Automated design of local search algorithms: Predicting algorithmic components with LSTM

Weiyao Meng[a,*], Rong Qu[a]

[a]*School of Computer Science, University of Nottingham, Nottingham, UK*

Abstract

With a recently defined AutoGCOP framework, the design of local search algorithms has been defined as the composition of elementary algorithmic components. The effective compositions of the best algorithms thus retain useful knowledge of effective algorithm design. This paper investigates machine learning to learn and extract useful knowledge in effective algorithmic compositions. The process of forecasting algorithmic components in the design of effective local search algorithms is defined as a sequence classification task, and solved by a long short-term memory (LSTM) neural network to systematically analyse algorithmic compositions. Compared with other learning models, the results reveal the superior prediction performance of the proposed LSTM. Further analysis identifies some key features of algorithmic compositions and confirms their effectiveness for improving the prediction, thus supporting effective automated algorithm design.

Keywords: Automated algorithm design, Search algorithms, LSTM, Imbalanced dataset, Data re-sampling

*Corresponding author

Email addresses: weiyao.meng@nottingham.ac.uk (Weiyao Meng), rong.qu@nottingham.ac.uk (Rong Qu)

1. Introduction

In optimisation research, the design of effective algorithms often presents a challenge and burden for researchers due to the extensive expertise and efforts required in making a large number of decisions (Pillay et al., 2018). The automation of algorithm design releases humans from the tedious design process and supports the more flexible exploration of a larger number of unseen algorithms which would otherwise require substantial time and expertise to design (Meng & Qu, 2021).

With recent successes of artificial intelligence, in particular machine learning, automation in algorithm design is fast emerging along with the successful research development in evolutionary computation (Qu, 2021). With the taxonomy defined in (Qu et al., 2020), the latest research in automated algorithm design has been categorised into three themes as follows:

- Automated configuration: to automatically determine parameter values of specific algorithms to solve a set of problem instances (Hutter et al., 2007).
- Automated selection: to automatically select the most appropriate algorithm from a portfolio of candidate algorithms for a set of problem instances.
- Automated composition: to automatically compose or combine heuristics or components of arbitrary algorithms to solve the problem at hand online (Qu et al., 2020).

The research in automated composition takes a bottom-up method to compose a set of algorithmic components flexibly, generating new and generic algorithms (Qu et al., 2020). The other two themes of research follow a top-down approach, resulting in variants and portfolios of the existing algorithms. In line with automated algorithm composition, a new model named General Combinatorial Optimisation Problem (GCOP) (Qu et al., 2020) formally defines the problem of algorithm design itself as a combinatorial optimisation problem. In GCOP, algorithm design decisions are defined as elementary algorithmic components, including basic operators, acceptance criteria and termination criteria (Meng & Qu, 2021). Various search algorithms can be modelled as compositions of these basic components. The GCOP model thus provides standard and theoretical support for automated algorithm design (Qu et al., 2020).

This paper investigates the effective algorithmic compositions based on GCOP, aiming to explore new knowledge to support automated algorithm design. In particular, we focus on and analyse three categories of algorithm composition based on the types of components in the GCOP search space, including specific, semi-specific, and general automated composition.

The specific automated composition in the literature combines common components of specific types of algorithms or target algorithms. The most studied specific algorithms include SAT solver (KhudaBukhsh et al., 2016), simulated annealing algorithms (Franzin & Stützle, 2019), iterated greedy algorithms (Mascia et al., 2013), stochastic local search algorithms (Pagnozzi & Stützle, 2019) and multi-objective evolutionary algorithms (Bezerra et al., 2015), (Lopez-Ibanez & Stutzle, 2012).

The semi-specific automated composition supports the flexible composition of specifically designed components. Hyper-heuristics (Pillay & Qu, 2018), which operate at a higher level on a set of low-level heuristics, represent a main line of research in this category. In one type of hyper-heuristics, selection hyper-heuristics, the components are problem-specific heuristics or operators. Various learning models have been applied to iteratively select low-level heuristics for automated composition, including Markov chain (McClymont & Keedwell, 2011), hidden Markov models (Kheiri & Keedwell, 2015), choice function (Cowling et al., 2000), multi-armed bandit (Sabar et al., 2015), (Ferreira et al., 2017) and reinforcement learning (Nareyek, 2003), (Burke et al., 2003), (Khamassi et al., 2011), (Di Gaspero & Urli, 2011), (Özcan et al., 2012) and (Di Gaspero & Urli, 2012). Another type of hyper-heuristics, generation hyper-heuristics, mainly uses genetic programming (Banzhaf et al., 1998) to compose the elements of a specific type of target heuristics. It can be seen as based on an algorithmic component set of common features of the target algorithms. The mostly studied methods in generation hyper-heuristics include dispatching rules for job scheduling problem (Pickardt et al., 2010), (Nguyen et al., 2012), local search algorithms for bin packing (Burke et al., 2011) and satisfiability testing (Fukunaga, 2008), and evolutionary algorithms for continuous optimisation problems (Miranda et al., 2017), function optimisation, travelling salesman problems and the quadratic assignment problem (Oltean, 2005), vehicle routing problems (Jacobsen-Grocott et al., 2017), and timetabling problems (Sabar et al., 2013), (Pillay & Özcan, 2019).

The general automated composition supports the flexible composition

of the elementary algorithmic components as defined in GCOP. A general AutoGCOP framework is built in (Meng & Qu, 2021) to support the automated design of local search algorithms by combining elementary algorithmic components in GCOP. In our previous study (Meng & Qu, 2021), a Markov chain-based GCOP method presents superior performance for designing new algorithms automatically within AutoGCOP on vehicle routing problems with time window constraints (VRPTW). A general search framework (GSF) based on GCOP is proposed in (Yi et al., 2022) to support the design of both local search and population-based algorithms. Reinforcement learning has been applied to design new general population-based algorithms on VRPTW.

In specific and semi-specific automated composition, more human decisions are involved while selecting and configuring the algorithmic component set. These components (particularly in selection hyper-heuristics) can be seen as manually defined compound components that combine some of the basic components in GCOP (Meng & Qu, 2021). Thus, the resulting algorithms are only a subset of those combined from the basic algorithmic components within AutoGCOP. In other words, the AutoGCOP framework supports the more flexible exploration of a larger space of new unseen local search algorithms.

From the aspect of machine learning, the search process generates a considerable volume of data (Karimi-Mamaghan et al., 2022), which has been discarded in the meta-heuristic literature. Some recent selection hyper-heuristics explore these data to determine an elite set of low-level heuristics (i.e., components) for composition space (Cowling & Chakhlevitch, 2003),

(Chakhlevitch & Cowling, 2005), (Mısır et al., 2013), (Remde et al., 2012) and (Soria-Alcaraz et al., 2017). With the historical data of the applied low-level heuristics, machine learning is used to predict which low-level heuristics to use. Various conventional machine learning models investigated include association classifier (Thabtah & Cowling, 2008), k-means classifier (Asta et al., 2013), decision trees (Asta & Özcan, 2014), and neural networks (Tyasnurita et al., 2015) and (Tyasnurita et al., 2017). The study in (Yi et al., 2023) identifies the effectiveness of learning from the combinations of algorithmic components to support algorithm design, suggesting that useful knowledge can be extracted from the algorithmic component sequences for automating the algorithm composition in problem-solving. Algorithmic compositions, however, are time-dependent sequences (components determined at each search step), with a temporal correlation between the components. This presents challenges to conventional learning models in learning to predict algorithmic components in the search.

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

In this paper, the automated composition is modelled as a sequence clas-

sification 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. An LSTM network model is proposed for the defined task to learn the potential temporal correlation in the algorithmic compositions for forecasting the selection of algorithmic components. The proposed LSTM model is investigated against a set of commonly used conventional classifiers, to demonstrate its effectiveness on the defined prediction task.

The contributions of this paper 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 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 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.

In the rest of the paper, Section 2 defines the new machine learning task on automated algorithm composition. Section 3 presents the algorithmic composition data for further investigations. Section 4 presents the proposed LSTM model for the defined prediction task. Section 5 discusses the experimental setup and result analysis, followed by conclusions and future works in Section 6.

2. Automated algorithm design defined as a machine learning task

2.1. An overview of the general automated composition framework

In the newly defined GCOP model in (Qu et al., 2020), various search algorithms are seen as compositions of a finite set A of elementary algorithmic components $a \in A$. Algorithm design can be defined as a combinatorial optimisation problem with decision variables taking these a as values from 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 (Meng & Qu, 2021) has been built to support the automated composition, with the following three categories of elementary algorithmic components a in an extended GCOP model:

- Operators $o_i \in A_{1.0_o}$: modify values of the decision variables in s_1 to generate a new solution s_2 in the search space of p.

- Acceptance criteria $a_j \in A_{1.0_a}$: determine if and how s_2 is accepted in the search.

- Termination criteria $t_k \in A_t$: control when to terminate a loop.

In other words, the existing algorithms can be seen as specific GCOP solutions composed manually to automatically design new local search algorithms.

With the AutoGCOP model, the design of various local search algorithms can be defined as the flexible composition of elementary algorithmic components (Meng & Qu, 2021). 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 (Qu et al., 2020), some of which may be difficult to identify by human experts.

2.2. 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 (1).

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

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

Sequence classification is a type of classification task of classifying sequences into existing categories (Xing et al., 2010). This study concerns the algorithm design problem of selecting operators to compose based on the information of applied operator sequences. This problem is formulated as a sequence classification problem as follows: given a finite set of operators A_o as a set of class labels, the task of sequence classification is to build a sequence classifier F, which maps an operator sequence q to a class label $o_i \in A_o$, written as $F : q \rightarrow o_i$.

The defined task aims to explore the hidden sequential relations between the operators within operator compositions. The defined task in this study can also be treated as a conventional multi-class classification problem (Crammer & 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.

2.3. Discussion on the newly defined task

It is important to note that the defined task is based on the hypothesis that algorithmic compositions might exhibit time-series characteristics and dependencies among the applied algorithmic components. Building upon this hypothesis, the data of algorithmic compositions can be seen as sequential

data, presenting an opportunity to explore the knowledge hidden in the data with machine learning techniques. However, no prior research discusses suitable learning techniques for the sequence classification of algorithmic compositions.

Among various applications of sequence classification, text classification which aims to retrieve information in text data (Xing et al., 2010), is highly relevant to this study. Text data includes sequences of text (e.g., words, tokens, or characters), each sequence associated with labels or categories that classification models aim to predict. The machine learning techniques that obtain good performance in text classification (Kowsari et al., 2019) might perform well in the defined task of this study.

Conventional classifiers, such as naive Bayes (Rish et al., 2001), logistic regression (Dreiseitl & Ohno-Machado, 2002), neural networks (Murtagh, 1991) and random forest (Breiman, 2001), have been widely used in text classification (Shah et al., 2020) because of their simple implementation and effective performance (Kowsari et al., 2019). Random forest has shown competitive performance among conventional classifiers in text classification (Chen et al., 2022), especially for imbalanced text data (Wu et al., 2014). It is important to note that conventional classifiers learn from text data by treating the text sequences as fixed-length feature vectors, thus the sequential nature of features within the sequence is difficult to capture. In other words, the text classification task is treated as a conventional classification problem by conventional classifiers.

Recently, deep learning techniques have been greatly applied in text classification with overall better performance than conventional classifiers (Razno,

2019). These techniques learn complex patterns and representations from text data with different mechanisms. Some representative methods include recurrent neural networks (Goodfellow et al., 2016) which learn sequential information and dependencies within the text for classification, and attention-based neural networks (Vaswani et al., 2017) which use attention mechanisms to focus on the most relevant information in the sequence for classification.

The nature of the data is important for choosing suitable learning techniques. The investigations of this study are built upon the hypothesis on the sequential feature of algorithmic composition data. If such a feature indeed exists, it is possible that conventional classifiers may not achieve satisfactory predictive performance compared to recurrent neural networks (RNNs) in the defined sequence classification task. Therefore, it would be interesting to perform a comparative analysis to evaluate the predictive performance of RNNs against conventional classifiers in the defined sequence classification task. This comparison might reveal the presence of the sequential feature in operator sequence data, which is useful for supporting the further analysis of algorithmic compositions.

3. Data of algorithm design for machine learning

To investigate the new sequence classification problem on algorithm composition, the widely studied vehicle routing problem with time window constraints (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.

3.1. The VRPTW problem

VRP is one of the most investigated combinatorial optimisation problems in operational research (Wong, 1983). The basic VRP (Fisher & Fisher, 1995) concerns assigning and ordering customer delivery demands to a set of vehicles while minimising the total travel costs serving all the customers. VRPTW is one of the most widely studied variants, where customers must be served within specified time intervals (Cordeau et al., 2007). 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 (2), where c is set to 1000 empirically and widely used in the literature (Walker et al., 2012).

$$c \times NV + TD \tag{2}$$

The investigations have been conducted on the benchmark Solomon 100 set (Solomon, 1987) as shown in Table 1, covering different instance features.

3.2. Collection of operator sequences with GCOP methods

To explore insights on effective algorithm compositions, the most basic operators o_i as shown in Table 2 have been investigated within AutoGCOP. This set of basic operators presents different characteristics for solving VRPTW (Meng & Qu, 2021).

Within AutoGCOP, MC-GCOP (Meng & Qu, 2021), which adaptively learns the transition performance between pairs of basic components, presents superior overall performance for composing algorithmic components to solve VRPTW problem instances (Meng & Qu, 2021). The MC-GCOP method is

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
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
R208	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

Table 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

applied to the problem instances in Table 1 to produce a collection of effective algorithmic compositions of the operators in Table 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 groups 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 (O_index) and features shown in Table 2 and whether the

solution quality is improved after it has been applied (SC).

- Instance features: problem instance features in Table 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 $\{f_{t-3}, f_{t-2}, f_{t-1}\}$ describe whether the solution quality is improved after using $o_i \in q$. Table 3 presents $q \rightarrow o_i$ with features as the input and output to the sequence classification task.

Table 3: An example operator sequence (q) described by features as input and the corresponding label (o_i) as output, to the sequence classification task.

Input: q											Output: o_i
Iter	O_index	NS	IR	OT	SC	SH	DT	ST	TWD	TWW	O_index
t-3	o^{in}_{xchg}	Small	1-route	Exchange	f_{t-3}	Short	C	90	75%	61.27	o^{bw}_{xchg}
t-2	o^{in}_{ins}	Small	1-route	Insert	f_{t-2}	Short	C	90	75%	61.27	
t-1	o_{rr}	Large	n-route	Ruin-recreate	f_{t-1}	Short	C	90	75%	61.27	

3.3. Feature processing

Table 4 outlines the features of the operator sequences, including both categorical and numerical data. The features of the original dataset are processed for modelling by transforming categorical features into numerical representations through label encoding. As numerical features present varying scales, normalisation is necessary to avoid potential biases. In this study, min-max scaling is employed to normalise the numerical features, converting them to a standardised scale ranging from 0 to 1 while maintaining the original distribution.

Table 4: A summary of features that describe operator sequences.

Feature groups	Features	Descriptions	Feature types
Search stage feature	Iter	Index of iteration	Numerical
Operator features	O_index	Index of operator	Categorical
	NS	Neighhourbood size	Categorical
	IR	Involved routes of operation	Categorical
	OT	Operation type	Categorical
	SC	Solution quality change	Categorical
Instance features	SH	Scheduling horizon	Categorical
	DT	Distribution type	Categorical
	ST	Service time	Categorical
	TWD	Time window density	Numerical
	TWW	Time window width	Numerical

3.4. Data imbalance in the operator sequence data set

One distinct observation on the collected data is that the labels o_{ins}^{bw}, o_{rr} and $2opt^*$ dominate the categories of the extracted sequences (as shown in Table 5), which leads to a seriously imbalanced classification problem (Chawla, 2009). This is due to their larger contributions to better performance on VRPTW (Meng & 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: The appearance of each o_i in Table 2 as the label of the extracted operator sequences.

Operator	o_{xchg}^{in}	o_{xchg}^{bw}	o_{ins}^{in}	o_{ins}^{bw}	o_{rr}	$2opt^*$
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 6, 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.

4. Learning from algorithmic components with LSTM

In solving sequence classification problems, recurrent neural networks (RNNs) (Goodfellow et al., 2016), a family of neural networks, 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) neural networks (Hochreiter & Schmidhuber, 1997) which

Table 6: Representative re-sampling methods.

Category	Strategy
Under-sampling	Random under-sampling (RU)
	NearMiss (NM) (Mani & Zhang, 2003)
	One-sided selection (OSS) (Kubat & Matwin, 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)

can capture the long and short-term dependencies or temporal differences in a sequence. This paper investigates LSTM neural networks for solving the newly defined sequence classification task.

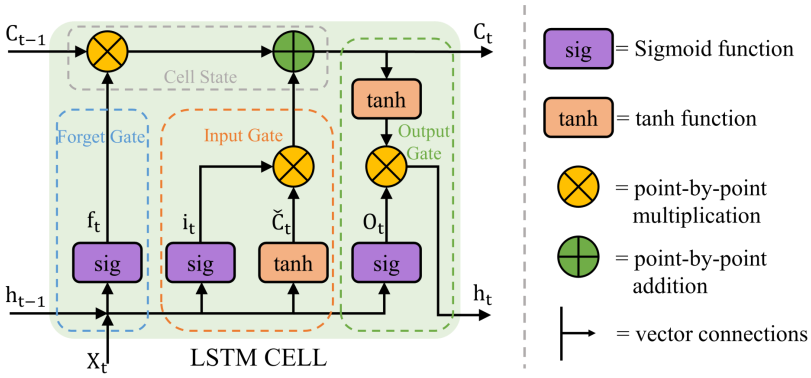
4.1. Preliminary fundamentals of LSTM

Recurrent neural networks (RNNs) (Mandic & Chambers, 2001) are a class of sequence-based neural networks in modelling sequence learning tasks (Nammous & 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 & Schmidhuber, 1997) is a variant of RNNs, which

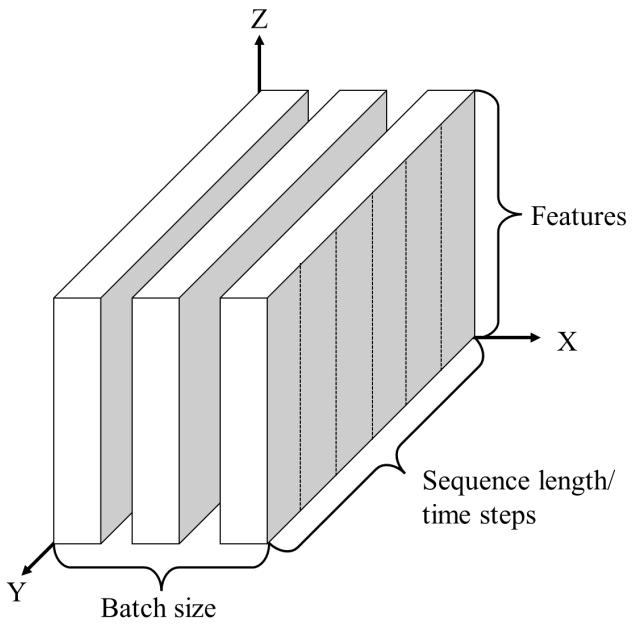
integrates a gating mechanism to resolve the long-term dependency issue, as shown in Figure 1. Each LSTM cell retains an internal cell state to store the memory of the last input, and a hidden state which stores the information of the last output. Three gate units, i.e., input gate, output gate and forget 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.

Figure 1: The structure of a basic LSTM cell (Smagulova & 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.



With this sequence specialisation, RNNs (including LSTM) interpret the input data as a data cube with three dimensions as shown in Figure 2. This representation is different from conventional classifiers (including MLP) which take a 2-dimensional matrix as the input, each row consisting of features of each sequence.

Figure 2: Input data representation in RNNs (including LSTM) (Skydt et al., 2021).

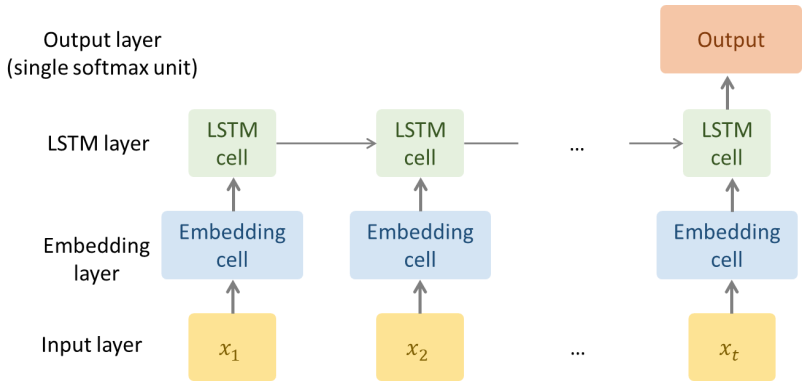


4.2. LSTM for operator sequences

For the operator sequence classification task defined in Section 2.2, this work proposes a four-layer LSTM as shown in Figure 3. The input data are as shown in Figure 2, 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, embedding 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 natural language processing (NLP) tasks (Levy & Goldberg, 2014), where neural embedding is more manageable with the lower

Figure 3: The structure of the proposed LSTM.



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. Findings of LSTM on automated algorithm composition

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

The investigations of the first research issue use operator sequences described solely by the applied operators. LSTM is applied in the defined sequence classification task to model operator sequences by capturing the sequential nature of the data. To verify the predictive performance of LSTM in the defined task, a set of baselines is used to compare against LSTM. The selected baseline models are the conventional classifiers that are widely used in text classification (Shah et al., 2020), including naive Bayes (NB), logistic regression (LR), multi-layer perceptron (MLP) and random forest (RF). Given the operator sequence data, conventional classifiers treat the operator sequence as a feature vector and learn to model the data without considering the sequential dependencies between these operators, while LSTM can capture the dependencies between operators within the sequence. The comparative analysis between LSTM and conventional classifiers thus presents insights into the performance of different learning models for the newly defined sequence classification task, while also revealing knowledge of hidden sequential relations between operators to support algorithm design.

Towards the first research issue, Section 5.1 presents the comparative analysis of the proposed LSTM model against baselines. An LSTM model with no embedding layer (denoted as LSTM-basic) is studied to show the effects of embeddings. The influence of different resampling methods in Section 5.2 and the length of operator sequences in Section 5.3 are also investigated.

For the second research issue, Section 5.4 analyses a set of features describing operator sequences with LSTM to identify useful information to support automated algorithm design.

Traditionally, accuracy is a widely used performance metric in classifica-

tion tasks (López et al., 2013), however, not an appropriate measure when learning on imbalanced data (Chawla, 2009). The area under the curve (AUC) (Swets, 1988) is used to evaluate the performance of the models.

5.1. Performance comparison of learning models

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 6, 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 7 presents the stats of the re-sampled training data and testing data.

To avoid potential over-fitting and consider the 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 8 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 9. 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

Table 7: Data size of re-sampled training data and testing data.

Class	Re-sampling methods	Training data size						
		o_{xchg}^{in}	o_{xchg}^{bw}	o_{ins}^{in}	o_{ins}^{bw}	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

Table 8: 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 9: 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.

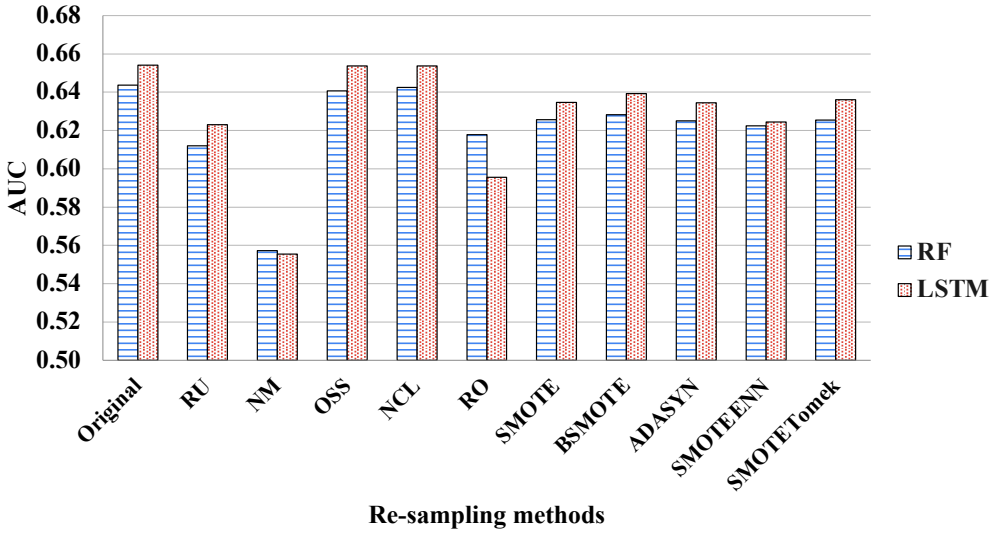
Resampling methods	RU	NM	OSS	NCL
LSTM↔LSTM-basic	+	+	+	+
LSTM↔RF	+	∼	+	+
Resampling methods	RO	SMOTE	BSMOTE	ADASYN
LSTM↔LSTM-basic	+	+	+	+
LSTM↔RF	-	+	+	+
Resampling methods	SMOTEENN	SMOTETomek	Original	
LSTM↔LSTM-basic	+	+	+	
LSTM↔RF	+	+	+	

improve the prediction performance. LSTM also outperforms the conventional classifier, due to its sequence architecture which captures features in modelling long texts.

5.2. Effects of re-sampling methods

To investigate the impact of re-sampling data on the learning models, Figure 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 an improvement in 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 is shown to be the best to process the data set for learning 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.

Figure 4: Performance comparison of RF and LSTM on the data sets processed with different re-sampling methods.



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) 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 models, 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 7. 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 10 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. Unsurprisingly, the learning models obtain better performance on more imbalanced data. It is interesting to observe in Figure 5, that from 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.

5.3. Impact from the length of operator sequences

The impacts from different operator sequence length settings are investigated by comparing the performance of the selected learning models on

Table 10: 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

Figure 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 denote RU with the ratio of majority class to minority class 1:1, 2:1, 3:1, 4:1 and 5:1, respectively.

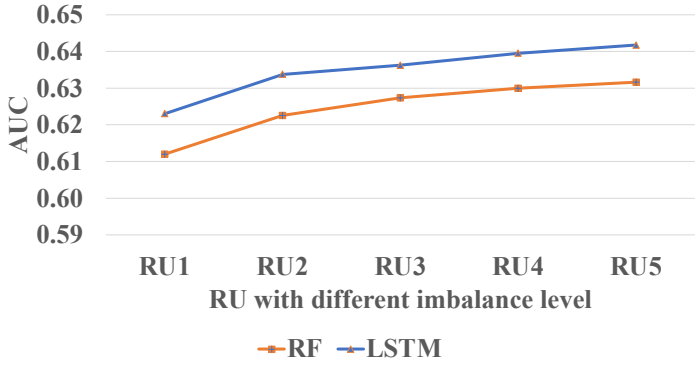
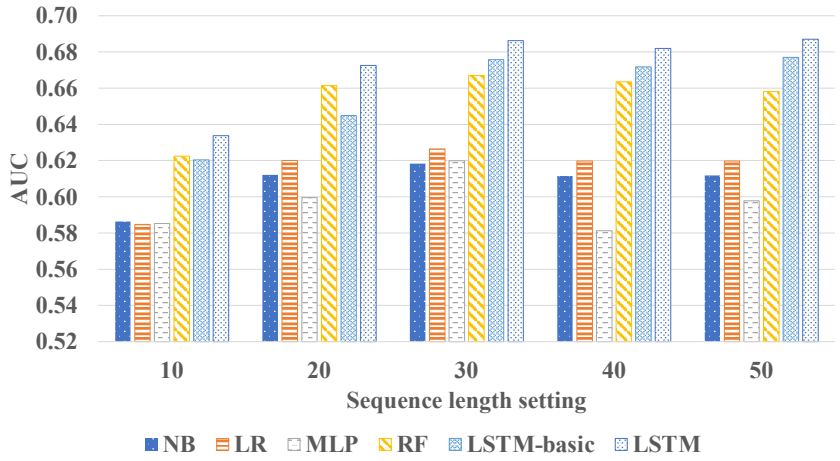


Figure 6: The comparison of learning models in terms of the AUC performance.



different length settings (i.e., Y in Figure 2) in the range of {10, 20, 30, 40, 50}. The 70% data is processed by RU2 with an imbalance level of 2:1 as the training data.

Figure 6 shows the AUC performance of learning models on the operator sequence data set with different length settings. Among the six classifiers in this study, LSTM achieves the best performance on operator sequences with different length settings. In addition, its performance increases with the increase of the length. This further justifies the effectiveness of LSTM, especially for 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 5.4.

5.4. Investigations on features of operator sequences

This section investigates the features of operator sequences in Table 3 using RF and LSTM models. The experimental study uses operator sequences with a length of 30. The 70% data is processed by RU2 with an imbalance level of 2:1 as the training data.

5.4.1. Effects of each feature

To investigate the impact of each feature, RF and LSTM are evaluated on the operator sequences described by features in Table 4, compared with the baseline original data (described by the applied operator O_index only), results shown in Figure 7. Both RF and LSTM perform better with additional features. Overall, LSTM outperforms RF on the same data set. For LSTM, solution quality change SC improves the performance the most, followed by search stage $Iter$. With operator neighbourhood size NS, LSTM 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 11, for RF, the additional features to the original data significantly improve its prediction performance. For LSTM, some features contribute to performance improvement. These include the search stage, instance features (i.e., scheduling horizon and time window width), and the solution quality change after the selected operators.

Figure 7: AUC performance comparison of learning models on data sets with different features in Table 4.

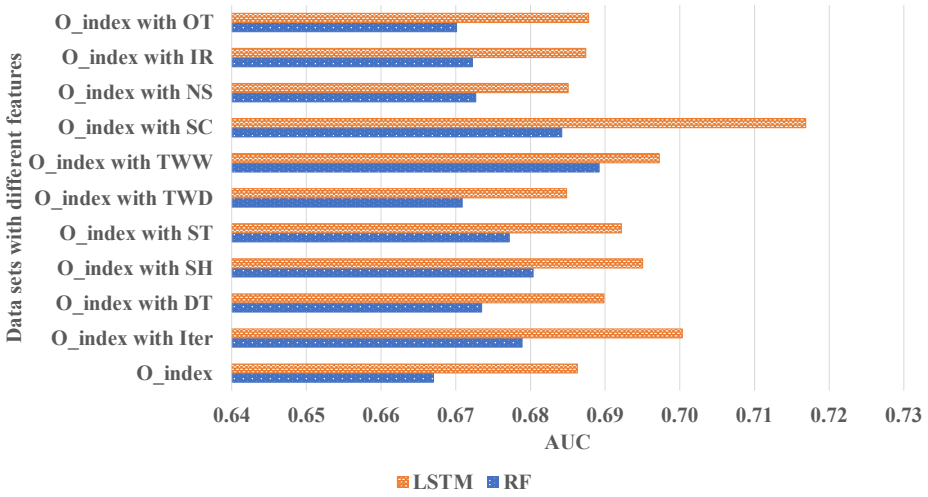


Table 11: Comparison of model performance on the data with *O_index* and the data with additional features, using the Mann–Whitney–Wilcoxon test. The +, -, or ∼ indicate if *O_index* with an additional feature is significantly better than, worse than, or statistically equivalent to data with just *O_index*, respectively.

	RF	LSTM
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	+	∼

Table 12: The top 10 important features found by RF.

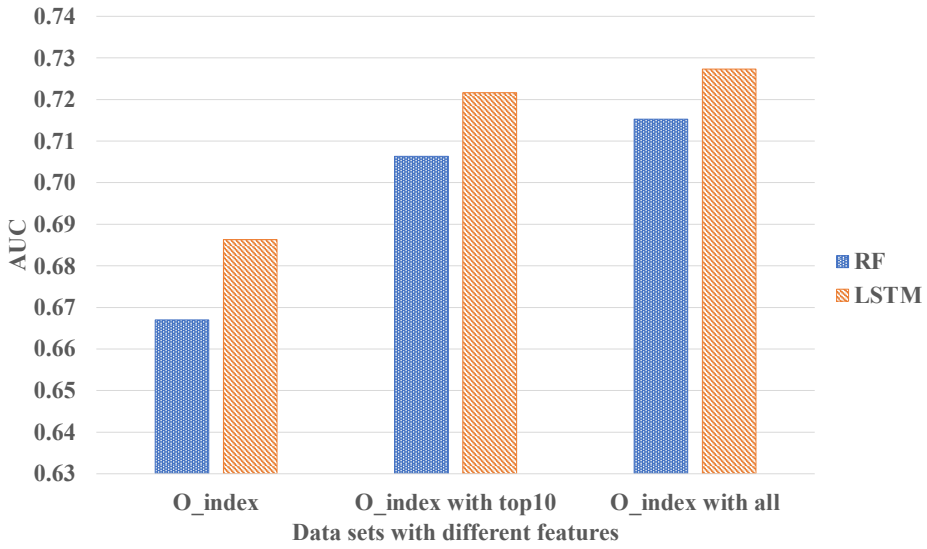
Rank	Importance	Feature group	Feature description
1	0.0765	Search stage feature	Index of iteration
2	0.0699	Instance feature	Time window width
3	0.0641	Instance feature	Scheduling horizon
4	0.0359	Operator feature	Operator index of the last operator
5	0.0168	Operator feature	Neighbourhood size of the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

5.4.2. Importance of the features

Further analysis is conducted to investigate the feature importance of the learning models. Table 12 presents the top 10 most important features identified by RF. The search stage feature *Iter* is the most important feature by RF. Similar to the observation on LSTM in Table 11, scheduling horizon 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 and LSTM are evaluated 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 12. Figure 8 shows that the top 10 important features can significantly improve

Figure 8: Comparison of different feature sets using the performance of learning models.



the performance of RF and LSTM. However, both models obtain further better performance with all the features.

The Mann–Whitney–Wilcoxon test with a 95% confidence level on different feature sets is shown in Table 13, confirming the effectiveness of the important feature set for both models. RF on the data with all features is significantly better than the data with important features. This indicates that the less important features are also useful for RF. However, for LSTM, there is no statistical significance in using all features compared with using only the important features. This indicates that LSTM can learn useful knowledge on the operator sequences with only a smaller subset of effective features.

Table 13: Performance comparison of RF and LSTM on operator sequence data with different feature sets, using the 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
O_index with important ↔ *O_index*	+	+
O_index with important ↔ *O_index* with all	-	∼

5.4.3. 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 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 be interpreted 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 both RF and LSTM. 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 fea-

tures should be considered.

Operator sequences. Both RF and LSTM perform better with longer operator sequences, but LSTM is better than RF. Particularly, 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. 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.

6. Conclusions and future workds

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 paper 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. Overall, this study makes a valuable step towards integrating machine learning into automated 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 is shown 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 is shown to be the best machine learning method due to its robust performance on the defined prediction task in forecasting the behaviour of algorithmic components.

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, par-

ticularly 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.

6.2. Discussion on the practical impacts of automated algorithm design

Automated algorithm design is attracting increasing research attention in solving complex COPs (Yi et al., 2022). It should be noted that the impact of automated algorithm design is not limited to optimisation research but has much wider practical perspectives.

Improved algorithm design. The performance of manually designed algorithms highly relies on the experience and effort of human experts, who may only consider a limited number of designs, limiting the exploration of potential algorithms (Hoos, 2008). Automation in algorithm design enables the exploration of a larger scope of candidate algorithms, some of which may never be considered by manual designs (Meng & Qu, 2021). Therefore, automated algorithm design can bring new and effective algorithms efficiently, leading to cost-efficient and better solutions for industry and business.

Better use of human resources. The design of high-performing metaheuristics involves extensive domain knowledge and efforts from human experts. Automation in algorithm design is an increasing demand from industry and business for removing the heavy reliance on human involvement (Pillay et al., 2018). By automating the design process, algorithm designers can

be freed from the tedious and time-consuming aspects of algorithm design and apply their expertise towards more creative tasks (Burke et al., 2009). They can provide valuable insights into the problems where optimisation is critical, make informed decisions about algorithmic choices (such as deciding on search frameworks), and align themselves with industry requirements (Hoos, 2008). This ultimately leads to cost savings and improved operational efficiency for industry and business.

Adaptability across domains. Effective meta-heuristics are demanding in various COPs, such as job shop scheduling, knapsack problem, personnel scheduling, timetabling, vehicle routing, and many others, as well as their extensions with various real-world constraints and features (Qu et al., 2020). While much time and effort have been invested in designing effective heuristics, most heuristics proposed in the literature are specific to certain problem instances or stages of problem-solving (Yi et al., 2022). The automated algorithm design can greatly reduce the costs of the designing algorithm that can be applied across multiple problem domains (Burke et al., 2009). For example, frameworks like HyFlex (Ochoa et al., 2012) and EvoHyp (Pillay & Beckedahl, 2017) are proposed for the automated composition of algorithms that can be applied to different problem domains with little development effort. This allows industry and business to obtain tailored solutions for different optimisation use cases efficiently.

This study mainly contributes to the field of automated algorithm design by introducing a machine learning task to support the exploration of the hidden knowledge in algorithmic compositions with machine learning, thus supporting effective algorithm design. The widely investigated VRPTW is

used as a testbed for the investigations of machine learning methods. While the main contributions are mainly on the algorithm design aspects, this research also holds several practical advantages in the context of algorithm design.

Insights into the use of machine learning. The utilisation of machine learning in algorithm design can lead to more efficient and effective solutions (Karimi-Mamaghan et al., 2022). By introducing the newly defined machine learning task, this study provides algorithm designers with a new direction of utilising machine learning in automated algorithm design. In addition, this study compares several widely used machine learning models in the defined task, providing algorithm designers insights into suitable techniques and allowing them to focus on higher-level decision-making.

Insights into algorithm designs for VRPs. Identifying useful features of algorithm design can assist in learning and decision-making in algorithm design (Yi et al., 2023). This study identifies the important search stage features and instance features for algorithm design on VRPTW. These key features can be used to support algorithm design for solving VRPTW and their extensions with various real-world constraints and features, leading to efficient route planning and scheduling.

Impacts beyond VRPs. The investigations of this study have broader practical benefits that extend beyond VRPs. 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 analyse algorithmic compositions can be extended and applied to other domains, which

can be beneficial for industries and businesses that operate beyond VRPs.

6.3. Future works

Further investigations of algorithmic compositions. This study uncovers the sequential dependencies hidden in the operator sequences by evaluating LSTM against conventional classifiers. These findings can serve as evidence to support future investigations into the sequential behaviour of operators in effective algorithmic compositions. Deep learning techniques that are widely used in sequential data (Kowsari et al., 2019) may also yield promising results in the defined sequence classification task.

Evaluation of automatically designed algorithms. The findings of this study establish a solid foundation for the potential implementation of sequence classification models in automated algorithm design. A future work direction is to incorporate the proposed machine learning task as an element of automated algorithms and evaluate them against the state-of-the-art manual methods for solving VRPTW (Vidal et al., 2013). This comparative analysis would provide valuable insights into the strengths and weaknesses of the proposed machine learning task and techniques in this study incorporated into evolutionary computation and meta-heuristics.

Investigation of different VRP scenarios. Effective algorithms are demanding for solving VRPs of varying scales and variants, especially in handling real-world scenarios (Kytöjoki et al., 2007). Building upon this, a promising extension of this study is to examine the variations in algorithmic compositions for VRPTW instances of diverse sizes and variants (Vidal et al., 2013) with machine learning. Potential knowledge can be identified to

support effective algorithm design across different problem scales and complexities.

Extension to problem domains beyond VRPs. It is important to note that the proposed sequence classification task for automated algorithm design is not limited to VRP alone. The basic algorithmic components in GCOP 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 same methodology of this study can be applied to different problem domains. In the future, the proposed machine learning task can guide further exploration of effective algorithmic compositions across different domains.

Acknowledgements

This work was supported by the School of Computer Science, University of Nottingham.

References

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

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

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming*. Springer.

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

Bezerra, L. C., López-Ibánez, M., & Stützle, T. (2015). Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *20*, 403–417.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Burke, E. K., Hyde, M. R., & Kendall, G. (2011). Grammatical evolution of local search heuristics. *IEEE Transactions on Evolutionary Computation*, *16*, 406–417.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2009). Exploring hyper-heuristic methodologies with genetic programming. *Computational intelligence: Collaboration, fusion and emergence*, (pp. 177–201).

Burke, E. K., Kendall, G., & Soubeiga, E. (2003). A tabu-search hyperheuristic for timetabling and rostering. *Journal of heuristics*, *9*, 451–470.

Chakhlevitch, K., & Cowling, P. (2005). Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (pp. 23–33). Springer.

Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, (pp. 875–886).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Chen, H., Wu, L., Chen, J., Lu, W., & Ding, J. (2022). A comparative study of automated legal text classification using random forests and deep learning. *Information Processing & Management*, *59*, 102798.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., & Vigo, D. (2007). Vehicle routing. *Handbooks in operations research and management science*, *14*, 367–428.

Cowling, P., & 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.* (pp. 1214–1221). IEEE volume 2.

Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 176–190). Springer.

Crammer, K., & Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine learning*, *47*, 201–233.

Di Gaspero, L., & 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., & Urli, T. (2012). Evaluation of a family of reinforcement learning cross-domain optimization heuristics. In *International Conference on Learning and Intelligent Optimization* (pp. 384–389). Springer.

Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, *35*, 352–359.

Ferreira, A. S., Gonçalves, R. A., & Pozo, A. (2017). A multi-armed bandit selection strategy for hyper-heuristics. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 525–532). IEEE.

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

Franzin, A., & Stützle, T. (2019). Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, *104*, 191–206.

Fukunaga, A. S. (2008). Automated discovery of local search heuristics for satisfiability testing. *Evolutionary computation*, *16*, 31–61.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220–239.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing* (pp. 878–887). Springer.

He, H., Bai, Y., Garcia, E. A., & 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)* (pp. 1322–1328). IEEE.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–1780.

Hoos, H. H. (2008). *Computer-aided design of high-performance algorithms*. Technical Report Technical Report TR-2008-16, University of British Columbia, Department of Computer Science.

Hutter, F., Hoos, H. H., & Stützle, T. (2007). Automatic algorithm configuration based on local search. In *Proceedings of the 22nd national conference on Artificial intelligence-Volume 2* (pp. 1152–1157).

Jacobsen-Grocott, J., Mei, Y., Chen, G., & Zhang, M. (2017). Evolving heuristics for dynamic vehicle routing with time windows using genetic programming. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1948–1955). IEEE.

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, *100*, 234–245.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & 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*, 393–422.

Khamassi, I., Hammami, M., & Ghédira, K. (2011). Ant-Q hyper-heuristic approach for solving 2-dimensional cutting stock problem. In *2011 IEEE Symposium on Swarm Intelligence* (pp. 1–7). IEEE.

Kheiri, A., & 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* (pp. 417–424). ACM.

KhudaBukhsh, A. R., Xu, L., Hoos, H. H., & Leyton-Brown, K. (2016). SATenstein: Automatically building local search SAT solvers from components. *Artificial Intelligence*, *232*, 20–42.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, *10*, 150.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML* (pp. 179–186). Citeseer volume 97.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research*, *34*, 2743–2757.

Laurikkala, J. (2001). Improving identification of difficult small classes by

balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe* (pp. 63–66). Springer.

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, *27*, 2177–2185.

López, V., Fernández, A., García, S., Palade, V., & 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., & Stutzle, T. (2012). The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, *16*, 861–875.

Mandic, D., & Chambers, J. (2001). *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley.

Mani, I., & Zhang, I. (2003). kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*. ICML United States volume 126.

Mascia, F., López-Ibánez, M., Dubois-Lacoste, J., & 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* (pp. 321–334). Springer.

McClymont, K., & Keedwell, E. C. (2011). 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* (pp. 2003–2010). ACM.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Miranda, P. B., Prudêncio, R. B., & Pappa, G. L. (2017). H3AD: A hybrid hyper-heuristic for algorithm design. *Information Sciences*, *414*, 340–354.

Mısır, M., Verbeeck, K., De Causmaecker, P., & Berghe, G. V. (2013). An investigation on the generality level of selection hyper-heuristics under different empirical conditions. *Applied Soft Computing*, *13*, 3335–3353.

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, *2*, 183–197.

Nammous, M. K., & Saeed, K. (2019). Natural language processing: Speaker, language, and gender identification with LSTM. In *Advanced Computing and Systems for Security* (pp. 143–156). Springer.

Nareyek, A. (2003). Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer decision-making* (pp. 523–544). Springer.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2012). A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation*, *17*, 621–639.

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 *Evolutionary Computation in Combinatorial Optimization: 12th European Conference, EvoCOP 2012, Málaga, Spain, April 11-13, 2012. Proceedings 12* (pp. 136–147). Springer.

Oltean, M. (2005). Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation*, *13*, 387–410.

Özcan, E., Misir, M., Ochoa, G., & Burke, E. K. (2012). A reinforcement learning: great-deluge hyper-heuristic for examination timetabling. In *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends* (pp. 34–55). IGI Global.

Pagnozzi, F., & Stützle, T. (2019). Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. *European journal of operational research*, *276*, 409–421.

Pickardt, C., Branke, J., Hildebrandt, T., Heger, J., & Scholz-Reiter, B. (2010). Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness. In *Proceedings of the 2010 Winter Simulation Conference* (pp. 2504–2515). IEEE.

Pillay, N., & Beckedahl, D. (2017). EvoHyp-a java toolkit for evolutionary algorithm hyper-heuristics. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2706–2713). IEEE.

Pillay, N., & Özcan, E. (2019). Automated generation of constructive ordering heuristics for educational timetabling. *Annals of Operations Research*, *275*, 181–208.

Pillay, N., & Qu, R. (2018). *Hyper-Heuristics: Theory and Applications*. Springer.

Pillay, N., Qu, R., Srinivasan, D., Hammer, B., & Sorensen, K. (2018). Automated design of machine learning and search algorithms [Guest Editorial]. *IEEE Computational Intelligence Magazine*, *13*, 16–17.

Qu, R. (2021). Recent developments of automated machine learning and search techniques. In *Automated Design of Machine Learning and Search Algorithms* (pp. 1–9). Springer.

Qu, R., Kendall, G., & Pillay, N. (2020). The general combinatorial optimization problem: Towards automated algorithm design. *IEEE Computational Intelligence Magazine*, *15*, 14–23.

Rao, G., Huang, W., Feng, Z., & Cong, Q. (2018). LSTM with sentence representations for document-level sentiment classification. *Neurocomputing*, *308*, 49–57.

Razno, M. (2019). Machine learning text classification model with NLP approach. *Computational Linguistics and Intelligent Systems*, *2*, 71–73.

Remde, S., Cowling, P., Dahal, K., Colledge, N., & Selensky, E. (2012). An empirical study of hyperheuristics for managing very large sets of low level heuristics. *Journal of the operational research society*, *63*, 392–405.

Rish, I. et al. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (pp. 41–46). volume 3.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2013). Grammatical evolution hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, *17*, 840–861.

Sabar, N. R., Zhang, X. J., & Song, A. (2015). A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows. In *2015 IEEE congress on evolutionary computation (CEC)* (pp. 830–837). IEEE.

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and knn models for the text classification. *Augmented Human Research*, *5*, 1–16.

Skydt, M. R., Bang, M., & 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., & James, A. P. (2020). Overview of long short-term memory neural networks. In *Deep Learning Classifiers with Memristive Networks* (pp. 139–153). Springer.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, *35*, 254–265.

Soria-Alcaraz, J. A., Ochoa, G., Sotelo-Figeroa, M. A., & Burke, E. K. (2017). A methodology for determining an effective subset of heuristics in selection hyper-heuristics. *European Journal of Operational Research*, *260*, 972–983.

Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*, 1285–1293.

Thabtah, F., & Cowling, P. (2008). Mining the data from a hyperheuristic approach using associative classification. *Expert systems with applications*, *34*, 1093–1101.

Tyasnurita, R., Özcan, E., & John, R. (2017). Learning heuristic selection using a time delay neural network for open vehicle routing. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1474–1481). IEEE.

Tyasnurita, R., Özcan, E., Shahriar, A., & John, R. (2015). Improving performance of a hyper-heuristic using a multilayer perceptron for vehicle routing. In *15th UK Workshop on Computational Intelligence*. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vidal, T., Crainic, T. G., Gendreau, M., & 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*, 475–489.

Walker, J. D., Ochoa, G., Gendreau, M., & 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* (pp. 265–276). Springer.

Wan, H., Guo, S., Yin, K., Liang, X., & Lin, Y. (2020). CTS-LSTM: LSTM-based neural networks for correlatedtime series prediction. *Knowledge-Based Systems*, *191*, 105239.

Wong, R. T. (1983). Combinatorial optimization: Algorithms and complexity (Christos H. Papadimitriou and Kenneth Steiglitz). *SIAM Review*, *25*, 424.

Wu, Q., Ye, Y., Zhang, H., Ng, M. K., & Ho, S.-S. (2014). Forestexter: An efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems*, *67*, 105–116.

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, *12*, 40–48.

Yates, W. B., & Keedwell, E. C. (2017). Offline learning for selection hyper-heuristics with Elman networks. In *International Conference on Artificial Evolution (Evolution Artificielle)* (pp. 217–230). Springer.

Yi, W., Qu, R., & Jiao, L. (2023). Automated algorithm design using proximal policy optimisation with identified features. *Expert Systems with Applications*, *216*, 119461.

Yi, W., Qu, R., Jiao, L., & Niu, B. (2022). Automated design of metaheuristics using reinforcement learning within a novel general search framework. *IEEE Transactions on Evolutionary Computation*, .

Zhou, L. (2013). Performance of corporate bankruptcy prediction models on imbalanced dataset: The effect of sampling methods. *Knowledge-Based Systems*, *41*, 16–25.