Evolving Systems (2024) 15:99–114
https://doi.org/10.1007/s12530-023-09560-7

ORIGINAL PAPER

Feature-based search space characterisation for data-driven adaptive operator selection

Mehmet Emin Aydin[2] · Rafet Durgut[1] · Abdur Rakib[3] · Hisham Ihshaish[2]

Received: 29 August 2023 / Accepted: 23 November 2023 / Published online: 22 December 2023

Abstract
Combinatorial optimisation problems are known as unpredictable and challenging due to their nature and complexity. One way to reduce the unpredictability of such problems is to identify features and the characteristics that can be utilised to guide the search using domain-knowledge and act accordingly. Many problem solving algorithms use multiple complementary operators in patterns to handle such unpredictable cases. A well-characterised search space may help to evaluate the problem states better and select/apply a neighbourhood operator to generate more productive new problem states that allow for a smoother path to the final/optimum solutions. This applies to the algorithms that use multiple operators to solve problems. However, the remaining challenge is determining how to select an operator in an optimal way from the set of operators while taking the search space conditions into consideration. Recent research shows the success of adaptive operator selection to address this problem. However, efficiency and scalability issues persist in this regard. In addition, selecting the most representative features remains crucial in addressing problem complexity and inducing commonality for transferring experience across domains. This paper investigates if a problem can be represented by a number of features identified by landscape analysis, and whether an adaptive operator selection scheme can be constructed using Machine Learning (ML) techniques to address the efficiency and scalability problem. The proposed method determines the optimal categorisation by analysing the predictivity of a set of features using the most well-known supervised ML techniques. The identified set of features is then used to construct an adaptive operator selection scheme. The findings of the experiments demonstrate that supervised ML algorithms are highly effective when building adaptable operator selectors.

Keywords Adaptive operator selection · Search space characterisation · Supervised machine learning · Artificial bee colonies · Binary optimisation

1 Introduction

Optimisation algorithms are developed with search techniques to find the best-fit solution within a search space. Neighbourhood functions, also known as operators, are the mathematical functions (functional instruments) used in search algorithms to handle moves within a local neighbourhood of search space. As part of the process of population-based problem solving algorithms, where substantial challenges remain, the operators assist in producing new solutions that are evaluated for promotion and replacement with the existing solutions. Various metaheuristic approaches instrumentalise different approaches to promote the produced solutions (Sotoudeh-Anvari and Hafezalkotob 2018). This is to maintain the exploration and exploitation rate as balanced as possible in order to keep the search process diverse and productive. The diversity of the population

✉ Mehmet Emin Aydin
mehmet.aydin@uwe.ac.uk

Rafet Durgut
rdurgut@bandirma.edu.tr

Abdur Rakib
Rakib.Abdur@coventry.ac.uk

Hisham Ihshaish
hisham.ihshaish@uwe.ac.uk

[1] Computer Engineering Department, Bandirma Onyedi Eylul University, Bandirma, Balikesir, Turkey

[2] School of Computing and Creative Technologies, UWE Bristol, Coldharbour Lane, Bristol BS16 1YJ, UK

[3] Centre for Future Transport and Cities, Coventry University, Priory Street, Coventry CV1 5FB, UK

of solutions and the productivity of the operators with respect to the efficiency of search is influenced by the structure of the neighbourhood, which can be characterised with various measures. Many studies, including (Fragata et al. 2019), emphasise the characteristics of the search space and fitness landscape, where more information could be extracted to improve promotion rules and increase success rates.

To preserve diversity and productivity, population-based algorithms designed with multiple operators need to incorporate systematic and efficient selection strategies, which are investigated in various contexts. Adaptive operator selection appears to be a useful avenue to maintain such diversity and richness in the search process in order to avoid potential local optima (Fialho 2010). Although it can also be implemented for individual-solution-driven algorithms, this approach is usually applied to population-based metaheuristics, i.e., evolutionary algorithms (Sun et al. 2020) and swarm intelligence algorithms (Durgut and Aydin 2021). The compelling challenge always enforces to pay more attention in the way how to build the adaptive selection scheme and which kind of information to be used in opting the most suitable operators. Few credit-based approaches have been proposed in the literature (Durgut and Aydin 2021; Wang et al. 2014; Fialho 2010) with reported limitations. The adaptive selection can be achieved enhancing the performance with mapping the problem states to the operators through machine learning, where the representation of the problems plays a crucial role. Binary problem representation has been studied by researchers to accomplish general and transferable approaches at the expense of scalability (He et al. 2018; Santana et al. 2019).

Fitness landscape studies have been attractive for a long time with which more auxiliary information can be extracted and used to identify the search circumstances and characterise the search space. More details can be found in one of the latest reviews (Fragata et al. 2019). Such auxiliary information can be utilised to harvest for representative and discriminating set of features to characterise the search circumstances, while, previously, the problem state based on binary representation was used to help characterise the search circumstances (Durgut and Aydin 2021; Durgut et al. 2022). However, due to the strong dependency on the problem size, the binary representation approach was not scalable for different sizes of problem instances.

The aim of this study is to open a pathway for a scalable adaptive operator selection approach through supervised machine learning techniques. A bespoke set of predictive features are expected to characterise the search space and fitness landscape in order to make the most effective choice when selecting the appropriate actions, such as activating the best-fitting/productive neighbourhood function. It is anticipated that predictive analysis would enable us to explore the causal effects underlying how neighbourhood functions behave in generating neighbouring solutions. This may lead to better representation with respect to the quality of the solution as well as the scalability. The details of the predictive analysis can be found in Nyce (2007). The initial results of this study, which focused on feature analysis, were previously published in Durgut et al. (2022). This article reports the extension of the study with respect to feature analysis and building an adaptive operator selection (AOS) scheme with supervised machine learning approaches. The complete experimental results of the proposed AOS approach integrated into the Artificial Bee Colonies (ABC), the swarm intelligence algorithm, for solving OneMax and set union knapsack (SUKP) problems, which are known as two popular combinatorial optimisation problems. The significance of the proposed approach is to facilitate the transfer of acquired experience to other scenarios, including different problem types, sizes, and new problem instances. The novelty is that we propose a more adaptive approach which resolves the issues of other traditional operator selection approaches while accomplishing scalability.

The rest of this paper is organised as follows; Sect. 2 provides the background and related work, while Sect. 3 presents the framework of the proposed approach in detail including the fitness landscape information items selected for use in characterising the search circumstances and problem states this study. Section 4 presents experiments, statistics, and other relevant analyses. Section 5 concludes the paper outlining the future work.

2 Related work

Data-driven and bottom-up approaches – using data analysis – in characterising unknown problems have been eased and facilitated with the introduction of big data, which escalated to dealing with a huge number of data instances and features. The search spaces in optimisation domain are known as unpredictable and dynamic processes, where the search space size increases exponentially as the number of dimensions grows. Attempts to characterise such search spaces faces increasing the computational complexity of most learning algorithms - for which the number of input features and sample size are critical parameters. To reduce space and computational complexities, the number of features of a given problem should be reduced (Durgut et al. 2020). Many predictors benefit from the feature selection process since it reduces overfitting and improves accuracy, among other things (Chandrashekar and Sahin 2014). In the literature (Wang et al. 2017; Macias-Escobar et al. 2019), fitness landscape analysis has been shown to be an effective technique for analysing the hardness of an optimisation problem by extracting its features. Here, we review some

existing approaches that are most closely related to the work proposed in this paper.

In Wang et al. (2017), the notion of population evolvability is introduced as an extension of dynamic fitness landscape analysis. The authors assumes a population-based algorithm for sampling, two metrics are then defined for a population of solutions and a set of neighbours from one iteration of the algorithm. Due to the exploration process that occurs during each generation, population evolvability can be a very expensive operation. To avoid a computationally intensive operation, the work suggests that the number of sampled generations must be carefully defined. In Macias-Escobar et al. (2019), a very similar approach has been proposed to apply population evolvability in a hyperheuristic, named Dynamic Population-Evolvability based Multi-objective Hyperheuristic. In Tan et al. (2021), the authors proposed a differential evolution (DE) with an adaptive mutation operator based on fitness landscape, where a random forest based on fitness landscape is implemented for an adaptive mutation operator that selects DE's mutation strategy online. Similarly, in both (Sallam et al. 2017) and Sallam et al. (2020), DE embedded with an adaptive operator selection (AOS) mechanism based on landscape analysis for continues functional optimisation problems.

A survey by Karimi-Mamaghan et al. (2022) presented the integration of ML and metaheuristics to tackle combinatorial optimisation problems. Their technical review is focused in the design of various Meta-heuristic elements for different purposes, such as algorithm selection, fitness assessment, initialisation, evolution, parameter setting, and cooperation. Another survey by Malan (2021) summarises recent advances in landscape analysis, including a variety of novel landscape analysis approaches and studies on sampling and measure robustness. It drives attention on landscape analysis applications for complex problems and explaining algorithm behaviour, as well as algorithm performance prediction and automated algorithm configuration and selection. In Teng et al. (2016), the authors propose a continuous state Markov Decision Process (MDP) model to select crossover operators based on the states during evolutionary search. For AOS, they propose employing a self-organising neural network. Unlike the Reinforcement Learning technique, which models AOS as a discrete state MDP, their neural network approach is better suited to models of AOS that have continuous states and discrete actions. However, usually MDP based model computationally expensive due to the state space explosion problem. In Reijnen et al. (2023), for the Adaptive Large Neighbourhood Search heuristic, the authors proposed DR-ALNS, a Deep Reinforcement Learning-based operator-selection technique. It has been used to solve the Time-Dependent Orienteering problem with Stochastic Weights and Time Windows, and they found some comparable results against existing classical Adaptive Large Neighbourhood Search approaches. In Pei et al. (2023), to investigate and quantify the relationship of search operators, the local optima correlation (LOC), a neighbourhood relationship measurement, is developed. On a range of benchmark instances for the capacitated vehicle routing problem, empirical analysis of LOC is conducted. Results demonstrate that a set of commonly used search operators have a consistent relationship. An operator selection framework called AOS-assisted LOC is then put forth with the aim of predicting the local optima of each operator based on data from the early stages of optimisation. To improve the effectiveness of the adaptive large neighbourhood search (ALNS) metaheuristic, the authors of Johnn et al. (2023) have presented an operator selection mechanism based on Deep Reinforcement Learning. They have proposed an operator selection that is dependent on the decision space properties of the current solution. They have shown that it performs better than the traditional Roulette Wheel and random operator selection, and it is possible to scale the model to handle large problem instances by utilising Graph Neural Networks.

Most of these studies have considered population-based landscape metrics to characterise the situation, while some have considered individual-based measures. In addition, the state-of-the-art approaches in literature implemented to solve functional optimisation problems, which are significantly different from combinatorial problems with respect to predictability and characterisation of fitness landscape. Moreover, the approaches used to represent the problem states and search space are either not sufficiently representative or not scalable. In this study, we attempt to use both population and individual-based metrics side-by-side for characterisation of the problem state – for representation purposes – evaluating the impact of each upon the prediction results for this purpose. We follow up with proposing an adaptive operator selection scheme built up with supervised machine learning approaches to solve combinatorial optimisation problems, where two combinatorial problems (binary in this case), known as NP-Hard problems.

3 Materials and methods

3.1 Operator selection for swarm optimisation

A family of optimisation algorithms referred to as *swarm intelligence* works with operators to generate new solutions and drive the swarm intelligence algorithms to search systematically for the optimal solution. However, operators do not always push to operate on suitable circumstances unless studied appropriately. This necessitates the operator selection process to be handled efficiently in order to guarantee a successful and efficient optimisation process.

Operator selection remains one of the problems that researchers attempt to solve in order to increase the effectiveness and performance of optimisation algorithms. It has an effect on the population/swarm diversity, which aids in searching through a search space with an acceptable level of richness. To get the desired efficiency level, the operators are applied in either a random or systematic order. Through the use of a selection scheme, the systematic order can be managed. For instance, genetic algorithms handle the change of operators in accordance with probabilistic rules, whereas variable neighbourhood search imposes a periodic change of operators.

An operator selection scheme assists to select the most appropriate operator subject to given search circumstances. The proposed approach is sketched in Fig. 1 in which a four-stage process is depicted. The first stage, "Random Search", is the phase of generating data through running the algorithm, the bespoke implementation of Artificial Bee Colony (ABC) (Karaboga et al. 2014), for a number of benchmark instances of the problem under consideration, e.g., OneMax or set union knapsack problem (SUKP). The second stage in the complete process is "Feature Selection" in which the most impactful set of features are selected. Then, "Training" is conducted for the best bespoke "Optimisation" algorithm.

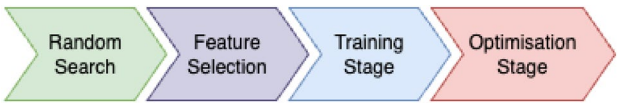

Fig. 1 The complete process for building data-driven operator selection scheme

The training stage follows data generation and pre-processing, including feature selection. This is introduced in Fig. 2, where the necessary data is collated through the state information stored in "Data", and the pool of operators indicated as "Operators". The data is labelled with merging the selected operator for each individual state and then fed into the "Training" component, which runs supervised learning algorithms such as Random Forest (RF), Multilayer Perceptron (MLP), and Support Vector Machine (SVM). "Training" process repeats with the same set of labelled data in various bespoke forms until the learning metrics are satisfied. The output of the system, then, turns to be the trained model, named as "Adaptive Operator Selection Scheme" holding knowledge of the mapped states with operators, accordingly.

Once the mapping between operators and the states has been satisfactorily achieved, the adaptive operator selections scheme would be readily built and can be used by the swarm optimisation algorithm, ABC henceforth, with which the operators will be pulled up from the pool of operators guided by the adaptive selection scheme. Figure 3 presents the logic how to utilise the trained machine learning model, which is inserted in-between the algorithm's evaluation module and the pool of operators. It is worth mentioning that the actual ABC algorithm employs bees to search through problem states; it selects one operator from the pool in accordance with the problem state enforced by the adaptive selection scheme, which predicts the most suitable operator for generating the next position of the bees.

The ABC algorithm implemented here is inspired by Durgut and Aydin (2021) and Durgut et al. (2022), which works with binary representation and a set of binary operators. The implemented ABC is sketched in Algorithm 1 presented in the Appendix. Lines 5 and 18 of the algorithm are the steps

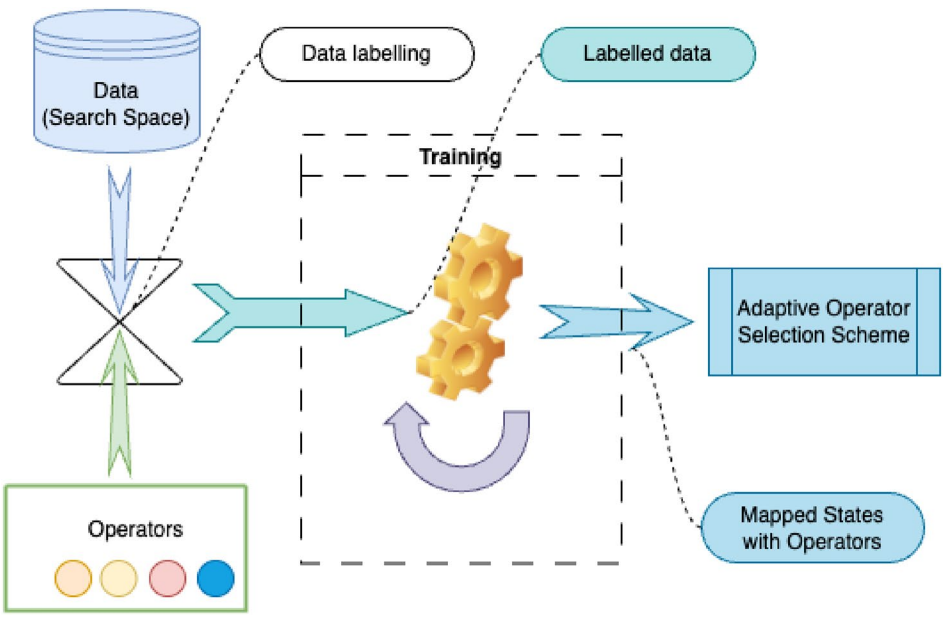

Fig. 2 Training stage of operator selection scheme

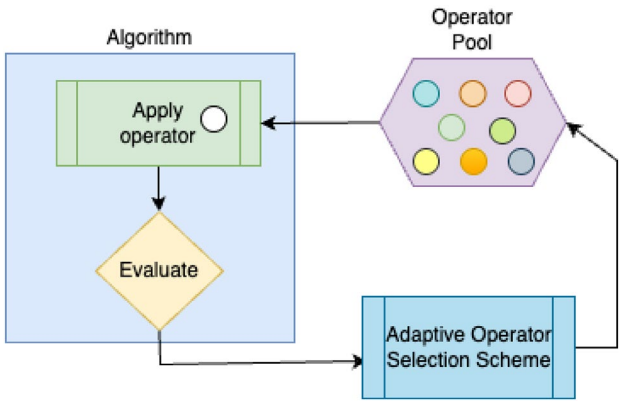

Fig. 3 Adaptive operator selection process trained with supervised machine learning algorithms interacting with ABC algorithm in each bee generation

where the Adaptive Operator Selection Scheme takes action to select the best fitting operator to the problem state in hand. Four state-of-art binary operators have been embedded in the pool of operators as part of the ABC implementation used in this study: *flip*ABC() inverts any randomly selected bit, while novel binary ABC (*nb*ABC) (Santana et al. 2019) crossovers n bits from a randomly chosen neighbouring solution. Likewise nABC (Xiang et al. 2021) crossovers a number of bits up to n in a randomised way. Finally, improved binary ABC (*ibin*ABC) (Durgut and Aydin 2021) applies XOR operator – normalised with the number of iterations – to two chosen solutions.

The complexity of the ABC algorithm has been studied in Wang et al. (2022), where the complexity of the algorithms is demonstrated to be $\mathcal{O}(\phi * |P|)$ working with a variety of ABC versions. The complexity does not change in this implementation, either, since only two simple operations are added. Here, $|P|$ is the size the swarm (colony), while the problem's self complexity is represented with ϕ. Neither the operator selection function nor any of the operators require any variable looping as part of the$$\{\mathop a\limits^{{..}} ir process, hence, the complexity remains the same.

3.2 Feature analysis and landscape features

Feature analysis and selection is the second stage of the process sketched in Fig. 1 with which the features identified to represent the problem state are reviewed and analysed with respect to their emphasis. The first, third, and fourth stages in the figure have been introduced in the previous section. Numerous previous research on the adaptive operator selection process took into account problem representation with a set of binary features, which were subsequently found to be non-scalable. The approach outlined below is scalable and makes use of the findings from earlier research.

Table 1 Population-based features

Feature	Formula
Population solution diversity (psd)	$\ddot{\mathbf{a}}_1 = \frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n}\|p_i - p_j\|}{D\frac{n(n-1)}{2}}$
Population fitness deviation (pfd)	$\ddot{\mathbf{a}}_2 = \frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n}\|f_i^p - f_j^p\|}{\frac{n(n-1)}{2}}$
Population of new best children (pic)	$\ddot{\mathbf{a}}_3 = \frac{\|c_i \| f_i^c > f_i^p\|}{N}$
Proportion of new improving children (pnd)	$\ddot{\mathbf{a}}_4 = \frac{\|c_i \| f_i^c > g_{best}\|}{N}$
Proportion of amount of improvements (pai)	$\ddot{\mathbf{a}}_5 = \frac{(f_i^c - f_i^p)/f_i^c \| f_i^c > f_i^p}{N}$
Proportion of convergence velocity (pcv)	$\ddot{\mathbf{a}}_6 = \frac{E[max(F^c) - max(F^p)]}{max(F^p)}$
Proportion of convergence reliability (pcr)	$\ddot{\mathbf{a}}_7 = \frac{E[\|\|x^* - x_i\| - \|x^* - x_{i+1}\|\|]}{D}$
Evolutionary ability of population (eap)	$\ddot{\mathbf{a}}_8 = \sum_{i \in N^*} \frac{\sigma(P)\|f^*(P) - f(Cf_i)}{N}$
Evolvability of population (evp)	$\ddot{\mathbf{a}}_9 = \ddot{\mathbf{a}}_8 \times \ddot{\mathbf{a}}_4$
Proportion of average trial Number (atn)	$\ddot{\mathbf{a}}_{10} = \frac{\sum_i^n tn_i}{N}$
The diameter of population (pdd)	$\ddot{\mathbf{a}}_{11} = max_{i,j \in \{P,C\}} \|p_i - c_i\|$

Fitness landscape analysis provides representative information, which can be used in the characterisation of the search space and the position of the state of the problem at hand. The most representative information can be established from a large body of literature that has been developed over the past few decades. For example, the relevant literature can be found in Fragata et al. (2019); Ochoa and Malan (2019); Pitzer and Affenzeller (2012).

Diversity is one of the very important aspects of swarms to help characterise states (Erwin and Engelbrecht 2020), while Wang et al. (2017) discuss the evolvability of populations with dynamic landscape structure. Let $\mathcal{A} = \{\mathbf{a}_i | i = 1 \ldots |\mathcal{A}|$ be the set of attributes, features, with which the problem states are characterised, where the set consists of the attributes for individual solutions $\dot{\mathcal{A}}$ and the attributes of population of solutions, $\ddot{A}$. This makes up a union; $A = \dot{A} \cup \ddot{A}$.

A number of features can be retrieved from state of art the literature as listed in Tables 1 and 2. The population-based metrics – considered as attributes, henceforth feature– are listed in Table 1 with corresponding calculation details. The first 5 metrics, $\{\ddot{a}_i | i = 1 \ldots 5, \ddot{a}_i \in A..\}$, have been collected from Teng et al. (2016) and implemented for (i.e. adjusted to) artificial bee colony algorithm (ABC), which is one of the very recently developed highly reputed swarm intelligence algorithms (Karaboga et al. 2014). In order to adjust them to binary problem solving, the metrics calculated based on distance measure

Table 2 Individual solution-based features

Feature	Formula
Distance between g_{best} and parent solutions (idg)	$\dot{\mathbf{a}}_1 = \frac{\|x^* - p_i\|}{D}$
Distance between parent and child solutions (idp)	$\dot{\mathbf{a}}_2 = \frac{\|p_i - c_i\|}{D}$
Fitness gap between g_{best} and child solutions (ifg)	$\dot{\mathbf{a}}_3 = (f^{x^*} - f_i^c)/f^{x^*}$
Fitness gap between the parent and the offspring (ifp)	$\dot{\mathbf{a}}_4 = (f_i^c - f_i^p)/f_i^c$
Distance between p_{best} and parent solutions (idb)	$\dot{\mathbf{a}}_5 = \frac{\|p_{best} - p_i\|}{D}$
Distance between p_{worst} and parent solutions (idw)	$\dot{\mathbf{a}}_6 = \frac{\|p_{worst} - p_i\|}{D}$
Proportion of Trial number (itn)	$\dot{\mathbf{a}}_7 = \frac{trial_i}{trial_{max}}$
Success rate of each operator (osr)	$\dot{\mathbf{a}}_{8,j} = \frac{sc_j}{tc_j}$

have been binarised using Hamming distance as in Erwin and Engelbrecht (2020). The metrics, $\{\ddot{a}_i | i = 6 \dots 9, \ddot{a}_i \in \ddot{A}\}$, are introduced and proposed in Wang et al. (2017) with sound demonstration, while $\ddot{\mathbf{a}}_{10}$ is obtained from the trail index used in ABC and utilised to measure/observe the iteration-wise hardness in problem solving. In addition, $\ddot{\mathbf{a}}_{11}$ is taken from Anescu and Ulmeanu (2017) to calculate the distance between two farthest individuals within a population/swarm.

The literature includes more metrics calculated through local search procedures. However, these kind of features, i.e. metrics, have been left out due to the scope of the study. In fact, it is known that access to preliminary information on search is not easy, hence, we encompass the change in instant search in formation online decision making.

The base notation of population-based features is as follows. Let $P = \{p_i | i = 0, 1, ..., N\}$ be the set of parent solutions and $C = \{c_i | i = 0, 1, ..., N\}$ be the set of children solutions reproduced from P, where each solution has D dimensions. Also, let $F^p = \{f_i^p | i = 0, 1, ..., N\}$ be the set of parent fitness values and $F^c = \{f_i^c | i = 0, 1, ..., N\}$ be set of children fitness values. g_{best} represents the best solution found so far and p_{best} represents the best solution in the current population.

On the other hand, a number of metrics, features, can be obtained from the auxiliary information of individual solution, which seem to serve efficiently in individual-specific aspects with which the operators can act upon significantly on case basis. Individual-related characteristics are tabulated in Table 2, which are mostly proposed by Teng et al. (2016) except $\dot{\mathbf{a}}_7$. $\dot{\mathbf{a}}_7$ has been introduced in this study for the first time. The last feature is the success rate per operator, i is calculated with $\dot{\mathbf{a}}_{8,j} = \frac{sc_j}{tc_j}$, where sr is the success counter and tc is the total usage counter.

4 Experimental results

The experimental results have been collected over multiple runs of an Artificial Bee Colony algorithm developed for earlier investigations embedded with a pool of operators selected at random each time a new solution is generated. Every successful move made during algorithm execution has been identified as a successful case and labelled accordingly.

The two well-known combinatorial optimisation problems have been considered as a test-bed; One-Max (Goëffon and Lardeux 2011) as a unimodal and Set Union Knapsack (SUKP) (Lin et al. 2019) as a multi-model problem. The size of benchmark problems taken under consideration for One-Max and SUKP are 1000 and 500, respectively, while the maximum number of iterations are 150 and 500, respectively.

The preliminary experimentation demonstrated that the level of hardness and complexity very much depends on the progress of the search process. Hence, the entire search period is divided into three phases as it is expected that the behaviour of the operators would vary significantly over the time and stage of iterations. The subsequent subsection provides relevant analysis.

4.1 Data generation and labelling

The first stage depicted in Fig. 1 is about running the implemented ABC algorithm for data generation purposes. The data is generated by a random operator selection scheme embedded into the proposed artificial bee colony optimisation algorithm for both problem types under consideration, OneMax and SUKP, and will be used for feature analysis, training, and testing purposes in the future. The implemented ABC algorithm has been devised with a pool of operators, as mentioned above, and a random selection scheme to select an operator each time needed for generating a new solution. Then the metrics mentioned above, $A = \dot{A} \cup \ddot{A}$, have been calculated to set up the values for each feature to represent the solution. Given the circumstances, the child solution, $c_{k,l}$ will be represented with $c_{k,l} = \{a_{k,l,i} | i = 1 \dots 19, a_{k,l,i} \in A\} \Leftarrow \{\ddot{a}_i | i = 1 \dots 11, \ddot{a}_i \in \ddot{A} \bigcup \dot{a}_i | i = 1 \dots 8, \dot{a}_i \in \dot{A}\}$, where k is the iteration index, while $l = 1 \dots N$ is the index of the solution within the population, i.e., the swarm. The solutions are paired with the operators selected, $o_i = \arg\min_{o_i \in O} \|o_i - r\|$, where $\mathbf{r}$ is a ran-

domly generated value. The new data point generated will be $\langle c_{k,l}, o_i \rangle$, where $c_{k,l}$ represents the solution state and o_i stands for the selected operator. The complete data set is generated accordingly and formatted in this way in order to label the data for later use.

The difficulty through search process varies. Due to this fact, the behaviour of operators changes throughout the process. In order to achieve better learning, we decided to split the search timeline into three phases, and treat each of these phases separately and independently. It is well recognised that improving performance is simpler to accomplish in the early stages of the search process than in the middle stages, but improvement in performance is still feasibly attainable. However, a positive gain in performance turns to be very difficult in later, i.e., in the final stage of the process. As a result, we have treated the complete search timeline as a three-phase process, where the behaviours of operators can alter. The data has been generated to cover all three phases equally, where the compete data set, D, is defined to be $D = D_1 + D_2 + D_3$, where D_1, D_2, and D_3 are the subsets of data, D representing the three phases of the search process, respectively. This factor has been considered in order to design training and test sets that are efficient and fair.

4.2 Feature exploratory analysis

The second stage of the framework described in Fig. 1 is to analyse the features for efficient algorithmic design. A set of exploratory analyses are conducted to explore both the relevance of input features as well as their relative importance to the task of operator selection. The latter is discussed further in Sect. 4.3.1. The tests are analysed and evaluated for each phase of the search process separately. That is, given the set of all input characteristics, $A \subset \mathcal{A}$, the objective is to examine if a subset $A' \in A$ is associated with the *target* success operators, corresponding to each search phase. The assumption made here is based on whether feature membership for A' is consistent. This in turn can be used to indicate the features that are most prevalent at predicting *success* operators for each search phase, and if comparable across the two different optimisation problems. In the first test, which is depicted in Fig. 4 for the One-Max problem and Fig. 5 for the SUKP, the strength of the linear relationship between input features with respect to each search phase was assessed.

There is clearly apparent linearity – as additionally expected, both positive and negative – among different

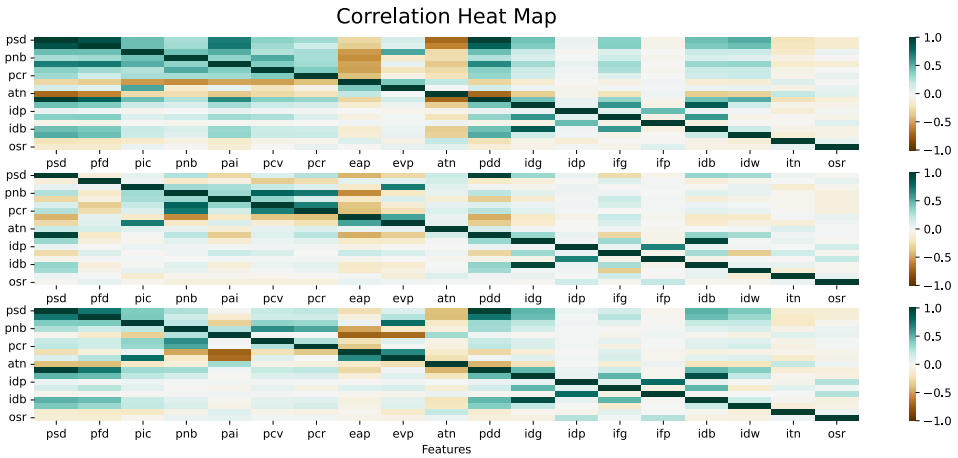

Fig. 4 Pearson correlation coefficient matrix for the features applied to One-Max problem. The matrices are ordered top-down per search phase; top is earlier and bottom is the final stage

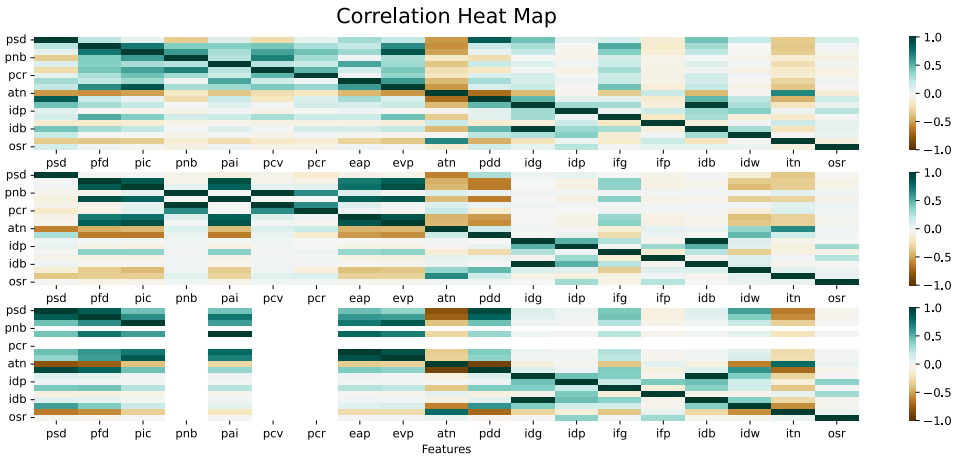

Fig. 5 Pearson correlation coefficient matrix for the features applied to SUKP problem. The matrices are ordered top-down per search phase

groups of features in both optimisation problems. Furthermore, the strength of relationship exhibits variability across the different search phases. In general, although relative strength of association can serve as a guide for feature selection processes, further analysis and evaluation of feature importance in relation to operator selection is still necessary. In particular, we test whether the selected subset of features can accurately learn the target variables, i.e., *success* operators, associated with each optimisation problem where membership in A' can be relatively stable between the two optimisation problems.

Accordingly, for both the One-Max and SUKP problems, the Chi-square (χ^2) test – a test on whether two variables are related or independent from one another– is conducted to examine the dependency of the response variable (*success* operator) on the set of input features. χ^2 statistic, computed for each pair of feature classes, provides a score on the relative dependency between the values of each attribute and the different target classes. In order to predict the target class, i.e., the search operator, attributes with higher values for the statistics, χ^2, can be considered as highly impactful. As a result, these attributes are typically chosen as input features in classifying the operators.

The resulted ranking with χ^2 for input features related to the both optimisation problems (i.e., OneMax and SUKP) is shown in Fig. 6, where the top bar chart is plotted with OneMax results and the bottom is with SUKP. The bar chart includes three bars with colour code labelled as 1.0, 2.0 and 3.0 representing three phases of the search process, which are the early, the middle, and late phases, respectively. The level of difficulty increases significantly through the labels across the stages. These labels are applied to all the figures, Figs. 6, 7 and 8.

Although these appear to differ in importance between the two problems – specifically, it seems that SUKP has more significant features than One-Max – there is still an interesting overlap between the two in terms of a subset of (dominant) input features $\{\dot{\mathbf{a}}_2, \dot{\mathbf{a}}_4, \dot{\mathbf{a}}_{8,j}\}$ – labelled in the figures with *idp*, *ifp*, *osr* – as well as an agreement on the relative irrelevance of further features to search operators. This additionally persists across the three search phases corresponding to both the problems under consideration. It is interesting to see that the ranks of the early phase of the search are more apparent than the others, which indicates the significance of the features across the phases. Although such finding can result primitive – not the least conclusive given the nature of the examined problems – the resulted similarity can nonetheless be critical to examining potential prospects leading to learning a solution path (or important features) from one problem to another.

4.3 Building adaptive selection scheme with supervised learning

The third phase, as described in Fig. 1, is the process in which an adoptive data-driven operator selection scheme

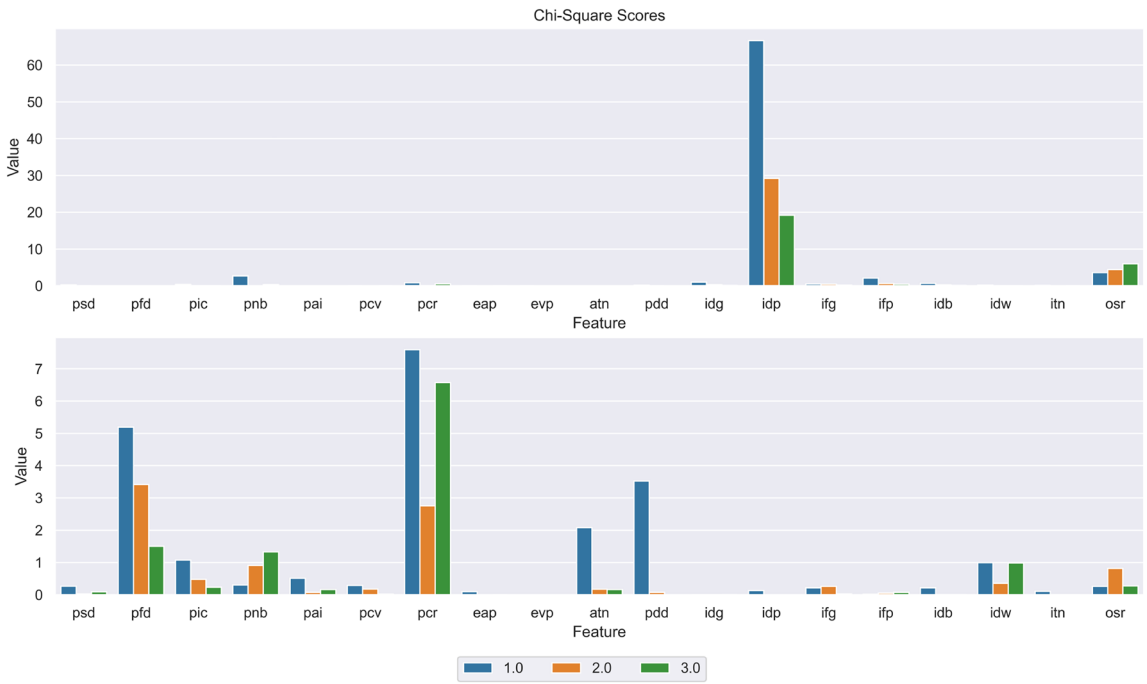

Fig. 6 Ranking with χ^2 for input features on successful search operators. Again, in both top (OneMax) and bottom (SUKP), ranking is ordered top-down per search phase

is developed via supervised machine learning approaches. As seen in Fig. 2, the data is prepossessed, labelled, and used for training and testing purposes towards building up an adaptive scheme. These methods, which use classifiers based on the MLP, SVM, and RF algorithms, have been used for feature analysis as well as for developing an adaptable scheme. Operator selection and related supervised machine learning algorithms will be introduced in the following subsections, followed by discussions of each algorithm's performance. To further compare the performances, the problem instances of the two chosen problems-OneMax and SUKP-will then be solved.

4.3.1 Operator selection/classification

Feature analysis helps derive insights into the impact of each feature in order to fine-tune the ML models for efficiency purpose. To assess the possible transferability of selected features from one search domain to another, the prediction of the different *success* operators at each search phase corresponding to the two different optimisation problems is subsequently evaluated. The *success* of operators relative to each search problem and each phase are shown in Table 3. This provides the setting for a supervised classification task in which problem features are the independent variables and the corresponding *success* operators are the target class.

In this regard, the *success* operators are predicted using three supervised classifiers; a Multi-layer Perceptron (MLP) with one hidden layer (feed-forward ANN with 'adam' solver), Support Vector Machine (SVM) classifier with radial basis function (RBF) kernel, and a Random Forest (RF) classifiers of size 200. All the models have been used in classification tasks very widely for decades, and the particular choice for RF and SVM was additionally due to their ability to provide explicit feature importance ranking alongside their prediction, which we aim to utilise in the proposed hypothesis. We report the accuracy score as the prediction measure of accuracy in Table 4.

Interestingly, the performance of the classifiers on both optimisation problems is relatively comparable. With the exception of SVM on One-Max, which seems to be underperforming than it does on SUKP, the predictability of success operators from both individual as well as population domain features is consistent. It should be noted that the reported performance of the three classifiers can be tuned for further optimisation, which we aim at providing in a further study. In this study, however, the aim is to examine whether the predictability of *success* operators can be achieved with a subset of input features learnt in different search problem(s). In such a way, the relative importance of input features for the classification tasks is computed and compared; the weighted coefficients of feature vectors in the SVM classifier, as well as the importance of features from the resulted Random Forest classifier, normalised across the 200 Decision Trees between 0 and 1. The results are shown in Fig. 7 for the One-Max problem and Fig. 8 for SUKP.

Once again, the results show promising findings, as a subset of features can be seen to have similar relative importance across both search problems. In fact, this emphasises the suggestion, as observed earlier in the Chi-sqaure test results, that there seems to be a subset of *effective* features, like A', to the task of operator selection that can be transferable from one problem to another. It is worth mentioning that in both Figs. 7 and 8, the relative feature importance is computed for the whole set of features, as the SVM considers weighing all input attributes, and the RF calculates class impurity – relative Shannon entropy – weighted by the probability of reaching the target class (*success* operator) corresponding to all features as these are re-sampled across 200 trees, and subsequently their scores normalised. That is to say, when selecting the subset of *effective* features, their relative importance should be considered instead of the values assigned to them.

Table 3 Success of operators for One-Max and SUKP search problems

Problem	Operator	Phase 1	Phase 2	Phase 3	Mean
One Max	OP 0	306	375	357	346.00
	OP 1	234	218	235	229.00
	OP 2	304	405	456	388.33
	OP 3	323	328	316	322.33
SUKP	OP 0	104	200	245	183.00
	OP 1	494	150	89	244.33
	OP 2	1397	1368	1487	1,417.33
	OP 3	916	777	649	780.67

Table 4 The accuracy results for both problem types achieved by machine learning approaches across 3 phases

	One Max			SUKP		
	RF	SVM	MLP	RF	SVM	MLP
Phase 1	0.79	0.52	0.70	0.71	0.62	0.68
Phase 2	0.85	0.63	0.73	0.79	0.72	0.75
Phase 3	0.84	0.65	0.71	0.83	0.77	0.80
Mean	0.83	0.60	0.71	0.77	0.71	0.74

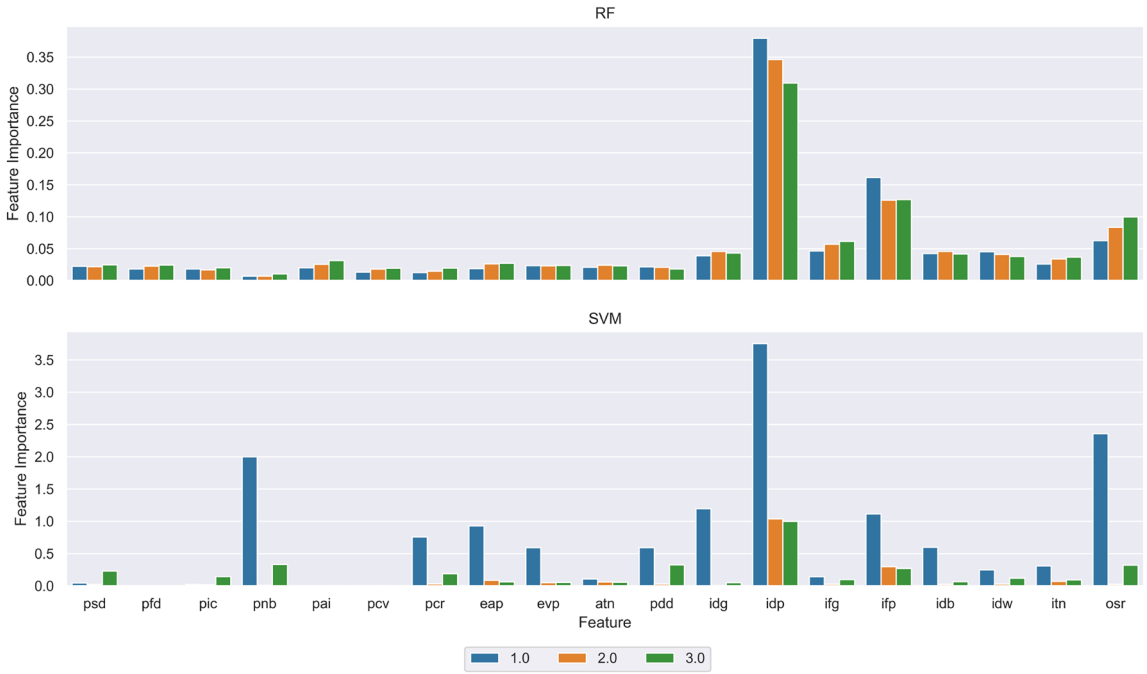

Fig. 7 Feature importance ranking for One-Max problem

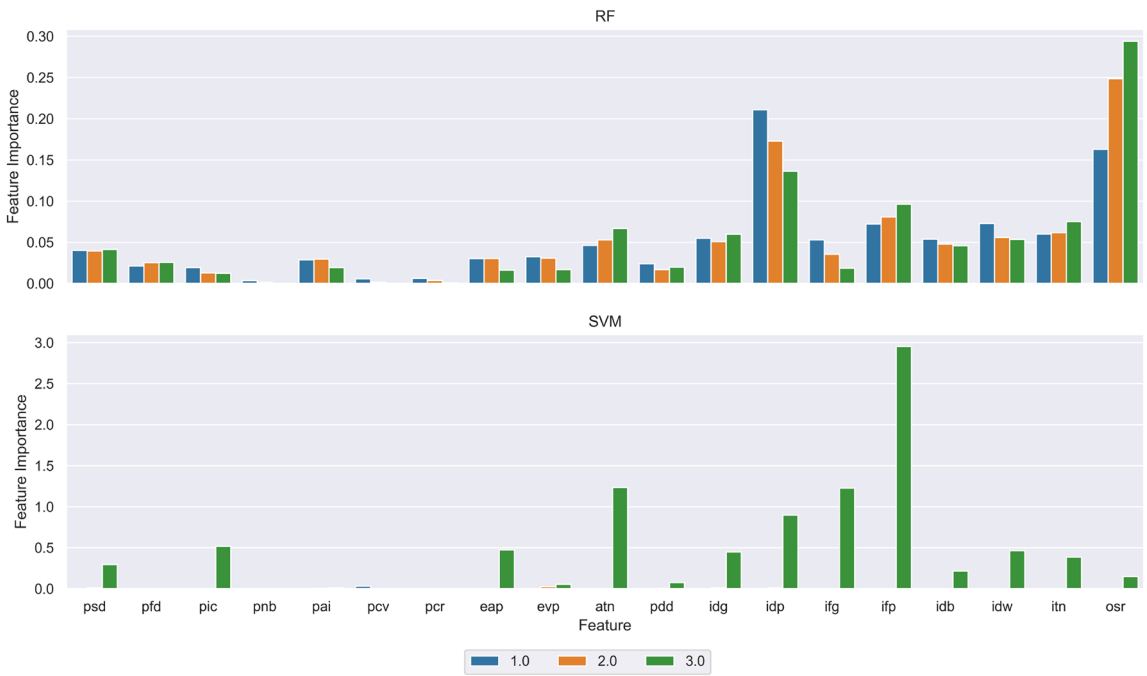

Fig. 8 Feature importance ranking for SUKP problem

The assessment on what specific features are most prevalent to the *success* operator selection, and why can be 'overenthusiastic' at this stage, especially so as this would require extensive characterisation of both search problems, which will be evaluated further in a later study. Here, however, the argument on finding a transferable A' from one search problem to another seems plausible. For this, the extent of predictability (solution quality) and robustness as

features are reduced and transferred across different search domains should be examined further.

4.3.2 Comparative results

Further to analysis and discussions provided over feature analysis and operator selection performances, this subsection is to dive down into comparative results of solving the instances of both One-Max and SUKP problems. Tables 5 and 6 present the details of the comparative results produced with all three supervised ML algorithms and random selection for solving One-Max instances by using the full set of features and selected 10 most impactful features, respectively. The results are shown in both tables using "Max" and "Mean" metrics, which make it easy to observe how significantly each algorithm improved against "Random Selection" in each case. The "Rank" column shows the relative performance in ranking order, with "Random Selection" appearing to outperform ML algorithms over the first 3-4 easier and smaller instances, but ML algorithms certainly outperform them over all remaining larger instances. In Tables 5 and 6, RF appears to be outperforming all with a consistent lowest rank of 1.95 and 1.324. Meanwhile, SVM appears to be performing better than MLP in both cases.

For a particular instance of the One-Max problem with 5000 dimensions, Fig. 9 plots the convergence data obtained from each of the six variants of ABC – embedded with operator selection schemes built with the variants of ML algorithms under consideration and taking either the full set or selected set of features. It appears that the convergence index, which is the value of the objective function as the quality of solution, starts drifting away around iteration 50, and carries on accordingly. The last 50 iterations are zoomed in the subplot in order to observe the differences more clearly. As seen, convergence appears higher with feature selection (FS) variants in comparison to the full set featured ones; RF, SVC, and MLP models have converged slower than their FS variants. Among all the three ML algorithms, RF remains as the fastest model in convergence speed, while the models with the selected 10 most impactful features help increase the performance. Although MLP-FS is not competitive enough with RF-FS and SVR-FS, it gains improvement in producing better solutions.

The comparative results for SUKP problem instances are provided in Tables 7 and 8 with full set of features and selected 10 features based on impacts identified through feature analysis, respectfully. Both tables present the results in two metrics alongside the ranks within the competitors; "Max" and "Mean". The ranks are calculated with respect to "Mean" measures, where all algorithms, i.e., RF, SVM and MLP, remain comparative with slight differences in performance. SVM performs clearly worse than RF and MLP in both cases; with and without feature selection, while MLP does better than RF, but RF-FS has higher average rank than MLP-FS. This suggests that feature selection contributes to

Table 5 Comparative results by supervised machine learning algorithms to build adaptive operator selection scheme for solving OneMax problem instances

	Random selection			RF			SVC			MLP		
ID	Max	Mean	Rank	Max	Mean	Rank	Max	Mean	Rank	Max	Mean	Rank
1	500	499.87	2	500	498.93	4	500	499.97	1	500	499.27	3
2	750	748.47	1	748	742.57	4	750	747.83	2	749	745.13	3
3	993	987.70	1	989	980.87	4	992	987.23	2	988	982.87	3
4	1221	1209.83	3	1221	1206.47	4	1221	1212.93	1	1218	1210.43	2
5	1430	1413.93	4	1433	1425.27	3	1447	1427.33	1	1437	1426.27	2
6	1629	1612.57	4	1658	1635.33	1	1645	1629.33	3	1645	1634.47	2
7	1827	1801.33	4	1856	1832.83	2	1854	1830.50	3	1851	1834.03	1
8	2014	1987.87	4	2043	2019.53	3	2049	2024.70	1	2045	2023.50	2
9	2195	2171.80	4	2232	2212.93	1	2233	2207.83	2	2236	2207.43	3
10	2383	2347.70	4	2433	2403.43	1	2429	2389.03	3	2417	2396.97	2
11	2557	2524.30	4	2613	2577.50	1	2609	2574.37	2	2614	2573.40	3
12	2723	2698.10	4	2778	2755.87	1	2786	2748.70	3	2784	2753.33	2
13	2889	2864.23	4	2964	2929.73	1	2961	2922.77	3	2953	2924.03	2
14	3073	3034.10	4	3146	3106.30	1	3158	3097.50	2	3124	3088.50	3
15	3245	3199.87	4	3306	3273.93	1	3320	3265.30	2	3294	3256.70	3
16	3401	3364.97	4	3479	3432.13	2	3488	3439.27	1	3480	3425.37	3
17	3558	3524.10	4	3643	3605.40	1	3651	3598.70	3	3646	3599.10	2
18	3723	3690.00	4	3810	3769.80	1	3817	3754.97	3	3812	3760.37	2
19	3901	3854.00	4	3979	3938.33	1	3975	3931.10	2	3962	3928.63	3
Avg:			3.53			1.95			2.11			2.42

Table 6 Comparison of ML models with 10-features for OneMax Problem

	Random selection			RF			SVC			MLP		
ID	Max	Mean	Rank	Max	Mean	Rank	Max	Mean	Rank	Max	Mean	Rank
1	500	499.87	1	500	499.63	2	500	499.23	3	500	499.20	4
2	750	748.47	1	750	745.43	2	749	743.83	3	747	743.63	4
3	993	987.70	1	990	980.73	3	988	979.23	4	990	980.77	2
4	1221	1209.83	2	1222	1212.70	1	1221	1207.57	4	1215	1208.40	3
5	1430	1413.93	4	1449	1433.27	1	1444	1428.17	3	1441	1428.20	2
6	1629	1612.57	4	1667	1646.10	1	1658	1641.93	2	1656	1638.73	3
7	1827	1801.33	4	1876	1850.67	1	1858	1843.90	3	1867	1844.70	2
8	2014	1987.87	4	2073	2048.70	1	2072	2048.30	2	2057	2036.67	3
9	2195	2171.80	4	2266	2242.60	1	2256	2231.93	3	2256	2233.63	2
10	2383	2347.70	4	2462	2429.10	1	2455	2426.27	2	2437	2416.70	3
11	2557	2524.30	4	2646	2615.53	1	2641	2615.20	2	2618	2598.13	3
12	2723	2698.10	4	2839	2797.90	1	2823	2791.90	2	2820	2780.77	3
13	2889	2864.23	4	3014	2977.07	1	2998	2973.83	2	2988	2961.87	3
14	3073	3034.10	4	3178	3150.67	2	3190	3152.53	1	3169	3139.27	3
15	3245	3199.87	4	3371	3329.53	1	3352	3325.73	2	3345	3308.40	3
16	3401	3364.97	4	3545	3496.60	2	3544	3499.40	1	3509	3475.67	3
17	3558	3524.10	4	3704	3671.23	1	3690	3665.87	2	3692	3650.97	3
18	3723	3690.00	4	3877	3846.37	1	3896	3837.80	2	3858	3817.47	3
19	3901	3854.00	4	4050	4013.33	1	4054	4011.63	2	4035	3989.37	3
Avg:			3.42			1.32			2.37			2.90

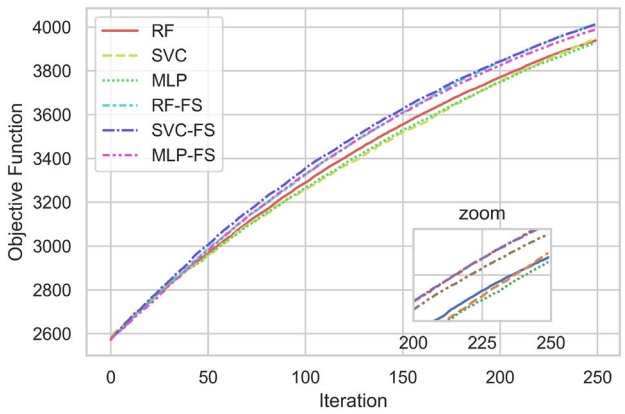

Fig. 9 Convergence graph on 5000 dimension OneMax problem

RF more than does to MLP despite the average rank score is slightly different.

Figure 10 plots the comparative results collected from the state-of-the-art approaches and RF-FS as the winner of this study. The results by GA (Schmitt 2001) and binDE (Engelbrecht and Pampara 2007) have been taken from the results tabulated in BABC (He et al. 2018), while GPSO are taken from Ozsoydan and Baykasoglu (2019). The performance indicator is on the vertical axis, while the instances are on the horizontal one. It is apparent that RF-FS sits on the top of all scattered graphs as seen in blue label. The results by RF and RF-FS for SUKP problem instances remain comparable with some other recent state-of-the-art works such as using Reinforcement Learning (RL) (Durgut and Aydin 2021; Durgut et al. 2022) for this purposes applying online learning policies. Since the set of operators are not the same, a direct comparison would not be fair, but, it seems the results look at least comparable.

5 Conclusions and future work

Development of an adaptive operator selection remains challenging, where a successfully developed scheme can be used to select the best operator given the search space and neighbourhood circumstances, this research has introduced an exploratory study to investigate a number of supervised machine learning algorithms. A predictive analysis has been applied in order to reveal the impact of the identified features and their domination if the full set of features or a selected subset to be used for characterisation of search space and the problem states for optimisation purposes. The idea is to identify the set of most impactful and prominent features that best represent a problem state and its standing within its neighbourhood so that the best fitting neighbourhood function, i.e., operator, among many alternatives can be selected to generate the next problem state avoiding local optima for higher efficiency in search

Table 7 Comparative results ML models with full set of features.for solving SUKP instances

	RF			SVC			MLP		
ID	Max	Mean	Rank	Max	Mean	Rank	Max	Mean	Rank
1_1	13044	13041.20	3	13044	13042.13	2	13167	13052.20	1
1_2	12130	11951.77	1	12145	11890.33	2	12130	11888.30	3
1_3	13402	13118.53	2	13271	13083.23	3	13319	13125.43	1
1_4	13671	13243.73	1	13671	13180.13	3	13660	13242.87	2
1_5	10845	10454.23	3	10861	10459.23	2	10834	10462.13	1
1_6	12245	11340.70	3	12035	11358.03	1	12012	11341.50	2
1_7	11244	10596.53	1	10991	10455.12	3	10974	10546.37	2
1_8	10168	10053.93	1	10168	10013.97	3	10168	10037.67	2
1_9	11288	10967.23	3	11399	11047.87	2	11399	11066.20	1
1_10	9486	9090.37	2	9296	9014.70	3	9628	9131.87	1
2_1	14044	13838.57	2	13963	13851.93	1	14044	13837.97	3
2_2	13407	13258.50	1	13407	13140.13	3	13498	13229.77	2
2_3	12328	11755.60	1	12007	11669.70	3	12328	11741.17	2
2_4	12317	11415.70	3	11821	11500.53	2	11817	11501.67	1
2_5	12677	12473.90	1	12644	12392.60	3	12644	12416.70	2
2_6	10724	10542.90	1	10735	10439.53	3	10782	10529.27	2
2_7	11048	10773.23	1	11154	10740.00	3	11055	10767.17	2
2_8	10355	9596.73	3	10355	9702.80	2	10355	9704.53	1
2_9	10735	10506.60	3	10700	10517.97	2	10700	10597.23	1
2_10	9896	9555.13	3	10194	9617.20	2	10194	9653.50	1
3_1	11851	11358.27	2	11752	11380.40	1	11659	11357.87	3
3_2	12369	11795.90	2	12369	11845.27	1	12369	11752.33	3
3_3	13355	13094.73	2	13374	13060.83	3	13374	13129.07	1
3_4	10920	10619.80	1	10831	10603.97	3	10920	10616.37	2
3_5	11538	11112.50	2	11538	11064.10	3	11538	11175.90	1
3_6	11377	10852.17	2	11241	10820.00	3	11226	10889.47	1
3_7	9976	9873.60	3	10088	9886.03	1	9976	9882.67	2
3_8	9617	9207.97	1	9491	9160.37	3	9574	9204.87	2
3_9	10720	10436.47	2	10728	10421.57	3	10720	10502.70	1
3_10	9514	9285.83	3	9596	9287.90	2	9654	9306.97	1
Avg			1.97			2.37			1.67

process. A swarm intelligence algorithm – Artificial Bee Colony – has been used with a pool of neighbourhood functions, i.e., operators, to solve two different types of combinatorial optimisation problems utilising an adaptive operator selection scheme. The set of most prominent features are elicited through a rank of weights using statistical and machine learning methods. The analysis demonstrated that a set of features mostly including individual features are found to be more discriminating than those of population-based metrics.

The research has shown that supervised machine learning techniques, such as Random Forest, Support Vector Machine, and Multi-layer Perceptron, are very useful in developing adaptive operator selections. For both combinatorial problems Random Forest provided the best results. However, the runner approach varies; SVM performs better than MLP in One-Max but worse in SUKP. The validity of the results has been verified by comparing the winner approach's success with the state-of-the approaches.

The interesting preliminary finding of the study is that even though the problem domain has changed, the most effective features have largely remained the same. This suggests that the information can be transferable between different problem domains. There are a number of interesting directions in this area for future research. For example, the success of transfer learning through the problems needs to be investigated in terms of robustness and solution quality. For dynamic and more realistic problems, the set of features and more data will be used considering Deep Learning techniques as well as active and Reinforcement Learning.

Table 8 Comparative results of ML models with the set of selected features for solving SUKP instances

	RF-FS			SVC-FS			MLP-FS		
ID	Max	Mean	Rank	Max	Mean	Rank	Max	Mean	Rank
1_1	13167	13848.80	1	13167	13840.80	3	13044	13847.50	2
1_2	12348	13180.30	3	12479	13214.80	2	12130	13225.80	1
1_3	13405	13046.20	2	13293	13048.10	1	13340	13043.10	3
1_4	13671	11912.90	3	13685	12002.90	1	13900	11966.40	2
1_5	10818	13082.80	1	10780	13052.30	2	10946	12998.40	3
1_6	12151	10637.00	2	12040	10677.00	1	12135	10615.80	3
1_7	11052	13116.40	1	11244	13111.90	3	11244	13114.10	2
1_8	10259	13358.50	1	10168	13290.60	3	10168	13319.90	2
1_9	11398	11734.30	1	11399	11667.60	3	11309	11683.10	2
1_10	9712	11489.70	1	9419	11445.30	3	9444	11475.50	2
2_1	13963	11117.10	3	14044	11153.10	2	13963	11219.70	1
2_2	13498	10831.30	2	13407	10798.10	3	13407	10841.90	1
2_3	12328	10500.90	3	12271	10526.40	1	12350	10521.30	2
2_4	11800	11584.50	1	11903	11407.60	3	12187	11440.30	2
2_5	12644	12436.70	3	12644	12451.80	2	12644	12453.70	1
2_6	10952	10456.80	2	10668	10400.50	3	10697	10470.10	1
2_7	11055	9884.23	1	11040	9825.37	3	11138	9848.87	2
2_8	10026	9280.47	2	10052	9256.60	3	9962	9289.27	1
2_9	10735	10633.50	3	10885	10693.80	2	10960	10734.10	1
2_10	9848	10026.30	1	9879	9941.60	3	10176	9991.00	2
3_1	11851	10790.40	2	11851	10722.90	3	12045	10798.30	1
3_2	12369	9666.67	1	12369	9665.47	2	12369	9577.77	3
3_3	13355	10531.80	1	13458	10465.20	3	13458	10514.30	2
3_4	10920	9339.53	1	10920	9287.87	2	10831	9264.90	3
3_5	11538	11129.10	1	11538	11030.70	3	11538	11128.20	2
3_6	11235	9106.37	2	11200	9075.23	3	11343	9130.33	1
3_7	10326	10573.00	2	10026	10558.20	3	10133	10582.30	1
3_8	9588	9582.03	1	9565	9494.87	3	9652	9563.00	2
3_9	10728	11439.60	1	10720	11390.90	3	10728	11404.50	2
3_10	9726	11814.80	2	9522	11816.50	1	9597	11744.90	3
Avg			1.7			2.43			1.86

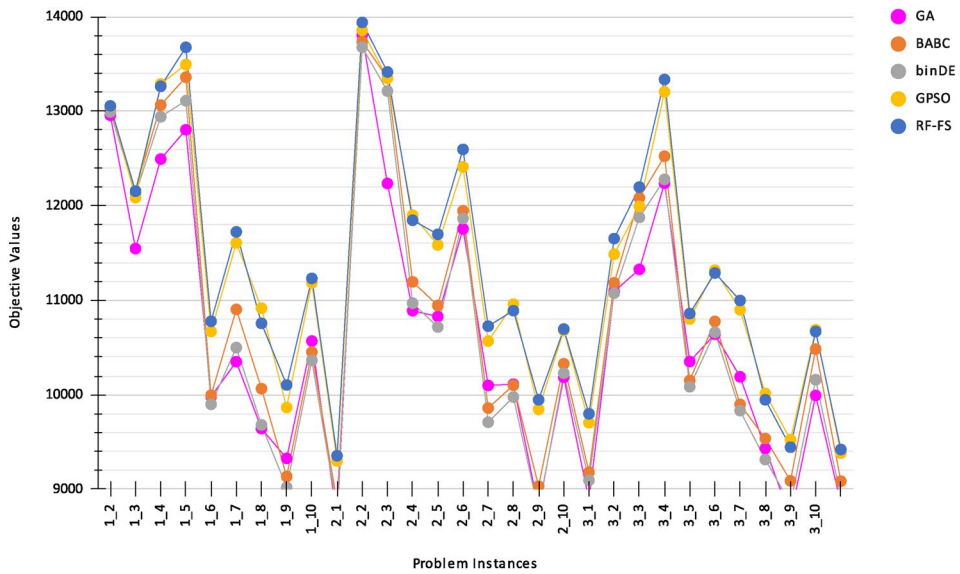

Fig. 10 Comparative results by RF-FS with the state-of-art approaches for SUKP problem instances

Appendix

Algorithm 1 Artificial Bee Colony with ML Model

```
Initialisation of ABC and ML Model
while Termination criteria is not met do
    Employed Bee Phase
    for each bee in colony do
        Select operator (o_i)            ▷ Using Adaptive Operator Selection Scheme
        Produce and evaluate candidate solution
        if Candidate is better than current bee then
            Replace current bee with candidate
        else
            Increment trial number of current bee
        end if
    end for
    Onlooker Bee Phase
    Calculate selection probability of each bee (P)
    for i ← 1 to n do
        Choose current bee according to P
        Select operator (o_i)            ▷ Using Adaptive Operator Selection Scheme
        Produce and evaluate candidate solution
        if Candidate is better than current bee then
            Replace current bee with candidate
        else
            Increment trial number of current bee
        end if
    end for
    Scout Bee Phase
    if trial number of any bee has exceeds the limit then
        Replace the first scout bee with random valid solution
    end if
    Update the best solution
end while
```

Availability of data and materials Data required has been randomly generated as described within the body text of the article. If required further guidance will be provided.

Declarations

Conflict of interest authors declare no conflict of interest.

References

Anescu G, Ulmeanu P (2017) A fast self-adaptive approach to reliability optimization problems. Rev Air Force Acad 2:23–30

Chandrashekar G, Sahin F (2014) A survey on feature selection methods. Comput Electr Eng 40(1):16–28

Durgut R, Aydin ME (2021) Adaptive binary artificial bee colony algorithm. Appl Soft Comput 101:107054

Durgut R, Baydilli YY, Aydin ME (2020) Feature selection with artificial bee colony algorithms for classifying parkinson's diseases. Int Conf Eng Appl Neural Netw. https://doi.org/10.1007/978-3-030-48791-1_26

Durgut R, Aydin ME, Rakib A (2022) Transfer learning for operator selection: A reinforcement learning approach. Algorithms 15(1):24

Durgut R, Aydin ME, Ihshaish H, Rakib A (2022) Analysing the predictivity of features to characterise the search space. In: International Conference on Artificial Neural Networks, pp. 1–13. Springer

Engelbrecht AP, Pampara G (2007) Binary differential evolution strategies. In: 2007 IEEE Congress on Evolutionary Computation, pp. 1942–1947. IEEE

Erwin K, Engelbrecht A (2020) Diversity measures for set-based metaheuristics. In: 2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI), pp. 45–50. IEEE

Fialho Á (2010) Adaptive operator selection for optimization. PhD thesis, Université Paris Sud-Paris XI

Fragata I, Blanckaert A, Louro MAD, Liberles DA, Bank C (2019) Evolution in the light of fitness landscape theory. Trends Ecol Evol 34(1):69–82

Goëffon A, Lardeux F (2011) Optimal one-max strategy with dynamic island models. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, pp. 485–488. IEEE

He Y, Xie H, Wong T-L, Wang X (2018) A novel binary artificial bee colony algorithm for the set-union knapsack problem. Futur Gener Comput Syst 78:77–86

Johnn S-N, Darvariu V-A, Handl J, Kalcsics J (2023) Graph reinforcement learning for operator selection in the ALNS metaheuristic

Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (abc) algorithm and applications. Artif Intell Rev 42(1):21–57

Karimi-Mamaghan M, Mohammadi M, Meyer P, Karimi-Mamaghan AM, Talbi E-G (2022) Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. Eur J Oper Res 296(2):393–422

Lin G, Guan J, Li Z, Feng H (2019) A hybrid binary particle swarm optimization with tabu search for the set-union knapsack problem. Expert Syst Appl 135:201–211

Macias-Escobar TE, Cruz-Reyes L, Dorronsoro B, Fraire-Huacuja H, Rangel-Valdez N, Gómez-Santillán C (2019) Application of population evolvability in a hyper-heuristic for dynamic multi-objective optimization. Technol Econ Dev Econ. https://doi.org/10.3846/tede.2019.10291

Malan KM (2021) A survey of advances in landscape analysis for optimisation. Algorithms 14(2):40

Nyce C (2007) Predictive analytics white paper, sl: American institute for chartered property casualty underwriters. Insurance Institute of America, 1

Ochoa G, Malan K (2019) Recent advances in fitness landscape analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1077–1094

Ozsoydan FB, Baykasoglu A (2019) A swarm intelligence-based algorithm for the set-union knapsack problem. Futur Gener Comput Syst 93:560–569

Pei J, Tong H, Liu J, Mei Y, Yao X (2023) Local optima correlation assisted adaptive operator selection. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '23, pp. 339–347. Association for Computing Machinery, New York, NY, USA

Pitzer E, Affenzeller M (2012) A comprehensive survey on fitness landscape analysis. Recent Adv Intell Eng Syst. https://doi.org/10.1007/978-3-642-23229-9_8

Reijnen R, Zhang Y, Lau HC, Bukhsh Z (2023) Online control of adaptive large neighborhood search using deep reinforcement learning

Sallam KM, Elsayed SM, Sarker RA, Essam DL (2017) Landscape-based adaptive operator selection mechanism for differential evolution. Inf Sci 418:383–404

Sallam KM, Elsayed SM, Sarker RA, Essam DL (2020) Landscape-assisted multi-operator differential evolution for solving constrained optimization problems. Expert Syst Appl 162:113033

Santana CJ Jr, Macedo M, Siqueira H, Gokhale A, Bastos-Filho CJ (2019) A novel binary artificial bee colony algorithm. Futur Gener Comput Syst 98:180–196

Schmitt LM (2001) Theory of genetic algorithms. Theoret Comput Sci 259(1–2):1–61

Sotoudeh-Anvari A, Hafezalkotob A (2018) A bibliography of metaheuristics-review from 2009 to 2015. Int J Knowl-Based Intell Eng Syst 22(1):83–95

Sun G, Yang B, Yang Z, Xu G (2020) An adaptive differential evolution with combined strategy for global numerical optimization. Soft Comput 24(9):6277–6296

Tan Z, Li K, Wang Y (2021) Differential evolution with adaptive mutation strategy based on fitness landscape analysis. Inf Sci 549:142–163

Teng T-H, Handoko SD, Lau HC (2016) Self-organizing neural network for adaptive operator selection in evolutionary search. Int Conf Learn Intell Optim Lect Notes Comput Sci 10079:187–202

Wang H, Wu Z, Rahnamayan S, Sun H, Liu Y, Pan J-s (2014) Multi-strategy ensemble artificial bee colony algorithm. Inf Sci 279:587–603

Wang M, Li B, Zhang G, Yao X (2017) Population evolvability: dynamic fitness landscape analysis for population-based metaheuristic algorithms. IEEE Trans Evol Comput 22(4):550–563

Wang C, Shang P, Shen P (2022) An improved artificial bee colony algorithm based on bayesian estimation. Complex Intell Syst 8(6):4971–4991

Xiang W-l, Li Y-z, He R-c, An M-q (2021) Artificial bee colony algorithm with a pure crossover operation for binary optimization. Comput Ind Eng 152:107011