A modified single and multi-objective bacteria foraging optimisation for the solution of quadratic assignment problem

Saeid Parvandeh*

Department of Computer Science,
University of Tulsa,
Tulsa, OK, USA
Email: saeid-parvandeh@utulsa.edu
and
Department of Molecular and Human Genetics,
Baylor College of Medicine,
Houston, TX, USA
Email: saeid.parvandeh@bcm.edu
*Corresponding author

Mohammadreza Boroomand

Department of Management,
University of Akdeniz,
Antalya, Turkey
Email: m_boroomand70@gmail.com

Fahimeh Boroumand and Pariya Soltani

Department of Computer Engineering,
Islamic Azad University of Boroujerd,
Boroujerd, Lorestan, Iran
Email: f.boroomand92@gmail.com
Email: p_soltani2011@yahoo.com

Abstract: Non-polynomial hard (NP-hard) problems are challenging due to time-constraint. The bacteria foraging optimisation (BFO) algorithm is a metaheuristics algorithm that is used for NP-hard problems. BFO is inspired by the behaviour of the bacteria foraging such as E. coli. The aim of BFO is to eliminate weak foraging properties bacteria and maintain breakthrough foraging properties bacteria toward the optimum. However, reaching to optimal solutions are time-demanding. In this paper, we modified single objective and multi-objective BFO (MOBFO) by adding mutation and crossover from genetic algorithm operators to update the solutions in each generation, and local tabu search algorithm to reach the local optimum solution. Additionally, we used fast non-dominated sort algorithm in MOBFO to find the best non-dominated solutions. We evaluated the performance of the proposed algorithms with quadratic assignment problem instances. The experimental results show that our approaches outperform some previous optimisation algorithms in both convergent and divergent aspects.

Keywords: bacteria foraging optimisation; multi-objective optimisation; NSGA-II; local tabu search; quadratic assignment problem; QAP; non-dominated sort algorithm.

Reference to this paper should be made as follows: Parvandeh, S., Boroomand, M., Boroumand, F. and Soltani, P. (2021) 'A modified single and multi-objective bacteria foraging optimisation for the solution of quadratic assignment problem', *Int. J. Bio-Inspired Computation*, Vol. 17, No. 1, pp.1–13.

Biographical notes: Saeid Parvandeh is Postdoctoral Research Fellow in Department of Molecular and Human Genetics, Baylor College of Medicine, USA. He received his PhD in Department of Computer Science from University of Tulsa, USA. His research interests are machine learning, bioinformatics, natural language processing, and evolutionary optimisation.

Mohammadreza Boroomand received his BSc in industrial management from university of Isfahan in Iran.He received his MBA from Akdeniz University in Turkey under the aegis of the competitive and reputable international Turkey scholarship award funded by the government of Turkey in 2019. His research interests include consumer behaviour, digital marketing, strategic

marketing, brand management, tourism, operation research, supply chain management, and optimisation.

Fahimeh Boroumand is a Lecturer in several educational institutions and high schools. She received her BSc in Computer Engineering in the field of software from Isfahan University of Applied Science and Technology in Iran. She received her MSc in Computer Engineering in the field of software from Islamic Azad university of Borujerd in 2015. Her interest field centred around computer software applications.

Pariya Soltani received her BSc in Computer Engineering in the field of software from Islamic Azad University of Mahshar in Iran. She received her MSc in Computer Engineering in the field of software from Islamic Azad university of Borujerd in 2016. Her interest fields include computer software applications.

1 Introduction

The goal of optimisation process is to find the optimal solution with minimum cost for non-deterministic polynomial hard (NP-hard) problems (Alothaimeen and Arditi, 2019; Gunantara, 2018). Basically, there are two types of NP-hard problems, single objective and multi-objective optimisation (Deb, 2004). Evolutionary algorithm (EA) is considered as an effective method to find optimal solutions for such problems (Ojha et al., 2019; Emmerich and Deutz, 2018). And, there are many state-of-the-art evolutionary computation algorithms that have been used to solve the combinatorial optimisation problems such as
non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002), Strength Pareto Evolutionary Algorithm Version 2 (SPEA2) (Zitzler and Thiele, 1999), multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li, 2007), and Pareto-Archived Evolution Strategy (PAES) (Knowles and Corne, 2000). Recently, Li and Wang (2017) improved cuckoo search algorithm by adding combination of the learning-evolve thought with Gaussian distribution to improve the convergence velocity and optimisation accuracy in unconstrained function optimisation problems. Wang and Zhang (2016) improved artificial bee colony algorithm by updating a new search equation for onlooker bees, where optimal and suboptimal solutions were considered in iteration process, and an opposition-based method applied to initial and final solution to enhance the global convergence. Wang and Song (2017) combined chaotic mapping strategy and the biogeography-based optimisation optimal migration model and applied on function optimisation problems. Luo et al. (2015) proposed a discrete bacterial foraging optimisation (DBFO) based on the idea of bacteria foraging optimisation (BFO) algorithm. They applied binary encoding to solve both discrete and continuous optimisation problems and rotation step to update velocities of bacteria. Then, they evaluated the efficiency of algorithm through three classical benchmark functions and spectrum allocation problem of cognitive radio. Cui et al. (2019, 2020) applied a modified particle swarm optimisation (PSO) and pigeon-inspired optimisation (PIO) to solve many-objective optimisation problems. Yi et al. (2016) applied multi-objective BFO (MOBFO) on aluminium electrolysis production process to find the optimal solutions. They used PAEA approach and adaptive foraging strategy (AFS) to balance convergence and diversity of Pareto front. Zhao et al. (2016) proposed a novel BFO to solve the permutation flow-shop scheduling problem (PFSP), that is a typical combinatorial NP-hard problem with discrete solution space. They showed that convergent speeds and entrapment in the local optimum that were incurred by original BFO can be handled by combining a differential evolution operator and a chaotic search operator in chemotactic part of BFO algorithm. Darvishi et al. (2014) used MOBFO to solve the optimal power flow (OPF), that is a multi-objective NP-hard problem. They proposed a new BFO algorithm by embedding the fuzzy strategy to original BFO to eliminate the problem of choosing the penalty factors for constraints and behave them just like objective function. Most of the proposed methods modified either chemotactic part or elimination-dispersal part of BFO algorithm for the single or multi-objective problems.

Quadratic assignment problem (QAP) (Koopmans and Beckmann, 1957) and multi-objective QAP (mQAP) (Knowles and Corne, 2003) are the NP-hard problems so that the solutions for them cannot be found in polynomial time. Thus, there should be an optimisation algorithm to solve such NP-hard problems. Recently, many EAs have been used for the solution of QAP and mQAP. Tosun (2014) used genetic algorithm (GA) (Tosun et al., 2013) to solve single objective QAP. He proposed a new recombination of operator based on order-1 crossover algorithm and used quick sort algorithm to generate different chromosomes partitions. (Benlic and Hao, 2015) proposed a population-based memetic algorithm (BMA) for the solution of single objective QAP. They used an effective local optimisation algorithm breakout local search (BLS) (Benlic and Hao, 2013) within the evolutionary computing framework for improvement of the results. Zinflou et al. (2013) used genetic immune strategy (GISMOO) for the solution of mQAP. GISMOO is hybrid between GA and artificial immune system (AIS) to find the solutions in mQAP. The updating part of this algorithm is classical GA operators and AIS cloning selection principle. Michalak (2016) proposed a new local search method that utilises the knowledge concerning promising search directions and can be used as a general framework and combined with an EA for the solution of mQAP and travelling salesman problem (TSP). Shukla (2015) used bat algorithm for the solution of QAP. They suggested that Bat algorithm cannot be directly used for solving discrete search space problem like QAP, thus they used smallest position value (SPV) a heuristic rule to enable the bat algorithm for the solution of QAP.

In this paper, we modified BFO for single objective problems and MOBFO for multi-objective problems. For single objective BFO, we added swap mutation into chemotactic part to avoid the repeated permutations and local Tabu search algorithm (Tabitha et al., 2009) to the end of the algorithm in order to optimise the cost of final permutation and avoid the local optimum. For MOBFO, we added swap mutation and uniform-like crossover (ULX) operators (Tosun, 2014; Tosun et al., 2013) in chemotactic part and local tabu search algorithm to the end of the algorithm. Additionally, we used fast non-dominated sort method (Deb et al., 2002) into elimination and dispersal part of the MOBFO algorithm to optimise obtained non-dominated set solutions in each iteration. We applied modified BFO and MOBFO on QAP and mQAP problems, respectively. Our contributions for this paper are: first, we modified BFO and MOBFO algorithms for the purpose of single and multi-objective problems. Second, regardless of the problem size, these algorithms can find compatible solutions for both QAP and mQAP. Third, however the optimum solutions may not be found eventually, but the experimental results show that the average percentage gap of the solutions for QAP are lower than state-of-the-arts algorithm and the non-dominated solutions for mQAP are very close to Pareto front.

2 Bacteria foraging optimisation

BFO algorithm is one of the bio-inspired optimisation algorithms that is inspired from the biomimicry of the *E. coli* bacteria. BFO introduced by Passino (2002, 2010) and its concept is to eliminate the bacteria that have the weak foraging properties and maintain the bacteria that have breakthrough foraging properties toward the nutrient collection or maximise the energy per unit time. Each bacterium communicates with other bacteria by sending the signals, and bacteria move to the next step for collecting nutrient if previous factors have been satisfied. Initially, BFO introduced to make a bridge between microbiology and engineering, and mimics some properties of the bacteria foraging such as chemotactic, reproduction, and quorum sensing, but later in 2010, he investigated and compared BFO with other optimisation algorithms (Passino, 2010). Basically, BFO consists of four principal parts: chemotactic, swarming, reproduction, elimination and dispersal (Das et al., 2009), and a brief introduction of each part is given as follows:

2.1 *Chemotactic*

In biology, the chemotactic is a process of bacteria movement for gaining the nutrient. One of the *E. coli* bacterium properties is that, it can move in two diverse ways, swimming and tumbling. In swimming, the bacterium swims in the same direction to search for nutrient, and in tumbling, bacterium changes the direction to another direction. Assume $\theta^i(j, k, l)$ shows the current position of ith bacterium, jth chemotactic step, kth reproduction step, and lth elimination and dispersal step, then the position of bacterium in the next chemotactic step by tumbling will be as:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{1}$$

where $C(i)$ shows the number of steps in the random direction that is specified by the tumbling of ith bacterium and $\Delta(i)$ indicates a vector of random directions in population size with continuous values between [–1, 1].

2.2 *Swarm*

A group of *E. coli* bacteria arrange themselves in travelling ring by moving toward the nutrient collection. When the cells stimulated by a high level of succinate, release an attractant aspartate, such that the cells aggregate into groups and move as concentric patterns of swarms with high bacteria density. The cell-to-cell signalling in *E. coli* swarm can be represented as following function:

$$\begin{aligned} J_{cc}\left(\theta, P(j,k,l)\right) &= \sum_{i=1}^{S} J_{cc}\left(\theta, \theta^i(j,k,l)\right) \\ &= \sum_{i=1}^{S}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m=1}^{P}\left(\theta_m-\theta_m^i\right)^2\right)\right] \\ &= \sum_{i=1}^{S}\left[-d_{repelent}\exp\left(-w_{repelent}\sum_{m=1}^{P}\left(\theta_m-\theta_m^i\right)^2\right)\right] \end{aligned} \tag{2}$$

where $J_{cc}(\theta, P(j, k, l))$ shows objective function cost and in each step the amount of this objective function is added to the main cost (where the total cost to be minimised). In other words, J_{cc} shows how far a bacterium is from the fittest bacterium. θ is the position of p-dimensional search space of ith bacterium, jth chemotactic step, kth reproduction step, and lth elimination-dispersal step, P is number of variables to be optimised in each bacterium i, S is the total number of bacteria, and $d_{attract}$, $w_{attract}$, $h_{repellent}$, $w_{repellent}$ are the diversity coefficients (constant values) (Das et al., 2009).

2.3 *Reproduction*

After chemotactic and swarming parts got completed, some of the bacteria have enough nutrient and some others not. In the reproduction part, those bacteria that have enough nutrient will be reproduced and others will be eliminated. To do so, a health status of each bacterium is calculated.

The health status is the sum of step fitness during its life and can be defined as:

$$J_{health}^{i} = \sum_{j=1}^{N_c} J(i, j, k, l) \tag{3}$$

where N_c is total number of chemotactic steps, and J_{health} is calculated for ith bacterium, jth chemotactic step, kth reproduction step, and lth elimination and dispersal step. Eventually, the health status of all bacteria will be sorted in ascending order and the first half of the bacteria reproduce and surrogate into second half of the bacteria based on their top health status. In other words, the smaller J_{health} values related to healthier bacteria. Thus, bacteria with smaller health values have more chance to survive. This process not only keeps the population constant, but also the healthier bacteria continue to next generation.

2.4 Elimination and dispersal

When the density of bacteria getting high in a small area, the temperature of this area getting high and may not be enough nutrient for all bacteria as well. Thus, in this part, the population of bacteria may randomly change their positions to avoid these flaws. Elimination and dispersal event relocates the bacteria in different environments to avoid the bacteria death and local optimum solution(s).

In single objective BFO, the aim is to find one single solution, whereas the aim of the MOBFO is to obtain a set of non-dominated solutions to be closed to optimum Pareto front. Thus, Niu et al. (2013) described MOBFO approach to solve multi-objective optimisation problems. The idea of integration between bacteria health sorting and Pareto front mechanism are developed to search for non-dominated set solutions of multi-objective problems. Consequently, BFO algorithm can be used for single objective problems and multi-objective problems.

3 Proposed method

QAP is one of discrete optimisation problems that can be introduced in two different ways: single objective QAP and mQAP. First, we describe the proposed single objective BFO algorithm for the solution of single objective QAP, and we continue with the proposed MOBFO for the solution of mQAP in the next section.

3.1 Single objective BFO

We proposed a new BFO algorithm by adding swap mutation operator into the chemotactic part and local search algorithm at the end of algorithm to be applied to final solution. Swap mutation (Soni and Kumar, 2014) operator is one of the genetic operators that are used to update the solutions in a population. Given a permutation of a QAP solution, swap mutation selects two locations of the solution space and exchanges them, this happens only one time for each solution in the population and the population gets updated for next run. In modified BFO, we generated a random number between [0, 1] for each permutation and if the random number is greater than 0.5, we swapped the entities of two random locations, otherwise we divided the permutation into three blocks and applied the swap mutations in each block. Tabu search algorithm (Tabitha et al., 2009) is one of the well-known local search algorithms that searches through solution space locally and periodically to improve the solution x to x' around x. Basically, Tabu search is mostly used for NP-hard problems and it varies from other local search algorithms based on the Tabu list. Tabu list is a typical short-term memory that includes previously visited solutions and stores some of the attributes of the solutions. Thus, it does not allow to revisit the solutions so that the algorithm examines those movements that are not in the Tabu list. Consequently, the Tabu search algorithm is applied on final solution, and the new solution will be replaced with old solution if it produces a better result. The aim in the proposed single objective BFO is to find a single permutation solution with minimum cost in QAP and it is designed in three parts:

1 chemotactic
2 reproduction
3 elimination and dispersal.

Table 1 Initial settings for algorithm parameters and coefficients

Variables	*Setting*
Dimension of search space (p)	50
Number of bacteria per generation (S)	100
Chemotactic steps (N_c)	100
Reproduction steps (N_{re})	4
Elimination-dispersal steps (N_{ed})	5
Probability for elimination-dispersal (P_{ed})	0.25

The proposed single objective BFO is given in Algorithm 1 and consists of ten steps. In step 1, all the variables are initialised with preconceived constant numbers (Table 1). In step 2, a set of permutations with p size are randomly generated and QAP fitness function (Koopmans and Beckmann, 1957) is computed for each permutation.

Algorithm 1

```
1   Parameter initialisation:
    $p, S, N_c, N_s, N_{re}, N_{ed}, S_r, P_{ed}$
    $p$             dimension of search space
    $S$             population of bacteria
    $N_c$           chemotactic parts for each bacterium
    $N_{re}$        reproduction part
    $N_{ed}$        elimination dispersal part
    $S_r = S/2$     bacteria split
    $P_{ed}$        probability of elimination-dispersal.
2   Generate random permutation for each bacterium $i = 1, 2,$
    $\ldots, S$, and compute the cost using fitness function $J(i, j, k)$.
```

```
3   Sort the costs in ascending order and select the minimum
    one as best so far.
4   Elimination-dispersal counter: $ell = ell + 1$.
5   Reproduction counter: $k = k + 1$.
6   Chemotactic counter: $j = j + 1$.
    a   Take the chemotactic part for ith bacterium, $i = 1, 2, \dots, S$ as follows.
    b   Apply mutation on each bacterium ith.
    c   Compute the objective function $J(i, j, k)$.
    d   Sort the costs in ascending order and replace the
        minimum cost with the best so far if that is smaller.
    e   Go to b if $(i + 1) \neq S$.
    f   Get the minimum cost so far.
    g   Go to step 6 if $j + 1 < N_c$.
7   Sort bacteria cost in descending order and remove the
    second half of the population and duplicate the first half in
    order to keep the population constant. Go to step 5 if
    $k + 1 < N_{re}$.
8   Regenerate random permutation for all bacteria based on a
    random probability value $P_{ed}$.
9   Get minimum cost and replace with the best so far if that
    is smaller.
10  Apply Tabu search algorithm on the permutation related to
    best so far cost. Compute the cost of new permutation and
    replace with best so far, if that is smaller. Go to step 4 if
    $ell + 1 < N_{ed}$.
```

In step 3, the minimum cost through all solutions is set as best so far. In steps 4, 5, and 6 the chemotactic, reproduction, and elimination and dispersal parts are started, respectively. In step 6, the chemotactic part takes place, where we added swap mutation to update the permutations. We compute the cost using QAP fitness function (Koopmans and Beckmann, 1957) again for all updated permutations, and we substitute the minimum cost if that is less than best so far. In step 7, all permutations are sorted in ascending order based on their cost and the first half of the solutions are substituted into second half. We always use an even number of bacteria (Table 1) to keep the population constant. In step 8, we generate a random number between 0 and 1, and if the generated random number is bigger than P_{ed} (0.25), step 2 will be repeated here, otherwise the algorithm continues to step 9 with previous solutions. In step 9 we find the minimum cost among the permutation costs and if it is smaller than best so far then it will be replaced. In step 10, where we applied Tabu search algorithm, the best so far solution gets optimised with Tabu search algorithm and substitutes with the old solution if that is better, and it goes to step 4.

3.2 Multi-objective BFO

Unlike single objective BFO that is designed for single objective optimisation problems, MOBFO is designed for multi-objective optimisation problems. The proposed single objective BFO described in previous section and finds a single optimum solution in QAP, whereas the proposed MOBFO finds a non-dominated set solutions. In multi-objective optimisation problems, solution a is better than solution b if and only if solution a dominates solution b. Deb et al. (2002) introduced fast non-dominated sort method to compare different solutions in the multi-objective optimisation problems and find the best non-dominated set solutions. Briefly, a fast non-dominated sort method goes through each solution ith and computes the number of solutions that dominate ith solution and save them in a list. The final list is number of dominated solutions per each solution ith. Eventually, the algorithm sorts the list in ascending order and selects those solutions that have zero rank and print them as a non-dominated set solutions or Pareto front solutions. We use fast non-dominated sort method for the solution of mQAP that is a minimisation problem, although there are other non-dominated sort algorithms that are used to find the maximised solutions set (Srinivas and Deb, 1994; Ghosh and Das, 2008; Yazdi et al., 2017).

Like proposed single objective BFO, we added a swap mutation operator into chemotactic part and improved the final solution using local Tabu search algorithm. Additionally, we added one crossover operator into chemotactic part in order to update the non-dominated set solutions. Crossover is one of the main operators in GA that randomly exchanges some elements in the parent chromosomes and generates the new offspring in the next generation. There are many kinds of crossover methods that can be used to update the population, but a few of them can be used for permutation style population (Misevicius and Kilda, 2005).

Thus, we investigated these kinds of crossover by evaluating the results of each and turns out that ULX (Tate and Smith, 1995; Misevicius and Kilda, 2005) can be the best one among them.

Algorithm 2

```
1   Parameter initialisation:
    $p, S, N_c, N_s, N_{re}, N_{ed}, S_r, P_{ed}$
    $p$              dimension of search space
    $S$              population of bacteria
    $N_c$            chemotactic parts for each bacterium
    $N_{re}$         reproduction part
    $N_{ed}$         elimination dispersal part
    $S_r = S/2$      bacteria split
    $P_{ed}$         probability of elimination-dispersal
    $M$              number of fitness functions.
2   Generate random permutation for each bacterium $i = 1, 2, \dots, S$, and compute the cost using fitness function $J_b(i, j, k)$,
    where $b$ is number of fitness functions, $b = 1, 2, \dots, M$.
3   Get non-dominated set solutions using fast non-dominated
    sort algorithm.
4   Elimination-dispersal counter: $ell = ell + 1$.
5   Reproduction counter: $k = k + 1$.
6   Chemotactic counter: $j = j + 1$.
    a   Take the chemotactic part for ith bacterium, $i = 1, 2, \dots,$
```

S as follows:

b Apply crossover and mutation on each bacterium i^{th}.

c Compute the fitness function $J_b(i, j, k)$.

d Get the non-dominated set solutions again.

e Go to b if $(i + 1) \neq S$.

f Store all non-dominated solutions in each iteration here these are basis for non-dominated set solutions.

g Let bacteria with better rank continue for next iteration. Go to step 6 if $j + 1 < N_c$.

7 Sort the solutions in descending order using fast non-dominated sort algorithm and remove the second half of the population and duplicate the first half in order to keep the population constant. Go to step 5 if $k + 1 < N_{re}$.

8 Regenerate random permutation for all bacteria based on a random probability value P_{ed}.

9 Get non-dominated set solutions. Apply multi-objective Tabu search algorithm on final non-dominated set solutions. Compute the fitness functions of new solutions and replace with old non-dominated set solutions, if new ones are better. Go to step 4 if $ell + 1 < N_{ed}$.

Briefly, the ULX compares two permutations *A* and *B* and copies the common elements into a new solution (permutation). For the rest of the uncommon elements, it starts with first element of permutation *A* and copies into new permutation, then switch on the permutation *B* and copies the first uncommon element to the next empty location in the new permutation if there is no repeated element in the new permutation, otherwise switches to other uncommon element in another permutation. This process continues for all uncommon elements in the permutations *A* and *B* to fill out the new permutation empty locations. Note, if all of the uncommon elements of both permutations have been visited and still some of the locations remains empty in the new permutation, then a random unduplicated number will be generated to fill out any empty location. In modified MOBFO, after inserting common element of permutation *A* and *B* into a new permutation list, we generated a random number between [0, 1], and if the random number is greater than 0.5, we begin copying of elements from permutation *A*, otherwise we begin with permutation *B*.

The proposed MOBFO is given in Algorithm 2 and consists of ten steps. The steps of the algorithm are similar to the proposed single objective BFO (Algorithm 1), though at some points it is designed for multi-objective optimisation and we describe them here. In step 1, there is an additional variable *M*, that shows the number of objective functions under study. In step 2, a random permutation is created for each bacterium and MOBFO computes *M* objective functions for each bacterium. In step 3, a non-dominated set solutions are computed using fast non-dominated sort algorithm. In step 6, in addition to swap mutation, ULX is applied to update the solutions and step 3 is repeated. In step 9, step 3 is repeated again to update non-dominated set, and Tabu search algorithm is applied on the final non-dominated set to optimise the solutions locally, and it goes to step 4.

4 Experimental results

We were assessed the performance of the proposed single objective BFO algorithm using the instances from the well-known QAPLIB website (Burkard et al., 1997). We ran experimental analyses on a P4 Laptop with 1.0 GB RAM and 2.7 GHz Intel CPU. The operating system was Windows 7 and the developing software was MATLAB 2013a.

Table 2 List of the name, size, and cost of instances obtained from QAPLIB

Name	*Size*	*Cost*	*Prop. BFO*	*Orig. BFO*	*GA*
chr12a	12	9,552	0.10	0.64	NA
chr15a	15	9,896	0.12	0.86	NA
chr18a	18	11,098	0.30	1.01	NA
chr20a	20	2,192	0.18	1.04	NA
chr22a	22	6,156	0.07	0.51	NA
chr25a	25	3,796	0.40	1.17	NA
esc16a	16	68	0.00	0.32	2.94
esc16h	16	996	0.00	0.10	0.00
esc32a	32	130	0.20	0.89	NA
esc32e	32	2	0.00	4.00	0.00
esc32f	32	2	0.00	6.00	0.00
esc64a	64	116	0.00	0.55	NA
esc128	128	64	0.00	2.94	NA
had12	12	1,652	0.00	0.08	0.00
had14	14	2,724	0.00	0.09	0.07

Notes: Last three columns indicate the comparison of percentage gap between the proposed BFO (prop. BFO), original BFO (orig. BFO), and GA algorithms. NA indicates not available.

Table 2 List of the name, size, and cost of instances obtained from QAPLIB (continued)

Name	*Size*	*Cost*	*Prop. BFO*	*Orig. BFO*	*GA*
had16	16	3,720	0.00	0.07	0.91
had18	18	5,358	0.00	0.06	1.12
had20	20	6,922	0.00	0.06	1.61
kra30a	30	88,900	0.05	0.30	NA
lipa20a	20	3,683	0.00	0.05	NA
lipa20b	20	27,076	0.00	0.26	NA
lipa30a	30	13,178	0.01	0.04	3.19
lipa40a	40	31,538	0.01	0.03	NA
lipa50a	50	1,210,244	0.00	0.26	2.27
lipa60a	60	107,218	0.00	0.02	2.09
lipa70a	70	169,755	0.00	0.02	1.81
lipa80a	80	253,195	0.00	0.02	1.65
lipa90a	90	360,630	0.00	0.02	1.53
nug12	12	578	0.01	0.17	NA
nug14	14	1,014	0.10	0.14	NA
nug15	15	1,150	0.00	0.17	NA
nug16a	16	1,610	0.01	0.17	NA
nug17	17	1,732	0.01	0.17	NA
nug18	18	1,930	0.00	0.17	7.56
nug20	20	2,570	0.00	0.20	NA
nug21	21	2,438	0.00	0.24	NA
nug22	22	3,596	0.01	0.18	NA
nug24	24	3,488	0.00	0.20	NA
nug25	25	3,744	0.00	0.20	NA
nug27	27	5,234	0.01	0.23	NA
nug28	28	5,166	0.01	0.23	NA
nug30	30	6,124	0.00	0.20	NA
rou12	12	235,528	0.02	0.12	NA
rou15	15	354,210	0.03	0.17	NA
rou20	20	725,522	0.00	0.15	NA
scr12	12	31,410	0.00	0.28	NA
scr15	15	51,140	0.04	0.34	NA
scr20	20	110,030	0.00	0.75	NA
sko42	42	15,812	0.00	0.18	NA
sko49	49	23,386	0.01	0.17	NA
sko56	56	34,458	0.01	0.18	NA
sko64	64	48,498	0.01	0.16	NA
sko72	72	66,256	0.02	0.16	NA
sko81	81	90,998	0.01	0.14	NA
sko90	90	115,534	0.00	0.15	NA
sko100a	100	152,002	0.01	0.14	NA
ste36a	36	9,526	0.08	0.61	NA
tai12a	12	224,416	0.01	0.18	NA
tai15a	15	388,214	0.02	0.14	NA
tai17a	17	491,812	0.02	0.15	NA

Notes: Last three columns indicate the comparison of percentage gap between the proposed BFO (prop. BFO), original BFO (orig. BFO), and GA algorithms. NA indicates not available.

Table 2 List of the name, size, and cost of instances obtained from QAPLIB (continued)

Name	*Size*	*Cost*	*Prop. BFO*	*Orig. BFO*	*GA*
tai20a	20	703,482	0.01	0.14	NA
tai25a	25	1,167,256	0.02	0.14	9.84
tai30a	30	1,818,146	0.03	0.16	NA
tai35a	35	2,422,002	0.03	0.15	NA
tai40a	40	3,139,370	0.02	0.14	NA
tai50a	50	4,938,796	0.03	0.14	NA
tai60a	60	7,205,962	0.02	0.14	NA
tai64c	64	1,855,928	0.00	0.17	5.31
tai80a	80	13,499,184	0.03	0.13	11.44
tai100a	100	21,052,466	0.02	0.12	11.02
tai150b	150	498,896,643	0.03	0.23	NA
tai256c	256	44,759,294	0.00	0.12	NA
tho30	30	149,936	0.04	0.21	NA
tho40	40	240,516	0.02	0.28	NA
tho150	150	8,133,398	0.02	0.16	NA
wil50	50	48,816	0.00	0.10	7.65
wil100	100	273,038	0.00	0.08	NA

Notes: Last three columns indicate the comparison of percentage gap between the proposed BFO (prop. BFO), original BFO (orig. BFO), and GA algorithms. NA indicates not available.

Table 2 shows the comparison of the proposed BFO, original BFO, and GA (Tosun, 2014) on QAPLIB instances. The comparison is based on the percentage gap between the solution found and optimum solution reported in QAPLIB. In this table, the first column is the name of instances, the second column is the size of instances, third column is the best solution found using the proposed BFO algorithm, and last three columns are the percentage gap of the proposed BFO, original BFO, and GA, respectively. The comparison of the results indicates that the proposed BFO found the optimum solution in most of the instances and outperforms the results of original BFO and GA. These instances selected among more than 100 instances in QAPLIB to test the performance of the proposed algorithm with different problem size (12–256).

Indeed, we have not compared the time-consumption between different approaches because of the fact that GA approach ran on different platforms. Instead we compare the ultimate results to prove that the proposed method can found good solution regardless of size and time. There are many other studies that have been shown the time-consumption, but in order to compare the time we had to implement the different approaches and ran on the same machine. This might be one of the pitfalls of this paper and we leave this for future research.

We were assessed the performance of the proposed MOBFO using 22 mQAP instances (Knowles and Corne, 2003). The experimental analyses environment as well as parameters initialisation were the same as the proposed BFO algorithm according to Table 1.

Table 3 shows the list of 22 mQAP instances with their size and number of objectives. All of the instances are based on two objectives with different sizes (10–50) and categories (read-like and uniform). We have not assessed the proposed MOBFO on three dimensional problems and leave this part for future research.

Table 3 List of mQAP instances

Instance name	*Category*	*Size*	*Objective*
KC10-2fl-1rl	Real-like	10	2
KC10-2fl-2rl	Real-like	10	2
KC10-2fl-3rl	Real-like	10	2
KC10-2fl-4rl	Real-like	10	2
KC10-2fl-5rl	Real-like	10	2
KC20-2fl-1rl	Real-like	20	2
KC20-2fl-2rl	Real-like	20	2
KC20-2fl-3rl	Real-like	20	2
KC20-2fl-4rl	Real-like	20	2
KC20-2fl-5rl	Real-like	20	2
KC50-2fl-1rl	Real-like	50	2
KC50-2fl-2rl	Real-like	50	2
KC50-2fl-3rl	Real-like	50	2
KC10-2fl-1uni	Uniform	10	2
KC10-2fl-2uni	Uniform	10	2
KC10-2fl-3uni	Uniform	10	2
KC20-2fl-1uni	Uniform	20	2
KC20-2fl-2uni	Uniform	20	2
KC20-2fl-3uni	Uniform	20	2
KC50-2fl-1uni	Uniform	50	2
KC50-2fl-2uni	Uniform	50	

We also evaluated the performance of the proposed MOBFO using three assessment metrics, average generational distance (GD), convergence measure (CM), and divergence measure (DM) (Mostaghim and Teich, 2004; Okabe et al., 2003). GD reports how far, in average, known Pareto front is from true Pareto front. In other words, GD metric evaluates the average distance between the non-dominated set solutions (*E*) and optimum Pareto front (P^*). GD can be defined as:

$$GD\left(E, P^*\right) = \frac{1}{|E|}\sum_{u \in E} \min\left\{dist(u, v) \middle| v \in P^*\right\} \tag{4}$$

where $dist(u, v)$ is the Euclidean distance in objective space between the solution $u \in E$ and the nearest member in the optimum Pareto front ($v \in P^*$). Indeed, this metric measures how far the approximation front is from the optimum Pareto front. Note, the smaller value of GD represents a better performance.

CM evaluates average similarity of non-dominated solutions and Pareto front solutions. CM can be defined as:

$$CM = \frac{\dfrac{1}{1 + \left|\sum_{j=1}^{m}\sum_{k}^{n} pf(n) - nds(m, n)\right|}}{pf} \tag{5}$$

where *pf* is Pareto optimum vector, *nds* is non-dominated set solutions, *m* is size of non-dominated set vector, and *n* is number of objectives. Note, the bigger the CM, the better the result.

DM evaluates the spread distribution of vectors throughout the non-dominated set solutions and can be defined as:

$$DM = d_{pf} + d_{nds} + \sqrt{\frac{1}{|nds| - 1}\sum_{i=1}^{|nds|-1}\left(d_i - \bar{d}\right)^2} \tag{6}$$

where d_i is the Euclidean distance between non-dominated set solutions, *i* is the number of non-dominated set solutions, d_{pf} and d_{nds} are the Pareto optimum vector and non-dominated set solutions. DM is expected to be better as the search area is wider, rather than single point, so the bigger the DM value, the better the result (Lim et al., 2014).

Consequently, we compared the results of these metrics (GD, CM, and DM) between proposed MOBFO and four well-known EA approaches: mGRASP/MH (Zinflou et al., 2013), fuzzy PSO (Zhao et al., 2008), NSGAII (Deb et al., 2002), and original MOBFO (Niu et al., 2013). Table 4 shows the performance of assessment metrics obtained by each algorithm for the 22 mQAP instances.

Table 4 Average GD, CM, and DM values for mGRASP/MH, fuzzy PSO, NSGAII, original MOBFO, and proposed MOBFO

Instance name	*Metric*	*mGRASP/MH*	*Fuzzy PSO*	*NSGAII*	*Original MOBFO*	*Proposed MOBFO*
KC10-2fl-1rl	GD	6.0364e+04	0	0	3.6304e+04	0
	CM	0.0173	0.2242	1	0.0865	1
	DM	0.2069	0.2241	1	0.3103	1
KC10-2fl-2rl	GD	7.7505e+04	0	0	0	0
	CM	0.0667	0.2667	1	0.6668	1
	DM	0.3333	0.2667	1	0.6667	1
KC10-2fl-3rl	GD	6.5790e+04	0	0	2.0828e+04	0
	CM	1.0486e-05	0.1455	1	0.1092	1
	DM	0.0909	0.1455	1	0.2545	1
KC10-2fl-4rl	GD	1.0145e+04	0	0	2.2742e+03	0
	CM	0.3435	0.2283	1	0.2492	1
	DM	0.4340	0.2264	1	0.3208	1
KC10-2fl-5rl	GD	3.7627e+04	NA	0	0	7.8350e+04
	CM	0.2858	NA	1	0.2042	1
	DM	0.4490	NA	1	0.3061	1
KC20-2fl-1rl	GD	1.0004e+06	NA	8.6505e+04	3.3506e+05	1.4131e+03
	CM	1.1602e-05	NA	7.8179e-05	4.0738e-05	0.9140
	DM	0.1290	NA	0.4194	0.1398	0.9570
KC20-2fl-2rl	GD	5.2403e+06	NA	2.5075e+06	4.8649e+06	1.4435e+06
	CM	1.1227e-05	NA	1.0746e-04	1.8974e-05	0.5366
	DM	0.0909	NA	0.2455	0.0818	0.7909

Notes: First and second columns are the name of instances and metrics, and the rest of the columns compare the performance of algorithms. NA indicates not available.

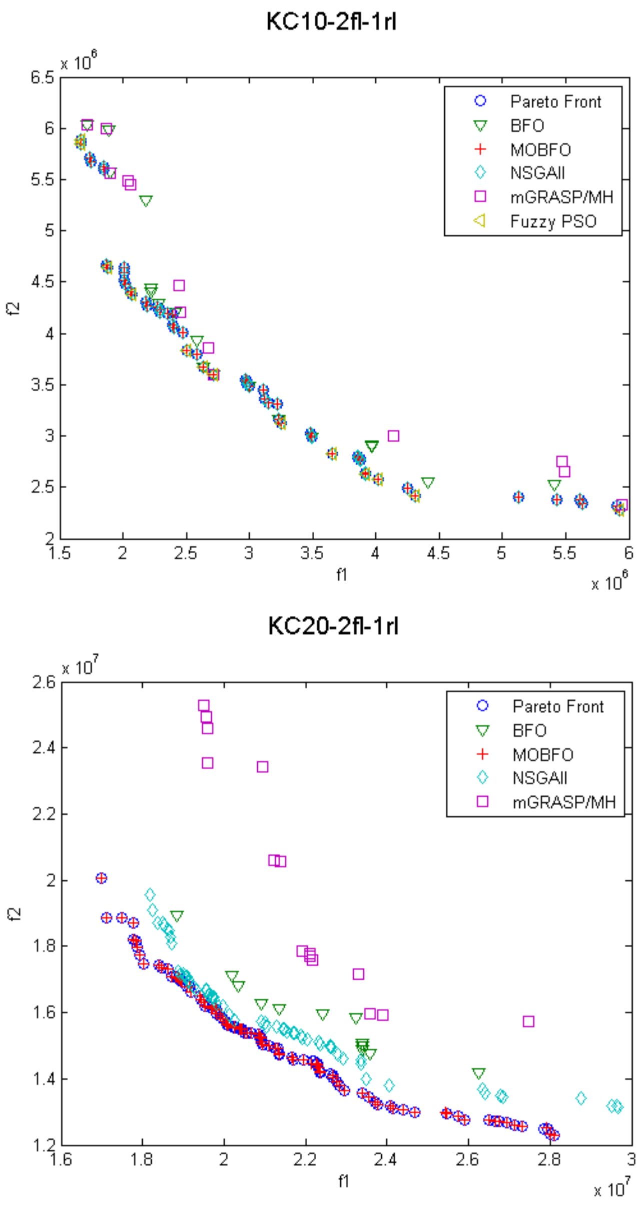
Table 4 Average GD, CM, and DM values for mGRASP/MH, fuzzy PSO, NSGAII, original MOBFO, and proposed MOBFO (continued)

Instance name	*Metric*	*mGRASP/MH*	*Fuzzy PSO*	*NSGAII*	*Original MOBFO*	*Proposed MOBFO*
KC20-2fl-3rl	GD	1.0763e+05	NA	4.1562e+04	4.0651e+06	7.9226e+03
	CM	2.0048e-05	NA	1.6249e-04	1.0101e-05	0.2711
	DM	0.0714	NA	0.3878	0.0408	0.5459
KC20-2fl-4rl	GD	9.4466e+05	NA	3.1785e+05	1.4372e+06	9.7568e+04
	CM	7.8523e-06	NA	0.0137	6.0072e-06	0.2467
	DM	0.1644	NA	0.4658	0.1096	0.6438
KC20-2fl-5rl	GD	7.8927e+05	NA	2.3552e+05	1.0829e+06	8.1897e+04
	CM	4.2539e-06	NA	3.3240e-05	1.9945e-06	0.0807
	DM	0.1129	NA	0.3952	0.0645	0.5000
KC50-2fl-1rl	GD	NA	NA	NA	1.8848e+06	2.4066e+04
	CM	NA	NA	NA	2.7198e-07	2.4554e-04
	DM	NA	NA	NA	0.0139	0.5278
KC50-2fl-2rl	GD	NA	NA	NA	2.1147e+06	3.5228e+04
	CM	NA	NA	NA	6.3955e-08	1.6289e-04
	DM	NA	NA	NA	0.0100	0.5116
KC50-2fl-3rl	GD	NA	NA	NA	1.2676e+06	2.7726e+04
	CM	NA	NA	NA	9.5983e-07	1.7589e-04
	DM	NA	NA	NA	0.0156	0.4630
KC10-2fl-1uni	GD	2.3550e+03	0	0	6.2979e+03	0
	CM	0.1554	1	1	0.0015	1
	DM	0.4615	1	1	0.4615	1
KC10-2fl-2uni	GD	8.5809e+03	0	0	0	0
	CM	1.2117e-04	1	1	1	1
	DM	1	1	1	1	1
KC10-2fl-3uni	GD	462.0308	0	0	427.8507	0
	CM	0.0488	1	1	0.0104	1
	DM	0.2077	1	1	0.1923	1
KC20-2fl-1uni	GD	1.4341e+04	NA	3.0844e+03	1.1162e+04	699.2820
	CM	9.6647e-04	NA	0.0026	8.2174e-04	0.5472
	DM	0.1818	NA	0.4727	0.1273	0.7455
KC20-2fl-2uni	GD	5.3530e+05	NA	5.2399e+05	8.5236e+05	3.4735e+05
	CM	1.6519e-04	NA	7.8772e-04	7.4803e-06	0.0059
	DM	0.3750	NA	0.5000	0.1250	0.7500
KC20-2fl-3uni	GD	2.1538e+03	NA	1.1816e+03	5.3478e+03	155.1058
	CM	1.8263e-07	NA	1.8208e-07	1.7929e-07	1.8413e-07
	DM	0.0051	NA	0.0051	0.0102	0.0051
KC50-2fl-1uni	GD	NA	NA	NA	7.5229e+04	2.3338e+03
	CM	NA	NA	NA	1.0256e-05	7.7719e-04
	DM	NA	NA	NA	0.0471	0.5294
KC50-2fl-2uni	GD	NA	NA	NA	2.2387e+05	6.0161e+03
	CM	NA	NA	NA	1.5426e-06	2.9983e-04
	DM	NA	NA	NA	0.0588	0.4118
KC50-2fl-3uni	GD	NA	NA	NA	2.3057e+04	417.7430
	CM	NA	NA	NA	1.0802e-04	0.0035
	DM	NA	NA	NA	0.0378	0.5689

Notes: First and second columns are the name of instances and metrics, and the rest of the columns compare the performance of algorithms. NA indicates not available.

The first column is the instance name, the second column is the evaluation metric name, and the rest of the columns indicate the results for each algorithm. Based on the results, the proposed MOBFO performed better than other algorithms in terms of the average GD, CM, and DM values. Other methods have either large GD value or small CM and DM values.

Figure 1 Comparison results of five algorithms, proposed MOBFO, original BFO, mGRASP/MH, fuzzy PSO, NSGA2, and Pareto front (see online version for colours)



Notes: In this figure KC10/KC20 related to problem size, and 2fl is related to number of objective functions. Also, rl/uni related to type of the problem whether is real (rl) or uniform (uni).

Additionally, we illustrated the graphical performance comparison of these five algorithms including optimum Pareto front for two instances from Table 4 (Figure 1). Although, the proposed MOBFO outperforms NSGAII, Fuzzy PSO, mGRASP/MH, and original MOBFO, but in very rare cases the approximation sets found by the proposed MOBFO are very close to Pareto optimum solutions. For the future direction, this algorithm can resolved hyper-parameter optimisation for machine learning algorithm (Parvandeh and McKinney, 2018; Parvandeh et al., 2019, 2020). Additionally, we can improve the document summarisation when we use cosine similarity to find the optimised summary from multi-document in natural language processing problems (Parvandeh et al., 2016).

5 Conclusions

In this paper, we proposed modified BFO and MOBFO algorithms for the solution of QAP and mQAP instances. The proposed BFO algorithm updates the solution using the swap mutation method in chemotactic part and improves the final solution using local Tabu search algorithm to reach local optimum. And, the proposed MOBFO attempts to find a non-dominated set solutions to be as close as possible to Pareto front solutions. MOBFO determines the non-dominated set solutions using fast non-dominated sort method and updates solutions set using swap mutation and ULX operation. Additionally, the final non-dominated set solutions improves using multi-objective version of local Tabu search to reach the local optimum. We assessed the proposed BFO and MOBFO algorithms on QAP and 22 mQAP instances from the QAPLIB. Computational results showed that BFO and MOBFO performed well on these instances. We compared both algorithms with the number of state-of-the-art algorithms as well as the best-known solutions from QAPLIB. The comparison showed that our results are very close to the best-known solutions and outperforms the state-of-the-art algorithms.

References

Alothaimeen, I. and Arditi, D. (2019) 'Overview of multi-objective optimization approaches in construction project management', in *Multi-Criteria Optimization – Pareto-Optimal and Related Principles*.

Benlic, U. and Hao, J-K. (2013) 'Breakout local search for the quadratic assignment problem', *Applied Mathematics and Computation*, Vol. 219, No. 9, pp.4800–4815.

Benlic, U. and Hao, J-K. (2015) 'Memetic search for the quadratic assignment problem', *Expert Systems with Applications*, Vol. 42, No. 1, pp.584–595.

Burkard, R.E., Karisch, S.E. and Rendl, F. (1997) 'QAPLIB – a quadratic assignment problem library', *Journal of Global Optimization*, Vol. 10, No. 4, pp.391–403.

Cui, Z., Zhang, J., Wang, Y., Cao, Y., Cai, X., Zhang, W. and Chen, J. (2019) 'A pigeon-inspired optimization algorithm for many-objective optimization problems', *Science China Information Sciences*, Vol. 62, No. 7, p.70212.

Cui, Z., Zhang, J., Wu, D., Cai, X., Wang, H., Zhang, W. and Chen, J. (2020) 'Hybrid many-objective particle swarm optimization algorithm for green coal production problem', *Information Sciences*, May, Vol. 518, No. 1, pp.256–271.

Darvishi, A., Alimardani, A., Vahidi, B. and Hosseinian, S.H. (2014) 'Bacterial foraging-based algorithm optimization based on fuzzy multiobjective technique for optimal power flow dispatch', *Science International*, Lahore, 26 August, pp.1057–1064.

Das, S., Biswas, A., Dasgupta, S. and Abraham, A. (2009) 'Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications', in Abraham, A., Hassanien, A-E., Siarry, P. and Engelbrecht, A. (Eds.): *Foundations of Computational Intelligence, Global Optimization*, Vol. 3, pp.23–55, Springer, Berlin, Heidelberg.

Deb, K. (2004) 'Single and multi-objective optimization using evolutionary computation', in *Hydroinformatics*, pp.14–35, World Scientific Publishing Company, World Scientific, Singapore.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp.182–197.

Emmerich, M.T.M. and Deutz, A.H. (2018) 'A tutorial on multiobjective optimization: fundamentals and evolutionary methods', *Natural Computing*, Vol. 17, No. 3, pp.585–609.

Ghosh, A. and Das, M. (2008) 'Non-dominated rank based sorting genetic algorithms', *Fundamenta Informaticae – FUIN '83*, January, pp.231–252.

Gunantara, N. (2018) 'A review of multi-objective optimization: methods and its applications', in Ai, Q. (Ed.): *Cogent Engineering*, Vol. 5, No. 1, p.1502242.

Knowles, J. and Corne, D. (2003) 'Instance generators and test suites for the multiobjective quadratic assignment problem', in Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L. and Deb, K. (Eds.): *Evolutionary Multi-Criterion Optimization*, pp.295–310, Springer, Berlin, Heidelberg.

Knowles, J.D. and Corne, D.W. (2000) 'Approximating the non-dominated front using the Pareto archived evolution strategy', *Evolutionary Computation*, Vol. 8, No. 2, pp.149–172.

Koopmans, T.C. and Beckmann, M. (1957) 'Assignment problems and the location of economic activities', *Econometrica*, Vol. 25, No. 1, pp.53–76.

Li, S-X. and Wang, J-S. (2017) 'Improved Cuckoo search algorithm with novel searching mechanism for solving unconstrained function optimization problem', *IAENG International Journal of Computer Science*, ,Vol. 44, No. 1, pp.8–12.

Lim, K.S., Buyamin, S., Ahmad, A., Shapiai, M.I., Naim, F., Mubin, M. and Kim, D.H. (2014) 'Improving vector evaluated particle swarm optimisation using multiple nondominated leaders', in Agarwal, P., Bhatnagar, V. and Zhang, Y. (Eds.): *The Scientific World Journal*, Vol. 14, p.364179.

Luo, C., Yin, X. and Ni, C. (20015) 'A novel discrete bacterial foraging algorithm and its application', *JCM*, Vol. 10, No. 4, pp.238–244.

Michalak, K. (2016) 'Evolutionary algorithm with a directional local search for multiobjective optimization in combinatorial problems', *Optimization Methods and Software*, Vol. 31, No. 2, pp.392–404.

Misevicius, A. and Kilda, B. (2005) 'Comparison of crossover operators for the quadratic assignment problem', *Information Technology and Control*, Vol. 34, No. 2, pp.109–119.

Mostaghim, S. and Teich, J. (2004) 'A new approach on many objective diversity measurement', in *Practical Approaches to Multi-Objective Optimization*, pp.1–15, Dagstuhl, Germany.

Niu, B., Wang, H., Wang, J. and Tan, L. (2013) 'Multi-objective bacterial foraging optimization', *Neurocomputing*, Vol. 116, pp.336–345.

Ojha, M., Singh, K.P., Chakraborty, P. and Verma, S. (2019) 'A review of multi-objective optimisation and decision making using evolutionary algorithms', *International Journal of Bio-Inspired Computation*, Vol. 14, No. 2, pp.69–84.

Okabe, T., Jin, Y. and Sendhoff, B. (2003) 'A critical survey of performance indices for multi-objective optimisation', in *The 2003 Congress on Evolutionary Computation, CEC '03*, Vol. 2, pp.878–885.

Parvandeh, S. and McKinney, B.A. (2018) 'EpistasisRank and EpistasisKatz: interaction network centrality methods that integrate prior knowledge networks', in Kelso, J. (Ed.): *Bioinformatics*, 1 July, Vol. 35, No. 13, pp.2329–2331.

Parvandeh, S., Lahiri, S. and Boroumand, F. (2016) 'PerSum: novel systems for document summarization in Persian', *International Journal of Asian Language Processing*, Vol. 26, No. 2, pp.67–108.

Parvandeh, S., Poland, G., Kennedy, R., McKinney, B., Parvandeh, S., Poland, G.A., Kennedy, R.B. and McKinney, B.A. (2019) 'Multi-level model to predict antibody response to influenza vaccine using gene expression interaction network feature selection', *Microorganisms*, Vol. 7, No. 3, p.79.

Parvandeh, S., Yeh, H-W., Paulus, M.P. and McKinney, B.A. (2020) 'Consensus features nested cross-validation', *Bioinformatics*, 15 May, Vol. 36, No. 10, pp.3093–3098.

Passino, K.M. (2002) 'Biomimicry of bacterial foraging for distributed optimization and control', *IEEE Control Systems Magazine*, Vol. 22, No. 3, pp.52–67.

Passino, K.M. (2010) 'Bacterial foraging optimization', *Int. J. Swarm. Intell. Res.*, Vol. 1, No. 1, pp.1–16.

Shukla, A. (2015) 'A modified bat algorithm for the quadratic assignment problem', in *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp.486–490.

Soni, N. and Kumar, T. (2014) 'Study of various mutation operators in genetic algorithms', *International Journal of Computer Science and Information Technologies*, Vol. 5, No. 3, pp.4519–4521.

Srinivas, N. and Deb, K. (1994) 'Muiltiobjective optimization using nondominated sorting in genetic algorithms', *Evolutionary Computation*, Vol. 2, No. 3, pp.221–248.

Tabitha, J., Rego, C. and Glover, F. (2009) 'A cooperative parallel Tabu search algorithm for the quadratic assignment problem', *European Journal of Operational Research*, Vol. 195, No. 3, pp.810–826.

Tate, D.M. and Smith, A.E. (1995) 'A genetic approach to the quadratic assignment problem', *Computers & Operations Research*, Vol. 22, No. 1, pp.73–83.

Tosun, U. (2014) 'A new recombination operator for the genetic algorithm solution of the quadratic assignment problem', *Procedia Computer Science*, Vol. 32, pp.29–36.

Tosun, U., Dokeroglu, T. and Cosar, A. (2013) 'A robust island parallel genetic algorithm for the quadratic assignment problem', *International Journal of Production Research*, Vol. 51, No. 14, pp.4117–4133.

Wang, C-F. and Zhang, Y-H. (2016) 'An improved artificial bee colony algorithm for solving optimization problems', *JOUR*, January, Vol. 43, pp.336–343.

Wang, J-S. and Song, J-D. (2017) 'Chaotic biogeography-based optimisation algorithm', *IAENG International Journal of Computer Science*, Vol. 44, No. 2, pp.127–139.

Yazdi, J., Choi, Y.H. and Kim, J.H. (2017) 'Non-dominated sorting harmony search differential evolution (NS-HS-DE): a hybrid algorithm for multi-objective design of water distribution networks', *Water*.

Yi, J., Huang, D., Fu, S., He, H. and Li, T. (2016) 'Multi-objective bacterial foraging optimization algorithm based on parallel cell entropy for aluminum electrolysis production process', *IEEE Transactions on Industrial Electronics*, Vol. 63, No. 4, pp.2488–2500.

Zhang, Q. and Li, H. (2007) 'MOEA/D: a multiobjective evolutionary algorithm based on decomposition', *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 6, pp.712–731.

Zhao, F., Liu, Y., Shao, Z., Jiang, X., Zhang, C. and Wang, J. (2016) 'A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem', *International Journal of Computer Integrated Manufacturing*, Vol. 29, No. 9, pp.962–981.

Zhao, M., Abraham, A., Grosan, C. and Liu, H. (2008) 'A fuzzy particle swarm approach to multiobjective quadratic assignment problems', in *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pp.516–521.

Zinflou, A., Gagné, C. and Gravel, M. (2013) 'A hybrid genetic/immune strategy to tackle the multiobjective quadratic assignment problem', *Artificial Life Conference Proceedings*, July, Vol. 25, pp.933–939.

Zitzler, E. and Thiele, L. (1999) 'Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach', *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp.257–271.