Reinforcement Learning-Based Adaptive Operator Selection

Rafet Durgut[1][0000−1111−2222−3333] and Mehmet E. Aydin[2][0000−0002−4890−5648]

[1] Karabuk University, Engineering Faculty, Computer Engineering Dept., Turkey
rafetdurgut@karabuk.edu.tr
[2] UWE Bristol, Dept. of Computer Science and Creative Technologies, Bristol, UK
mehmet.aydin@uwe.ac.uk

Abstract. Metaheuristic and swarm intelligence approaches require devising optimisation algorithms with operators to let produce neighbouring solutions to conduct a move. The efficiency of algorithms using single operator remains recessive in comparison with those with multiple operators. However, use of multiple operators require a selection mechanism, which may not be always as productive as expected; therefore an adaptive selection scheme is always needed. In this study, an experience-based, reinforcement learning algorithm has been used to build an adaptive selection scheme implemented to work with a binary artificial bee colony algorithm in which the selection mechanism learns when and subject to which circumstances an operator can help produce better and worse neighbours. The implementations have been tested with commonly used benchmarks of uncapacitated facility location problem. The results demonstrates that the selection scheme developed based on reinforcement learning, which can also be named as smart selection scheme, performs much better that state-of-art adaptive selection schemes.

Keywords: Adaptive Operator Selection · Reinforcement learning · Artificial Bee Colony · Uncapacitated Facility Location Problem (UFLP).

1 Introduction

Metaheuristic and swarm intelligence algorithms have gained a deserved popularity through the success accomplished over last few decades. Although they do not guarantee globally optimal solutions within a reasonable time, the success in offering useful near-optimum solutions within an affordable time has helped gain such credit. This does not mean that metaheuristic and swarm intelligence algorithms can be seamlessly implemented for a productive algorithmic solution. The main shortcoming arises in handling local optima capabilities, which enforces researchers to build a balance in exploration for new and fresh solutions while exploiting the gained success level within the search space. That is known as Exploration versus Exploitation (*EvE*) rate in the literature [5]. *EvE* rate guides to search through as many neighbourhoods as possible while retaining exploitation of achieved success and gained experience for a better performance, where weaker exploration causes falling in local optima while weaker exploitation would cause higher fluctuations in performance [13].

Metaheuristic approaches, especially population-based ones, use neighbourhood functions, also known as operators, to let the search process identify next solutions to move to. It is conceivable that search with single operators would have higher likelihood to stick in a local optima than multiple operators. Many hybridisation approaches and memetic algorithms have been designed to help diversify the search through a balanced *EvE*, which usually appear in the form of using multiple operators subject to a selection scheme. The idea an operator to apply after another would prevent the search falling in local optima contributing to diversification of the search. It appears that the nature of the operators to be applied in an order and the order managed in use play very important role in the success level of the algorithms. Adaptive operator selection schemes have been studied for a while to achieve a useful balance in *EvE* and level of diversification in search [12].

Adaptive operator selection is a process of two phases; (i) *credit assignment* in which the selected operators are credited based on the level of success measured, or (ii) *operator selection* in which an operator is identified to run based on the credit level in order to produce a neighbour [11]. The amount of credit to assign is decided using either the positive difference achieved in fitness values or the categories of success or fail [10]. Credit assignment phase also covers the calculation of the time window in which the amount of credit to assign to selected operators is estimated [4]. On the other hand, operator selection phase imposes prioritisation/rank of operators within a pool of functions/operators. *Probability Matching* (PM), *Adaptive Pursuit* (AP) and *Upper Confidence Bound* (UCB) are known to be among state-of-art operators selection schemes [4].

Adaptive operator selection schemes have been used in the literature with evolutionary algorithms and swarm intelligence. Failho et. al [9] uses a multi-armed bandits approach with genetic algorithms, while Durgut and Aydin [7] comparatively studied the success of PM, AP, and UCB schemes to supply a binary artificial bee colony algorithm. Yue et. al. [19] proposes a self-adaptive particle swarm optimisation algorithm adaptively selecting among 5 operators to solve large scale feature selection problems.

Adaptive operator selection schemes estimate likelihood of each operator within the pool relying on credits gained to the time. The selection happens through the estimated likelihoods irrespective of the problem state in hand. It is clear that the success of selected operator is not sensitive to the problem state; whether it is in a harsh neighbourhood or trapped in a difficult local optima or not. Reinforcement learning (RL) gains more and more popularity day-by-day to solve dynamic problems progressively, gaining experiences through problems solving process [3, 17]. There are renown powerful RL algorithms let map input sets to outputs through experiencing the the problems states and collecting environmental responses to the actions taken [20].

In this study, an artificial bee colony (ABC) algorithm has been implemented for solving uncapacitated facility location problems (UFLP) represented in binary form. ABC algorithms have been implemented to solve many real-world engineering problems. Among them are combinatorial optimisation problems, which formulated as binary optimisation problems. ABC can be viewed as multi-start hill-climbing algorithms in optimisation, where new neighbouring solutions are generated with operators as discussed above. In this study, the ABC algorithm is furnished with multiple operators selected with reinforcement learning-based selection scheme.

The rest of this paper is organised as follows; Adaptive operator selection schemes are introduced in Section 2, the operator selection scheme developed based on reinforcement algorithm is explained in Section 3. Experimentation and results are presented and discussed in Section 4 while conclusions are briefed in Section 5.

2 Adaptive Operator Selection

One of the common problem of heuristic-based optimisation algorithms is that search is inevitably driven into local optima, which sometimes remains as the offered final solution. The aim of use multiple operator is to help rescue the search from local optima by the means of diversifying search using different neighbourhood functions/operators interchangeably or systematically. Operator selection schemes are used for this purpose.

Operator selection is not necessarily to be adaptive by nature, but, most of recent studies have been developed as adaptive to insert smartness in the process of selection. Metaheuristic and evolutionary approaches can come up with self-imposing operator selection. Evolutionary algorithms such as genetic algorithms and genetic programming have self-contained probabilistic operator selection while metaheuristics such as variable neighbourhood search imposes a systematic count-based operator change mechanism to achieve diversity in search and manage neighbourhood change. Operator selection built-in algorithms do not offer much flexibility in working with multiple operators, while memetic algorithms, hill-climbing style heuristic algorithms and modern swarm intelligence algorithms allow customising operator selection mechanism to engineer bespoke efficient optimisation algorithms.

Adaptive operator selection is the process of prioritisation of the operators based on merits, which can be imported in the algorithms via crediting each operator based on achievements gained. Although there are a number of adaptive operator selection schemes studied, the general mechanism is depicted in Fig. 1 in which a two phase process is run; (i) *operator selection* and (ii) *credit assignment*. As suggested, the pool of operators holds a finite number of operators to select an operator from in order to produce neighbours to move to, while the selected operators is credited upon its action and success level it achieves in producing new solutions. The credit level to assign to the selected operator is estimated based on preferred rules.

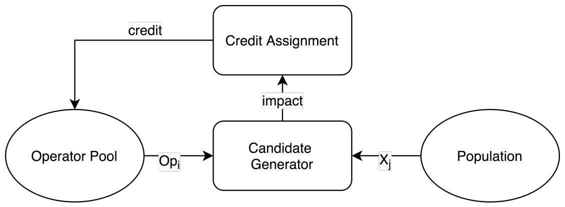

Fig. 1. General overview of adaptive operator selection process with support of population and pool of operators

2.1 Operator Selection

The first phase of operator selection process is to execute the selection rule imposed by operator selection scheme in order to produce neighbouring solutions to move to. The main aim is to keep a *EvE* rate as balanced as possible so that the search to be intensified within the neighbourhood as long as it produces positively and to be diversified as soon as it turns to negative productivity. Literature reports a number of operator selection schemes; random selection, merit-based selection, probability matching, adaptive pursuit and multi-arm bandit approaches, e.g. upper confidence bound (UCB). Random selection chooses an operator from the pool completely randomly, Roulette-wheel takes the success counts of each operator into account to calculate a probability-based prioritisation, while probability matching (PM) approach accounts the success as merits and lets to increase the selectability of non-chosen operators using the following rule:

$$p_{i,t} = p_{min} + (1 - Kp_{min})\frac{q_{i,t}}{\sum_{j=1}^{K} q_{j,t}}, \quad i = 1, 2..K \tag{1}$$

where K is the number of operators in the pool, $p_{min} \in [0,1]$ represents the minimum probability of being selected, and $q_{i,t}$ is the credit level/value of operation i at time t. Both PM and AP use p_{min} to set a base probability for each operator, which would help address the *EvE* dilemma with allocating a minimum chance to every operators to be selected. PM imposes to calculate the probabilities of being selected per operation, while AP uses the strategy of "*winner takes all*" approach that credits more to promising options. adaptive pursuit (AP) calculates the probabilities with Eq. 2.

$$p_{i,t} = \begin{cases} p_{i,t-1} + \beta(p_{max} - p_{i,t-1}), & \text{if } i = i_t* \\ p_{i,t-1} + \beta(p_{min} - p_{i,t-1}), & \text{otherwise} \end{cases} \tag{2}$$

Both of PM and AP impose higher dominance for exploitation, which is aimed to decrease by UCB using the following rule, which selects the operator with highest probability.

$$p_{i,t} = \begin{cases} 1 - p_{min} * (K - 1) & \text{if } i = i_t* \\ p_{min}, & \text{otherwise} \end{cases} \tag{3}$$

where K is the number of operators in the pool, $p_{min} \in [0,1]$ represents the minimum (base) probability for being selected, i_t* is calculated with 4.

$$i_t* = \arg\max_{i=1,..,K} \{q_{i,t} + C \times \sqrt{\frac{2\log\sum_{j=1}^{K} n_{j,t}}{n_{i,t}}}\} \tag{4}$$

where op_t represents the selected operator, C works as a scaling factor, n is number of times the operator selected while $q_{i,t}$ and $n_{i,t}$ on the right-hand-side of equation help control *EvE* dilemma, respectively.

2.2 Credit Assignment

The next phase of adaptive operator selection process is to estimate a credit to be assigned to the operator just used. This involves how to estimate the amount of reward to assign and what to be the base for estimate of a credit. Literature suggests that mainly two classes of approaches have been implemented; whether a success has been achieved or not, or how much positive difference accomplished. The former approach considers if the result is "success" or "fail", while the latter processes the amount of achievement in quantity to estimate the level of reward to assign.

The process of credit assignment entails clarifying the time window with which the reward level is to be estimated. The time window can span from last single step to a pre-defined number of previous steps in which the credit level and/or the achievement level can be averaged. This reveals that a credit can be decided as *instant* credit, an *averaged* credit or the *maximum* credit.

3 Proposed Approach: Adaptive selection with Reinforced-Clusters

Operator selection adaptively developed and used for higher efficiency in diversification of search process. The operator selection schemes, even the adaptive ones, propose choosing an operator based on credits gained over the success counts through out the search, but, regardless of the input sets, the problem state, and search circumstances. The merit-based schemes usually select operators through a blind process, where the total gained credit is relied on regardless of the status of search etc. It is known that operators do not always produce success due to their limitations; each performs better under some circumstances, while does worse in other circumstances. Once the fruitful circumstances are ascertained for each operator, a complementary policy can be customised for deliberative selection to achieve success.

This study aims to propose a more conscious selection process developed based on reinforcement learning approach implemented into a distance-based clustering algorithm in which the distance in between the input set and the fine-tuned cluster centres is estimated and made reference index in operator selection. The idea of setting up a selection scheme based on clusters is discussed and implemented in machine learning studies. Reinforcement learning is known to be very useful in handling dynamically changing environment and for solving dynamic problems, particularly for operating within unknown dynamic environments. One of earlier studies proposes embedding reinforcement learning in a distance-based clustering algorithm, namely hard-c-means algorithm, to train agents to select the best scheduling operator subject to dynamic production environments to solve dynamic scheduling problems [2]. Inspiring of this study, a reinforced-clustering algorithm is put together to optimise the cluster centres so that the problem states can be classified with optimised clusters, where each cluster will correspond to an operator. The algorithm will impose selecting the cluster centre, operator, closer to the input set in distance. This will facilitates a selection scheme conscious with problem state.

Operators are selected based on probabilities, $p_{i,t}$, calculated as in Eq. 3, where the best operator is determined using Eq. 5. The other operators are also prioritised based on the distance in between the problem state at time t, $\mathbf{x_t}$, and the cluster centres, $\mathbf{c_t}$ - corresponding to the operators. Here, the distance metric used in this study is *hamming* distance due to the binary representation of the problem and the operators.

$$i_t* = \arg \min_{i=1,..,K} \{\beta q_{i,t} + \gamma e_i(x_t)\} \tag{5}$$

where $q_{i,t}$ is the credit level/value of operation i at time t, while $e_i(x_t) = \|\mathbf{x_t} - c_i\|$, the estimated distance between an input set and cluster c_i, β and γ are coefficients to balance between credit and distance metrics. Note that unlike other methods, the reward value of good solutions is reflected as negative.

4 Experimental Results

The reinforced-clustering-based operator selection scheme has been tested with a binary ABC algorithm to solve uncapacitated facility location problem (UFLP) instances, which is one of well-known NP-Hard combinatorial problem. The details of UFLP benchmarking instances taken form OR-Library can be found in many articles [1, 8].

The problem solving algorithm to use reinforced-clustering-based operator selection scheme is chosen as the standard artificial bee colony (ABC) algorithm reported in [14]. The standard ABC is designed for continuous numerical optimisation problems, while UFLP is a combinatorial optimisation problem represented in binary form [18]. The algorithm has been rearranged to work with state-of-art binary operators; *binABC* [16] and *ibinABC* [6] work on the basis of XOR logical operator and *disABC* [15] uses a hamming distance-based binary logic.

Algorithm 1 presents a pseudo code of ABC algorithm embedded with reinforced-cluster-based operator selection scheme implemented for UFL problems. As seen, ABC imposes a three-phase process to evolve a swarm (population) of solutions. The first phase exploits *employed* bees to generate new solutions with selected

Algorithm 1 The pseudo code of binary ABC embedded with reinforced-cluster based operator selection scheme

```
1: Initialisation phase:
2: Set algorithm parameters
3: Create initial population
4: while Termination criteria is not met do
5:     Employed bee phase:
6:     Select operators and assign to bees
7:     for i=1 to N do
8:         Select neighbour, apply operator and obtain candidate solution ($v_i$)
9:         if $f(v_i)$ is better than $f(x_i)$ then
10:            Replace $v_i$ with $x_i$
11:            Get reward and add to $r_{op,t}$ and update centroid of $c_{op,t}$
12:            Reset trial counter
13:        else
14:            Increment trial counter
15:        end if
16:    end for
17:    Onlooker bee phase:
18:    Calculate probabilities for food sources
19:    Select operators and assign to bees
20:    Increment operator counter, t=0
21:    for i=1 to N do
22:        Determine current solution according to probability
23:        Select neighbour food source
24:        Apply operator and obtain candidate solution ($v_c$)
25:        if $f(v_c)$ is better than $f(x_c)$ then
26:            Replace $v_c$ with $x_c$
27:            Get reward and add to $r_{op,t}$ and update centroid of $c_{op,t}$
28:            Reset trial counter
29:        else
30:            Increment trial counter
31:        end if
32:    end for
33:    Update Phase:
34:    Credit assignment
35:    Memorisation
36:    Scout bee phase:
37:    if Limit is exceed for any bee then
38:        Create random solution for the first exceeding bee and evaluate it
39:    end if
40: end while
```

binary operators applying to the materials taken from a selected solutions and one of its neighbours. The generated solution is added to the swarm if it is better than the parents, the amount of reward to allocate to the operators is estimated and the position of centre for selected and used operator is updated. If the the generated new solution is not better than the parent solution no reward is generated and the trail counter is incremented.

The *onlooker* bees conduct the second phase of ABC in which the solutions are selected with a probabilistic approach to let randomness contribute the diversity of the swarm. Similar to the first phase, the operator selection, the reward estimation and crediting are performed and the corresponding cluster centres are updated. The *scout* bees follow up the *onlookers* to replace from non-improvable solutions with randomly generated ones to keep the swarm further divers.

The experimentation has started with parametric study to fine-tune parameters used in both the algorithm and within the mechanics of the operator selection scheme. The experimentation for parametric study has been conducted using the hardest benchmarking instance of UFL problem, which is known as CapC. The parameters configured for best fit are tabulated in Table 4 and averaged over 30 repetitions.

Table 1. Parameter configurations tested

Parameter	Values		
Reward	Inst	Avrg	Max
P_{min}	0.10	0.20	0.30
W	10.00	25.00	50.00
β	0.01	0.05	0.10
γ	0.10	0.50	0.90

Table 2 presents the *hit* metric, which is the number of trails attained the optimum. The best performance so far is 25 hits out of 30 trails, where $\gamma = 0.5$, $\beta = 0.01$ and $P_{min} = 0.1$ are found and setup. Next, the reward estimation across a time/iteration window is fine-tuned, where the parametric study results obtained for *average* and *extreme* rewards are tabulated in Table 3. The best hit values are obtained *25* and *27* out of *30* trails for *average* and *extreme* reward cases. respectively.

Table 2. Parameter tuning for Instant reward measured with *hit* metric

γ	P_{min}								
	0.1			0.2			0.3		
	β			β			β		
	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1
0.1	24	16	20	18	24	24	24	24	24
0.5	25	21	19	19	19	21	24	19	14
0.9	16	21	14	21	21	14	17	21	17

The window size (W) of *25* and *50* produce best results, while all trails are tested with $P_{min} = 0.1$, $\beta = 0.05$ and $\gamma = 0.1$. The averaged achievements conclude that $W = 25$ produces the best configuration.

The best configuration concluded out of parametric study has been run with hardest benchmark instances, CapC, to trace the operator selection through timeline, where the progress of operation selection is plotted in Fig. 2. The plot demonstrates that *disABC* operates best over the first 200 iterations and then *ibinABC* takes over the best delivery. *binABC* doesn't perform well in comparison to other two as suggested in the plot.

The results by the proposed approach have been tabulated in Table 4 alongside of other adaptive operator selection methods explained above for comparative purposes. As seen, all adaptive methods embedded in binary ABC algorithm have assisted solve all UFLP benchmark instances with 100% success except CapC, where the *Gap* and *St. Dev* metrics are *0* and the *hit* measure is 30 out of 30 for all instances except CapC. It is paramount to define the *gap* as the average difference in between the optimum value and the fitness/cost value found,

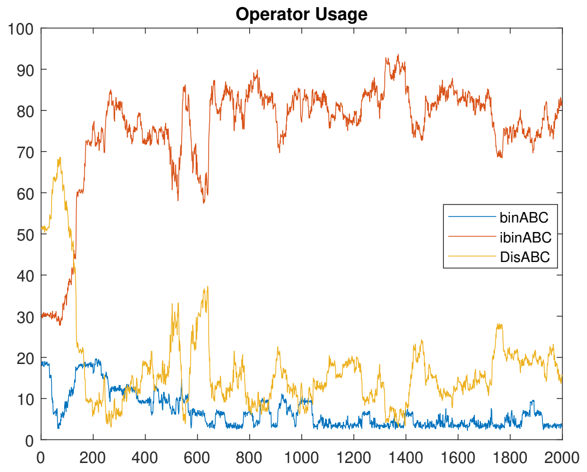

Fig. 2. Operator usage rates through search process

Table 3. Parametric fine-tuning results in *hit* metric for both *average* and *extreme* rewards

			Average Reward			Extreme Reward						Average Reward			Extreme Reward		
			γ			γ						γ			γ		
W	P_{min}	β	0.1	0.3	0.9	0.1	0.3	0.9	W	P_{min}	β	0.1	0.3	0.9	0.1	0.3	0.9
		0.01	16	21	16	24	25	24			0.01	22	20	22	22	23	24
	0.1	0.05	23	19	17	23	24	23		0.1	0.05	25	17	19	27	21	23
		0.1	21	19	18	23	24	19			0.1	24	21	18	22	19	21
		0.01	24	21	23	19	19	23			0.01	23	21	21	17	25	23
5	0.2	0.05	22	20	18	21	19	17	25	0.2	0.05	15	19	21	21	22	22
		0.1	20	21	19	19	18	19			0.1	22	20	21	17	25	23
		0.01	20	21	23	19	18	20			0.01	21	20	20	20	16	26
	0.3	0.05	20	23	20	21	20	19		0.3	0.05	20	18	18	22	23	17
		0.1	22	15	18	21	16	19			0.1	21	18	19	25	22	16
		0.01	25	21	24	22	16	25			0.01	23	19	19	21	18	19
	0.1	0.05	21	23	20	20	18	22		0.1	0.05	21	22	19	27	21	19
		0.1	24	15	23	22	19	22			0.1	20	20	18	21	18	23
		0.01	25	19	20	21	18	13			0.01	18	20	25	21	22	19
10	0.2	0.05	21	22	20	14	23	17	50	0.2	0.05	19	21	19	21	18	18
		0.1	24	21	20	15	21	21			0.1	23	14	22	22	17	19
		0.01	24	20	20	24	20	19			0.01	16	25	20	21	18	20
	0.3	0.05	21	14	16	20	24	19		0.3	0.05	22	17	21	22	16	20
		0.1	23	21	19	20	20	22			0.1	16	19	16	21	14	18

while *St. Dev.* is the standard deviation calculated over 30 repeated trails. CapC seems to be the hardest benchmark instance, which helps fine-tuning the hyper parameters and comparing the results produced by each rival approaches. The proposed method, labelled as "C-BABC" in the tables, produces the lowest *gap* and *st. dev* and the highest *hit* in comparisons to "PM-BABC", "AP-BABC" and "UCB-BABC", which are the binary ABC algorithms embedded with *PM*, *AP* and *UCB* as explained above.

Table 4. The comparative results obtained; the proposed operator selection scheme vs alternatives

Benchmarks	PM-ABC			AP-BABC			UCB-BABC			C-BABC		
	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit
Cap71	0	0	30	0	0	30	0	0	30	0	0	30
Cap72	0	0	30	0	0	30	0	0	30	0	0	30
Cap73	0	0	30	0	0	30	0	0	30	0	0	30
Cap74	0	0	30	0	0	30	0	0	30	0	0	30
Cap101	0	0	30	0	0	30	0	0	30	0	0	30
Cap102	0	0	30	0	0	30	0	0	30	0	0.	30
Cap103	0	0	30	0	0	30	0	0	30	0	0	30
Cap104	0	0	30	0	0	30	0	0	30	0	0	30
Cap131	0	0	30	0	0	30	0	0	30	0	0	30
Cap132	0	0	30	0	0	30	0	0	30	0	0	30
Cap133	0	0	30	0	0	30	0	0	30	0	0	30
Cap134	0	0	30	0	0	30	0	0	30	0	0	30
CapA	0	0	30	0	0	30	0	0	30	0	0	30
CapB	0	0	30	0	0	30	0	0	30	0	0	30
CapC	0.0055	1428.003	25	0.0043	1302.539	26	0.0087	1694.457	22	0.0033	1149.5	27

The success of proposed method has been comparatively tested with a number of recently published studies, which can be considered as state-of-art works. The comparative results have been picked up form corresponding articles [1] and tabulated with the results produced by the proposed approach. As clearly seen on Table 5, the proposed method, C-BABC, outperforms all the algorithms known to be the state-of-the-art with a 100% success of solving all benchmark instances except CapC, which is solved with the highest score, while binAAA and JayaX solve all instances except CapB and CapC. Due to level of hardness in solving CapB and CapC approaches are tested with, so is the proposed approach in comparative way. The difference between the results by the proposed approach and other competitor algorithms have been tested statistically with Wilcoxon signed rank and the results are presented in Table 6, where C-BABC, the proposed method is significantly performed better.

Table 5. Comparative results; The proposed method (C-BABC) versus some state-of-art approaches

Benchmark	GA-SP			BPSO			binAAA			JayaX			C-BABC		
	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit	Gap	Std. Dev.	Hit
Cap71	0	0	30	0	0	30	0	0	30	0	0	30	0	0	30
Cap72	0	0	30	0	0	30	0	0	30	0	0	30	0	0	30
Cap73	0.066	899.65	19	0.024	634.625	26	0	0	30	0	0	30	0	0	30
Cap74	0	0	30	0.0088	500.272	29	0	0	30	0	0	30	0	0	30
Cap101	0.068	421.655	11	0.0432	428.658	18	0	0	30	0	0	30	0	0	30
Cap102	0	0	30	0.00989	321.588	28	0	0	30	0	0	30	0	0	30
Cap103	0.063	505.036	6	0.04939	521.237	14	0	0	30	0	0	30	0	0	30
Cap104	0	0	30	0.040	1432.239	28	0	0	30	0	0	30	0	0	30
Cap131	0.068	720.877	16	0.171	1505.749	10	0	0	30	0	0	30	0	0	30
Cap132	0	0	30	0.058	1055.238	21	0	0	30	0	0	30	0	0	30
Cap133	0.091	685.076	10	0.082	690.192	10	0	0	30	0	0	30	0	0	30
Cap134	0	0	30	0.195	2594.211	18	0	0	30	0	0	30	0	0	30
CapA	0.046	22451.21	24	1.69	319855.4	8	0	0	30	0	0	30	0	0	30
CapB	0.58	66658.65	9	1.40	135326.7	5	0.24	39224.74	15	0.07	27033.02	26	0	0	30
CapC	0.70	51848.28	2	1.62	115156.4	1	0.29	29766.31	1	0.021	5455.94	17	0.0033	1149.5	27

Table 6. Statistical test results for state-of-art methods compared with proposed approach

	binAAA		JayaX		BPSO		GA-SP	
Benchmarks	p-value	H	p-value	H	p-value	H	p-value	H
Cap71	1	0	1	0	1	0	1	0
Cap72	1	0	1	0	1	0	1	0
Cap73	1	0	1	0	1.E-01	0	1.E-03	1
Cap74	1	0	1	0	3.E-06	1	4.E-08	1
Cap101	1	0	1	0	2.E-01	0	4.E-04	1
Cap102	1	0	1	0	5.E-01	0	1	0
Cap103	1	0	1	0	1.E-06	1	1.E-06	1
Cap104	1	0	1	0	5.E-01	0	1	0
Cap131	1	0	1	0	1.E-06	1	1.E-06	1
Cap132	1	0	1	0	1.E+00	0	4.E-08	1
Cap133	1	0	1	0	2.E-06	1	1.E-06	1
Cap134	1	0	1	0	5.E-04	1	1	0
CapA	1	0	1	0	5.E-05	1	1.E-01	0
CapB	6.E-05	1	2.E-07	1	2.E-06	1	2.E-06	1
CapC	4.E-06	1	1.E-04	1	3.E-06	1	4.E-06	1

5 Conclusion

This study has been done to investigate how machine learning can help adapt a dynamically updating scheme for operator selection within ABC algorithms as one of recently developed swarm intelligence approaches in solving binary problems. The research has been done embedding an online learning mechanism into binary ABC to learn which operator performs better in given circumstances. The main contribution of this research is that the adaptive operator selection has been achieved through reinforcement learning which is implemented with Hard-C-means clustering algorithm converted its unsupervised nature into reinforcement learning. Unlike the previously suggested adaptive selection schemes, this approach maps the binary input set into corresponding operators, hence, each time the hamming distance between both binary sets is used to make the selection, while the centres of the clusters are optimised/fine-tuned with estimated rewards per operator selection. The optimised cluster centres remain as the basis of operator selection. The proposed algorithm is tested with solving UFL problems, and statistically verified that the proposed approach significantly outperforms the state-of-art approaches in solving the same benchmark instances. It is also demonstrated that other existing adaptive approaches are also outperformed.

References

1. Aslan, M., Gunduz, M., Kiran, M.S.: Jayax: Jaya algorithm with xor operator for binary optimization. Applied Soft Computing **82**, 105576 (2019)
2. Aydin, M.E., Öztemel, E.: Dynamic job-shop scheduling using reinforcement learning agents. Robotics and Autonomous Systems **33**(2-3), 169–178 (2000)

3. Coronato, A., Naeem, M., De Pietro, G., Paragliola, G.: Reinforcement learning for intelligent healthcare applications: A survey. Artificial Intelligence in Medicine **109**, 101964 (2020)
4. DaCosta, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. pp. 913–920 (2008)
5. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040 (2019)
6. Durgut, R.: Improved binary artificial bee colony algorithm. Frontiers of Information Technology & Electronic Engineering **(in press)** (2020)
7. Durgut, R., Aydin, M.E.: Adaptive binary artificial bee colony algorithm. Applied Soft Computing **101**, 107054 (2021)
8. Durgut, R., Aydin, M.E.: Adaptive binary artificial bee colony algorithm. EasyChair Preprint no. 4687 (EasyChair, 2020)
9. Fialho, Á.: Adaptive operator selection for optimization. Ph.D. thesis, Université Paris Sud-Paris XI (2010)
10. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme value based adaptive operator selection. In: International Conference on Parallel Problem Solving from Nature. pp. 175–184. Springer (2008)
11. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Annals of Mathematics and Artificial Intelligence **60**(1-2), 25–64 (2010)
12. Hussain, A., Muhammad, Y.S.: Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. Complex & Intelligent Systems pp. 1–14 (2019)
13. Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y.: On the exploration and exploitation in popular swarm-based metaheuristic algorithms. Neural Computing and Applications **31**(11), 7665–7683 (2019)
14. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (abc) algorithm. Applied soft computing **8**(1), 687–697 (2008)
15. Kashan, M.H., Nahavandi, N., Kashan, A.H.: Disabc: A new artificial bee colony algorithm for binary optimization. Applied Soft Computing **12**(1), 342–352 (2012)
16. Kiran, M.S., Gündüz, M.: Xor-based artificial bee colony algorithm for binary optimization. Turkish Journal of Electrical Engineering & Computer Sciences **21**(Sup. 2), 2307–2328 (2013)
17. Moerland, T.M., Broekens, J., Jonker, C.M.: Model-based reinforcement learning: A survey. arXiv preprint arXiv:2006.16712 (2020)
18. Ozturk, C., Hancer, E., Karaboga, D.: Dynamic clustering with improved binary artificial bee colony algorithm. Applied Soft Computing **28**, 69–80 (2015)
19. Xue, Y., Xue, B., Zhang, M.: Self-adaptive particle swarm optimization for large-scale feature selection in classification. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(5), 1–27 (2019)
20. Yang, T., Zhao, L., Li, W., Zomaya, A.Y.: Reinforcement learning in sustainable energy and electric systems: A survey. Annual Reviews in Control (2020)