Sequential Rule Mining for Automated Design of Meta-heuristics

Weiyao Meng
School of Computer Science
University of Nottingham
Nottingham, UK
Weiyao.Meng@nottingham.ac.uk

Rong Qu
School of Computer Science
University of Nottingham
Nottingham, UK
Rong.Qu@nottingham.ac.uk

ABSTRACT

With a recently defined AutoGCOP framework, the design of local search algorithms can be defined as the composition of the basic elementary algorithmic components. These compositions into the best algorithms thus retain useful knowledge of effective algorithm design. This paper investigates effective algorithmic compositions with sequential rule mining techniques to discover valuable knowledge in algorithm design. With the collected effective algorithmic compositions, sequential rules of basic algorithmic components are extracted and further analysed to automatically compose basic algorithmic components within the general AutoGCOP framework to develop new effective meta-heuristics. The sequential rules present superior performance in composing the basic algorithmic components for solving the benchmark vehicle routing problems with time window constraints, demonstrating its effectiveness in designing new algorithms automatically.

CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; **Rule learning**; • **Theory of computation** → **Design and analysis of algorithms**.

KEYWORDS

Data Mining, Sequential Rule Mining, Automated Algorithm Design, Meta-heuristics, Vehicle Routing Problems

ACM Reference Format:
Weiyao Meng and Rong Qu. 2023. Sequential Rule Mining for Automated Design of Meta-heuristics. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583133.3596303

1 INTRODUCTION

In solving complex combinatorial optimisation problems (COPs), the design and development of effective meta-heuristics present a challenge for researchers due to the time-consuming experiments and experience required in decision-making [21]. The automated design of search algorithms for solving COPs removes the heavy burden on researchers and allows the exploration of a larger scope of unseen effective algorithms [19].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0120-7/23/07…$15.00
https://doi.org/10.1145/3583133.3596303

Recently, the problem of algorithm design is formally defined as a COP with a new model named the General Combinatorial Optimisation Problem (GCOP) [22]. A general AutoGCOP framework is built in [19] to support the automated design of local search algorithms by combining basic elementary algorithmic components in GCOP, supporting more flexible exploration of a larger space of new unseen local search algorithms. Effective patterns from different algorithmic compositions can be discovered within AutoGCOP to design new effective algorithms [19].

A large amount of data on the components selected and composed during the search process can be collected and further analysed to learn useful knowledge for algorithm design [17]. Many machine learning methods have been used, for example, for predicting basic algorithmic components with a Markov Chain model within AutoGCOP [19], and predicting low-level heuristics in hyper-heuristics with linear regression [2], decision tree [2], k-means classification [3], neural networks [31], [30], and associative classification [29]. The effectiveness of the Markov Chain model and the associative classification method indicates the importance of capturing the sequential relationships between algorithmic components in algorithm design.

However, the knowledge captured and embedded in the learning models is implicit, thus hard to further analyse 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 paper investigates the data generated in the search process as a new data mining application, aiming to explore the potentially useful knowledge hidden in the algorithmic compositions and interpret the knowledge to support effective automated algorithm design. The widely investigated Vehicle Routing Problems with Time Window constraints (VRPTW) are used as the domain problem. In particular, we employ sequential rule mining to extract sequential rules of basic operators from effective algorithmic compositions for solving benchmark 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. A novel Sequential Rule-based GCOP method (denoted as SeqRuleGCOP) is proposed to automatically compose algorithmic components within AutoGCOP. The extracted sequential rules of basic operators are evaluated with SeqRuleGCOP for solving VRPTW instances. In comparison with other GCOP methods and the Variable Neighbourhood Descent (VND), SeqRuleGCOP shows superior performance, demonstrating the effectiveness of the sequential rules for automated algorithm design.

In the rest of the paper, Section 2 presents related work on the GCOP framework and sequential rule mining. Section 3 describes

the algorithmic composition data investigated in this study. Section 4 presents the applied sequential rule mining technique and the proposed SeqRuleGCOP method. Section 5 discusses the experimental setup and result analysis, followed by conclusions in Section 6.

2 RELATED WORKS

2.1 An overview of the general automated composition framework

In the newly defined GCOP model in [22], various search algorithms are seen as the compositions of a finite set A of elementary algorithmic components $a \in A$ during the search. Algorithm design can thus be defined as a COP with decision variables taking values a from domain A. The solution space of GCOP consists of algorithmic compositions c upon a. Each c represents a new algorithm for solving optimisation problems p, i.e., a solution s for p is obtained by a corresponding algorithmic composition c, $c \rightarrow s$. The objective of GCOP is to search for the optimal c^* which produces the optimal s^* for p, i.e., $c^* \rightarrow s^*$.

A general framework named AutoGCOP [19] (Algorithm 1) has been built to support the automated composition of the elementary algorithmic components a. In AutoGCOP, the common procedures of various local search meta-heuristics are encapsulated as the most basic processes to compose the following three categories of algorithmic components in an extended GCOP model:

- Operators $o_i \in A_o$: modify values of the decision variables in solution s_1 to generate a new solution s_2 in the search space of p.
- Acceptance criteria $a_j \in A_a$: determine if and how s_2 is accepted in the search.
- Termination criteria $t_k \in A_t$: control when to terminate a loop.

With the general AutoGCOP framework, various local search algorithms in the literature [4] can be defined in a unified template by composing specific algorithmic components [19]. In other words, the existing algorithms can be seen as specific GCOP solutions composed manually into local search algorithms.

In the optimisation process for solving GCOP, compositions c of a for solving p, i.e., the design of various search algorithms, can be then obtained automatically within AutoGCOP. This large number of new algorithms generated from AutoGCOP presents a considerable amount of data on algorithm design. Further analysis of the optimal or most effective algorithmic compositions may lead to new knowledge and a deeper understanding of algorithm design [22], some of which may be difficult to identify by human experts.

2.2 Sequential rule mining

Sequential data consists of sequences of items that are associated with time-related attributes [34], thus is of high complexity. One of the great interests of research on sequential data concerns the specific order of the occurrences of the items [34]. Initiated by [1], sequential pattern mining is proposed as the problem of mining interesting sub-sequences in a set of sequences [11]. Numerous sequential pattern mining algorithms have been developed to search for sequential patterns efficiently with different strategies and data structures [11].

Algorithm 1 : The general AutoGCOP framework

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

```
1:  procedure Construction
2:      s ← An empty solution for p;
3:      t_kcon ← Select(A_tconstruct);
4:      while t_kcon is not met do
5:          o_i ← Select(A_oconstruct);
6:          s ← ApplyOperator(o_i, s);
7:      end while
8:  end procedure
9:
10: procedure Improvement
11:     t_kmain ← Select(A_timprove);
12:     while t_kmain is not met do
13:         t_kinner ← Select(A_tinner);
14:         while t_kinner is not met do
15:             o_i ← Select(A_oimprove);
16:             a_j ← Select(A_a);
17:             s_new ← ApplyOperator(o_i, s);
18:             s ← ApplyAcceptance(a_j, s_new, s);
19:             s_best ← Update the best-recorded solution;
20:         end while
21:     end while
22: end procedure
```

For sequential patterns, although a sub-sequence occurs frequently (e.g., {X, Y}), one of the items ({X}) might not have a high possibility to be followed by the other items ({Y}) 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 [11].

Sequential rule mining is a variation of sequential pattern mining to address the above-mentioned limitation with more accurate sequence prediction. According to the definition of sequential rule mining [12], 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 (sets of I), i.e., $s = I_1, I_2, \ldots, I_z$. 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 [12].

Two measures have been defined for a sequential rule [12] 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.

A variety of algorithms have been proposed for sequential rule mining, including the apriori-based algorithm CMRules [8], rule

growth-based algorithm RuleGrowth [12], and equivalence class-based algorithm ERMiner [9], etc. 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 the others [13]. 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 [13] for mining the top-k sequential rules and TNS [14] for mining the non-redundant top-k sequential rules.

In this study, a large amount of data on effective compositions of basic algorithmic components is collected consistently within AutoGCOP, supporting systematic analysis to identify new knowledge towards automated algorithm design. Those effective algorithmic compositions can be seen as a sequential database for sequential rule mining. To the best of our knowledge, this is the first study on sequential rule mining techniques in the investigation of the automated design of search algorithms.

3 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) [19] 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 data mining.

3.1 The VRPTW problem

In operation research, VRP is one of the most investigated COPs [33]. The basic VRP [7] concerns the routing of a set of vehicles to serve customer delivery demands while minimising the total travel costs serving all the customers. The VRPTW, one of the most widely studied variants, concerns the constraints that customers must be served within specified time intervals [5].

The VRPTW investigated in this study considers the dual objectives of minimising the number of vehicles (NV) and minimising the total travel distance (TD), as shown in Equation (1), where c is set to 1000 empirically and widely used in the literature [32].

$$c \times NV + TD \tag{1}$$

The data collection has been conducted on the benchmark Solomon 100 data set as shown in Table 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 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

3.2 Data collection within the general AutoGCOP framework

To explore insights on effective algorithm compositions, this study focuses on the most basic operators o_i as shown in Table 2 within the general AutoGCOP framework. These basic operators are different in terms of the operation, leading to different performances for solving VRPTW [19].

Table 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^{in}_{xchg}	Small	1-route	Exchange
o^{bw}_{xchg}	Small	2-route	Exchange
o^{in}_{ins}	Small	1-route	Insert
o^{bw}_{ins}	Small	2-route	Insert
o_{rr}	Large	n-route	Ruin-recreate

To collect algorithmic compositions with AutoGCOP, this study utilises a random GCOP method (i.e., RN-GCOP) [19] to flexibly compose the basic operators in Table 2 for solving the selected VRPTW instances in Table 1. The information in each search iteration is recorded, including the current index of iteration, the index of the applied o_i, and the best-found solution after using o_i.

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.

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. 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.

4 AUTOMATED ALGORITHM DESIGN WITH SEQUENTIAL RULE MINING

With the data collected, the sequential rule mining technique is applied to acquire knowledge hidden in the collected algorithmic compositions to support algorithm design. A sequential rule-based GCOP method (SeqRuleGCOP) is proposed to utilise the extracted rules in the automated composition of basic operators for problem-solving.

4.1 Sequential rule mining on algorithmic compositions

As described in Section 3.2, a large number of algorithmic compositions are processed into a collection of operator sequences that obtains elite performance improvement in problem-solving. This collection can be seen as a sequence database, where each operator sequence s_o is an ordered list of operator sets (sets of operators $o_i \in A_o$). This study aims at investigating the sequential rules of common basic operators in elite operator sequences.

A sequential rule of operators can be represented in the form of $X_o \Rightarrow Y_o$, describing a relationship between two sets of operators $X_o, Y_o \subseteq A_o$. The interpretation of $X_o \Rightarrow Y_o$ is that if operators in X_o occur in a sequence, the operators in Y_o are likely to occur afterwards (at any position after X_o) in the same sequence within a certain length of l. Each rule can be measured by sequential support and confidence values (denoted by sup and $conf$, respectively).

The TNS sequential rule mining algorithm [14] 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 [10]. The extracted sequential rules are further investigated to explore the behaviour of basic operators and the effectiveness in algorithm design.

4.2 SeqRuleGCOP: automated composition with sequential rules

This study proposes a novel method named SeqRuleGCOP to support the automated composition of algorithmic components using sequential rules within AutoGCOP. SeqRuleGCOP takes an additional input with AutoGCOP, i.e., a set of sequential rules of basic operators S_o. Each $s_o \in S_o$ is an ordered list of operator sets (sets of operators $o_i \in A_{o_{improve}}$) and associated with a support value sup and a confidence value $conf$.

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 $s'_o \in S_o$ with the roulette wheel selection strategy according to $conf$. Then, the $Select(A_{o_{improve}})$ procedure takes each $o_i \in s'_o$ as the output o_i.

The proposed SeqRuleGCOP method supports the automated design of new and unseen algorithms by 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.

5 EXPERIMENTAL STUDIES

The experimental studies aim to address the following two research issues, 1) analysing the extracted sequential rules of o_i, and 2) assessing the extracted sequential rules in the automated composition of o_i with the proposed SeqRuleGCOP method.

In Section 5.1, 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.

Section 5.2 investigates the performance of the proposed SeqRuleGCOP method for solving different types of VRPTW instances. In comparison, a simple baseline RN-GCOP method and a Variable Neighbourhood Descent (VND) algorithm are tested. RNGCOP and VND do not learn from the operator sequences. The comparison aims to show the effectiveness of the sequential rules acquired from offline data mining in the automated composition of basic o_i for problem-solving.

5.1 Findings of sequential rules for algorithm design

5.1.1 General sequential rules of basic operators. Table 3 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 3: Top 10 sequential rules for the datasets 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 they are in the form of $X_o \rightarrow Y_o$, while X_o mainly 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 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}.

Considering the different operations of each basic o_i, they have different impacts on the optimization objectives of VRPTW. 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, this study categorises the basic $o_i \in A_o$ into three sets, as shown in Table 4.

With these new sets in Table 4, a general sequential rule of basic operators 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.

When discussing the features of operators, the literature usually focuses on the neighbourhood size (cardinality) [16] and operation types, e.g., either they are intensifiers or diversifiers [18], or the type of mutation, local search or evolutionary operators [32]. The commonly occurring frequent 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 for algorithm design by using data mining within the general AutoGCOP framework.

5.1.2 Impact of instance features to sequential rules. 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 5 and Table 6 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 5, 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, 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}). 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^{bw}_{ins}).

5.2 Effectiveness of sequential rules for automated composition

5.2.1 Experiment settings. The experiment studies evaluate the performance of the extracted sequential rules on the VRPTW instances, covering different instance characteristics. The Solomon 100 customer datasets in Table 1 are used to validate the effectiveness of the extracted rules, while another collection of the Solomon 100 customer datasets [27] in Table 7 is used to test their generality for solving new instances.

The study mainly focuses on the investigations on the basic $o_i \in A_{o_{improve}}$ in Table 2 which are 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.

5.2.2 Performance comparison against RN-GCOP. 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. In comparison, both RN-GCOP 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 7 shows the overall better performance of SeqRuleGCOP against RN-GCOP on the selected VRPTW instances except in two instances, where RN-GCOP achieves better AVG and BEST values. This supports the extracted frequent sequential rules contain useful patterns for automatically designing a search algorithm within AutoGCOP, particularly for solving Type-R and Type-C instances with statistical significance (measured by Mann–Whitney–Wilcoxon test with $p < 0.05$ and indicated by * in all the tables of results).

Similar observations can be reached regarding the comparison between SeqRuleGCOP against RN-GCOP on the new VRPTW instances in Table 8, i.e., SeqRuleGCOP achieves overall better performance compared to RN-GCOP. This verifies that the extracted frequent sequential rules contain general and useful knowledge which can be used to automatically design a search algorithm within AutoGCOP.

5.2.3 Performance comparison against VND. 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) [15]. It is based on the systematic change in a set of neighbourhood structures during the search process. The change of neighbourhoods is performed in a deterministic way (usually manually specified from smaller to larger ones), aiming to escape from local optimum [16].

Within AutoGCOP, various well-known local search-based metaheuristics, such as VNS, can be modelled in a unified template by composing specific algorithmic components in the Improvement

Table 4: 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_o^1	$\{o_{xchg}^{in}, o_{xchg}^{bw}, o_{ins}^{in}\}$	No	Small
Type-2 A_o^2	$\{o_{ins}^{bw}\}$	Small	Small
Type-3 A_o^3	$\{o_{rr}\}$	Large	Large

Table 5: Top 10 sequential rules for the datasets 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:	conf:	Rules	sup:	conf:	Rules	sup:	conf:
$o_{ins}^{in} \rightarrow o_{rr}$	**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

Table 6: Top 10 sequential rules for the datasets 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:	conf:	Rules	sup:	conf:
$o_{xchg}^{in} \rightarrow o_{ins}^{bw}$	**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

procedure [19]. With the basic $o_i \in A_{o_{improve}}$ ordered according to their impact as $[o_{xchg}^{in}, o_{xchg}^{bw}, o_{ins}^{in}, o_{ins}^{bw}, o_{rr}]$, 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;

Table 7: Comparison between SeqRuleGCOP against RN-GCOP for effectiveness validation.

Instances	Best-known solutions in the literature	RN-GCOP				SeqRuleGCOP			
		AVG	SD	BEST	GAP	AVG	SD	BEST	GAP
C102	10828.94[23]	13126.54	269.49	12875.06	0.189	**12899.69***	**165.65**	**12540.88**	**0.158**
C202	3591.56[23]	3787.12	49.30	3727.07	0.038	**3745.93***	**27.48**	**3707.10**	**0.032**
R102	18486.12[23]	21033.67	**39.80**	20976.51	0.135	**20658.85***	464.99	**20040.94**	**0.084**
R202	4191.70[24]	**5542.48**	28.22	**5502.88**	**0.313**	5544.64	**19.48**	5509.80	0.314
RC102	13554.75[28]	17810.05	**401.25**	**16925.97**	**0.249**	**17688.55**	449.91	17070.83	0.259
RC202	4365.65[6]	5754.53	**22.44**	5723.58	0.311	**5730.50**	37.47	**5684.91**	**0.302**

Table 8: Comparison between SeqRuleGCOP against RN-GCOP for generality evaluation.

Instances	Best-known solutions in the literature	RN-GCOP				SeqRuleGCOP			
		AVG	SD	BEST	GAP	AVG	SD	BEST	GAP
C103	10,828.06[23]	12,364.31	416.60	**11,738.51**	**0.084**	**12,042.12**	**229.02**	11,745.85	0.085
C203	3,591.17[23]	4,502.51	514.51	3,881.98	0.081	**4,296.84**	**512.80**	**3,849.52**	**0.072**
R107	11,104.66[25]	14,564.69	27.27	14,520.74	0.308	**14,544.92**	**23.53**	**14,518.62**	**0.307**
R208	2,726.82[20]	4,087.51	**16.36**	4,055.09	0.487	**4,074.72**	21.87	**4,017.37**	**0.473**
RC103	12,261.67[26]	**14,881.08**	**297.94**	14,691.40	0.198	15,216.38	519.09	**14,665.19**	**0.196**
RC203	4,049.62[6]	4,784.47	355.59	4,539.13	0.121	**4,595.81**	**45.08**	**4,523.08**	**0.117**

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

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 $o_i \in A_{o_{improve}}$.

Table 9 presents the overall better performance of SeqRuleGCOP against VND, 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 10 shows SeqRuleGCOP performs better than VND in terms of AVG and SD. 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.

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.

6 CONCLUSIONS

Recently more machine learning techniques have been investigated 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 paper investigated the data of effective algorithm designs with data mining techniques to gain insightful and interpretive knowledge from the effective algorithmic compositions with 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 most 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. The analysis of the common sequential behaviour of the basic operators for solving different types of VRPTW instances 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.

In addition, the general sequential rules of basic operators are applied in the automated composition of basic operators within AutoGCOP. Experiment results show the effectiveness of the sequential rules for solving VRPTW instances in comparison with different methods.

This study is good evidence showing the information collected over the search run(s) can be useful for identifying hidden algorithm design by offline analysis of the search process. Particularly, this

Table 9: 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[23]	12,613.48	475.57	**12,157.42**	**0.123**	**12,550.22**	**62.37**	12,447.53	0.149
C202	3591.56[23]	5,269.61	707.78	4,659.64	0.297	**3,669.13***	**9.12**	**3,652.54**	**0.017**
R102	18486.12[23]	20,728.66	1,011.19	19,663.67	0.064	**19,836.42**	**280.07**	**18,998.78**	**0.028**
R202	4191.7[24]	8,257.50	1,108.62	6,338.63	0.512	**5,459.83***	**16.39**	**5,429.78**	**0.295**
RC102	13554.75[28]	17,277.48	692.83	**16,630.28**	**0.227**	**16,892.99**	**60.19**	16,796.01	0.239
RC202	4365.65[6]	8,944.58	935.99	7,540.14	0.727	**5,645.09***	**31.94**	**5,583.99**	**0.279**

Table 10: 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[23]	12,215.85	744.04	**11,069.86**	**0.022**	**11,602.83***	**82.58**	11,484.92	0.061
C203	3,591.17[23]	5,414.02	709.79	4,701.21	0.309	**3,738.50***	**31.10**	**3,708.68**	**0.033**
R107	11,104.66[25]	14,784.25	736.35	**13,223.33**	**0.191**	**14,358.96**	**289.68**	13,488.79	0.215
R208	2,726.82[20]	5,834.27	999.93	**3,877.13**	**0.422**	**4,022.69***	**17.23**	3,997.70	0.466
RC103	12,261.67[26]	15,467.41	550.61	**14,530.65**	**0.185**	**14,634.14**	**19.42**	14,603.39	0.191
RC203	4,049.62[6]	7,384.64	1,369.62	5,487.94	0.355	**4,497.68***	**20.72**	**4,465.26**	**0.103**

paper highlights the importance of knowledge interpretability in promoting research in automated algorithm design. 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. 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 both operators and acceptance criteria.

ACKNOWLEDGMENTS

This work was supported by the University of Nottingham, UK.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*. IEEE, 3–14.

[2] Shahriar Asta and Ender Özcan. 2014. An apprenticeship learning hyper-heuristic for vehicle routing in hyflex. In *2014 IEEE symposium on evolving and autonomous learning systems (EALS)*. IEEE, 65–72.

[3] Shahriar Asta, Ender Özcan, Andrew J Parkes, and A Şima Etaner-Uyar. 2013. Generalizing hyper-heuristics via apprenticeship learning. In *Evolutionary Computation in Combinatorial Optimization: 13th European Conference, EvoCOP 2013, Vienna, Austria, April 3-5, 2013. Proceedings 13*. Springer, 169–178.

[4] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)* 35, 3 (2003), 268–308.

[5] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. 2007. Vehicle routing. *Handbooks in operations research and management science* 14 (2007), 367–428.

[6] Zbigniew J Czech and Piotr Czarnas. 2002. Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th Euromicro workshop on parallel, distributed and network-based processing*. IEEE, 376–383.

[7] M Fisher and M Fisher. 1995. Chapter 1 vehicle routing. *Handbooks in Operations Research and Management Science* 8 (1995), 1–33.

[8] Philippe Fournier-Viger, Usef Faghihi, Roger Nkambou, and Engelbert Mephu Nguifo. 2010. CMRules: an efficient algorithm for mining sequential rules common to several sequences. In *Twenty-Third International FLAIRS Conference*.

[9] Philippe Fournier-Viger, Ted Gueniche, Souleymane Zida, and Vincent S Tseng. 2014. ERMiner: sequential rule mining using equivalence classes. In *International Symposium on Intelligent Data Analysis*. Springer, 108–119.

[10] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. 2016. The SPMF open-source data mining library version 2. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 36–40.

[11] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. 2017. A survey of sequential pattern mining. *Data Science and Pattern Recognition* 1, 1 (2017), 54–77.

[12] Philippe Fournier-Viger, Roger Nkambou, and Vincent Shin-Mu Tseng. 2011. RuleGrowth: mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 2011 ACM symposium on applied computing*. 956–961.

[13] Philippe Fournier-Viger and Vincent S Tseng. 2011. Mining top-k sequential rules. In *International Conference on Advanced Data Mining and Applications*. Springer, 180–194.

[14] Philippe Fournier-Viger and Vincent S Tseng. 2013. TNS: mining top-k non-redundant sequential rules. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. 164–166.

[15] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A Moreno Pérez. 2019. *Variable neighborhood search*. Springer.

[16] Pierre Hansen, Nenad Mladenović, and José A Moreno Pérez. 2010. Variable neighbourhood search: methods and applications. *Annals of Operations Research* 175, 1 (2010), 367–407.

[17] Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Patrick Meyer, Amir Mohammad Karimi-Mamaghan, and El-Ghazali Talbi. 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 (2022), 393–422.

[18] David Meignan, Abderrafiaa Koukam, and Jean-Charles Créput. 2010. Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism. *Journal of Heuristics* 16, 6 (2010), 859–879.

[19] Weiyao Meng and Rong Qu. 2021. Automated design of search algorithms: Learning on algorithmic components. *Expert Systems with Applications* 185 (2021), 115493.
[20] David Mester, Olli Bräysy, and Wout Dullaert. 2007. A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications* 32, 2 (2007), 508–517.
[21] Nelishia Pillay, Rong Qu, Dipti Srinivasan, Barbara Hammer, and Kenneth Sorensen. 2018. Automated Design of Machine Learning and Search Algorithms [Guest Editorial]. *IEEE Computational intelligence magazine* 13, 2 (2018), 16–17.
[22] Rong Qu, Graham Kendall, and Nelishia Pillay. 2020. The General Combinatorial Optimization Problem: Towards Automated Algorithm Design. *IEEE Computational Intelligence Magazine* 15, 2 (2020), 14–23.
[23] Yves Rochat and Éric D Taillard. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics* 1 (1995), 147–167.
[24] Louis-Martin Rousseau, Michel Gendreau, and Gilles Pesant. 2002. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of heuristics* 8 (2002), 43–58.
[25] Paul Shaw. 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 (1997).
[26] Paul Shaw. 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*. Springer, 417–431.
[27] Marius M Solomon. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research* 35, 2 (1987), 254–265.
[28] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science* 31, 2 (1997), 170–186.
[29] Fadi Thabtah and Peter Cowling. 2008. Mining the data from a hyperheuristic approach using associative classification. *Expert systems with applications* 34, 2 (2008), 1093–1101.
[30] Raras Tyasnurita, Ender Özcan, and Robert John. 2017. Learning heuristic selection using a time delay neural network for open vehicle routing. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1474–1481.
[31] Raras Tyasnurita, Ender Özcan, Asta Shahriar, and Robert John. 2015. Improving performance of a hyper-heuristic using a multilayer perceptron for vehicle routing. (2015).
[32] James D Walker, Gabriela Ochoa, Michel Gendreau, and Edmund K Burke. 2012. Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework. In *International Conference on Learning and Intelligent Optimization*. Springer, 265–276.
[33] Richard T Wong. 1983. Combinatorial optimization: Algorithms and complexity (Christos H. Papadimitriou and Kenneth Steiglitz). *SIAM Rev.* 25, 3 (1983), 424.
[34] Qiankun Zhao and Sourav S Bhowmick. 2003. Sequential pattern mining: A survey. *ITechnical Report CAIS Nayang Technological University Singapore* 1, 26 (2003), 135.