Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB

Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB

A H Ismail[1,2], N Hartono[1,3], S Zeybek[1,4], D T Pham[1]

[1]Department of Mechanical Engineering, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK
[2]ahi729@student.bham.ac.uk, [3]nxh886@student.bham.ac.uk,
[4]sxz879@student.bham.ac.uk

Abstract. The Bees Algorithm (BA) is a metaheuristic algorithm to find good solutions to optimisation problems in reasonable computing times. This paper is the first to report on the use of the BA to solve 9 combinatorial optimisation problems (COP) with more than 100 cities from TSPLIB rigorously to test the performance of the algorithm. The work employed a basic version of the BA for COP and TSPLIB datasets involving between 100 and 200 cities. The results obtained show that deviations from the best-found tour lengths for the datasets with 100 cities and 200 cities were approximately 2.5% and 7.5%. The reason for this jump in deviations was that the number of iterations was kept constant for all experiments while the solution space increased factorially with the number of cities. This research can be replicated and modified through Google Colab.
Keywords: bees algorithm, combinatorial optimisation problem, metaheuristics algorithm, travelling salesman problem.

1. Introduction

One of the most intensive fields studied in mathematics, engineering, and computer science is the optimisation which involves finding the best solution or a feasible solution from candidates in the solution space. An important class of practical optimisation problems is that of combinatorial problems. Many COPs are NP-hard and thus challenging to solve because the number of combinations exponentially increases with the size of the problem [1]. The need to find acceptable near-optimal solutions in reasonable computing times has motivated researchers to develop heuristic and meta-heuristic algorithms. According to Ouaraab, an advantage of metaheuristic algorithms is that they are simple and flexible [1]. Nature-inspired metaheuristic algorithms including those based on the behaviour of social insects such as honey bees have attracted the attention of many researchers [2].

As stated by Lones, there are 15 nature-inspired metaheuristic algorithms that have each received more than 1,000 citations [3]. One of those algorithms is the Bees Algorithm (BA) proposed by Pham et al. in 2005 [4], [5]. The BA is inspired by the natural honey bees foraging behaviour and can be used to solve continuous and combinatorial optimisation problems. A comparative study of the BA, Particle Swam Optimisation (PSO), and Evolutionary Algorithm (EA) show that the BA outperforms PSO and EA [6]. The BA also outperforms the deterministic simplex method, genetic algorithm, ant colony system, and stochastic annealing optimisation procedure [5]. The BA has been very popular since its development and it has many applications in multi-discipline area from work scheduling, image processing, pattern recognition, manufacturing, arc routing, PCB manufacturing, matrices structure, data partitioning, data transmission in a network, power distribution networks, and robot navigation [7],[8].

The BA has already found more than 400 different applications covering a variety of continuous or combinatorial problems, such as pattern recognition, scheduling, engineering design, neural network training, manufacturing cell formation, control system tuning, dynamic control, data clustering, mechanical design and supply chain design [7], [8], [9], [10]. There have been at least 56 endeavours to develop new versions of the BA. However, only 4 out of those 56 attempts have focused on combinatorial optimisation although COP formed approximately 33% of the total number of applications of the algorithm and the problem of combinatorial optimisation arises in many fields of engineering and other scientific disciplines that use computational approaches, such as artificial intelligence, operations research, electronic commerce, and bioinformatics. The authors believe that the reason for the lack of activity on combinatorial versions of the BA is that there are no available results on the performance of the BA on standard data-sets (test functions) that can be used for comparison in developing the algorithm.

Although there are many publications addressing real-world applications in the combinatorial domain such as machine scheduling [11],[12], cellular manufacturing [13], Printed Circuit Board (PCB) manufacturing [14],[15],[16], vehicle routing [17], it is important to use standard data sets when assessing the performance of an algorithm. The results for one or more standard data sets could be employed to compare different algorithms.

The basic combinatorial version developed by Koc [18] was adopted in this work as the basic version of the BA for combinatorial optimisation (“the Combinatorial BA”). This study extends Koc’s previous work that used simple problems up to 51 cities. The performance of many algorithms is satisfactory for simple problems. However, the problem becomes challenging when the number of cities increases. The Google Colab link for this work is provided in the appendix to give a clear starting point to develop the Combinatorial BA.

One of the most common problems in combinatorial optimisation is the travelling salesman problem (TSP). The problem has many variants such as PCB manufacturing, machine scheduling, cellular manufacturing, arc routing, vehicle routing, job shop scheduling, data routing, machine sequencing, computer wiring, DNA sequencing, etc. TSP is a NP-hard optimisation problem because the complexity of the computation increases exponentially with the number of cities [1]. The TSP was initially introduced in 1930 and is one of the most studied COP as it can be used as a benchmark for evaluating the performance of many combinatorial optimisation algorithms [19].

The most widely used dataset library for COP is TSPLIB, which was published by Reinelt in 1991 [20]. The library itself has been cited more than 2,400 times, according to Google Scholar (October 2019). It is a very popular library and has been used mostly by researchers to evaluate optimisation algorithms such as the GA, PSO and ACO. However, no work has been done to measure the performance of the BA on TSPLIB problems involving more than 100 cities. This paper will present the results of testing the basic version of the BA on TSPLIB datasets with 100 to 200 cities. A reason for choosing those datasets was that most metaheuristic algorithms, including the BA, give good results for datasets with fewer than 100 cities.

The basic combinatorial version of the BA adopted in this research employed a permutation random number generator as the global search operator. The basic local search operator was a combination of swap, insertion, and reversion. This local search operator was chosen as the algorithm performed poorly when swap, insertion or reversion alone was used.

1.1. Bees Algorithm

Pham et al. introduced the BA as a nature-inspired metaheuristic optimisation method inspired by the food foraging behaviour of the swarm honey bees [4], [5]. The BA classified as one of the swarm intelligence algorithms. Figure 1 illustrates the flow chart of The BA.

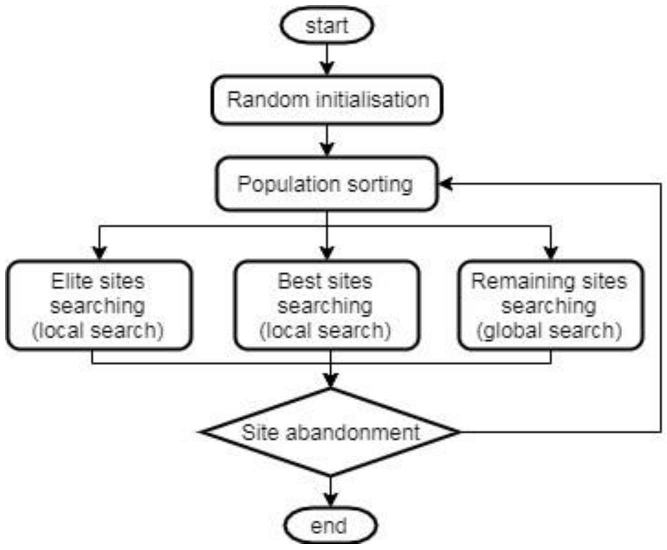

start
Random initialisation
Population sorting
Elite sites searching (local search)
Best sites searching (local search)
Remaining sites searching (global search)
Site abandonment
end

Figure 1. Flowchart of Bees Algorithm (basic version)[5]

The BA for the continuous domain has become more advanced and better structured compared to the version for the combinatorial domain. Work on the algorithm can be classified into four major sections: parameter tuning and setting, initialisation, local search (exploitation), and global search (exploration). Most of the developments primarily concentrate on the local search procedure of the BA [7]. All of the previous work on the BA in the continuous domain had different characteristics from the combinatorial version, such as the solution space is real rather than discrete. Therefore, the BA needs to be modified to perform exploitation and exploration in the combinatorial domain without changing the principal procedure of the algorithm.

1.2. Bees Algorithm in Combinatorial Optimisation Problem

The first COP solved by BA was reported in 2007 for finding the solution of the machine scheduling [11]. In the same year, BA also had solved PCB assembly sequence as an early TSP based COP [14]. After those publication in 2007, other important works began to emerge in the combinatorial BA algorithm such as timetabling [21], [22], [23], [24], PCB assembly planning [15], [16], circuit designing [25], scheduling [12], [26], [27], [28], [29], cellular manufacturing [13], vehicle routing [17], disassembly sequence planning [30], [31], and path planning [32], [33].

Combinatorial and continuous problems have completely different search principles. The task of searching as the main feature in the combinatorial problem is distinctive from the continuous problem where there is no real distance value [18]. The combinatorial domain has main characteristics [34]: the search space is discrete; the constraints are finite; the solution has ordered sequence and a cost function which related to the combination. Since the BA was originally developed to solve continuous domains, it is important to change the searching section in global and local search with a discrete search operator.

In essence, the BA for COP is almost identical to the algorithm for continuous one. The main difference between continuous and combinatorial version is lying on their (global and local search) operators. The combinatorial version, of course, has to use a discrete random generator replacing the real number generator on the continuous version.

1.3. Travelling Salesman Problem

TSP can be classified as symmetrical or asymmetrical TSP. In symmetrical TSPs, the distances between the two cities do not rely on the trajectory direction. For example, if the distance between two cities i and j when travelling from i to j is denoted by di,j and if $d_{i,j} = d_{j,i}$ then the TSP is symmetrical and vice versa.

TSP's graph represented the Graph $G = (V,E)$, where V is a set of vertices representing the cities, and all the connecting lines between the cities is E. Every edge indicates a possible route between two connected vertices or cities. The variable $d_{i,j}$ is associated with an edge (i,j) and represents the Euclidean distance from vertex (x_i,y_i) to (x_j,y_j) as Equation (1). Before executing the BA, these distances of all edges have calculated and store as a distance matrix.

$$d_{i,j} = ((x_i - x_j)^2 - (y_i - y_j)^2)^{0.5} \tag{1}$$

The TSP objective is to find the minimal total tour length of the final closed Hamilton cycle (only visiting once of all cities) tour as defined in Equation (2).

$$tour\ length = \sum_{i=1}^{n} d_{i,i+1} + d_{n,1} \tag{2}$$

Koc, Otri and Zeybek were three of the researchers developing the BA for solving TSPs taken from the TSPLIB dataset up to 100 cities [18],[34],[35].

2. Methods

The first step before running the BA is setting up the parameters. There are 5 parameters that have to be initially set up as number of scout bees (n), elite bees (nep), best bees (nsp), elite sites (e), and best sites (m). All the parameters dependently set to the n. The parameter nsp and m are 50% of n. The e is 40% of m. The nep has a double number of nsp. The n set equal to 40 bees (see Table 1).

After setting up the parameters, random initialisation (see Figure 1) will generate n initial solutions. The best e out of n initial solutions will be the elite sites which best seeds to be improved. The m sites excluding the e sites are sufficiently good seeds as sites to be exploited as well.

Table 1. Parameter value of the running experiments

Parameter	Value
Number of runs	10
Number of iterations	3,000
Number of scout bees *(n)*	40
Number of elite bees *(nep)*	40
Number of best bees *(nsp)*	20
Number of elite sites *(e)*	16
Number of best sites *(m)*	20

The elite and best sites are the places needed to be exploited by worker bees (*nep* and *nsp*) using neighbourhood or local search (see Figure 2 to 5). The remaining ($n - e - m$) bees as insufficiently seeds will explore the other solution spaces using the global search operator. All worker bees will continuously renew the patch a better position for every iteration and abandoned the worn-out one. The position that a worker bees had is representing a solution (complete tour). The best solution is generated by sorting all the best positions from all sites. The code is written in Python and could be retrieved in Google Colab.

```
Input: Cities: List,selection for Swap, Insertion and Reversion: integer
Output: Changed List of cities indexes
procedure LOCAL SEARCH($List, LS$)
    begin
        $N := sizeOfList(List)$
        $i_{th} := generateRandomInteger[0, N-1]$
        $j_{th} := generateRandomInteger[0, N-1]$
        if $LS == 0$ then
            $LS \leftarrow generateRandomInteger[0, 3]$
        if $LS == 1$ then
            $List \leftarrow$ SWAP($List, i, j$)
        if $LS == 2$ then
            $List \leftarrow$ INSERTION($List, i, j$)
        if $LS == 3$ then
            $List \leftarrow$ REVERSION($List, i, j$)
        return List
    end
```

Figure 2. Pseudo-code of the basic local search operator of the combinatorial BA

```
Input: Cities indexes: List, i: integer, j: integer
Output: Changed List of cities indexes
procedure SWAP($List, i, j$)
    begin
        $Temp = List[i]$
        $List[i] \leftarrow List[j]$
        $List[j] \leftarrow Temp$
        return $List$
    end
```

Figure 3. Pseudo-code of swap operator

```
Input: Cities indexes:List, i: integer, j: integer
Output: Changed List of cities indexes
procedure INSERTION($List, i, j$)
    begin
        while $i < j$ do
            $Temp = List[i+1]$
            $List[i+1] \leftarrow List[j]$
            $i \leftarrow i+1$
            $j \leftarrow j-1$
        return $List$
    end
```

Figure 4. Pseudo-code of insertion operator

```
Input: Cities indexes:List, i: integer, j: integer
Output: Changed List of cities indexes
procedure REVERSION($List, i, j$)
    begin
        while $i < j$ do
            $Temp = List[i]$
            $List[i] \leftarrow List[j]$
            $List[j] = Temp$
            $i \leftarrow i+1$
            $j \leftarrow j-1$
        return $List$
    end
```

Figure 5. Pseudo-code of reversion operator

3. Result and Discussion

Nine benchmark problems from TSPLIB [20] were used in this work to evaluate the performance of the combinatorial BA with the basic local search operator. The experimental results are summarised in Table 2. The first column of Table 2 gives the TSPLIB dataset names and the tour length for the best-known solution (*BKS*). The second column lists the tour length for the best solution of 10 runs, each of 3,000 iterations (*Best*). The third column gives the average tour length of the solutions found by the ten runs (*Avg*). The fourth column gives the deviation of the Avg from BKS (*Davg*). The last column lists the deviation of the Best from BKS (*Dbest*). *Davg* and *Dbest* as the accuracy indicators are defined in Equation (3) and (4).

$$Davg = (Avg - BKS).(BKS)^{-1}.100\% \tag{3}$$

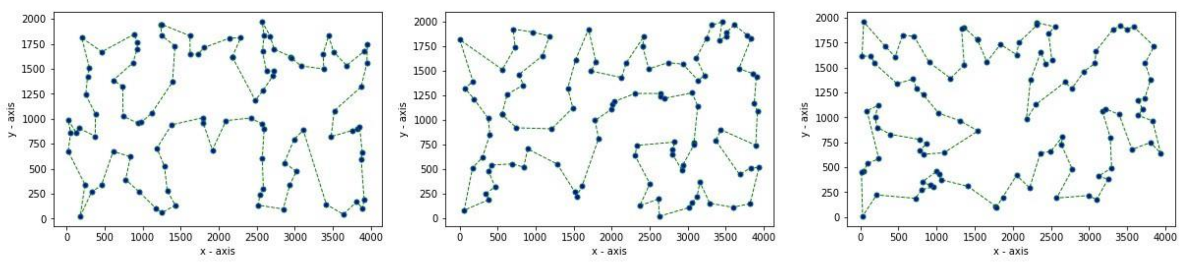
$$Dbest = (Best - BKS).(BKS)^{-1}.100\% \tag{4}$$

Table 2. Results for ten runs each of 3000 iterations for 100-200 cities data-sets from TSPLIB using the combinatorial BA

Dataset names (BKS)	Best	Avg	Davg	Dbest
KroA100 (21,282)	21,469	21,712.4	2.02%	0.88%
KroB100 (22,141)	22,618	22,879.0	3.33%	2.15%
KroC100 (20,749)	20,969	21,339.8	2.85%	1.06%
KroD100 (21,389)	21,392	21,896.7	2.37%	0.01%
KroE100 (22,068)	22,266	22,496.1	1.94%	0.90%
KroA150 (26,524)	27,527	27,871.2	5.08%	3.78%
KroB150 (26,130)	27,095	27,458.5	5.08%	3.69%
KroA200 (29,368)	31,550	32,214.0	9.69%	7.43%
KroB200 (29,437)	32,024	32,519.7	10.47%	8.79%

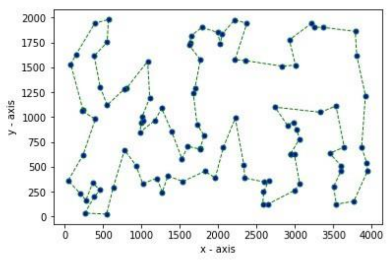
The overall result shows that the *Davg* starts from 1.94% up to 10.47% and for the *Dbest* from 0.01% up to 8.79%. The results of 100 cities' datasets show that the *Davg* have range between 1.94% and 3.33% while the *Dbest* range between 0.01% and 2.15%. For the 150-200 cities results, the highest *Davg* is 10.47% and the lowest is 5.08%, while the *Dbest* has a range from 3.69% to 8.79%. The performance for 150-200 cities' shows that the deviation is around three times larger than performance of 100 cities' results due to the solution spaces were factorially increasing while the number of iteration remains the same. The authors believe that the accuracy performance of the combinatorial BA will increase if the number of iteration of the algorithm were adjusted according to the problem complexity (number of cities). The poor convergence has been found as other problem in this work. ACO's previous work [36] in 1999 had proved that ACO convergence performance would increase when ACO utilising a constructive heuristic (nearest neighbour) on the candidate solution generator. Later, It provokes the other researchers to be more intense to develop the constructive heuristic [37],[35],[30] as initialisation or global search operator for the algorithm development.

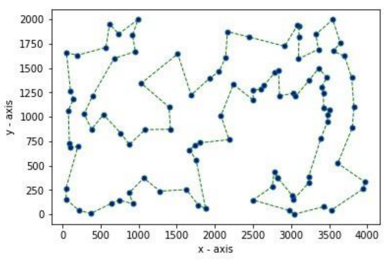
Overall, the basic combinatorial BA has the ability to provide good solutions with average error around 2.5% for 100 cities datasets and 7.5% for 150-200 cities datasets. Figure 6, Figure 7,and Figure 8 show the best-found tour length results for every TSPLIB's datasets in this research.

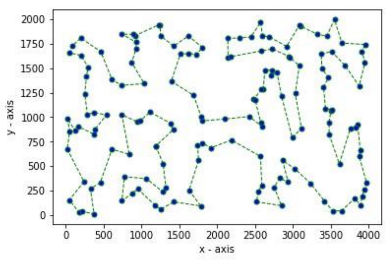


(a) tour length: 21,469 **(b)** tour length: 22,618 **(c)** tour length: 20,969

Figure 6. Best solution of (a) KroA100, (b) KroB100, (c) KroC100

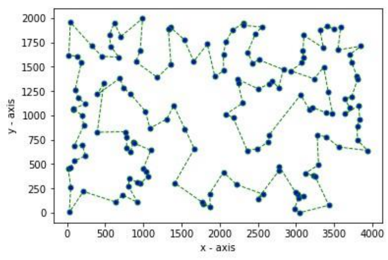


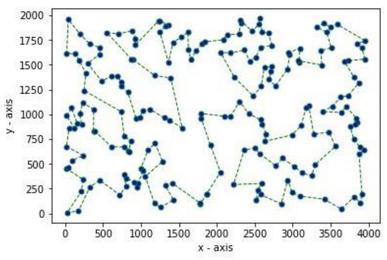


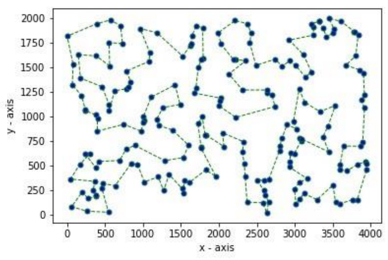


(a) tour length: 21,392 **(b)** tour length: 22,266 **(c)** tour length: 27,527

Figure 7. Best solution of (a) KroD100, (b) KroE100, (c) KroA150







(a) tour length: 27,095 **(b)** tour length: 31,550 **(c)** tour length: 32,024

Figure 8. Best solution of (a) KroB150, (b) KroA200, (c) KroB200

4. Conclusion

This paper has shown that the BA can find near optimal solutions to the COP. The algorithm uses the permutation random number generator as global search operator and a combination of swap, insertion and reversion as the basic local search operator. Experiments using TSPLIB datasets with 100 to 200 cities show that the deviation increases around three times from the 100 cities dataset to the 200 cities dataset. The reason for this surge was that the number of iterations was kept constant for all experiments while the solution space increased factorially with the number of cities. This poor convergence for larger problems is a weakness common to many swarm intelligence or nature-inspired algorithms. The convergence problem is a challenge to address in developing these nature-inspired algorithms including the BA.

For further research, the other constructive and improved heuristic algorithms potentially hybridised to the BA either as a global or local search operator to improve the performance of the algorithm for solving the COP.

Appendix A. Google Colab link

Google Colab link for this research: https://bit.ly/2MaJTyG or http://bit.ly/2LR7rIO or https://bit.ly/2YYXQVo

Appendix B. Authors' ORCID

https://orcid.org/0000-0002-6580-4654, https://orcid.org/0000-0003-2314-1394, https://orcid.org/0000-0002-1298-9499

Acknowledgments

The authors would like to acknowledge the generous support from the Newton Fund-Ristekdikti, Indonesia Endowment Fund for Education (Lembaga Pengelola Dana Pendidikan Republik Indonesia), University of Birmingham, University of Pancasila, Industrial Engineering University of Pelita Harapan, The Scientific and Technological Research Council of Turkey (TÜBİTAK) and The Fatih Sultan Mehmet Vakif University (FSMVU).

5. References

[1] Ouaarab A, Ahiod B and Yang X S 2014 Discrete cuckoo search algorithm for the travelling salesman problem *Neural Computing and Applications* **24** (Springer) pp 1659–69

[2] Bitam S, Batouche M and Talbi E g 2010 A survey on bee colony algorithms *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW) (IEEE)* pp 1–8

[3] Lones M A 2020 Mitigating metaphors: A comprehensible guide to recent natureinspired algorithms *SN Computer Science* **1** (Springer) p 49

[4] Pham D, Ghanbarzadeh A, Koc E, Otri S, Rahim S and Zaidi M 2005 The bees algorithm *Technical Note,* Manufacturing Engineering Centre, Cardiff University, UK

[5] Pham D T, Ghanbarzadeh A, Koc¸ E, Otri S, Rahim S and Zaidi M 2006 The bees algorithm—a novel tool for complex optimisation problems *Intelligent Production Machines and Systems* (Elsevier) pp 454–59

[6] Pham D T, Castellani M and Le Thi H A 2014 Nature-inspired intelligent optimisation using the bees algorithm *Transactions on Computational Intelligence XIII* (Springer) pp 38–69

[7] Hussein W A, Sahran S and Abdullah S N H S 2017 The variants of the bees algorithm (ba): *A survey Artificial Intelligence Review* **47** (Springer) pp 67– 121

[8] Rajasekhar A, Lynn N, Das S and Suganthan P N 2017 Computing with the collective intelligence of honey bees–a survey *Swarm and Evolutionary Computation* **32** (Elsevier) pp 25–48

[9] Yuce B, Packianather M, Mastrocinque E, Pham D and Lambiase A 2013 Honey bees inspired optimization method: the bees algorithm *Insects* **4** (Multidisciplinary Digital Publishing Institute) pp 646–62

[10] Pham D T and Castellani M 2015 A comparative study of the bees algorithm as a tool for function optimisation *Cogent Engineering* **2** (Taylor & Francis) p 1091540

[11] Pham D, Koc E, Lee J and Phrueksanant J 2007 Using the bees algorithm to schedule jobs for a machine *Proceedings of Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance* pp 430–39

[12] Packianather M S, Yuce B, Mastrocinque E, Fruggiero F, Pham D T and Lambiase A 2014 Novel genetic bees algorithm applied to single machine scheduling problem *2014 World Automation Congress (WAC) (IEEE)* pp 906–11

[13] Pham D T, Afify A and Koc E 2007 Manufacturing cell formation using the bees algorithm *Innovative Production Machines and Systems Virtual Conference*, Cardiff, UK

[14] Pham D, Otri S and Darwish A H 2007 Application of the bees algorithm to pcb assembly optimisation *Proceedings 3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)* pp 511–16

[15] Mei C A, Pham D T, Anthony J S and Kok W N 2010 Pcb assembly optimisation using the bees algorithm enhanced with triz operators *36th Annual Conference on IEEE Industrial Electronics Society (IEEE)* pp 2708–13

[16] Ang M C, Ng K W, Pham D T and Soroka A 2013 Simulations of pcb assembly optimisation based on the bees algorithm with triz-inspired operators *International Visual Informatics Conference* (Springer) pp 335–46

[17] Fenton A 2016 The bees algorithm for the vehicle routing problem *Thesis* (University of Auckland, NZ) arXiv preprint arXiv:1605.05448

[18] Koc E 2010 Bees algorithm: theory, improvements and applications *Thesis* (Cardiff University, UK)

[19] Gutin G and Punnen A P 2006 The traveling salesman problem and its variations (Springer Science & Business Media)

[20] Reinelt G 1991 Tsplib—a traveling salesman problem library *ORSA journal on computing* **3** (INFORMS) pp 376–84

[21] Lara C, Flores J J and Caldero´n F 2008 Solving a school timetabling problem using a bee algorithm *Mexican International Conference on Artificial Intelligence* (Springer) pp 664–74

[22] Phuc N B, Khang N T T M and Nuong T T H 2011 A new hybrid ga-bees algorithm for a real-world university timetabling problem *2011 International Conference on Intelligent Computation and Bio-Medical Instrumentation (IEEE)* pp 321–326

[23] Khang N T T M, Phuc N B and Nuong T T H 2011 The bees algorithm for a practical university timetabling problem in vietnam *2011 IEEE International Conference on Computer Science and Automation Engineering* **4** (IEEE) pp 42–7

[24] Abdullah S and Alzaqebah M 2013 A hybrid self-adaptive bees algorithm for examination timetabling problems *Applied Soft Computing* **13** (Elsevier) pp 3608–20

[25] Mollabakhshi N and Eshghi M 2013 Combinational circuit design using bees algorithm *IEEE Conference Anthology (IEEE)* pp 1–4

[26] Yuce B, Pham D, Packianather M S and Mastrocinque E 2015 An enhancement to the bees algorithm with slope angle computation and hill climbing algorithm and its applications on scheduling and continuous-type optimisation problem *Production & Manufacturing Research* **3** (Taylor & Francis) pp 3–19

[27] Mayteekrieangkrai N and Wongthatsanekorn W 2015 Optimized ready mixed concrete truck scheduling for uncertain factors using bee algorithm Songklanakarin *J. Sci. Techn.* **37** 221–30

[28] Ghasemi B, Sadeghi A, Roghani M and Branch S T 2015 The solution of multiobjective multimode resource-constrained project scheduling problem (rcpsp) with partial precedence relations by multi-objective bees algorithm *Silvae Genetica* (ISSN: 0037-5349) **57**

[29] Yuce B, Fruggiero F, Packianather M S, Pham D T, Mastrocinque E, Lambiase A and Fera M 2017 Hybrid genetic bees algorithm applied to single machine scheduling with earliness and tardiness penalties *Computers & Industrial Engineering* **113** (Elsevier) pp 842–58

[30] Liu J, Zhou Z, Pham D T, Xu W, Ji C and Liu Q 2018 Robotic disassembly sequence planning using enhanced discrete bees algorithm in remanufacturing *International Journal of Production Research* **56** (Taylor & Francis) pp 3134–51

[31] Laili Y, Tao F, Pham D T, Wang Y and Zhang L 2019 Robotic disassembly replanning using a two-pointer detection strategy and a super-fast bees algorithm *Robotics and Computer-Integrated Manufacturing* **59** (Elsevier) pp 130–42

[32] Darwish A H, Joukhadar A and Kashkash M 2018 Using the bees algorithm for wheeled mobile robot path planning in an indoor dynamic environment *Cogent Engineering* **5** (Taylor & Francis) p 1426539

[33] Sabri A N, Radzi N H M and Samah A A 2018 A study on bee algorithm and a algorithm for pathfinding in games *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE) (IEEE)* pp 224–29

[34] Otri S 2011 Improving the bees algorithm for complex optimisation problems *Thesis* (Cardiff University, UK)

[35] Zeybek S and Ko¸c E 2015 The vantage point bees algorithm *7th International Joint Conference on Computational Intelligence (IJCCI)* **1** pp 340–45

[36] Stützle T, Dorigo M et al. 1999 Aco algorithms for the traveling salesman problem *Evolutionary algorithms in engineering and computer science* **4** 163–83

[37] Ismail A H 2019 Domino algorithm: a novel constructive heuristics for traveling salesman problem *IOP Conference Series: Materials Science and Engineering* **528** (IOP Publishing) p 012043