (NLP) (Zeyer et al., 2019). Transformer networks are fundamentally different from LSTMs in the way of processing sequences.

In NLP, LSTMs have also been a target of criticism for their limited performance in handling lengthy sentences (Giuliari et al., 2021). When processing a lengthy sentence, if the output depends on a specific input word, the last LSTM unit may not be able to capture the total essence of the sentence. Transformer networks can give proper attention to it, thus better

at long sequence-to-sequence prediction. Even though Transformer outperforms LSTM in many applications of NLP, it also seems to have more problems with generalisation in some domains (Zeyer et al., 2019).

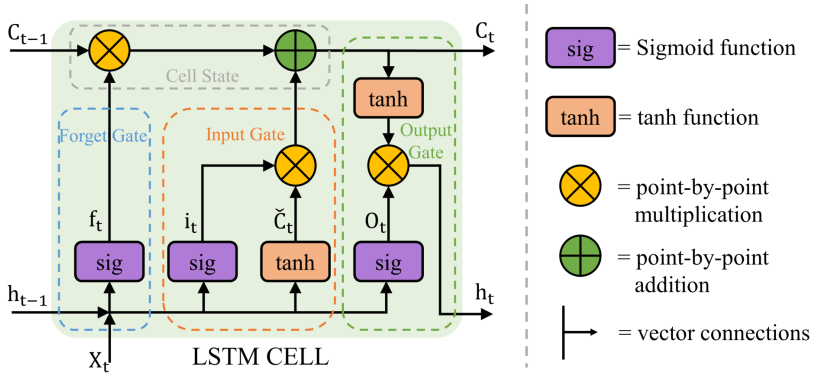
LSTM

Recurrent neural networks (RNNs) (Mandic and Chambers, 2001) are a class of sequence-based neural networks in modelling sequence learning tasks (Nammous and Saeed, 2019). The structure of an RNN is similar to a standard multi-layer network, with additional hidden units associated with the time when connected (Jurgovsky et al., 2018). Such connection between hidden units allows information from one step to be passed to the next, thus discovering temporal correlations in a sequence of inputs. A major issue with RNNs is that long-time lags are inaccessible in backpropagation, so it is hard to handle long-term dependencies in sequences (Wan et al., 2020).

LSTM (Hochreiter and Schmidhuber, 1997) is a variant of RNNs, which integrates a gating mechanism to resolve the long-term dependency issue, as shown in Figure 2.7. Each LSTM cell retains an internal cell state to store the memory of the last input, and a hidden state which stores the information of the last output. Three gate units, i.e., input gate, output gate and forget gate are introduced in LSTM to optionally let information through, the amount of which is decided by the employed sigmoid activation function (output between 0 and 1). The processed new information, i.e., cell state and hidden state, is then carried over to the next time step.

With this sequence specialisation, RNNs (including LSTM) interpret the input data as a data cube with three dimensions as shown in Figure 2.8. This representation is different from conventional classifiers which take a

Figure 2.7: The structure of a basic LSTM cell (Smagulova and James, 2020). At each time step t, X_t is an input vector, C_t denotes the cell state vector, and h_t is the hidden state vector calculated based on C_t. Three gating units, i.e., input gate, forget gate and output gate, return vectors denoted as i_t, f_t and O_t, respectively.



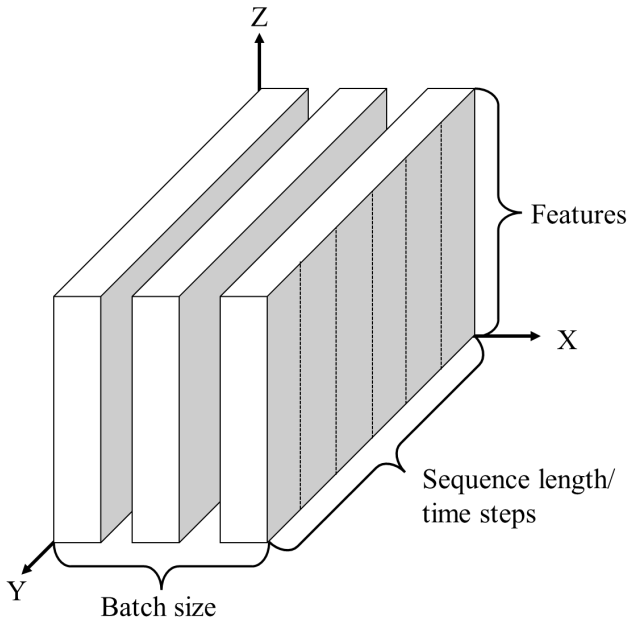
2-dimensional matrix as the input, each row consisting of features of each sequence.

Transformer

Transformer networks are the state-of-the-art NLP algorithms since proposed by (Vaswani et al., 2017). Instead of sequential processing mechanisms in RNNs, They are solely based on attention mechanisms which enhance some parts of the input data to focus on the small, but important, parts of the data (Vaswani et al., 2017). Utilising the attention mechanisms, Transformers estimate which part of the input sentence to focus on when modelling sequences, thus better at capturing the key component and logic of sentences for understanding the semantics of a whole document.

The original transformer networks utilise an encoder-decoder structure, as shown in Figure 2.9 (Vaswani et al., 2017), for sequence-to-sequence learning. The encoder structure (on the left half of Figure 2.9) takes the input sequence in the symbol representations to the Input Embedding layer for learning the continuous representations of the sequence. To make use of

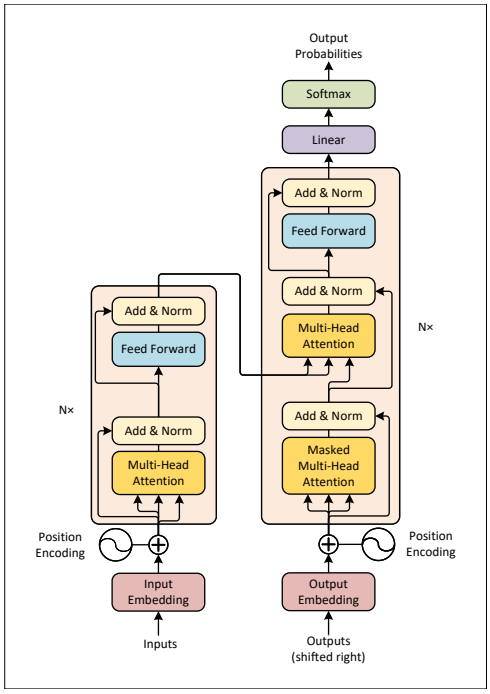
Figure 2.8: Input data representation in RNNs (including LSTM) (Skydt et al., 2021).



the order of the sequence, the transformer networks utilize the Position Encoding layer to use positional encoding vectors to inject the positional information of elements in the sequence. The sequence in vector representation is then fed into the Transformer block. The Transformer block consists of N identical layers, each consisting of two sub-layers. The first layer implements the multi-head self-attention mechanism. The second layer is composed of position-wise fully connected feed-forward networks. Each of the two sub-layers is linked with a residual connection, followed by layer normalisation.

The decoder structure (on the right half of Figure 2.9) receives the output of the encoder and combines it with the decoder output at the previous time step to generate the next. It is also composed of N identical layers. For the decoder, there is an additional sub-layer to perform multi-head attention on the output of the encoder.

Figure 2.9: The structure of the original Transformer network for sequence-to-sequence learning (Vaswani et al., 2017).



2.4.3 Sequential rule mining

Temporal or time-series data consists of sequences of items that are associated with time-related attributes (Zhao and Bhowmick, 2003), thus are of high complexity. A sequential database is a special case of a time-series database that consists of sequences of ordered events with or without the concrete notion of time (Zhao and Bhowmick, 2003). In the sequential database, a sequence is an ordered list of symbols or nominal values (Fournier-Viger et al., 2017).

Exploring the specific order of the occurrences of the events is a significant research direction in the research of sequential database (Zhao and Bhowmick, 2003). Initiated by (Agrawal and Srikant, 1995), sequential pattern mining is proposed as the problem of mining interesting sub-sequences in a set of sequences (Fournier-Viger et al., 2017). Numerous sequential pattern mining algorithms have been developed to search for sequential patterns efficiently with different strategies and data structures (Fournier-

Viger et al., 2017).

For sequential patterns, although a sub-sequence occurs frequently (e.g., {a, b}), one of the items ({a}) might not have a high possibility to be followed by the other items ({b}) among all the sequences. Therefore, one of the important limitations of sequential pattern mining is that sequential patterns might be worthless for decision-making or prediction (Fournier-Viger et al., 2017). Sequential rule mining is a variation of sequential pattern mining to address the above-mentioned limitation with more accurate sequence prediction.

As defined in (Fournier-Viger et al., 2011), a sequence database is a set of sequences $S = \{s_1, s_2, \ldots, s_x\}$ with a set of items $I = \{i_1, i_2, \ldots, i_y\}$ occurring in S. Each sequence is an ordered list of itemsets, i.e., $s =< I_1, I_2, \ldots, I_z >$. An example sequence database is shown in Figure 2.10 (left). It contains four sequences identified by IDs 1, 2, 3 and 4, each composed of itemsets derived from the items within the set $I = \{a, b, c, d, e, f, g\}$. The first sequence $< \{a, b\}, \{c\}, \{f\}, \{g\}, \{e\} >$ contains five itemsets, indicating that items a and b appeared simultaneously, followed by c, then f, subsequently by g and lastly e.

Figure 2.10: A sequence database (left) and some sequential rules found (right) (Fournier-Viger et al., 2011).

ID	Sequences
seq1	<{a, b}, {c}, {f}, {g}, {e}>
seq2	<{a, d}, {c}, {b}, {a, b, e, f}>
seq3	<{a}, {b}, {f}, {e}>
seq4	<{b}, {f, g, h}>

ID	Rules	sup	conf
r1	{a, b, c}=>{e}	0.5	1.0
r2	{a}=>{c, e, f}	0.5	0.66
r3	{b}=>{e, f}	0.75	0.75
r4	{c}=>{f}	0.5	1.0

A sequential rule $X \rightarrow Y$ is a relationship between two itemsets $X, Y \subseteq I$, suggesting if items in X occur in a sequence, the items in Y are likely to occur afterwards in the same sequence (at any position after X) (Fournier-

Viger et al., 2011). Figure 2.10 (right) shows four rules found in the database as an example. The first sequential rule $\{a, b, c\} \rightarrow \{e\}$ can be interpreted as if the items in $\{a, b, c\}$ are contained in a sequence, the item of $\{e\}$ are likely to occur afterwards in the same sequence. In the example sequence database, the rule $\{a, b, c\} \rightarrow \{e\}$ occurs in the sequence $< \{a, b\}, \{c\}, \{f\}, \{g\}, \{e\} >$, whereas the rule $\{a, b, f\} \rightarrow \{c\}$ does not, because item c does not occur after f.

Two measures have been defined for a sequential rule (Fournier-Viger et al., 2011) as follows:

- Support of a rule $X \rightarrow Y$: $sup(X \rightarrow Y) = sup(X \blacksquare Y)/|S|$. $sup(X \blacksquare Y)$ represents the number of sequences where all the items of X appear before all the items of Y.
- Confidence of a rule $X \rightarrow Y$: $conf(X \rightarrow Y) = sup(X \blacksquare Y)/sup(X)$. $sup(X)$ denotes the number of sequences that contains the items of X.

The sequential rule mining problem is given certain user-defined thresholds $minSup$ and $minConf$ as the lower bounds of the support and confidence of all sequential rules from a sequence database.

The studies of sequential rule mining generally include two lines, i.e., discovering sequential rules in a single sequence or common to several sequences (Fournier-Viger and Tseng, 2013). This study is interested in the latter. There are a variety of algorithms that have been proposed to perform sequential rule mining in a dataset of many sequences, including the apriori-based algorithm CMRules (Fournier-Viger et al., 2010), rule growth-based algorithm RuleGrowth (Fournier-Viger et al., 2011), and equivalence class-based algorithm ERMiner (Fournier-Viger et al., 2014), etc. Some al-

gorithms proposed to deal with different constraints in sequential rule mining, such as TRuleGrowth (Fournier-Viger et al., 2012c), (Fournier-Viger et al., 2015) for the window size constraints. Considering the efforts of tuning the parameters for rule mining, there is an interest in discovering a certain amount of sequential rules, such as TopSeqRules (Fournier-Viger and Tseng, 2011) for mining the top-k sequential rules and TNS (Fournier-Viger and Tseng, 2013) for mining the non-redundant top-k sequential rules.

All these sequential rule mining algorithms take as input a sequence database and the user-defined thresholds and output the set of frequent sequential rules. Since they utilise different strategies and data structures to search for the sequential rules efficiently, some algorithms are more efficient than others (Fournier-Viger and Tseng, 2011).

2.5 Summary

This chapter presents the overview of the representative studies related to the automated design of search algorithms for COPs, especially on the VRPTW. The aim is to provide the necessary context for understanding the main works in this thesis for the automated design of search algorithms.

Firstly, an introduction to the basic definitions of VRPs and the model of the classic VRPTW is given. The existing heuristic methods for solving VRPs are reviewed, with a particular emphasis on single solution-based (local search-based) meta-heuristics. While significant advances have been made in search methodologies for solving VRPs, developing high-quality heuristics for VRPs remains a challenging task. A review of automated algorithm design is provided, with a focus on the role of learning in different automation tasks. Finally, the machine learning techniques that have been

used in this thesis are summarised.

The literature review of automated algorithm design highlights a series of research gaps that are worth investigating for gaining valuable insights into automated algorithm design with machine learning. The thesis mainly focuses on the automated composition within the line of research in automated algorithm design. The main research gaps (RGs) in automated composition are summarised as follows, supporting the research questions (RQs) of the thesis as identified in Chapter 1.2:

- RG1: the automated algorithm design with elementary algorithmic components in GCOP lacks the support of standardised algorithm design frameworks. Such frameworks can lay the foundation for automated algorithm design by enabling the flexible composition of different algorithmic components. Addressing RG1 is directly linked to RQ1.

- RG2: the limited research in modelling algorithm composition as machine learning tasks presents a research gap in the field of automated algorithm design. Investigating this gap can offer valuable insights into identifying appropriate machine learning techniques in automated algorithm design. RG2 aligns with RQ2, specifically RQ2.b.

- RG3: the insufficient investigation and interpretation of the hidden knowledge captured by machine learning techniques create a research gap in automated algorithm design. Addressing this gap is crucial for gaining a deeper understanding of the decision-making processes involved in algorithm design. RG3 aligns with RQ2.a.

This thesis aims to fill the identified research gaps and answer the proposed RQs through a series of studies and investigations into the use of machine

learning in automated algorithm design. Inspired by the first research gap and RQ1 in Chapter 1.2, a general algorithm framework for the automated design of local search-based algorithms is proposed in Chapter 3, which provides a solid foundation for the automated design of local search algorithms by enabling the composition of different algorithmic components. This framework serves as a basis for subsequent investigations in Chapters 4, 5, and 6, where the algorithm design task is modelled as different machine learning tasks to explore various learning approaches in automated algorithm composition based on other research gaps.

Chapter 3

AutoGCOP: A General Framework for Automated Design of Local Search Algorithms

Contents

3.1 Introduction

The newly established GCOP model (Qu et al., 2020) requires coherent frameworks to assess the performance of the elementary algorithmic components and explore insights on designing effective algorithm compositions with these components. Existing frameworks in the automated composition concern only different subsets and combinations of the algorithmic components in GCOP, and thus cannot provide sufficient support for automated algorithm design based on GCOP.

Based on the GCOP model, this chapter presents a new general AutoGCOP framework to automatically compose elementary algorithmic components, thus supporting the automated design of local search algorithms and systematic investigations on automated composition. Various algorithmic procedures in the literature can be modelled and encapsulated as general procedures within AutoGCOP. These general algorithmic procedures operate flexibly upon the elementary algorithmic components in GCOP, leading to novel local search algorithms which may not be designed manually. In other words, various local search algorithms can be automatically composed with basic components within the general AutoGCOP framework.

Within the consistent AutoGCOP framework, this chapter investigates the performance of these elementary algorithmic components for automated composition, using the VRPTW as the domain application. This is a base to conduct further investigations on effective algorithm compositions. With the elementary components in the GCOP model, it can be observed that the performance of the composed new algorithms is satisfying, confirming the effectiveness of the most basic components in local search algorithms.

In this chapter, Section 3.2 presents the proposed AutoGCOP framework

with the extended GCOP model. Section 3.3 shows the assessment of the basic algorithmic components based on AutoGCOP. Section 3.4 concludes the research in this chapter. This chapter provides the baseline for the following chapters in the thesis which focuses on learning in automated composition within the general AutoGCOP framework.

3.2 The AutoGCOP framework with extended GCOP model

AutoGCOP is a new general framework to support the automatic composition of elementary algorithmic components based on the extended GCOP model, thus supporting the automated design of local search algorithms. Within AutoGCOP, algorithmic procedures are encapsulated as general procedures, allowing various local search algorithms in the literature to be instantiated and novel search algorithms composed automatically.

Section 3.2.1 describes the extended GCOP model. Section 3.2.2 presents the AutoGCOP framework and the instantiation of local search algorithms with AutoGCOP. The differences between AutoGCOP and existing frameworks in the automated composition are discussed in Section 3.2.3.

3.2.1 An Overview of the Extended GCOP Model

In the novel GCOP model defined in (Qu et al., 2020), various search algorithms are broken into a finite set A of elementary algorithmic components $a \in A$. These a serve as the domain of decision variables in GCOP, defining algorithm design itself as a COP.

The solution space of GCOP consists of algorithmic composition c, each of them is a composition of algorithmic components a. Each c represents a new algorithm which can be used to solve optimisation problems p. The objective function of GCOP measures the performance of c when applied to solve the optimisation problem at hand p. This evaluation can be extended to evaluate how well c performs in several runs for solving one or multiple p, as well as the computation time of c for addressing p.

The solution space S of p consists of the direct solutions $s \in S$. The solution s to the optimisation problem p can obtained by the corresponding c, i.e. $c \rightarrow s$, denoting each algorithm composition c for the GCOP maps to solutions s for the optimisation problem. The objective of GCOP is to search for the optimal c^* which produces the optimal s^* for p, so that the objective function of GCOP is optimised. With the optimisation process for solving GCOP, the search algorithms for solving p can be automatically designed.

In GCOP, there are two categories of algorithmic components $a \in A_{1.0}$, i.e. operators $o_i \in A_{1.0\_o}$, and acceptance criteria $a_j \in A_{1.0\_a}$, each with their associated heuristic and parametric settings (Qu et al., 2020). The operators o_i modify values of the decision variables in a solution s_1 to generate a new solution s_2 in the search space of p. The acceptance criteria a_j determine if s_2 is accepted in the search. In recent studies (Yi et al., 2022), algorithmic components in $A_{1.0}$ are extended by including the algorithmic components of evolutionary algorithms, such as crossover operators and selection strategies.

In building the AutoGCOP framework in Section 3.2.2, this study extends the elementary algorithmic components $a \in A_{1.0}$ in the GCOP model with termination criteria in local search algorithms. Based on the widely in-

vestigated meta-heuristics (Blum and Roli, 2003) in the literature, various termination criteria have been modelled and added as basic procedure algorithmic components in the extended GCOP model, i.e. $t_k \in A_t$ as shown in Table 3.1.

Table 3.1: The algorithmic components $t_k \in A_t$ in the extended GCOP model.

A_t	t_k with parameters h, n. h: measure of convergence; n: number of iterations, CPU time or threshold.
$t_{construct}$	Terminate when a complete solution is constructed
$t_{converge}(h)$	Terminate upon the convergence h
$t_{iteration}(n)$	Terminate as the number of iterations reaches n
$t_{time}(n)$	Terminate when the elapsed CPU time reaches n
$t_{threshold}(r, q, n)$	Terminate when r increased by q reaches the threshold n

With the extended algorithmic component set in the GCOP model, the AutoGCOP general framework is built in Section 3.2.2 to support automated algorithm composition. Note that both the acceptance criteria a_j and termination criteria t_k have been built to model elements across different search algorithms into general algorithmic components, and can be used in designing any local search algorithms for any problem p.

3.2.2 The AutoGCOP Framework with Extended GCOP Model

Based on the extended GCOP model, the AutoGCOP framework as shown in Algorithm 1 is proposed to automatically design local search algorithms by composing the basic algorithmic components $o_i \in A_{1.0\_o}$, $a_j \in A_{1.0\_a}$ and $t_k \in A_t$. The underlying idea in building the AutoGCOP framework is to model various local search meta-heuristics by encapsulating their common procedures (operations upon solutions s) as the most basic processes in search algorithms. In particular, the following three most basic processes have been modelled in local search algorithms.

- *Select*(A): select a basic component $o_i \in A_{1.0\_o}$, $a_j \in A_{1.0\_a}$ or $t_k \in A_t$.
- *ApplyOperator*(o_i, s): return a new solution by applying an operator o_i to solution s.
- *ApplyAcceptance*(a_j, s_{new}, s): return solution s_{new} if it is accepted by an acceptance criterion a_j; otherwise, return solution s.

As shown in Algorithm 1, AutoGCOP consists of the Construction procedure and the Improvement procedure. These basic procedures compose the corresponding elementary algorithmic components in the extended GCOP model, i.e. decision variables in GCOP. The Construction procedure constructs a complete solution s for the optimisation problem p by composing the corresponding component sets $t_k \in A_{t_{construct}}$ and $o_i \in A_{o_{construct}}$. Usually $A_{t_{construct}}$ includes $t_{construct}$ in Table 3.1 in the Construction procedure, thus the construction procedure terminates when a complete solution is constructed. The Improvement procedure improves s as an initial solution searching for the best possible solution s_{best}, thus automates the composition of the corresponding component sets $t_k \in A_{t_{improve}} \cup A_{t_{inner}}$, $o_i \in A_{o_{improve}}$ and $a_j \in A_a$.

With the general AutoGCOP framework, various local search algorithms in the literature (Blum and Roli, 2003) can be defined in a unified template by composing specific algorithmic components in the Improvement procedure as shown in Table 3.2. In other words, these meta-heuristics can be seen as specific GCOP solutions composed *manually* by selecting the specific algorithmic components within AutoGCOP.

With the general AutoGCOP framework, a large number of new and unseen local search algorithms can be designed *automatically* by searching for solutions for GCOP, i.e. compositions c of $o_i \in A_{1.0\_o}$, $a_j \in A_{1.0\_a}$ and

$t_k \in A_t$. Different techniques and algorithms can be developed within this general framework to compose algorithmic components from the respective sets in the GCOP model. This can be seen as to automate the process of human experts hand-picking algorithmic components during algorithm design.

Algorithm 1 : The general AutoGCOP framework

Input: p: an optimisation problem,
A_t: a set of termination criteria t_k, including a subset for Construction procedure $A_{t_{construct}}$, a subset for Improvement procedure $A_{t_{improve}}$ and a subset for inner loops of the Improvement procedure $A_{t_{inner}}$,
A_o: a set of operators o_i, including a subset for the Construction procedure $A_{o_{construct}}$ and a subset for the Improvement procedure $A_{o_{improve}}$,
A_a: a set of acceptance criteria a_j,
Output: s_{best}: the best-recorded solution,

procedure CONSTRUCTION
    $s \leftarrow$ An empty solution for p;
    $t_{k_{con}} \leftarrow Select(A_{t_{construct}})$;
    while $t_{k_{con}}$ is not met **do**
        $o_i \leftarrow Select(A_{o_{construct}})$;
        $s \leftarrow ApplyOperator(o_i, s)$;
    end while
end procedure

procedure IMPROVEMENT
    $t_{k_{main}} \leftarrow Select(A_{t_{improve}})$;
    while $t_{k_{main}}$ is not met **do**
        $t_{k_{inner}} \leftarrow Select(A_{t_{inner}})$;
        while $t_{k_{inner}}$ is not met **do**
            $o_i \leftarrow Select(A_{o_{improve}})$;
            $a_j \leftarrow Select(A_a)$;
            $s_{new} \leftarrow ApplyOperator(o_i, s)$;
            $s \leftarrow ApplyAcceptance(a_j, s_{new}, s)$;
            $s_{best} \leftarrow$ Update the best-recorded solution;
        end while
    end while
end procedure

Table 3.2: Instantiation of widely used local search metaheuristics in the literature using different elementary algorithmic components in the Improvement procedure within the unified AutoGCOP framework.

Local search algorithms	Termination criteria, operators and acceptance criteria used in Algorithm 1 ($x \leftarrow y$ denotes use y as x)
Tabu search	$t_{k_{main}} \leftarrow t_{converge}(h)$ in line 11, $t_{k_{inner}} \leftarrow t_{iteration}(1)$ in line 13, $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ in line 15, $a_j \leftarrow a_{tabu}$ in line 17.
Simulated annealing	$t_{k_{main}} \leftarrow t_{converge}(h)$ in line 11, $t_{k_{inner}} \leftarrow t_{iteration}(1)$ in line 13, $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ in line 15, $a_j \leftarrow a_{sa}$ in line 17.
Iterated local search	$t_{k_{main}} \leftarrow t_{converge}(h)$ in line 11, line 13-20 repeat with different $t_{k_{inner}}$, o_i and a_j as follows: firstly, $t_{k_{inner}} \leftarrow t_{iteration}(1)$ in line 13, $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ in line 15, $a_j \leftarrow$ *none* in line 17; secondly, $t_{k_{inner}} \leftarrow t_{converge}(h)$ in line 13, $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ in line 15, $a_j \leftarrow a_{oi}$ in line 17; thirdly, $t_{k_{inner}} \leftarrow t_{iteration}(1)$ in line 13, $o_i \leftarrow$ *none* in line 15, $a_j \leftarrow a_{oi}$ in line 17.
Variable neighborhood search	$t_{k_{main}} \leftarrow t_{converge}(h)$ in line 11, line 13-20 repeat with different $t_{k_{inner}}$, o_i and a_j as follows: firstly, $t_{k_{inner}} \leftarrow t_{iteration}(1)$ in line 13, $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ based on a certain order in line 15, $a_j \leftarrow$ *none* in line 17; secondly, $t_{k_{inner}} \leftarrow t_{converge}(h)$ in line 13, $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ based on a certain order in line 15, $a_j \leftarrow a_{oi}$ in line 17; thirdly, $t_{k_{inner}} \leftarrow t_{iteration}(1)$ in line 13, $o_i \leftarrow$ *none* in line 15, $a_j \leftarrow a_{oi}$ in line 17.

3.2.3 Differences between AutoGCOP and existing frameworks

Generally, the AutoGCOP framework is built on the extended GCOP model, supporting flexible exploration in algorithmic compositions of elementary algorithmic components. Existing frameworks in the automated composition concern only a subset of algorithmic components in GCOP, providing limited scope for automated algorithm design.

The framework that most closely resembles the AutoGCOP framework is the selection hyper-heuristics (SHHs) (Pillay and Qu, 2018), which can be

seen as automatically design of search algorithms by freely composing a set of low-level heuristics chosen by human experts. However, the generality of SHH is limited when compared to AutoGCOP in terms of two aspects as follows:

- The algorithmic components. The SHH selects pre-defined problem-specific low-level heuristics rather than elementary algorithmic components (i.e. basic operators, acceptance criteria and termination criteria). The low-level heuristics can be seen as compound components combining and accumulating the basic components in GCOP. The resulting algorithms of SHH are therefore only a subset of algorithms which can be composed of basic algorithmic components within AutoGCOP.

- The components to manage algorithmic components. The SHH framework usually applies a selection strategy to manage low-level heuristics (Özcan et al., 2008) and uses a pre-defined acceptance criteria. Therefore, the SHH framework is insufficient to manage different types of basic components in GCOP, i.e. basic operators, acceptance criteria and termination criteria.

These above issues not only limit the number of local search algorithms that can be composed with SHH but also involve more human decisions while selecting and configuring the low-level heuristics.

3.3 Effectiveness of algorithmic components on VRPTW

To explore the insights on designing effective algorithm compositions with elementary algorithmic components, this section investigates a subset of the most basic components operators $o_i \in A_{1.0\_o}$ and acceptance criteria $a_j \in A_{1.0\_a}$ in the GCOP model (Qu et al., 2020) and a subset of termination criteria $t_k \in A_t$ in Table 3.1, as shown in Table 3.3, within the AutoGCOP framework. The focus is on the behaviour of operators $o_i \in A_{o_{improve}}$ (line 15, Algorithm 1), with specific components fixed in other procedures.

Table 3.3: The component sets considered in the AutoGCOP framework, i.e. termination criteria $t_k \in A_t$, operators $o_i \in A_o$, and acceptance criteria $a_j \in A_a$.

Component set A_t	Termination criteria t_k in A_t
$A_{t_{construct}}$	$t_{construct}$: terminate when a complete candidate solution is constructed.
$A_{t_{improve}}$	$t_{iteration}(n)$: terminate as the number of iterations reaches n. $t_{time}(n)$: terminate when the elapsed CPU time reaches n.
$A_{t_{inner}}$	$t_{iteration}(1)$: terminate after conducting one iteration.
Component set A_o	Operators o_i in A_o with parameters as defined in GCOP (Qu et al., 2020) h_1: heuristics to choose the customer with the highest proximity to the most recently inserted customer based on distance and time (Walker et al., 2012). h_2: heuristics to choose the next position of the most recently inserted customer (Walker et al., 2012). h_3: random strategy.
$A_{o_{construct}}$	$o_{ins}(1, h_1, h_2)$: insert one customer chosen by h_1 to the position selected by h_2.
$A_{o_{improve}}$	$o^{in}_{xchg}(1, 1, h_3)$: swap two customers chosen by h_3. Selected customers are within one route. $o^{bw}_{xchg}(1, 1, h_3)$: swap two customers chosen by h_3. Selected customers are from different routes. $o^{in}_{ins}(1, h_3, h_3)$: insert one customer chosen by h_3 to other position selected by h_3 within the same route. $o^{bw}_{ins}(1, h_3, h_3)$: insert one customer chosen by h_3 to other position selected by h_3 in a different route. $o_{rr}(10, h_3, h_3)$: remove 10% customers chosen by h_3, and re-assign them using h_3. 2-*opt**: swap the end sections of two routes to generate two new routes (Burke et al., 2010).
Component set A_a	Acceptance criteria a_j in A_a
A_a	a_{naive}: accept all improvements; worse solutions are accepted with a probability of 0.5 (Burke et al., 2010).

Among the most basic $o_i \in A_{o_{improve}}$ adopted in the *Improvement* procedure, 2-*opt** is a problem-specific compound operator designed manually in the literature, which showed to be especially effective for VRPTW (Potvin and Rousseau, 1995). We therefore investigate the performance of o_i grouped into two sets as follows in the experiments:

- $O_{basic} = \{o^{in}_{xchg}, o^{bw}_{xchg}, o^{in}_{ins}, o^{bw}_{ins}, o_{rr}\}$;
- $O_{vrp-basic} = O_{basic} \cup \{2\text{-}opt^*\}$.

To assess the performance of composing the basic algorithmic components o_i in the AutoGCOP framework, the algorithm performance of each o_i in $O_{vrp-basic} \subseteq A_{o_{improve}}$ is compared with a random GCOP method (i.e. RN-GCOP) with $O_{vrp-basic}$ in Section 3.3.1. The same number of evaluations is set as the stopping condition, i.e. $t_{iteration}(n)$ is adopted as $t_{k_{main}}$ in Algorithm 1, for all methods.

The performance of the elementary algorithmic components $O_{basic} \subseteq A_{o_{improve}}$ is then compared against the basic problem-specific compound algorithmic components $O_{vrp-basic} \subseteq A_{o_{improve}}$. The performance of RN-GCOP with O_{basic} is compared against the same method with $O_{vrp-basic}$, which includes a compound operator 2-*opt** based on the solution quality in Section 3.3.2 and the algorithm convergence in Section 3.3.3. The same time is set as the stopping condition, i.e., $t_{time}(n)$ is adopted as $t_{k_{main}}$ in Algorithm 1 for all methods.

The VRPTW concerned in this work considers the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD). A weighted sum objective function is adopted from the literature to evaluate VRPTW solutions s as shown in Equation (5.2), where c

is set to 1000 empirically (Walker et al., 2012).

$$f(s) = c \times NV + TD \tag{3.1}$$

The investigations are conducted on two sets of the widely studied benchmark VRPTW, i.e. the Solomon 100 customers set (Solomon, 1987) and the Homberger 1000 customer set (Gehring and Homberger, 1999) as shown in Table 4.2, covering different instance characteristics. In particular, customers in type-R instances are randomly distributed geographically. In type-C instances, customers are distributed in clusters. Type-RC instances are a mix of them.

Table 3.4: Characteristics of the benchmark VRPTW instances.

Benchmark	Name	Size	Vehicle	Capacity	Type
Solomon	R101	100	25	200	R
Solomon	R201	100	25	1000	R
Solomon	C101	100	25	200	C
Solomon	C206	100	25	700	C
Solomon	RC103	100	25	200	RC
Solomon	RC207	100	25	1000	RC
Homberger	R1-10-1	1000	250	200	R
Homberger	R2-10-6	1000	250	1000	R
Homberger	C1-10-8	1000	250	200	C
Homberger	C2-10-1	1000	250	700	C
Homberger	RC1-10-5	1000	250	200	RC
Homberger	RC2-10-1	1000	250	1000	RC

3.3.1 Performance evaluation on composing algorithmic components

Figure 3.1 and Figure 3.2 show the difference in solution value of each operator in the operator set $O_{vrp-basic}$ against the random RN-GCOP method with $O_{vrp-basic}$ for instances with 100 and 1000 customers, respectively. Comparing the algorithm performance with each of the operators, RN-GCOP achieves better overall performance. Only for instance RC103 and

C1-10-8, o^{bw}_{ins} obtains better results than other methods. The problem-specific compound operator 2-opt^* achieves better performance for instance C2-10-1. The algorithm performance of different operators varies according to problem instances. Among the operators in $O_{vrp-basic}$, the performance of o^{bw}_{ins}, o_{rr} and the 2-opt^* (denoted as $o3$, $o4$ and $o5$, respectively) are relatively better than others.

Figure 3.1: Comparison in the average solution objective value (out of ten runs) between each operator in $O_{vrp-basic}$ against RN-GCOP with $O_{vrp-basic}$ on the 100-customer instances, with error bars representing the standard deviation. $o0$: o^{in}_{xchg}. $o1$: o^{bw}_{xchg}. $o2$: o^{in}_{ins}. $o3$: o^{bw}_{ins}. $o4$: o_{rr}. $o5$: 2-opt^*.

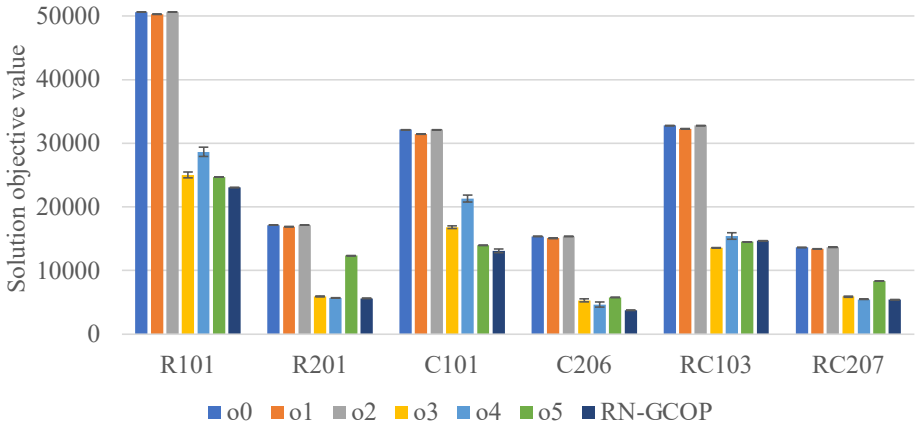
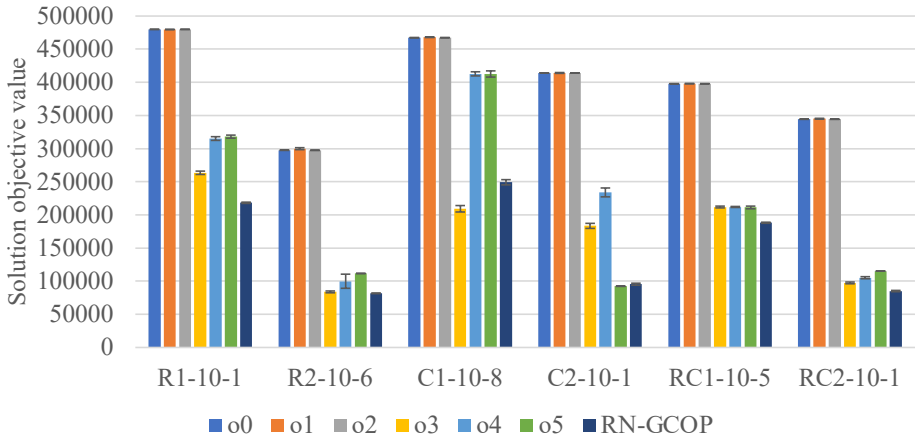


Figure 3.2: Comparison in the average solution objective value (out of ten runs) between each operator in $O_{vrp-basic}$ on the 1000-customer instances, with error bars representing the standard deviation. $o0$: o^{in}_{xchg}. $o1$: o^{bw}_{xchg}. $o2$: o^{in}_{ins}. $o3$: o^{bw}_{ins}. $o4$: o_{rr}. $o5$: 2-opt^*.



In a rigorous analysis on selection hyper-heuristics for function optimisation (Lissovoi et al., 2020b), it is also evidenced that multiple low-level heuristics, which can be seen as compound and specific operators in GCOP, are

necessary to achieve optimal performance. This provides further support for the multiple elementary operators in the AutoGCOP framework.

To further investigate whether the worse-performing operators o^{in}_{xchg}, o^{bw}_{xchg} and o^{in}_{ins} (denoted by $o0$, $o1$ and $o2$) are useful for all problem instances, the performance of RN-GCOP with $O_{vrp-basic}$ is compared against the same method with a subset of $O_{vrp-basic}$ which excludes the worse-performing operators. The same number of evaluations is set as the stopping condition for each method.

Table 3.5 shows the comparison in solution objective value between RN-GCOP with different operator sets. The results support the observations in Figure 3.1 and Figure 3.2 that the performance of different operators can be relatively different according to problem instances. RN-GCOP with the six operators $O_{vrp-basic}$ (denoted as RN6) achieves better results for instance R2-10-6, C2-10-1 and RC2-10-1. This supports that the three worse-performing operators are also useful in some cases.

Table 3.5: Performance comparison of RN-GCOP with different operator sets. Average of objective function values out of 10 runs are presented. The best results are in bold. The results are highlighted with $*$ if one method is significantly better than the other method based on Mann–Whitney–Wilcoxon test at a 95% confidence level. RN3: RN-GCOP with a subset of $O_{vrp-basic}$ (which excludes three worse-performing operators). RN6: RN-GCOP with six operators in $O_{vrp-basic}$.

Instance	R101	R201	C101	C206	RC103	RC207
RN3	**21750.22***	**5533.27***	**11978.00***	**3690.10***	**14192.18***	**5300.13***
RN6	23047.82	5631.21	13103.39	3752.15	14675.39	5432.32
Instance	R1-10-1	R2-10-6	C1-10-8	C2-10-1	RC1-10-5	RC2-10-1
RN3	**214630.66***	85006.18	**245291.17***	106239.80	**185248.76***	90826.98
RN6	218268.34	**81627.15***	249231.61	**95457.94***	188173.73	**85106.16***

In general, it is better to combine operators during the search than to use one operator in all cases. This confirms the effectiveness of the idea of GCOP which utilises algorithmic components with complementary strengths. The difference in the performance of operators requires effective GCOP methods, such as learning, to adapt to different problem instances in the automated composition process.

3.3.2 Performance evaluation on solution quality

Table 3.6 presents the results of the random RN-GCOP method with different operator sets $O_{vrp-basic}$ and O_{basic}. It is obvious that RN-GCOP with $O_{vrp-basic}$ outperforms that with O_{basic} in all instances.

Table 3.6: Solution quality of RN-GCOP with different operator sets $O_{vrp-basic}$ and O_{basic} using the same computational time. RN_basic: RN-GCOP with O_{basic}; RN_vrp: RN-GCOP with $O_{vrp-basic}$. Average of objective function values out of 10 runs are presented.

Instance	R101	R201	C101	C206	RC103	RC207
RN_basic	23451.70	5569.99	15080.49	3698.41	14029.68	5352.06
RN_vrp	**20975.12**	**5507.85**	**11728.95**	**3664.95**	**13532.13**	**5274.04**
Instance	R1-10-1	R2-10-6	C1-10-8	C2-10-1	RC1-10-5	RC2-10-1
RN_basic	277990.26	101657.86	341597.41	191577.04	201094.10	101420.24
RN_vrp	**224601.66**	**84549.46**	**264271.42**	**103884.28**	**183797.42**	**82266.58**

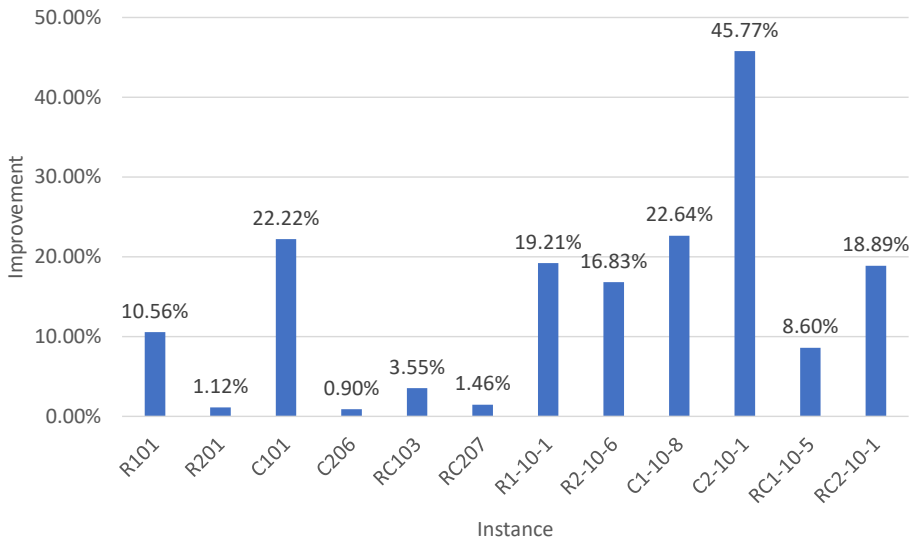
Figure 3.3 shows the improvement from $O_{vrp-basic}$ in RN-GCOP against O_{basic}. Although improvements vary among instances, RN-GCOP with $O_{vrp-basic}$ is better on solving larger instances of 1000 customers. Among the different types of customer distributions, the performance of $O_{vrp-basic}$ is relatively better for solving instances of type-C compared to O_{basic}, although the improvements vary between type-C and type-R. Improvements on type-RC of mixed customer distributions are smaller with $O_{vrp-basic}$, i.e. with the 2-*opt** operator.

3.3.3 Performance evaluation on algorithm convergence

Figure 3.4 and Figure 3.5 present further detailed convergence of RN-GCOP with different operator sets for different types of instances, mapping the final results of type-C, type-R and type-RC instances in Figure 3.3.

It is shown in Figure 3.5 that in general, for type-C instances, the gaps are bigger, followed by type-R instances as reflected in both Figure 3.3

Figure 3.3: The improvement of the random GCOP method with $O_{vrp-basic}$ (denoted as RN_vrp) compared to the same method with O_{basic} (denoted as RN_basic). The amount of Improvements = $(RN\_basic - RN\_vrp)/RN\_basic$.

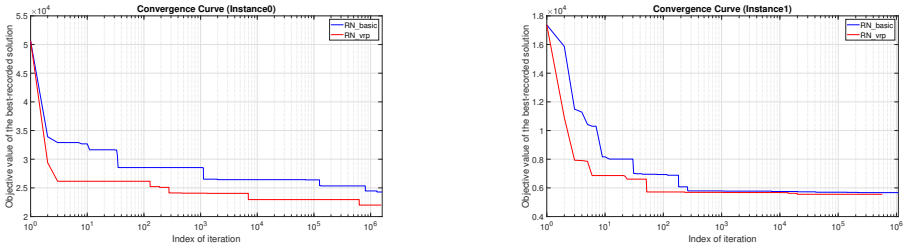


and Figure 3.5. For type-RC instances, the gaps are much smaller. RN-GCOP with $O_{vrp-basic}$ converges faster and outperforms RN-GCOP with O_{basic} throughout the search for all types of instances.

However, RN_vrp is not always better than RN_basic, e.g. Figure 3.4 (b) and (f) suggest RN_basic may reach satisfactory performance with the elementary algorithmic components o_i. As reflected in both Figure 3.3 and Figure 3.4 (b) and (f), the gaps when the algorithms converge are relatively small.

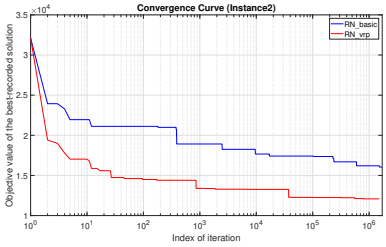
3.3.4 Discussions

In summary, with the most basic algorithmic components, GCOP methods can obtain satisfying performance, reaching performance as good as using human-designed problem-specific compound components on some benchmark instances. Larger instances may require a longer time for the most basic components to reach better results compared to those obtained using specifically designed components by human experts.

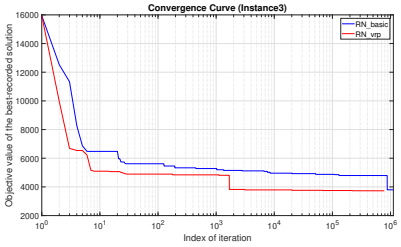


(a) Instance R101

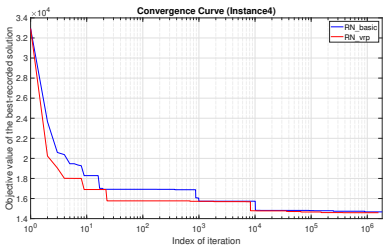
(b) Instance R201



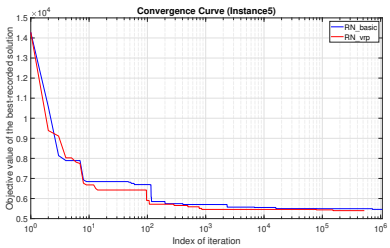
(c) Instance C101



(d) Instance C206



(e) Instance RC103

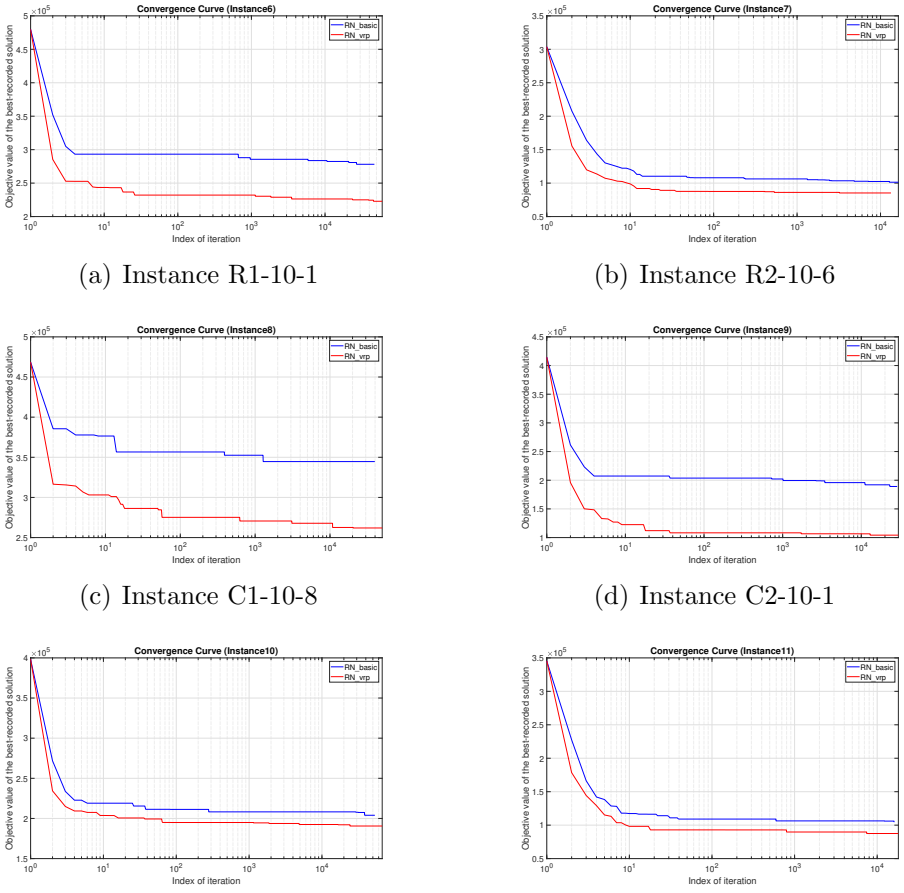


(f) Instance RC207

Figure 3.4: Convergence curves of RN-GCOP with different operator sets on the 100 customer instances given the same time. RN_basic: RN-GCOP with O_{basic}; RN_vrp: RN-GCOP with $O_{vrp-basic}$.

The algorithm performance of each operator is different according to problem instances. For type-RC instances, the improvements in solution quality from problem-specific compound components are relatively small. The most basic components should be given a longer computation time to reach comparable solution quality for type-R instances. For type-C instances, particularly for larger instances, the improvement in solution quality and computation time from problem-specific components are relatively significant.

The problem-specific compound component 2-opt^* swaps the end sections of two routes thus can more likely retain better sections in the solution, moving the search towards promising areas in the solution space. The



(a) Instance R1-10-1

(b) Instance R2-10-6

(c) Instance C1-10-8

(d) Instance C2-10-1

(e) Instance RC1-10-5

(f) Instance RC2-10-1

Figure 3.5: Convergence curves of RN-GCOP with different operator sets on the 1000 customer instances given the same time. RN_basic: RN-GCOP with O_{basic}; RN_vrp: RN-GCOP with $O_{vrp-basic}$.

improvements are due to the domain knowledge used to devise 2-*opt**. The most elementary components make only the most basic moves in the search space, and are applicable to different problems. Their generality needs to be compensated by more computational time to reach solutions obtained by 2-*opt**. In the following performance analysis, $O_{vrp-basic}$ is employed to assess the learning models.

3.4 Conclusions

Based on the GCOP model which defines the problem of algorithm design as a COP, new algorithms can be designed automatically by searching

in a space of compositions of elementary algorithmic components. In this chapter, a general AutoGCOP framework is built to support the automatic composition of elementary algorithmic components based on the general GCOP model, thus designing local search algorithms automatically.

With the encapsulated common processes in local search algorithms, AutoGCOP allows instantiations of existing algorithms designed by manually determining algorithmic components. That is, a large number of existing local search algorithms can be seen as specific solutions of GCOP implemented in AutoGCOP. Furthermore, the AutoGCOP framework underpins the automated design of new and unseen algorithms by using different GCOP methods which compose the algorithmic components automatically.

Based on the AutoGCOP framework, this chapter analyses the performance of the most basic algorithmic components and justifies their effectiveness in designing search algorithms capable of solving complex COPs.

The basic elementary algorithmic components present a satisfying performance given enough computational time, which confirms their effectiveness in automatically designing search algorithms to solve VRPTW instances. In addition, including problem-specific algorithmic components in the basic component set can greatly improve the efficiency of search, reaching a similar solution quality with less computation time, especially for solving larger instances with specific problem structures. This efficiency is gained by the domain expert knowledge used to devise problem-specific algorithmic components, which may not be available in practice. The general AutoGCOP with elementary algorithmic components presents a promising framework across different problems and may be employed by developers of different expertise.

The research presented in this chapter sets the base for future research

directions in the automated design of search algorithms with basic algorithmic components. With the GCOP model which defines a vast design space of algorithms, useful knowledge from different algorithmic compositions may be discovered within AutoGCOP using machine learning. Such new knowledge can be extracted offline from the design space of algorithms and applied to support the design new effective algorithms.

Chapter 4

Online learning to predict algorithmic components for automated algorithm composition

Contents

4.1 Introduction

With the proposed new AutoGCOP framework, different GCOP methods can be developed to optimise the compositions of elementary algorithmic components in the GCOP model to automatically design new algorithms. The AutoGCOP framework underpins the automated design of new and unseen algorithms by using different GCOP methods which compose the algorithmic components automatically.

This chapter investigates the online learning of effective composition of the basic algorithmic components for automated algorithm design based on AutoGCOP. Within AutoGCOP, two learning models, namely Individual Performance (IP) learning and Transition Performance (TP) learning in Section 4.2, have been investigated. Based on probabilistic reasoning, they learn to compose operators $o_i \in A_{o_{improve}}$ intelligently, i.e. learning-based methods $Select(A_{o_{improve}})$ (line 15, Algorithm 1). The widely studied VRPTW is tested to demonstrate the effectiveness of the GCOP methods.

IP learning focuses on the performance of each o_i using a simple probability matrix based on the concept of reinforcement learning. TP learning focuses on the transition between pairs of components. It is based on Markov chain which applies a transition probability matrix, where each o_i is a state. The difference between IP and TP is that TP conducts more detailed learning, where the performance of a specific component can be seen as the sum of the performance of other possible components transferred to this component. The main research aim in this chapter is not to beat the state-of-the-art learning methods and the tailor-made heuristic algorithms by yet another algorithm but to investigate the learning in automatic design of new algorithms by comparing two different learning perspectives based on the general AutoGCOP framework.

In comparison, two simple strategies with $Select(A_{o_{improve}})$ are tested as the baseline GCOP methods to demonstrate the effectiveness of the learning models, including a random strategy (RN) which chooses o_i with equal probability and a random gradient strategy (RG) which chooses a random o_i and continues to apply it as long as it is successful. More specifically, the simple random strategy does not attempt to learn from the behaviour of o_i, while the random gradient strategy can be considered as using a reinforcement learning mechanism with the shortest memory length possible to exploit the currently selected o_i as long as it is successful (Lissovoi et al., 2020b).

In the rest of the chapter, Section 4.2 describes the proposed GCOP methods with learning models within AutoGCOP. Section 4.3 presents the experimental studies addressing the concerned research issues, followed by conclusions in Section 4.4.

4.2 Learning models

The GCOP methods with the proposed learning models in $Select(A_{o_{improve}})$ are named as the IP-GCOP method learning individual $o_i \in A_{o_{improve}}$, and the TP-GCOP method learning the transition between o_i. The purpose of the learning models is to observe the behaviour of $o_i \in A_{o_{improve}}$ in Table 3.3, thus to predict their performance and choose the most appropriate without human involvement to solve the problem adaptively.

Based on the general AutoGCOP framework in Algorithm 1, the GCOP methods only add the learning model M for selecting $o_i \in A_{o_{improve}}$ (line 15, Algorithm 1) and the method to update the learning model $Update()$ after updating s_{best} (line 19, Algorithm 1).

4.2.1 IP-GCOP: learning the individual performance of components

A reinforcement learning method interacts with the environment by trial and error and takes actions given a state based on a policy, aiming to accumulate reward relating to its goal (Sutton et al., 1998). The IP-GCOP method follows a simple reinforcement learning scheme, i.e., a simple reward and penalty scheme, to learn the individual performance of elementary algorithmic components by updating the reward and penalty of each component through sequences of actions (i.e., selection of operators) to adapt to the scenarios of the search environment. A probability matrix is used as a fundamental model to record the individual performance (i.e., the reward and penalty) of elementary algorithmic components, supporting a reinforcement scheme to update the reward and penalty of each component based on its performance during the search. The promising algorithmic components can be selected and applied based on the probability matrix during the search.

The IP-GCOP method uses a simple $2 \times n$ probability matrix M_{IP}, to record the accumulated performance of each individual o_i ($i = 1, ..., n$, $n = |A_{o_{improve}}|$) when a better solution than the current best is found. The two rows record the reward and penalty of each o_i, respectively.

At each iteration of the Improvement procedure, an $o_i \in A_{o_{improve}}$ is selected using M_{IP}. With the learning models, o_i with better-accumulated performance in M are chosen for the next iteration using the roulette wheel selection in the proposed GCOP methods. The accumulated individual performance of each o_i is calculated based on the likelihood (L) of each o_i

achieving improvement in the next iteration, as shown in Equation (4.1):

$$L_i = \frac{M_{IP}[1,i]}{M_{IP}[1,i] + M_{IP}[2,i]} \tag{4.1}$$

The IP-GCOP method uses a roulette wheel selection strategy to select the next o_l with a probability $P_{IP}(l)$ in proportion to $L(i)$, as shown in Equation (4.2):

$$P_{IP}(l) = \frac{L_l}{\sum_{k=1}^{n} L_k} \tag{4.2}$$

At the end of each iteration, the learning model M_{IP} is updated using $Update()$ based on the performance of the selected $o_i \in A_{o_{improve}}$ depending on if o_i leads to a new best solution s_{best} during the search. In M_{IP}, o_i is rewarded by increasing its corresponding value in the first row, i.e. $M_{IP}[1, i]$. Otherwise, o_i is punished by increasing its corresponding value in the second row, i.e. $M_{IP}[2, i]$.

4.2.2 TP-GCOP: learning the transition performance of components

As introduced in Section 2.4.1, a Markov chain models a sequence of states with certain probabilities to transfer between each other (Kemeny and Snell, 1976). A transition probability matrix describes the transition probabilities between states.

The TP-GCOP method is based on a Markov chain model combined with a simple reinforcement scheme. The TP-GCOP method uses a Markov chain model to represent transitions between the elementary algorithmic components and a transition matrix to record the transition probabilities of pairs of algorithmic components statistically. The transition matrix as a fun-

damental model supports the reinforcement scheme to learn the transition performance between algorithmic components and update the transition probabilities of states (i.e., elementary algorithmic components), thus supporting the selection of the promising algorithmic components during the search.

In the TP-GCOP method, a $n \times n$ transition probability matrix M_{TP} is built based on the concept of Markov chain (Kemeny and Snell, 1976), regarding each $o_i \in A_{o_{improve}}$ as a state. The values in M_{TP} record the performance of one o_i transferring to another, learning the transition performance of pairs of o_i and o_l which contributes to a new best solution. Given the current o_i, TP-GCOP uses a roulette wheel selection strategy to select the next o_l with a probability $P_{TP}(l)$ in proportion to $M_{TP}[i, l]$, as shown in Equation (4.3):

$$P_{TP}(l) = \frac{M_{TP}[i, l]}{\sum_{k=1}^{n} M_{TP}[i, k]} \tag{4.3}$$

In M_{TP}, at the end of each iteration, a transition from o_i to o_l which leads to a new better s_{best} is rewarded by increasing the corresponding value of $M_{TP}[i, l]$.

In the proposed IP-GCOP and TP-GCOP, it is important to note that the roulette wheel selection is mainly based on probability. The higher the score, the higher the probability of selection. A variant selection method is to use the ranks of each basic operator, i.e., assign ranks to each operator based on their scores. The higher the ranking, the higher the probability of selection. The rank-based selection is based on relative rankings rather than the absolute magnitude of scores, thus can be more robust to extreme scores. However, it may lose information about the actual magnitude of differences in scores. This study applies the probability-based selection to reflect the actual solution quality differences of basic operators.

4.2.3 Update mechanisms for learning models

The update strategy on M is shown to be an important factor in learning algorithm design in the proposed GCOP methods. In this work, a set of $Update()$ methods as shown in Table 4.1 is tested to analyse their influence on the performance of o_i during the search.

Table 4.1: A set of update strategies in the proposed GCOP methods, i.e. $Update()$ for updating the learning models M.

$Update()$	Strategies to update the corresponding value in M
$Simple()$	By 1
$Linear()$	By the index of the current iteration
$Improve()$	By the amount of improvement/deterioration in the current iteration
$NoImprove()$	By the number of iterations since s_{best} has not been updated
$NoCall()$	For each o_i, by the number of iterations since o_i has been last called

4.2.4 An illustrative example

Given three operators (denoted by o_1, o_2, o_3), assume a search process with six iterations has been conducted within the AutoGCOP framework in Algorithm 1. Each o_i is determined by $Select(A_{o_{improve}})$ in Algorithm 1 with M. $Simple()$ in Table 4.1 is adopted in $Update()$ to update M. Initially, the values in M are all set to 1, thus each o_i is chosen with an equal probability.

With M_{IP}, assume o_2 in the first iteration generates a better solution s_1, thus $M_{IP}[1,\ 2]$ for o_2 is increased by 1. In the next iteration, o_2 is more likely to be selected by applying the roulette wheel on M_{IP}. Assume o_1 is selected, generating a non-improving solution s_2, so $M_{IP}[2,\ 1]$ for o_1 is increased. This is repeated in the following iterations, where o_i with better-accumulated performance as recorded in M_{IP} for finding new best

solutions is more likely to be selected. Other operators, however, have the potential to be selected as well but with a smaller probability. Figure 4.1 presents how M_{IP} is updated during six iterations. With M_{IP}, in the seventh iteration, o_2 with a higher probability in M_{IP} is more likely to be selected.

With M_{TP}, the learning starts from the second iteration. After o_2, assume o_1 is selected using M_{TP}, generating a non-improving solution s_2, so there is no reward to update $M_{TP}[2, 1]$, i.e. the transition from o_2 to o_1. Using the roulette wheel, each o_i has the same probability to be chosen after o_1. Assume o_3 is selected leading to a better solution s_3. $M_{TP}[1, 3]$ is thus increased by 1 to reward the transition from o_1 to o_3. In the following iterations, assume $M_{TP}[3, 2]$ and $M_{TP}[3, 1]$ are updated to reward the transitions from o_3 to o_2 and o_3 to o_1 after selecting o_2, o_3 and o_1. Figure 4.2 presents how M_{TP} is updated during six iterations. Checking the element $M_{TP}[1, 3]$ suggests o_3 has a higher probability to be selected in the next iteration.

Figure 4.1: The M_{IP} updated during six iterations

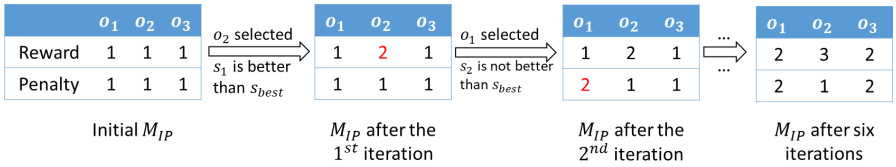
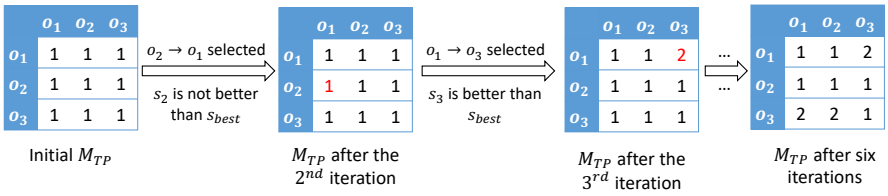


Figure 4.2: The M_{TP} updated during six iterations



4.3 Effectiveness of the learning models on VRPTW

The experimental investigations aim to address the research issue, i.e., analysing the automated composition of o_i in the proposed GCOP methods using the learning models. In Section 4.3.1, the proposed learning-based GCOP methods are compared against the random GCOP methods to demonstrate the effectiveness of the learning models. In evaluating the proposed learning models, the influence of different $Update()$ methods is also analysed in Section 4.3.2. Section 4.3.3 compares the results of the proposed GCOP methods with the published best results by the state-of-the-art methods.

The VRPTW concerned in this work considers the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD). A weighted sum objective function is adopted from the literature to evaluate VRPTW solutions s as shown in Equation (4.4), where c is set to 1000 empirically (Walker et al., 2012).

$$f(s) = c \times NV + TD \tag{4.4}$$

The investigations are conducted on two sets of the widely studied benchmark VRPTW, i.e. the Solomon 100 customers set (Solomon, 1987) and the Homberger 1000 customer set (Gehring and Homberger, 1999) as shown in Table 4.2, covering different instance characteristics. In particular, customers in type-R instances are randomly distributed geographically. In type-C instances, customers are distributed in clusters. RC type instances are a mix of them.

Table 4.2: Characteristics of the benchmark VRPTW instances.

Benchmark	Name	Size	Vehicle	Capacity	Type
Solomon	R101	100	25	200	R
Solomon	R201	100	25	1000	R
Solomon	C101	100	25	200	C
Solomon	C206	100	25	700	C
Solomon	RC103	100	25	200	RC
Solomon	RC207	100	25	1000	RC
Homberger	R1-10-1	1000	250	200	R
Homberger	R2-10-6	1000	250	1000	R
Homberger	C1-10-8	1000	250	200	C
Homberger	C2-10-1	1000	250	700	C
Homberger	RC1-10-5	1000	250	200	RC
Homberger	RC2-10-1	1000	250	1000	RC

4.3.1 Comparisons with random GCOP methods

The best and average results obtained from IP-GCOP and TP-GCOP are compared with those from RN-GCOP and RG-GCOP in Table 4.3. Both learning models are embedded with $Simple()$ update methods. The same number of evaluations is set as the stopping condition, i.e. $t_{iteration}(n)$ is adopted as $t_{k_{main}}$ in Algorithm 1, for all GCOP methods. Similar computational time is observed for these approaches. Overall, TP-GCOP performs better than IP-GCOP, which is better than RN-GCOP and RG-GCOP in most instances. Only for one small instance, RG-GCOP obtains better results than other methods.

To analyse whether the differences observed between IP-GCOP and TP-GCOP are statistically significant, the Lilliefors test is used, showing that they do not always follow a normal distribution. The Mann–Whitney–Wilcoxon test is therefore performed with a 95% confidence level to conduct the pairwise comparisons between the two GCOP methods. Table 4.4 shows that TP-GCOP has a better overall performance compared to IP-GCOP, especially for solving large instances.

The proportion each operator o_i is called in the best algorithm compositions

Table 4.3: Comparison between GCOP methods with different learning (IP-GCOP and TP-GCOP) against the random RN-GCOP and RG-GCOP methods. The best, average (AVG) and standard deviation (SD) of objective function values out of 31 runs are presented.

Instance		R101	R201	C101	C206	RC103	RC207
RN	Best	22941.88	5569.55	12027.70	3709.01	14609.87	5355.37
	AVG	23055.69	5637.15	13030.92	3752.32	14684.06	5437.76
	AVG Time(s)	176	436	175	316	169	452
	SD	**184.22**	27.50	389.22	21.70	**33.31**	30.71
RG	Best	21914.76	5584.46	12158.35	**3704.71**	14622.91	5385.61
	AVG	22946.41	5630.69	12802.19	3756.83	14687.69	5444.48
	AVG Time(s)	175	436	176	317	169	452
	SD	413.39	**24.46**	475.67	23.81	36.05	**22.99**
IP	Best	21792.20	5481.94	**10828.94**	3707.82	14545.99	5325.75
	AVG	22027.43	**5592.05**	11823.04	**3737.37**	14618.26	5384.51
	AVG Time(s)	225	818	261	579	223	809
	SD	369.44	36.86	**378.16**	15.71	35.67	27.02
TP	Best	**20683.49**	**5476.38**	**10828.94**	3708.99	**13523.36**	**5302.97**
	AVG	**21599.58**	5600.57	**11072.53**	3738.01	**14538.34**	**5368.54**
	AVG Time(s)	234	1069	284	598	234	970
	SD	686.53	146.18	454.18	**14.87**	257.26	35.24
Instance		R1-10-1	R2-10-6	C1-10-8	C2-10-1	RC1-10-5	RC2-10-1
RN	Best	214494.95	80507.45	238474.41	92016.80	185565.28	81506.01
	AVG	218091.44	81341.35	248982.83	94935.60	188031.34	84957.79
	AVG Time(s)	225	1044	280	509	2142	799
	SD	**1386.09**	**510.60**	**4126.22**	1710.45	**840.41**	1260.17
RG	Best	192876.50	72191.55	216253.93	79846.80	176046.18	73084.74
	AVG	206357.04	78072.89	235120.04	89442.99	183384.09	79385.30
	AVG Time(s)	224	1026	281	500	210	802
	SD	7543.64	2619.88	7873.45	4407.01	3141.50	3027.83
IP	Best	187732.06	71466.97	184622.87	63594.81	175301.01	71337.83
	AVG	198588.32	74280.80	213053.01	70601.03	179682.39	74856.27
	AVG Time(s)	257	1578	277	610	235	918
	SD	4091.59	1174.01	9732.88	3164.21	2092.27	1699.35
TP	Best	**160065.59**	**62520.70**	**155129.10**	**50841.53**	**152887.15**	**61935.10**
	AVG	**164202.91**	**63939.58**	**173015.06**	**52943.00**	**162072.30**	**63673.53**
	AVG Time(s)	221	1785	231	794	225	1133
	SD	3526.15	1377.38	15841.72	**1090.93**	7183.38	**1215.29**

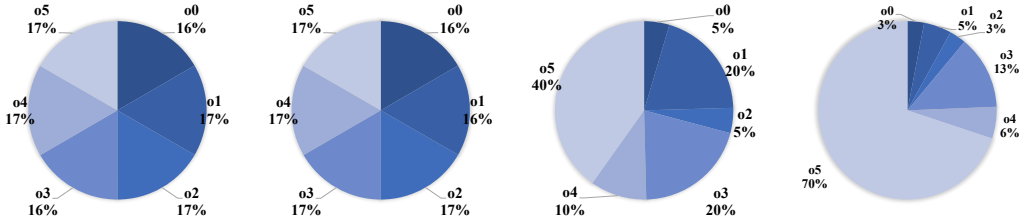
Table 4.4: Performance comparison between IP-GCOP and TP-GCOP using the Mann–Whitney–Wilcoxon test. The comparison between TP $\leftrightarrow$ IP is shown as +, -, or $\sim$ when TP-GCOP is significantly better than, worse than, or statistically equivalent to IP-GCOP, respectively.

Instance	R101	R201	C101	C206	RC103	RC207
TP $\leftrightarrow$ IP	$\sim$	+	$\sim$	+	+	+
Instance	R1-10-1	R2-10-6	C1-10-8	C2-10-1	RC1-10-5	RC2-10-1
TP $\leftrightarrow$ IP	+	+	+	+	+	+

by different GCOP methods are compared on two example instances C101 and C206, in Figure 4.3 and Figure 4.4, respectively. The o_i selected in IP-GCOP and TP-GCOP are highly different, indicating the algorithm compositions, i.e. new algorithms automatically designed with the two learning models, are highly different. Both learning models identify 2-opt^*

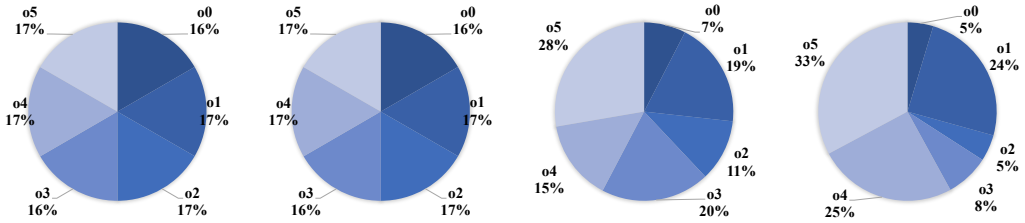
(denoted as o_5) as the most selected component in the best algorithms, although it is automatically selected more often by TP-GCOP compared to IP-GCOP.

Figure 4.3: Proportion of each operator called in the best algorithm compositions obtained by IP-GCOP and TP-GCOP, compared with RN-GCOP and RG-GCOP, for solving instance C101.



(a) RN (b) RG (c) IP (d) TP

Figure 4.4: Proportion of each operator called in the best algorithm compositions obtained by IP-GCOP and TP-GCOP, compared with RN-GCOP and RG-GCOP, for solving instance C206.



(a) RN (b) RG (c) IP (d) TP

4.3.2 Influence of different update methods to the learning models

The influence of different *Update*() methods as specified in Table 4.1 is examined to identify the best intra-domain general method (of the performance across multiple instances from the same domain) for updating the learning model in IP-GCOP and TP-GCOP, respectively. All methods are evaluated for the same number of times, to compare the results out of ten runs.

The results across different instances differ by a large scale and are therefore normalised into a range [0, 1]. The normalisation scheme in (Di Gaspero

and Urli, 2012), as shown in Equation (4.5), is used, where $x(i)$ represent the objective function values calculated using Equation (4.4), and x_{best} and x_{worst} is the best and worst results obtained, respectively. An intra-domain performance score is then calculated as the sum of normalised results over all instances. The normalised scores of the learning models with different update strategies in Table 4.5 show that TP-GCOP with $Simple()$ obtained the best intra-domain performance. For IP-GCOP, the most suitable update method is $Linear()$.

$$x_{norm(i)} = \frac{x(i) - x_{best}}{x_{worst} - x_{best}}. \tag{4.5}$$

Table 4.5: The intra-domain scores for IP-GCOP and TP-GCOP with different update methods. The best results (smallest values) for each method are in bold.

$Update()$	IP-GCOP	TP-GCOP
$Simple()$	28.34	**15.52**
$Improve()$	20.98	52.31
$NoCall()$	23.79	23.31
$Linear()$	**16.14**	43.64
$NoImprove()$	21.68	21.51

Further analysis on the intra-domain performance scores of the update strategies using the Lilliefors test showed that they do not always follow a normal distribution. The Mann–Whitney–Wilcoxon test is therefore performed with a 95% confidence level to conduct the pairwise comparisons between the two GCOP methods with different update strategies statistically.

Table 4.6 shows that IP-GCOP with $Linear()$ has a better overall performance compared to other update methods, especially for solving large instances. For TP-GCOP in Table 4.7, $Simple()$ led to better overall performance, especially for solving small instances.

Table 4.6: Pairwise performance comparison between different *Update*() methods with IP-GCOP using the Mann–Whitney–Wilcoxon test. The comparison between A ↔ B is shown as +, -, or ∼ when A is significantly better than, worse than, or statistically equivalent to B, respectively.

Update()	Instance					
A ↔ B	R101	R201	C101	C206	RC103	RC207
Linear ↔ *Simple*	∼	∼	+	∼	+	∼
Linear ↔ *Improve*	∼	∼	∼	−	∼	∼
Linear ↔ *NoCall*	∼	∼	+	∼	+	∼
Linear ↔ *NoImprove*	∼	∼	∼	∼	∼	∼
Update()	Instance					
A ↔ B	R1-10-1	R2-10-6	C1-10-8	C2-10-1	RC1-10-5	RC2-10-1
Linear ↔ *Simple*	+	+	+	+	+	+
Linear ↔ *Improve*	+	+	+	+	+	+
Linear ↔ *NoCall*	+	+	+	+	+	+
Linear ↔ *NoImprove*	∼	∼	∼	+	∼	∼

Table 4.7: Pairwise performance comparison for the TP-GCOP with different *Update*() methods using the Mann–Whitney–Wilcoxon test.

Update()	Instance					
A ↔ B	R101	R201	C101	C206	RC103	RC207
Simple ↔ *Improve*	+	+	∼	∼	∼	∼
Simple ↔ *NoCall*	+	∼	+	∼	∼	∼
Simple ↔ *Linear*	+	+	∼	∼	∼	∼
Simple ↔ *NoImprove*	∼	∼	∼	∼	∼	+
Update()	Instance					
A ↔ B	R1-10-1	R2-10-6	C1-10-8	C2-10-1	RC1-10-5	RC2-10-1
Simple ↔ *Improve*	+	−	∼	+	∼	+
Simple ↔ *NoCall*	∼	∼	∼	∼	∼	∼
Simple ↔ *Linear*	+	∼	+	+	∼	−
Simple ↔ *NoImprove*	∼	−	∼	−	−	−

4.3.3 Comparisons with the best-known approaches

The best solutions from TP-GCOP with *Simple*() and IP-GCOP with *Linear*() are compared with the published best results produced by various state-of-the-art methods. In the VRP literature, the results of the best-known solutions for VRPTW are usually ranked using a hierarchical objective function, considering the number of vehicles NV as the primary objective and the total travel distance TD as the second objective (Bräysy and Gendreau, 2005b). In this study, a solution with lower NV is considered better than the others with higher NV. For those solutions with the same NV, the lower TD the better.

Table 4.8 shows the results of the proposed GCOP methods compared to the best results reported in the literature by different approaches. It can be seen that the proposed GCOP methods can obtain competitive results in the number of vehicles (i.e. NV) in most small instances, especially on instance C101, where both of the proposed methods obtained the best solution in the literature.

It should be noted that the results from the proposed GCOP methods are obtained by automatically designed new algorithms without any human involvement, while the best results in the literature are obtained by different methods specifically designed for VRPTW. The main research objective in this study is not to beat the tailor-made state-of-the-art approaches by yet another algorithm, but to investigate the learning in automatic design of new algorithms by composing only the most basic algorithmic components within the general AutoGCOP framework.

Table 4.8: Comparison of solution quality between the published best-known results and the best solutions from IP-GCOP and TP-GCOP out of ten runs. NV denotes the number of vehicles. TD denotes the total travel distance. Results that are better than or the same as the best known are in bold.

Instance	Best-known results			IP-GCOP		TP-GCOP	
	NV	TD	Ref.	NV	TD	NV	TD
R101	19	1650.80	(SINTEF, a)	**19**	1653.76	**19**	1654.07
R201	4	1252.37	(SINTEF, a)	**4**	1624.12	**4**	1443.91
C101	10	828.94	(SINTEF, a)	**10**	**828.94**	**10**	**828.94**
C206	3	588.49	(SINTEF, a)	**3**	754.75	**3**	671.54
RC103	11	1261.67	(SINTEF, a)	12	1401.44	12	1399.13
RC207	3	1061.14	(SINTEF, a)	4	1297.91	4	1258.75
R1-10-1	100	53412.11	(SINTEF, b)	101	58815.26	**100**	62024.51
R2-10-6	19	29978.02	(SINTEF, b)	21	41170.48	21	40851.73
C1-10-8	92	42629.91	(SINTEF, b)	103	44867.96	106	47013.85
C2-10-1	30	16879.24	(SINTEF, b)	32	18141.82	33	18149.23
RC1-10-5	90	45069.37	(SINTEF, b)	95	53594.62	97	54482.19
RC2-10-1	20	30276.27	(SINTEF, b)	29	33446.19	28	32627.48

4.4 Concluisons

Based on the AutoGCOP framework in Chapter 3, this chapter investigates online learning in the automated composition of elementary algorithmic components. Two learning models have been studied based on probabilistic reasoning on the behaviour of algorithmic components during the search, comparing the effectiveness of two different learning perspectives.

The proposed GCOP methods based on the AutoGCOP framework have been investigated with effective learning ability to observe the behaviour of algorithmic components. Particularly, compared to the learning model which records and learns from the performance of individual components, the Markov chain-based learning model which adaptively records the transition performance between pairs of basic components shows superior overall performance for problems of different sizes and structures.

As mentioned in Chapter 3, AutoGCOP provides a unified common template to support the investigations on the learning in automated algorithm design. Effective patterns can be potentially extracted from the design space of algorithms, and thus be applied offline to design new effective algorithms. The GCOP methods investigated in this chapter can produce numerous different algorithmic compositions for further investigations in the following chapters.

Chapter 5

Offline learning to predict algorithmic components for automated algorithm composition

Contents

5.1 Introduction

From the aspect of machine learning, the search process of meta-heuristics generates a considerable amount of data, potentially carrying useful knowledge (Karimi-Mamaghan et al., 2022). Some recent selection hyper-heuristics explore the historical data with conventional machine learning techniques for predicting low-level heuristics to use, such as Association Classifier (Thabtah and Cowling, 2008), K-means Classifier (Asta et al., 2013), Decision Trees (Asta and Özcan, 2014), and Neural Networks (Tyasnurita et al., 2015) and (Tyasnurita et al., 2017). These studies support the hypothesis that useful knowledge can be extracted from the historical data of hyper-heuristics, and such knowledge can support the automated composition of low-level heuristics for optimisation.

Algorithmic compositions, however, are sequential data with a sequential correlation between the components. This presents challenges to conventional learning models in learning to predict algorithmic components in the search.

In recent breakthroughs analysing sequential data and text prediction, deep recurrent networks and attention-based neural networks are widely investigated (Wan et al., 2020). Elman network, a variant of recurrent networks, has been used to predict the types of low-level heuristics (Yates and Keedwell, 2017). Aside from (Yates and Keedwell, 2017), sequence classification seems to be an under-explored terrain in the automated design of search algorithms.

The proposed AutoGCOP framework supports more flexible exploration on a larger space of new unseen local search algorithms. With the new AutoGCOP framework, a large amount of data on effective compositions of

basic algorithmic components can be collected consistently, supporting systematic analysis to identify new knowledge towards automated algorithm design.

In this chapter, the automated composition is modelled as a sequence classification task upon standard basic algorithmic components. A large amount of data on effective algorithmic compositions of the basic components are collected within the AutoGCOP framework. For the defined task, an LSTM network and a Transformer network are proposed to learn the knowledge hidden in the algorithmic compositions for forecasting the selection of algorithmic components. The proposed LSTM and Transformer models are investigated against a set of commonly used conventional classifiers, to demonstrate their effectiveness on the defined prediction task. The analysis presents insights into different learning models for learning effective algorithmic compositions.

The contributions of this chapter are threefold as follows:

- Firstly, the prediction of algorithmic components in automated algorithm compositions is formally defined as a sequence prediction task for machine learning, supported by the underlying GCOP model theoretically. With the collected data upon the basic GCOP components as benchmark data, the newly defined machine learning task brings new challenges to the machine learning community and encourages cross-disciplinary collaborations between evolutionary computation and machine learning.
- Secondly, this study confirms the superior performance of LSTM and Transformer in the defined new machine learning task on automated algorithm design. Particularly, this study identifies the superior performance of the LSTM in capturing the knowledge hidden in operator

sequences and supports Transformer in learning to forecast long operator sequences. To the best of our knowledge, it is the first attempt to propose an LSTM model and a Transformer model in learning from the automated compositions for the automated design of search algorithms.

- Thirdly, the analysis of different types of information confirms the effectiveness of problem instance features and search stage in algorithmic compositions. These identified two types of features offer new insights and inform further effective algorithm design.

5.2 The new machine learning task on algorithm composition

Within AutoGCOP, the most effective compositions which produce the best solutions at the end of the search have been collected. Among these effective compositions, the sequences of operators o_i that lead to improvement in solution quality are of the most interest in this research.

We denote a sequence $q \in Q$ of length l as an ordered list of operators o_i applied in the last l iterations, denoted by Equation (5.1).

$$q = \{o_{i-l}, \ldots, o_{i-2}, o_{i-1}\}, l = |q|. \tag{5.1}$$

At each search step, the information of each applied o_i is stored in a vector.

Sequence classification is the task of classifying sequences into existing categories (Xing et al., 2010). Given a finite set of operators A_o as a set of class labels, the task of sequence classification is to build a sequence classi-

fier F, which maps an operator sequence q to a class label $o_i \in A_o$, written as $F : q \rightarrow o_i, o_i \in A_o$.

The defined task aims to explore the hidden sequential relations between the operators within operator compositions. It can be treated as a conventional multi-class classification problem (Crammer and Singer, 2002) by transforming the sequence into a feature vector, solved by conventional classifiers (such as decision trees and neural networks) (Xing et al., 2010). The conventional classification problem, however, treats each o_i in q as an independent feature and analyses them in isolation. The temporal dependencies between operators o_i within the sequences are thus lost in this transformation (Xing et al., 2010). The transformation also increases the dimensions of the input data, leading to more challenges to conventional models.

5.3 Data collection and process for machine learning

To investigate the new sequence classification problem on algorithm composition, the widely studied VRPTW is used as the domain problem. It has been observed that the data collected is extremely imbalanced, thus in-depth analysis has been conducted using re-sampling methods.

5.3.1 The VRPTW problem

The VRPTW concerned in this work considers the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD), as shown in Equation (5.2), where c is set to 1000 empirically

and widely used in the literature (Walker et al., 2012).

$$c \times NV + TD \tag{5.2}$$

The investigations have been conducted on the benchmark Solomon 100 set (Solomon, 1987) as shown in Table 5.1, covering different instance features.

Table 5.1: Features of the benchmark VRPTW instances, including vehicle capacity (VC), scheduling horizon (SH), customer distribution type (DT), service time (ST), time window density (TWD) and width (TWW).

Name	VC	SH	DT	ST	TWD	TWW
C102	200	Short	C	90	75%	61.27
C103	200	Short	C	90	50%	59.90
C104	200	Short	C	90	25%	60.64
C105	200	Short	C	90	100%	121.61
C202	700	Long	C	90	75%	160.00
C203	700	Long	C	90	50%	160.00
C204	700	Long	C	90	25%	160.00
C205	700	Long	C	90	100%	320.00
R102	200	Short	R	10	75%	10.00
R107	200	Short	R	10	50%	30.00
R108	200	Short	R	10	25%	30.00
R109	200	Short	R	10	100%	58.89
R202	1000	Long	R	10	75%	115.23
R203	1000	Long	R	10	50%	117.34
F208	1000	Long	R	10	100%	349.50
R209	1000	Long	R	10	100%	383.27
RC102	200	Short	RC	10	75%	30.00
RC103	200	Short	RC	10	50%	30.00
RC104	200	Short	RC	10	25%	30.00
RC105	200	Short	RC	10	100%	54.33
RC202	1000	Long	RC	10	75%	120.00
RC203	1000	Long	RC	10	50%	120.00
RC204	1000	Long	RC	10	25%	120.00
RC205	1000	Long	RC	10	100%	223.06

5.3.2 Collection of operator sequences with GCOP methods

To explore insights on effective algorithm compositions, the basic operators o_i as shown in Table 5.2 (including the most basic operators and a problem-specific operator 2-opt^*) have been investigated within AutoGCOP. This set of basic operators presents different characteristics for solving VRPTW (Meng and Qu, 2021).

Table 5.2: Features of the operators in operator sequences, including relative neighbourhood size (NS), involved routes of operation (IR) and operation type (OT).

Operator	NS	IR	OT
o_{xchg}^{in}	Small	1-route	Exchange
o_{xchg}^{bw}	Small	2-route	Exchange
o_{ins}^{in}	Small	1-route	Insert
o_{ins}^{bw}	Small	2-route	Insert
o_{rr}	Large	n-route	Ruin-recreate
2-opt^*	Medium	2-route	Exchange

Within AutoGCOP, a Markov Chain-based GCOP method (MC-GCOP), which adaptively learns the transition performance between pairs of basic o_i, presents superior overall performance for composing algorithmic components to solve VRPTW problem instances (Meng and Qu, 2021). The MC-GCOP method is applied to the problem instances in Table 5.1 to produce a collection of effective algorithmic compositions of o_i in Table 5.2. The information in each search iteration recorded includes the current index of iteration, index of the applied o_i, and the objective function value of the current solution, new solution and the best-found solution after using o_i (denoted by $f(s_{current})$, $f(s_{new})$ and $f(s_{best})$, respectively).

The best 10% algorithm compositions according to the solution quality in the current iteration are first collected for each problem instance. The information of the iterations within these elite algorithm compositions which lead to $f(s_{best})$ improvements is then retained in a collection of operator

sequences.

Each operator sequence consists of three types of features as follows:

- Search stage feature: information of the current iteration, stored by the index of iteration of the search ($Iter$).
- Operator features: each operator in the operator sequence is described by its index and features shown in Table 5.2 and whether the solution quality is improved after it has been applied (SC).
- Instance features: problem instance features in Table 5.1.

Let $q = \{o^{in}_{xchg}, o^{in}_{ins}, o_{rr}\}$ be an operator sequence labeled with o^{bw}_{xchg} to be applied at the t_{th} iteration for instance C102. Let $\{f_{t-3}, f_{t-2}, f_{t-1}\}$ be the solution quality change after using $o_i \in q$. Features of this sequence $q \rightarrow o_i$ can be presented as shown in Figure 5.1.

Figure 5.1: An example operator sequence represented by features, including features of the sequence and the corresponding label.

	Iter	Operator	NS	IR	OT	SC	SH	DT	ST	TWD	TWW
$\mathbf{q}$ =	$t-3$	o^{in}_{xchg}	Small	$1-$route	Exchange	f_{i-3}	Short	C	90	75%	61.27
	$t-2$	o^{in}_{ins}	Small	$1-$route	Insert	f_{i-2}	Short	C	90	75%	61.27
	$t-1$	o_{rr}	Large	$n-$route	Ruin − recreate	f_{i-1}	Short	C	90	75%	61.27

$\longrightarrow \mathbf{o_i} = \{o^{bw}_{xchg}\}$

5.3.3 Data imbalance in the operator sequence data set

One distinct observation on the collected data is that the labels o^{bw}_{ins}, o_{rr} and 2-opt^* dominate the categories of the extracted sequences (as shown in Table 5.3), which leads to a seriously imbalanced classification problem (Chawla, 2009). This is due to their larger contributions to better performance on VRPTW (Meng and Qu, 2021). The imbalanced samples present

challenges to learning models where there is a lack of enough data on the minority classes for the learning (Batista et al., 2004).

Table 5.3: The appearance of each o_i in Table 5.2 as the label of the extracted operator sequences.

Operator	o^{in}_{xchg}	o^{bw}_{xchg}	o^{in}_{ins}	o^{bw}_{ins}	o_{rr}	2-opt^*
Appearance	3.8%	0.9%	3.8%	14.8%	18.8%	57.8%

Learning on imbalanced data has been widely investigated in recent research (Zhou, 2013), (López et al., 2013), (Haixiang et al., 2017). Re-sampling techniques, a class of pre-processing methods, showed to be promising in addressing data balance (Zhou, 2013). Re-sampling techniques can be categorised into three groups as follows:

- Under-sampling methods, which select a portion of the majority classes to achieve the distribution balance. The major drawback is that they can discard potentially useful data.
- Over-sampling methods, which replicate some cases or generate new cases from existing ones. This may likely lead to over-fitting or require a clear structure in the imbalanced data to avoid introducing errors.
- Hybrid methods, which combine the above two methods.

This study investigates some of the most representative re-sampling methods, as shown in Table 5.4, for processing the imbalanced operator sequences. The systematic evaluation aims to provide insights for the new machine learning task on the operator sequence data for automated algorithm design.

Table 5.4: Representative re-sampling methods.

Category	Strategy
Under-sampling	Random under-sampling (RU)
	NearMiss (NM) (Mani and Zhang, 2003)
	One-sided selection (OSS) (Kubat et al., 1997)
	Neighborhood Cleaning Rule (NCL) (Laurikkala, 2001)
Over-sampling	Random over-sampling (RO)
	Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002)
	Borderline-SMOTE (BSMOTE) (Han et al., 2005)
	Adaptive Synthetic Sampling (ADASYN) (He et al., 2008)
Hybrid	SMOTEENN (Batista et al., 2004)
	SMOTETomek (Batista et al., 2004)

5.4 Learning from algorithmic components

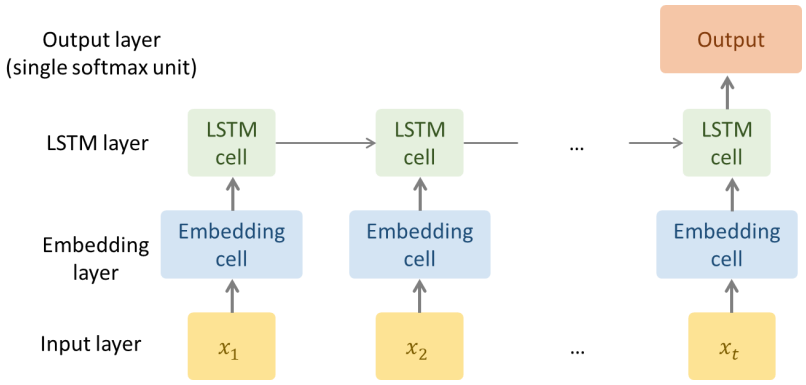
LSTM and Transformers have received great attention in solving sequence classification problems as introduced in Section 2.4.2 and Section 2.4.2. This paper investigates LSTM neural networks and Transformer networks for solving the newly defined sequence classification task. In the context of the automated composition of operators, these two models interpret the operator sequences in different ways: the LSTM focuses more on the sequential relations between operators in sequences, while the transformer learns to put attention to key information and operators in sequences.

5.4.1 Learning operator sequences with LSTM

For the operator sequence classification task defined in Section 5.2, this work proposes a four-layer LSTM as shown in Figure 5.2. The input data are as shown in Figure 2.8, where each slice of the data tube represents one operator sequence and the batch size is the number of sequences in the data.

An embedding layer turns integer representations of operators into dense vectors of fixed size by capturing the underlying structure of the input data and the relation between operators. In the context of neural networks, em-

Figure 5.2: The structure of the proposed LSTM.



bedding is a vector representation of discrete variables learned from the data. It has been widely applied and performs well to represent words as dense vectors in a variety of NLP tasks (Levy and Goldberg, 2014), where neural embedding is more manageable with the lower dimensions of the vectors for high-cardinality variables. The learned vectors with neural embedding explicitly encode many linguistic patterns (Mikolov et al., 2013). The embedding layer in the proposed LSTM model aims to learn vector representations of the operators from the new data on algorithm composition.

In the LSTM layer, the operator embeddings and other sequence features are concatenated to feed into the LSTM cells. The output layer is a dense layer with a SoftMax activation function which outputs the probability of each class.

5.4.2 Learning operator sequences with Transformer

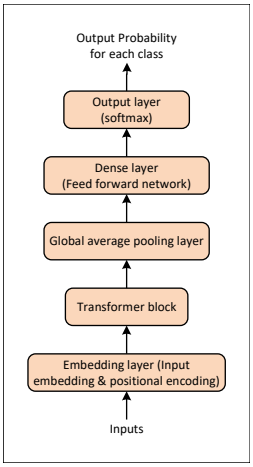
For the operator sequence classification task defined in Section 5.2, this work proposes a sequence-to-one Transformer network as shown in Figure 5.3. The proposed Transformer model can be seen as a simplified version

of the transformer model in Figure 2.9 for sequence-to-sequence learning in (Vaswani et al., 2017). It is created using the Transformer encoder block, followed by three layers for classification rather than the decoder structure.

The input data in Figure 5.3 is in the structure as shown in Figure 2.8. The input operator sequence is fed into an embedding block which includes the Input Embedding layer and the Positional Encoding layer for operator embedding and position embedding. The learned vector and other sequence features are concatenated to feed into the Transformer block. The output of the Transformer block passes through a global average pooling layer, a dropout, and a dense layer with a feedforward network. Finally, an output layer using the softmax function will return the probabilities of the possibilities of the operator sequence such that the sum of the probabilities is 1.

The transformer block in Figure 5.3 implements the multi-head attention mechanism, where the layers are defined following a similar structure in Figure 2.9. It includes the attention layer, a dropout, the normalisation layer, the feed-forward network, a dropout, and batch normalisation.

Figure 5.3: The structure of the proposed Transformer network.



5.5 Findings of classification methods on automated algorithm composition

Intensive analysis has been conducted in the experiments to address three main research issues, 1) assessing the performance of LSTM on the new sequence classification task, 2) assessing the performance of the Transformer model on the new sequence classification task, and 3) identifying and analysing the key features of operator sequences.

In Section 5.5.1, the proposed LSTM model is compared with commonly used conventional classifiers, including Naive Bayes (NB), Logistic Regression (LR), Multi-layer Perceptron (MLP) and Random Forest (RF). The conventional classifiers learn from the operator sequences in a different way. An LSTM model with no embedding layer (denoted as LSTM-basic) is studied to show the effects of embeddings. This analysis thus presents insights into different learning models for the newly defined task on effective algorithmic compositions and reveals knowledge of hidden sequential relations between operators in designing search algorithms. The influence of different re-sampling methods is also investigated in Section 5.5.2.

In Section 5.5.3, the length of operator sequences is investigated to assess the performance of the proposed Transformer network. The Transformer network is compared against the conventional classifiers and LSTM for learning from operator sequences with different lengths. The analysis is to further explore different learning models for the newly defined task on effective algorithmic compositions, particularly the switch from the sequential step-by-step processing of LSTMs to the only-attention-based memory mechanisms of Transformers on processing long operator sequences.

For the third research issue, Section 5.5.4 analyses a set of features describ-

ing operator sequences with the proposed learning models to identify useful information to support automated algorithm design.

Traditionally, accuracy is a widely used performance metric in classification tasks (López et al., 2013), however, not an appropriate measure when learning on imbalanced data (Chawla, 2009). Accuracy is the percentage of the correctly classified examples in all classes. In imbalanced data, the number of instances in the majority class is greatly larger than those in the minority classes. Therefore, a model that simply predicts all instances as the majority class can attain a high accuracy, even if it fails to make any useful predictions for the minority class. This might give the illusion of a well-performing model, but, in reality, the model falls short of providing useful predictions for other classes (Jeni et al., 2013). Area Under Curve (AUC) (Swets, 1988) measures the ability of learning models to distinguish between different classes across various probability thresholds, and it is less influenced by the imbalanced class distribution, especially the size of the majority class (Chawla, 2009). Therefore, it provides a comprehensive view of a model's performance. In this study, AUC is used to evaluate the performance of the models.

5.5.1 Performance assessment for LSTM

The original data set is split into 70% for training and 30% for testing. Since the extracted operator sequence data is highly imbalanced, the data set is processed with re-sampling methods, as shown in Table 5.4, to obtain balanced training data. In all cases, the aim is to try to obtain balanced training data. The testing data is used to evaluate the performance of learning models. Table 5.5 presents the stats of the re-sampled training data and testing data.

Table 5.5: Data size of re-sampled training data and testing data.

Class	Re-sampling methods	Training data size						
		o^{in}_{xchg}	o^{bw}_{xchg}	o^{in}_{ins}	o^{bw}_{ins}	o_{rr}	2-opt^*	Total
	Original	12178	2853	12133	47419	60625	181199	316407
Under	RU	2853	2853	2853	2853	2853	2853	17118
	NM	2853	2853	2853	2853	2853	2853	17118
	OSS	6820	2853	6766	29570	39494	148099	233602
	NCL	12178	2853	12133	47419	60625	115757	250965
Over	RO	181199	181199	181199	181199	181199	181199	1087194
	SMOTE	95680	33232	179983	151639	159881	179983	800398
	BSMOTE	55583	8746	55085	147419	158250	180355	605438
	ADASYN	96286	33251	97522	153951	148195	179965	709170
Hybrid	SMOTEENN	93742	32852	92661	132789	126747	14468	493259
	SMOTETomek	95498	33190	94327	151520	159161	178720	712416
		Testing data size						
	Original	5021	1153	5138	19567	24542	80181	135602

To avoid potential over-fitting and consider computational efficiency, the hyper-parameters in the learning models have been tuned to obtain the best performance in terms of AUC on the training data. Table 5.6 shows the average AUC of 10 runs of the six classifiers on each data set. To compare the overall performance of these six classifiers, their average ranking according to their performance is compared. Overall, LSTM achieves the best performance. RF is worse than LSTM but still superior to other classifiers. Among the other classifiers, LSTM-basic obtain better performance, followed by MLP, NB and LR, respectively.

To investigate further the LSTM and RF and also the contributions of the embedding layer in LSTM, the Mann-Whitney-Wilcoxon test is conducted on the LSTM against LSTM-basic and RF, as shown in Table 5.7. It can be observed that the LSTM with an embedding layer is statistically better, and can learn a proper representation of the operators in the sequences to improve the prediction performance. LSTM also outperforms the conventional classifier, due to its sequence architecture which captures features in modelling long texts.

Table 5.6: The AUC results of different learning models on data sets processed by different re-sampling methods.

Models	RU	NM	OSS	NCL
NB	0.5714	0.5562	0.6014	0.6012
LR	0.5677	**0.5679**	0.6002	0.6009
MLP	0.5714	0.5300	0.6318	0.6401
RF	0.6120	0.5573	0.6407	0.6425
LSTM-basic	0.6088	0.5407	0.6479	0.6492
LSTM	**0.6230**	0.5554	**0.6536**	**0.6552**
Models	RO	SMOTE	BSMOTE	ADASYN
NB	0.5750	0.5966	0.5973	0.5972
LR	0.5726	0.5939	0.5942	0.5942
MLP	0.5869	0.6214	0.6243	0.6208
RF	**0.6179**	0.6256	0.6282	0.6251
LSTM-basic	0.5703	0.6308	0.6354	0.6305
LSTM	0.5956	**0.6347**	**0.6393**	**0.6344**
Models	SMOTEENN	SMOTETomek	Original	
NB	0.5847	0.5967	0.6009	
LR	0.5831	0.5941	0.5994	
MLP	0.6123	0.6206	0.6368	
RF	0.6225	0.6255	0.6436	
LSTM-basic	0.6214	0.6312	0.6510	
LSTM	**0.6245**	**0.6360**	**0.6541**	

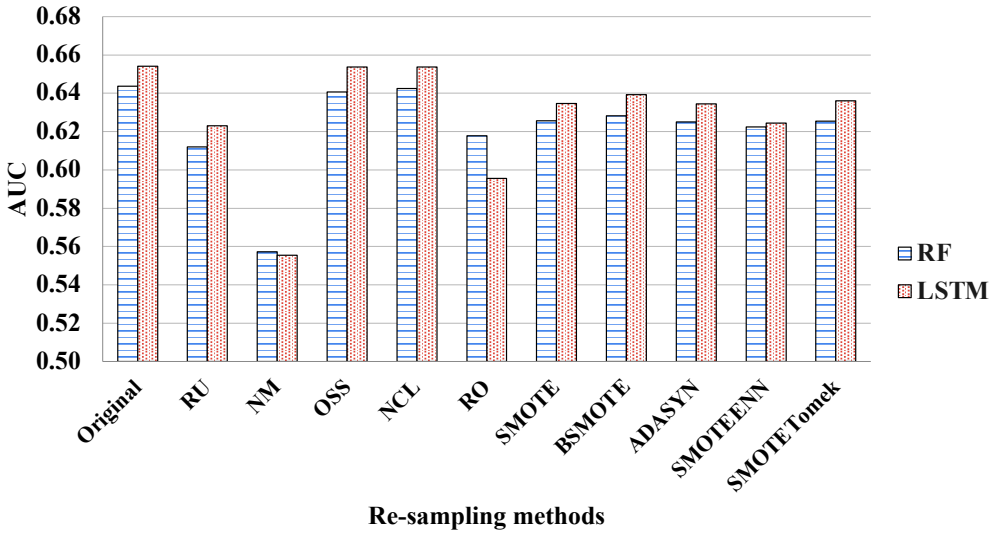
Table 5.7: Pairwise performance comparison on the LSTM with LSTM-basic and RF using the Mann–Whitney–Wilcoxon test. The +, -, or ∼ indicates if the LSTM is significantly better than, worse than, or statistically equivalent to LSTM-basic and RF, respectively.

Re-sampling methods	RU	NM	OSS	NCL
LSTM↔LSTM-basic	+	+	+	+
LSTM↔RF	+	∼	+	+
Re-sampling methods	RO	SMOTE	BSMOTE	ADASYN
LSTM↔LSTM-basic	+	+	+	+
LSTM↔RF	-	+	+	+
Re-sampling methods	SMOTEENN	SMOTETomek	Original	
LSTM↔LSTM-basic	+	+	+	
LSTM↔RF	+	+	+	

5.5.2 Effects of re-sampling methods

To investigate the impact of re-sampling data on the learning models, Figure 5.4 presents the comparisons of AUC for RF and LSTM using different re-sampling methods in each data set. None of the re-sampling methods shows to improve the performance with using the original data set. However, LSTM obtains similar performance on the data sets processed by using OSS and NCL to the original data. Among the selected re-sampling methods, NCL shows to be the best to process the data set for learning

Figure 5.4: Performance comparison of RF and LSTM on the data sets processed with different re-sampling methods.



models to obtain the best overall performance. The over-sampling methods and hybrid re-sampling methods, except RO, have a similar impact on the operator sequence data.

The over-sampling methods introduce new samples to the data to add significantly more information to the minority class examples. However, RO makes exact copies of the minority class cases, thus only introducing redundant information to the data. The syncretization based over-sampling methods (e.g., SMOTE (Chawla et al., 2002)) utilise the inter-correlations rather than temporal inner-correlations of operator sequences. Therefore, the extracted knowledge by the learning models from the newly introduced data fails to reveal the real knowledge of algorithmic composition and thus cannot generalise to the testing data.

Compared with over-sampling methods, under-sampling methods seem more suitable for processing the imbalanced operator sequence data since they would not introduce wrong or useless information to the data. However, after under-sampling methods (or over-sampling), there is a limited amount of data. With only a limited amount of data observed by learning mod-

Table 5.8: Sample size of the training data re-sampled by RU with different imbalance levels.

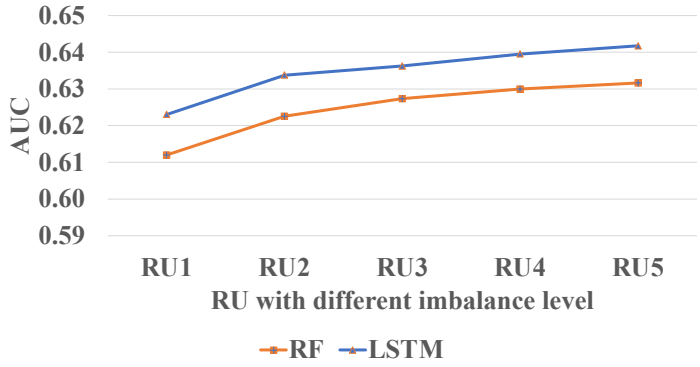
Data set	Training data size							Imbalance level
	o^{in}_{xchg}	o^{bw}_{xchg}	o^{in}_{ins}	o^{bw}_{ins}	o_{rr}	2-opt^*	Total	2-opt^* : o^{bw}_{xchg}
RU1	2853	2853	2853	2853	2853	2853	17118	1 : 1
RU2	5706	2853	5706	5706	5706	5706	31383	2 : 1
RU3	8559	2853	8559	8559	8559	8559	45648	3 : 1
RU4	11412	2853	11412	11412	11412	11412	59913	4 : 1
RU5	12178	2853	12133	14265	14265	14265	69959	5 : 1

els, the extracted knowledge has a limited level of generality in testing. The performance of the NM method is not satisfying, indicating it failed to identify the underlying structure in the high-dimensional operator sequence data. OSS and NCL select a subset of data with data cleaning procedures. However, the resulting data sets are still highly imbalanced, as shown in Table 5.5. Among these under-sampling methods, RU seems more suitable for operator sequences, resulting in a balanced data set.

To successfully apply the under-sampling methods, the RU under-sampling method is further examined on the operator sequence data to strike a balance in the training data while maintaining a suitable information loss. Table 5.8 presents the training data with different imbalance levels after applying RU.

RF and LSTM are then trained with the RU-processed training data, evaluations are shown in Figure 5.5. Unsurprisingly, the learning models obtain better performance on more imbalanced data. It is interesting to observe in Figure 5.5, that from a balanced data (i.e., RU1) to the data with an imbalance level of 2:1 (i.e., RU2 of majority: minority ratio), the learning models obtain significant performance improvement. The improvement of model performance reduces along with increasing imbalance levels. Considering the trade-off between the overall prediction performance and data imbalance level, RU2 is used as a suitable re-sampling method for data pre-processing in this study.

Figure 5.5: The performance change of RF and LSTM on the data sets processed by RU with different data imbalance levels. RU1, RU2, RU3, RU4 and RU5 denotes RU with the ratio of majority class to minority class 1:1, 2:1, 3:1, 4:1 and 5:1, respectively.



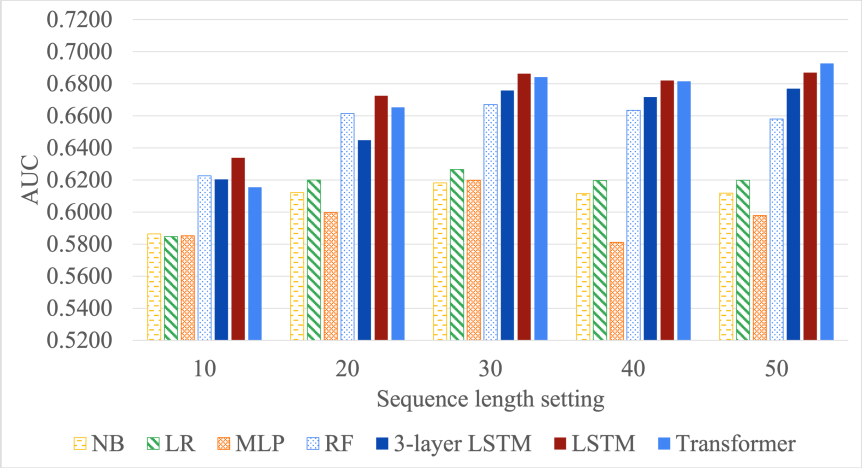
5.5.3 Performance assessment for Transformer

The proposed Transformer model is investigated for handling operator sequences with different length settings (i.e., Y in Figure 2.8) in the range of {10, 20, 30, 40, 50}. The performance of the Transformer model is compared against the selected conventional classifiers and LSTM to reveal the impacts of different operator sequence length settings on different learning mechanisms. The 70% data is processed by RU2 with an imbalance level of 2:1 as the training data.

Figure 5.6 shows the AUC performance of learning models on the operator sequence data set with different length settings. Among the seven classifiers in this study, LSTM achieves the best performance on operator sequences with different length settings overall. In addition, its performance increases with the increase of the length. This further justifies the effectiveness of LSTM.

For short operator sequences, the Transformer model obtains slightly worse performance than LSTM. However, its performance increases with the increase in length. When the sequences consist of 40 operators, the Transformer model obtains a similar performance to LSTM. For longer sequences

Figure 5.6: The comparison of learning models in terms of the AUC performance.



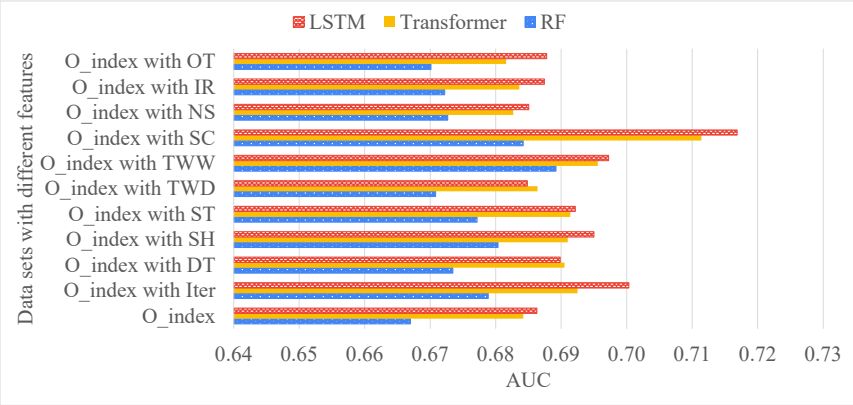
with 50 operators, Transformer obtains the best performance. This confirms that the Transformer model is superior at handling long operator sequences.

The conventional classifiers perform better on the operator sequences of length 30. This suggests their limited learning ability when the number of features increases. Among the conventional classifiers, RF achieves the best performance in different sequence lengths and thus will be used for further investigations. The operator sequence length is set as 30 in Section 5.5.4 for analysing the impacts of different operator sequence features for different machine learning models.

5.5.4 Investigations on features of operator sequences

To investigate the impact of various features in operator sequences in Figure 5.1, RF, LSTM and Transformer are evaluated, representing different types of machine learning models.

Figure 5.7: The AUC performance comparison of learning models on data sets with different features in Figure 5.1.



Effects of each feature

To investigate the impact of each feature, RF, LSTM and Transformer are evaluated on the operator sequences as described in Figure 5.1, compared with the baseline original data (described by the applied operator *O_index* only), results shown in Figure 5.7. All the models perform better with additional features. Overall, LSTM achieves the best performance and Transformer outperforms RF on the same data set. For LSTM and Transformer, the solution quality change SC improves the performance the most, followed by search stage $Iter$. With operator features, the performance of LSTM and Transformer is slightly worse than when it is applied to the original data. RF performs the best on the data with time window width TWW, followed by the SC.

The Mann–Whitney–Wilcoxon test with a 95% confidence level is conducted in pairwise comparisons between the performance on the original data (i.e., *O_index*) and data with one more feature. As shown in Table 5.9, for RF, the additional features to the original data significantly improve its prediction performance. For LSTM and Transformer, some features contribute to their performance improvement, including the solution quality change after the selected operators and the time window width

Table 5.9: Comparison of model performance on the data with only *O_index* and the data with additional features, based on the Mann–Whitney–Wilcoxon test. The +, -, or ∼ indicate if *O_index* with an additional feature is significantly better than, worse than, or statistically equivalent to data with just *O_index*, respectively.

	RF	LSTM	Transformer
O_index with Iter ↔O_index	+	+	∼
O_index with DT ↔O_index	+	∼	+
O_index with SH ↔O_index	+	+	∼
O_index with ST ↔O_index	+	∼	+
O_index with TWD ↔O_index	+	∼	∼
O_index with TWW ↔O_index	+	+	+
O_index with SC ↔O_index	+	+	+
O_index with NS ↔O_index	+	∼	∼
O_index with IR ↔O_index	+	∼	∼
O_index with OT ↔O_index	+	∼	∼

of the problem instances. In addition, the search stage and the instance scheduling horizon are beneficial for improving the prediction of LSTM. The instance features, such as customer distribution type and service time, are useful for the Transformer network. The learning models treat the features differently, supporting the different learning behaviour among the models.

Importance of the features

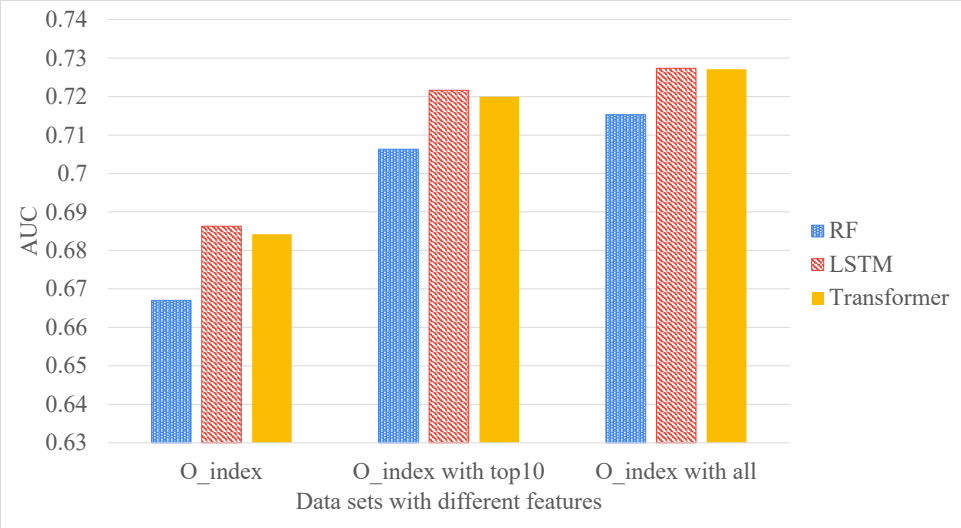
Further analysis is conducted to investigate the importance of each feature for the learning models. Table 5.10 presents the top 10 most important features identified by RF. The search stage feature *Iter* is the most important feature for RF. Similar to the observation on LSTM and Transformer in Table 5.9, the fitness change and time window width are also identified as important features by RF. It is worth noting that RF identifies that the most recently used operators and operator features are important for the prediction. This suggests that its better performance compared with the conventional classifiers may be due to its ability to identify the correlation between operators in the sequences.

To analyse the remaining features, RF, LSTM and Transformer are evalu-

Table 5.10: The top 10 important features found by RF.

Rank	Importance	Feature type	Feature description
1	0.0765	Search stage feature	Index of iteration
2	0.0699	Instance feature	Time window width
3	0.0641	Instance feature	Scheduling horizon
4	0.0359	Operator index	Operator index of the last operator
5	0.0168	Operator feature	Neighbourhood size of the last operator
6	0.0167	Operator index	Operator index of the second last operator
7	0.0154	Fitness	Solution quality change of the last operator
8	0.0133	Operator feature	Operation type of the last operator
9	0.0121	Operator index	Operator index of the third last operator
10	0.0114	Fitness	Solution quality change of the second last operator

Figure 5.8: Comparison of different feature sets using the performance of learning models.



ated on the data set with all features and compared with their performance on the same operator sequence set with only the top 10 important features in Table 5.10. Figure 5.8 shows that the top 10 important features can significantly improve the performance of RF, LSTM and Transformer. However, all models obtain further better performance with all the features. Particularly, the Transformer model obtains similar performance as LSTM with all the features.

The Mann–Whitney–Wilcoxon test with a 95% confidence level on different feature sets is shown in Table 5.11, confirming the effectiveness of the important feature set for both models. RF and Transformer on the data with all features are significantly better than the data with important fea-

Table 5.11: Comparison of model performance on operator sequence data with different feature sets using Mann–Whitney–Wilcoxon test. The +, -, or ∼ indicates that *O_index* with important features is significantly better than, worse than, or statistically equivalent to *O_index* with other features, respectively.

	RF	LSTM	Transformer
O_index with important ↔ *O_index*	+	+	+
O_index with important ↔ *O_index* with all	-	∼	-

tures. This indicates that the less-important features are also useful for both models. However, for LSTM, there is no statistical significance in using all features compared with using only the important features. This indicates that LSTM is able to learn useful knowledge on the operator sequences with only a smaller subset of effective features.

Discussions of the features for operator sequences

Search stage information. The search stage feature is identified as the most important feature by RF. It also contributes to significant improvement for LSTM, as shown in Figure 5.7. This suggests that in different search stages, the effective composition of algorithmic components may follow certain different patterns, thus justifying the use of GCOP methods to adapt to the search stages while flexibly composing algorithmic components. In addition, such patterns in operator sequences may be hard to observe or interpret directly but can be explored by machine learning models, supporting the use of machine learning models in automated algorithm composition.

Instance features. In the literature, the studies which generalise offline learned knowledge to solve unseen VRPTW instances mainly focus on customer type and scheduling horizon. Among the instance features involved in this study, time window widths are identified as more important features for RF, LSTM and Transformer. This suggests a potential difference in the hidden knowledge of algorithmic compositions for problem instances

with distinct time window widths and scheduling horizons. To achieve a higher level of generality in algorithm design for solving VRPTW these two important features should be considered.

Operator sequences. Both RF and LSTM perform better with longer operator sequences, but LSTM is better than RF. However, Transformer is better than all the selected models for learning from sequences consisting of more than 40 operators. To be noted, RF identifies that the recent operators are more important than the previously used operators for prediction. This confirms again the sequential relations between operators in effective algorithmic compositions. In the search, the recently visited neighbourhoods are more important for determining the next neighbourhood.

Operator features. Of the investigated operator-related features, the most important is the solution quality change. The other three operator features, i.e., Neighbourhood Size, Involved Routes and Operation Type, make no contribution to improving the performance of LSTM and Transformer. However, they are useful features for RF. These three features could be seen as important parameters when designing new operators. Investigations of operators with various settings based on these three features may reveal new knowledge for algorithm design in our future research.

5.6 Conclusions and discussions

5.6.1 Conclusions

Various learning methods have been used to automate the algorithm design process in the literature. Within a unified AutoGCOP algorithm design framework which supports the composition of elementary algorithmic

components, this chapter investigated machine learning techniques for automated algorithm design. The aim is to gain insightful knowledge from the effective algorithmic compositions to forecast the behaviour of algorithmic components, thus supporting algorithm design.

The algorithm design problem of determining algorithmic components to use has been defined as a new sequence classification problem. Machine learning methods have been studied to learn a mapping from sub-sequences of algorithmic compositions to the algorithmic component to be applied. This newly defined machine learning task thus supports the automated design of new unseen local search algorithms. With the AutoGCOP framework, a considerable number of new algorithmic compositions can be automatically generated for further investigations for solving VRPTW and other optimisation problems.

In predicting components in algorithmic composition, the proposed LSTM model was compared against commonly used conventional classifiers. LSTM shows to perform better at capturing the sequential relations in algorithmic compositions due to its sequence specialisation of the network structure. To address the issue of highly imbalanced algorithmic composition data, the learning models have been examined using data sets processed with several commonly used re-sampling methods. LSTM shows to be the best machine learning method due to its robust performance on the defined prediction task in forecasting the behaviour of algorithmic components.

In learning from longer algorithmic compositions, the proposed Transformer model is compared to the proposed LSTM model and the commonly used classifiers. Both LSTM and Transformer have increasing prediction performance with longer operator sequences. However, Transformer shows to be more effective at capturing knowledge from longer operator sequences

than LSTM.

Furthermore, various features utilised in automated composition have been analysed with machine learning models. The results confirm the effectiveness of the search stage, operator features and instance features in designing local search algorithms. Certain VRPTW instance features, particularly the scheduling horizon and time window width, have been identified as important features for determining suitable algorithmic components. The search stage, as a general feature, can be a useful indicator when determining suitable algorithmic components in algorithm design for different problem domains. Solution quality change, which represents the performance of the applied algorithmic components, can also be effectively utilised for automated design.

The study in this chapter provides valuable insights into learning knowledge from the data produced by the search process within AutoGCOP. The learned knowledge can be useful in determining suitable operators during automated composition. The findings in this chapter encourage further exploration of the data with new techniques to gain a deeper understanding of the hidden knowledge. This research direction serves as the primary focus of Chapter 6, which conducts a comprehensive analysis of the data gathered with the aim of enhancing the interpretability of learning to support automated composition.

5.6.2 Discussions on the application for practical scenarios

It is important to note that the proposed sequence classification task for automated composition is not limited to VRPTW alone. The investigations

of algorithmic compositions, built upon the general GCOP model (Qu et al., 2020), operate in a distinct space from the solution space of the specific optimisation problem being addressed. Therefore, the methodology applied to learn from algorithmic compositions can be extended and applied to other domains, which can be beneficial for industries and businesses that operate beyond VRPs (Meng and Qu, 2023a). However, the data collection procedure requires careful planning and design in practical scenarios.

Algorithmic components to use. The basic algorithmic components in GCOP (Qu et al., 2020) are not tailored to any specific problems but rather for various problems with minimal development effort. By simply replacing the problem-specific algorithmic components and keeping the same basic components, the underlying methodology of this study can be applied to different problem domains (Meng and Qu, 2023a). Therefore, in real-world scenarios, algorithm designers can use the basic GCOP components with minimal development effort to produce a collection of algorithmic compositions for learning.

Features of algorithmic compositions. The algorithmic composition dataset mainly consists of problem-independent features (i.e., search stage feature and operator features) and problem-dependent features (i.e., instance features). In practical applications, it is advisable to retain the problem-independent features while replacing the problem-dependent features with relevant alternatives. Feature engineering process can be useful in identifying the key features for enhancing the predictive performance of machine learning models.

GCOP methods for producing algorithmic compositions. In this study, learning is conducted on the elite algorithmic compositions produced by the MC-GCOP method which obtains superior performance in auto-

mated composition in Chapter 4. It is important to note that ineffective algorithmic compositions may not yield valuable knowledge, underscoring the significance of the GCOP method for data collection. In practical applications, alternative learning models for automated composition, such as the multi-armed bandit (Almeida et al., 2020) or deep reinforcement learning models (Yi et al., 2022), may hold promise for producing superior algorithmic compositions. To reduce human efforts, algorithm designers can consider the random GCOP method which can effortlessly generate a large pool of algorithmic compositions. Learning from the most elite compositions can also reveal valuable insights for algorithm design (Meng and Qu, 2023b).

Chapter 6

Sequential rule mining on the compositions of algorithmic components

Contents

6.1 Introduction

As stated in Chapter 3, a large amount of data on the search process can be produced within the AutoGCOP framework in the automated algorithm composition process. New knowledge in the data can be potentially discovered to support algorithm design (Qu et al., 2020).

Chapter 4 verifies the effectiveness of Markov Chain-based GCOP method (MC-GCOP) which learns the transitions between algorithmic components in automated composition. Chapter 5 defines algorithm design as a classification problem and investigates different classification methods for acquiring the knowledge hidden in the algorithmic compositions. The LSTM model, which can capture the sequential relations between elements, obtains a better performance for predicting basic operators than conventional classifiers. These studies indicate that there exist sequential relationships between basic operators hidden in the algorithmic compositions which sequential machine learning models could capture.

However, the knowledge captured and embedded in the learning models is implicit, thus hard to analyse further and interpret. Having a good understanding of this hidden knowledge captured in the learning models can further enhance effective algorithm designs for developing new algorithms. This chapter aims to acquire knowledge in a more explicit form from algorithmic compositions, thus enhancing the understanding of how it can assist effective algorithm designs.

Machine learning has been widely used in many real-life scenarios for finding valuable patterns in sequential datasets, such as in bioinformatics (Wang et al., 2007), webpage click-stream analysis (Fournier-Viger et al., 2012b) and market basket analysis (Srikant and Agrawal, 1996). Ignoring

the sequential relationship between events or elements may cause the loss of important patterns in the data (Fournier-Viger et al., 2017). Sequential rule mining techniques (Fournier-Viger et al., 2012a) have been widely investigated for discovering useful subsequences in sequential data, thus supporting decision-making and prediction (Fournier-Viger et al., 2017).

This chapter investigates the data generated in the search process as a new machine learning application, aiming to explore the potentially useful knowledge hidden in the algorithmic compositions and interpret the knowledge to support effective automated algorithm design. In particular, we employ sequential rule mining to extract sequential rules of basic operators from effective algorithmic compositions for solving VRPTW. To obtain insights from the sequential rules, a systematic analysis has been conducted to reveal the sequential relationships between basic operators and the influence of the problem instance types.

Given the extracted sequential rules, this chapter evaluates the effectiveness of using these rules to automate the composition of basic operators. A novel Sequential Rule-based GCOP method (denoted as SeqRuleGCOP) is proposed to support the automated composition within the general AutoGCOP framework. The extracted sequential rules of basic operators are evaluated with SeqRuleGCOP for solving VRPTW. In comparison with other GCOP methods and the Variable Neighbourhood Descent (VND), SeqRuleGCOP shows superior performance, demonstrating its effectiveness in designing new algorithms automatically.

In addition, the frequent sequential rules of basic operators indicate a new attribute of operators, i.e., their impact on optimisation objectives. The basic operators are categorised into three groups based on their impact. Experimental results demonstrate the benefits of using the proposed groups of

basic operators in not only the automated composition within AutoGCOP but also the VND search framework for solving VRPTW.

The contributions of this chapter are threefold as follows:

- Firstly, this study investigated the sequential rules of basic operators for solving VRPTW. To the best of our knowledge, this is the first study on sequential rule mining techniques in the investigation of automated algorithm composition.
- Secondly, this study proposes a new GCOP method to support the use of sequential rules within AutoGCOP, confirming the satisfying performance of the extracted sequential rules in automated composition for solving VRPTW. The comparison between SeqRuleGCOP and VND also supports the superior performance of automated algorithm design over manual algorithm design.
- Thirdly, the identified attribute of operators provides some new insights into algorithm design in the literature. Firstly, it introduces a new categorisation of basic operators to the literature. Using this categorisation of operators in automated composition can improve the optimisation performance for solving VRPTW. On the other hand, this categorisation can lead to new algorithm designs in the metaheuristics search framework. Experiment results verify that changing groups of operators with different optimisation impacts in the VND search framework can achieve better performance for solving VRPTW.

In the rest of this chapter, Section 6.2 presents the data of algorithmic compositions for sequential rule mining, followed by Section 6.3 which describes the sequential rule mining techniques for mining the algorithmic

compositions and the proposed GCOP method. The findings and analysis of the extracted sequential rules of algorithmic components are presented in Section 6.4, followed by Section 6.5 presenting the experimental study on the effectiveness of using the sequential rules and grouping operators in automated composition. Section 6.6 presents the conclusions of this chapter.

6.2 Data of algorithm design for data mining

To investigate the sequential rules of effective algorithmic compositions within AutoGCOP, a simple random GCOP method (RN-GCOP) (Meng and Qu, 2021) is used to randomly compose the basic algorithmic components for solving VRPTW, producing a large database of algorithmic compositions. An elite set of algorithmic compositions of the basic operators are retained for rule mining.

6.2.1 The VRPTW problem

The VRPTW concerned in this work is consistent with the studies in Chapter 4 and Chapter 5, i.e., considering the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD), to evaluate VRPTW solutions s as shown in Equation (6.1), where c is set to 1000 empirically (Walker et al., 2012).

$$f(s) = c \times NV + TD \tag{6.1}$$

The data collection has been conducted on the benchmark Solomon 100 data set as shown in Table 6.1, covering different instance features. The customer distribution types and scheduling horizons are of the most interest in many studies, and thus are the main focus of this study. The Solomon benchmark covers different customer distribution types (i.e., R, C and RC) and scheduling horizons (i.e., short and long). In Type-C instances, customers are located in a number of clusters. Customers of Type-R instances are randomly distributed geographically, while Type-RC instances are a mix of them. The scheduling horizons in Type-1 instances are short, and their vehicle capacities are relatively low (200). Type-2 instances have longer scheduling horizons with larger vehicle capacities (700 or 1000).

Table 6.1: Features of the benchmark VRPTW instances, including vehicle capacity (VC), scheduling horizon (SH), customer distribution type (DT), service time (ST), time window density (TWD) and width (TWW).

Name	VC	SH	DT	ST	TWD	TWW
C102	200	Short	C	90	75%	61.27
C202	700	Long	C	90	75%	160.00
R102	200	Short	R	10	75%	10.00
R202	1000	Long	R	10	75%	115.23
RC102	200	Short	RC	10	75%	30.00
RC202	1000	Long	RC	10	75%	120.00

6.2.2 Collection of operator sequences with GCOP methods

To explore insights on effective algorithm compositions, this study focuses on the most basic operators as shown in Table 6.2 within the general AutoGCOP framework. These basic operators are different in terms of the operation, leading to different performances for solving VRPTW (Meng and Qu, 2021).

To collect algorithmic compositions with AutoGCOP, a GCOP method

Table 6.2: Features of the basic operators for solving VRPTW, including relative neighbourhood size (NS), involved routes of operation (IR) and operation type (OT).

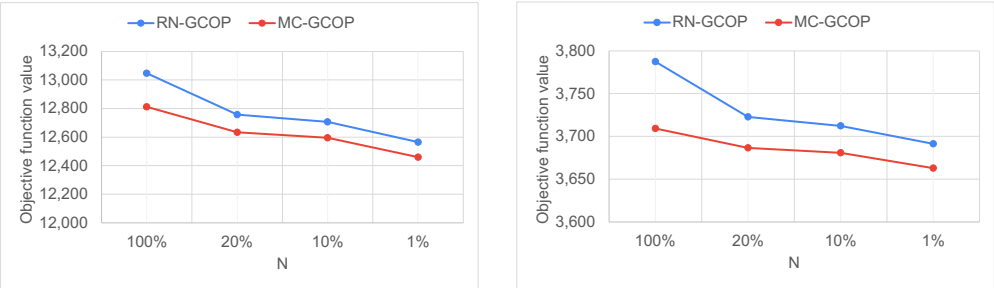
Operator	NS	IR	OT
o_{xchg}^{in}	Small	1-route	Exchange
o_{xchg}^{bw}	Small	2-route	Exchange
o_{ins}^{in}	Small	1-route	Insert
o_{ins}^{bw}	Small	2-route	Insert
o_{rr}	Large	n-route	Ruin-recreate

should be used to compose the basic operators in Table 6.2. In Chapter 5, the Markov Chain-based GCOP method (MC-GCOP) is used for data collection because of its superior overall performance in automated composition for solving VRPTW (Meng and Qu, 2021). MC-GCOP learns the transition performance between pairs of basic components, however, the sequential relations between operators are naturally hidden in the produced algorithmic compositions.

This study utilises a random GCOP method (i.e. RN-GCOP) for data collection. Different from MC-GCOP, RN-GCOP randomly selects operators to apply and compose during optimisation. Based on batch runs on two randomly selected VRPTW instances, Figure 6.1 presents the performance comparison of the RN-GCOP and MC-GCOP in terms of the elite set of algorithmic compositions. These two GCOP methods present similar performances. It would be interesting to see whether there are effective sequential rules of basic operators that can be acquired by using RN-GCOP in automated composition.

During the data collection, each run of RN-GCOP is executed for a sufficiently long duration to guarantee convergence, specifically with $t_{k_{main}} = 3,000,000$ evaluations. Preliminary experiments indicated that the majority of solution quality improvements are found within the initial 200,000 iterations. Therefore, the sequence extraction process is specifically directed towards capturing the hidden knowledge during this critical early phase.

Figure 6.1: Comparison between RN-GCOP and MC-GCOP according to the AVG results of the best N runs out of 1000 runs, for solving C102 and C202.



(a) C102 (b) C202

The information in each search iteration is recorded, including the current index of iteration, the index of the applied operators and the best-found solution after using operators.

The best 1% algorithm compositions according to the best solution's quality found at the end of each run are first collected for each problem instance. The operator sequences within these elite algorithm compositions which lead to improvements to the best-found solution quality are then retained in a collection of short operator sequences with the length of l.

The choice of the sequence length (l) for extracting sequential rules should be guided by the specific objectives of the analysis. However, it can impact the sequential rules that can be discovered from the data. Longer sequences allow the exploration in a boarder context, thus can capture long-term or more complex relationships within the data. However, longer rules might become complex and difficult to interpret, leading to challenges in deriving valuable insights from them. Additionally, some patterns might occur multiple times within a long sequence, but are far from the time step that the solution quality gets improvement, thus might be irrelevant to the decision-making process. In contrast, shorter sequences allow for the capture of short-term relationships between consecutive operators. This allows the detection of more immediate dependencies within the data, which is the

main focus of this study. To determine an appropriate sequence length, this study experiments with different relatively short sequence lengths to compare the resulting patterns.

In the extracted data set, each operator sequence consists of a number of operators. Each operator in the operator sequence is described by its index. The operator sequence data sets of each instance are composed for extracting the general sequential rules of operators across different instances. For instances of the same type, the corresponding operator sequence data sets are also composed for investigating the impacts of instance types on the behaviour of operators.

6.3 Automated algorithm design with sequential rule mining

6.3.1 Sequential rules mining in algorithmic compositions

Sequential rules describe the likelihood between the occurrence of one set of events, items, or itemsets followed by another set of events, items, or itemsets in a sequence of events or transactions (Fournier-Viger et al., 2011). As introduced in Section 2.4.3, a sequential rule $X \rightarrow Y$ describes a sequential relationship between two itemsets X, Y. The interpretation of $X \rightarrow Y$ is if items in X occur in a sequence, the items in Y are likely to occur afterwards in the same sequence (Fournier-Viger et al., 2011). For example, the rule $\{a, b, c\} \rightarrow \{e\}$ occurs in the sequence $< \{a, b\}, \{c\}, \{f\}, \{g\}, \{e\} >$, whereas the rule $\{a, b, f\} \rightarrow \{c\}$ does not, because item c does not occur after f.

Given the processed operator sequences, this study aims to investigate the sequential rules of common basic operators in elite operator sequences. This operator sequence data can be seen as a sequence database, where each operator sequence is an ordered list of operators. A sequential rule thus describes the relationship between two sets of operators.

For example with a set of operators (denoted by $\{o_1, o_2, o_3\}$), assume an operator sequence in the sequence database is $< \{o_2\}, \{o_1\}, \{o_3\} >$. A sequential rule $\{o_1, o_2\} \rightarrow \{o_3\}$ is contained in the sequence, whereas $\{o_3\} \rightarrow \{o_2\}$ does not, because operator o_2 does not occur after o_3. The sequential rule $\{o_1, o_2\} \rightarrow \{o_3\}$ can be interpreted as the occurrence of operators o_1 and o_2 is likely to be followed by o_3. The likelihood of each rule can be measured by sequential support and confidence values (denoted by $\#sup$ and $\#conf$, respectively).

The TNS sequential rule mining algorithm (Fournier-Viger and Tseng, 2013) is applied to the operator sequence collection to search for the top-k non-redundant frequent sequential rules. The implementation of the TNS algorithm adapted a widely used open-source pattern mining platform SPMF (Fournier-Viger et al., 2016). The extracted sequential rules are further investigated to explore the behaviour of basic operators and their effectiveness in algorithm design.

6.3.2 SeqRuleGCOP: automated composition with sequential rules

This study proposes a novel Sequential Rule-based GCOP method (denoted as SeqRuleGCOP) to support using the extracted sequential rules in the automated composition of algorithmic components within AutoGCOP.

SeqRuleGCOP takes an additional input with AutoGCOP, i.e., a set of sequential rules of basic operators. Each sequential rule is an ordered list of operator sets, associated with a support value and a confidence value.

Within AutoGCOP (Algorithm 1), various methods can utilise the procedure $Select(A_{o_{improve}})$ (line 15, Algorithm 1) to compose $o_i \in A_{o_{improve}}$ during the search process. The SeqRuleGCOP method first adds a rule selection procedure in $Select(A_{o_{improve}})$ to select one of the sequential rules with the roulette wheel selection strategy according to the confidence value. Then, the $Select(A_{o_{improve}})$ procedure takes each operator in the sequential rule as the output o_i.

The proposed SeqRuleGCOP method supports the automated design of new and unseen algorithms using the sequential rules of basic operators. It utilises offline learned knowledge in automated algorithm design, allowing the investigation of the effectiveness of the extracted knowledge in the form of sequential rules.

6.4 Findings of sequential rules for algorithm design

This section presents the analysis of the extracted sequential rules of basic operators. Section 6.4.1 presents the impacts of operator sequence length settings on the resulting rules. In Section 6.4.2, the systematic analysis of the extracted sequential rules focuses on the general hidden patterns of operator sequences and the impact of instance features, i.e., the customer distribution types and scheduling horizons, on the behaviour of the operators. The analysis presents the hidden patterns of composing operators for

coping with different types of instances.

6.4.1 Impact of sequence length for rule mining

Sequential rule mining is applied to the operator sequence datasets with the length settings of {5, 10, 20, 30}, aiming to observe how the extracted rules change. Table 6.3 shows that the most frequent sequential rules remain consistent across data sets with different length settings, except for the longest sequence data $l = 30$. This supports that if the sequence is too short ($|l| < 30$), long rules might not occur frequently enough to be retrievable. In addition, a relatively small length setting like $l = 5$ can ensure an adequate level of detail captured.

Table 6.3: Top 10 sequential rules with the #sup and #conf values and the proportional values (indicated by *prop*) on the dataset with different length settings l. Commonly occurred sequential rules in the four sets are in bold.

$l = 5$			$l = 10$		
Rules	#sup (prop)	#conf (prop)	Rules	#sup (prop)	#conf (prop)
$o^{bw}_{xchg} \rightarrow o_{rr}$	**1132 (0.11)**	**0.60 (0.12)**	$o^{in}_{ins} \rightarrow o_{rr}$	**2696 (0.11)**	**0.81 (0.11)**
$o^{in}_{ins} \rightarrow o_{rr}$	**1134 (0.11)**	**0.59 (0.12)**	$o^{bw}_{xchg} \rightarrow o_{rr}$	**2654 (0.10)**	**0.81 (0.11)**
$o^{in}_{xchg} \rightarrow o_{rr}$	**1111 (0.11)**	**0.57 (0.12)**	$o^{in}_{xchg} \rightarrow o_{rr}$	**2669 (0.11)**	**0.81 (0.11)**
$o^{bw}_{xchg} \rightarrow o^{bw}_{ins}$	**1018 (0.10)**	**0.54 (0.11)**	$o^{bw}_{ins} \rightarrow o_{rr}$	**2733 (0.11)**	**0.78 (0.10)**
$o^{in}_{xchg} \rightarrow o^{bw}_{ins}$	**1050 (0.10)**	**0.53 (0.11)**	$o^{in}_{xchg} \rightarrow o^{bw}_{ins}$	**2547 (0.10)**	**0.77 (0.10)**
$o^{in}_{ins} \rightarrow o^{bw}_{ins}$	**990 (0.10)**	**0.51 (0.10)**	$o^{bw}_{xchg} \rightarrow o^{bw}_{ins}$	**2509 (0.10)**	**0.77 (0.10)**
$o^{bw}_{ins} \rightarrow o_{rr}$	**1198 (0.12)**	**0.51 (0.10)**	$o^{in}_{ins} \rightarrow o^{bw}_{ins}$	**2530 (0.10)**	**0.76 (0.10)**
$o_{rr} \rightarrow o^{bw}_{ins}$	**1005 (0.10)**	**0.41 (0.09)**	$o_{rr} \rightarrow o^{bw}_{ins}$	**2530 (0.10)**	**0.72 (0.10)**
$o^{bw}_{ins} \rightarrow o^{in}_{xchg}$	**735 (0.07)**	**0.31 (0.06)**	$o^{bw}_{ins} \rightarrow o^{in}_{xchg}$	**2271 (0.09)**	**0.65 (0.09)**
$o^{bw}_{ins} \rightarrow o^{in}_{ins}$	**715 (0.07)**	**0.30 (0.06)**	$o^{bw}_{ins} \rightarrow o^{in}_{ins}$	**2260 (0.09)**	**0.65 (0.09)**
$l = 20$			$l = 30$		
Rules	#sup (prop)	#conf (prop)	Rules	#sup (prop)	#conf (prop)
$o^{bw}_{xchg} \rightarrow o_{rr}$	**3380 (0.10)**	**0.96 (0.10)**	$o^{in}_{ins} \rightarrow o_{rr}$	**3375 (0.10)**	**1.00 (0.10)**
$o^{in}_{xchg} \rightarrow o_{rr}$	**3386 (0.10)**	**0.96 (0.10)**	$o^{bw}_{ins} \rightarrow o_{rr}$	**3374 (0.10)**	**1.00 (0.10)**
$o^{in}_{ins} \rightarrow o_{rr}$	**3400 (0.10)**	**0.96 (0.10)**	$o^{bw}_{ins} \rightarrow o^{in}_{ins}$	3370 (0.10)	0.99 (0.10)
$o^{in}_{xchg} \rightarrow o^{bw}_{ins}$	**3368 (0.10)**	**0.96 (0.10)**	$o^{bw}_{xchg} \rightarrow o^{bw}_{ins}$	**3366 (0.10)**	**0.99 (0.10)**
$o^{bw}_{ins} \rightarrow o_{rr}$	**3394 (0.10)**	**0.96 (0.10)**	$o^{in}_{ins} \rightarrow o^{bw}_{ins}$	**3368 (0.10)**	**0.99 (0.10)**
$o^{bw}_{xchg} \rightarrow o^{bw}_{ins}$	**3355 (0.10)**	**0.96 (0.10)**	$o_{rr} \rightarrow o^{bw}_{ins}$	**3366 (0.10)**	**0.99 (0.10)**
$o^{in}_{ins} \rightarrow o^{bw}_{ins}$	**3370 (0.10)**	**0.95 (0.10)**	$o^{in}_{xchg} \rightarrow o^{bw}_{ins}$	**3361 (0.10)**	**0.99 (0.10)**
$o_{rr} \rightarrow o^{bw}_{ins}$	**3346 (0.10)**	**0.94 (0.10)**	$o^{in}_{xchg} \rightarrow o_{rr}$	**3361 (0.10)**	**0.99 (0.10)**
$o^{bw}_{ins} \rightarrow o^{in}_{ins}$	**3336 (0.10)**	**0.94 (0.10)**	o^{in}_{ins},$o^{bw}_{ins} \rightarrow o_{rr}$	3362 (0.10)	0.99 (0.10)
$o^{bw}_{ins} \rightarrow o^{in}_{xchg}$	**3297 (0.10)**	**0.93 (0.10)**	$o^{bw}_{xchg} \rightarrow o_{rr}$	**3361 (0.10)**	**0.99 (0.10)**

Although the rules are the same across most data sets, they show different #sup and #conf values. Specifically, the rules extracted from data sets

with longer lengths tend to have higher support and confidence. However, the proportional values of the #sup and #conf values indicate that the concurrency of these rules within the same dataset tends to be similar as the sequence length increases. This might be because some rules occur multiple times within a long sequence. To focus on immediate dependencies between operators, the study uses $l = 5$ for the following investigations.

6.4.2 Analysis of the sequential rules for different types of instances

General sequential rules of basic operators

Table 6.4 presents the top 10 sequential rules of operators in the operator sequence database of all selected instances. The highlighted sequential rules are the top 8 rules that frequently occurred in each type of instance. These commonly occurring frequent sequences can be defined as rules representing general knowledge extracted by data mining in effective algorithmic compositions. However, their reusability and generality need to be investigated.

Table 6.4: Top 10 sequential rules for the data sets of all selected instances. Commonly occurred sequential rules in the three sets are in bold.

Rules	#sup	#conf
$o^{bw}_{xchg} \rightarrow o_{rr}$	**1132**	**0.60**
$o^{in}_{ins} \rightarrow o_{rr}$	**1134**	**0.59**
$o^{in}_{xchg} \rightarrow o_{rr}$	**1111**	**0.57**
$o^{bw}_{xchg} \rightarrow o^{bw}_{ins}$	**1018**	**0.54**
$o^{in}_{xchg} \rightarrow o^{bw}_{ins}$	**1050**	**0.53**
$o^{in}_{ins} \rightarrow o^{bw}_{ins}$	**990**	**0.51**
$o^{bw}_{ins} \rightarrow o_{rr}$	**1198**	**0.51**
$o_{rr} \rightarrow o^{bw}_{ins}$	**1005**	**0.41**
$o^{bw}_{ins} \rightarrow o^{in}_{xchg}$	735	0.31
$o^{bw}_{ins} \rightarrow o^{in}_{ins}$	715	0.30

An interesting finding from these extracted rules is that they are in the form of $X_o \rightarrow Y_o$, while X_o takes $o_i \in \{o^{in}_{xchg}, o^{bw}_{xchg}, o^{in}_{ins}\}$ and Y_o takes $o_i \in \{o^{bw}_{ins}, o_{rr}\}$. This indicates the different common behaviour of the group of operators $o^{in}_{xchg}, o^{bw}_{xchg}, o^{in}_{ins}$ compared with that of o^{bw}_{ins} and o_{rr}.

The investigated basic o_i have different operations to the solutions of the optimisation problem, thus having different impacts on the optimisation objectives. For example, the operations of o^{in}_{xchg} and o^{in}_{ins} are in-route operations, thus making small changes to the value of TD but no impact on NV. In comparison, the operator o_{rr} leads to a relatively bigger impact on both NV and TD. According to the impact on the VRPTW optimisation objectives, the basic $o_i \in A_o$ can be categorised into three sets, as shown in Table 6.5.

Table 6.5: Categorisation of the basic $o_i \in A_o$ based on their impact on VRPTW optimisation objectives, i.e., number of vehicles (NV) and total travel distance (TD).

Operator groups	Operators	Impact on NV	Impact on TD
Type-1 A^1_o	$\{o^{in}_{xchg}, o^{bw}_{xchg}, o^{in}_{ins}\}$	No	Small
Type-2 A^2_o	$\{o^{bw}_{ins}\}$	Small	Small
Type-3 A^3_o	$\{o_{rr}\}$	Large	Large

With the new categorisation of basic o_i in Table 6.5, a general sequential rule of basic o_i can be induced, i.e., $X_o \Rightarrow Y_o$, where X_o takes $o_i \in A^1_o$ and Y_o takes $o_i \in A^2_o \cup A^3_o$. It can be interpreted as the use of the Type-1 operators should be followed by the Type-2 or Type-3 operators. To be more specific, the search in a smaller-impact neighbourhood should be followed by exploration in a larger-impact neighbourhood.

Impact of instance features to sequential rules

Problem instance type plays an important role in analysing the knowledge hidden in the algorithmic compositions as stated in Chapter 5. As described in Section 2, the data of the search process of RN-GCOP was collected on the Solomon benchmark covering different customer distribution types (i.e., R, C and RC type) and scheduling horizons (i.e., Type-1 and Type-2).

For the instances of the same customer distribution type and the same scheduling horizon, the corresponding operator sequence collections are composed for sequential rule mining. Table 6.6 and Table 6.7 present the top 10 sequential rules of operator sequences with the different customer distribution types and different scheduling horizon types, respectively. It can be observed that the top 8 rules occurred commonly for the selected instances of different types, however, with different #sup and #conf. The impact of instance features can be observed from the two uncommon sequential rules.

As shown in Table 6.6, the uncommon two rules of Type-C instances consist of Type-1 operators (i.e., o^{in}_{xchg}, o^{bw}_{xchg}, and o^{in}_{ins}), indicating the frequent use of the Type-1 operators. For Type-R and Type-RC instances, the two rules suggest more usage of Type-2 operators (i.e., o^{bw}_{ins}). Considering the customer distribution impacts, solving Type-R and Type-RC instances requires exploration in a farther neighbourhood to achieve an improvement in solution quality, compared to solving Type-C instances.

In Table 6.7, the remaining two uncommon sequential rules (not in bold) are also related to the impact of instance scheduling horizon types. It can be observed that the Type-1 instances (shorter scheduling horizon) call more rules consisting of Type-1 operators (i.e., o^{in}_{xchg}, o^{bw}_{xchg}, and o^{in}_{ins}).

Table 6.6: Top 10 sequential rules for the data sets of different customer distribution types (i.e., Type-C, Type-R and Type-RC). Commonly occurred sequential rules in the three sets are in bold.

Type-C instances			Type-R instances			Type-RC instances		
Rules	#sup	#con	Rules	#sup	#con	Rules	#sup	#con
$o_{ins}^{in} \rightarrow o_{rr}$	**411**	**0.66**	$o_{xchg}^{bw} \rightarrow o_{ins}^{bw}$	**333**	**0.58**	$o_{xchg}^{bw} \rightarrow o_{rr}$	**416**	**0.59**
$o_{xchg}^{bw} \rightarrow o_{rr}$	**395**	**0.64**	$o_{xchg}^{bw} \rightarrow o_{rr}$	**321**	**0.56**	$o_{ins}^{in} \rightarrow o_{rr}$	**409**	**0.56**
$o_{xchg}^{in} \rightarrow o_{rr}$	**390**	**0.63**	$o_{xchg}^{in} \rightarrow o_{ins}^{bw}$	**335**	**0.55**	$o_{xchg}^{in} \rightarrow o_{rr}$	**395**	**0.54**
$o_{ins}^{bw} \rightarrow o_{rr}$	**413**	**0.56**	$o_{ins}^{in} \rightarrow o_{rr}$	**314**	**0.54**	$o_{xchg}^{in} \rightarrow o_{ins}^{bw}$	**385**	**0.53**
$o_{xchg}^{in} \rightarrow o_{ins}^{bw}$	**330**	**0.53**	$o_{xchg}^{in} \rightarrow o_{rr}$	**326**	**0.53**	$o_{xchg}^{bw} \rightarrow o_{ins}^{bw}$	**364**	**0.52**
$o_{xchg}^{bw} \rightarrow o_{ins}^{bw}$	**321**	**0.52**	$o_{ins}^{in} \rightarrow o_{ins}^{bw}$	**307**	**0.52**	$o_{ins}^{bw} \rightarrow o_{rr}$	**426**	**0.50**
$o_{ins}^{in} \rightarrow o_{ins}^{bw}$	**323**	**0.52**	$o_{ins}^{bw} \rightarrow o_{rr}$	**359**	**0.48**	$o_{ins}^{in} \rightarrow o_{ins}^{bw}$	**360**	**0.50**
$o_{rr} \rightarrow o_{ins}^{bw}$	**314**	**0.39**	$o_{rr} \rightarrow o_{ins}^{bw}$	**311**	**0.44**	$o_{rr} \rightarrow o_{ins}^{bw}$	**380**	**0.42**
$o_{xchg}^{bw} \rightarrow o_{xchg}^{in}$	209	0.34	$o_{ins}^{bw} \rightarrow o_{ins}^{in}$	241	0.32	$o_{ins}^{bw} \rightarrow o_{xchg}^{in}$	292	0.34
$o_{xchg}^{in} \rightarrow o_{ins}^{ins}$	209	0.34	$o_{ins}^{bw} \rightarrow o_{xchg}^{in}$	239	0.32	$o_{ins}^{bw} \rightarrow o_{ins}^{in}$	265	0.31

For instances of a shorter scheduling horizon, each route consists of fewer customers than those of a longer scheduling horizon. Thus, for the former cases, smaller operators can lead to a big solution quality improvement. For solving Type-2 instances (longer scheduling horizon), larger operations are needed to improve the solution, thus requiring more Type-2 operators (i.e., o_{ins}^{bw}).

Table 6.7: Top 10 sequential rules for the data sets of different scheduling horizon types (i.e., Type-1 and Type-2). Commonly occurred sequential rules in the two sets are in bold.

Type-1 instances			Type-2 instances		
Rules	#sup	#con	Rules	#sup	#con
$o_{xchg}^{in} \rightarrow o_{ins}^{bw}$	**471**	**0.63**	$o_{xchg}^{bw} \rightarrow o_{rr}$	**767**	**0.67**
$o_{xchg}^{bw} \rightarrow o_{ins}^{bw}$	**466**	**0.62**	$o_{ins}^{in} \rightarrow o_{rr}$	**770**	**0.66**
$o_{ins}^{in} \rightarrow o_{ins}^{bw}$	**473**	**0.618**	$o_{xchg}^{in} \rightarrow o_{rr}$	**756**	**0.62**
$o_{rr} \rightarrow o_{ins}^{bw}$	**444**	**0.52**	$o_{ins}^{bw} \rightarrow o_{rr}$	**798**	**0.59**
$o_{xchg}^{bw} \rightarrow o_{rr}$	**365**	**0.49**	$o_{xchg}^{bw} \rightarrow o_{ins}^{bw}$	**552**	**0.48**
$o_{xchg}^{in} \rightarrow o_{rr}$	**355**	**0.47**	$o_{xchg}^{in} \rightarrow o_{ins}^{bw}$	**579**	**0.48**
$o_{ins}^{in} \rightarrow o_{rr}$	**364**	**0.47**	$o_{ins}^{in} \rightarrow o_{ins}^{bw}$	**517**	**0.45**
$o_{ins}^{bw} \rightarrow o_{rr}$	**400**	**0.40**	$o_{xchg}^{bw} \rightarrow o_{xchg}^{in}$	433	0.38
$o_{xchg}^{bw} \rightarrow o_{ins}^{in}$	272	0.36	$o_{rr} \rightarrow o_{ins}^{bw}$	**561**	**0.36**
$o_{xchg}^{in} \rightarrow o_{ins}^{in}$	266	0.35=	$o_{ins}^{bw} \rightarrow o_{xchg}^{in}$	443	0.33

Discussions

To be noted, the #sup and #conf values vary for the sequential rules of different types of instances. Thus, it is hard to draw a conclusion about how the customer distribution type and scheduling horizon features impact the utilisation of a sequence of operators. This indicates that the different behaviour of operators for different instances can be too sophisticated to be fully captured by analysing the difference in sequential rules. This supports the use of more advanced learning techniques to capture the knowledge hidden in the high-dimensional algorithm compositions, such as the studies of the advanced machine learning models in Chapter 5.

Interesting findings are gained from the analysis of the commonly occurring sequential rules. These frequent sequential rules of the operator might perform generally well for solving different types of instances, however, to be investigated within AutoGCOP.

It is important to note that the commonly occurring sequential rules reveal a new feature of operators, i.e., the impact of optimisation objectives. When discussing the features of operators, the literature usually focuses on the neighbourhood size (cardinality) (Hansen et al., 2010) and operation types, e.g., either they are intensifiers or diversifiers (Meignan et al., 2010), or the type of mutation, local search or evolutionary operators (Walker et al., 2012). The common sequential rules identify and highlight the impact of operators on the optimisation objectives as an important feature of operators in algorithm design and provide new insights into algorithm design.

Firstly, the identified new attribute of basic operators defines a new way of ranking the basic operators, which can be incorporated into different

algorithm frameworks for new algorithm designs. The idea of ranking the operators has been adopted in some local search algorithms, such as Variable Neighbourhood Descent (VND) (Hansen et al., 2010). However, in VND, the neighbourhoods usually are usually manually specified, i.e., ranked from the smallest to the largest according to the neighbourhood cardinality (Hansen et al., 2010). In this study, the ranking of operators based on their impacts is induced from the knowledge acquired by machine learning, which derives new designs of neighbourhood ordering in VND and other local search algorithms.

In addition, the new attribute of operators provides a new categorisation of operators. The new categorisation of basic operators can be incorporated into the automated composition of algorithmic compositions. In automated composition, composing groups of operators greatly reduce the search space of algorithm design compared to composing individual operators. In the literature, the operators and heuristics are usually categorised based on the operation types (Meignan et al., 2010), (Walker et al., 2012). The proposed new categorisation method highlights the importance of considering the impact of algorithmic components on the optimisation objectives in automated composition. How the optimisation performance can be benefits from the new categorisation method is to be investigated.

6.5 Performance of sequential rules for automated algorithm composition

Based on the analysis of the general sequential rules in Section 6.4, the experimental studies in this section aim to address the following two research issues, i.e., (1) the effectiveness of the frequent sequential rules of the basic

operators in automated composition, and (2) the effectiveness of categorising basic operators based on their impact in automated composition.

To address the first research issue, Section 6.5.1 investigates the performance of the proposed SeqRuleGCOP method for solving VRPTW. In comparison, a simple baseline RN-GCOP method and an online learning-based MC-GCOP method are tested, along with a Variable Neighbourhood Descent (VND) algorithm. RN-GCOP and VND do not learn from the operator sequences. The comparison aims to show the effectiveness of the sequential rules acquired from offline learning in the automated composition of basic operators for problem-solving.

To address the second research issue, Section 6.5.2 compares the performance of composing groups of operators against composing individual operators with different GCOP methods and VND. The experiments aim to demonstrate whether the proposed categorisation of basic operators is useful for improving optimisation effectiveness not only in the AutoGCOP framework but also in the metaheuristics framework.

The experiment studies use VRPTW instances with different instance characteristics. The Solomon 100 customer data in Table 6.1 is used to validate the effectiveness of the extracted rules, while another collection of the Solomon 100 customer data in Table 6.8 is used to test their generality for solving new instances.

This study focuses on the investigations on the basic $o_i \in A_{o_{improve}}$ in Table 6.2 which shown to be crucial in automated algorithm design, the other algorithmic components within AutoGCOP are thus fixed.

The investigated methods are compared based on four performance indicators, i.e., the average fitness value (AVG), the standard deviation of fitness

value (SD), the best fitness value within 10 runs (BEST), and the gap between BEST and the best-known solution in the literature (GAP). In each run, the algorithm terminates after the same number of evaluations, i.e., $t_{iteration}(n)$ is adopted as $t_{k_{main}}$ in Algorithm 1, for all methods.

6.5.1 Effectiveness of sequential rules in automated composition

Comparisons between different GCOP methods

The benchmark RN-GCOP method uses a random selection strategy in the procedure $Select(A_{o_{improve}})$ in Improvement within AutoGCOP. It allows a flexible composition of $o_i \in A_{o_{improve}}$ for problem-solving without utilising any knowledge or mechanism. The MC-GCOP method utilises a Markov Chain model with a reinforcement learning scheme for the selection of $o_i \in A_{o_{improve}}$, following the implementation in Chapter 4. In comparison, both the benchmark GCOP methods and SeqRuleGCOP apply the selected o_i for one search iteration and accept all resulting moves, i.e., a_{all} as a_j and $t_{iteration}(1)$ as $t_{k_{inner}}$ in the Improvement procedure within AutoGCOP (lines 10-22 in Algorithm 1).

Table 6.8 shows the overall better performance of MC-GCOP against SeqRuleGCOP and RN-GCOP on the selected validation instances except in one instance, where SeqRuleGCOP obtains the best BEST results. SeqRuleGCOP performs better against RN-GCOP on the selected VRPTW instances except in two instances, where RN-GCOP achieves better AVG and BEST values.

To analyse whether the differences observed between SeqRuleGCOP and

the other two GCOP methods are statistically significant, the Lilliefors test is used, showing that they do not always follow a normal distribution. The Mann–Whitney–Wilcoxon test is therefore performed with a 95% confidence level to conduct the pairwise comparisons between SeqRuleGCOP and RN-GCOP and SeqRuleGCOP and MC-GCOP methods. Statistical difference in terms of AVG is indicated by * in all the tables of results.

Table 6.8 shows SeqRuleGCOP is statistically better than RN-GCOP in three instances. This supports the extracted frequent sequential rules containing useful patterns for automatically designing a search algorithm within AutoGCOP, particularly for solving Type-R and Type-C instances with statistical significance. The comparison between the MC-GCOP and SeqRuleGCOP indicates online learning is more effective than the offline learned sequential rules at automatically composing the basic operators, mainly for solving Type-C instances with statistical significance.

Similar observations can be reached regarding the comparison between SeqRuleGCOP against RN-GCOP and MC-GCOP on the testing instances in Table 6.9, i.e., SeqRuleGCOP achieves overall better performance compared to RN-GCOP, however, slightly worse than MC-GCOP. This verifies that the extracted frequent sequential rules contain general and useful knowledge which can be used to automatically design search algorithms within AutoGCOP with satisfying optimisation performance.

Table 6.8: Comparison between the SeqRuleGCOP method with the extracted top 10 frequent sequential rules against RN-GCOP and MC-GCOP for effectiveness validation. The best and second-best results are in bold and italics, respectively.

Instances	Best-known solutions	RN-GCOP				SeqRuleGCOP				MC-GCOP			
	in the literature	AVG	SD	BEST	GAP	AVG	SD	BEST	GAP	AVG	SD	BEST	GAP
C102	10828.94(Rochat and Taillard, 1995)	13126.54	269.49	12875.06	0.189	*12899.69**	*165.65*	**12540.88**	**0.158**	**12807.32***	**68.71**	*12709.09*	*0.174*
C202	3591.56(Rochat and Taillard, 1995)	3787.12	49.30	3727.07	0.038	*3745.93**	*27.48*	*3707.10*	*0.032*	**3711.08***	**17.69**	**3691.55**	**0.028**
R102	18486.12(Rochat and Taillard, 1995)	21033.67	**39.80**	20976.51	0.135	*20658.85**	*464.99*	*20040.94*	*0.084*	**20381.03**	549.04	**19869.93**	**0.075**
R202	4191.7(Rousseau et al., 2002)	*5542.48*	28.22	**5502.88**	**0.313**	5544.64	**19.48**	5509.80	0.314	**5531.00**	*19.69*	*5504.44*	*0.313*
RC102	13554.75(Taillard et al., 1997)	17810.05	**401.25**	*16925.97*	*0.249*	*17688.55*	*449.91*	17070.83	0.259	**17487.70**	550.34	**16869.10**	**0.245**
RC202	4365.65(Czech and Czarnas, 2002)	5754.53	**22.44**	5723.58	0.311	*5730.50*	37.47	*5684.91*	*0.302*	**5727.94**	*33.85*	**5654.04**	**0.295**

Table 6.9: Comparison between the SeqRuleGCOP method with the extracted top 10 frequent sequential rules against RN-GCOP and MC-GCOP for generality evaluation. The best and second-best results are in bold and italics, respectively.

Instances	Best-known solutions	RN-GCOP				SeqRuleGCOP				MC-GCOP			
	in the literature	AVG	SD	BEST	GAP	AVG	SD	BEST	GAP	AVG	SD	BEST	GAP
C103	10,828.06(Rochat and Taillard, 1995)	12,364.31	416.60	*11,738.51*	*0.084*	*12,042.12**	*229.02*	11,745.85	0.085	**11897.11***	**126.47**	**11608.78**	**0.072**
C203	3,591.17(Rochat and Taillard, 1995)	4,502.51	514.51	3,881.98	0.081	*4,296.84*	512.80	*3,849.52*	*0.072*	**3878.74***	**41.94**	**3819.53**	**0.064**
R107	11,104.66(Shaw, 1997)	14,564.69	*27.27*	14,520.74	0.308	*14,544.92*	**23.53**	*14,518.62*	*0.307*	**14416.76***	310.58	**13538.13**	**0.219**
R208	2,726.82(Mester et al., 2007)	4,087.51	*16.36*	4,055.09	0.487	*4,074.72*	21.87	**4,017.37**	**0.473**	**4069.40**	***17.76***	*4044.94*	0.483
RC103	12,261.67(Shaw, 1998)	*14,881.08*	*297.94*	*14,691.40*	0.198	15,216.38	*519.09*	**14,665.19**	**0.196**	**14768.24**	35.22	14699.95	0.199
RC203	4,049.62(Czech and Czarnas, 2002)	4,784.47	355.59	*4,539.13*	*0.121*	**4,595.81**	**45.08**	**4,523.08**	**0.117**	*4620.33*	49.00	4563.58	0.127

Comparison with VND

To further analyse the effectiveness of the extracted sequential rules for problem-solving, the proposed SeqRuleGCOP is further compared against the widely studied VND method. VND is the simplest and effective variant in the family of Variable Neighborhood Search (VNS) (Hansen et al., 2019). It is based on the systematic change in a set of neighbourhood structures during the search process. In VND, the change of neighbourhoods is performed in a deterministic way (usually manually specified from smaller to larger ones), aiming to escape from local optimum (Hansen et al., 2010). Different from VND, the SeqRuleGCOP method can be seen as moving among different neighbourhoods based on sequential rules which are offline learned knowledge from effective algorithmic compositions.

Within AutoGCOP, various well-known local search meta-heuristics, such as VNS, can be modelled in a unified template by composing specific algorithmic components in the Improvement procedure (Meng and Qu, 2021). With the basic $o_i \in A_{o_{improve}}$ ordered according to their impact as $[o^{in}_{xchg}, o^{bw}_{xchg}, o^{in}_{ins}, o^{bw}_{ins}, o_{rr}]$, the basic VND can be instantiated in the unified AutoGCOP framework by specifying t_k, o_i and a_j in the Improvement procedure within AutoGCOP (in Algorithm 1) as follows:

- $t_{k_{inner}} \leftarrow t_{converge}(h)$ in line 13,
- $o_i \leftarrow$Specific o_i from $A_{o_{improve}}$ based on a certain order in line 15,
- $a_j \leftarrow a_{oi}$ in line 17;

where $\leftarrow$ denotes the assignment of a in A in Algorithm 1, $t_{converge}(h)$ indicates termination upon the convergence h, and a_{oi} denotes the acceptance

criteria of improvement only.

In VND, the local search procedure is conducted to achieve exploration and exploitation in the neighbourhood. The SeqRuleGCOP method adopts a similar mechanism by setting a_{naive} (to accept all improvements; worse solutions are accepted with a probability of 0.5) for a_j in the Improvement procedure, i.e., to conduct exploration or exploitation randomly. Thus, the comparison focuses on the analysis of $o_i \in A_{o_{improve}}$. In VND, the $o_i \in A_{o_{improve}}$ are ordered as $[o^{in}_{xchg}, o^{bw}_{xchg}, o^{in}_{ins}, o^{bw}_{ins}, o_{rr}]$, following the impact of operators on optimisation objectives.

Table 6.10 presents the overall better performance of SeqRuleGCOP against VND on the validation instances, except for two instances where VND obtains better BEST values. This further validates the effectiveness of the extracted sequential rules.

Regarding the generality of the extracted sequential rules on new VRPTW instances, Table 6.11 shows SeqRuleGCOP performs better than VND in terms of AVG and SD on the testing instances. Particularly for Type-2 instances, SeqRuleGCOP is statistically better than the VND. In terms of the BEST values, SeqRuleGCOP achieves better results than VND in two instances and obtains quite similar BEST to the current best-known results in the literature (i.e., the GAP values are less than 5% in most instances), which further verifies the effectiveness and generality of the extracted sequential rules.

Table 6.10: Comparison between SeqRuleGCOP against VND for effectiveness validation.

Instances	Best-known solutions in the literature	VND		SeqRuleGCOP					
		AVG	SD	BEST	GAP	AVG	SD	BEST	GAP
C102	10828.94(Rochat and Taillard, 1995)	12,613.48	475.57	**12,157.42**	**0.123**	**12,550.22**	**62.37**	12,447.53	0.149
C202	3591.56(Rochat and Taillard, 1995)	5,269.61	707.78	4,659.64	0.297	**3,669.13***	**9.12**	**3,652.54**	**0.017**
R102	18486.12(Rochat and Taillard, 1995)	20,728.66	1,011.19	19,663.67	0.064	**19,836.42**	**280.07**	**18,998.78**	**0.028**
R202	4191.7(Rousseau et al., 2002)	8,257.50	1,108.62	6,338.63	0.512	**5,459.83***	**16.39**	**5,429.78**	**0.295**
RC102	13554.75(Taillard et al., 1997)	17,277.48	692.83	**16,630.28**	**0.227**	**16,892.99**	**60.19**	16,796.01	0.239
RC202	4365.65(Czech and Czarnas, 2002)	8,944.58	935.99	7,540.14	0.727	**5,645.09***	**31.94**	**5,583.99**	**0.279**

Table 6.11: Comparison between SeqRuleGCOP against VND for generality evaluation.

Instances	Best-known solutions in the literature	VND		SeqRuleGCOP					
		AVG	SD	BEST	GAP	AVG	SD	BEST	GAP
C103	10,828.06(Rochat and Taillard, 1995)	12,215.85	744.04	**11,069.86**	**0.022**	**11,602.83***	**82.58**	11,484.92	0.061
C203	3,591.17(Rochat and Taillard, 1995)	5,414.02	709.79	4,701.21	0.309	**3,738.50***	**31.10**	**3,708.68**	**0.033**
R107	11,104.66(Shaw, 1997)	14,784.25	736.35	**13,223.33**	**0.191**	**14,358.96**	**289.68**	13,488.79	0.215
R208	2,726.82(Mester et al., 2007)	5,834.27	999.93	**3,877.13**	**0.422**	**4,022.69***	**17.23**	3,997.70	0.466
RC103	12,261.67(Shaw, 1998)	15,467.41	550.61	**14,530.65**	**0.185**	**14,634.14**	**19.42**	14,603.39	0.191
RC203	4,049.62(Czech and Czarnas, 2002)	7,384.64	1,369.62	5,487.94	0.355	**4,497.68***	**20.72**	**4,465.26**	**0.103**

The better performance of SeqRuleGCOP compared to the instantiated VND indicates that the automatically designed algorithms are better than the manually designed meta-heuristics within the unified AutoGCOP framework. This analysis further supports the effectiveness of the knowledge learned offline by sequential rule mining.

Discussions

In summary, the experiment results confirm the effectiveness of sequential rules of basic operators for automatically designing algorithms to solve VRPTW. It is to be noted that compared with an effective online learning GCOP method, the effectiveness of sequential rules is not significantly better but satisfying. These dues to the fact that the operator composition in MC-GCOP can be seen as utilising a variety of sequential rules, which are more flexible than the extracted top 10 sequential rules. Therefore, effective online learning methods can lead to better optimisation performance compared with the use of sequential rules within AutoGCOP.

In addition, the comparison between SeqRuleGCOP and VND confirms the automatically acquired knowledge in the form of sequential rules can lead to a better optimisation performance than manual-specified operator composition in VND. This comparison tries to solve a fundamental research issue in automated algorithm design, i.e., whether the automated algorithm designs are better than manual algorithm designs. The experiment result can be good evidence to support the research in automated algorithm design.

6.5.2 Effectiveness of composing groups of operators

This section investigates the effectiveness of operator grouping based on the proposed new categorisation in automated composition. The experiments aim to compare the optimisation performance of composing groups of operators against composing individual operators without the group setting. The performance difference is demonstrated in the AutoGCOP framework and the VND search framework.

Investigation in GCOP methods

Within AutoGCOP, RN-GCOP and MC-GCOP are tested to demonstrate the performance difference in automated composition between automated composition with the operator grouping setting and composing basic o_i without the grouping setting. Both GCOP methods add an operator group selection to the $Select(A_{o_{improve}})$ procedure within AutoGCOP (line 15 in Algorithm 1), i.e., select an operator group $A'_o \in A^1_o, A^2_o, A^3_o$ before selecting an operator $o_i \in A'_o$. Therefore, with the operator grouping setting, RN-GCOP and MC-GCOP implement the $Select(A_{o_{improve}})$ procedure in different ways as follows:

- RN-GCOP_group: it selects $A'_o \in A^1_o, A^2_o, A^3_o$ randomly, then chooses a $o_i \in A'_o$ to compose randomly.
- MC-GCOP_group: the learning model M of MC-GCOP in Chapter 4 learns the transitions between $A'_o \in A^1_o, A^2_o, A^3_o$ instead of o_i. With the selected $A'_o \in A^1_o, A^2_o, A^3_o$ by M, $o_i \in A'_o$ is then selected randomly.

Table 6.12 and Table 6.13 present the comparison between automated composition with operator grouping setting against the same composition

Table 6.12: Performance comparison between RN-GCOP and RN-GCOP_group. The best results are in bold. The AVG results are highlighted with $*$ if one method is significantly better than the other method based on Mann–Whitney–Wilcoxon test.

Instances	RN-GCOP			RN-GCOP_group		
	AVG	BEST	STDEV	AVG	BEST	STDEV
C102	13,126.54	12,875.06	269.49	**12848.23***	**12574.38**	**152.44**
C202	3,787.12	3,727.07	49.30	**3737.78***	**3706.70**	**25.97**
R102	21,033.67	20,976.51	**39.80**	**20811.86***	**20000.25**	420.49
R202	**5,542.48**	5,502.88	**28.22**	**5545.77**	**5514.24**	**17.63**
RC102	**17,810.05**	**16,925.97**	401.25	**17908.18**	17018.36	**312.96**
RC202	5,754.53	**5,723.58**	**22.44**	**5728.76***	**5695.29**	29.22
C103	12,364.31	**11,738.51**	416.60	**11592.21***	**11471.27**	**78.57**
C203	4,502.51	3,881.98	514.51	**3727.00***	**3687.34**	**19.49**
R107	14,564.69	14,520.74	**27.27**	**14056.63***	**13436.47**	494.86
R208	4,087.51	4,055.09	16.36	**4022.62***	**4000.13**	**15.57**
RC103	14,881.08	14,691.40	**297.94**	**14664.40***	**14639.33**	**14.34**
RC203	4,784.47	4,539.13	355.59	**4499.23***	**4451.43**	**51.23**

Table 6.13: Performance comparison between MC-GCOP and MC-GCOP_group. The best results are in bold. The AVG results are highlighted with $*$ if one method is significantly better than the other method based on Mann–Whitney–Wilcoxon test.

Instances	MC-GCOP			MC-GCOP_group		
	AVG	BEST	STDEV	AVG	BEST	STDEV
C102	12807.32	12709.09	68.71	**12744.35***	**12491.54**	162.04
C202	3711.08	3691.55	17.69	**3695.03***	**3685.84**	**7.42**
R102	20381.03	19869.93	**549.04**	**20033.91**	**18893.74**	577.82
R202	5531.00	5504.44	**19.69**	**5439.61**	**4669.65**	270.74
RC102	17487.70	**16869.10**	550.34	**17227.43**	16965.31	**397.58**
RC202	5727.94	**5654.04**	**33.85**	**5719.49**	5672.40	36.89
C103	11897.11	**11608.78**	126.47	**11509.54***	11261.04	**109.84**
C203	3878.74	3819.53	41.94	**3725.59***	**3701.03**	**14.36**
R107	14416.76	13538.13	**310.58**	**13652.44***	**13359.77**	415.61
R208	4069.40	4044.94	17.76	**4011.46***	**3994.07**	**13.10**
RC103	14768.24	14699.95	**35.22**	**14317.22***	**13532.17**	487.57
RC203	4620.33	4563.58	49.00	**4486.06***	**4430.24**	**39.03**

method without the grouping setting in RN-GCOP and MC-GCOP, respectively. For both GCOP methods, the automated composition of groups of operators improves the optimisation performance on most selected instances. Particularly, for the selected Type-C instances, the solution quality improvement is significant for both RN-GCOP and MC-GCOP.

For both GCOP methods, the performance improvement by composing groups of operators can be supported by the reduction of the algorithm design search space. The study in (Meng and Qu, 2021) confirms the better individual performance of o_{ins}^{bw} and o_{rr} among the basic operators. Since o_{ins}^{bw} and o_{rr} are the only operators in A_o^2 and A_o^3, randomly composing groups of

operators actually gives a higher probability for these two operators instead of the same probabilities for each o_i.

Investigation in VND

The effectiveness of grouping operators for algorithm design is tested within the VND search framework. The implementation of VND follows the instantiation within AutoGCOP. However, rather than ordering $o_i \in A_{o_{improve}}$, a variant of VND (denoted as VND_group) is operated on an ordered list of operator groups $[A_o^1, A_o^2, A_o^3]$. Within a specific operator group A'_o, an operator o_i is randomly selected and applied during the search. To be noted, the ranking of the operator groups follows the attribute of operator impact to the optimisation objectives, which can be seen as the implementation of the induced general sequential rules of basic operators.

Table 6.14: Performance comparison between VND and VND_group. The best results are in bold. The AVG results are highlighted with $*$ if one method is significantly better than the other method based on Mann–Whitney–Wilcoxon test.

Instances	VND			VND_group		
	AVG	BEST	STDEV	AVG	BEST	STDEV
C102	**12,613.48***	**12,157.42**	**475.57**	**13403.46**	**12253.34**	**879.32**
C202	5,269.61	**4,659.64**	707.78	**5038.49**	**4792.27**	**398.54**
R102	20,728.66	19,663.67	**1,011.19**	**20132.81**	**19647.95**	**557.66**
R202	**8,257.50**	**6,338.63**	**1,108.62**	**8162.85**	**6451.09**	**1033.17**
RC102	**17,277.48**	**16,630.28**	692.83	**16930.72**	16673.96	**478.34**
RC202	**8,944.58**	**7,540.14**	**935.99**	**9352.78**	**7470.38**	**876.93**
C103	12,215.85	**11,069.86**	744.04	**11952.24**	**11183.68**	**622.95**
C203	5,414.02	4,701.21	709.79	**5151.62**	**4677.52**	**477.95**
R107	14,784.25	**13,223.33**	**736.35**	**14398.89**	**13253.74**	906.67
R208	5,834.27	3,877.13	**999.93**	**5630.14**	**3966.11**	**1243.26**
RC103	15,467.41	14,530.65	**550.61**	**15374.45**	**14479.81**	**659.94**
RC203	7,384.64	**5,487.94**	1,369.62	**7355.15**	**6290.41**	**973.56**

Table 6.14 presents the comparison between the VND with the basic operators and the VND_group which operates on the three groups of operators. The results show composing the operators with the grouping setting achieves an improvement in the AVG solution quality within VND, however, not statistically significant. Particularly, the VND with individual

operators can achieve better BEST results for solving some instances.

Although for the RN-GCOP and MC-GCOP methods, reducing the search space of algorithm design from five basic operators to three groups of operators is beneficial for improving the optimisation effectiveness, it is not sufficient to improve the performance of VND. This is possibly due to the idea of VND which systematically changes the exploration in different neighbourhood structures and can conduct exploitation in specific neighbourhoods with local search. This guarantees high flexibility during the search process.

Discussions

The above experimental study confirms that composing operators based on the groups in automated composition can improve the optimisation performance more than composing individual operators. The performance improvement supports that the basic operators are categorised in a reasonable way such that the search space in algorithm design is reduced effectively. In the frameworks of local search algorithms, such as the VND, the influence of using the operator grouping setting might not be as significant as in the AutoGCOP framework.

An effective categorisation of operators is essential for reducing the search space of algorithm design. There are different ways of categorising operators in the literature. However, some relevant research issues are still under-explored, such as how the categorisation of operators affects the optimisation performance in different problem domains, and how different algorithm frameworks can benefit from reducing the search space of algorithm design. This experimental study can be seen as the first attempt at these research issues for the automated algorithm composition within the

AutoGCOP framework for solving VRPTW.

6.6 Conclusions

Machine learning has been applied to automate the design of search algorithms. The rich and new knowledge generated during the search can be collected as data and captured in the machine learning models. This hidden knowledge is, however, implicit to interpret. This chapter investigated the data of effective algorithm designs with machine learning techniques to gain insightful and interpretive knowledge from the effective algorithmic compositions. The analysis of the knowledge leads to a better understanding of the behaviour of algorithmic components, thus supporting automated algorithm design.

With a newly defined problem model GCOP, the algorithm can be seen as the composition of elementary algorithmic components. This study treats the algorithmic compositions as sequential data to explore the knowledge hidden in the effective algorithmic compositions. Within a unified AutoGCOP algorithm design framework, a considerable number of effective algorithmic compositions are collected for solving benchmark VRPTW instances and further processed for investigations.

The elite algorithmic compositions are investigated with sequential rule mining to extract a set of sequential rules of basic operators. To support algorithm design, this study utilises the frequent sequential rules in an explicit way by applying the extracted sequential rules in the automated composition of basic operators within AutoGCOP. In comparison with different methods, the sequential rules can effectively automate the design of local search algorithms to solve VRPTW instances.

In addition, the analysis of the common sequential rules of operators reveals an important feature of operators, i.e., their impact on optimisation objectives. The basic operators can thus be categorised into different groups according to their impact. This newly introduced categorisation to the literature could be adapted as a useful indicator when determining suitable algorithmic components in algorithm design, which can be seen as an implicit way of using the extracted knowledge. Experiment results verify the effectiveness of automatically composing groups of operators for solving VRPTW. This supports that grouping operators based on their impact can effectively reduce the search space of algorithm design.

This chapter highlights the importance of knowledge interpretability in promoting research in automated algorithm design, which complements Chapter 5 which focuses on more advanced machine learning models for learning knowledge. This study is good evidence showing the information collected over the search run(s) can be useful and that some design principles can be identified by offline learning. The exploration of sequential rule mining in automated algorithm design opens a number of potential research directions for future research.

Firstly, it calls for the exploration of different knowledge representations and knowledge discovery techniques which may lead to interesting findings to support algorithm design. The performance improvement of sequential rules in the automated composition is limited compared to the MC-GCOP method in Chapter 4. It could be due to the knowledge hidden in algorithm designs can be too sophisticated to be represented by rules, thus difficult to be captured by rule mining techniques in complex problem domains. Apparently, the online reinforcement learning scheme in MC-GCOP in Chapter 4 allows more flexibility in automated composition than offline learning. The knowledge in the form of sequential rules can be seen as

a subset of knowledge acquired by an effective online learning model. In this regard, this work can be extended by investigating the effective algorithm compositions with other rule-mining techniques, such as patterns (episodes) that appear in a long single algorithm composition, and patterns that periodically appear in many algorithm compositions.

In addition, in principle, the applied sequential rule mining could be extended to handle other combinatorial problems. It will be important to test the proposed SeqRuleGCOP method in real-world applications and compare it against other learning-based methods to further verify the effectiveness, re-usability and generality of sequential rules in automated algorithm design. Moreover, while this study focused on the behaviour of basic operators, it would be interesting to learn the sequential rules of operators and acceptance criteria and furthermore, under the evolutionary search paradigms.

Chapter 7

Conclusions and future works

Contents

7.1 Conclusions

This thesis has studied machine learning in the automated design of local search-based algorithms based on a newly defined GCOP model. The main research aim (RA) is to systematically investigate how to learn useful knowledge from algorithm designs to automatically compose elementary algorithmic components in the GCOP model, thus enhancing the understanding of algorithm design and introducing new effective algorithms to the literature.

Through the review of studies in automated algorithm design, the thesis introduced an extended taxonomy for different automation tasks involved in the algorithm design process. This extended taxonomy enables a more systematic classification and review of studies related to each specific automation task. During the review process, several research gaps (RGs) are identified in the field of automated composition, highlighting areas that require further investigation to achieve the RA of the thesis. To address the RGs and achieve the RA of the thesis, a series of research questions (RQs) are proposed. Several research objectives (ROs) are identified to effectively address the RQs. The widely studied VRPTW is used as the case study in the thesis.

The research starts by proposing a general AutoGCOP framework to support the automated composition of elementary algorithmic components in GCOP. Based on AutoGCOP, the thesis investigates the performance of elementary algorithmic components and systematically investigates different learning techniques in the automated composition of these components for solving the VRPTW. The main findings and achievements are summarised below.

7.1.1 A general framework for automated design of local search algorithms

Chapter 3 proposes AutoGCOP, a new general framework for the automated design of local search algorithms based on the GCOP model. With the encapsulated common processes in local search algorithms, AutoGCOP defines a general framework to instantiate existing local search algorithms designed by manually picked algorithmic components and support the automatic design of new novel algorithms which may be highly different from those manually designed in the literature.

Within the consistent AutoGCOP framework, various elementary algorithmic components are analysed in Chapter 3 for solving the benchmark VRPTW. The study confirms the satisfying performance of the elementary algorithmic components for VRPTW. The problem-specific algorithmic components present significant effectiveness when included in the basic component set, however, such algorithmic components require domain knowledge for the design which may not be available or consistent in practice. The general AutoGCOP with elementary algorithmic components presents a promising framework across different problems and may be employed by developers of different expertise.

The research presented in this chapter sets the base for future research directions in the automated design of search algorithms with basic algorithmic components. On the one hand, the AutoGCOP framework underpins the automated design of new and unseen algorithms by using different GCOP methods which compose the algorithmic components automatically. On the other hand, a considerable number of new algorithmic compositions of the basic GCOP components can be collected within AutoGCOP as benchmark data, calling for further investigations which could lead to

new knowledge for designing new effective algorithms.

7.1.2 Online learning with Markov Chain and Reinforcement Learning

Chapter 4 investigates the online learning of effective composition of the basic algorithmic components for automated algorithm design within the AutoGCOP framework. Two learning models based on Reinforcement Learning (RL) and Markov chain (MC) are investigated to learn from the feedback in problem-solving to enhance the compositions of algorithmic components. These two learning models learn the behaviour of algorithmic components during the search in different ways and compare the effectiveness of two different learning perspectives. Specifically, the simple RL scheme learns the individual performance of algorithmic components, while the MC model enhanced by the RL learns the transition performance of algorithmic components during the search.

With VRPTW as the testbed, the MC model presents a superior performance in learning the compositions of algorithmic components during the search, demonstrating the effectiveness of learning the transition between pairs of algorithmic components in composing new algorithms automatically. The result of this research indicates useful knowledge might lie in the sequential behaviour of algorithmic compositions in designing new algorithms automatically.

7.1.3 Offline learning with LSTM and Transformer

Chapter 5 investigates the predictive knowledge in effective algorithmic compositions with machine learning models. The aim is to gain insightful knowledge from the effective algorithmic compositions to forecast the behaviour of algorithmic components, thus supporting algorithm design.

Firstly, the process of forecasting algorithmic components in the design of effective local search algorithms is defined as a sequence classification task. This newly defined machine learning task enables the use of various machine learning models to predict the algorithmic components to compose, thus supporting the automated design of new unseen local search algorithms. This new task brings the research communities of both evolutionary computation and machine learning and presents new challenges to machine learning experts.

Chapter 5 investigated various machine learning models in the new machine learning task, finding the superior performance of the Long Short-term Memory (LSTM) neural network. Due to the imbalance problem of the algorithmic composition data, LSTM is compared against various machine learning models on the data sets processed with the commonly used resampling methods. Due to its sequence specialisation of the network structure, LSTM presents robust superior prediction performance compared to other learning models at extracting the sequential relations in algorithmic compositions.

To further investigate its performance on learning from longer algorithmic compositions, Chapter 5 compared LSTM and other machine learning models against a Transformer network. Both LSTM and Transformer show increasing performance at forecasting the behaviour of algorithmic compo-

nents given longer algorithmic compositions. Furthermore, Transformer outperforms LSTM in effectively learning knowledge from longer algorithmic compositions.

To the best of our knowledge, it is the first attempt to propose an LSTM model and a Transformer model in learning from the automated compositions for the automated design of search algorithms. The research in Chapter 5 confirms the superior performance of LSTM and Transformer in the defined new machine learning task on automated algorithm design.

Along with the investigations of the learning models, Chapter 5 investigated various features utilised in the effective algorithmic composition data. Among different types of information, the search stage, operator features, solution quality change and instance features show great effectiveness in predicting suitable algorithmic components. Specifically, search stage and instance features (particularly the scheduling horizon and time window width) are the key features, which offer new insights in designing new effective local search algorithms.

7.1.4 Offline learning with sequential rule mining

Chapter 6 investigates explicit knowledge hidden in effective algorithmic compositions with sequential rule mining techniques towards automated algorithm design within AutoGCOP. The aim is to interpret the sequential relations hidden in algorithmic compositions, hoping to gain new insights into designing effective algorithms automatically. To the best of our knowledge, this is the first study on sequential rule mining techniques in the investigation of automated algorithm composition.

The extracted sequential rules of algorithmic components are applied within

AutoGCOP in automated composition to further verify the effectiveness of the extracted explicit knowledge. A sequential rule-based GCOP method is developed within AutoGCOP to support the investigation. The offline learned sequential rules show satisfying performance in the automated composition of basic algorithmic components for solving VRPTW, however, are worse than the MC-based online model in Chapter 4. This indicates that algorithmic compositions lead to complex behaviours which cannot be fully captured by rule mining techniques but can be partially captured to some extent.

In addition, the extracted sequential rules show important attributes of operators, i.e., operator impact on optimisation objectives, which provides some new insights into algorithm design in the literature. The results of the research confirm the effectiveness of categorising the basic operators not only in automated composition but also in enhancing existing local search algorithms for solving VRPTW.

7.1.5 Remarks

In summary, the thesis systematically investigated different perspectives of learning in automated algorithm design to achieve the research aim, i.e., how to leverage machine learning to automatically design effective heuristic algorithms based on the new GCOP standard. The studies in Chapter 3 addresses RQ1 by achieving RO1 and RO2. The general AutoGCOP framework proposed in Chapter 3 underpins the research in the following chapters for addressing RQ2.

The main research focus of the following chapters is different according to the specific ROs. Chapter 4 focuses on RO3, i.e., the behaviour of

Table 7.1: A summary of the main studies of different learning methods in the thesis.

Chapters	Learning tasks	Learning methods	Learning style	Knowledge type	Aim of learning
Chapter 4	RL	MC enhanced with RL	Online	Predictive	To forecast the next operator given the current operator
Chapter 5	Sequence classification	LSTM, Transformer	Offline	Predictive	To forecast the next operator given the previously applied operators and other information
Chapter 6	Rule inference	Sequential rule mining RL	Offline	Descriptive	To find frequent sequential rules between operators

learning in GCOP methods. Chapter 5 and Chapter 6 focus on discovering useful knowledge in algorithm design towards automated composition (i.e., RO4 and RO5, respectively). Table 7.1 overviews the different learning methods investigated within AutoGCOP in the thesis, showing the different aspects of learning that have been investigated in the thesis for enhancing the understanding of learning in automated algorithm design. It is worth noting that these studies are mainly different in terms of:

- The machine learning task, which is modelled from the decision-making in algorithm design.
- The way of knowledge discovery, in terms of learning methods and learning styles (i.e., online or offline).
- The types of knowledge, i.e., predictive (implicit) or descriptive (explicit).
- The aim of learning, i.e., how the knowledge can be used in decision-making, either directly determining the action in automated composition or inducing patterns and rules.

In addition, these studies are different in terms of the efforts on enhancing the effectiveness of the designed algorithms. Specifically, the online learning

method in Chapter 4 considers the final optimisation performance during learning by the RL scheme which maximises the accumulated immediate reward during problem-solving. The offline learning methods in Chapter 5 and Chapter 6 consider the final optimisation performance by using the effective algorithmic compositions as the data for the investigation, thus the efforts are made before the learning.

These studies share a common focus, i.e., they all focus on learning the sequential relations between algorithmic components in the automated composition. They are all based on the hypothesis that the algorithmic compositions are sequential data and the decision-making of the next algorithmic component is influenced by the previous ones used. The results of the thesis confirm the effectiveness of the investigated learning methods, which supports the hypothesis and highlights the need for analysing the behaviour of the algorithmic components during automated composition.

7.2 Future research directions

The research in the thesis shows that machine learning techniques can greatly contribute to automated algorithm design from different aspects. The proposed future research directions build upon the main works and contributions of this thesis and are centred around four key aspects, i.e., the way of learning, the knowledge acquired through learning, the target of learning and the application domain. Through these research directions, we mainly aim to explore new possibilities in machine learning for improving the field of automated algorithm design.

7.2.1 Innovative approaches to implementing machine learning

This thesis mainly focuses on acquiring useful knowledge hidden in the search data with machine learning by modelling the problem of algorithm design (i.e., the determination of algorithmic components to compose) as different machine learning tasks. However, there are many other machine learning tasks, such as regression and clustering (Bishop and Nasrabadi, 2006). It would be interesting to further explore the potential of modelling automated algorithm design tasks as new machine learning tasks, to support the exploration of new machine learning methods.

In addition, it would be interesting to explore the effectiveness of different approaches for modelling algorithm design tasks as learning tasks. This exploration can valuable insights into the effectiveness and applicability of different approaches in the decision-making of the algorithm design process. The development of appropriate evaluation metrics is crucial for assessing different learning approaches in the context of automated algorithm design. These metrics should take into account several key factors to provide a comprehensive evaluation, such as solution quality in problem-solving (i.e., solving optimisation problems), the efficiency of the learning process and problem-solving, scalability, and adaptability to diverse problem domains.

Furthermore, automated algorithm design can be further enhanced with the recent advances in general solution encoding (Stone et al., 2021) and end-to-end problem-solving by machine learning(Vesselinova et al., 2020), (Kotary et al., 2021). It is important to note that in terms of problem-solving, search algorithms search the solution space by manually designed strategies, while machine learning builds a model to be able to produce solutions by learning the problem structure. Recent advances confirm the

superior performance of end-to-end approaches with machine learning over conventional heuristics (Wu et al., 2021). This suggests that machine learning is able to learn the complex structure of optimisation problems, some of which are very difficult to identify by human experts, and thus can further enhance the performance of search algorithms. Considering such incorporation into automated algorithm design could further raise the level of automation by reducing human involvement in algorithm design without the loss of generality for addressing different problem domains of common structures.

7.2.2 Knowledge interpretability

The research on machine learning in this thesis confirms that effective algorithm designs contain hidden patterns that can be acquired through rule inference learning methods, leading to further improvement in automated algorithm design. This suggests the importance of exploring different knowledge representations and knowledge discovery techniques that can reveal interesting findings and support algorithm design. In this regard, it is worthwhile to investigate other rule inference learning techniques on effective algorithm designs, such as analysing periodical patterns or frequent patterns in long, single algorithm compositions.

7.2.3 Learning different types of algorithmic components

This thesis investigates different learning perspectives of basic GCOP components, in particular the basic operators. However, other algorithmic components, such as acceptance criteria, also show significant effects on

the performance of local search algorithms in the literature (Jackson et al., 2018). For example, attention is given to both low-level heuristics and acceptance criteria in selection hyper-heuristics (Asta and Özcan, 2014), (Tyasnurita et al., 2015). Within the general AutoGCOP framework, it would be interesting to extend the investigations of effective algorithmic compositions to the behaviour of acceptance criteria, as well as the combination of basic operators and acceptance criteria. A recent study in (Yi et al., 2022) models both local search-based and population-based algorithms with a wider range of algorithmic components into a general search framework. Thus, there is a wide range of possible directions that should be explored to further benefit from the investigations of different types of algorithmic components.

7.2.4 Other problem domains

This thesis has introduced the general AutoGCOP framework for supporting the design of local search-based search algorithms. The promising results achieved in the thesis show that the automatically designed algorithms based on AutoGCOP with the basic algorithmic components are indeed effective for solving VRPTW. In line with the work of (Yi et al., 2022), there is mounting evidence that the basic GCOP components are effective in VRPTW, and potentially can greatly contribute to multiple problem domains. Thus, there is a wide range of possible extensions for solving other optimisation problems in addition to VRPs.

Appendices

Appendix A

Machine Learning for Evolutionary Computation - the Vehicle Routing Problems Competition

A.1 Introduction

This appendix introduces The Competition of Machine Learning for Evolutionary Computation for Solving the Vehicle Routing Problems (ML4VRP) [1] that was organised as an essential part of this research study. This competition aims to serve as a vehicle to bring together the latest developments of machine learning-assisted evolutionary computation for VRPs. It engages participants to explore innovative and effective approaches to incorporate machine learning into meta-heuristics and evolutionary computation for solving VRPs.

[1] https://sites.google.com/view/ml4vrp

The motivation behind this competition aligns closely with the objective of the thesis. The thesis focuses on how machine learning can be used in the automated design of search algorithms. In the competition, participants can explore machine learning for not only designing search algorithms but also enhancing existing ones. Thus, the competition encourages a wider range of possibilities for machine learning in search algorithms compared to the research studies in the thesis. The studies conducted within the thesis can be seen as specific scenarios that fall within the broader scope of the competition. The competition, in which I work as one of the organisers, can be seen as an important supporting evidence for this thesis, thus strengthening the credibility of the conducted research.

To effectively evaluate submissions to the competition, a solution evaluator tool has been developed using Python. It is a stand-alone tool that can be easily integrated into various algorithm design frameworks aimed at solving VRPs, making it a valuable application for algorithm designers.

A.2 Description of the competition

The research in evolutionary computational algorithms and meta-heuristics mainly focuses on improving the optimisation performance, discarding a vast amount of data produced in the evolution/optimisation process. New useful knowledge can be extracted from the data with machine learning to enhance human-designed evolutionary computation (Pillay, 2021). VRP variants of different difficulties provide an ideal testbed to enable a performance comparison of machine learning-assisted computational optimisation.

The ML4VRP competition calls for the development of machine learning-

assisted evolutionary computation for VRPs. The competition focuses on the VRP with Time Windows (VRPTW), a widely studied problem model in the VRP literature. Participants must develop machine learning techniques which can design and enhance evolutionary computational algorithms or meta-heuristics for solving VRPTW.

Solomon (Solomon, 1987) and Hombeger-Gehring (Homberger and Gehring, 1999) instances are widely tested in the VRPTW literature. Both data sets consist of six types of instances, i.e., C1, C2, R1, R2, RC1, and RC2, which differ with respect to the customers' geographical locations, vehicle capacity, density, and tightness of the time windows.

The problem instances provided in the competition are taken from the following two sources:

- Solomon (Solomon, 1987) data set of 100 customers.
- Homberger and Gehring (Homberger and Gehring, 1999) data set of 200 customers and 400 customers.

The provided problem instances are randomly selected from these three sized problem instances, covering different instance types. These instances are available to download from the competition's GitHub repository and can also be found in CVRPLIB[2]. A subset of these instances will be used to evaluate the submitted solutions, which will remain unknown to the participants until the results are released.

Participants are required to submit descriptions of the developed algorithms and the corresponding solutions for the provided problem instances. The submissions will be evaluated on randomly selected instances (from the

[2]CVRPLIB website: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

provided instances) using an evaluator available in the GitHub repository[3] dedicated to the ML4VRP competition. The most widely adapted evaluation function, i.e. to minimise the number of vehicles and total travel distance, is used to determine the best machine learning-assisted evolutionary algorithms for solving VRPs. The algorithm which produced the best average fitness for solving VRPs will receive the highest score.

A.3 Solution evaluator

The solution evaluator dedicated to the competition is implemented using Python. Given a solution in a specific format and the corresponding problem instance, the evaluator performs a series of checks to determine the feasibility of the solution. If the solution is feasible, the evaluator proceeds with the calculation and prints out the objective function value, the number of routes and the total travel distance. If the solution is infeasible, the evaluator returns a failure status.

Several studies on VRPTW use the hierarchical objective function, where minimising the number of vehicles (routes) is the primary objective, followed by minimising the total travel distance or travel time as the secondary objective (Bräysy and Gendreau, 2005a). Some search algorithms in the literature adopt the weighted sum objective function (Bräysy and Gendreau, 2005b). This competition follows the convention (as used in the Cross-domain Heuristic Search Challenge (CHeSC) (Burke et al., 2011) for the hidden domain VRPTW (Walker et al., 2012)), i.e., considering the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD), which is the Equation 2.12 as used in the

[3]GitHub repository for the ML4VRP competition: https://github.com/ML4VRP2023/ML4VRP2023

thesis.

The evaluator can be used as a stand-alone tool to be incorporated into other algorithm frameworks for solving VRPTW. Python is used as the programming language to allow easy evaluation and result analysis.

A.4 Summary

This appendix introduces the ML4VRP competition which encourages the collaboration between machine learning and evolutionary computation to further enhance the efficiency and effectiveness of search algorithms in solving VRPs. Compared to the research aim of the thesis, the competition focuses on a wider perspective of the use of machine learning in search algorithms. Thus, the competition can be seen as supporting evidence for the research studies conducted in the thesis. A solution evaluator for VRPTW is developed for the evaluation of the solutions submitted to the competition. The ease of integration makes this solution evaluator a practical tool for algorithm designers aimed at solving VRPTW.

The submissions and results collected during the competition will be further analysed to serve as a valuable benchmark to the literature. Comparing the performance of the submissions could offer practical insights for future work in developing novel machine learning strategies for the development of effective algorithms. In the future, the competition can be extended to other VRP domains, such as the classic VRP with capacity constraints (CVRP), as well as Timetabling and other widely studied Combinatorial Optimisation Problems.

Bibliography

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1.

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

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE.

Ai, T. J. and Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702.

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

Alfa, A. S., Heragu, S. S., and Chen, M. (1991). A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21(1-4):635–639.

Almeida, C. P., Gonçalves, R. A., Venske, S., Lüders, R., and Delgado, M.

(2020). Hyper-heuristics using multi-armed bandit models for multi-objective optimization. *Applied Soft Computing*, 95:106520.

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Altinkemer, K. and Gavish, B. (1991). Parallel savings based heuristics for the delivery problem. *Operations research*, 39(3):456–469.

Ansótegui, C., Sellmann, M., and Tierney, K. (2009). A gender-based genetic algorithm for the automatic configuration of algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pages 142–157. Springer.

Araya, I. and Riff, M.-C. (2014). A filtering method for algorithm configuration based on consistency techniques. *Knowledge-Based Systems*, 60:73–81.

Asta, S. and Özcan, E. (2014). An apprenticeship learning hyper-heuristic for vehicle routing in HyFlex. In *2014 IEEE symposium on evolving and autonomous learning systems (EALS)*, pages 65–72. IEEE.

Asta, S., Özcan, E., Parkes, A. J., and Etaner-Uyar, A. Ş. (2013). Generalizing hyper-heuristics via apprenticeship learning. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 169–178. Springer.

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Badeau, P., Guertin, F., Gendreau, M., Potvin, J.-Y., and Taillard, E. (1997). A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5(2):109–122.

Baker, B. M. and Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800.

Balaprakash, P., Birattari, M., and Stützle, T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In *International workshop on hybrid metaheuristics*, pages 108–122. Springer.

Balaprakash, P., Birattari, M., Stützle, T., and Dorigo, M. (2010). Estimation-based metaheuristics for the probabilistic traveling salesman problem. *Computers & Operations Research*, 37(11):1939–1951.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

Battiti, R. and Brunato, M. (2010). Reactive search optimization: learning while optimizing. In *Handbook of Metaheuristics*, pages 543–571. Springer.

Battiti, R. and Campigotto, P. (2011). An investigation of reinforcement learning for reactive search optimization. In *Autonomous Search*, pages 131–160. Springer.

Battiti, R. and Protasi, M. (1997). Reactive search, a history-sensitive heuristic for MAX-SAT. *Journal of Experimental Algorithmics (JEA)*, 2:2–es.

Battiti, R. and Protasi, M. (2001). Reactive local search for the maximum clique problem 1. *Algorithmica*, 29(4):610–637.

Battiti, R. and Tecchiolli, G. (1994). The reactive tabu search. *ORSA journal on computing*, 6(2):126–140.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

Beasley, J. E. (1983). Route first—cluster second methods for vehicle routing. *Omega*, 11(4):403–408.

Beheshti, Z. and Shamsuddin, S. M. H. (2013). A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl*, 5(1):1–35.

Bell, J. E. and McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1):41–48.

Ben-Ameur, W. (2004). Computing the initial temperature of simulated annealing. *Computational Optimization and Applications*, 29(3):369–385.

Berger, J., Barkaoui, M., and Bräysy, O. (2003). A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 41(2):179–194.

Bezerra, L. C., López-Ibánez, M., and Stützle, T. (2015). Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 11–18. Morgan Kaufmann Publishers Inc.

Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010). F-race and iterated F-Race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., et al. (2016). Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Blanton Jr, J. L. and Wainwright, R. L. (1993). Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 452–459.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373.

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

Bogyrbayeva, A., Meraliyev, M., Mustakhov, T., and Dauletbayev, B. (2022). Learning to solve vehicle routing problems: A survey. *arXiv preprint arXiv:2205.02453*.

Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information sciences*, 237:82–117.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368.

Bräysy, O. and Gendreau, M. (2001). Genetic algorithms for the vehicle routing problem with time windows. *Arpakannus,(1)*, pages 33–38.

Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139.

Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J. A., and Gendreau, M. (2010). Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In *IEEE congress on evolutionary computation*, pages 1–8. IEEE.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A. J., and Petrovic, S. (2011). The cross-domain heuristic search challenge–an international research competition. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 631–634. Springer.

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

Caccetta, L., Alameen, M., and Abdul-Niby, M. (2013). An improved clarke

and wright algorithm to solve the capacitated vehicle routing problem. *Engineering, Technology & Applied Science Research*, 3(2):413–415.

Carchrae, T. and Beck, J. C. (2005). Applying machine learning to low-knowledge control of optimization algorithms. *Computational Intelligence*, 21(4):372–387.

Cattaruzza, D., Absi, N., Feillet, D., and Vidal, T. (2014). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236(3):833–848.

Cengiz Toklu, M. (2022). A fuzzy multi-criteria approach based on clarke and wright savings algorithm for vehicle routing problem in humanitarian aid distribution. *Journal of Intelligent Manufacturing*, pages 1–21.

Chakhlevitch, K. and Cowling, P. (2005). Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 23–33. Springer.

Chaovalitwongse, W., Kim, D., and Pardalos, P. M. (2003). Grasp with a new local search scheme for vehicle routing problems with time windows. *Journal of Combinatorial Optimization*, 7:179–207.

Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, pages 875–886.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chen, B. (2018). *Optimisation models and algorithms for real-life trans-*

portation routing and scheduling problems. PhD thesis, University of Nottingham.

Chen, B., Qu, R., Bai, R., and Ishibuchi, H. (2016). A variable neighbourhood search algorithm with compound neighbourhoods for vrptw.

Chen, C., Demir, E., and Huang, Y. (2021). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research*, 294(3):1164–1180.

Chen, T. K., Hay, L. L., and Ke, O. (2001). Hybrid genetic algorithms in solving vehicle routing problems with time window constraints. *Asia-Pacific Journal of Operational Research*, 18(1):121.

Chiang, W.-C. and Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27.

Ching, W.-K. and Ng, M. K. (2006). Markov chains. *Models, algorithms and applications*.

Christofides, N. (1979). The vehicle routing problem. *Combinatorial optimization*.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581.

Connolly, D. (1992). General purpose simulated annealing. *Journal of the Operational Research Society*, 43(5):495–505.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.-S. (2005). *New heuristics for the vehicle routing problem*. Springer.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research society*, 53(5):512–522.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Vehicle routing. *Handbooks in operations research and management science*, 14:367–428.

Cowling, P. and Chakhlevitch, K. (2003). Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 2, pages 1214–1221. IEEE.

Cowling, P., Kendall, G., and Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 176–190. Springer.

Cowling, P., Kendall, G., and Soubeiga, E. (2002). Hyperheuristics: A robust optimisation method applied to nurse scheduling. In *International Conference on Parallel Problem Solving from Nature*, pages 851–860. Springer.

Coy, S. P., Golden, B. L., Runger, G. C., and Wasil, E. A. (2001). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97.

Crammer, K. and Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine learning*, 47(2):201–233.

Cruz-Reyes, L., Gómez-Santillán, C., Pérez-Ortega, J., Landero, V.,

Quiroz, M., and Ochoa, A. (2012). Algorithm selection: From meta-learning to hyper-heuristics. In *Intelligent Systems*. IntechOpen.

Cuervo, D. P., Goos, P., Sörensen, K., and Arráiz, E. (2014). An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 237(2):454–464.

Czech, Z. J. and Czarnas, P. (2002). Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th Euromicro workshop on parallel, distributed and network-based processing*, pages 376–383. IEEE.

d O Costa, P. R., Rhuggenaath, J., Zhang, Y., and Akcay, A. (2020). Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian Conference on Machine Learning*, pages 465–480. PMLR.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

De Jong, K. A. and Spears, W. M. (1991). An analysis of the interacting roles of population size and crossover in genetic algorithms. In *Parallel Problem Solving from Nature: 1st Workshop, PPSN I Dortmund, FRG, October 1–3, 1990 Proceedings 1*, pages 38–47. Springer.

Desale, S., Rasool, A., Andhale, S., and Rane, P. (2015). Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. *Int. J. Comput. Eng. Res. Trends*, 351(5):2349–7084.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. (2018). Learning heuristics for the tsp by policy gradient. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft,*

The Netherlands, June 26–29, 2018, Proceedings 15, pages 170–181. Springer.

Di Gaspero, L. and Urli, T. (2011). A reinforcement learning approach for the cross-domain heuristic search challenge. In *Proceedings of the 9th Metaheuristics International Conference (MIC 2011), Udine, Italy.*

Di Gaspero, L. and Urli, T. (2012). Evaluation of a family of reinforcement learning cross-domain optimization heuristics. In *International Conference on Learning and Intelligent Optimization*, pages 384–389. Springer.

Dietterich, T. G. (2002). Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*, pages 15–30. Springer.

Doerner, K. F., Hartl, R. F., and Lucka, M. (2005). A parallel version of the d-ant algorithm for the vehicle routing problem. *Parallel Numerics*, 5:109–118.

Doerr, B., Lissovoi, A., Oliveto, P. S., and Warwicker, J. A. (2018). On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1015–1022.

Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE.

dos Santos, J. P. Q., de Melo, J. D., Neto, A. D. D., and Aloise, D. (2014). Reactive search strategies using reinforcement learning, local search

algorithms and variable neighborhood search. *Expert Systems with Applications*, 41(10):4939–4949.

Drake, J. H., Kheiri, A., Özcan, E., and Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405–428.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee.

Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483.

El-Sherbeny, N. A. (2010). Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University-Science*, 22(3):123–131.

Elshaer, R. and Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, 140:106242.

Feo, T. A. and Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.

Fisher, M. and Fisher, M. (1995). Chapter 1 vehicle routing. *Handbooks in Operations Research and Management Science*, 8:1–33.

Fisher, M. L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124.

Fleszar, K., Osman, I. H., and Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3):803–809.

Fournier-Viger, P., Faghihi, U., Nkambou, R., and Nguifo, E. M. (2010). Cmrules: an efficient algorithm for mining sequential rules common to several sequences. In *Twenty-Third International FLAIRS Conference*.

Fournier-Viger, P., Faghihi, U., Nkambou, R., and Nguifo, E. M. (2012a). Cmrules: Mining sequential rules common to several sequences. *Knowledge-Based Systems*, 25(1):63–76.

Fournier-Viger, P., Gueniche, T., and Tseng, V. S. (2012b). Using partially-ordered sequential rules to generate more accurate sequence prediction. In *International Conference on Advanced Data Mining and Applications*, pages 431–442. Springer.

Fournier-Viger, P., Gueniche, T., Zida, S., and Tseng, V. S. (2014). Erminer: sequential rule mining using equivalence classes. In *International Symposium on Intelligent Data Analysis*, pages 108–119. Springer.

Fournier-Viger, P., Lin, J. C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., and Lam, H. T. (2016). The spmf open-source data mining library version 2. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 36–40. Springer.

Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77.

Fournier-Viger, P., Nkambou, R., and Tseng, V. S.-M. (2011). Rulegrowth: mining sequential rules common to several sequences by

pattern-growth. In *Proceedings of the 2011 ACM symposium on applied computing*, pages 956–961.

Fournier-Viger, P. and Tseng, V. S. (2011). Mining top-k sequential rules. In *International Conference on Advanced Data Mining and Applications*, pages 180–194. Springer.

Fournier-Viger, P. and Tseng, V. S. (2013). Tns: mining top-k non-redundant sequential rules. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 164–166.

Fournier-Viger, P., Wu, C.-W., Tseng, V. S., Cao, L., and Nkambou, R. (2015). Mining partially-ordered sequential rules common to multiple sequences. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2203–2216.

Fournier-Viger, P., Wu, C.-W., Tseng, V. S., and Nkambou, R. (2012c). Mining sequential rules common to several sequences with the window size constraint. In *Canadian Conference on Artificial Intelligence*, pages 299–304. Springer.

Franzin, A. and Stützle, T. (2019). Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, 104:191–206.

Funke, B., Grünert, T., and Irnich, S. (2005). Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of heuristics*, 11:267–306.

Gagliolo, M. and Schmidhuber, J. (2006). Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence*, 47(3-4):295–328.

Gagliolo, M., Zhumatiy, V., and Schmidhuber, J. (2004). Adaptive online time allocation to search algorithms. In *European conference on machine learning*, pages 134–143. Springer.

Gambardella, L. M., Taillard, É., and Agazzi, G. (1999). Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows.

Gao, L., Chen, M., Chen, Q., Luo, G., Zhu, N., and Liu, Z. (2020). Learn to design the heuristics for vehicle routing problem. *arXiv preprint arXiv:2002.08539*.

Garcia, B.-L., Potvin, J.-Y., and Rousseau, J.-M. (1994). A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, 21(9):1025–1033.

Gaskell, T. (1967). Bases for vehicle fleet scheduling. *Journal of the Operational Research Society*, 18(3):281–295.

Gehring, H. and Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, volume 2, pages 57–64. Citeseer.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290.

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., and Løkketangen,

A. (2008). *Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography.* Springer.

Ghoseiri, K. and Ghannadpour, S. F. (2010). Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10(4):1096–1107.

Gillett, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349.

Giuliari, F., Hasan, I., Cristani, M., and Galasso, F. (2021). Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*, pages 10335–10342. IEEE.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

Glover, F. and Laguna, M. (1998). *Tabu search.* Springer.

Goksal, F. P., Karaoglan, I., and Altiparmak, F. (2013). A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, 65(1):39–53.

Goldman, B. W. and Tauritz, D. R. (2011). Meta-evolved empirical evidence of the effectiveness of dynamic parameters. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 155–156.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning.* MIT press.

Guerri, A. and Milano, M. (2004). Learning techniques for automatic algorithm portfolio selection. In *ECAI*, volume 16, page 475.

Gümüş, D. B., Özcan, E., Atkin, J., and Drake, J. H. (2023). An investigation of f-race training strategies for cross domain optimisation with memetic algorithms. *Information Sciences*, 619:153–171.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Haim, S. and Walsh, T. (2009). Restart strategy selection using machine learning techniques. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 312–325. Springer.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., and Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239.

Han, A. F.-W. and Cho, Y.-J. (2002). A gids metaheuristic approach to the fleet size and mix vehicle routing problem. In *Essays and surveys in metaheuristics*, pages 399–413. Springer.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer.

Hansen, P. and Mladenović, N. (1999). An introduction to variable neighborhood search. In *Meta-heuristics*, pages 433–458. Springer.

Hansen, P. and Mladenović, N. (2003). Variable neighborhood search. In *Handbook of metaheuristics*, pages 145–184. Springer.

Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2019). *Variable neighborhood search*. Springer.

Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Tibshirani, R., and Friedman, J. (2009). Unsupervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, pages 485–585.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE.

He, X., Zhao, K., and Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622.

He, Y., Yuen, S. Y., Lou, Y., and Zhang, X. (2019). A sequential algorithm portfolio approach for black box optimization. *Swarm and evolutionary computation*, 44:559–570.

Ho, S. C. and Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research*, 31(12):1947–1964.

Ho, W., Ho, G. T., Ji, P., and Lau, H. C. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering applications of artificial intelligence*, 21(4):548–557.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.

Homberger, J. and Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 37(3):297–318.

Hong, T.-P., Wang, H.-S., Lin, W.-Y., and Lee, W.-Y. (2002). Evolution of appropriate crossover and mutation operators in a genetic process. *Applied intelligence*, 16:7–17.

Hoos, H. H. (2008). Computer-aided design of high-performance algorithms. Technical report, Technical Report TR-2008-16, University of British Columbia, Department of ....

Hoos, H. H. (2011). Automated algorithm configuration and parameter tuning. In *Autonomous search*, pages 37–71. Springer.

Horvitz, E., Ruan, Y., Gomes, C., Kautz, H., Selman, B., and Chickering, M. (2001). A bayesian approach to tackling hard computational problems (preliminary report). *Electronic Notes in Discrete Mathematics*, 9:376–391.

Hottung, A. and Tierney, K. (2019). Neural large neighborhood search for the capacitated vehicle routing problem. *arXiv preprint arXiv:1911.09539*.

Hu, B. and Raidl, G. R. (2006). Variable neighborhood descent with self-adaptive neighborhood-ordering. In *Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics*.

Hutter, F., Hamadi, Y., Hoos, H. H., and Leyton-Brown, K. (2006). Performance prediction and automated tuning of randomized and parametric algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pages 213–228. Springer.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2010a). Automated configuration of mixed integer programming solvers. In *International Conference on Integration of Artificial Intelligence (AI) and Operations*

Research (OR) Techniques in Constraint Programming, pages 186–202. Springer.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2010b). Sequential model-based optimization for general algorithm configuration (extended version). *Technical Report TR-2010–10, University of British Columbia, Computer Science, Tech. Rep.*

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Stützle, T. (2009). ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306.

Hutter, F., Hoos, H. H., and Stützle, T. (2007). Automatic algorithm configuration based on local search. In *Aaai*, volume 7, pages 1152–1157.

Hutter, F., Xu, L., Hoos, H. H., and Leyton-Brown, K. (2014). Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111.

Ioannou, G., Kritikos, M., and Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, 52(5):523–537.

Jackson, W. G., Özcan, E., and John, R. I. (2018). Move acceptance in local search metaheuristics for cross-domain search. *Expert Systems with Applications*, 109:131–151.

Jeni, L. A., Cohn, J. F., and De La Torre, F. (2013). Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine association conference on affective computing and intelligent interaction*, pages 245–251. IEEE.

Jin, J., Crainic, T. G., and Løkketangen, A. (2012). A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*, 222(3):441–451.

Juan, A. A., Faulin, J., Ruiz, R., Barrios, B., and Caballé, S. (2010). The sr-gcws hybrid algorithm for solving the capacitated vehicle routing problem. *Applied Soft Computing*, 10(1):215–224.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., and Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100:234–245.

Kachitvichyanukul, V. et al. (2007). A particle swarm optimization for the capacitated vehicle routing problem. *International journal of logistics and SCM systems*, 2(1):50–55.

Kadioglu, S., Malitsky, Y., Sellmann, M., and Tierney, K. (2010). ISAC-Instance-specific algorithm configuration. In *ECAI*, volume 215, pages 751–756.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., and Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422.

Kemeny, J. G. and Snell, J. L. (1976). *Markov chains*. Springer-Verlag, New York.

Kendall, G., Cowling, P., and Soubeiga, E. (2002). Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, pages 667–671.

Kerschke, P., Hoos, H. H., Neumann, F., and Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30.

Khamassi, I. (2011). Ant-Q hyper heuristic approach applied to the cross-domain heuristic search challenge problems.

Kheiri, A. and Keedwell, E. (2015). A sequence-based selection hyper-heuristic utilising a hidden Markov model. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 417–424. ACM.

KhudaBukhsh, A. R. (2009). *SATenstein: Automatically building local search SAT solvers from components*. PhD thesis, University of British Columbia.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kontoravdis, G. and Bard, J. F. (1995). A grasp for the vehicle routing problem with time windows. *ORSA journal on Computing*, 7(1):10–23.

Kotary, J., Fioretto, F., Van Hentenryck, P., and Wilder, B. (2021). End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Kotthoff, L., Gent, I. P., and Miguel, I. (2011). A preliminary evaluation of machine learning in algorithm selection for search problems. In *Fourth Annual Symposium on Combinatorial Search*.

Kotthoff, L., Gent, I. P., and Miguel, I. (2012). An evaluation of machine learning in algorithm selection for search problems. *Ai Communications*, 25(3):257–270.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Citeseer.

Küçükoğlu, İ. and Öztürk, N. (2015). An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. *Computers & Industrial Engineering*, 86:60–68.

Kumar, S. N. and Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lagoudakis, M. G. and Littman, M. L. (2000). Algorithm selection using reinforcement learning. In *ICML*, pages 511–518.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300.

Laporte, G., Louveaux, F., and Mercure, H. (1992). The vehicle routing problem with stochastic travel times. *Transportation science*, 26(3):161–170.

Laporte, G., Ropke, S., and Vidal, T. (2014). Chapter 4: Heuristics for the vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 87–116. SIAM.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 63–66. Springer.

Layeb, A., Ammi, M., and Chikhi, S. (2013). A grasp algorithm based on new randomized heuristic for vehicle routing problem. *Journal of computing and information technology*, 21(1):35–46.

Lenstra, J. K. and Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27:2177–2185.

Li, B., Wu, G., He, Y., Fan, M., and Pedrycz, W. (2022). An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem. *IEEE/CAA Journal of Automatica Sinica*, 9(7):1115–1138.

Li, X. and Tian, P. (2006). An ant colony system for the open vehicle routing problem. In *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006. Proceedings 5*, pages 356–363. Springer.

Liefooghe, A., Jourdan, L., and Talbi, E.-G. (2011). A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. *European Journal of Operational Research*, 209(2):104–112.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269.

Lin, S.-W., Lee, Z.-J., Ying, K.-C., and Lee, C.-Y. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications*, 36(2):1505–1512.

Lindauer, M., Hoos, H., and Hutter, F. (2015). From sequential algorithm selection to parallel portfolio selection. In *International Conference on Learning and Intelligent Optimization*, pages 1–16. Springer.

Lissovoi, A., Oliveto, P. S., and Warwicker, J. A. (2017). On the runtime analysis of generalised selection hyper-heuristics for pseudo-boolean optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 849–856.

Lissovoi, A., Oliveto, P. S., and Warwicker, J. A. (2020a). How the duration of the learning period affects the performance of random gradient selection hyper-heuristics. In *AAAI*, pages 2376–2383.

Lissovoi, A., Oliveto, P. S., and Warwicker, J. A. (2020b). Simple hyper-heuristics control the neighbourhood size of randomised local search optimally for leadingones. *Evolutionary computation*, 28(3):437–461.

Liu, F., Lu, C., Gui, L., Zhang, Q., Tong, X., and Yuan, M. (2023). Heuristics for vehicle routing problem: A survey and recent advances. *arXiv preprint arXiv:2303.04147*.

Liu, F. and Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the tsp. *Expert Systems with Applications*, 36(3):6995–7001.

López, V., Fernández, A., García, S., Palade, V., and Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141.

Lopez-Ibanez, M. and Stutzle, T. (2012). The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). *Iterated local search*. Springer.

Lundy, M. and Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical programming*, 34(1):111–124.

Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. (2011). Non-model-based algorithm portfolios for SAT. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 369–370. Springer.

Malitsky, Y. and Sellmann, M. (2010). Stochastic offline programming. *International Journal on Artificial Intelligence Tools*, 19(04):351–371.

Mandic, D. and Chambers, J. (2001). *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley.

Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126. ICML United States.

Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Marín-Blázquez, J. G. and Schulenburg, S. (2003). A hyper-heuristic framework with xcs: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients. In *Learning classifier systems*, pages 193–218. Springer.

Marín-Blázquez, J. G. and Schulenburg, S. (2006). Multi-step environment learning classifier systems applied to hyper-heuristics. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1521–1528.

Marinakis, Y. (2012). Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8):6807–6815.

Marinakis, Y., Iordanidou, G.-R., and Marinaki, M. (2013). Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing*, 13(4):1693–1704.

Mascia, F., López-Ibánez, M., Dubois-Lacoste, J., and Stützle, T. (2013). From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In *International Conference on Learning and Intelligent Optimization*, pages 321–334. Springer.

Mazzeo, S. and Loiseau, I. (2004). An ant colony algorithm for the capacitated vehicle routing. *Electronic Notes in Discrete Mathematics*, 18:181–186.

McClymont, K. and Keedwell, E. (2011a). A single objective variant of the online selective Markov chain hyper-heuristic (MCHH-S).

McClymont, K. and Keedwell, E. C. (2011b). Markov chain hyper-heuristic (MCHH) an online selective hyper-heuristic for multi-objective continuous problems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 2003–2010.

Meignan, D., Koukam, A., and Créput, J.-C. (2010). Coalition-based meta-

heuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism. *Journal of Heuristics*, 16(6):859–879.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, 37(11):1886–1898.

Meng, W. and Qu, R. (2021). Automated design of search algorithms: Learning on algorithmic components. *Expert Systems with Applications*, 185:115493.

Meng, W. and Qu, R. (2023a). Automated design of local search algorithms: Predicting algorithmic components with lstm. *Expert Systems with Applications*, page 121431.

Meng, W. and Qu, R. (2023b). Sequential rule mining for automated design of meta-heuristics. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 1727–1735.

Messelis, T. and De Causmaecker, P. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 233(3):511–528.

Mester, D., Bräysy, O., and Dullaert, W. (2007). A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications*, 32(2):508–517.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mısır, M. and Sebag, M. (2017). Alors: An algorithm recommender system. *Artificial Intelligence*, 244:291–314.

Misir, M., Verbeeck, K., De Causmaecker, P., and Berghe, G. V. (2010). Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

Mısır, M., Verbeeck, K., De Causmaecker, P., and Berghe, G. V. (2012). An intelligent hyper-heuristic framework for chesc 2011. In *International Conference on Learning and Intelligent Optimization*, pages 461–466. Springer.

Mısır, M., Verbeeck, K., De Causmaecker, P., and Berghe, G. V. (2013). An investigation on the generality level of selection hyper-heuristics under different empirical conditions. *Applied Soft Computing*, 13(7):3335–3353.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mole, R. and Jameson, S. (1976). A sequential route-building algorithm employing a generalised savings criterion. *Journal of the Operational Research Society*, 27(2):503–511.

Moscato, P. et al. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826(1989):37.

Na, B., Jun, Y., and Kim, B.-I. (2011). Some extensions to the sweep algorithm. *The International Journal of Advanced Manufacturing Technology*, 56:1057–1067.

Nagata, Y., Bräysy, O., and Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & operations research*, 37(4):724–737.

Nammous, M. K. and Saeed, K. (2019). Natural language processing: Speaker, language, and gender identification with LSTM. In *Advanced Computing and Systems for Security*, pages 143–156. Springer.

Nanry, W. P. and Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121.

Ngueveu, S. U., Prins, C., and Calvo, R. W. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 37(11):1877–1885.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A. J., Petrovic, S., et al. (2012). Hyflex: A benchmark framework for cross-domain heuristic search. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 136–147. Springer.

Or, I. (1976). *TRAVELING SALESMAN TYPE COMBINATORIAL PROBLEMS AND THEIR RELATION TO THE LOGISTICS OF REGIONAL BLOOD BANKING.* Northwestern University.

Ortiz-Bayliss, J. C., Terashima-Marín, H., and Conant-Pablos, S. E. (2013a). A supervised learning approach to construct hyper-heuristics for constraint satisfaction. In *Mexican Conference on Pattern Recognition*, pages 284–293. Springer.

Ortiz-Bayliss, J. C., Terashima-Marín, H., and Conant-Pablos, S. E. (2013b). Using learning classifier systems to design selective hyper-heuristics for constraint satisfaction problems. In *2013 IEEE Congress on Evolutionary Computation*, pages 2618–2625. IEEE.

Ortiz-Bayliss, J. C., Terashima-Marín, H., Ross, P., and Conant-Pablos, S. E. (2011). Evolution of neural networks topologies and learning parameters to produce hyper-heuristics for constraint satisfaction problems. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 261–262.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41:421–451.

Osman, I. H. and Wassan, N. A. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, 5(4):263–285.

Özcan, E., Bilgin, B., and Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23.

O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., and O'Sullivan, B. (2008). Using case-based reasoning in an algorithm portfolio for con-

straint solving. In *Irish conference on artificial intelligence and cognitive science*, pages 210–216.

Pappa, G. L., Ochoa, G., Hyde, M. R., Freitas, A. A., Woodward, J., and Swan, J. (2014). Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, 15:3–35.

Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G., and Prastacos, G. P. (2008). A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, 14(5):425–455.

Pellegrini, P. and Birattari, M. (2006). The relevance of tuning the parameters of metaheuristics. In *Technical Report*. Technical report, IRIDIA, Université Libre de Bruxelles.

Peng, F., Tang, K., Chen, G., and Yao, X. (2010). Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on evolutionary computation*, 14(5):782–800.

Penna, P. H. V., Subramanian, A., and Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232.

Petke, J., Haraldsson, S. O., Harman, M., Langdon, W. B., White, D. R., and Woodward, J. R. (2017). Genetic improvement of software: a comprehensive survey. *IEEE Transactions on Evolutionary Computation*, 22(3):415–432.

Peya, Z. J., Akhand, M., Sultana, T., and Rahman, M. H. (2019). Distance based sweep nearest algorithm to solve capacitated vehicle routing

problem. *International Journal of Advanced Computer Science and Applications*, 10(10):259–264.

Pihera, J. and Musliu, N. (2014). Application of machine learning to algorithm selection for tsp. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pages 47–54. IEEE.

Pillay, N. (2021). Automated design (autodes): Current trends and future research directions. *Automated Design of Machine Learning and Search Algorithms*, pages 185–187.

Pillay, N. and Beckedahl, D. (2017). EvoHyp-a Java toolkit for evolutionary algorithm hyper-heuristics. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2706–2713. IEEE.

Pillay, N. and Qu, R. (2018). *Hyper-Heuristics: Theory and Applications*. Springer.

Pillay, N., Qu, R., Pillay, N., and Qu, R. (2018a). Vehicle routing problems. *Hyper-Heuristics: Theory and Applications*, pages 51–60.

Pillay, N., Qu, R., Srinivasan, D., Hammer, B., and Sorensen, K. (2018b). Automated design of machine learning and search algorithms [guest editorial]. *IEEE Computational intelligence magazine*, 13(2):16–17.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435.

Polacek, M., Hartl, R. F., Doerner, K., and Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, 10:613–627.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm intelligence*, 1:33–57.

Pomerol, J.-C. (1997). Artificial intelligence and human decision making. *European Journal of Operational Research*, 99(1):3–25.

Potvin, J.-Y. and Bengio, S. (1996). The vehicle routing problem with time windows part ii: genetic search. *INFORMS journal on Computing*, 8(2):165–172.

Potvin, J.-Y., Duhamel, C., and Guertin, F. (1996). A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6:345–355.

Potvin, J.-Y. and Rousseau, J.-M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340.

Potvin, J.-Y. and Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

Qi, Y., Hou, Z., Li, H., Huang, J., and Li, X. (2015). A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. *Computers & Operations Research*, 62:61–77.

Qu, R. (2021a). A general model for automated algorithm design. In *Automated Design of Machine Learning and Search Algorithms*, pages 29–43. Springer.

Qu, R. (2021b). Recent developments of automated machine learning and search techniques. In *Automated Design of Machine Learning and Search Algorithms*, pages 1–9. Springer.

Qu, R., Kendall, G., and Pillay, N. (2020). The general combinatorial optimization problem: Towards automated algorithm design. *IEEE Computational Intelligence Magazine*, 15(2):14–23.

Ramos, I. C., Goldbarg, M. C., Goldbarg, E. G., and Neto, A. D. D. (2005). Logistic regression for parameter tuning on an evolutionary algorithm. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1061–1068. IEEE.

Rao, G., Huang, W., Feng, Z., and Cong, Q. (2018). LSTM with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57.

Reimann, M., Doerner, K., and Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591.

Remde, S., Cowling, P., Dahal, K., Colledge, N., and Selensky, E. (2012). An empirical study of hyperheuristics for managing very large sets of low level heuristics. *Journal of the operational research society*, 63(3):392–405.

Roberts, M. and Howe, A. E. (2006). Directing a portfolio with learning. In *AAAI 2006 Workshop on Learning for Search*, pages 129–135.

Rochat, Y. and Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics*, 1:147–167.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Ross, P., Schulenburg, S., Marín-Bläzquez, J. G., and Hart, E. (2002). Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 942–948.

Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L. M., Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B., et al. (2002). A comparison of the performance of different metaheuristics on the timetabling problem. In *International conference on the practice and theory of automated timetabling*, pages 329–351. Springer.

Rousseau, L.-M., Gendreau, M., and Pesant, G. (2002). Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of heuristics*, 8:43–58.

Schirmer, A. (2000). Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics (NRL)*, 47(3):201–222.

Schulze, J. and Fahle, T. (1999). A parallel algorithm for the vehicle routing problem with time window constraints. *annals of operations research*, 86(0):585–607.

Shafi, K., Bender, A., and Abbass, H. A. (2012). Multi objective learning classifier systems based hyperheuristics for modularised fleet mix problem. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 381–390. Springer.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 46.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98: 4th International Conference, CP98 Pisa, Italy, October 26–30, 1998 Proceedings 4*, pages 417–431. Springer.

SINTEF. VRPTW benchmark problems, on the sintef transport optimisation portal. https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/. Published April 18, 2008.

SINTEF. VRPTW benchmark problems, on the sintef transport optimisation portal. https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark/1000-customers/. Published April 18, 2008.

Skydt, M. R., Bang, M., and Shaker, H. R. (2021). A probabilistic sequence classification approach for early fault prediction in distribution grids using long short-term memory neural networks. *Measurement*, 170:108691.

Smagulova, K. and James, A. P. (2020). Overview of long short-term memory neural networks. In *Deep Learning Classifiers with Memristive Networks*, pages 139–153. Springer.

Smith, S. F., Lassila, O., and Becker, M. (1996). Configurable, mixed-initiative systems for planning and scheduling. *Advanced Planning Technology*, pages 235–241.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Soria-Alcaraz, J. A., Ochoa, G., Sotelo-Figeroa, M. A., and Burke, E. K. (2017). A methodology for determining an effective subset of heuris-

tics in selection hyper-heuristics. *European Journal of Operational Research*, 260(3):972–983.

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *International conference on extending database technology*, pages 1–17. Springer.

Stone, C., Hart, E., and Paechter, B. (2021). A cross-domain method for generation of constructive and perturbative heuristics. *Automated Design of Machine Learning and Search Algorithms*, pages 91–107.

Sui, J., Ding, S., Liu, R., Xu, L., and Bu, D. (2021). Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning. In *Asian Conference on Machine Learning*, pages 1301–1316. PMLR.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Swan, J., Özcan, E., and Kendall, G. (2011). Hyperion–a recursive hyper-heuristic framework. In *International Conference on Learning and Intelligent Optimization*, pages 616–630. Springer.

Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293.

Taillard, É. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.

Tan, K. C., Lee, L. H., Zhu, Q., and Ou, K. (2001). Heuristic methods for vehicle routing problem with time windows. *Artificial intelligence in Engineering*, 15(3):281–295.

Tan, S.-Y. and Yeh, W.-C. (2021). The vehicle routing problem: State-of-the-art classification and review. *Applied Sciences*, 11(21):10295.

Tang, K., Peng, F., Chen, G., and Yao, X. (2014). Population-based algorithm portfolios with automated constituent algorithms selection. *Information Sciences*, 279:94–104.

Terashima-Marín, H., Flores-Alvarez, E., and Ross, P. (2005). Hyper-heuristics and classifier systems for solving 2d-regular cutting stock problems. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 637–643.

Thabtah, F. and Cowling, P. (2008). Mining the data from a hyperheuristic approach using associative classification. *Expert systems with applications*, 34(2):1093–1101.

Thangiah, S. R. (1993). *Vehicle routing with time windows using genetic algorithms*. Citeseer.

Thangiah, S. R., Nygard, K. E., and Juell, P. L. (1991). Gideon: A genetic algorithm system for vehicle routing with time windows. In *Proceedings The Seventh IEEE Conference on Artificial Intelligence Application*, pages 322–323. IEEE Computer Society.

Tian, H., Wang, X.-F., Mohammad, M. A., Gou, G.-Y., Wu, F., Yang, Y., and Ren, T.-L. (2018). A hardware Markov chain algorithm realized in a single device for machine learning. *Nature communications*, 9(1):4305.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Turky, A., Sabar, N. R., Dunstall, S., and Song, A. (2020). Hyper-heuristic local search for combinatorial optimisation problems. *Knowledge-Based Systems*, page 106264.

Tyasnurita, R., Özcan, E., and John, R. (2017). Learning heuristic selection using a time delay neural network for open vehicle routing. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1474–1481. IEEE.

Tyasnurita, R., Özcan, E., Shahriar, A., and John, R. (2015). Improving performance of a hyper-heuristic using a multilayer perceptron for vehicle routing.

Urbanowicz, R. J. and Moore, J. H. (2009). Learning classifier systems: a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009.

Van Breedam, A. (1994). *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints*. RUCA.

Van Breedam, A. (1995). Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86(3):480–490.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vesselinova, N., Steinert, R., Perez-Ramirez, D. F., and Boman, M. (2020). Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 8:120388–120416.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, 40(1):475–489.

Vrakas, D., Tsoumakas, G., Bassiliades, N., and Vlahavas, I. P. (2003). Learning rules for adaptive planning. In *ICAPS*, pages 82–91.

Walker, J. D., Ochoa, G., Gendreau, M., and Burke, E. K. (2012). Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework. In *International Conference on Learning and Intelligent Optimization*, pages 265–276. Springer.

Wan, H., Guo, S., Yin, K., Liang, X., and Lin, Y. (2020). CTS-LSTM: LSTM-based neural networks for correlatedtime series prediction. *Knowledge-Based Systems*, 191:105239.

Wang, J., Han, J., and Li, C. (2007). Frequent closed sequence mining without candidate maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1042–1056.

Wark, P. and Holt, J. (1994). A repeated matching heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, 45(10):1156–1167.

Wassan, N. A., Wassan, A. H., and Nagy, G. (2008). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of combinatorial optimization*, 15(4):368–386.

Willard, J. (1989). Vehicle routing using r-optimal tabu search. *Master's thesis, The ManagementSchool, ImperialCollege, London*.

Wong, R. T. (1983). Combinatorial optimization: Algorithms and complex-

ity (Christos H. Papadimitriou and Kenneth Steiglitz). *SIAM Review*, 25(3):424.

Woodward, J. R., Johnson, C. G., and Brownlee, A. E. (2016). Connecting automatic parameter tuning, genetic programming as a hyper-heuristic, and genetic improvement programming. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 1357–1358.

Wu, Y., Song, W., Cao, Z., Zhang, J., and Lim, A. (2021). Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9):5057–5069.

Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48.

Xu, J. and Kelly, J. P. (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation science*, 30(4):379–393.

Xu, L., Hoos, H., and Leyton-Brown, K. (2010). Hydra: Automatically configuring algorithms for portfolio-based selection. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*. Citeseer.

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2007). SATzilla-07: the design and analysis of an algorithm portfolio for SAT. In *International Conference on Principles and Practice of Constraint Programming*, pages 712–727. Springer.

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2008). SATzilla: portfolio-based algorithm selection for SAT. *Journal of artificial intelligence research*, 32:565–606.

Yates, W. B. and Keedwell, E. C. (2017). Offline learning for selection

hyper-heuristics with Elman networks. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 217–230. Springer.

Yellow, P. (1970). A computational modification to the savings method of vehicle scheduling. *Journal of the Operational Research Society*, 21(2):281–283.

Yi, W., Qu, R., Jiao, L., and Niu, B. (2022). Automated design of metaheuristics using reinforcement learning within a novel general search framework. *IEEE Transactions on Evolutionary Computation*.

Yuen, S. Y., Lou, Y., and Zhang, X. (2019). Selecting evolutionary algorithms for black box design optimization problems. *Soft Computing*, 23(15):6511–6531.

Zeyer, A., Bahar, P., Irie, K., Schlüter, R., and Ney, H. (2019). A comparison of transformer and LSTM encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE.

Zhao, Q. and Bhowmick, S. S. (2003). Sequential pattern mining: A survey. *ITechnical Report CAIS Nayang Technological University Singapore*, 1(26):135.

Zhao, Y. W., Wu, B., Wang, W., Ma, Y. L., Wang, W., and Sun, H. (2004). Particle swarm optimization for vehicle routing problem with time windows. In *Materials Science Forum*, volume 471, pages 801–805. Trans Tech Publ.

Zheng, J., He, K., Zhou, J., Jin, Y., and Li, C.-M. (2021). Combining reinforcement learning with lin-kernighan-helsgaun algorithm for the traveling salesman problem. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 12445–12452.

Zhou, J., Wu, Y., Cao, Z., Song, W., Zhang, J., and Chen, Z. (2023). Learning large neighborhood search for vehicle routing in airport ground handling. *IEEE Transactions on Knowledge and Data Engineering*.

Zhou, L. (2013). Performance of corporate bankruptcy prediction models on imbalanced dataset: The effect of sampling methods. *Knowledge-Based Systems*, 41:16–25.