Computers & Industrial Engineering 76 (2014) 1–15

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Pareto based artificial bee colony algorithm for multi objective single model assembly line balancing with uncertain task times ☆

Ullah Saif [a,b], Zailin Guan [a], Weiqi Liu [a], Chaoyong Zhang [a,*], Baoxi Wang [a]

[a] State Key Lab of Digital Manufacturing Equipment and Technology, HUST-SANY Joint Laboratory of Advanced Manufacturing Technology, Huazhong University of Science and Technology, Wuhan 430074, Hubei, PR China
[b] Department of Industrial Engineering, University of Engineering and Technology, Taxila, Pakistan

A R T I C L E   I N F O

Article history:
Received 18 December 2012
Received in revised form 20 November 2013
Accepted 10 July 2014
Available online 18 July 2014

Keywords:
Single model assembly line balancing
Pareto solutions
Artificial bee colony algorithm
Uncertain task time
Taguchi method

A B S T R A C T

Assembly line balancing is significant for efficient and cost effective production of the products and is therefore gaining popularity in recent years. However, several uncertain events in assembly lines might causes variation in the task time and due to these variations there always remains a possibility that completion time of tasks might exceed the predefined cycle time. To hedge against this issue, a single model assembly line balancing problem with uncertain task times and multiple objectives is presented. Current research is aimed to minimize cycle time in addition to maximize the probability that completion time of tasks on stations will not exceed the cycle time and minimize smoothness index simultaneously. A Pareto based artificial bee colony algorithm is proposed to get Pareto solution of the multiple objectives. The proposed algorithm called Pareto based artificial bee colony algorithm (PBABC) introduces some extra steps i.e., sorting of food sources, niche technique and preserve some elitists in the standard artificial bee colony algorithm (ABC) to get Pareto solution. Furthermore, the effective parameters of the proposed algorithm are tuned using Taguchi method. Experiments are performed to solve standard assembly line balancing problems taken from operations research (OR) library. The performance of proposed PBABC algorithm is compared with a famous multi objective optimization algorithm NSGA II, in literature. Computational result shows that proposed PBABC algorithm outperforms NSGA II in terms of the quality of Pareto solutions and computational time.

1. Introduction

Assembly lines are flow oriented production systems used for cost efficient and mass production of products (Shtub & Dar-El, 1989). An assembly line consists of certain number of work stations located besides material handling system (e.g., on conveyer belt etc.). Assembly work pieces are moved down the assembly line from one station to another for different assembly operations. The problem of optimally distributing assembly work load among all stations with respect to certain objectives is called an assembly line balancing problem (ALBP) (Becker & Scholl, 2006).

In assembly environment, assembly of parts is divided into set of small number of operations. These operations are called tasks related to certain assembly product and there exists precedence relation among different tasks. These precedence relations among tasks are used to define appropriate priority of performing certain tasks relative to other tasks in the assembly operation of product. Assembly line balancing problems are mostly focused to identify feasible line balance which can satisfy all the precedence constraints and some other restrictions which may include some of the objectives of the problem. Different types of assembly line problems have been studied in literature. The well known and mostly studied one is simple assembly line balancing problem (SALBP). Most of the SALBP considers certain assumptions for example, a single homogeneous product is assumed to produce on the assembly line, task times are assumed as constant and no variations can occur in their tasks times etc. However, these assumptions most probably are violated in actual environment. Therefore, recent research is focused to eliminate or reduce these assumptions. The assembly line problems in which one or more of the assumptions of SALBP are eliminated lies under the category

☆ This manuscript was processed by Area Editor Surendra M. Gupta.
* Corresponding author at: State Key Lab of Digital Manufacturing Equipment and Technology, HUST-SANY Joint Laboratory of Advanced Manufacturing Technology, Huazhong University of Science and Technology, Wuhan 430074, Hubei, PR China. Tel.: +8613429882087.
E-mail addresses: saifullah47@yahoo.com (U. Saif), zlguan@hust.edu.cn (Z. Guan), lwqi2010@gmail.com (W. Liu), zcyhust@mail.hust.edu.cn (C. Zhang), wbxjulia@163.com (B. Wang).

http://dx.doi.org/10.1016/j.cie.2014.07.009
0360-8352/© 2014 Elsevier Ltd. All rights reserved.

Abbreviations and acronyms

ALBP	Assembly linebalancing problem	*MG*	Maximum grade in the sorting list of non-dominated food sources
SALBP	Simple assembly line balancing problem	*X*	a limit value on normally distributed number
GALBP	General assembly line balancing problem	σ	variance value of x
ABC	Artificial bee colony	μ	mean value of x
PBABC	Pareto based artificial bee colony	n_j	number of tasks assigned to station j
GD	Generational distance	r,x_1,x_2	indexes used to represent a food source
Nee	Nectar amount of food source	G_r	grade of food source r
n	number of tasks	Num_r	number of food sources of grade G
m	number of stations	$\delta(r,q)$	shared function between food source r and q
i	index used to represent a task	$d(r,q)$	Euclidean distance between food source r and q
j	index used to represent a station	ε	niche radius
t_i	random value of task time of task i	Z_v^r and Z_v^q	objective function values of food source r and q
t_i^μ	mean value of task time of task i	h	number of non dominated vectors
t_i^σ	variance value of task time of task i	d_v	Euclidean distance b/w each non dominated vector and nearest member of true Pareto front
CT	cycle time	$\left[C_i^j\right]^\mu$	mean value of completion time of task i on station j
S_j	set of tasks assigned to station j	$\left[C_i^j\right]^\sigma$	variance value of completion time of task i on station j
C_i^j	completion time of task i, assigned to station j		
t_k^j	task time of task k, assigned to station j		
x	*a* normally distributed number		

of general assembly line balancing problem (GALBP). Boysen, Fliedner, and Scholl (2007) classified assembly lines as single model, mixed model and multi model assembly lines based on the type of product assembled in them. Single model assembly line which can assemble one kind of product is considered in the current research. Assembly lines are balanced based on different objective functions among which the most famous objectives includes minimization of cycle time (Chica, Cordón, & Damas, 2011; Hamta, Fatemi Ghomi, Jolai, & Akbarpour Shirazi, 2013), minimization of number of workstations (Chica et al., 2011; Ponnambalam, Aravindan, & Mogileeswar Naidu, 2000), maximization of line efficiency (Wei & Chao, 2011), minimization of smoothness index (Ponnambalam et al., 2000; Nourmohammadi & Zandieh, 2011), minimization of design cost (Cakir, Altiparmak, & Dengiz, 2011) and maximization of system utilization (Mcmullen & Tarasewich, 2006) etc.

In real environment of assembly lines, two or more objectives are significantly desired to achieve simultaneously. In most of the times, these objectives can be conflicting and performance of each objective might not be improved without sacrificing the performance of at least one. Therefore, solution of multi objective problems exists in the form of a trade off between objectives. Furthermore, multi objective problems are more likely towards real situation and therefore, optimization of multi objectives in ALBP has been discussed by several researchers. For example, Kim, Kim, and Kim (1996), Malakooti and Kumar (1996), Askin and Zhou (1997), Gokcen and Erel (1997), Merengo, Nava, and Pozetti (1999), Ponnambalam et al. (2000), Chen, Lu, and Yu (2002), Vilarinho and Simaria (2002), Bukchin and Rubinovitz (2003), Mansouri (2005), Gamberini, Grassi, and Rimini (2006), Baykasoglu (2006), Nearchou (2008), Hwang, Katayama, and Gen (2008) and Nourmohammadi and Zandieh (2011) projected different solution approaches for multi objective optimization of ALBP.

Nevertheless, in most of the research, processing time of tasks is assumed as deterministic. On contrary, in real production environment, task times might not be deterministic and can varies randomly due to several uncertainties in the assembly line. These variations in task times might occur due to uncertain workers fatigue, low skill level, poorly maintained equipment, defects in the raw material etc (Shin, 1990). Therefore, multi objective optimization of stochastic assembly line which includes random task times is significant to investigate for assembly lines. A little effort has been paid in literature to describe multi objective stochastic assembly line balancing problem in which task times are considered as unknown variable (Cakir et al., 2011; Hamta et al., 2013; Mcmullen & Tarasewich, 2006).

Mcmullen and Tarasewich (2006) considered optimization of system utilization, design cost and probability of jobs to accomplish in a certain time limit in stochastic assembly line balancing problem. They developed a composite function (i.e., linear combination of the objectives) for optimization of multiple objectives. However, optimization of a composite function might not accurately decide which of the objective from all objectives of the problem is significantly optimized. This is because, solution of multi objective problems exists in the form of a set of alternative trade off solutions called Pareto optimal set. In Pareto set there is not only one solution but there are number of solutions which can give different values of the multiple objectives in each solution. The solution which can give most optimistic objective values of all objectives from these solutions are considered as better solution from all. In literature different optimization algorithms have been proposed to obtain Pareto optimal solution for multi objective problems (Agrawal, Dashora, Tiwari, & Son, 2008; Coello, Pulido, & Lechuga, 2004; Deb, Pratap, Agarwal, & Meyarivan, 2002; Kukkonen & Lampinen, 2009; Wang, Dang, Li, Han, & Wei, 2009; Yen & Leong, 2009; Zamuda, Brest, Boskovic, & Zumer, 2009; Zhang & Gen, 2011; Ayyuce & Orhan, 2013). Cakir et al. (2011) considered minimization of smoothness index and design cost in the stochastic assembly line balancing problem. They proposed a simulating annealing based algorithm to get Pareto optimal solutions. Zhang and Gen (2011) considered minimization of cycle time, minimization of variation in workload and minimization of workers cost simultaneously and proposed multi objective genetic algorithm to get Pareto solutions. Ayyuce and Orhan (2013) presented multi objective optimization of stochastic disassembly line. They considered minimization of number of stations and minimization of design cost associated with the labors and the equipment. Further, they proposed a new multi objective genetic algorithm to get Pareto solutions. Hamta et al. (2013) considered minimization of cycle time, total cost and smoothness index in stochastic assembly line balancing problem. They presented a hybrid particle swamp optimization algorithm for the optimization of multiple objectives. Several studies of multi objective optimization of assembly line problems has considered minimization of cycle time as one of their objective (Zhang & Gen, 2011; Chica et al., 2011; Nourmohammadi & Zandieh, 2011; Hamta et al., 2013). In stochastic assembly lines only minimization of cycle time may not be much sufficient because uncertain variations in the task times can also cause variations in the completion time of the tasks on different stations. Due to these variations, there remains a possibility that all the tasks assigned to the stations might not be accomplished in the expected cycle time. Therefore it might be more substantial to consider maximization of the probability to complete all tasks before an expected cycle time on a station along with minimization of cycle time, simultaneously, in stochastic assembly line balancing. Furthermore, smooth workload distribution in assembly lines can also be significantly essential to reduce workload variation on stations. Therefore, minimization of cycle time, maximization of probability to complete all tasks on stations before cycle time and minimization of smoothness index are desirable to be considered simultaneously in stochastic assembly line balancing. This motivates us to study these three objective simultaneously i.e., minimization of cycle time, maximization of probability of ensuring that tasks on stations can be accomplished before the cycle time and minimization of smoothness index in the stochastic assembly line balancing.

In literature, different approaches including nonlinear integer programming (Askin & Zhou, 1997), branch and bound algorithm (Bukchin & Rubinovitz, 2003), genetic algorithm (Ponnambalam et al., 2000; Zhang & Gen, 2011; Ayyuce & Orhan, 2013), simulated annealing (Baykasoglu, 2006) and particle swarm optimization (Hamta et al., 2013) has been proposed by researchers for assembly line balancing problem. In recent years, Karaboga (2005) proposed artificial bee colony (ABC) algorithm which is based on the simulating behavior of honey bee swarm intelligence. ABC algorithm is becoming popular because it needs less control parameters, easy to implement and can be used effectively for solving multi objective optimization problems (Akbari, Hedayatzadeh, Ziarati, & Hassanizadeh, 2012). However, in literature ABC has been mostly used to solve continuous problems (Akbari et al., 2012; Omkar, Senthilnath, Khandelwal, Narayana Naik, & Gopalakrishnan, 2011). A little effort has been paid to get Pareto optimal solution of multi objective optimization problems using ABC algorithm (Akbari et al., 2012; Omkar et al., 2011; Zou, Zhu, Chen, & Shen, 2011a, Zou, Zhu, Chen, & Zhang, 2011b). Recently, Li, Pan, and Gao (2011) introduced a hybrid Pareto based discrete version of artificial bee colony (P-DABC) algorithm for multi objective optimization. They proposed to combine local search with crossover operation in standard ABC algorithm to design P-DABC. They used local search in three different stages of their proposed algorithm, and due to more than one objective involved in the optimization, they used non dominating sorting after local search at different stages. The extra efforts of local search operations in many steps of their proposed algorithm and the requirement of non-dominated sorting after every local search may make P-DABC a little bit more complex and might needs larger efforts for computation. Furthermore, the P-DABC algorithm they presented is merely for flexible job shop scheduling which is quite different from the problem considered in our current study, which means that the compound local search and evolutionary approach they proposed may be effective mainly for the category of problems in their study. This motivates us to introduce a new Pareto based artificial bee colony (PBABC) algorithm for the multi objective optimization of our current assembly line problem. In order to obtain good quality solution along with the diversified solutions, the proposed PBABC algorithm is designed to include a nectar amount function which has characteristics of best Pareto solutions and the separation distance desired between different solutions which might be fruitful to diversify the search mechanism. The search process in our proposed PBABC algorithm can be expected to reduce in some extent the number of calculation steps.

Rest of the paper is organized as follows: Section 2 describes single model multi objective assembly line balancing problem. Section 3 illustrates multi objective algorithms and proposed Pareto based artificial bee colony algorithm. Section 4 describes tuning of the parameters of the proposed algorithm using Taguchi method (Taguchi, Chowdhury, & Wu, 2005). Section 5 illustrates computational experiments and results over several benchmark instances. Section 6 presents conclusion and some future aspects of the research.

2. Multi objective assembly line balancing problem

The most famous problem in assembly line balancing is the simple assembly line balancing problem (SALBP). However, due to lot of simplified assumptions taken in SALBP, it may not perform better in real environment (Boysen et al., 2007). For example, task times are considered as deterministic in some research (Arcus, 1966; Lau & Shtub, 1987). However, this assumption has been replaced in literature by introducing stochastic task times in assembly lines (Freeman & Jucker, 1967; Vrat & Virani, 1976, Subhash C. Sarin, Erel, & Dar-El, 1999). Some of the assumptions are also eliminated here by adding few of the realistic aspects of assembly line to form a multi objective assembly line balancing problem. The assumptions used are illustrated below:

- Single model assembly line balancing problem is considered here in which n number of tasks are assumed to process on m number of stations.
- There is no buffer in the assembly line and number of stations is known and constant.
- The tasks time t_i of tasks is taken as a random variable of normal distribution with mean t_i^{μ} and variance of t_i^{σ}.
- The setup time of tasks is included in their task times.
- All tasks are processed according to their predecessor constraints.
- Each task i is assigned to only one workstation j, processed once, and only one task is allowed to process on a single station at a time.

The current assembly line problem is aimed to divide all n tasks among m stations while optimizing the objectives illustrated in Eqs. (1)–(3).

$$Min(Z_1) = CT \tag{1}$$

$$Min(Z_2)_j = 1 - \left[\text{Minimum of}\left\{\frac{1}{n_j} \times \left(\sum_{i=1}^{n_j} \text{Probability}\left[C_i^j(S_j) \leqslant CT\right]\right)\right\}\right]$$
$$\forall\ \{j = 1,2,3,\ldots\ldots,m\} \tag{2}$$

$$Min(Z_3) = \sqrt{\sum_{j=1}^{m} [CT - t(S_j)]^2} \tag{3}$$

where, $t(S_j)$ is the station load on any station j and its value shown in Eq. (4).

$$t(S_j) = \sum_{k=1}^{n_j} t_k^j \tag{4}$$

The first objective shown in Eq. (1) is significant to reduce the cycle time of the assembly line, second objective indicated in Eq. (2) gives surety that tasks can accomplish before the cycle time in the workstation and third objective illustrated in Eq. (3) represents smoothness in workload distribution on different workstations in the assembly line.

From the assumptions used, task time is normally distributed therefore completion time of all tasks is also considered to be normally distributed. For any normally distributed number x, the probability that it will not exceed a limit value of X, expressed in standard normal distribution is indicated in Eqs. (5) and (6).

$$\text{Probability}(x \leqslant X) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-(t^2/2)} dt \tag{5}$$

$$\text{Probability}(x \leqslant X) = \frac{1}{2} + \varphi(z) \tag{6}$$

where, $z = \frac{x-\mu}{\sigma}$, and function $\varphi(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{-(t^2/2)} dt$. However, $\varphi(z)$ is difficult to solve for exact solution and an approximate technique can be used for its computation and therefore, Hayter (1996) illustrated an approximation method to calculate $\varphi\,(z)$ which is shown in Eq. (7). The second objective can be computed from the simultaneous solution of the relations shown in Eqs. (7)–(9).

$$\phi(z) \approx \varphi(z) = \begin{cases} 0.1z(4.4 - z) & (0 \leqslant z \leqslant 2.2) \\ 0.49 & (2.2 < z < 2.6) \\ 0.50 & (z \geqslant 2.6) \end{cases} \tag{7}$$

$$\text{Probability}\left[C_i^j(S_j) \leqslant CT\right] = \frac{1}{2} + \varphi(z) \tag{8}$$

$$z = \frac{CT - \left[C_i^j\right]^{\mu}}{\left[C_i^j\right]^{\sigma}} \tag{9}$$

3. Multi objective algorithm

In real environment more than one objective is desired to attain simultaneously and in many situations the considered multiple objectives in real systems are conflicting. For these conflicting objectives, optimization of one objective may result in unacceptable outcome for some other objectives. Multi objective problems do not have a single solution but contains a set of feasible solutions. The goal of multi objective optimization algorithms is to investigate this set of solutions (called Pareto set) and search out some best solutions from it. Different algorithms have been proposed in literature for multi objective optimization. The first multi objective genetic algorithm (GA) called vector evaluated genetic algorithm (VEGA) is proposed by Schaffer (1985). Afterwards, several multi objective algorithms have been developed including, multi objective genetic algorithm (MOGA) (Fonseca & Fleming, 1993), non-dominated sorting algorithm (NSGA) (Srinivas & Deb, 1994), strength Pareto evolutionary algorithm (SPEA) (Zitzler & Thiele, 1999) etc. Among these algorithms in literature, NSGA II (Deb et al., 2002) is a famous algorithm for multi objective optimization. The procedure of NSGA II algorithm is briefly explained and later proposed PBABC algorithm is described in this section.

4.1. NSGAII algorithm

NSGA II algorithm is proposed by Deb et al. (2002) for Pareto optimization of multi objectives. It is a population based algorithm. The main steps of NSGA II algorithm are presented below.

Step 1: Create random parent population of size N.
Step 2: Apply cross over and mutation to the solution of the parent population to create offspring population of size N.
Step 3: If the stopping criteria is satisfied, then algorithm is stopped.
Step 4: Add up the offspring population with parent population to make a combined population.
Step 5: In this step, non-dominated relation between solutions of the combined population is obtained. A solution x_1 is said to dominate another solution x_2 i.e., $x_1 \prec x_2$, if, solution x_1 is not worse than solution x_2 in all of its objective values. Further, x_1 is strictly better than x_2 in at least one of the objective value. Once the non-dominated relation between solutions is obtained, these solutions are sorted into different grades. Non-dominated sorting algorithm is used to identify the different grades $F_1, F_2, \ldots, F_R$ of the non-dominated solutions.
Step 6: In this step for each grade, the solutions in that grade are sorted for each objective function o in ascending order. Later crowding distance of each solution is obtained using following procedure. Suppose, $l = |F_x|$ and $S_{[k,o]}$ represent the k th solution in the sorted list with respect to the objective function o. Assign $cd_o(S_{[1,o]}) = \infty$ and for $cd_o\ (S_{[l,o]}) = \infty$ and for $i = 2, 3, \ldots, l-1$ assign

$$cd_o(S_{[i,o]}) = \frac{z_o(S_{[i+1,o]}) - z_o\left(S^o_{[i-1,o]}\right)}{z_o^{\max} - z_o^{\min}}$$

Sum the crowding distance $cd(S)$ of a solution S with respect to each objective i.e., $cd(S) = \sum_o cd_o(S)$.
Step 7: In this step the solution's crowding distances are compared. The crowded-comparison operator $\prec_n$ introduced by Deb et al. (2002) is used for selection and rejection of solution. In this method, every solution has two attributes, i.e., its new non-dominated grade p_{grade} and crowding distance $p_{distance}$. Then according to the crowded comparison operator, solution p and q are related according to following relations

$$p \prec_n q \text{ if}, (p_{grade} < q_{grade})$$
$$\text{Or}, [(p_{rank} = q_{grade}) \text{ and } (p_{distance} > q_{distance})$$

Step 8: In this step binary tournament selection based on the crowd comparison method is used to select parent solution for the next generation of parents and again the procedure is repeated from Step 2.

4.2. Pareto based artificial bee colony algorithm

Artificial bee colony algorithm (ABC) proposed by Karaboga (2005) induces intelligent behavior of honeybee swarms for optimization problems. ABC algorithm is a simple algorithm, requires less control parameters and is easy to implement which makes it popular to solve different optimization problems in recent years (Li et al., 2011). ABC algorithm consists of three kinds of bees called employed bees, onlooker bees and scout bees. The steps of standard ABC algorithm are described in Fig. 1. The standard ABC algorithm is significant for single objective optimization. However, for multi objective optimization, Pareto concepts are desired to be added in it. The proposed algorithm called Pareto based artificial bee colony algorithm (PBABC) introduces some extra steps i.e., sorting of food sources, niche technique and preserves some elitists, to get Pareto solutions. The flowchart of PBABC is shown in Fig. 2 and its step by step procedure is presented in this section.

Step 1: Create initial population of food sources
Step 2: Send employed bees to the food sources and extract their nectar amounts
Step 3: Using onlooker bees, select the best food sources and memorize them
Step 4: Send the scouts to search for new food sources
Step 5: If termination occurs, Stop, otherwise go to step 2

Fig. 1. Steps of standard ABC algorithm.

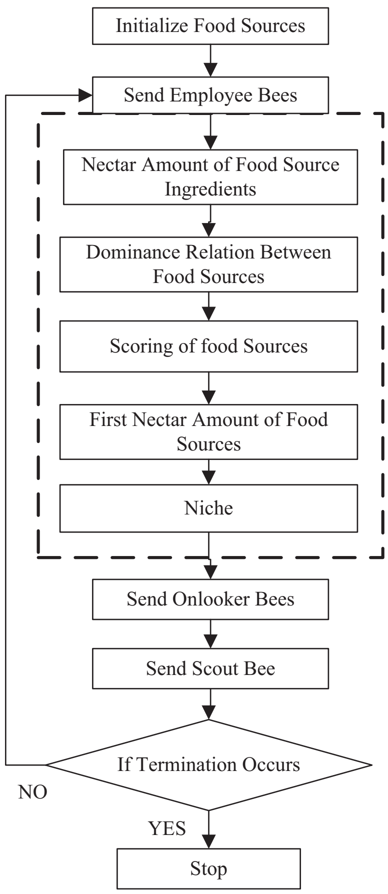

Fig. 2. Flowchart of the proposed PBABC algorithm.

3.1. Initializing food sources

In PBABC algorithm, each food source describes a solution and is represented by a vector. The number of elements in the vector indicates total number of tasks in the assembly line and the numbers shown in these elements indicate the stations assigned to the tasks. For example, a food source shown in Fig. 3 represents a solution of seven different tasks assigned to six stations. Further, the number 3 located in the fifth element of the food source vector indicates that fifth task is assigned to station 3 in the assembly line. In order to introduce different kind of food sources, initial food sources are randomly generated in the proposed PBABC algorithm.

3.2. Send employee bees

In standard ABC algorithm, employed bees investigate the nectar amount of food sources assigned to them. However, due to more than one objective, the nectar amount of a food source depends on values of all objective functions. Therefore, in PBABC algorithm, employed bees searches the food sources which can give best Pareto solutions. Due to more than one objective, nectar amount of a food source is considered as combination of nectar amounts of food source ingredients. The number of food source ingredients is equal to the number of objectives considered and they defines the value of each objective. Following steps are followed to investigate nectar amount of a food source:

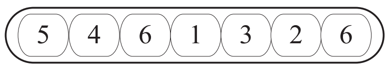

Fig. 3. A food source representing stations assigned to the tasks.

3.2.1. Nectar amount of food source ingredients

In PBABC algorithm, the employed bees compute nectar amount of all food source ingredients. The nectar amounts of the ingredients of a food source r are obtained from Eqs. (10)–(12) respectively.

$$(Nec_1)_r = (CT)_r \tag{10}$$

$$(Nec_2)_r = 1 - \left[\text{Minimum of}\left\{\frac{1}{n_j} \times \left(\sum_{i=1}^{n_j} \text{Probability}\left[C_i^j(S_j) \leqslant CT\right]\right)\right\}\right]_r \quad \forall\ \{j = 1,2,3,\ldots..,m\} \tag{11}$$

$$(Nec_3)_r = \left[\sqrt{\sum_{j=1}^{m} [CT - t(S_j)]^2}\right]_r \tag{12}$$

3.2.2. Dominance relation between food sources

The food sources containing three number of food source ingredients due to three objectives in the current problem. The comparison of different food sources can be performed by obtaining dominance relation between different food sources. The nectar amount of each food source ingredient is significant to determine the relative dominance between all food sources. A food source x_1 is said to dominate another food source x_2 i.e., $x_1 \prec x_2$, if, food source x_1 is not worse than food source x_2 in all of its food source ingredients. Further, x_1 is strictly better than x_2 in at least one of the food source ingredient.

3.2.3. Sorting of food sources

The food sources are sorted into sets of non-dominated food sources by assigning different grades to the solutions (Zitzler, Laumanns, & Bleuler, 2004). The step by step sorting process of food sources is shown in Fig. 4. In the first step, non-dominated food sources from all food sources are identified and defined as belonging to grade 1. In the next step, grade 1 food sources are deleted and non-dominated food sources are determined from the remaining food sources and defined as belonging to grade 2 food sources. Similarly, in the next step, grade 2 food sources are also deleted and non-dominated food sources from the remaining food sources are identified and defined as belonging to grade 3. This process is continued until all food sources got their grades. Then all food sources are sorted according to their grades. The food sources containing smaller grade values are considered as best ones.

3.2.4. Compute first nectar amount of food source

The nectar amount of a food source which includes nectar amount of all its ingredients is obtained from Eq. (13) which is similar in relation to the fitness of an individual in multi objective genetic algorithm (Che & Chiang, 2010). However, this nectar amount is called as first nectar amount in the proposed PBABC algorithm. It can be seen from the Eq. (13) that first nectar amount of food source includes grade value of a food source and also includes the number of food sources in different grades. From Eq. (13), first nectar amount of a food source r is better if it has lower grade value. Furthermore, the first nectar amount of food source r is better if there is less number of food sources in higher grades. This may increase the possibility that food source can get better value of first nectar amount if there would be less number of non-dominating food sources at higher grades. So, higher nectar amount of a food source can give advantage and may help to give the direction to search the food sources near lower grade non-dominated food sources in the next search cycles of the algorithm.

$$\text{Nec}_r = \frac{\text{MG} - G_r + 1}{\sum_{x=1}^{\text{MG}} x \times \text{Num}_x} \tag{13}$$

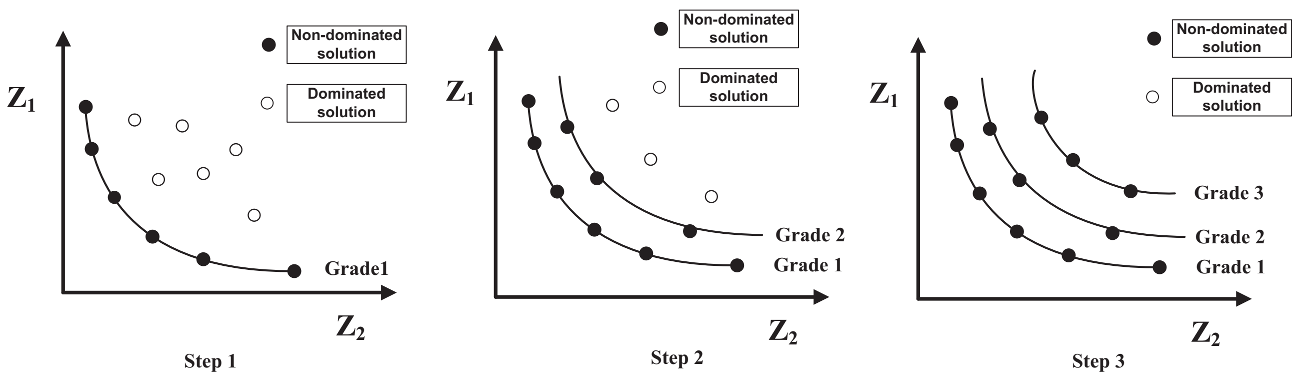

Fig. 4. Step by Step sorting process of food sources.

Table 1
Effective parameters of the proposed algorithm.

Levels	Number of food sources	Maximum number of algorithm cycles	Limit number of cycles for food sources
1	200	400	10
2	300	600	15
3	400	800	20
4	500	1000	25
5	600	1200	30

Table 2
Orthogonal array (OA) for Taguchi design of experiments for the proposed algorithm

Experiment	Level of parameters		
	Population size or number of food sources	Maximum number of algorithm cycles	Limit number of cycles for food sources
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
5	1	5	5
6	2	1	2
7	2	2	3
8	2	3	4
9	2	4	5
10	2	5	1
11	3	1	3
12	3	2	4
13	3	3	5
14	3	4	1
15	3	5	2
16	4	1	4
17	4	2	5
18	4	3	1
19	4	4	2
20	4	5	3
21	5	1	5
22	5	2	1
23	5	3	2
24	5	4	3
25	5	5	4

Table 3
Standard assembly line balancing problems are their category according to their size

Size of problems	Number of tasks
Small	
Buxey	29
Sawyer30	30
Lutz1	32
Low medium	
Gunther	35
Kilbrid	45
Hahn	53
Warnecke	58
Medium	
Tonge70	70
Wee-Mag	75
Arc 83	83
High medium	
Lutz2	89
Lutz3	89
Mukherje	94
Large	
ARC 111	111
Barthol2	148
BartholD	148
Scholl	297

3.2.5. Niche

During search of optimal solution, there is possibility that solution may trap in the local optima. In order to reduce this possibility, Che and Chiang (2010), proposed a new objective function for Pareto optimization of multi objective problems. Their proposed function is treated as final nectar amount of the food source in PBABC and is used here in the selection process of food sources, as described in Eq. (14).

$$Nec^*(r) = \begin{cases} Nec^r & \sum_{t=1}^{o} \delta(r, q_t) \leqslant 1 \\ \frac{Nec^r}{\sum_{t=1}^{o} \delta(r, q_t)} & \sum_{t=1}^{o} \delta(r, q_t) > 1 \end{cases} \tag{14}$$

Where,

$$\delta(r, q) = \begin{cases} 0 & d(r, q) \geqslant \varepsilon \\ 1 - d(r, q)/\varepsilon & d(r, q) < \varepsilon \end{cases} \tag{15}$$

$$d(r, q) = \sqrt{\sum_{v=1}^{h} [Z_v^r - Z_v^q]^2} \tag{16}$$

The value of niche radius ε is fixed by the user at some estimate of the minimum separation desired or expected between two food sources (Horn, Nafpliotis, & Goldberg, 1994). It can be seen from Eq. (14) that final nectar amount of food source depends both on the first nectar amount and niche radius. The niche radius can help to increase the diversity in the PBABC and first nectar amount can help to improve and direct the search process towards better food sources. These features of exploring better food sources and diversity in food sources included in final nectar amount function may

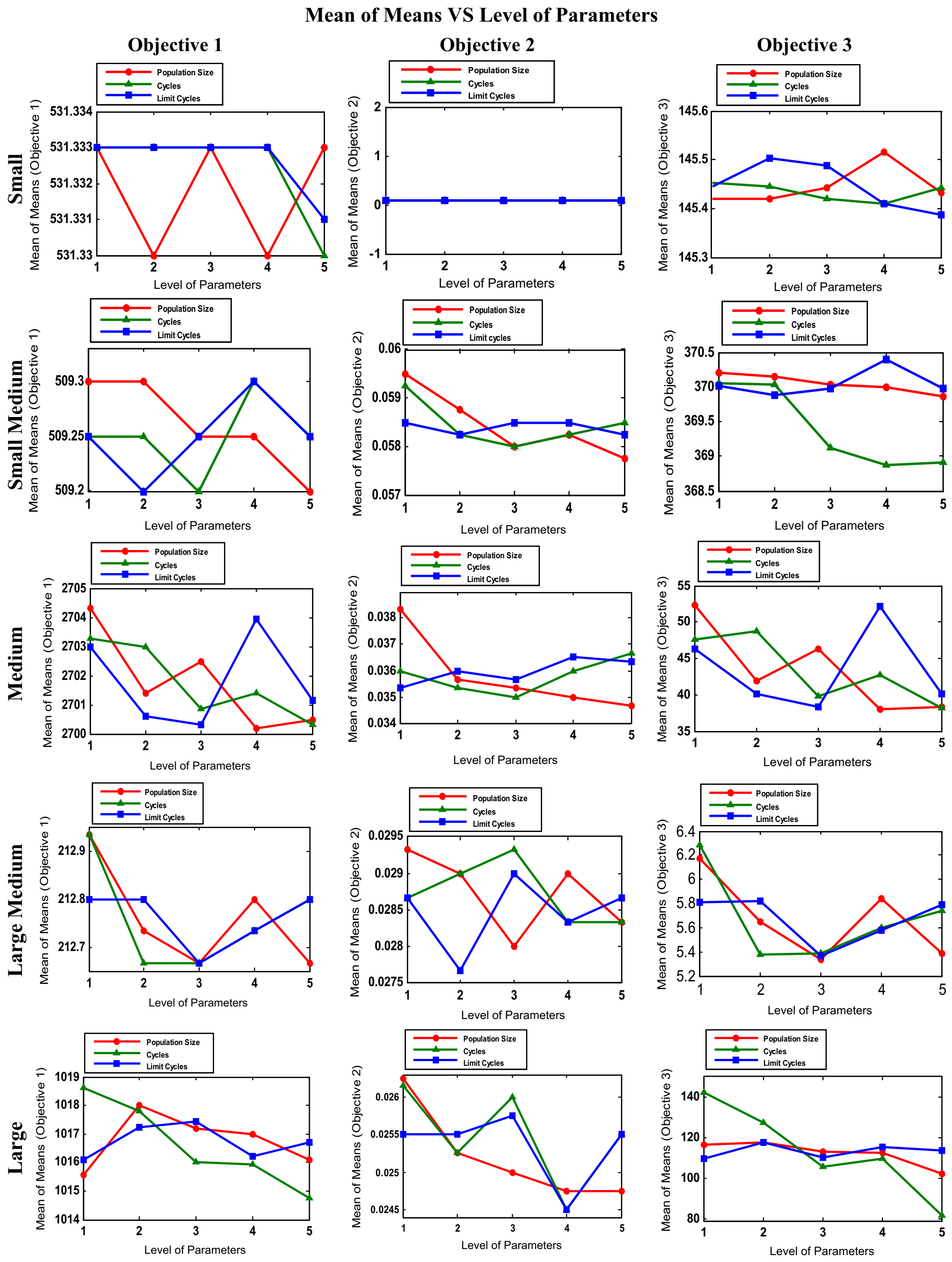

Fig. 5. Mean value of means for different level of parameters.

reduce the calculation process in the algorithm to search better food sources and explore a diversified search space.

3.3. Onlooker bees

In ABC algorithm, onlooker bees select the food sources on the basis of their selection probability or on the basis of the percent of nectar amount of each food source among the total nectar amount. Thus the food sources which have higher final nectar amount have more probability to be selected. Therefore, there is possibility that solution may fall in local optima. Furthermore, above approach may take more computational time to evaluate the nectar amount of each food source. Therefore, tournament selection is used in PBABC algorithm which computes the nectar amount of few food sources during selection process. In tournament selection, two food sources are randomly taken and tournament among these two randomly selected food sources is performed and final nectar amount difference between them is

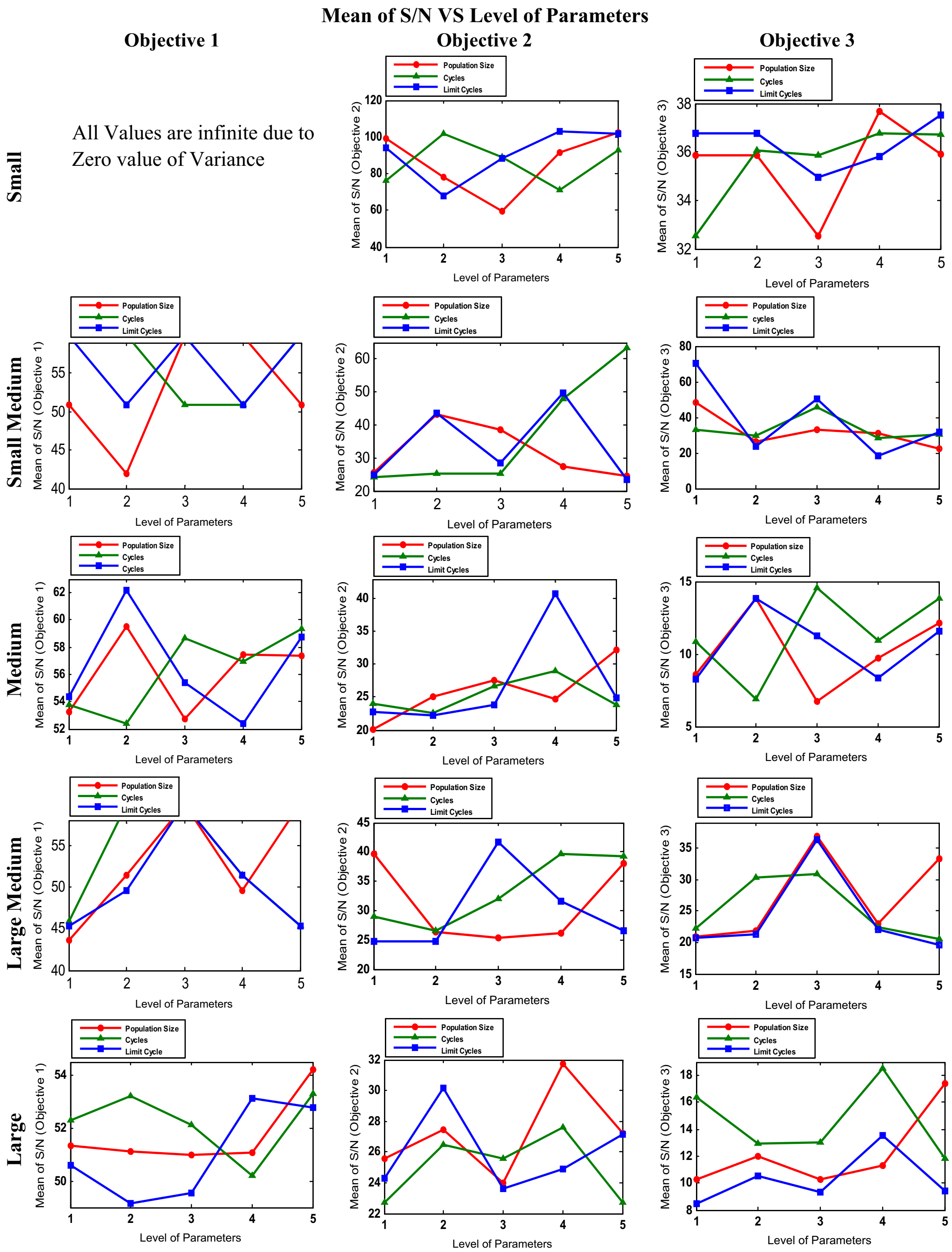

Fig. 6. Mean value of S/N for different level of parameters.

Table 4
Optimum level of parameters of proposed PBABC for each category of problem

Problem category	Level of parameter		
	Population size	Cycles	Limit cycles
Small size problem	2	5	5
Small medium size problem	3	3	2
Medium size problem	5	3	2
Large medium size problem	5	3	3
Large size problem	5	4	4

found. This difference generates the selection pressure and forces the algorithm to select the food sources containing higher nectar amount. The food source containing higher value of final nectar amount is selected from the randomly taken food sources.

3.4. Scout bees

In PBABC algorithm, when no improvement in the final nectar amount of food source occurs, tournament selection process is performed to find new food sources using scout bees.

Table 5
Parameter values for PBABC and NSGA II algorithm for different benchmark problems

Problem category	Number of stations	Parameters				
		PBABC			NSGA II	
		Population size	Cycles	Limit cycles	Population size	Number of generation
Small						
Buxey	10	300	1200	30	500	1600
Sawyer30	10	300	1200	30	500	1600
Lutz1	10	300	1200	30	500	1600
Small medium						
Gunther	10	400	800	15	600	1200
Kilbrid	10	400	800	15	600	1200
Hahn	10	400	800	15	600	1200
Warnecke	10	400	800	15	600	1200
Medium						
Tonge70	10	600	800	15	800	1200
Wee-Mag	10	600	800	15	800	1200
Arc 83	10	600	800	15	800	1200
Large medium						
Lutz2	10	600	800	20	800	1200
Lutz3	10	600	800	20	800	1200
Mukherje	10	600	800	20	800	1200
Large size						
ARC 111	10	600	1000	25	800	1400
Barthol2	30	600	1000	25	800	1400
BartholD	10	600	1000	25	800	1400
Scholl	30	600	1000	25	800	1400

Table 6
Variance in task times of the some benchmark problems, from the assumed interval of $\left[\frac{1}{8}t_i^\mu, \frac{1}{4}t_i^\mu\right]$

Benchmark problem									
Buxey		Hahn				Warnecke			
Task	Variance	Task	Variance	Task	Variance	Task	Variance	Task	Variance
1	1.64815	1	187.535	30	14.2082	1	2.4448	30	3.48779
2	4.26024	2	29.298	31	15.49	2	10.5688	31	11.3825
3	3.36169	3	21.6513	32	27.8894	3	8.26597	32	6.59147
4	1.17225	4	18.4271	33	34.6492	4	8.56836	33	5.31565
5	1.62035	5	25.1818	34	11.4989	5	4.5638	34	4.43573
6	1.91164	6	20.7875	35	11.0025	6	4.15344	35	2.77683
7	1.22883	7	16.9602	36	10.5375	7	6.9455	36	7.87431
8	3.56517	8	208.928	37	20.3682	8	4.75563	37	1.88536
9	0.321734	9	31.2107	38	53.2282	9	1.77921	38	5.347
10	1.05721	10	41.8908	39	10.7197	10	8.87072	39	8.18513
11	3.24762	11	15.5888	40	24.1096	11	5.79743	40	1.72399
12	1.69809	12	9.07475	41	12.9206	12	5.74821	41	3.53828
13	1.24877	13	22.1025	42	271.935	13	2.74854	42	3.14933
14	0.613636	14	71.3469	43	13.0646	14	8.73167	43	4.3879
15	1.9334	15	247.798	44	22.0081	15	2.57775	44	5.21881
16	1.09274	16	44.6916	45	18.1348	16	4.4999	45	8.78943
17	2.7892	17	34.7601	46	91.044	17	7.22384	46	4.12188
18	2.98922	18	20.8398	47	24.776	18	1.42269	47	3.83444
19	1.73387	19	142.927	48	13.2771	19	2.34319	48	2.17694
20	3.60363	20	11.7624	49	16.8999	20	2.74639	49	2.32936
21	0.143952	21	18.3444	50	11.8767	21	7.05839	50	1.59252
22	2.06477	22	14.5562	51	5.82995	22	6.48364	51	5.04357
23	4.81029	23	17.1324	52	158.51	23	10.6353	52	1.83161
24	3.14735	24	22.1885	53	199.329	24	3.07435	53	8.09192
25	2.62337	25	17.7714			25	5.70537	54	1.84031
26	0.422254	26	69.468			26	2.12331	55	4.8539
27	1.95708	27	20.3718			27	3.18583	56	10.8436
28	0.989532	28	9.99928			28	2.48527	57	1.01453
29	4.36384	29	19.3018			29	6.24301	58	3.49164

4. Tuning of proposed algorithm parameters with Taguchi method

The performance of algorithms can be judged from the quality of results it can produce as well as the degree of variation in these results it can give. Taguchi method (Taguchi et al., 2005) is significant to improve the quality of the proposed algorithm by minimizing the effects of causes which can produce variation in the results without eliminating those causes. This method is efficient to optimize proposed algorithm parameters by performing some

Table 7
Assembly line balancing results for the selected benchmark problem instance.

Benchmark instant	Tasks on station									
	St 1	St 2	St 3	St 4	St 5	St 6	St 7	St 8	St 9	St 10
Buxey	1,7,2	6,9,10,3	4,5,8	12,13, 15	19,21, 10	14,17, 20	16,18,22	23,28	24,25,26	27,29
Sayerr 30	2,1,3	16,12,5,6	13,14,20	4,7,15	8,9,21	10,22, 23	24,25,26	27,17	29,28,18	19,30,11
Lutz1	4	3,1,5	2,6,9, 11	10,7,8,12	13,14, 16	15,17, 19	20,18,21, 22	23,24, 25	26,27	28,29,30, 31,32
Gunther	1,5,6	2,3,17, 10,8,7	4,11,14	15,12	18,9,13	16,19	20,21,25, 22	23,26, 24,27	34,28,29,30	31,32,33, 35
Kilbrid	11,1,7,2,12,137	433,8,13,15,16	14,29,31,30,17,27	39,3,32,25,18,19	23,2	33,34, 35,4,6, 20	21	22,28,5	9,10,36, 26	38,40,41, 42,44,45
Hahn	1,4,5,7,2,6	8,9,10,11	12,15	3,13,16,18,17,19	21,24, 14,22, 25,26, 27,28, 33	32,31, 30,34, 29,35, 20,23, 36,37, 40,38, 39,41	42	44,49, 43,48, 47,46	50,45,51,52	53
Warnecke	12,28,7,8,5,6	1,9,11,15,16,2	33,35, 14,13, 17	3,32,20,25,21	44,18, 19,24, 26,29, 22	27,30,34,23,31	36,38,37, 41,39	42,41, 43,45, 46,47, 49	48,51,52,53,54,56	50,55,57, 58,4,10
Tonge70	9,1,41,15,16,2	10,3, 70,5,4,18	11,69,6,7,8	17,12, 14,68, 19,20	13,57, 22,58, 59,30, 21,23	24,25, 29,26, 27,28	31,32,33, 34,35,60, 48	62,63, 64,44, 45,56	36,37,51,52,38,39, 53,40, 54,42,43,66,67	49,61,65, 55,46,47, 50
Wee-Mag	1,5,12,6,10,2,11	9,4,8,15,3, 20,27	36,35, 23,40, 43,19, 46	48,42, 47,50, 49,61, 53	58,24, 25,28, 52,57, 38	59,55, 66,64,7,68,69, 70	73,65,74,34,29,60,62,67	16,21, 63,72, 17,13, 71,14	54,18,26,22,32,44,75,31	39,33,41, 37,30,51, 45,56
Arc 83	1,2,5,4,7	3,6,10,13,8	9,17,25,14,19,24,22, 29	30,28, 11,15, 20,31, 32,33, 34	36,38, 35,39, 44,26, 18,12	43,21, 27,42, 41,16, 37,40, 23,46, 48	45,47,49, 69,71,50, 51,52,53, 54,55,56, 70	72,73, 57,58, 59,60, 61,62, 63,64, 65	66,67,68,74,75,76,78,77,79	80,82,81, 83
Lutz2	1,2,3,4,7,5,8,9,10	6,11, 12,13,14,15,17,18,19,22	23,20, 28,24, 25,29, 31,32, 27	33,21, 26,34, 35,37, 36,38, 39	43,45, 46,47, 41,49, 16,42, 50	51,40, 44,30, 54,52, 55,56, 57	63,64,62, 59,61,58, 53,69,60	66,65, 67,70, 71,73, 74,75, 68	72,80,77,78,79,76,81,84	85,82,83, 86,87,89, 88,48
Lutz 3	1,2,30,3,4,5,,7,6, 11,12,13	8,9,10,14,15,17,18,19	22,23, 24,25, 26,20, 21,16	54,55, 56,57, 60,63, 64,59, 28	558,62,29,31, 32,27, 33,34	35,66, 65,61, 37,36, 38	39,41,43, 45,46,47, 49,67,42, 40,44	50,51, 52,53, 71,69, 73,74, 75	77,76,70,68,72,78,80,79,81,85	48,82,83, 84,86,87, 89,88
Mukherje	1,9,10,3,7,2,5,4	16,6, 15,11,14,25,63,24,37	73,68, 67,32, 50,69, 70,65, 57,27, 62	56,55, 31,44, 26,8,29,45,47,48,46, 23	34,13, 30,36, 42,59, 21,53	28,35, 54,51, 43,33, 40,17, 22,49, 58,38, 72	60,39,41, 64,74,66, 18,12,52, 61,77	20,78, 79,80, 81,71, 76,19, 75	82,83,85,84,86,87,88,89	92,94,90, 91,93
ARC 111	1,2,3,4,10,11,12, 18,9, 20,8	13,14,22,25,19,30,6,17, 28,23,31,33	34,32, 41,36, 44,16, 27,35, 42	47,55, 29,37, 45,59, 67,62, 38,61	57,65, 58,66, 54,24, 70,50,5,39	46,40, 56,64, 72,52,7,74,76	80,81,43, 87,90,63, 71,15,26	82,69, 60,68, 73,75, 77,83, 51,49, 79,85, 78	84,89,91,93,92,94,48,53,95	97,99,88, 96,102, 104,98, 103,106, 108,110, 109,21, 101,105, 86,107, 100,111
BartholD	32,1, 59,42,43,64,65, 66,11,93,50,51,69,79	2,85, 80,81,94,33, 58,44,52,53,144	145, 138, 139, 140,12,13,35, 99,100,30,141	60,61, 62,63, 142, 148, 147,3,4,8,74, 75	132,67,68,98, 95,101,102, 103,127,10,6,9,5, 91,105, 119	146,56,143,7, 14,15, 16,17, 18,19, 20,21, 22,24, 23,28, 92,34, 73	88,86,87, 25,26,29, 31,36,37, 38,39,40, 41,45,46, 96,104	47,54, 89,76, 55,90, 111,112,113,116,114, 49,57, 117,120,121, 118,122	115,123,128,129,77,82,83,84, 106, 107,108	126,48, 130,131, 137,133, 109,110, 70,97,124,125,72, 134,135, 136,71
Scholl										
	St 1	St 2	St 3	St 4	St 5	St 6	St 7	St 8	St 9	St 10
	1,2,3,4,134,27,109,111	116, 122,129,141,151,22,24,25	259, 266,56,61,65, 94,136,29,121,128,60	68,73,5,6,10,9,7,8,13	18,11, 16,139,247,33,38,41, 253,48,221	223,225,40,44, 142,152,164,49,53,15	21,20,86, 93,105,110,162,167, 171,31,172,175,180	252,258,265, 272,275,280, 179,45,260,267,273	82,88,89,83,90,95, 26,30,34,35	42,276,281,50,54,58,46,138, 191,51,81,87,55,59
	St 11	St 12	St 13	St 14	St 15	St 16	St 17	St 18	St 19	St 20
	64,72,78,79,125, 80, 100, 104, 99, 103, 108, 114, 119, 115, 120	127, 157, 150, 161, 292, 297, 12,17	14,290,296, 192, 201, 85,92, 98,102	107, 113, 118,126, 291,19,23,28, 32,37	36,39, 43,47, 52,57, 62,66, 69	74,76, 63,67, 70,75, 71,77, 97	84,91,96, 101,106, 112,117, 124,123, 146	149,160,130,144,148, 159,145,155, 156,257,264,154,147, 158	131,132,133,135,140, 143,169,153,165,176	200,168, 173,177, 163,166, 170,174, 288,287, 178,181
	St 21	St 22	St 23	St 24	St 25	St 26	St 27	St 28	St 29	St 30
	187, 196, 185, 188, 197, 295, 183, 193	182, 198, 184, 194, 189, 186, 190, 195, 227, 199, 205, 203, 208	206, 209, 211, 230, 289, 271, 131, 229	236, 239, 279, 202, 204, 207, 210, 213, 212, 214, 234	238,285,215, 216,217,218, 286,235,219	220,222,224, 226,228,231, 233,237,240	243,241, 242,244, 246,255, 245,	262,248,249, 251,284,250, 256,254	261,268,269,263,294, 270,274,278	282,277, 283,293
*Barthol*2										
	St 1	St 2	St 3	St 4	St 5	St 6	St 7	St 8	St 9	St 10
	61,56,58,73,138	50,62,94,88,11,63	96,144,145	92,51, 69,70, 132	79,80, 85,64	59,81, 141,142	12,146,60,143	147,86,33,93	1,35,2,32,74,75	139,140, 30,104,65
	St 11	St 12	St 13	St 14	St 15	St 16	St 17	St 18	St 19	St 20
	148, 71,3, 6,5	4,8,7,42	43,52,44,9	87,10, 14,16	15,17, 19,18, 20,23, 24	21,53, 22,26, 27,91, 105	25,99,66, 67,119	100,28,29,31, 34	36,37,97,68,95,98	101,102, 45
	St 21	St 22	St 23	St 24	St 25	St 26	St 27	St 28	St 29	St 30
	103,46,47, 38	39,40,41,54,76,90	111, 112, 57,55, 72,48	127, 134,77,78,82, 133	135,83,113,116,117	118,120,121, 122,128,114	129,130, 131,123, 124,84,106	136,115,125,49	126,107,108	109,110, 89,137,13

designed set of experiments. Taguchi method is used to define a set of experiments containing different level (different values) of the parameters. In this method an orthogonal array (OA) of different level of parameters is used to determine signal to noise ratio (S/N) for each set of parameter levels. Each level corresponds to some value of the parameter. OA defines the level of parameters in a matrix in which the number of columns represents number of parameters and number of rows defines the number of experiments. Each row defines a set of values of different parameters for certain level for an experiment. S/N is used to define the ratio of the value of objective function with the variance value of the objective function. Taguchi experiment is designed to determine the best level of parameter for each parameter at which the S/N is maximum i.e., quality of optimizing objective is better with less variance.

In the first step of this method, different parameters which can influence the performance of algorithm are identified and certain level for each value of parameter is assigned to them. For example, in the current study, the parameters which might influence the quality of the results of ABC algorithm are the size of population of the food sources, the number of cycles of the algorithm and the limit number of cycles after which if no improvement can be observed from the employee bee and it becomes scout bee (Li et al., 2011). These three parameters are named here as population size, cycles and limit cycles respectively. Table 1 illustrates the proposed effective algorithm parameters with their different levels for the proposed PBABC algorithm. It can be seen from Table 1 that each column of the table indicates different values of the parameters and their corresponding levels for each value. For example, level 1 represents that the number of food sources in the experiment is 200, maximum number of cycles of the algorithm to run is 400 and the limit cycles is 10 after which an employee bee can become a scout bee if there is no improvement in its nectar value.

In the current study, five levels for each parameter are considered and for each level, 5 numbers of experiments are performed. The current set of experiments is used to test 25 experiments containing five levels and three different parameters. Table 2 indicates orthogonal array (OA) for the Taguchi design of experiments for the proposed algorithm. In Table 2, each row indicates the level of factors which are considered in each experiment and each column represent a specific level for that parameter and it has different value in each experiment.

In Taguchi design of experiments both controllable and uncontrollable factors which might affect the quality of solutions are considered. The parameters and their levels can be controlled but some other factors called noise factors which are out of control. These factors cause variation in the performance from its target values and are identified by the Taguchi method without removing these factors from the proposed algorithm. Therefore, repeated experiments are performed in different parameter settings to note down the effect of noise factors in different parameter settings.

The proposed PBABC is tested on standard benchmark of assembly line balancing problems and these problems are categorized into different sizes on the basis of the number of tasks in these problems. This categorization of benchmark problems into different sizes might be significant to accurately measure noise (or variation in the quality of results) in each category of problems. The proposed category of the benchmark problems is indicated in Table 3.

In the current experiment, each benchmark problem is tested according to different level of parameter as mentioned in the proposed OA and the corresponding values of the three objective functions are computed. Once each benchmark is tested according to OA set of parameters, the mean value of the objectives, for each level of each parameter is computed for each benchmark. For example, the mean value of objectives for the parameter 'population size' is obtained from first five experiments of the OA matrix because they can give mean value of objectives at level 1 for parameter 'population size'. Similarly, mean value for parameter 'population size' for level 2 is obtained by taking the average of objective values obtained from the next five experiments. Similar procedure is adopted to get mean of each parameter for each level. Later mean of mean objective values (called mean of means) for each level of each category of problems is computed. Furthermore, the corresponding value of signal to noise ratio (S/N) for each objective is also computed using the relation shown in Eq. (17)

$$(S/N)_{\text{Nominal}} = 10\log\left(\frac{(mean)^2}{(Variance)^2}\right) \tag{17}$$

Where, $(mean)^2$ indicates the mean value of the optimizing objective and $(Variance)^2$ is the variance value in the optimizing objectives. S/N values of different problems for each objective are computed according to the OA and then mean value of S/N of each objective for each level of parameter is computed. Later, mean value of S/N values (called mean of S/N) for each level of each category of problems is computed. The mean value of means and mean value of S/N for each category of problem size for each objective are indicated graphically in Figs. 5 and 6 respectively. Graphical method is employed here to identify the specific level of different parameters for each category of the benchmark problems.

In the current case objective functions are the minimizing objectives so the level of parameter which gives small value of the optimizing objectives is preferred. Furthermore, the level of parameter at which maximum value of S/N is obtained is preferred. The optimum level of parameter for each category of benchmark problem is obtained by observing both mean value of means and mean of S/N values of all objectives for each category of benchmark problem. The optimum level of parameters obtained after observing the mean of means and mean of S/N values for each category of benchmark problem are illustrated in Table 4.

5. Computational experiments and results

In this section, performance of the proposed PBABC algorithm is tested using the optimum level of parameters obtained in the previous section. The standard assembly line balancing benchmark (Scholl, 1993, 1999) taken from operations library (OR) are analyzed using both PBABC and NSGA II algorithm. One instance of each benchmark is tested in the experiments. The parameters used for NSGA II are obtained from different run of experiments and the parameters values which can give good results for NSGA II are

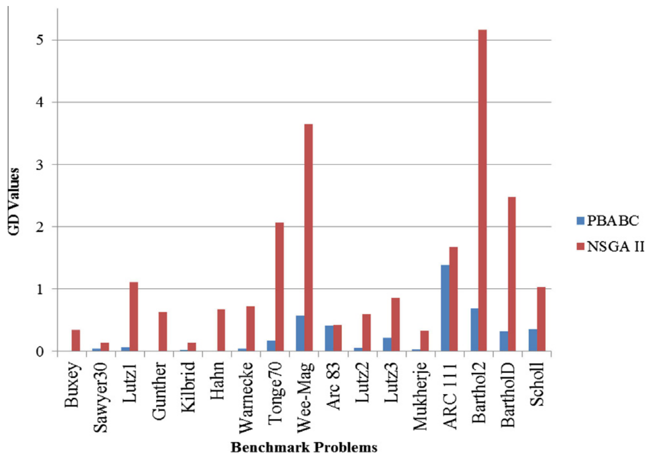

Fig. 7. Comparison of GD values of PBABC and NSGAII for selected benchmarks.

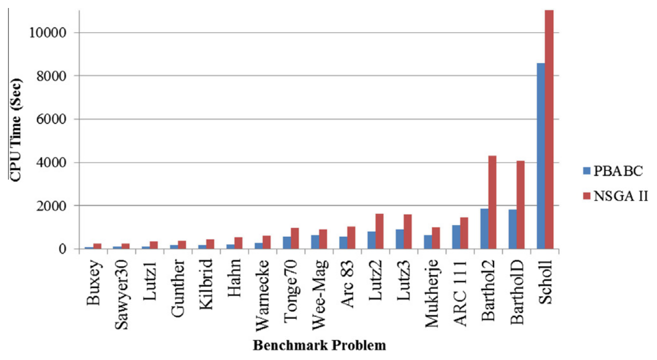

Fig. 8. Comparison of CPU time (Sec) for PBABC and NSGAII for selected benchmarks.

selected for it. The selected values of parameters of both PBABC and NSGA II algorithm for the selected benchmark instances are illustrated in Table 5. The performance of PBABC algorithm is compared with a famous algorithm NSGA II (Deb et al., 2002) to solve one instance from each standard assembly line balancing problem. Both PBABC and NSGA II are coded in Visual C++ for analysis. Since the task times available in this data only have mean values t_i^{μ}, so the variance t_i^{σ} in task times are estimated from an assumed interval of $\left[\frac{1}{8}t_i^{\mu}, \frac{1}{4}t_i^{\mu}\right]$. The randomly obtained variance t_i^{σ} values in task time of the some benchmark problems are summarized as a reference in Table 6.

The assembly line balancing results which can describe the tasks assigned to different stations for the selected benchmark problem instance are illustrated in Table 7. These results indicate only one solution from the set of Pareto solution for reference.

5.1. Comparison of results

5.1.1. Inverted generational distance

The current problem has three objectives, so the results of all benchmark problems form both PBABC and NSGA II algorithms are in the form of Pareto fronts. Therefore, inverted generational distance (GD) concept given by Coello and Cortes (2005) is used

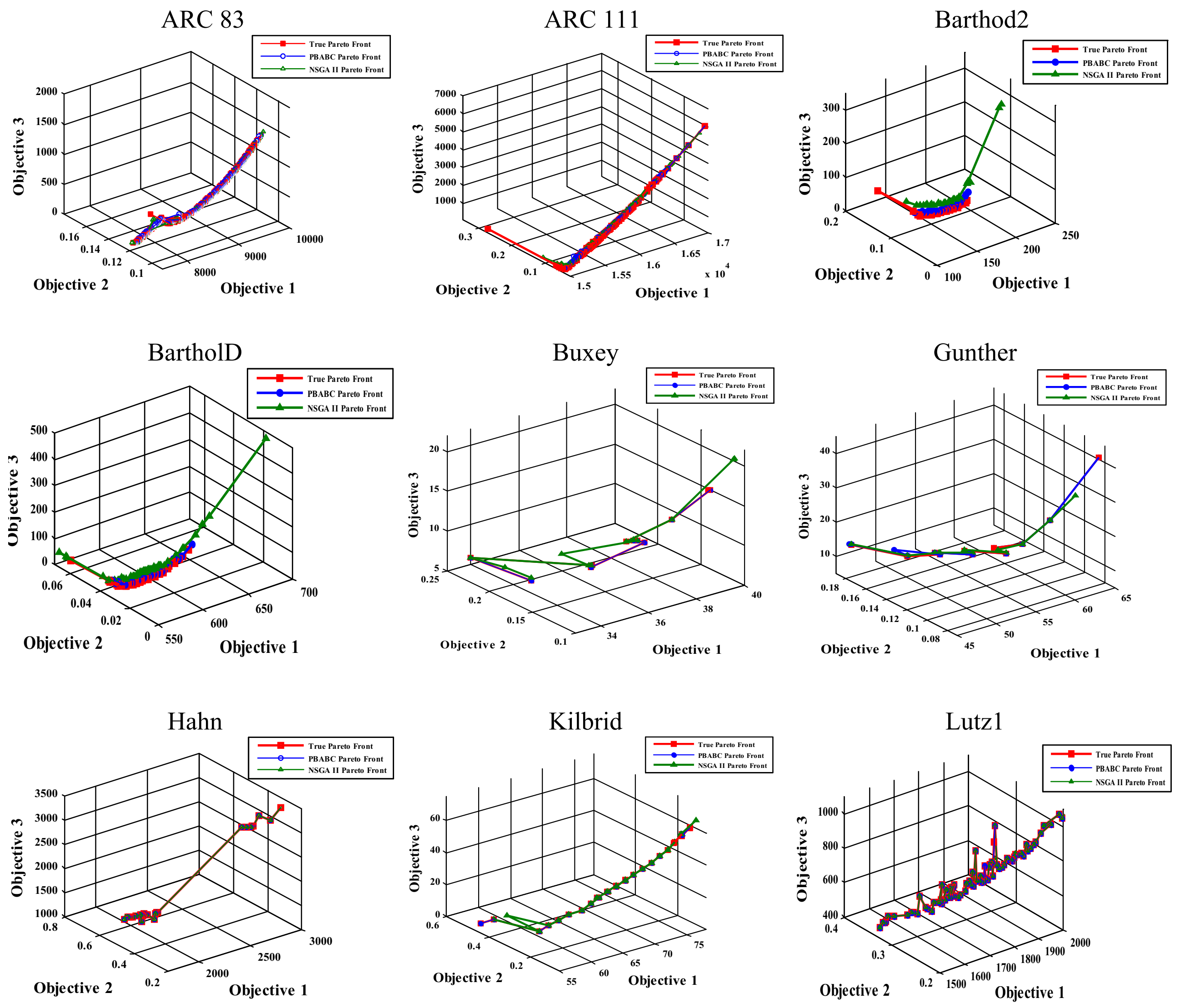

Fig. 9. Comparison of Pareto Fronts of PBABC and NSGAII for selected benchmarks.

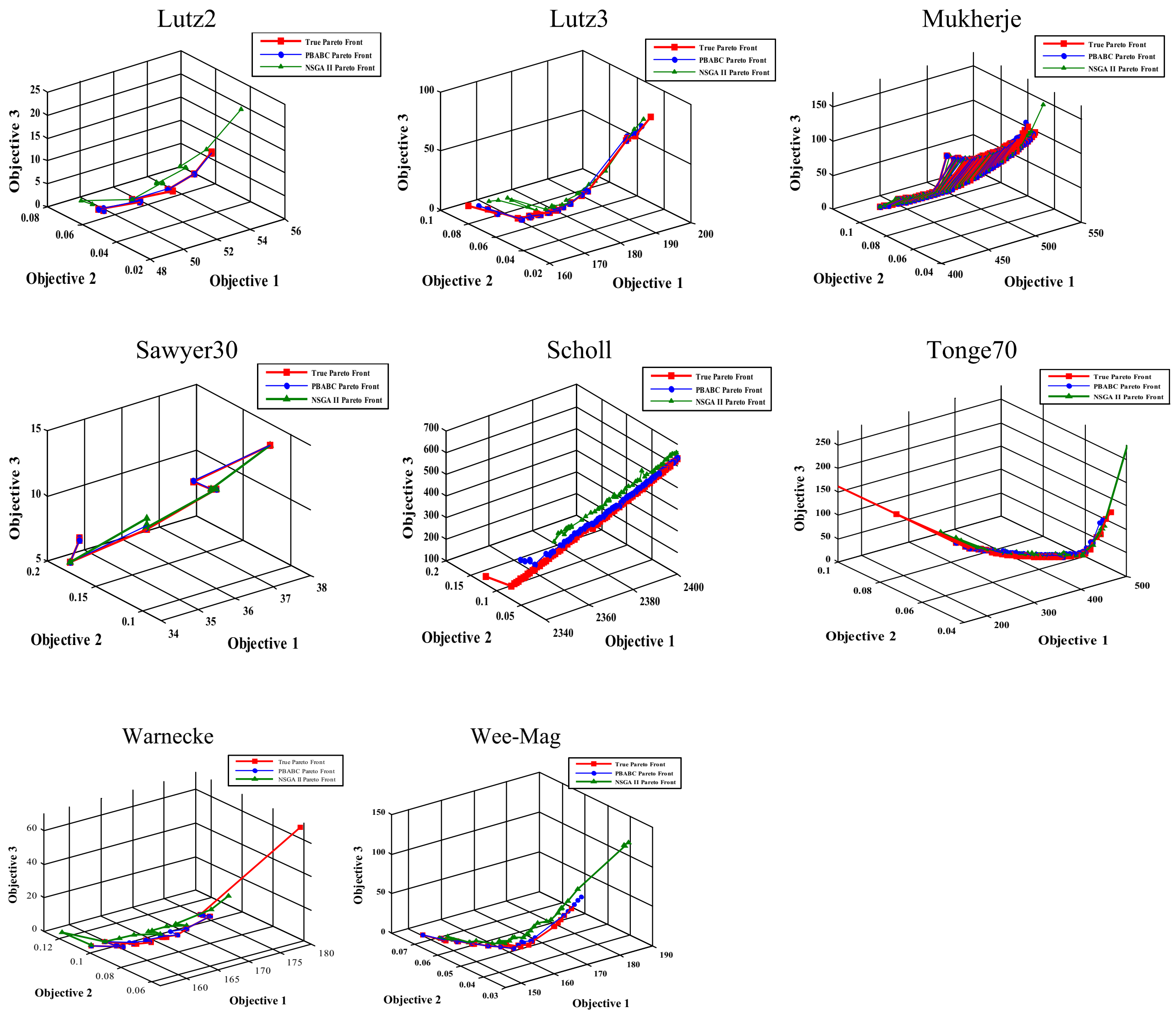

Fig. 9 (*continued*)

to estimate the elements distance of the Pareto solutions of PBABC and NSGA II algorithms from the true Pareto front to investigate the performances of the PBABC and NSGA II. The value of GD is computed from the relation indicated in Eq. (17).

$$\mathrm{GD} = \frac{\sqrt{\sum_{v=1}^{h} d_v^2}}{h} \tag{17}$$

The smaller value of GD indicates that Pareto solution is closer towards true Pareto front. The comparison of the PBABC and NSGA II on the basis of GD value for the selected benchmark is shown in Fig. 7. It can be seen from Fig. 7 that, the GD value obtained from PBABC, for different benchmark problems are smaller than that obtained from NSGAII. For instance, benchmark Barthol2 gives maximum difference in GD value between proposed PBABC and NSGA II. Different benchmark problems indicate that GD values obtained from PBABC algorithm are significantly smaller than GD values of NSGA II algorithm. These results suggest that proposed algorithm can give Pareto results which are neater to the true Pareto front and PBABC out performs NSGA II in terms of the quality of Pareto solutions.

5.1.2. Computation time

The performance of PBABC and NSGA II results on the basis of computation time is also significant to compare to know the significance of algorithms. The comparison of CPU time (Second) which PBABC and NSGA II algorithm takes to solve the selected instance of benchmark problems is shown in Fig. 8. It can be seen from Fig. 8 that PBABC algorithm takes less computation time to solve the selected instance of each benchmark problem as compared to NSGA II. These results illustrate the significance of the proposed PBABC algorithm on NSGA II on the basis of computational time.

5.1.3. Pareto fronts

The performance of proposed PBABC and NSGA II algorithm on the basis of their Pareto fronts for the selected instance of benchmark problems is illustrated in Fig. 9. It can be seen from Fig. 9 that in different benchmark problems, the Pareto fronts generated by the proposed PBABC algorithm are nearer to the true Pareto front. As, the distance of Pareto points of the PBABC front is very small from the true Pareto front so graphically these results might not be so clear to observe but the GD values as presented before can give clear picture of the significance of the Pareto results of the PBABC algorithm.

Table 8
Pareto solution obtained from PBABC and NSGA II for each benchmark problem instance.

Benchmark selected instance	PBABC results Objective function values			NSGA II results Objective function values		
	1	2	3	1	2	3
Buxey	34	0.179398	6.63325	34	0.179398	6.9282
Sayerr 30	34	0.166669	8.12404	34	0.176435	5.83095
Lutz1	1526	0.339909	424.603	1526	0.339909	424.603
Gunther	50	0.214082	6.245	50	0.214082	6.245
Kilbrid	56	0.467216	3.16228	57	0.434654	6.16441
Hahn	1775	0.5	1466.16	1775	0.5	1466.17
Warnecke	156	0.100006	4.24264	156	0.100006	4.69042
Tonge70	353	0.0826904	9.16515	354	0.089551	13.0384
Wee-Mag	150	0.0734977	1	151	0.0676603	4.12311
Arc 83	7598	0.119385	125.774	7592	0.12192	88.4929
Lutz2	49	0.0568609	3	49	0.063261	2.64575
Lutz 3	166	0.0740908	6.48074	167	0.082857	10.9545
Mukherje	424	0.102345	11.1355	424	0.102345	11.1355
ARC 111	15073	0.0558314	138.337	15077	0.0810657	140.253
Barthol2	144	0.126564	19.8494	146	0.148725	32.0624
BartholD	566	0.042379	10.198	565	0.070923	8.3666
Scholl	2350	0.0865729	221.908	2361	0.0667335	335.113

One Pareto solution from Pareto set obtained from PBABC and NSGA II algorithm for each selected benchmark problem instance is presented in Table 8 for reference.

6. Conclusion

Assembly line balancing has got lot of attentions in recent years because it is significant for initial and running cost of the production system. In most of the assembly lines or other production environment, processing time or assembly time of tasks is assumed as deterministic or constant in most of the literature. However, in real production environment, task times may not be constant and varies randomly due to several uncertainties in the assembly line. These variations in task times may occur due to uncertain workers fatigue, low skill level, poorly maintained equipment, defects in the raw material etc. Due to these variations completion time of tasks in stations also varies and there always remains a possibility that the completion time of tasks might exceed the predefined cycle time on some stations. In order to overcome this issue, a single model assembly line which considers task time as normal distributed variable, is presented here. Furthermore, multi objective optimization is significant in actual environment and therefore, current research is aimed to minimize cycle time in addition to maximize the probability that completion time of tasks will not exceed the cycle time and minimize the smoothness index simultaneously. Moreover, a Pareto based artificial bee colony algorithm is presented to get Pareto solution of the multiple objectives. The presented algorithm called Pareto based artificial bee colony algorithm (PBABC) introduces Pareto optimality concepts in the original artificial bee colony (ABC) algorithm.

The effective parameters of the proposed PBABC algorithm are tuned with robust experimental design procedure using Taguchi method. In this method different assembly line balancing benchmark problems are taken from OR library and are categorized according to their sizes. The proposed PBABC algorithm parameters are identified and tuned for each category of problem with Taguchi method. In order to investigate the performance of proposed PBABC algorithm, experiments are performed to solve standard assembly line benchmarks and the results of proposed PBABC algorithm are tested against the results of a famous multi objective algorithm NSGA II, in literature. Computational results showed that PBABC algorithm outperforms NSGA II in terms of the quality of Pareto results and computation. Future research can be extended to introduce hybrid algorithm of PBABC so as to improve its diversity mechanism for its better performance.

Acknowledgements

This work has been supported by MOST (the Ministry of Science & Technology of China) under the grants No. 2013AA040206, 2012BAF12B20, & 2012BAH08F04, and by the National Natural Science Foundation of China (Grants No. 51035001 & 71131004). We are very grateful for the valuable advices and suggestions given by Prof. Quan-Ke Pan of Huazhong University of Science and Technology, China and his colleagues on the comparison of our proposed algorithm with their algorithm presented in literature.

References

Agrawal, S., Dashora, Y., Tiwari, M. K., & Son, Y. J. (2008). Interactive particle swarm: A Pareto-adaptive metaheuristic to multi objective optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part A (Systems and Humans), 38*(2), 258–277.
Akbari, R., Hedayatzadeh, R., Ziarati, K., & Hassanizadeh, B. (2012). A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation, 2*, 39–52.
Arcus, AL. (1966). COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research, 4*(4), 259–277.
Askin, R. G., & Zhou, M. (1997). A parallel station heuristic for the mixed-model production line balancing problem. *International Journal of Production Research, 35*, 3095–3105.
Ayyuce, Aydemir-Karadag, & Orhan, Turkbey (2013). Multi-objective optimization of stochastic disassembly line balancing with station paralleling. *Computers and Industrial Engineering, 65*, 413–425.
Baykasoglu, A. (2006). Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. *Journal of Intelligent Manufacturing, 17*, 217–232.
Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*, 694–715.
Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research, 183*, 674–693.
Bukchin, J., & Rubinovitz, J. (2003). A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions, 35*, 73–85.
Cakir, B., Altiparmak, F., & Dengiz, B. (2011). Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers and Industrial Engineering, 60*, 376–384.
Che, Z. H., & Chiang, C. J. (2010). A modified Pareto genetic algorithm for multi-objective build-to-order supply chain planning with product assembly. *Advances in Engineering Software, 41*, 1011–1022.
Chen, R. S., Lu, K. Y., & Yu, S. C. (2002). A hybrid genetic algorithm approach on multi-objective of assembly planning problem. *Engineering Applications of Artificial Intelligence, 15*(5), 447–457.
Chica, M., Cordón, Ó., & Damas, S. (2011). An advanced multi objective genetic algorithm design for the time and space assembly line balancing problem. *Computers and Industrial Engineering, 61*, 103–117.

Coello, C. A. C., & Cortes, N. C. (2005). Solving multi objective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines, 6*, 163–190.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*(3), 256–279.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Fonseca, C.M., Fleming, P.J. (1993). Multi-objective genetic algorithms. In: IEE colloquium on genetic algorithms for control systems engineering, London, UK.

Freeman, D. R., & Jucker, J. V. (1967). The line balancing problem. *The Journal of Industrial Engineering, 18*(6), 361–364.

Gamberini, R., Grassi, A., & Rimini, B. (2006). A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. *International Journal of Production Economics, 102*(2), 226–243.

Gokcen, H., & Erel, E. (1997). A goal programming approach to mixed-model assembly line balancing problem. *International Journal of Production Economics, 48*, 177–185.

Hamta, N., Fatemi Ghomi, S. M. T., Jolai, F., & Akbarpour Shirazi, M. (2013). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics, 141*, 99–111.

Hayter, A. J. (1996). *Probability and statistics for engineers and scientists*. PWB Publishing Company.

Horn, J., Nafpliotis, N., Goldberg, D.E. (1994). A niched Pareto Genetic Algorithm for multi objective optimization, In: Proceedings of first IEEE conference on evolutionary computation, 1, 82–87.

Hwang, R. K., Katayama, H., & Gen, M. (2008). U-shaped assembly line balancing problem with genetic algorithm. *International Journal of Production Research, 46*(16), 4637–4649.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization, technical report TR06*. Turkey: Computer Engineering Department, Erciyes University.

Kim, Y. K., Kim, Y. J., & Kim, Y. (1996). Genetic algorithms for assembly line balancing with various objectives. *Computers and Industrial Engineering, 30*(3), 397–409.

Kukkonen, S., Lampinen, J. (2009). Performance assessment of generalized differential evolution with a given set of constrained multi-objective test problems. In: Congress on evolutionary computation proceedings (pp. 1943–1950). New Jersey: USA.

Lau, HS., & Shtub, A. (1987). An exploratory study on stopping a paced line when incompletion occur. *IIE Transactions, 19*(4), 463–467.

Li, Jun-Qing, Pan, Quan-Ke, & Gao, Kai-Zhou (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *International Journal of Advance Manufacturing Technology, 55*, 1159–1169.

Malakooti, B., & Kumar, A. (1996). A knowledge-based system for solving multi-objective assembly line balancing problems. *International Journal of Production Research, 34*(9), 2533–2552.

Mansouri, S. A. (2005). A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines. *European Journal of Operational Research, 167*(3), 696–716.

Mcmullen, P. R., & Tarasewich, P. (2006). Multi-objective assembly line balancing via a modified ant colony optimization technique. *International Journal of production Research, 44*, 27–42.

Merengo, C., Nava, F., & Pozetti, A. (1999). Balancing and sequencing manual mixed model assembly lines. *International Journal of Production Research, 37*, 2835–2860.

Nearchou, A. C. (2008). Multi-objective balancing of assembly lines by population heuristics. *International Journal of Production Research, 46*(8), 2275–2297.

Nourmohammadi, A., & Zandieh, M. (2011). Assembly line balancing by a new multi-objective differential evolution algorithm based on TOPSIS. *International Journal of Production Research, 49*, 2833–2855.

Omkar, S. N., Senthilnath, J., Khandelwal, R., Narayana Naik, G., & Gopalakrishnan, S. (2011). Artificial bee colony (ABC) for multi-objective design optimization of composite structures. *Journal of Applied Soft Computing, 11*(1), 489–499.

Ponnambalam, S. G., Aravindan, P., & Mogileeswar Naidu, G. (2000). A multi-objective genetic algorithm for solving assembly line balancing problem. *International Journal of Advanced Manufacturing Technology, 16*(5), 341–352.

Sarin, Subhash C., Erel, Erdal, & Dar-El, Ezey M. (1999). A methodology for solving single-model, stochastic assembly line balancing problem. *Omega International Journal of Management Science, 27*, 525–535.

Schaffer, J.D., (1985). Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the international conference on genetic algorithm and their applications.

Scholl, A., (1993). Data of assembly line balancing problems. Schriften zur Quantitativen Betriebswirtschaftslehre 16/93, TU Darmstadt.

Scholl, A. (1999). *Balancing and sequencing assembly lines* (2nd ed.). Heidelberg: Physica.

Shin, D. (1990). An efficient heuristic for solving stochastic assembly line balancing problems. *Computers and Industrial Engineering, 18*(3), 285–295.

Shtub, A., & Dar-El, E. M. (1989). A methodology for the selection of assembly systems. *International Journal of Production Research, 27*, 175–186.

Srinivas, N., & Deb, K. (1994). Multi-objective optimization using non-dominated sorting in genetic algorithms. *Journal of Evolutionary Computing, 2*(3), 221–248.

Taguchi, G., Chowdhury, S., & Wu, Y. (2005). *Taguchi quality engineering handbook*. New York: John Wiley and Sons.

Vilarinho, P. M., & Simaria, A. S. (2002). A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research, 40*, 1405–1420.

Vrat, P., & Virani, A. (1976). A cost model for optimal mix of balanced stochastic assembly line and the modular assembly system for a customer oriented production system. *International Journal of Production Research, 14*(4), 445–463.

Wang, Y., Dang, C., Li, H., Han, L., Wei, J. (2009). A clustering multi-objective evolutionary algorithm based on orthogonal and uniform design. In: Congress on evolutionary computation proceedings (pp. 2927–2933). New Jersey: USA.

Wei, N. C., & Chao, I. M. (2011). A solution procedure for type E-simple assembly line balancing problem. *Computers and Industrial Engineering, 61*, 824–830.

Yen, G. G., & Leong, W. F. (2009). Dynamic multiple swarms in multi objective particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part A (Systems and Humans), 39*(4), 1013–1027.

Zamuda, A., Brest, J., Boskovic, B., Zumer, V. (2009). Differential evolution with self adaptation and local search for constrained multi objective optimization. In: Congress on Evolutionary Computation proceedings (pp. 192–202). New Jersey: USA.

Zhang, Wenqiang, & Gen, Mitsuo (2011). An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. *Intelligent Manufacturing, 22*, 367–378.

Zitzler, E., Laumanns, M., & Bleuler, S. (2004). A tutorial on evolutionary multi objective optimization. In X. Gandibleux, M. Sevaux, K. Sörensen, & V. T'kindt (Eds.), *Metaheuristics for multi objective optimization. lecture notes in economics and mathematical systems* (pp. 3–37). 535: Springer.

Zitzler, E., & Thiele, L. (1999). Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.

Zou, W., Zhu, Y., Chen, H., Shen, H. (2011a). A novel multi-objective optimization algorithm based on artificial bee colony. Genetic and Evolutionary Computation Conference (pp. 103–104). Dublin, Ireland.

Zou, W., Zhu, Y., Chen, H., & Zhang, B. (2011b). *Solving multi objective optimization problems using artificial bee colony algorithm. Discrete dynamics in nature and society*. Hindawi Publishing Corporation. Article ID 569784, 37 pages.