Journal of Mechanical Engineering and Sciences
ISSN (Print): 2289-4659; e-ISSN: 2231-8380
Volume 13, Issue 4, pp. 5905-5921, December 2019
© Universiti Malaysia Pahang, Malaysia
DOI: https://doi.org/10.15282/jmes.13.4.2019.13.0469

A modified artificial bee colony algorithm to optimise integrated assembly sequence planning and assembly line balancing

M. F. F. Ab. Rashid[1*], N. M. Z. Nik Mohamed[1] and A. N. Mohd Rose[1]

[1] Faculty of Mechanical & Manufacturing Engineering,
Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
[*]Email: ffaisae@ump.edu.my
Phone: +6094246321; Fax: +6094246222

ABSTRACT

Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) are traditionally optimised independently. However recently, integrated ASP and ALB optimisation has become more relevant to obtain better quality solution and to reduce time to market. Despite many optimisation algorithms that were proposed to optimise this problem, the existing researches on this problem were limited to Evolutionary Algorithm (EA), Ant Colony Optimisation (ACO), and Particle Swarm Optimisation (PSO). This paper proposed a modified Artificial Bee Colony algorithm (MABC) to optimise the integrated ASP and ALB problem. The proposed algorithm adopts beewolves predatory concept from Grey Wolf Optimiser to improve the exploitation ability in Artificial Bee Colony (ABC) algorithm. The proposed MABC was tested with a set of benchmark problems. The results indicated that the MABC outperformed the comparison algorithms in 91% of the benchmark problems. Furthermore, a statistical test reported that the MABC had significant performances in 80% of the cases.

Keywords: Manufacturing system; artificial bee colony; assembly sequence planning; assembly line balancing.

INTRODUCTION

Global competitiveness continues to put pressure on manufacturers to produce a product at its maximum efficiency. In order to assemble the product at maximum efficiency, assembly optimisation activities play an important role. Assembly optimisation starts with product design stage that involves the assembly design [1]. The purpose of the assembly optimisation in the design stage is to reduce the number of parts and to ease the assembly process. This will lead to lower cost and less time taken to assemble the product in a flow line system [2].

Assembly sequence planning (ASP) is assembly optimisation activity that occurs during the production planning stage that widely known because of it uncertainty [3, 4]. The purpose of the ASP is to identify the best feasible assembly sequence according to the evaluation criteria [5]. In ASP optimisation, various evaluation criteria were used by the researchers to determine the best assembly sequence. Among the popular evaluation criteria mentioned in the published research were to minimise assembly direction change, tool change, and assembly complexity level [6].

On the other hand, the assembly line balancing (ALB) refers to the assignment of assembly task into the workstations, so that the workload between the workstations will be balanced or almost balanced [7, 8]. ALB problems are categorised as Simple Assembly Line Balancing Problem (SALBP) and Generalise Assembly Line Balancing Problem (GALBP). The SALBP considers a single model assembly problem that runs on a specific assembly line. Meanwhile, the GALBP includes all other versions of the problem other than SALBP's [9].

Recently, researchers discovered that the integrated assembly optimisation is able to lead to better assembly plan quality and reduce the error rate in assembly planning and costing [10]. In other words, the assembly process in production line could be improved by reducing non-value added activities such as unnecessary tool/direction changes and additional workstations via integrated ASP and ALB optimisation. Furthermore, the assembly time could be improved by reducing the cycle time [11].

Realising the benefits of integrated optimisation, the ASP and ALB have a good potential to integrate because both activities aim to achieve optimum set up in the assembly process. However, the number of published researches on integrated ASP and ALB is still lacking compared to independent optimisation for ASP and ALB.

In general, research on integrated ASP and ALB optimisation can be divided into two categories according to problem modelling. The first category is modelled based on assembly connectors [10]. In this approach, the assembly information such as direction, tool, and time are defined based on the connector used to assemble the components such as screw, pressing fit and welding joints. The main advantage of this approach is that the problem size is smaller because a set of assembly tasks can be grouped under one connector. This makes the optimisation process easier because of the smaller search space.

The second category of the integrated ASP and ALB optimisation uses the task-based modelling [12]. In the task-based modelling, the assembly information is defined according to the assembly task which consists of mating two components and/or subassemblies. This modelling approach is more popular because the assembly task is closely linked to the assembly process rather than the assembly connector. Furthermore, the task-based modelling is the most popular modelling approach in ALB.

The integrated ASP and ALB optimisation involves a multi-objective problem. The most frequent optimisation objectives considered in this problem are minimise assembly direction change, tool change, cycle time, and number of workstation. In some literature, the researchers consider the combination type that depends on the type of connectors [10]. To deal with the multi-objective problem, some researchers combined all optimisation objectives using the weighted sum approach [13]. While some others implement the domination concept to search for the Pareto optimal solutions [10, 14].

Although various optimisation algorithms were proposed in different applications, to the best of our knowledge, only three types of algorithms were implemented to optimise the integrated ASP and ALB. These algorithms are Evolutionary Algorithm (EA), Ant Colony Optimisation (ACO), and Particle Swarm Optimisation (PSO). Based on the published researches on integrated ASP and ALB, earlier researchers implemented EA to optimise the problem [10, 15–17]. However, in the past five years, researchers started to implement the swarm algorithms such as ACO and PSO algorithms with different justifications [18, 12, 14].

The EA showed satisfactory performance by optimising the integrated ASP and ALB. However, a similar drawback was reported where the EA parameter needs to be re-tuned when the problem size increases [15, 16]. Meanwhile, the ACO and PSO algorithms were globally known for their premature convergence problem. In ACO, a global pheromone mechanism was used to avoid the local optima solution [12]. On the other hand, researcher adopted the crowding distance mechanism in PSO to avoid the algorithm from being trapped in local optima [14].

In the meantime, in many different optimisation problems that compare various optimisation algorithms, the Artificial Bee Colony (ABC) shows better performance compared to other algorithms [19, 20]. In combinatorial problem such as a sequence-dependent disassembly line balancing for instance, the ABC algorithm shows superior performance compared to the other six algorithms, including PSO, ACO, and Genetic Algorithm (GA) [21]. In a different combinatorial problem, the ABC algorithm also performed better than the GA and PSO algorithms to optimise the scheduling problem [22, 23]. However, the ABC in its original form, is having a problem with slow convergence due to poor exploitation ability [24, 25]. The exploitation refers to the ability to make use of the existing solution in order to reproduce a better solution. This drawback makes the ABC require larger iteration numbers compared to other algorithms.

This work therefore, aims to increase the ABC's performance by improving the reproduction mechanism in this algorithm. According to the original ABC steps, the main weakness of the ABC is the absence of elitism in the reproduction process. The solution reproduction is conducted by mating a particular solution with a random solution within the population. In this case, the authors proposed to replace the onlooker bee phase with a beewolf predatory concept. The beewolf predatory implement a leadership hierarchy concept from the Grey Wolf Optimiser (GWO) to guide the search direction in ABC. Compared to other algorithms, the GWO algorithm is guided by three leaders to determine the search direction. This mechanism is predicted to be more efficient because it will avoid the algorithm from being trapped in local optima. In algorithms with a single leader, the solution will converge toward a single leader that makes the chance to be trapped in local optima higher compared to the multiple leaders.

INTEGRATED ASP AND ALB PROBLEMS

The integrated ASP and ALB problem consists of two elements (*G, I*). *G* represents the precedence relation for the problem, $G = (V, C)$. *V* is a set of assembly task from 1 to *n*, $V = \{1, 2, \ldots, n\}$, while *C* represents the set of precedence relation between task *i* and *j*. Meanwhile, the *I* element characterises the assembly information, which includes the assembly time (t_i), assembly direction (D_i), and assembly tool (T_i) for task $i = 1,2,\ldots,n$. In addition, the maximum allowable cycle time ($ct_{\max}$) for a particular problem is also determined, $I = (t, D, T, ct_{\max})$.

Figure 1 presents an example of a precedence graph that reflects the precedence relation, *G*. In the precedence graph, the number in the node represents the assembly task. In this context, the assembly task is referred to the smallest working element that consists of two components and/or subassemblies. Meanwhile, the arc represents the precedence in assembly. The task with an outgoing arc refers to the precedence for the task with incoming arc.

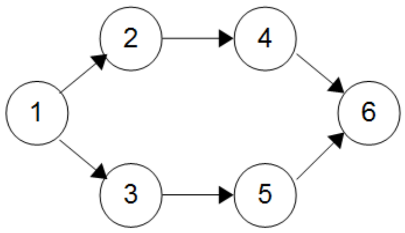


Figure 1. Example of precedence graph.

For the ALB, the task-based representation is the most common way to present the problem, since the assembly time is measured based on the assembly activity [26]. While in ASP, the most popular representation approach is on the assembly component basis. This is because the important parameters for ASP were measured based on the component, such as assembly direction and tool. In order to simultaneously optimise the ASP and ALB, the assembly direction and tool must be redefined based on the task-based representation. In this work, the authors focus to optimise the simple assembly line balancing problem, type E (SALBP-E), which aims to minimise both cycle time and workstation together.

Since the task-based representation consists of two components and/or subassemblies, the authors define one component as a moving part, while the other as a fixed part. Therefore, the assembly direction is redefined as the direction of bringing the moving part to the fixed part in an assembly task. On the other hand, the assembly tool can simply be determined by the type of tool used to accomplish the *i*th assembly task.

Objective Function and Constraints

To formulate the objective function, the authors have identified and considered the related optimisation objectives for ASP and ALB. For ASP optimisation, the optimisation objective is to minimise the number of direction change (n_D) and tool change (n_T). Meanwhile, the optimisation objectives for ALB include minimise cycle time (*ct*), number of workstation (*nws*), and workload variation (*h*).

$$n_D = \sum_{i=1}^{n-1} D_i; \qquad D_i = \begin{cases} 1 \text{ if the } i^{th} \text{direction} \neq (i^{th}+1) \text{ direction} \\ 0 \text{ if the } i^{th} \text{direction} = (i^{th}+1) \text{ direction} \end{cases} \tag{1}$$

$$n_T = \sum_{i=1}^{n-1} T_i; \qquad T_i = \begin{cases} 1 \text{ if the } i^{th} \text{ tool} \neq (i^{th}+1) \text{ tool} \\ 0 \text{ if the } i^{th} \text{ tool} = (i^{th}+1) \text{ tool} \end{cases} \tag{2}$$

$$ct = \max_{m=1:nws} [pt_m] \tag{3}$$

In Equation. (1), D_i refers to assembly direction for the *i*th task, while T_i is assembly tool for the *i*th task in Equation. (2). The processing time (pt_m) refers to the summation of the task time in the *m*th workstation, which cannot exceed the maximum allowable cycle time (ct_{max}).

$$h = \frac{\sum_{m=1}^{nws}(ct - pt_m)}{nws} \tag{4}$$

The optimisation objectives in Equation. (1) to (4) need to be normalised to ensure that they have a similar range of value to form an objective function. For this purpose, the optimisation objectives are normalised using the following formulas:

$$\hat{n}_D = \frac{n_D - n_{Dmin}}{n_{Dmax} - n_{Dmin}} \tag{5}$$

$$\hat{n}_T = \frac{n_T - n_{Tmin}}{n_{Tmax} - n_{Tmin}} \tag{6}$$

$$\widehat{ct} = \frac{ct - ct_{min}}{ct_{max} - ct_{min}} \tag{7}$$

$$\widehat{nws} = \frac{nws - nws_{min}}{nws_{max} - nws_{min}} \tag{8}$$

$$\hat{h} = \frac{h - h_{min}}{h_{max} - h_{min}} \tag{9}$$

Therefore, the objective function for this problem can be formulated as follows:

$$\text{Minimise } f = w_1\hat{n}_D + w_2\hat{n}_T + w_3\widehat{ct} + w_4\widehat{nws} + w_5\hat{h} \tag{10}$$

In this function, the authors set w_1, w_2, w_3, w_4, w_5 = 0.2. The objective function is calculated subjected to the following constraints:

$$\sum_{m=1}^{nws} x_{i,m} = 1 \qquad i = 1, \dots, n \tag{11}$$

$$\sum_{m=1}^{nws} x_{a,m} - \sum_{m=1}^{nws} x_{b,m} \le 0 \qquad a \in n, b \in F_a \tag{12}$$

$$\sum_{i=1}^{n} t_i x_{i,m} \le ct_{max} \qquad \forall m \tag{13}$$

The first constraint in Equation. (11) ensures that an assembly task is assigned into one workstation. Equation. (12) represents the precedence constraint that must be followed. The F_a refers to the set of successors for task *i*. In other words, this constraint ensures that the successor/s for task *i* will be assigned in a similar or the following workstation. The constraint in Equation. (13) ensures that the maximum cycle time (ct_{max}) is obeyed, for m^{th} workstation.

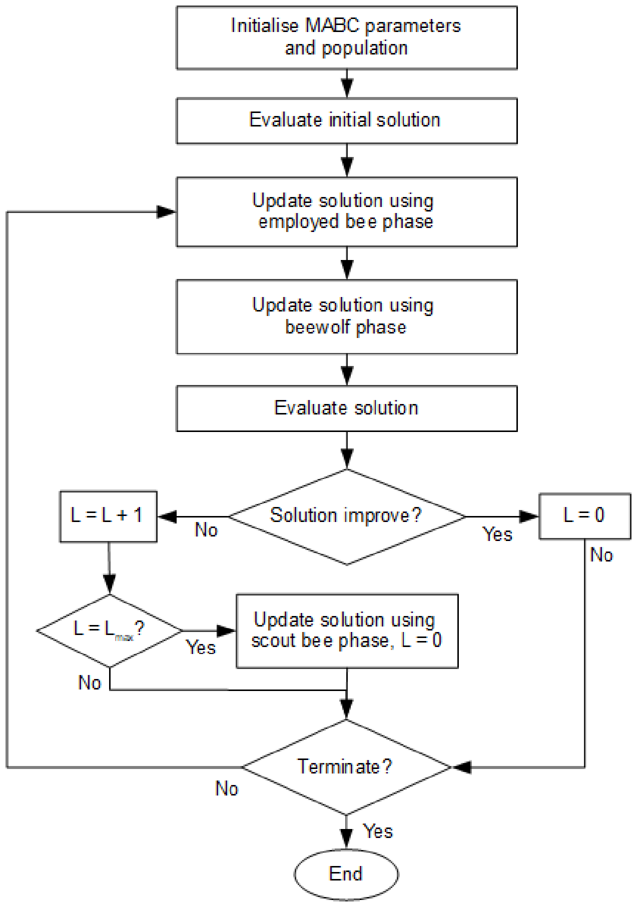
MODIFIED ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony (ABC) algorithm is developed based on the behaviour of foraging in bee colonies. This algorithm was proposed by Karaboga in 2005 [27]. The ABC algorithm comprises three groups of bees: Scout, onlooker, and employed bees. The employed bee is the group that works to search for a solution. The onlooker bee works to further improve the solution, while the scout bee works to avoid the solution from being trapped in local optimum.

According to the original ABC procedure, the regeneration process in the employed bee phase does not involve any elitism element from the best solution (or leader). The solution is generated using Equation. (16), where the x_k is randomly selected. On the other hand, in the onlooker bee phase, the algorithm tries to further improve the leaders by mating them with another randomly selected solution.

In this work, the authors proposed to replace the onlooker bee phase with a leadership inheriting mechanism in the ABC to guide the search direction. The leadership inheriting mechanism refers to the solution regeneration process that involves the best solution from the population. However, the existing algorithms with the leader inheriting mechanism, such as the PSO and ACO, are having a problem with being trapped in a local optimum [12, 14]. This problem occurred because the regeneration process is relying on a single leader.

This work therefore, adopted a beewolves predatory mechanism originally from the Grey Wolf Optimiser (GWO) [28]. In GWO, the search direction is guided by three leaders instead of a single leader in other algorithms. The flowchart of the proposed modified Artificial Bee Colony (MABC) algorithm is presented in Figure 2, while the pseudocode is in Figure 3.



Initialise MABC parameters and population
Evaluate initial solution
Update solution using employed bee phase
Update solution using beewolf phase
Evaluate solution
Solution improve?
No
Yes
L = L + 1
L = 0
No
L = L_max ?
Yes
No
Update solution using scout bee phase, L = 0
Terminate?
No
Yes
End

Figure 2. Flowchart of the proposed MABC.

```
Procedure of MABC
Initialise MABC parameters: Swarm size ($N_{swarm}$), number of employed bees ($NS$), number of
    beewolves, limit for scout ($L_{max}$) and maximum iteration ($iter_{max}$)
Initialise random population $x_i$ for $i = 1, 2, \ldots, NS$
Decode the $x_i$ into feasible assembly sequence
Evaluate the fitness function for $i^{th}$ solution, $f_i$
        Save the best $x_\alpha$, second best $x_\beta$ and third best $x_\delta$ solutions
Calculate $fit_i$ using Equation. (15)
Set iteration counter, $iter = 1$
Set Limit counter for $i^{th}$ solution, $L_i = 0$
While $iter \le iter_{max}$
        For $i = 1, 2, \ldots, NS$ ➔ Employed bees phase
                Generate new solution, $v_i$ from the $x_i$ solution using Equation. (16)
                Evaluate fitness for new solution, $f(v_i)$
                If $f(v_i) < f_i$
                        Update $x_i = v_i$
                        Reset $L_i = 0$
                End
                Calculate the probability for $i^{th}$ solution ($p_i$) using Equation. (17)
        End
        Sort solution according to better $p_i$

        For $w = 1, 2, \ldots, NS$ ➔ Beewolf phase
                Calculate the distance between $x_w$ and $x_\alpha$, $x_\beta$ and $x_\delta$ using Equation. (22) and (23)
                Regenerate new solution, $v_w$ using Equation. (24)
                Evaluate fitness for solution $v_w$, $f(v_w)$
                If $f(v_w) < f(x_w)$
                        Update $x_w = v_w$
                        Reset $L_i = 0$
                End
        End
        Update the best solution $x_\alpha$, $x_\beta$ and $x_\delta$
        $L_i = L_i + 1$
        If $L_i > L_{max}$ ➔ Scout bees phase
                Replace $x_i$ with a new random solution using Equation. (25)
        End
        $iter = iter + 1$
End
```

Figure 3. Procedure of the proposed MABC

Initialisation

The initialisation stage in the MABC algorithm involved defining the control parameters, such as swarm size (N_{swarm}), number of employed bees (*NS*), onlooker bees (*NO*), and limit for scout (L_{max}). In the original ABC, the numbers of employed bees and onlooker bees were both set as 50% from the swarm size. In the proposed MABC, the *NS* is maintained at 50% of the N_{swarm}, while the 50% balances are shared between onlooker bees (25%) and beewolves (25%).

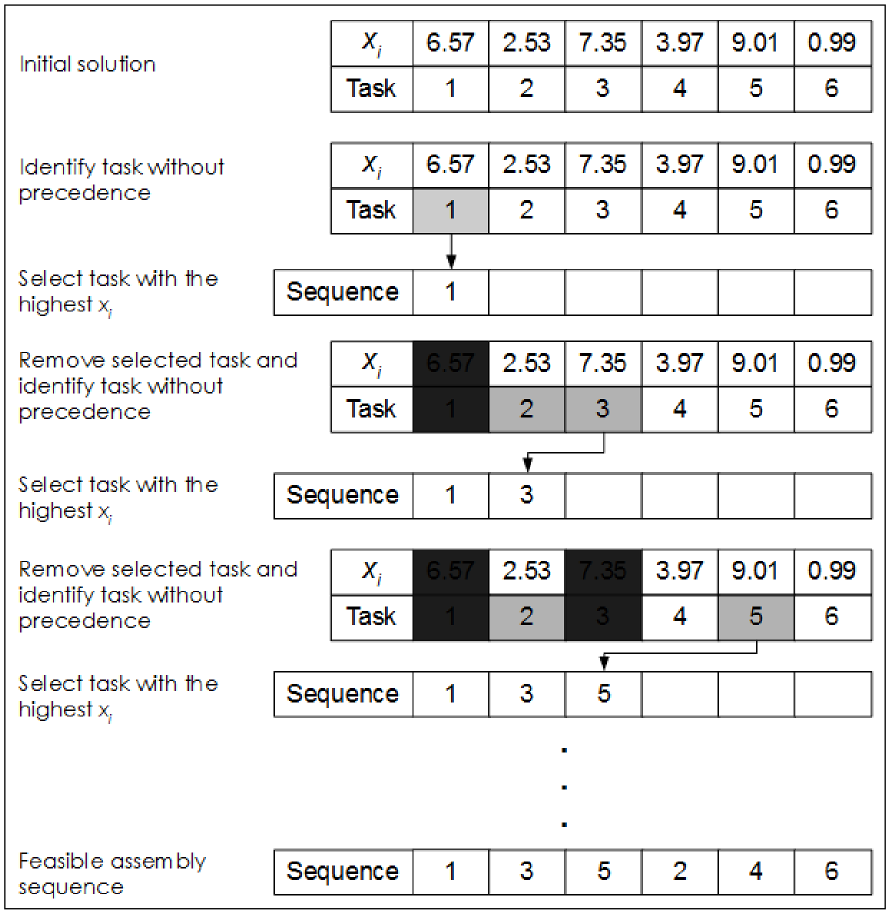
Meanwhile, the limit for scout bees (L_{max}) presents the maximum number of the iteration for the *i*th solution to discover an improved solution. If the *i*th solution does not show any improvement after L_{max} iteration, the *i*th solution will be replaced with a new solution using Equation. (25). The limit for scout (L_{max}) is calculated as follows:

$$L_{max} = \frac{N_{swarm} * dim}{2} \tag{14}$$

Where, *dim* is the dimension of the problem. The initial population is randomly generated within the lower and upper bounds for the *j*th dimension. The number of initial solution is equivalent to the number of food source and employed bees (*NS*).

Evaluation

Next, the initial population is evaluated using the fitness function. The fitness value for solution x_i is remarked as f_i. The solution x_i is decoded using topological sort as presented in Figure 4. An example of the decoding procedure in Figure 4 is taken from the precedence graph in Figure 1. The decoding procedure starts by identifying the candidate task without the precedence constraint. The candidate task without the precedence relation is in the grey box. Then the assembly task with higher x_i value will be selected among the candidate tasks. The selected assembly task will be removed from the next consideration. These steps are repeated until all of the assembly tasks are selected.



Initial solution
x_i 6.57 2.53 7.35 3.97 9.01 0.99
Task 1 2 3 4 5 6
Identify task without precedence
Select task with the highest x_i
Sequence 1
Remove selected task and identify task without precedence
Select task with the highest x_i
Sequence 1 3
Remove selected task and identify task without precedence
Select task with the highest x_i
Sequence 1 3 5
Feasible assembly sequence
Sequence 1 3 5 2 4 6

Figure 4. Example of solution decoding.

In general practice, the optimisation problem is defined as minimisation problem, where the smaller the fitness value obtained, the better the solution will be. In this case, the maximisation problem is converted into minimisation by giving the negative sign in front of the fitness function. In ABC, only the fittest solution from the initial population is kept as x_{best}. However, in MABC, three top solutions are saved and known as x_α, x_β, and x_δ. The x_α, x_β, and x_δ represent the best, second best, and third best solutions, respectively. This mechanism is mimicking the leadership hierarchy in the grey wolf group [28].

Next, the fitness function is converted into the larger the better term. This fitness function for solution x_i is noted as fit_i and calculated as follows:

$$fit_i = \begin{cases} \dfrac{1}{1+f_i} & \text{if } f_i \geq 0 \\ 1 + abs(f_i) & \text{if } f_i < 0 \end{cases} \tag{15}$$

Employed Bee Phase

Later, the new solution, v_i will be generated from the existing solution, x_i. The v_i is generated using the following formula:

$$v_{i,j} = x_{i,j} + \Phi(x_{i,j} - x_{k,j}) \tag{16}$$

The $x_{i,j}$ refers to the existing *i*th solution for the *j*th dimension. Φ is a random number in the range [-1,1]. Meanwhile, x_k represents a solution from the existing population that is being chosen randomly. The new solution is later evaluated using the fitness function. The fitness value for v_i is compared with x_i. In this case, the greedy selection approach is used. If the fitness value for v_i is better than x_i, the v_i will replace the existing solution x_i. Otherwise, the existing solution x_i will remain.

Next, the probability value (p_i) for solution x_i is calculated using the following equation. In this equation, the better fitness of the solution will give a higher probability value.

$$P_i = \frac{fit_i}{\sum_{i=1}^{N_{swarm}/2} fit_i} \tag{17}$$

The solution now will be sorted starting from the highest p_i value. At this point, the solution is divided into two groups. The first group contains half of the NS, which consists of a solution with better p_i, known as x_{on}. This group will be used for the onlooker bee phase. Meanwhile, the remaining half of the solution (x_w) will undergo the grey wolf phase.

Beewolves Phase

Beewolf or scientifically known as *Philanthus triangulum* is a predator bee that targeting the employed bee as their prey [29]. In this work, the beewolves concept is adopted from grey wolf optimizer [28] to replaced onlooker bee phase in ABC algorithm. The beewolves try to hunt, attack and paralyse the employed bee. In the proposed algorithm, this mechanism is simulated by replacing a specific employed bee with a better performance bee.

The beewolves work in a group which consist of α, the most dominant beewolf, followed by β and δ, while the rest of the population is known as ω. The first step is conducted to roughly determine the position of the prey (i.e. employed bee). This is conducted using the following equations:

$$B = \left|c.x_p - x_w\right| \tag{18}$$

$$x_w = x_p - A.B \tag{19}$$

In this equation, *B* represents the distance between the prey (x_p) and the beewolves. Meanwhile, *A* and *c* are the coefficients that are calculated as follows:

$$A = 2a.r_1 - a \tag{20}$$

$$c = 2.r_2 \tag{21}$$

In this case, r_1 and r_2 are random numbers between [0, 1], while *a* is a component that linearly decreases from 2 to 0 over the iteration. However, since the actual position of the prey is unknown, the algorithm is relying on the x_α, x_β, and x_δ that have better knowledge about the prey. Therefore, in the hunting stage, the algorithm will calculate the average vector distance between the beewolves and the x_α, x_β and x_δ.

$$B_\alpha = |c_1.x_\alpha - x_w|;\ B_\beta = \left|c_2.x_\beta - x_w\right|;\ B_\delta = |c_3.x_\delta - x_w| \tag{22}$$

$$x_{w,1} = x_\alpha - a_1.(B_\alpha);\ x_{w,2} = x_\beta - a_2.\left(B_\beta\right); x_{w,3} = x_\delta - a_3.(B_\delta); \tag{23}$$

The algorithm will also generate a new solution from the vector distance.

$$v_w = \frac{x_{w,1} + x_{w,2} + x_{w,3}}{3} \tag{24}$$

The attacking prey step represents the exploitation of the solution. Since the A and c rely on the random numbers r_1 and r_2, the new position of beewolves (v_w) can be in any position from the current and the x_α, x_β, and x_δ. Furthermore, the component a is also decreasing over the iteration, which makes the search direction for this algorithm becomes more diverse.

The new solution v_w will be evaluated and compared with x_w. If the fitness function for the v_w is better, the new solution will replace x_w. At this point, if the existing solution does not show any improvement compared to the previous iteration, the limit counter for x_i (L_i) will be updated. However, if the solution has improved, the L_i will be reset to zero. For solution x_i with L_i larger than L, the scout bee procedure will be conducted.

Scout Bee Phase

The scout bee phase refers to the random regeneration of solution x_i. Theoretically, when a particular solution does not show any improvement for a given duration, the solution must be regenerated to ensure better exploitation in the search space. This procedure is conducted using the following equation:

$$x_{i,j} = lb_j + \text{rand}(0,1) \cdot (ub_j - lb_j) \tag{25}$$

The new solution is randomly generated within the lower bound (lb) and upper bound (ub). The newly generated solution will replace the solution x_i.

RESULTS AND DISCUSSIONS

A computational experiment was conducted to test the performance of the proposed algorithm. For this purpose, the authors have selected the 12 SALBP test problems from line balancing benchmark set [30]. The problem was categorised as small ($n \leq 40$), medium ($40 < n \leq 80$), and large ($n > 80$). Since the original SALBP data set only considers the assembly time information, the assembly direction and tool data for these problems are randomly generated.

For comparison purpose, the authors implemented the GA, ACO, and PSO algorithms because these algorithms were used in previous literature for integrated ASP and ALB optimisation. Besides that, the proposed MABC is also compared with the original ABC and GWO algorithms. For each of the algorithm, the population size is 20 and the maximum iteration is 300. To eliminate pseudo-random effect, the authors conducted 20 repetitions for the optimisation run for each of the test problems. This computational experiment was conducted using HP Z400 Workstation, Intel Xeon 3.00 GHz processor, and 8.00 GB RAM. The optimisation results are presented in Table 1. The bolded data represents the best result for each problem.

Table 1. Optimisation results for integrated ASP and ALB.

Test Problem	**Indicator**	**Algorithm**					
		GA	**ACO**	**PSO**	**ABC**	**GWO**	**MABC**
Mitchell (21 tasks)	Min Fit	0.3992	0.3667	0.3992	0.3692	0.3817	**0.3165**
	Max Fit	0.4367	0.3992	0.4267	0.3992	0.4242	**0.3282**
	Mean Fit	0.4082	0.3797	0.4097	0.3897	0.4082	**0.3258**
	SD Fit	0.0162	0.0156	0.0113	0.0137	0.0157	**0.0052**
Roszieg (25 tasks)	Min Fit	0.3969	0.3445	0.3669	0.3445	0.3561	**0.3133**
	Max Fit	0.4869	0.3769	0.4569	0.3669	0.4176	**0.3425**
	Mean Fit	0.4409	0.3644	0.4182	0.3599	0.3804	**0.3328**
	SD Fit	0.0391	0.0119	0.0365	0.**0096**	0.0236	0.0137
Sawyer (30 tasks)	Min Fit	0.6515	**0.5555**	0.6432	0.6221	0.6265	**0.5555**
	Max Fit	0.6682	**0.6348**	0.6765	**0.6348**	0.6598	**0.6348**
	Mean Fit	0.6598	0.6072	0.6531	0.6297	0.6481	**0.6051**
	SD Fit	0.0083	0.0325	0.0148	**0.0069**	0.0139	0.0328
Gunther (35 tasks)	Min Fit	0.6646	0.5855	0.6095	0.5855	0.5855	**0.5815**
	Max Fit	0.6726	**0.5935**	0.6646	0.6095	0.6966	0.6015
	Mean Fit	0.6662	0.5903	0.6433	0.5983	0.6301	**0.5895**
	SD Fit	**0.00357**	0.00438	0.0221	0.0091	0.0414	0.0079
Kilbridge (45 tasks)	Min Fit	0.6540	0.5578	0.5874	0.6312	0.6394	**0.4653**
	Max Fit	0.7614	**0.6255**	0.6828	0.6468	0.6685	0.6394
	Mean Fit	0.6885	**0.5879**	0.6418	0.6408	0.6570	0.5987
	SD Fit	0.0427	0.0321	0.0437	**0.0064**	0.0129	0.0747
Hahn (53 tasks)	Min Fit	0.6413	0.6265	0.6801	0.6318	0.6371	**0.5928**
	Max Fit	0.6693	0.6402	0.6853	0.6455	0.6800	**0.5993**
	Mean Fit	0.6593	0.6303	0.6836	0.6404	0.6575	**0.5954**
	SD Fit	0.0116	0.0059	**0.00239**	0.0059	0.0163	0.0030
Tonge (70 tasks)	Min Fit	0.5216	0.4642	0.4554	0.4984	0.4873	**0.4502**
	Max Fit	0.5307	0.5029	0.5062	0.5207	0.5237	**0.4591**
	Mean Fit	0.5247	0.4893	0.4888	0.5096	0.5049	**0.4535**
	SD Fit	0.0051	0.0217	0.0289	0.0111	0.0182	**0.0048**
Weemag (75 tasks)	Min Fit	0.5516	0.5831	0.5698	0.5867	0.5649	**0.5486**
	Max Fit	0.5734	0.5903	0.5867	0.5939	0.5903	**0.5644**
	Mean Fit	0.5633	0.5869	0.5806	0.5907	0.5790	**0.5579**
	SD Fit	0.0109	**0.0036**	0.0093	0.0036	0.0129	0.0082
Lutz2 (89 tasks)	Min Fit	0.6251	0.6041	0.6386	0.6149	0.6230	**0.5937**
	Max Fit	0.6535	0.6197	0.6589	0.6305	0.6494	**0.6072**
	Mean Fit	0.6355	0.6145	0.6463	0.6219	0.6343	**0.5999**

	SD Fit	0.0156	0.0090	0.0109	0.0079	0.0136	**0.0055**
Mukherjee (94 tasks)	Min Fit	0.5281	0.5301	0.5147	0.5405	**0.5069**	0.5163
	Max Fit	0.5512	0.5396	0.5347	0.5496	**0.5211**	0.5330
	Mean Fit	0.5359	0.5348	0.5242	0.5452	**0.5150**	0.5253
	SD Fit	0.0132	0.0047	0.0100	**0.0045**	0.00733	0.0084
Arc (111 tasks)	Min Fit	0.7484	0.7491	0.7849	0.8207	0.7598	**0.7422**
	Max Fit	**0.7795**	0.8199	0.8735	0.8655	0.8270	0.8089
	Mean Fit	**0.7649**	0.7786	0.8368	0.8454	0.7945	0.7693
	SD Fit	**0.0106**	0.0211	0.0272	0.0127	0.0214	0.0236
Bartholdi (148 tasks)	Min Fit	0.4705	0.4679	0.5312	0.4793	0.4743	**0.4627**
	Max Fit	0.5634	0.4969	0.6033	0.5085	0.5893	**0.4943**
	Mean Fit	0.5049	0.4851	0.5559	0.4908	0.5571	**0.4794**
	SD Fit	0.0433	0.0090	0.02372	0.0089	0.0296	**0.0086**

Based on Table 1, the proposed MABC consistently came out with better minimum fitness in small and medium size problems. The proposed algorithm also shows better mean fitness in all four small size problems. Besides that, the MABC performed better in maximum and mean fitness for medium size test problems in Hahn, Tonge and Weemag. For the large size problem, the proposed MABC obtained minimum fitness in three out of four test problems. In Mukherjee's test problem, the proposed algorithm was behind the GWO and PSO algorithms in terms of minimum and mean fitness.

According to Table 1, the smallest mean of standard deviation (SD) was found in ABC algorithm, followed by MABC and ACO. The SD values in the results showed that the MABC was among the algorithms that came out with a consistent output. On the other hand, the GWO and PSO were the algorithms with the largest mean of SD.

The overall performance from the numerical experiment showed that the MABC had better output compared to the comparison algorithms. The MABC found a better solution in 91.6% of the problems (11 out of 12 test problems) and better mean fitness in 75% of the problems from 20 runs. The results mean that in general, the MABC has a better performance compared to the comparison algorithms. To confirm the performance of the MABC, a one-way ANOVA test was conducted. The purpose of this test is to identify any significant differences between the mean values obtained using different algorithms.

For the ANOVA test, the following hypotheses are applied:

H_0: $\mu_{GA} = \mu_{ACO} = \mu_{PSO} = \mu_{ABC} = \mu_{GWO} = \mu_{MABC}$

H_1: The means are not all equal

The null hypothesis, H_0 stated that the means of the fitness for all algorithms are the same. While the alternative hypothesis, H_1 stated that there are differences in the means of fitness. For this test, the confidence interval was set at 0.05. The output of the ANOVA test is presented in Table 2. In this case, when the *P*-value is smaller than the confidence interval, the null hypothesis is rejected. According to the result in Table 2, all of the *P*-values were smaller than 0.05. Therefore, the null hypotheses for all test problems were rejected. In other words, the results showed that there were significant differences in the mean values of the groups. However, the ANOVA test did not specifically reveal the algorithm with a significant difference in the results.

Therefore, a *post hoc* analysis was conducted to identify the significant difference for the proposed MABC compared to other algorithms. For this purpose, the Fisher's least significant difference (LSD) test was conducted. The LSD is calculated using the following formula:

$$LSD = tc. \sqrt{MSW\left(\frac{1}{N_1} + \frac{1}{N_2}\right)} \tag{26}$$

In Equation. (26), *tc* refers to critical *t*-value from the *t*-distribution table, for 0.05 confidence interval and 114 degrees of freedom. *MSW* represents the mean square within the group, N_1 and N_2 are the numbers of sample data in the considered groups. Next, the absolute mean difference between MABC and the comparison algorithms was calculated. When the absolute mean difference is larger than LSD, there is a significant difference between MABC and the comparison algorithms. The results of the LSD test are presented in Table 2.

Table 2. Results of statistical tests

Test Problem	***P*-value**	***LSD***	**Mean Difference between MABC and comparison algorithms**				
			GA	**ACO**	**PSO**	**ABC**	**GWO**
Mitchell	2.486E-09	0.0085	0.0823	0.0538	0.0838	0.0638	0.0823
Roszieg	5.024E-06	0.0158	0.1080	0.0315	0.0853	0.0270	0.0475
Sawyer	7.359E-04	0.0132	0.0546	0.0021	0.0480	0.0245	0.0430
Gunther	4.132E-06	0.0124	0.0767	0.0008	0.0538	0.0088	0.0406
Kilbridge	0.0065700	0.0272	0.0898	0.0108	0.0430	0.0007	0.0582
Hahn	1.965E-12	0.0056	0.3194	0.0349	0.0882	0.0450	0.0621
Tonge	5.588E-03	0.0109	0.0712	0.0358	0.0353	0.0560	0.0514
WeeMag	3.810E-03	0.0055	0.0053	0.0289	0.0226	0.0328	0.0211
Lutz2	1.825E-03	0.0068	0.0379	0.0508	0.0487	0.0243	0.0367
Mukherjee	1.417E-02	0.0053	0.0105	0.0095	0.0011	0.0198	(0.0103)
Arc	3.153E-14	0.0127	0.0044	0.0093	0.0674	0.0760	0.0251
Barthold	3.809E-14	0.0152	0.0255	(0.0719)	0.0765	0.0114	0.0777

The bolded value in Table 2 shows that the MABC has a significant performance over the comparison algorithm. Meanwhile, the value in the bracket means that the comparison algorithm has a significant performance compared to MABC. Based on the LSD results in Table 2, no single algorithm is completely dominated by the MABC. However, the MABC has a significant performance in 91% of the problems compared to PSO and GWO. This is followed by 83% compared to GA. In comparison with ABC, the MABC significantly performed better in 75% of the problems and when compared with ACO, the MABC performed better in 58% of the problems.

To evaluate the algorithm performance in different problem sizes, the number of cases in which the MABC had a significant difference with the comparison algorithms within a particular problem size was calculated. For example, in small size problems (Mitchell, Roszieg, Sawyer and Gunther), the MABC had a significant difference in 17 out of 20 cases in all comparison algorithms. This makes the MABC have a significant performance in 85% of the cases. This percentage is also the same for medium size problems. However, when the problem size increases to a larger size, the percentage reduces to 70% of the cases. This trend is related with the size of the search space. When the size of the problem increases, the search space will decrease excessively.

The performance of MABC indicated that this algorithm has better exploitation ability. This is because the MABC is able to search for minimum fitness value in most of the test problems. The leadership hierarchy concept from the GWO was able to improve the performance of the proposed modified algorithm. The GWO in the original form however, was too dependent on the leaders to determine the search direction. This made the GWO have less freedom to explore the different angles in the search space. Meanwhile, in comparison with the original ABC, the MABC was able to speed up the convergence to an optimum solution. The search direction was guided by three leaders from the GWO, while maintaining the exploration features from ABC.

CONCLUSIONS

This paper proposed a modified algorithm based on the Artificial Bee Colony (ABC) by adopting the beewolf predatory concept to optimise the integrated ASP and ALB problem. This concept is originally implemented from leadership hierarchy mechanism of the Grey Wolf Optimiser (GWO). ABC in the original form has a drawback in terms of convergence due to its poor exploitation ability. Meanwhile, the GWO has a good leadership hierarchy mechanism, but has a high dependency on the leaders.

The proposed modified Artificial Bee Colony algorithm (MABC) was tested using a set of benchmark test problems and compared with five algorithms, including the ABC and GWO. The results indicated that the MABC is able to search for better minimum fitness in 91% of the benchmark test problems. A statistical test was conducted to confirm the significance of MABC performance compared to the comparison algorithms. The statistical test showed that the MABC has a significant performance in 80% of the cases, mostly in small and medium size problems.

The results from this work indicated that the exploitation ability in the ABC was improved by adopting the beewolf predatory concept. At the same time, the exploration ability in the ABC using the employed bee's concepts was maintained to make this modified algorithm to be not too dependent on the leaders. Finally, the proposed MABC indicated a balanced portion between exploration and exploitation abilities in swarm algorithm.

ACKNOWLEDGEMENT

The authors would like to acknowledge Universiti Malaysia Pahang for funding this research under grants PGRS1703107 and RDU190317.

REFERENCES

[1] Mastura M, Sapuan S, Mansor M. A framework for prioritizing customer requirements in product design: Incorporation of FAHP with AHP. Journal of Mechaical Engineerig and Sciences. 2015; 9: 1655–1670.

[2] Horng S-C, Lin S-S. Embedding advanced harmony search in ordinal optimization to maximize throughput rate of flow line. Arabian Journal for Science and Engineering. 2018; 43: 1015–1031.

[3] Marian RM. Optimisation of assembly sequences using genetic algorithm. University of South Australia, 2003.

[4] Andrew-Munot M, Yassin A, Shazali ST, et al. Analysis of production planning activities in remanufacturing system. Journal of Mechanical and Engineering Sciences. 2018; 12: 3548–3565.

[5] Abdullah MA, Ab. Rashid MFF, Ponnambalam SG, Ghazali, Z. Energy efficient modeling and optimization for assembly sequence planning using moth flame optimization. Assembly Automation. 2019; 39: 356–368.

[6] Rashid MFF, Hutabarat W, Tiwari A. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. Int Journal of Advance Manufacturing Technology. 2012; 59: 335–349.

[7] Akpinar S, Elmi A, Bektaş T. Combinatorial Benders cuts for assembly line balancing problems with setups. European Journal of Operational Research. 2017; 259: 527–537.

[8] Álvarez-Miranda E, Pereira J. On the complexity of assembly line balancing problems. Computer and Operations Research. 2019; 108: 182–186.

[9] Boysen N, Fliedner M, Scholl A. A classification of assembly line balancing problems. European Journal of Operational Research. 2007; 183: 674–693.

[10] Wang HS, Che ZH, Chiang CJ. A hybrid genetic algorithm for multi-objective product plan selection problem with ASP and ALB. Expert System with Applications. 2012; 39: 5440–5450.

[11] Ab Rashid MFF, Mohamed NMZN, Rose ANM, Kor KY. Simulation study of a vehicle production line for productivity improvement. Journal of Mechanical Engineering and Sciences. 2015; 8: 1283–1292.

[12] Lu C, Yang Z. Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach. International Journal of Advanced Manufacturing Technology. 2016; 83: 243–256.

[13] Ouaarab A, Ahiod B, Yang X-S. Discrete cuckoo search algorithm for the travelling salesman problem. Neural Computing and Applications. 2013; 24: 1659–1669.

[14] Ab Rashid MFF, Hutabarat W, Tiwari A. Multi-objective discrete particle swarm optimisation algorithm for integrated assembly sequence planning and assembly line balancing. Proceeding Inst. Mech. Eng. Part B: Journal of Engineering Manufacture. 2018; 232: 1444–1459.

[15] Tseng H-E, Tang C-E. A sequential consideration for assembly sequence planning and assembly line balancing using the connector concept. International Journal of Production Research. 2006; 44: 97–116.

[16] Tseng H-E, Chen M-H, Chang C-C, Wang W-P. Hybrid evolutionary multi-objective algorithms for integrating assembly sequence planning and assembly line balancing.

International Journal of Production Research. 2008; 46: 5951–5977.

[17] Chen R, Lu K, Yu S. A hybrid genetic algorithm approach on multi-objective of assembly planning problem. Engineering Applications of Artificial Intelligence. 2002; 15: 447–457.

[18] Yang Z, Lu C, Zhao HW. An Ant Colony Algorithm for Integrating Assembly Sequence Planning and Assembly Line Balancing. Applied Mechanics and Materials. 2013; 397–400: 2570–2573.

[19] Lutfy OF. Adaptive Direct Inverse Control Scheme Utilizing a Global Best Artificial Bee Colony to Control Nonlinear Systems. Arabian Journal for Science and Engineering. 2018; 43: 2873–2888.

[20] Liu F, Sun Y, Wang G, Wu T-T. An Artificial Bee Colony Algorithm Based on Dynamic Penalty and Lévy Flight for Constrained Optimization Problems. Arab Journal for Science and Engineering. Epub ahead of print January 2018.

[21] Kalayci C, Gupta S. Artificial Bee Colony Algorithm for Solving Sequence-dependent Disassembly Line Balancing Problem. Expert Systems with Applications. 2013; 40: 7231–7241.

[22] Li J-Q, Pan Q-K, Tasgetiren MF. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. Applied Mathematical Modelling. 2014; 38: 1111–1132.

[23] Zhang F, Li L, Liu J, Chu X. Artificial Bee Colony Optimization for Yard Truck Scheduling and Storage Allocation Problem. Springer, Cham, 908–917.

[24] Zhong F, Li H, Zhong S. An improved artificial bee colony algorithm with modified-neighborhood-based update operator and independent-inheriting-search strategy for global optimization. Engineering Application of Artificial Intelligence. 2017; 58: 134–156.

[25] Huang F, Wang L, Yang C. A new improved artificial bee colony algorithm for ship hull form optimization. Engineering Optimization. 2016; 48: 672–686.

[26] Battaïa O, Dolgui A. A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics. 2013; 142: 259–277.

[27] Karaboga D. An idea based on honey bee swarm for numerical optimization. Kayseri, Turkey, 2005.

[28] Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer. Adv Eng Softw 2014; 69: 46–61.

[29] Herzner G, Schmitt T, Linsenmair KE, Strohm E. Prey recognition by females of the European beewolf and its potential for a sensory trap. Animal Behaviour. 2005; 70: 1411–1418.

[30] Scholl A. Benchmark Data Sets by Scholl. Assembly Line Balancing Data Dets & Research Topics, http://assembly-line-balancing.mansci.de/salbp/benchmark-data-sets-1993/ (1993).