Computers & Industrial Engineering 123 (2018) 189–208

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

An enhanced NSGA-II algorithm for fuzzy bi-objective assembly line balancing problems

Hossein Babazadeh[a], M.H. Alavidoost[b,*], M.H. Fazel Zarandi[b], S.T. Sayyari[a]

[a] Department of Industrial Engineering, University of Science and Technology, Tehran, Iran
[b] Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Keywords:
Fuzzy programming
Multi-objective genetic algorithm
Assembly line balancing
NSGA-II

ABSTRACT

Due to immense role of an efficient assembly line in today's manufacturing systems, the consideration of this paper is devoted to address the straight and U-shaped assembly line balancing problems. Assembly Line Balancing Problem (ALBP) considers conflicting objective functions that should be optimized simultaneously subject to set of constraints. To this end, this paper spends endeavor to develop a new fuzzy linear programming model. Having dealt with uncertain nature of the real production system, triangular fuzzy numbers (TFNs) are employed in order to represent uncertainty and vagueness associated with the task processing times. The proposed model can be regarded as a basis of fuzzy programming for further practical development in assembly line problems. To solve the problem, an efficient Multi Objective Genetic Algorithm (MOGA) is proposed. For this purpose, having respected several special characteristics of both fuzzy straight and U-shaped assembly line balancing problems for algorithm segments, including initial generation, encoding and decoding schemes, and GA's operations, an new approach is proposed to promote population diversity and search efficiency. Finally, the proposed algorithm is evaluated though several benchmarks as well as that of exact method. The obtained results proved high efficiency of our method over other approaches suggested in the literature.

1. Introduction

The current competition in the global market has motivated producers to follow all productivity improvement programs in order to boost effectiveness and efficiency. Thus, one of the major plans for manufacturing systems is designing an efficient assembly line (Baudin, 2002). ALBP by definition is as an allocation problem either in batch or mass production system in which a set of the required tasks must be assigned to the set of stations in order to yield a product so that some objective functions should be optimized subjected to a set of constraint. In this point of view, the sequences of tasks are considered as the most important issues in an assembly line development (Kao, 1976). Therefore, the main purpose of ALBPs is to design the proper order and sequence of the tasks along with the assembly line (Kao, 1976). ALBP is classified into two main categories such as Simple ALBP (SALBP) and Generalized ALBP (GALBP). GALBPs have some extra characteristics like cost goals, station parallelization, mixed-model production, etc., when compared with SALBs (Becker & Scholl, 2006). In this study, the attention is paid to SALB problems. According to their objective functions, SALBPs are classified in to four groups including, SALBP-F, SALBP-I, SALBP-II, and SALBP-E. ALBP-F is concerned as a feasibility problem with a specified combination of stations number and cycle time. SALBP-I and SALBP-II models present a dual relationship, but the first one tries to minimize the cycle time concerning a given number of stations; the second one tries to minimize the number of stations concerning a given cycle time. Moreover, SALB-E model objects minimizing the cycle time and the number of stations so that the efficiency has to be maximized (Sivasankaran & Shahabudeen, 2014). Fig. 1 shows the types of ALB problems.

The ALBP can also be grouped into other classifications according to variety of products and its workstation's layout. Although, ALBP can be categorized into two main groups, including straight and U-shaped assembly lines according to layout point of view. As regards to the variety of products perspective, they can be classified into the single and mixed model production methods. This study attempted to address two main problems, which include simple assembly line balancing problem (SALB) and U-shaped type of SALB (SULB); it should be stated that SALB and SULB represent single model production method of straight and U-shaped assembly line problems, respectively. Fig. 2 provides the illustration of classifications according to variety of products.

SALB was considered as the main element of traditional mass

* Corresponding author.
E-mail addresses: mostafa.alavidoost@aut.ac.ir (M.H. Alavidoost), zarrandi@aut.ac.ir (M.H. Fazel Zarandi).

https://doi.org/10.1016/j.cie.2018.06.014
Received 10 January 2015; Received in revised form 11 April 2018; Accepted 7 June 2018
Available online 15 June 2018
0360-8352/

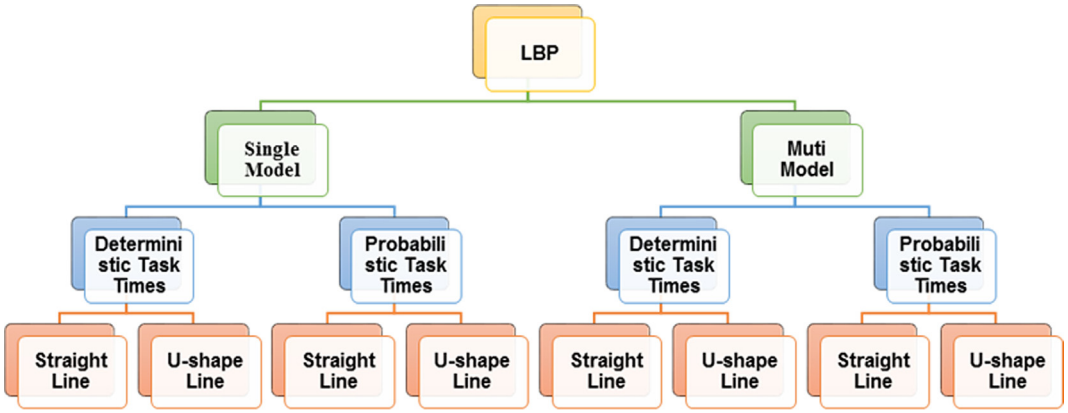

Fig. 1. Types of ALB problems.

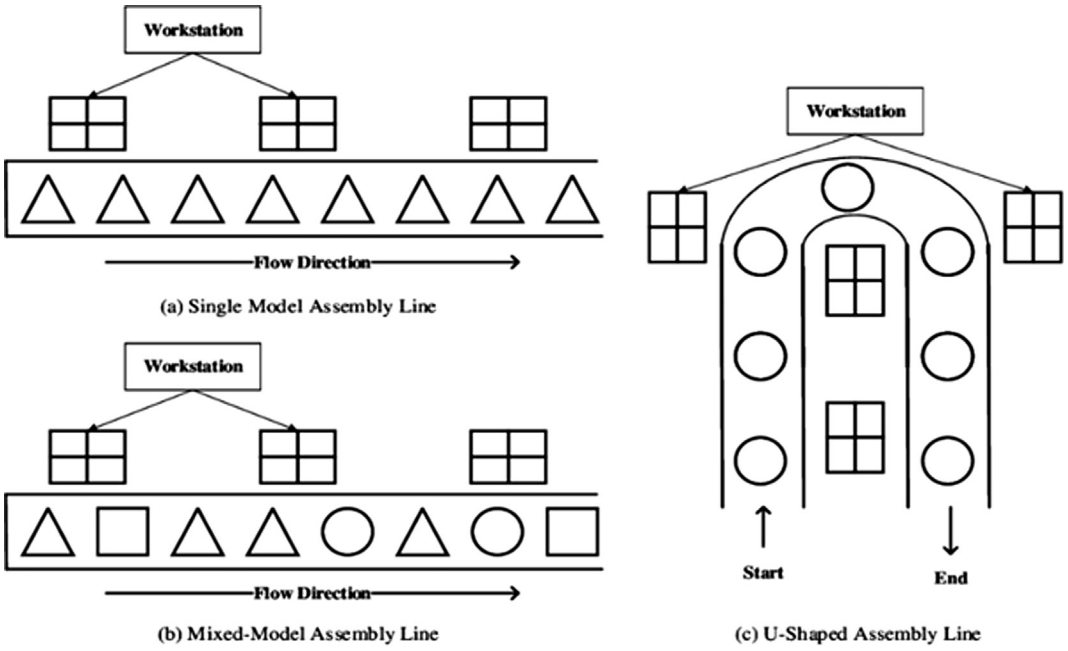

Fig. 2. Single and mixed SALB and SULB problems.

method in the past until U-shaped assembly line was introduced through application of continuous improvement and cost reduction concepts which are proposed in just-in-time (JIT) system (Monden, 2012). In comparison with SALBs, U-shaped assembly line ones have several extra advantages, such as flexibility, productivity increase, quality improvement, and lower WIP inventory. It is noteworthy that, the main characteristic which makes the U-shaped assembly line different from a straight one, is that in the former, the exit and entrance points are placed in the same position. It should be emphasized that this paper contributes both to the theory and practice in ALBP. As regards to the theory, this study can be considered as one of the first studies that developed fuzzy programming that an interactive approach along with meta-heuristic algorithm are employed to solve the problem. With respect to practical usage, an efficient interactive approach is proposed to deal with uncertainty associated with operational parameters as well as DM' satisfactory preferences. The remainder of this paper proceeds as follows. Section 2 is dedicated to provide the literature review of the works in this research area. In section 34, the new fuzzy mathematical models for both SALB and SULB is proposed. A novel design of meta-heuristic approach (NSGA-II) as a solution procedure for solving the problem is proposed in Section 4. In Section 5, validity of the proposed approach and the solution method are benchmarked with test problem considered in (Kara et al., 2009) along with the assessment of its performance over several test problems and benchmarks. The comparison result is discussed in Section 6 and the numerical illustration is provided in Section 7. Finally, conclusion remarks and proposal for future works are presented in Section 8.

2. Literature review

2.1. Fuzzy ALBP models

In comparison with crisp version of ALBP, inadequate attention has been paid to ALBP in uncertain environment and few studied have addressed ALBP with uncertainty (Baybars, 1986b; Erel & Sarin, 1998; Ghosh & Gagnon, 1989; Salveson, 1955; Scholl & Becker, 2006). Tsujimura, Gen, & Kubota, (1995) and Gen, Tsujimura, & Li (1996) were the first who developed a GA algorithm as a solution procedure dealing with fuzzy SALB (*f*-SALB) problem employing fuzzy task processing times. (Zhang & Cheng, 2010) also made attempt to present an approach to solve fuzzy SULB (*f*-SULB) problem. Having deployed fuzzy programming approach, (La Scalia, 2013) conducted the only study in *f*-SALB context that its new mathematical formulation is solved by using exact solution procedure where task processing times are considered as

fuzzy numbers. The work of Zhang and Cheng (2010) is the only one which dealt with the *f*-SULB problem concerning fuzzy parameters. Applying fuzzy goal programming (FGP) approach, (Toklu & özcan, 2008) developed FGP model to address the SULB problem. In this line, use of binary fuzzy goal programming (BFGP) method, (Kara, Paksoy, & Chang, 2009) proposed a mathematical formulation with a new solution approach to deal with both the multi-objective SALB and SULB problems. (Hazır & Dolgui, 2015) developed a robust formulation for U-type ALBP by means of interval data.

2.2. Heuristic and meta-heuristic approaches for ALBPs

For driving an optimum solution, considering ALBP with J number of tasks and P number of constraints, it is needed for searching $j!/2^P$ of solutions. Therefore, assembly line problems can be classified as NP-hard problem (Gutjahr & Nemhauser, 1964; Ajenblit & Wainwright, 1998). Due to the difficulty of such problems, lots of attempts have been dedicated to develop a heuristic method to solve these problems such as the ranked positional weighting technique (RPWT) (Helgeson & Birnie, 1961), LBHA method (Baybars, 1986a), MALB technique (Dar-El, 1973), MUST technique (Dar-El & Rubinovitch, 1979), COMSOAL technique (ARCUS, 1966), a critical path method (CPM) based approach (Avikal, Jain, Mishra, & Yadav, 2013). In this direction, meta-heuristic methods are considered as an efficient approach to deal with the complexity associated with solving ALBPs such as genetic algorithm (GA) (Ajenblit & Wainwright, 1998; Falkenauer & Delchambre, 1992), simulated annealing (SA) (Baykasoglu, 2006), Tabu search (TS) (Lapierre, Ruiz, & Soriano, 2006; Peterson, 1993), ant colony optimization (ACO) (Sabuncuoglu, Erel, & Alp, 2009), and Particle swarm optimization (PSO) (Jian-sha, Ling-ling, & Xiu-lin, 2009). In this context, among many meta-heuristic approaches offered by the literature, GA, due to its capability of providing controlled random search to identify optimum solution, has received significant attention in most scholars and considered as a valuable alternative method for traditional optimization approach applied sophisticated problems. A multi-objective GA for solving U-shaped ALBP is suggested by (Hwang, Katayama, & Gen, 2008) so that a comparison study between Straight and U-shaped Assembly Lines is performed. Kim, Song, and Kim (2009) presented a mathematical model for two-sided assembly line employing GA to solve it. Hwang and Katayama (2009) addressed mixed-model U-shaped lines providing a multi-decision genetic approach as a solving procedure. A TS algorithm is suggested by Özcan and Toklu (2009) for solving two-sided ALBP where the delivered results were evaluated by benchmark approaches. Yu and Yin (2010) developed an adaptive GA in order to solve ALBP such that validation study is conducted for the generated solutions. A hybrid GA as well as a multi-objective GA proposed by Zhang and Gen (2011) and Akpınar and Mirac Bayhan (2011) to deal with simple mixed model ALB and mixed model ALB considering parallel station and zoning constraints, respectively. Additionally, a two-stage GA derived by Kazemi et al., (2011) in order to tackle mixed model U-shaped ALBP. Nearchou (2011) developed a novel PSO approach to deal with SALB in which the results of this approach are compared with those derived by existing benchmark.

With concerning of productivity concept in ALBP, (Chang, Huang, & Ting, 2012) offered the use of GA with External Self Evolving Multiple Archives approach as a solving procedure. Having dealt with a multi-objective two-sided mixed-model assembly line, (Chutima & Chimklai, 2012) deployed PSO algorithm to solve it where it is illustrated that, the quality of obtained solution would be considerably more effective if their proposed algorithm is being combined with local search scheme. In other attempt, (Purnomo, Wee, & Rau, 2013) developed a novel mathematical model of two-sided ALBP in which GA and iterative first fit rule approaches are deployed to tackle it such that and the produced results are compared together as well. In this line, (Manavizadeh, Hosseini, Rabbani, & Jolai, 2013 offered SA as a solving approach to deal with mixed model U (type-I) ALBP where the created results are compared with those reached by exact method. Furthermore, (Hamzadayi & Yildiz, 2013) employed SA algorithm to detect ALBP with respect to sequencing rule in SULB as well. An integer programming model accompany with a novel meta-heuristic method for mixed-model ALBP presented in the work of Yuan, Zhang, and Shao (2013). Having addressed SALBP-1 in Dou, Li, and Su (2013), a discrete PSO approach is proposed as a solving procedure where the produced results compared with GA. In an attempt to solve sequence-dependent disassembly line balancing, (Kalayci & Gupta, 2013) employed PSO with a neighborhood-based mutation operator. A multi-objective teaching–learning-based optimization method is proposed by Li et al. (2014) to solve two-sided ALBP. A modified PSO is also provided by Delice, Kızılkaya Aydoğan, Özcan, and İlkay (2014) to consider two-sided ALBP. A new hybrid ACO algorithm suggested by Zha and Yu (2014) to treat U-shaped balancing and rebalancing problems where the obtained results are compared with those produced by benchmarks. Also, (Al-Zuheri, Luong, & Xing, 2014) employed GA to treat a mixed-model ALBP.

Although there exist many scholars covering other versions of ALBP, the ones that utilized heuristic and meta-heuristic algorithms to solve the problem in fuzzy circumstance, are significantly rare. This direction is initialized by Tsujimura et al. (1995) and Gen et al. (1996) in which utilizing fuzzy GA, they solved SALBP-1 where fuzzy numbers are employed to consider tasks processing times. While a combined approach having incorporated GA with Branch and Bound method developed by Brudaru and Valmar (2004) in order to solve SALBP-1, a modified Ranked Positional Weighting Technique combined with COMSOAL method considering fuzzy numbers is suggested by Fonseca, Guest, Elam, and Karr (2005) to solve such problems. However, a heuristic approach is offered by Hop (2006) to treat a fuzzy mixed-model ALBP whereas, (Zhang, Cheng, Song, & Yu, 2009) developed a heuristic method to solve SULBP having considered fuzzy numbers. Moreover, (Özbakır & Tapkan, 2010) addressed a two-sided ALB problem in which Bees algorithm is deployed as a solving approach. In an inspiring work, a new multi-objective GA introduced by Zacharia and Nearchou (2012) with object of solving SALBP-2 considering fuzzy numbers so that weighted sum of objectives approach was applied. They also suggested a meta-heuristic algorithm based on GA to deal with SALBP-E (Zacharia & Nearchou, 2013). An adaptive Multi Objective GA approach is developed by Alavidoost, Fazel Zarandi, Tarimoradi, and Nemati (2014) to solve SALB-1 and SULB-1 employing fuzzy numbers such that the performance of GA is enhanced use of a fuzzy controller as well as One Fifth Success Rule. Also, in their recent work (Alavidoost, Tarimoradi, & Fazel Zarandi, 2015; Tarimoradi, Alavidoost, & Fazel Zarandi, 2015), this approach is improved in which by introducing a modification of fuzzy controller and GA operators, they established a balance between convergence and diversity such that having compared the obtained results with those derived from other approaches in the literature, they succeeded to show the superiority of its performance. One of the scholars that multi objective approach is deployed to solve SALB is (Alavidoost & Nayeri, 2014) where fuzzy objectives and NSGA-II are combined to cope with uncertainty associated with the objectives.

3. Research objectives

In real world production systems, decision-making process should be performed with respect to vague or uncertain conditions. This vagueness and uncertainty can be resulted by either imprecision in problem parameters or values of goals. Accordingly, ALBPs are subjected to various types of the uncertainty which may originate from uncertain data associated with both machine and human factors (Tsujimura et al., 1995).

One of these uncertainty sources corresponds with variation in task processing times that may be considered especially for those whose lines have manual operations. In this line, the establishment of more reliable assembly line that is able to detect uncertainty specifically, for

those in mature enterprises such as automotive and high-tech industries is highly required. Meanwhile, since in some industries like power industries, new product development, etc., reliable historical data or probability distributions, may not exist, an efficient approach is significantly required to be used to deal with ALBP.

In other hand exact solving method consumes huge amount of time, cost and effort especially in large and practical problems (like automotive industries) Since the exact solving method is having so many constraints, there is need for an efficient framework that simultaneously considers ALBP modeling in the presence of uncertainty and a validated solving approach that has the capability of dealing with ALBPs in uncertain environment.

In this regard, since the DM may consider various goals, which implies the need for Pareto solution instead of one point solution to offer an efficient solving approach which can treat the problem and produce a result that is most compatible with his preference. On this note, after considering achieving these objectives, this paper presents an efficient framework, which is capable of dealing with such problems with respect to delivering valuable results. It should be noted that the outcome of this research can be useful to both researchers and practitioners.

4. Problem formulation

This section provides mathematical formulation for both *f*-SALB and *f*-SULB problems.

4.1. Notations

The following indices, parameters, decision variables, and indicator variables are used in the proposed formulation:

Indices:	
i, a, b, c	Task
j, r, q	Station
k	Objective function
Parameters:	
I	The set of tasks $I = \{i \mid i = 1, 2, \ldots, n\}$
J	The Set of station $J = \{j \mid j = 1, 2, \ldots, m_{max}\}$
P	The set of precedence relations
$\widetilde{t}_i$	Fuzzy task processing time i
θ_k	The weight of k^{th} objective function
γ	Balancing coefficient
Decision Variables:	
S_j	Set of tasks which are done at station j $S_j = \{i \mid \text{task } i \text{ is done at station } j\}, \forall\ jJ$
nS	Number of utilized station
$\widetilde{c}$	Fuzzy cycle time of assembly line
x_{ij}	1, **if** task i from the original diagram is assigned to station j, **otherwise** 0.
y_{ij}	1, **if** task i from the phantom diagram is assigned to station j, **otherwise** 0.
X_j	1, **if** at least one task is assigned to station j, **otherwise** 0.
f_k	k^{th} objective function
$\mu_k(x)$	k^{th} linear membership function (LMF)
M^{∞}	A very large positive number
λ_0	Minimum satisfaction level of goals
C_p	Allowed cycle time

4.2. Formulation of f-SALB and f-UALB problems

As stated earlier, the assembly line balancing problems involves various conflicting objective functions; it is necessary to optimize them simultaneously. The ALBPs are classified into two main categories including containing assignment and precedence constraint. The assignment constraint ensures that each task is assigned to exactly one station, while the precedence constraint guarantees that all precedence relations among tasks are satisfied. Accordingly, SALB problem can be formulated as follows:

$$Objective = Optimize(f_1, f_2, \ldots, f_k)$$
$$S.\ t \tag{1}$$

$$\begin{cases} \bigcup_{j=1}^{m_{max}} S_j = J \\ S_j \bigcap_{j \neq r} S_r = \varnothing \end{cases} \tag{2}$$

$$If\ \ (a, b) \in P,\ \ a \in S_j,\ \ b \in S_r,\ \ then\ j \leqslant r,\ \ for\ all\ a \tag{3}$$

Eq. (1) presents the objective function of the problem whereas Eqs. (2) and (3) provide set of assignment and precedence constraints, respectively. While the first constraint guarantees that all tasks must be allocated to the workstations, the second one ensures that each task will be allocated only to one workstation. By transforming Eq (3) into Eq. (4), we have;

$$\begin{cases} If\ (a, b) \in P,\ a \in S_j,\ b \in S_r,\ then\ j \leqslant r,\ for\ all\ a; \\ or,\ \ If\ (b, c) \in P,\ b \in S_j,\ c \in S_q,\ then\ q \leqslant j,\ for\ all\ c \end{cases} \tag{4}$$

In the following, an integrated fuzzy mixed-integer linear programming is presented.

Among several objectives which have been proposed in the literature for ALBP (Gökçen & Ağpak, 2006; Zacharia & Nearchou, 2012), this paper aims to optimize two objective functions. The first objective is minimizing the number of work stations, whereas, the second one intends to minimize the fuzzy cycle time, according to Eqs. (5) and (6), respectively.

$$f_1 = Min(nS) = Min\left\{\sum_{j=1}^{m_{max}} X_j\right\} \tag{5}$$

$$\widetilde{f}_2 = Min(\widetilde{c}) \tag{6}$$

In the presented study, the constraints which are specified for this problem are combined with those of constraints suggested by Deckro and Rangachari (1990), Gen et al. (1996) and Gökçen & Ağpak (2006) are considered as model constraints. Following this approach, new set of constraints for *f*-SALB problem based on fuzzy coefficients and variables can be formulated as Eqs. (7)–(13).

$$\sum_{j=1}^{m_{max}} x_{ij} = 1\ ,\ \ \forall\ i\ \in I \tag{7}$$

$$\sum_{j=1}^{m_{max}} (m_{max} - j + 1).\ x_{aj} - \sum_{j=1}^{m_{max}} (m_{max} - j + 1).\ x_{bj} \geqslant 0\ ,\ \ \forall\ (a, b) \in P \tag{8}$$

$$\sum_{i=1}^{n} \widetilde{t}_i(x_{ij}) \leqslant \widetilde{c},\ \ \forall\ j \in J \tag{9}$$

$$\sum_{i=1}^{n} \widetilde{t}_i(x_{ij}) + (1 - U_j) \cdot M^{\infty} \geqslant \widetilde{c},\ \ \forall\ j \in J \tag{10}$$

$$\sum_{j=1}^{m_{max}} U_j = 1 \tag{11}$$

$$\sum_{i=1}^{n} (x_{ij}) - n * X_j \leqslant 0\ ,\ \ \forall\ j \in J \tag{12}$$

$$\widetilde{c} \text{ is a } TFN,\ \ x_{ij},\ \ X_j,\ \ U_j \in \{0,\ 1\},\ \ \forall\ j \in J,\ \ \forall\ i \in I \tag{13}$$

Accordingly, regarding fuzzy coefficients and new variables, the

new set of constraints for *f*-SULB problem can be formulated as Eqs. (14)–(21)

$$\sum_{j=1}^{m_{max}} (x_{ij} + y_{ij}) = 1 \ , \ \forall\, i \in I \tag{14}$$

$$\sum_{j=1}^{m_{max}} (m_{max} - j + 1).\, x_{aj} - \sum_{j=1}^{m_{max}} (m_{max} - j + 1).\, x_{bj} \geqslant 0 \ , \ \forall\, (a, b) \in P \tag{15}$$

$$\sum_{j=1}^{m_{max}} (m_{max} - j + 1) y_{bj} - \sum_{j=1}^{m_{max}} (m_{max} - j + 1) y_{aj} \geqslant 0 \ , \ \forall\, (a, b) \in P \tag{16}$$

$$\sum_{i=1}^{n} \tilde{t}_i (x_{ij} + y_{ij}) \leqslant \tilde{c}, \ \forall\, j \in J \tag{17}$$

$$\sum_{i=1}^{n} \tilde{t}_i (x_{ij} + y_{ij}) + (1 - U_j) \cdot M^{\infty} \geqslant \tilde{c}, \ \forall\, j \in J \tag{18}$$

$$\sum_{j=1}^{m_{max}} U_j = 1 \tag{19}$$

$$\sum_{i=1}^{n} (x_{ij} + y_{ij}) - n * X_j \leqslant 0 \ , \ \forall\, j \in J \tag{20}$$

$$\tilde{c} \text{ is a} TFN, \ x_{ij}, \ y_{ij}, \ X_j, \ U_j \in \{0, \ 1\}, \ \forall\, j \in J, \ \forall\, i \in I \tag{21}$$

Constraints (7) and (14) as well as constraints (8) and (15), (16) are assignment and precedence constraints for *f*-SULB and *f*-SULB problems, respectively. Sets of constraints including (9)–(11) and (17)–(19) are the cycle time constraints such that ensure each processing fuzzy time of work stations do not exceed the fuzzy cycle time. Constraints (11) and (20) are considered as the work station constraints. Finally, constraints (13) and (21) denote the feasible area associated with the variables of *f*-SULB and *f*-SULB problems, respectively.

5. Solving approach

Regarding aforementioned objective functions and constraints, a two-phase approach is adopted in this study to solve fuzzy ALBP. While in the first phase, the fuzzy mathematical model is converted into an equivalent auxiliary crisp multi-objective mixed integer linear programming (MOMILP) model, in the second phase, a novel two-phase interactive fuzzy programming approach is proposed to achieve a satisfactory compromise solution. Additionally, this paper not only utilizes exact algorithms to solve fuzzy ALBPs, but also it deploys meta-heuristic ones as a solving procedure. In this regard, while two-phase approach is proposed as an exact method, an enhanced NSGA-II algorithm is developed as a meta-heuristic algorithm. However, to design the elements of multi objective meta-heuristic algorithm, such as encoding/decoding schemes, initial generation and GA operators, consideration is also given to special characteristics of *f*-SULB and *f*-SALB. In addition, a new approach is developed to promote the population diversity and search efficiency. Before presenting the solving approach, basic definitions for multi-objective problems and fuzzy arithmetic are provided as follows.

5.1. Multi objective optimization

Multi-objective optimization generally involves balancing all conflicting objectives and searches for a set of compromise solutions between the objectives while satisfying various constraints (Zitzler, 1999; Deb, Pratap, Agarwal, & Meyarivan, 2002). The general structure of multi objective optimization model is as follows.

$$\begin{aligned} &\text{Optimize } \mathrm{F(x)} = \{\mathrm{F_1(x)}, \boldsymbol{F}_2(\mathrm{x}), ..., \boldsymbol{F}_{\mathrm{z}}(\mathrm{x})\} \\ &s.\ t.\ : \\ &x \in \mathbf{X} \end{aligned} \tag{22}$$

where in Eq. (22), F(x) and **X** represent the objective functions as well as set of problem constraints, respectively. In such contexts, this set of solutions is considered as the set of Pareto-optimal solutions (Coello, Lamont, & Van Veldhuizen, 2007). As regards to non-dominated optimum borderline solutions in respect to each other, it is not possible to improve one objective function without worsening at least one of the other target functions. The concept of domination in multi-objective problems is defined as follows:

The solution y is considered to be efficient (that is, Pareto-optimal or non-dominated) in a two-dimensional space of risk and return if no solution b exists such that x dominates a. Solution x is considered the dominant solution of *y* if and only if Q_1 or Q_2 hold:

$$Q_1 \text{ IF } f_1(y) \leqslant f_1(x) \bigwedge f_2(y) > f_2(x) \ ; \forall\, k \in \{1, 2, ..., z\} \tag{23}$$

$$Q_2 \text{ IF } f_2(y) \geqslant f_2(x) \bigwedge f_1(y) < f_1(x) \ ; \exists\, k, \tag{24}$$

Eq. (23) expresses that solution y should not be better than x in no way while Eq. (24) implies that solution y should be better than solution x at least in one way.

5.2. Fuzzy arithmetic

The α-cut fuzzy arithmetic approach is usually used to transform fuzzy numbers from a singleton into an interval; here, the interval arithmetic could be defined according to Klir and Yuan (1995) by means of formula (25) (suppose that the * donates four basic operations {+,−,·,&/}).

$$[d, e] * [g, h] = \{f * u | d \leqslant f \leqslant e, \ g \leqslant u \leqslant h\} \tag{25}$$

It should be noted that four basic operations can be performed on the closed intervals using formulas (26)–(30).

$$[d, e] + [g, h] = [d + g, e + h] \tag{26}$$

$$[d, e] - [g, h] = [d - h, e - g] \tag{27}$$

$$[d, e] \cdot [g, h] = [min(d \cdot g, d \cdot h, e \cdot g, e \cdot h) \ , \ max(d \cdot g, d \cdot h, e \cdot g, e \cdot h)] \tag{28}$$

$$\frac{[d, e]}{[g, h]} = \left[min\left(\frac{d}{g}, \frac{d}{h}, \frac{e}{g}, \frac{e}{h}\right) \ , \ max\left(\frac{d}{g}, \frac{d}{h}, \frac{e}{g}, \frac{e}{h}\right) \right]; \ 0 \notin [g, h] \tag{29}$$

$$[d, e] = [g, h] \tag{30}$$

Having treated with the difficulty associated with employing α-cut in computation, TFNs are deployed because they offer more simple computations than other fuzzy numbers. Triplet points $\tilde{t} = (t^o, t^m, t^P)$ are generally used to define a triangular fuzzy task processing time. A membership function (MF) for TFN as well as its corresponding equation are presented in Fig. 3 and Eq. (31), respectively (Shahrasbi, Shamizanjani, Alavidoost, & Akhgar, 2017).

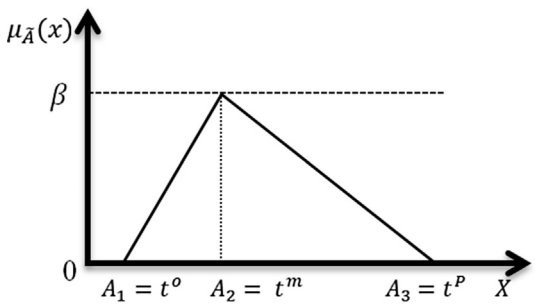

Fig. 3. A triangular membership function (Fazel Zarandi, Tarimoradi, Alavidoost, & Shirazi, 2015; Fazel Zarandi, Tarimoradi, Alavidoost, & Shakeri, 2015).

$$\mu(x)=\begin{cases}\beta\left(\frac{x-A_1}{A_2-A_1}\right), & A_1\leqslant x\leqslant A_2\\ \beta\left(\frac{A_3-x}{A_3-A_2}\right), & A_2\leqslant x\leqslant A_3\\ 0 & O.W.\end{cases},\ 0<\beta\leqslant 1 \tag{31}$$

As illustrated in Fig. 3, a fuzzy triangular MF includes three main elements consisting of; the most pessimistic value (t^p), the most optimistic value (t^o) and the most possible value (t^m). According to this figure, although the two first ones are located in the lower bound and upper bound of feasible region, respectively, they received the lowest membership degree ($\mu_{\widetilde{T}}(t^p)=\mu_{\widetilde{T}}(t^o)=0$). Also, the most possible value is located within the interval and has the maximum degree of membership ($\mu_{\widetilde{T}}(t^m)=1$). It is assumed that all other points located between these three points and have membership degree whose values vary linearly between zero and one. Although TFNs suggest less complexity than other fuzzy numbers, but it suffers from complexity associated with α -cut approach. In order to cope with this problem, the operation results are approximated into their linear form to face computational complexity and to preserve the fuzzy number shape simultaneously (Eqs. (32)–(36)) (Shermeh, Najafi, & Alavidoost, 2016; Nemati, Madhoshi, & Ghadikolaei, 2017; Nemati, Madhoushi, & Safaei Ghadikolaei, 2017).

$$\widetilde{A}+\widetilde{B}=(A_1+B_1,\ A_2+B_2,\ A_3+B_3) \tag{32}$$

$$\widetilde{A}-\widetilde{B}=(A_1-B_3,\ A_2-B_2,\ A_3-B_1) \tag{33}$$

$$\widetilde{A}\times\widetilde{B}=(min(A_1\cdot B_1,\ A_1\cdot B_3,\ A_3\cdot B_1, A_3\cdot B_3), A_2\cdot B_2, max(A_1\cdot B_1,\ A_1\cdot B_3,\ A_3\cdot B_1, A_3\cdot B_3)) \tag{34}$$

$$\frac{\widetilde{A}}{\widetilde{B}}=(min\left(\frac{A_1}{B_1},\frac{A_1}{B_3},\frac{A_3}{B_1},\frac{A_3}{B_3}\right), A_2/B_2, max\left(\frac{A_1}{B_1},\frac{A_1}{B_3},\frac{A_3}{B_1},\frac{A_3}{B_3}\right);\tilde{0}=(0,0,0)\notin\widetilde{B} \tag{35}$$

$$\text{Defuzzify}(\widetilde{A})=(A_1+2\times A_2+A_3)/4 \tag{36}$$

It is important to note that, defuzzification approach converts a fuzzy number to its crisp form, according to Eq. (36). Finally, fuzzy relations are presented in Eqs. (37) and (38).

$$A\leqslant or=or\geqslant\widetilde{B}\ \Leftrightarrow\ A\ \leqslant or=or\geqslant \text{Defuzzify}(\widetilde{B}) \tag{37}$$

$$\widetilde{A}\leqslant or=or\geqslant\widetilde{B}\ \Leftrightarrow\ \begin{cases}A_1\leqslant or=or\geqslant B_1\\ A_2\leqslant or=or\geqslant B_2\\ A_3\leqslant or=or\geqslant B_3\end{cases} \tag{38}$$

5.3. Exact solution approach for SALB & SULB

5.3.1. Dealing with fuzzy objective function

As stated earlier, having converted fuzzy number into a crisp form employing Eq. (36), the objective functions associated with the Auxiliary MOLP are defined by using Eqs. (39) and (40).

$$f_1=Min(nWS)=Min\{\sum_{j=1}^{m_{max}}X_j\} \tag{39}$$

$$f_2=Min\{\text{Defuzzify}(\tilde{c})\}=Min\{\frac{(c^o+2*c^m+c^P)}{4}\} \tag{40}$$

5.3.2. Dealing with fuzzy constraints

By considering Eq. (38), each fuzzy constraint is equivalent to three non-fuzzy constraints. Following this approach, for *f*-SALB, constraints (9)–(11) are converted to those presented in (41)–(43), respectively.

$$\sum_{i=1}^{n}t_i^{Point}(x_{ij})\leqslant c^{Point},\ \forall j\in J,\ Point\in\{o,m,p\} \tag{41}$$

$$\sum_{i=1}^{n}t_i^{Point}(x_{ij})+(1-U_j^{Point})\cdot M^{\infty}\geqslant c^{Point},\ \forall j\in J,\ Point\in\{o,m,p\} \tag{42}$$

$$\sum_{j=1}^{m_{max}}U_j^{Point}=1,\ \forall\ Point\in\{o,m,p\} \tag{43}$$

Accordingly, for *f*-SULB, constraints (17)–(19) are converted to those provided in (44)–(46), respectively.

$$\sum_{i=1}^{n}t_i^{Point}(x_{ij}+y_{ij})\leqslant c^{Point},\ \forall j\in J,\ Point\in\{o,m,p\} \tag{44}$$

$$\sum_{i=1}^{n}t_i^{Point}(x_{ij}+y_{ij})+(1-U_j^{Point})\cdot M^{\infty}\geqslant c^{Point},\ \forall j\in J,\ Point\in\{o,m,p\} \tag{45}$$

$$\sum_{j=1}^{m_{max}}U_j^{Point}=1,\ \forall\ Point\in\{o,m,p\} \tag{46}$$

Thus, having converted the *f*-SALB model into an auxiliary crisp MOMILP one, a set of objectives as presented in Eqs. (39) and (40), as well as a set of constraints addressed by Eqs. (7), (8), (12), (13), and (41)–(43), will be derived. Similarly, by transforming *f*-SULB model into an auxiliary crisp MOMILP form, set of objectives which are presented in Eqs. (39) and (40), as well as set of constraints provided in Eqs. (14)–(16), (20), (21), and (44)–(46), are obtained.

5.3.3. Phase II: interactive fuzzy programming approach

5.3.3.1. Fuzzy programming approach. Fuzzy decision making and Fuzzy Programming are the two methods which have been proposed by Bellman and Zadeh (1970) and Zimmermann (1978) respectively, to solve MOLP. In these approaches, the same mechanism is considered in which both methods object to obtain the positive ideal solutions (*PIS*) as well as the negative ideal solutions (*NIS*) of all the corresponding objective functions. These ideal can be determined either by the DM's preferences (between the worst and best values of each function) or the worst and best values of each objective function (Eq. (47)).

$$\begin{aligned}&f_1^{PIS}=\min f_1=\min(nWS),\ f_1^{NIS}=\max f_1=\max(nWS)\\ &f_2^{PIS}=\min f_2=Min\left\{\frac{(c^o+2*c^m+c^P)}{4}\right\},\\ &f_2^{NIS}=\max f_2=Max\left\{\frac{(c^o+2*c^m+c^P)}{4}\right\}\end{aligned} \tag{47}$$

An equivalent linear MF corresponding to each objective function can be obtained by means of Eqs. (48) and (49). The graphical diagrams for these MFs is depicted in Fig. 4 (Alavidoost, Babazadeh, & Sayyari, 2016; Nemati, & Alavidoost, 2018).

$$\mu_1(x)=\begin{cases}1 & if\ \ f_1\leqslant f_1^{PIS}\\ \frac{f_1^{NIS}-f_1}{f_1^{NIS}-f_1^{PIS}} & if\ \ f_1^{PIS}\leqslant f_1\leqslant f_1^{NIS}\\ 0 & if\ \ f_1\geqslant f_1^{NIS}\end{cases} \tag{48}$$

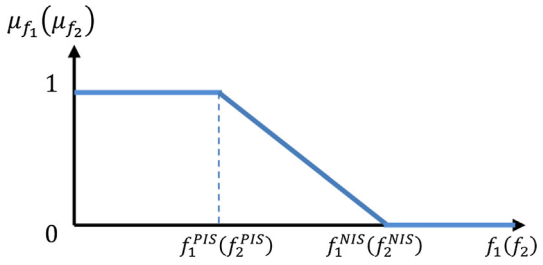

Fig. 4. Liner membership function equivalent with $f_1(f_2)$.

$$\mu_2(x) = \begin{cases} 1 & if \quad f_2 \leqslant f_2^{PIS} \\ \frac{f_2^{NIS} - f_2}{f_2^{NIS} - f_2^{PIS}} & if \quad f_2^{PIS} \leqslant f_2 \leqslant f_2^{NIS} \\ 0 & if \quad f_2 \geqslant f_2^{NIS} \end{cases} \tag{49}$$

where $\mu_k(x)$ represents the LMF for the k^{th} fuzzy objective.

5.3.3.2. Interactive fuzzy programming. The ability to provide a systematic framework is the most valuable characteristics of interactive approaches; this framework enables the DM to adjust the parameters regarding their preferences interactively until an efficient satisfactory solution is reached. In this approach, the DM adjusts interactively the search direction during the solution procedure, until the preference parameter values is extracted (Abd El-Wahed & Lee, 2006). This study extends the fuzzy programming approach developed by Abd El-Wahed and Lee (2006). In this accord, first, fuzzy MILP model is converted to an equivalent auxiliary crisp MOMILP one and then the *PIS* and *NIS* of each objective function (by treating with the corresponding MILP model according to Eq. (47)) accompany with specifying LMF correspond to each objective function, are determined (use of Eqs. (48) and (49)). Then, our extended approach respects the following steps:

Step 1. Conversion of the MOMILP model into an equivalent MILP using the following proposed auxiliary crisp formulation (Eq. (50)) next, solve the problem.

$$\begin{aligned} &Max\lambda(x) = \lambda_0 + \delta \sum_{k=1}^{K} \theta_k \lambda_k \\ &S.\ t:\ \ \theta_k(\lambda_0) + \lambda_k \leqslant \mu_k(x), \quad k = 1...K \\ &x \in F(x) \end{aligned} \tag{50}$$

where δ represents the small positive number which is normally set to be 0.01.

Step 2. The obtained solution is delivered to the DM; **if** the solution satisfies the preference of the DM, it is considered as the compromise solutions. **Otherwise**, the following step is taken.

Step 3. Interactive revision should be performed on the about DM's preference to the favorable solution. This process is repeated until the favorable solution is reached using the solution approach; here, the increase (or decrease) of the *NIS* for maximization objectives (or minimization objectives) and changes in weights of objectives are allowed. It should be noted that since any changes made in NIS can lead to large variation in the obtained solutions, the size of the change in NIS should be as small as possible to stop producing infeasible result.

5.4. Meta-heuristic approach

Meta-heuristic algorithms are known as powerful random searching approaches that seek optimum solutions inspired from nature. Therefore, the main prominence deserved by multi-objective meta-heuristic methods as compared to the classic methods is to produce a set of optimum points as solution in once implementation instead of several attempts, which may be spent in classic ones. For this reason, this paper a modified a multi-objective meta-heuristic method called "*Non-dominated sorting genetic algorithm* (*NSGA-II*)" that presented by Deb et al. (2002).

NSGA-II is one of the most popular GA-based which is employed in the variety of applications. This algorithm is based on Pareto-dominance framework so that its simplicity and modularity accompany with the lower number of parameters make it a powerful approach for multi-objective optimization. Two most prominent operators of deploy in this algorithm are non-dominated ordering and crowding distance. While non-dominating sorting meant is employed for delivering adequate convergence by classifying the obtained solutions in various Pareto front, the crowding distance operator, which takes care of maintaining the diversity of the responses of algorithm among non-dominated solutions use of calculates the dispersion of solutions in any front. Crossover and mutation operators are adopted from GAs for recombination process.

Main loop of NSGA-II algorithm

The main loop of NSGA-II algorithm is expressed as follows: Initial generation process starts with creating a random parent's population. Then, the population is sorted according to the non-domination approach. Next, the new offsprings is produced by participating in non-dominated solutions with the use of binary tournament selection, mutation as well as recombination. After the initial generation process, parents and offsprings population are combined together. Therefore, a population is created with the size of 2N (N is number of solutions). After this step, the population is sorted based on non-domination approach in which the new population will be produced with the selection procedure following to the ranking and crowding distance parameter of a solution. This process proceeds until the number of generation is reached. The pseudo code for NSGA-II algorithm is provided as follow (Alavidoost, Tarimoradi, & Zarandi, 2015):

Procedure: NSGA-II
BEGIN
Step1. Determine population size (*nPop*), crossover rate (β), mutation rate (γ), and iteration numbers (*Itr*).
Step2. Randomly generate first population.
Step3. Sort individuals using Non-dominated Sorting (NS) Algorithm and calculate Crowding Distance for each of them.
3.1. Select a couple of individuals from population randomly.
3.2. IF they are from different fronts, **THEN** select one with less front number, **ELSE,** select one with higher Crowding Distance.
Step4. Deploy crossover and mutation operators on parents for generating offspring.
Step5. Sort parents and offspring using Non-dominated Sorting (NS) Algorithm and calculate Crowding Distance for each of them.
Step6. Select best individuals (in rank and crowding distance) from existing population and Offspring and generate new generation.
Step7. Consider termination condition and back to step 3 if the condition is not satisfied.
END.

The most efficient solution of this algorithm is produced if the NSGA-II operators are defined appropriately and then well adapted to the problem as well. Thus, at first, it is necessary to properly define the algorithm operators for both SALB and SULB. The general process of NSGA-II is represented as in Fig. 5. If the NSGA-II operators are properly defined, then the most efficient solution of this algorithm is produced and will be well adapted to the problem. Thus,

5.4.1. Chromosome representation (encoding)

5.4.1.1. Chromosome for f-SALB problem. Despite the several procedures which have been proposed for chromosome encoding, strings of integers are considered as the most simple and popular method in ALBP (see (Tasan & Tunali, 2008) as a survey). However, in all approaches, chromosomes should remain feasible so that the task sequence of chromosomes should not violate the precedence relations.

In this study, feasible sequence representation approach is used in the form of structure in which each chromosome consists of two segments such that the first one is devoted to integer numbers permutation from one to number of tasks which presents task priorities whilst the second one dedicated to express allowed cycle time (C_p) which is a real number so that $C_p \in [f_2^{PIS}, f_2^{NIS}]$. Consider a 9-task ALBP whose

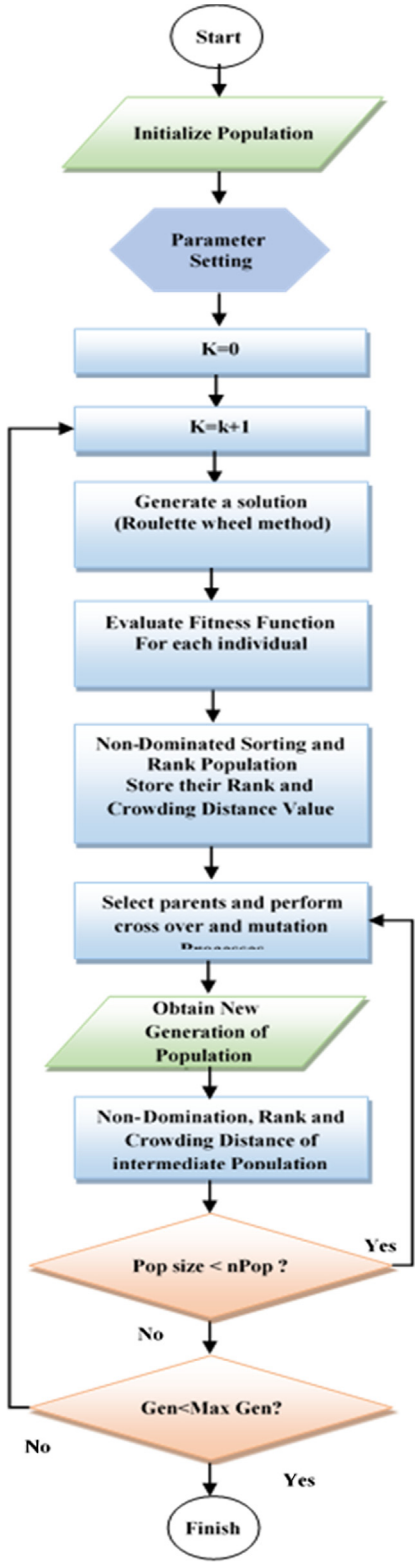

Fig. 5. The general process of NSGA-II method.

predecessor and successor relations are shown in Fig. 6 (the digit in each circle corresponds to the task number; what is addressed earlier is the fuzzy processing time in terms of a TFN). Fig. 7 displays an instance for genotype of chromosome where task priorities and cycle times are specified by gray and blue colors, respectively. In this example, the highest priority for assignment is the task 6 (first gene); then, it is followed by task 5 (second gene), etc.

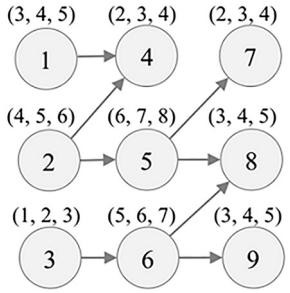

Fig. 6. Precedency constraints graph for a 9-task for ALBP.

Fig. 7. An example for genotype of *f*-SALB chromosome.

5.4.1.2. Chromosome for f-SULB problem. Although, similar to *f*-SALB, chromosome representation in *f*-SULB utilizes both feasible sequence and structure approaches, the main difference between them refers to the number of sections, which they use including two and three sections according to *f*-SALB and *f*-SULB problems, respectively. The other difference of these two models is related to the task assignment procedure so that in the SALB, tasks should be to selected from the beginning of the chromosome and be assigned to the workstations, whereas in the SULB, tasks can be chosen from either the beginning or the end of the chromosome.

Fig. 8 presents genotype of *f*-SULB chromosome, such that, like *f*-SALB, the first and third sections (specified by gray and blue colors) illustrate task priorities as well as cycle time associated with *f*-SULB. Meanwhile, the second section (specified by orange color) represents a string of binary numbers whose length is as equal as the number of tasks that defines the assignment procedure of gens along with the process. This process will be discussed in the following.

5.5. Constraint handling in GA

In most of the practical applications, multi objective optimization problems are subjected to set of constraints (Kramer, 2010). Two most well-known constraint-handling approaches which have been considered in GAs are penalty and repair methods. In this regard, in the penalty approach, the constrained problem is converted into an unconstrained one, while the repairing approach take advantage of the problem's characteristics with the objects to fix infeasible solutions until it reach a feasible one. Since the use of penalty method requires to define an appropriate penalty function for augmenting the constraints to the objective function (which is almost difficult to do due to the high sensitivity of solution quality on the penalty term), the most attempts in literature of constraint handling have been performed to extend the repairing method where this procedure can efficiently use the characteristic of problem by making the relationship between decision variables and constraints. This study attempt to propose a new repairing mechanism in GA stochastic search process.

5.5.1. Repairing mechanism

First, all chromosomes should be changed into their feasible form in which the order sequence of tasks should not violate the precedence relationship. As shown in Figs. 4 and 5, task 6 deserved the highest priority value rather than other tasks. This statement is saying this task should be assigned before others because it cannot be accomplished due to violating precedence relationship (as shown in Fig. 6). To deal with this problem, this paper uses a repairing mechanism. The main deficiency of this procedure is associated with decreasing the diversity of the algorithm's solutions obtained, thus making the algorithm to

Fig. 8. An example for genotype of *f*-SULB chromosome.

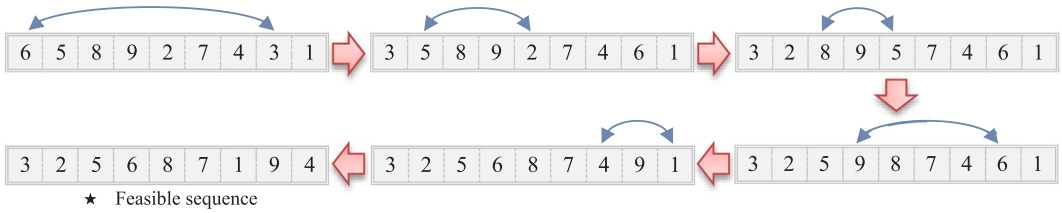

Fig. 9. Swap repairing procedure for presented example.

converge to local optimum solution. In the other words, the algorithm lacks the capability to search the solution in all the solution space. Having considered this drawback, this study introduces a new repairing mechanism by combining several repairing methods suggested in the literature. A brief review of these methods is presented in the following.

5.5.1.1. Swap repairing (SR). In this mechanism, first, all tasks that violated the predecessor relationship are identified and after that every pair of these tasks are being swapped with each other. The SR procedure is provided as follows. To make it much illustrative, the SR mechanism implemented for the example considered in Fig. 6, which its result is demonstrated as shown in Fig. 9.

5.5.1.2. Shift forward & shift backward repairing (SFR & SBR). In the SFR mechanism, the predecessor-violated tasks are identified first and then these tasks are forwarded to the first next gen whose predeceasing relations are not violated. The SFR procedure is provided as follows:

While SBR mechanism is similar to the SFR approach in identifying the predecessor violated tasks, but compared to SFR approach, SBR backwards these tasks to the first pervious gen whose predeceasing relations are not violated instead of transmitting them into the first next gen as in SFR. The results of implementing SFR and SR mechanisms for the presented example are as shown in Figs. 10 and 11, respectively.

5.5.1.3. Random shift forward & random shift backward repairing (RSFR & RSBR). Although both RSBR and RSFR utilize the same procedure as used in SBR and SFR, respectively, but when compared with SFR and SBR approaches where tasks are transmitted to the first gen whose predeceasing relations are not violated, in RSFR and RSBR methods, tasks are randomly transmitted to those gens in which tasks predeceasing relations are not violated. The instant results of implementing RSFR and RSBR mechanisms for the presented example are as shown in Fig. 12.

5.5.1.4. Seven fold population rule. In this paper, a new repairing mechanism is proposed in which combines several repairing methods including Swap Repairing (SR), Shift Forward Repairing (SFR), Shift Backward Repairing (SBR), Random Shift Forward Repairing RSFR and RSBR are considered simultaneously. In this line, instead of employing one repairing approach which may leads to repeat repairing process several times that may consume may much time, all repairing approaches are considered in once. In this accord, after performing all seven repairing approaches, fitness evaluation function is calculated for each repaired chromosome and then the chromosome whose fitness evaluation value is higher than others, is considered as the best chromosome. This procedure carried out similarly for both SAB and SULB. The process demonstration of implementing this approach is provided as in Fig. 13.

The proposed GA process including our new repairing mechanism, which has been utilized in this study, is expressed as follows. In the first step, with encoding chromosome, after setting GA parameters, an initial chromosome is generated by employing three main operators of GA. In the next step, the evaluation process is performed in which the associated fitness value of the chromosome is assessed. Then, feasibility check, the need of repairing process is identified. If the chromosome is identified as an infeasible one, by means of the proposed repairing procedure including the use of Swap repairing, Shift forward/backward repairing and Random Shift forward/backward repairing, the infeasibility of the solution is resolved. It should be noted that in this approach the repairing method which provides maximum level of fitness value is selected to proceed crossing the algorithm. This procedure proceeds until the stopping criteria is met. Fig. 14 provides the graphical representation f the proposed process of GA.

5.5.2. Chromosome decoding

Chromosome decoding is defined as process in which tasks are assigned into workstations such that, when the current station is j, tasks are allocated to j^{th} station regarding to their priority until the workstation's cycle time is not greater than that of the expected fuzzy cycle time of assembly line. In other accord, task a can be allocated into the current workstation, if the condition $\{\sum_{iS_j} t_i + t_a \leqslant C_p\}$ is hold and at the same time a deserved the highest priority, otherwise, another workstations should be constructed in order to cover task. The process of chromosome decoding in SALB includes the following steps.

Procedure: Chromosome Decoding Process for SALBP
Begin
Step 1. Consider
$j = 1$, UAJ = {Set of tasks which have not been assigned yet}
Step 2. IF there is no task which should be assigned; $UAJ = \{\}$, **Then** stop. **ELSE**, go to step 3.
Step 3. Identify the task, which deserved the highest priority for assignment.
Step 4. IF the condition $\{\sum_{iS_j} t_i + t_a \leqslant C_p\}$ holds, **Then** go to step 5; **ELSE**, go to step 6.

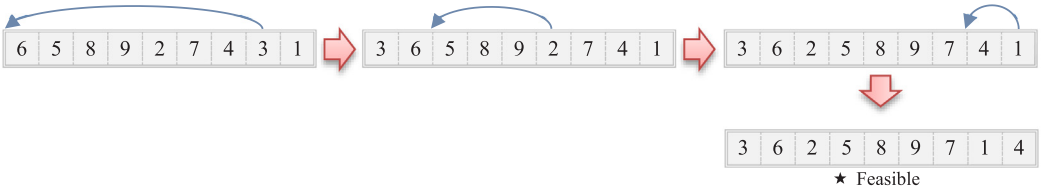

Fig. 10. Shift Forward Repairing for presented example.

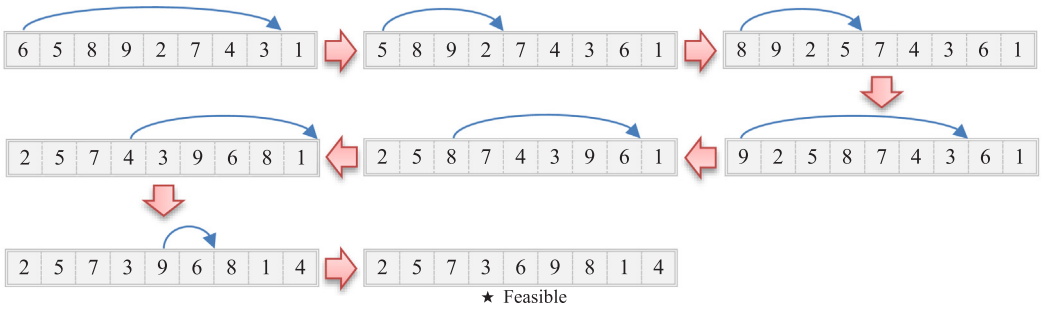

Fig. 11. Shift Backward Repairing for presented example.

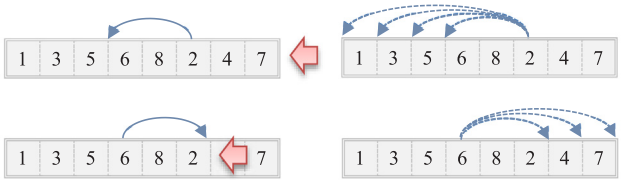

Fig. 12. Random Shift Forward/Backward Repairing for presented example.

Step 6. Assign task a to the current workstation and remove it from the set UAJ, **Then**, go back to step 2.

Step 7. Construct a new workstation (j= j+ 1), assign task a into that station, remove task a from the set UAJ, **Then**, go back to step 2.

END

As concerned earlier, although the chromosome decoding process follows similar procedures in SALB and SULB, determining task priorities in these models is considerably different where the following section is devoted to explore it.

5.5.3. Fitness evaluation

As discussed before, utilizing fuzzy objectives in order to reflect vagueness associated with value of goals could be resulted in creating a set of none-dominated solutions, which is closer to those in optimal condition. Following this approach, having established membership function for problem objectives ($\mu_1(f_1)$, $\mu_2(f_2)$), fuzzy goals should be determined by considering the cycle time value and number of stations in fuzzy objective. In this respect, while the number of stations is determined according to chromosome decoding process, the cycle time identified as equal as $\max_j\{t(S_j)\}$. Accordingly, the cycle time is determined as a real number respecting to the interval $c \in [C_g, C_p]$.

5.5.4. Initial generation

The initial generation is randomly produced in which the first part is devoted to perform permutation of an integer between one and the number of the tasks. In this regard, a string of binary number is created for the second part, whilst in the third part a real random number is produced with respect to the interval $[C_g, \overline{C}]$. It is noteworthy that since the chromosomes, which are provided in initial generation process are feasible sequence, the repairing process should be carried out for the first part of each chromosome.

5.5.5. Evolution operators

Having explored in solution space as well as exploited on the caught solutions, GA primes optimum solution for the problem. While the greater the focus on exploration leads to the weaker convergence rate which the algorithm would achieve, the more focus on the exploitation may result in to more decrease in diversity that it could deliver. As a result, to reach an appropriate solution in a reasonable time, a good tradeoff between convergence and diversity is necessarily required. For this purpose, GA applies its main operators including crossover and mutation for exploration and exploitation, respectively so that well-prepared operators lead to higher performance for the algorithm.

5.5.5.1. Crossover. In spite of different approaches which have been proposed to implement crossover operator, those that are considered as the most well-known ones including single-point crossover (SPX), double-point crossover (DPX), and uniform crossover (UX), are concerned in this study.

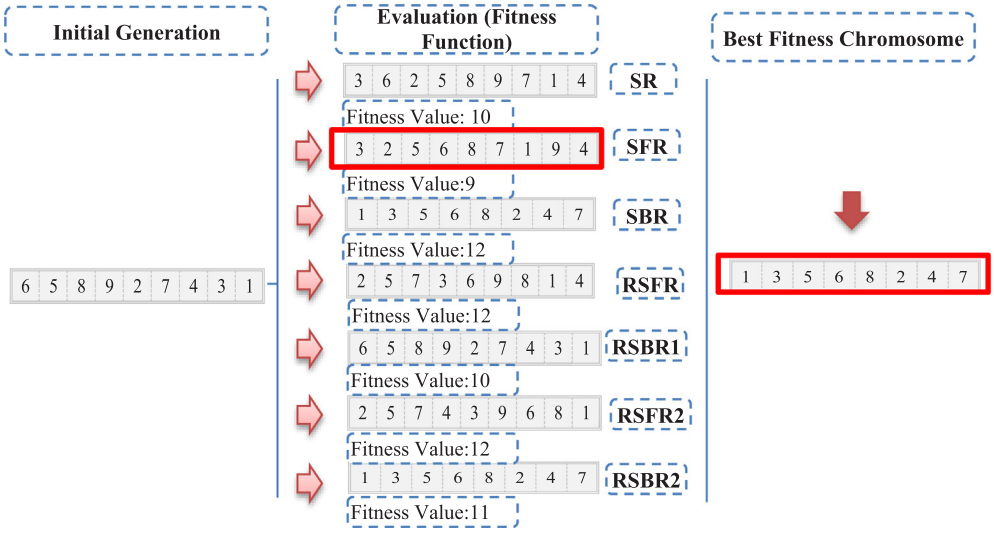

Fig. 13. The process of the proposed repairing mechanism.

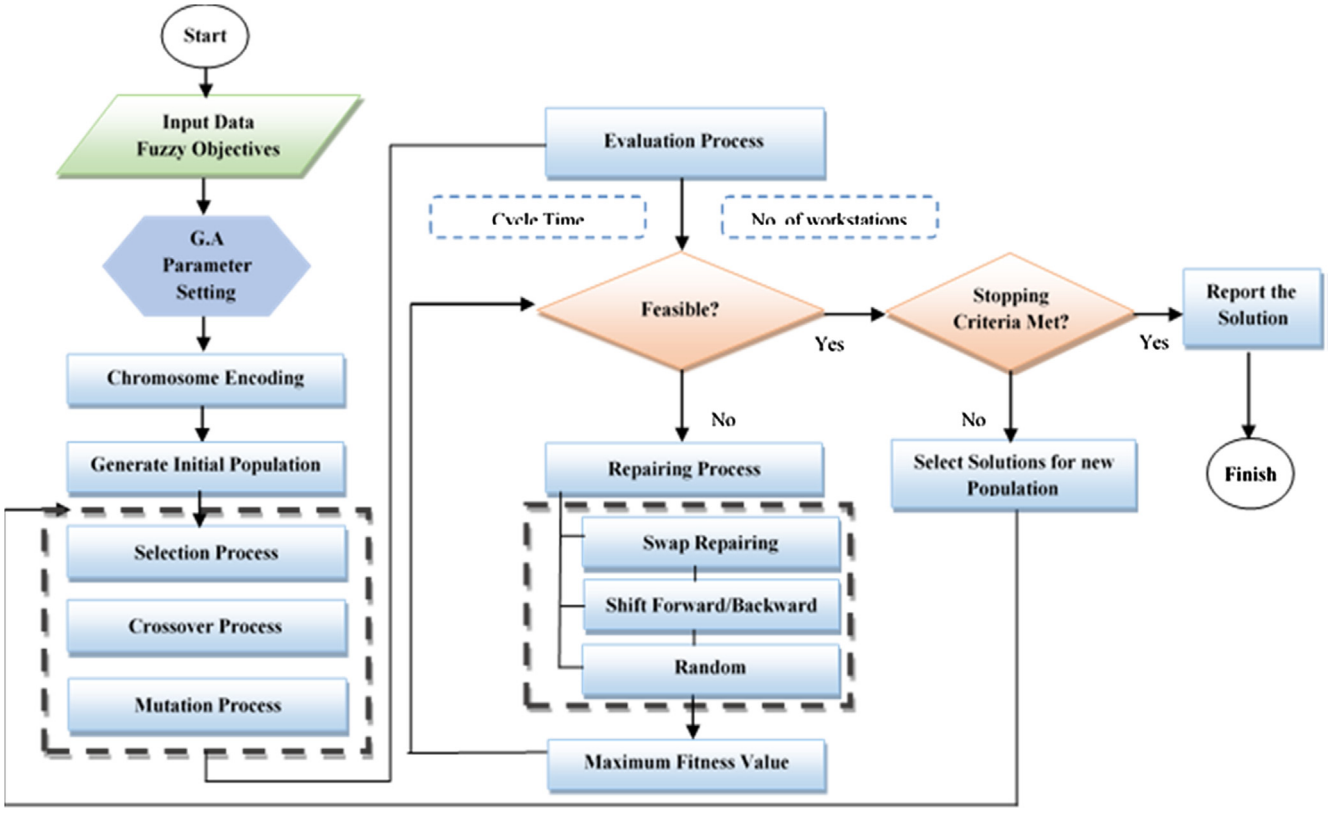

Fig. 14. The proposed GA approaches.

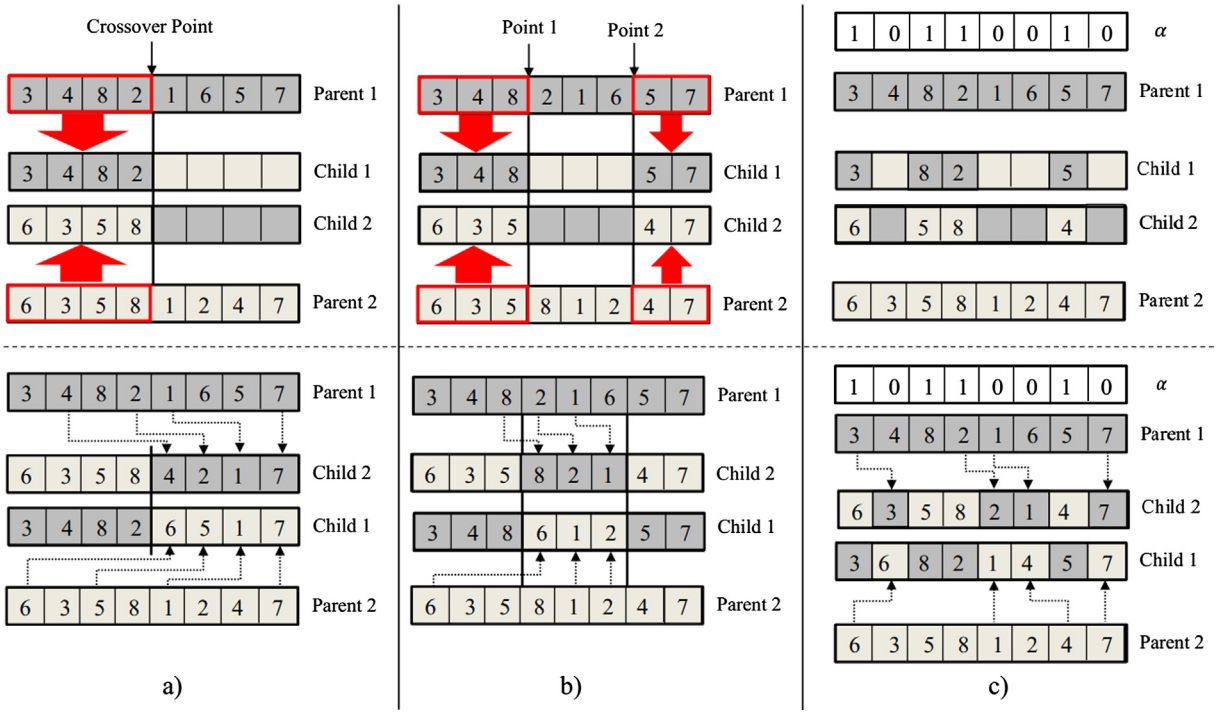

Fig. 15. An example for crossover, (a) SPX, (b) DPX, & (c) UX.

(a) SPX

This crossover operators by dividing parents into two parts by generating a random number (Crossover Point) which can be derived from interval [1, n − 1]. The offspring inherits from both two parts of population distinctively. In this line, while in the first part offspring inherits the first part directly, they do so for the second part indirectly, i.e. they would own a numerical sequence Fig. 15a.

(b) DPX

To implement this crossover operator, the parents are divided into

Fig. 16. An example for mutation, (a) SWM, (b) MSM.

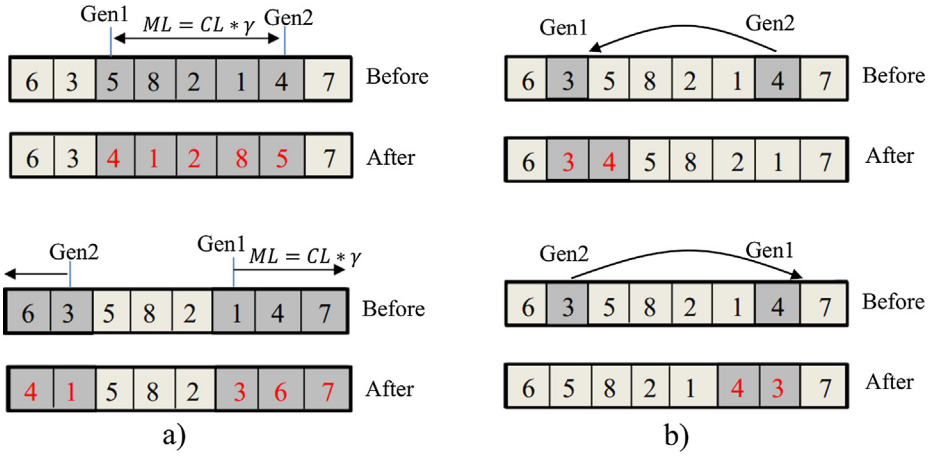

Fig. 17. An example for mutation, (a) RM, (b) SHM.

three parts use of two random numbers (Point1 and Point 2) from the interval [1, n − 1]. Following this approach, offspring inherit their first and third part from one of the parents and the middle from another parent indirectly according to numerical order Fig. 15b.

(c) UX

To deploy this operator, a Boolean valued random array α should be produced which its length is equal to the number of the genes. If $\alpha_i = 1$, then the offspring inherits the corresponding gene directly from one of the parents. Otherwise, when $\alpha_i = 0$ the corresponding gene would be inherited from the other parent indirectly according to the numerical sequence Fig. 15c.

5.5.5.2. Mutation. There have been proposed several methods for the mutation process. Among them, this paper utilized Integer numerical methods including Swap Mutation (SWM), Multiple-Swap Mutation (MSM), Reverse Mutation (RM), and Shift Mutation (ShM).

(a) SWM

Selecting two genes and then changing them as depicted in Fig. 16a.

(b) MSM

This approach is a generalized version of former method in which the SWM operators perform on chromosomes for several times as depicted in Fig. 16b. In this approach, for each MSM, the number of times that SWM performs on chromosomes can be obtained applying formula (51).

$$nSWM = CL/2 \times \gamma \tag{51}$$

wherein Eq. (51), nSWM denotes the number of SWM, CL indicates the chromosome length, and γ refers to the mutation rate (which is between 0 and 1). As it is clear from Eq. (51), the greater γ leads to an increase in exploration, whilst the smaller γ results in to an increase in algorithm exploitation. It should be noted that χ and χ correspond to floor and ceiling of the χ, respectively.

(c) RM

In this approach, first a gene is randomly selected and after that the place of the second gene is determined deploying formula (52), and then these genes are reversed as depicted in Fig. 16c.

$$ML = CL \times \gamma \tag{52}$$

In Eq (52), ML denotes the mutation length where its value varies along with change in γ.

(d) ShM

In this method two genes are selected then the second gene is transmitted into the first neighborhood as depicted in Fig. 17b.

6. Comparison results

6.1. Parameter tuning

There have been proposed several approaches to calibrate parameters of meta-heuristic algorithm where some of them are full factorial design in which all possible combinations are investigated that leads to consume much amount of time and cost (Montgomery, 2008; Ruiz, Maroto, & Alcaraz, 2006). Having treated to this drawback, Taguchi method has been introduced by Taguchi (1986) so that employing a special design of orthogonal arrays, the entire parameters space can be studied with a small number of experiments. However, in contrast to other full factorial approaches, Taguchi method needs less time and cost of experiment (Alavidoost et al., 2014).

6.2. Taguchi method

The Taguchi method organizes the factors into two main classes consist of controllable and uncontrollable ones (noise factors). Because noise factors are uncontrollable, so eliminating them is impossible and

Table 1
Parameters and Their Levels.

Algorithm	Parameter	Symbol	Levels
NSGA-II	Population Size	nPopA	nPopA:100,nPopA:200, nPopA:300
	Number of Iteration	ItrA	ItrA:50, ItrA:100, ItrA:200
	Crossover Rate	βA	βA: 0.7, βA: 0.8, βA: 0.9
	Mutation Rate	γA	γA: 0.1, γA: 0.2, γA: 0.3

Table 2
Selected level for the parameters.

Algorithm	Population Size	Max. number of Iteration	Crossover Rate	Mutation Rate
NSGA-II	nPopA: 100	ItrA: 100	βA: 0.9	γA: 0.3

Table 3
Fuzzy processing time of tasks for test problem 1.

Task No.	Processing time	Task No.	Processing time
1	(4, 5, 6)	5	(3, 4, 5)
2	(3, 4, 6)	6	(2, 3, 4)
3	(6, 7, 8)	7	(1, 2, 3)
4	(2, 3, 4)	8	(2, 3, 4)

unpractical. To cope with this deficiency, the Taguchi approach intends to achieve the best level for controllable factors with regard to robustness criterion. Additionally, along with determining the best level, Taguchi method established a relative relationship between the objective function and effect of each factor (Alavidoost et al., 2015).

Having utilized the Taguchi approach, a single signal-to-noise (S/N) ratio is employed in order to analyze the experimental data as well as to determine optimal combination of factors so that this criterion has to be maximized with respect to that. In this regard, the Related Deviation Index (RDI) is also deployed accompany with S/N ratio, to identify the optimal combination of factors as well. It is should be remarked that the RDI value takes care of deviation from the best existing optimal solution that should be minimized [Eq. (53)].

$$\boldsymbol{RDI} = \frac{|Alg_{sol} - Best_{sol}|}{Max_{sol} - Min_{sol}} \tag{53}$$

In Eq. (53), Alg_{sol} indicates the solution derived from the algorithm, meanwhile $Best_{sol}$ ،Max_{sol} and Min_{sol} denote the best, maximum, and minimum results among the solutions, respectively.

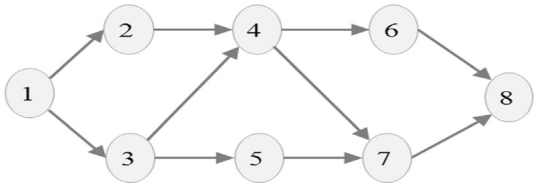

Fig. 19. Precedence diagram of test problem 1.

In light of the presented discussion, the desired level for factor will be established as the one whose means of S/N ratio have the maximum value in comparison with other levels, whilst the means of RDI have minimum value for that level. If these requirement conditions have not been fulfilled for a parameter simultaneously, additional experiment should be performed for it. As can be seen in Table 1, the controllable parameters and their associated levels are provided.

Implementation of the Taguchi method is performed use of *Minitab17* software. In addition, the S/N ratio for parameters of algorithm with their corresponded level are provided in Fig. 18 and Table 2, respectively.

7. Numerical illustration

To evaluate the validity and efficiency of the proposed model and algorithm, a computational study is carried out. For this purpose, the performance of the model is evaluated through 3 test problems suggested in the literature. In this regard, the SALB and SULB problems are solved use of both proposed interactive fuzzy approach as well as the enhanced NSGA-II. In addition, the results obtained by enhanced NSGA-II are compared with those delivered by original NSGA-II. Moreover, a comparison study is conducted between the results obtained from our model and algorithm against those in the benchmarks considered in open literature. It should be remarked that the algorithm parameters are the same in all problems for both original and enhanced NSGA-II.

7.1. Validation study

Before studying the efficiency of the proposed approach, the validation study is performed via test problem 1.

Test Problem 1

The characteristics of the first test problem are provided in Table 3 whilst Fig. 19 illustrates its precedence diagram. It should be remarked that processing time associated with the test problem's tasks are considered as TFN.

The validation process proceeds as follows. First, we solve ALBP

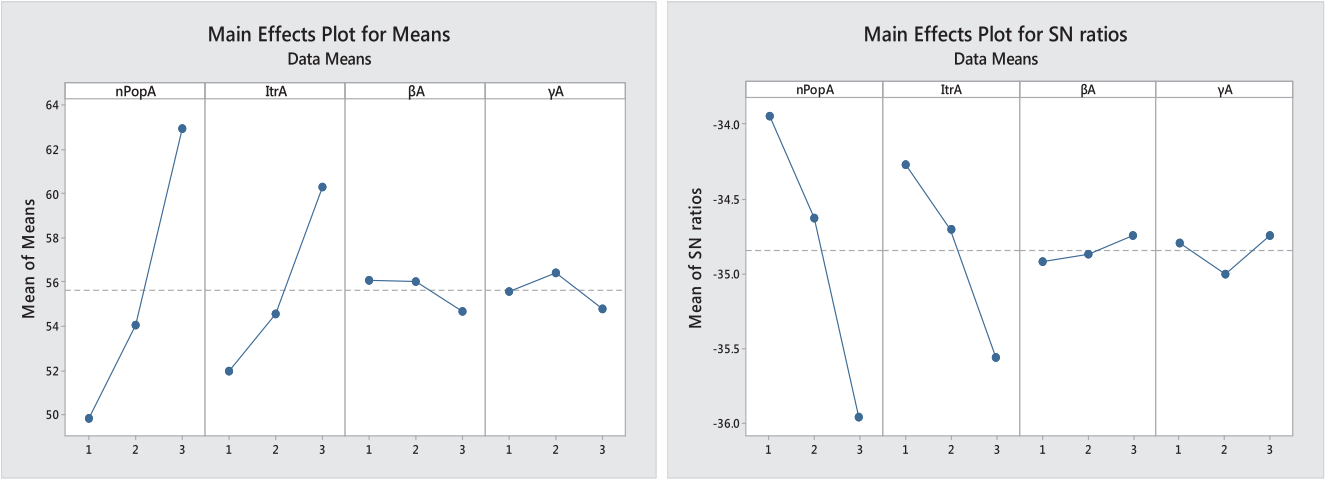

Fig. 18. S/N Ratios and RDI for Each Parameter in NSGA-II.

Table 4
Optimization points for SALB problems.

Algorithm	*WS*	Assigned tasks	Cycle Time	*f*
First NSGA-II point solution for SALB	1	1, 2	9	$f_1 = 1$ $f_2 = 0.5$
	2	3	7	$f_1 = 1$ $f_2 = 0.5$
	3	4, 6	6	$f_1 = 0.5$ $f_2 = 1$
	4	5, 7, 8	9	$f_1 = 0.5$ $f_2 = 1$
Second NSGA-II point solution for SALB	1	1, 2	9	$f_1 = 0.5$ $f_2 = 1$
	2	3	7	$f_1 = 0.5$ $f_2 = 1$
	3	4, 5	7	$f_1 = 1$ $f_2 = 0.5$
	4	6, 7, 8	8	$f_1 = 1$ $f_2 = 0.5$
Third NSGA-II point solution for SALB	1	1	5	$f_1 = 0.5$ $f_2 = 1$
	2	3	7	$f_1 = 0.5$ $f_2 = 1$
	3	2, 4	7	$f_1 = 0.5$ $f_2 = 1$
	4	5, 6	7	$f_1 = 0.5$ $f_2 = 1$
	5	7, 8	5	$f_1 = 1$ $f_2 = 0.5$
Fourth NSGA-II point solution for SALB	1	1	5	$f_1 = 1$ $f_2 = 0.5$
	2	3	7	$f_1 = 0.5$ $f_2 = 1$
	3	2, 5	8	$f_1 = 0.5$ $f_2 = 1$
	4	4, 6, 7	8	$f_1 = 0.5$ $f_2 = 1$
	5	8	3	$f_1 = 0.5$ $f_2 = 1$
Fifth NSGA-II point solution for SALB	1	1	5	$f_1 = 1$ $f_2 = 0.5$
	2	2	4	$f_1 = 1$ $f_2 = 0.5$
	3	3	7	$f_1 = 0.5$ $f_2 = 1$
	4	4, 5	7	$f_1 = 0.5$ $f_2 = 1$
	5	6, 7, 8	8	$f_1 = 0.5$ $f_2 = 1$
Optimal BFGP point solution for SALB	1	1	5	$f_1 = 0.5$ $f_2 = 1$
	2	2	4	$f_1 = 1$ $f_2 = 0.5$
	3	3	7	$f_1 = 1$ $f_2 = 0.5$
	4	4, 5	7	$f_1 = 0.5$ $f_2 = 1$
	5	6, 7, 8	8	$f_1 = 0.5$ $f_2 = 1$

problems through NSGA-II algorithms. Afterwards, the BFGP approach presented by Kara et al. (2009) is treated exactly by Lingo software. The obtained solutions for straight assembly line as well as U-shape one, are presented in Tables 4 and 5.

Regarding to Table 3, rows from first to fifth are related to solutions obtained by NSGA-II fuzzy algorithm whereas sixth row is related to BFGP model. Accordingly, the first two raw are devoted to present the results of applying NSGA-II whereas the third one provides the result of BFGP approach. As shown in the tables, while in traditional methods, classic algorithms should be implemented several times to obtain different solutions, use of the proposed approach leads to produce five Pareto solutions at once. In addition, precise classical methods cannot be used for high scale. Indeed, the presented algorithm is an appropriate approach deal with large scale problems.

Table 5
Optimization points for SULB problems.

Algorithm	*WS*	Assigned tasks	Cycle Time	*f*
First NSGA-II point solution for U-shaped ALB	1	1, 8	8	$f_1 = 1$ $f_2 = 1$
	2	4, 6, 7	8	
	3	3	7	
	4	2, 5	8	
Second NSGA-II point solution for U-shaped ALB	1	6, 7, 8	8	$f_1 = 1$ $f_2 = 1$
	2	1, 4	8	
	3	2, 5	8	
	4	3	7	
Optimal BFGP point solution for U-shaped ALB	1	1, 8	8	$f_1 = 1$ $f_2 = 1$
	2	4, 6, 7	8	
	3	3	7	
	4	2, 5	8	

7.2. Comparison of proposed algorithm and interactive approach

In this section, the performance of applying NSGA-II algorithm is compared against the interactive programming approach by employing test problem offered by using Tsujimura, et al. (1995). In this regard, since the NSGA-II approach leads to several solutions instead of the one outcome of interactive process, some criteria are defined including fuzzy idle time, idle time percentage and fuzzy cycle time enabling to fair decision of DM in selecting most desirable ones. The specifications of test problem 2 are illustrated in the following.

Test Problem 2

The properties of the second test problem are summarized in Table 6 whilst Fig. 20 displays its precedence diagram. (Each node represents a task and the number above, represents the processing time for each

Table 6
Fuzzy processing time of tasks for test problem 2.

Task No.	Processing time	Task No.	Processing time
1	(5 6 8)	7	(15 16 18)
2	(3 5 6)	8	(3 5 6)
3	(7 8 9)	9	(5 7 8)
4	(8 10 11)	10	(11 15 17)
5	(5 7 8)	11	(9 10 11)
6	(16 18 20)	12	(16 18 19)

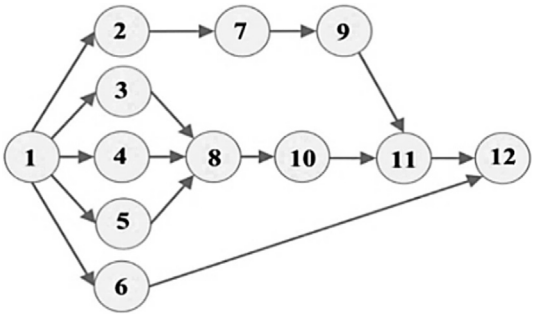

Fig. 20. Precedence diagram of test problem 2.

Table 7
Result of NSGA-II compared to interactive programming for test problem 2.

Test Problem 2			K	Task Assignment	$\widetilde{I}_k$	Fuzzy Idle Percentage	$\widetilde{I}_k$
SALB							
Enhanced NSGA-II	Solution 1		1	1, 2, 4, 5, 3, 8	(1, 9, 20)	(3.9216, 16.6667, 34.0136)	(36, 43, 47)
			2	6, 7, 9	(3, 9, 15)		
			3	10, 11, 12	(2, 7, 15)		
	Solution 2		1	1, 4, 2, 3, 5, 8	(1, 9, 20)	(7.843137, 16.6667, 34.0136)	(11, 15, 21)
			2	7, 10, 6	(0, 1, 9)		
			3	9, 11, 12	(11, 15, 21)		
	Solution 3		1	1, 2, 4, 6	(4, 11, 19)	(26.9608, 37.5,51.53061)	(32, 39, 45)
			2	5, 3, 8, 10	(9, 15, 25)		
			3	7, 9, 11	(12, 17, 22)		
			4	12	(30, 32, 35)		
Interactive Programming			1	1, 4, 2, 3, 5, 8	(1, 9, 20)	(3.92, 16.67, 34.01)	(11, 15, 21)
			2	7, 10, 6	(0, 1, 9)		
			3	9, 11, 12	(11, 15, 21)		
SULB							
Enhanced NSGA-II		Solution 1	1	1, 6, 12	(2, 8, 14)	(3.9216, 16.6667, 34.0136)	(37, 42, 47)
			2	2, 5, 8, 10, 11	(1, 8, 20)		
			3	3, 4, 7, 9	(3, 9, 16)		
		Solution 2	1	1, 5, 4, 2, 7	(0, 6, 15)	(5.228758, 16.6667, 34.0136)	(6, 12, 20)
			2	6, 3, 9, 8	(6, 12, 20)		
			3	10, 11, 12	(2, 7, 15)		
		Solution 3	1	1, 4, 11, 12	(0, 6, 13)	(7.843137, 16.6667, 34.0136)	(12, 17, 24)
			2	2, 3, 5, 7, 8, 9	(0, 2, 13)		
			3	6, 10	(12, 17, 24)		
Interactive Programming			1	1, 5, 4, 2, 7	(36, 44, 51)	(3.92, 16.67, 34.01)	(6, 12, 20)
			2	6, 3, 9, 8	(31, 38, 43)		
			3	10, 11, 12	(36, 43, 47)		

Table 8
Fuzzy processing time of tasks for test problem 3.

Task No.	Processing time	Task No.	Processing time
1	(3, 4, 5)	14	(2, 3, 4)
2	(2, 3, 4)	15	(4, 5, 6)
3	(8, 9, 10)	16	(2, 3, 4)
4	(4, 5, 6)	17	(12, 13, 14)
5	(8, 9, 10)	18	(4, 5, 6)
6	(3, 4, 5)	19	(1, 2, 3)
7	(1, 2, 3)	20	(2, 3, 4)
8	(4, 5, 6)	21	(6, 7, 8)
9	(4, 5, 6)	22	(4, 5, 6)
10	(1, 2, 3)	23	(2, 3, 4)
11	(2, 3, 4)	24	(7, 8, 9)
12	(1, 2, 3)	25	(3, 4, 5)
13	(4, 5, 6)		

Table 9
Fuzzy processing time of tasks for test problem 4.

Task No.	Processing time	Task No.	Processing time
1	(28, 29, 30)	19	(18, 19, 20)
2	(2, 3, 4)	20	(28, 29, 30)
3	(4, 5, 6)	21	(5, 6, 7)
4	(21, 22, 23)	22	(9, 10, 11)
5	(4, 5, 6)	23	(15, 16, 17)
6	(14, 15, 16)	24	(22, 23, 24)
7	(1, 2, 3)	25	(4, 5, 6)
8	(4, 5, 6)	26	(4, 5, 6)
9	(21, 22, 23)	27	(4, 5, 6)
10	(29, 30, 31)	28	(39, 40, 41)
11	(22, 23, 24)	29	(1, 2, 3)
12	(29, 30, 31)	30	(4, 5, 6)
13	(22, 23, 24)	31	(4, 5, 6)
14	(1, 2, 3)	32	(1, 2, 3)
15	(18, 19, 20)	33	(39, 40, 41)
16	(28, 29, 30)	34	(1, 2, 3)
17	(1, 2, 3)	35	(1, 2, 3)
18	(1, 2, 3)		

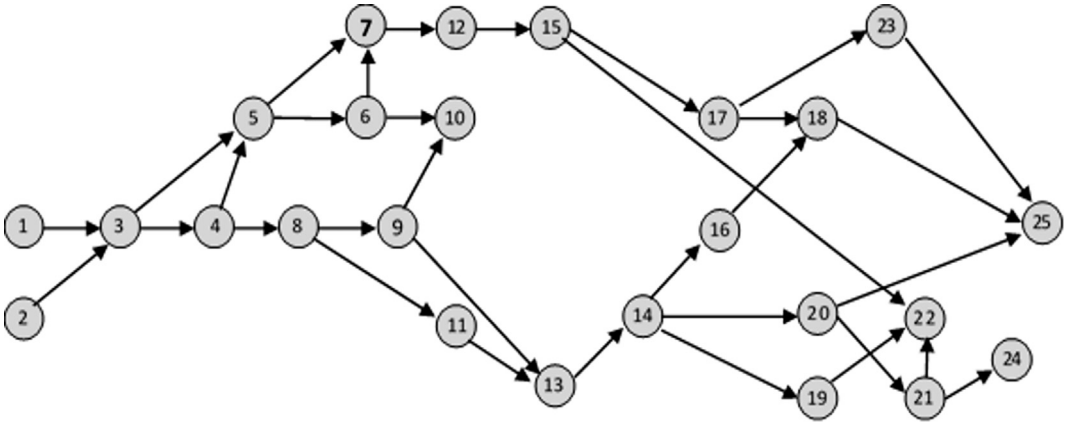

Fig. 21. Precedence diagram of Roszieg problem 3.

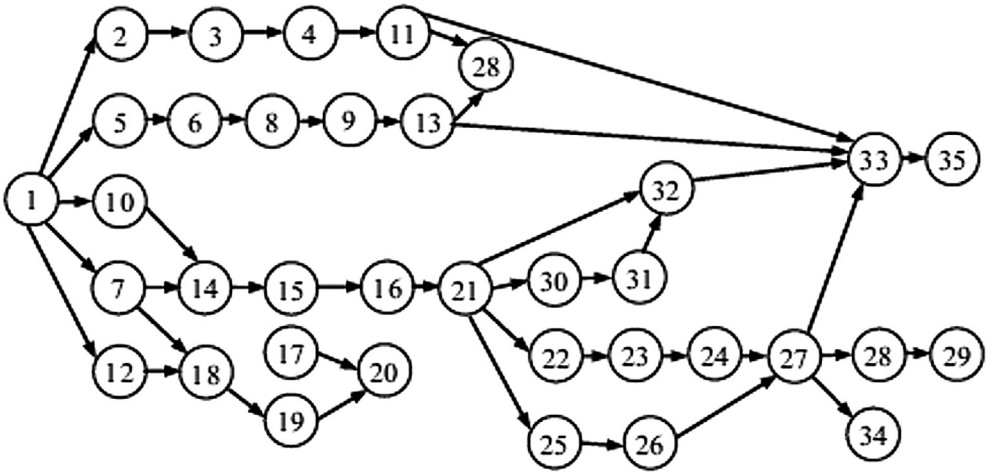

Fig. 22. Precedence diagram of test problem 4.

Table10
Result of best solution obtained by proposed NSGA-II compared to that of delivered by interactive programming for test problems.

Solving Approach	Enhanced NSGA-II (Best Solution)		Interactive Programming	
Test Problem	m	$\tilde{c}$	m	$\tilde{c}$
Test Problem	m	$\tilde{c}$	m	$\tilde{c}$
Test Problem 3				
SALB	8	(15.60, 16.00, 16.40)	8	(15.60, 16.00, 16.40)
	8	(17.60, 18.00, 18.40)	8	(17.60, 18.00, 18.40)
	6	(20.50, 21.00, 21.50)	6	(20.50, 21.00, 21.50)
	6	(23.50, 24.00, 24.50)	6	(24.50, 25.00, 25.50)
	4	(31.40, 32.00, 32.60)	4	(31.40, 32.00, 32.60)
SULB	8	(15.70, 16.00, 16.30)	8	(15.70, 16.00, 16.30)
	7	(16.60, 17.00, 16.40)	7	(17.60, 18.00, 18.40)
	6	(20.60, 21.00, 21.40)	6	(20.60, 21.00, 21.40)
	5	(21.50, 22.00, 22.50)	5	24.50, 25.00, 25.50)
	4	(31.60, 32.00, 32.40)	4	(31.60, 32.00, 32.40)
Test Problem 4				
SALB	7	(73.30, 74.00, 74.70)	7	(79.30, 81.00, 81.70)
	8	(63.20, 64.00, 64.80)	8	(68.20, 69.00, 69.80)
	9	(55.40, 56.00, 56.60)	9	(59.40, 61.00, 61.60)
	11	(47.30, 48.00, 48.70)	11	(48.30, 49.00, 49.70)
	12	(43.40, 44.00, 44.60)	12	(43.40, 44.00, 44.60)
SULB	6	(80.40, 81.00, 81.60)	6	(80.40, 81.00, 81.60)
	8	(61.60, 62.00, 62.40)	7	(68.60, 69.00, 69.40)
	9	(55.40, 56.00, 56.60)	8	(60.40, 61.00, 61.40)
	11	(46.30, 47.00, 47.70)	10	(48.30, 49.00, 49.70)
	12	(42.40, 43.00, 43.60)	12	(43.40, 44.00, 44.60)

Table 11
Comparison between obtained solutions by NSGA-II and optimal ones.

	SALB			SULB		
Benchmarks	Avg%dev	max%dev	%opt	Avg%dev	max%dev	%opt
Test problem 1	0.00	0.00	100	0.00	0.00	100
Test problem 2	0.00	0.00	100	0.00	0.00	100
Test problem 3	0.61	11.3	96.41	0.81	13.18	89.16.
Test problem 4	0.89	14.42	93.39	1.13	15.06	82.29

node). Also, the parameter values for the test problem is provided as follows:

The results of test problem 2, derived by using NSGA-II algorithms as well as those that are provided by fuzzy interactive programming are presented in Table 7. In this paper, the results of interactive programming are considered as the optimum solutions. As can be shown, the solutions produced of SALB and SULB problems by using NSGA-II with 3 solution points are provided in the first 3 rows while the ones corresponds to optimum results are presented in the next row.

Table 12
Selected Benchmarks.

Small		Medium		Large I		Large II	
Benchmark	Size	Benchmark	Size	Benchmark	Size	Benchmark	Size
Mertens	7	Roszieg	25	Kilbridge	45	Tonge	70
Jaeschke	9	Sawyer	30	Hahn	53	Wee-Mag	75
Jackson	11	Gunther	35	Warnecke	58	Arcus1	83

As can be observed from the Table 7, while the interactive method delivers the optimum result as a one point, the proposed approach presents three ones simultaneously which is significantly competitive with those which presented by optimum results. As can be implied from the table, the results from the proposed algorithm will converge to the optimum solution. In addition, along with presenting the optimum solution, NSGA-II offers a range of solutions with could be adopted according to preference of DM with respecting to several criteria including workstation's idle time, fuzzy idle percentage and fuzzy cycle time. The main issue which is demonstrated in this Table 7 is the flexibility of providing result which offered to DM that enables him to select the one which is most compatible with his preference, instead of presenting one optimum result derived by interactive programming.

Test Problem 3

This test problem is a 25-task ALBP proposed by Roszieg where the task processing times as well as the precedence constraints are depicted in Table 8 and Fig. 21, respectively.

Test Problem 4

This test problem is a 35-task ALBP proposed by Gunther where the task processing times as well as the precedence constraints are depicted in Table 9 and Fig. 22, respectively.

The results of medium sized benchmarks which are derived use of NSGA-II algorithms as well as those that are provided by fuzzy interactive programming are presented in Table 10. As can be shown, the solutions produced by using the proposed NSGA-II (included by number of workstations and cycle time) are presented in rows 2–3 while the ones corresponds to optimum results are presented in the next two row.

As can be found in the Table 10, our proposed NSGA-II approach presents high performance in delivering solutions which are very close to optimal solutions derived by interactive method.

To investigate the effectiveness of the proposed NSGA-II approach to deliver qualified solutions for these test problems, their output can be

Table 13
Result of average performance of algorithm over benchmarks.

Algorithms Problem	Size	Optimal Solution n	C_{max}	m^*	Enhanced NSGA-II *E%*	*WID*	*ID*	*SX*
Small	SALB	7	6	6	0.853	2.1E−02	19.44	0.81
			10	3	0.972	2.8E−02	3.33	0.97
			15	2	0.975	1.2E−02	3.33	0.97
		9	7	7	0.841	2.5E−02	24.49	0.76
			8	6	0.966	2.3E−02	22.92	0.77
			18	3	0.971	1.4E−02	31.48	0.73
		11	9	6	0.881	2.4E−02	14.81	0.85
			13	4	0.918	2.8E−02	11.54	0.96
			21	3	0.959	1.3E−02	26.98	0.96
	SULB	7	6	6	0.853	2.1E−02	19.44	0.81
			10	3	0.975	2.6E−02	3.33	0.97
			15	2	0.975	1.2E−02	0.33	0.97
		9	7	7	0.843	1.9E−02	24.49	0.76
			8	6	0.976	2.6E−02	22.92	0.77
			18	3	0.971	1.3E−02	31.48	0.88
		11	9	6	0.881	2.4E−02	14.81	0.85
			13	4	0.924	2.8E−02	11.54	0.96
			21	3	0.961	1.5E−02	26.98	0.96
Medium	SALB	25	14	10	0.898	3.3E−03	10.71	0.89
			18	8	0.98	2.5E−03	13.19	0.98
			25	6	0.998	1.1E−03	16.67	0.99
		30	27	13	0.895	2.1E−03	7.69	0.959
			36	10	0.882	2.7E−03	10.00	0.953
			54	7	0.89	4.3E−03	14.29	0.964
		35	44	12	0.865	3.6E−03	8.52	0.91
			54	9	0.859	3.8E−03	10.56	0.97
			69	8	0.876	5.1E−03	12.50	0.94
	SULB	25	14	9	0.898	9.1E−03	1.79	0.98
			18	7	0.99	4.2E−03	1.79	0.98
			25	5	0.998	2.1E−03	2.44	0.98
		30	27	13	0.895	2.1E−03	7.69	0.923
			36	9	0.881	2.7E−03	10.00	0.926
			54	6	0.89	4.3E−03	14.29	0.945
		35	44	12	0.865	3.6E−03	8.52	0.94
			54	9	0.853	3.9E−03	10.56	0.97
			69	7	0.886	4.5E−03	12.50	0.97
	SALB	45	62	9	0.826	5.1E−03	10.97	0.97
Large I			92	6	0.844	6.2E−03	12.66	0.97
			111	5	0.879	6.6E−04	2.54	0.98
		53	2, 004	8	0.9103	3.5E−03	12.51	1028.8
			2, 806	6	0.8967	4.1E−03	16.69	1368.7
			4, 676	4	0.7896	7.2E−03	25.01	3161.3
		58	56	29	0.806	6.7E−03	13.62	0.864
			60	27	0.821	5.9E−03	11.03	0.905
			92	17	0.814	6.4E−04	11.44	0.926
	SULB	45	62	9	0.826	5.1E−03	1.08	0.98
			92	6	0.848	6.3E−03	1.92	0.98
			111	5	0.879	6.6E−04	2.54	0.98
		53	2, 004	8	0.9529	4.9E−03	12.51	742.6
			2, 806	5	0.9769	5.7E−03	16.69	1149.5
			4, 676	3	0.9999	6.9E−03	0.01	1.41
		58	56	29	0.806	6.7E−03	13.62	0.88
			60	27	0.821	5.9E−03	11.03	0.9
			92	17	0.817	6.6E−04	6.52	0.93
Large II	SALB	70	185	20	0.949	4.4E−04	5.14	59.19
			270	14	0.953	2.3E−03	7.14	92.26
			410	9	0.987	1.2E−03	4.88	60.75
		75	35	60	0.796	8.3E−03	28.62	0.71
			40	60	0.81	7.5E−03	37.54	0.71
			56	30	0.897	8.1E−04	10.77	0.89
		83	6842	12	0.9691	4.1E−03	7.79	0.97
			8412	10	0.9555	7.1E−03	10	0.97
			10,816	8	0.9258	1.3E−04	12.51	0.98
	SULB	70	185	19	0.949	1.1E−04	5.14	99.67
			270	13	0.953	3.9E−03	7.14	84.29
			410	9	0.982	1.8E−03	4.88	64.27
		75	35	60	0.779	8.3E−03	28.62	0.71
			40	60	0.825	7.5E−03	37.54	0.71
			56	30	0.875	1.1E−03	10.77	0.89
		83	6842	12	0.9681	4.3E−03	7.79	0.98
			8412	10	0.9567	3.8E−03	10	0.98
			10,816	7	0.9535	1.6E−03	12.51	0.98

compared with optimal ones. For this purpose, the quality of obtained solutions are determined by the relative deviation metric which can be defined as follows:

$$\text{Deviation} = (\{(x - x^*)/x\} \times 100) \tag{54}$$

A summary of the results obtained by test problems are provided in Table. In the Eq. (58), Avg%dev indicates the Average of deviation, whereas %opt and max%dev represent the optimum solution percentage and maximum percent of deviations, respectively. It should be noted that the optimal solution are extracted from the literature as produced by Scholl (1993).

As can be seen from Table 11, the proposed algorithm can achieve exactly the optimum results in two first test problems which demonstrates the high performance of this algorithm, meanwhile, it can provide near optimal results two last problems. The bigger the solution space, the lower the solutions quality, which is clear Table 12. Compared to SULB, SALB offers better results due to its smaller size of the solution space. All in all, the proposed algorithm presents high performance in solving all test problems.

7.3. Efficiency study

In this section, the performance of applying NSGA-II algorithm is assessed through several benchmarks proposed in the literature. In this respect, four managerial criteria are deployed to account the effectiveness of the proposed algorithm so that it some practical ALB metrics including line efficiency (E) and Workload Deviation (WLD), line Idle time (ID) and Smoothness index (SX) are considered. It should be noted that while the higher value of assembly line efficiency, the values corresponds to other criteria should be minimized, respectively. The following criteria are considered in this section.

$$\text{Assembly Line Efficiency, } E = \sum_{k=1}^{m} t(S_k)/(m \times \max_k\{t(S_k)\}). \tag{55}$$

where $t(S_k) = \sum_{i S_k} t_i$ for $k = 1, \ldots, m$.

$$\text{Work Load Deviation, } WLD = \sqrt{\sum_{k=1}^{m} (U_k - \overline{U})^2/(m)} \tag{56}$$

wherein $U_k = t(S_k)/\max_k\{t(S_k)\}$ and $\overline{U} = \sum_{k=1}^{m} U_k/m$.

$$\text{The line Idle time } ID = \sum_{k=1}^{m} (\widetilde{C}_{max} - \widetilde{t}(S_k))/(m \times \widetilde{C}_{max}) \tag{57}$$

$$\text{The Line Smoothness Index, } SX = \sqrt{\sum_{k=1}^{m} (C_{max} - t(S_k))^2} \tag{58}$$

To further asses the performance of the proposed NSGA-II algorithm, especially to confirm its applicability for practical assembly lines, we implement this approach in several benchmarks. In this regard, the problem is solved over several benchmarks including small, medium and large ones to evaluate performance of our proposed algorithm along with both SALB and SULB problems. The benchmarks are classified according to their problem size so that three classes are defined including small ($n < 25$), medium ($25 \leqslant n < 45$) and large-sized ($n \geqslant 75$) benchmarks, respectively. The considered classes are depicted in Table 12.

More details of these benchmarks are reachable on Scholl (1993) and http://alb.mansci.de.

Considering Table 13, the result of such performance analysis correspond to line efficiency, workload deviation, idle time and smoothness are illustrated. It is remarkable that column m∗ provides the minimum number of workstations derived by employing interactive programming approach as solving approach. It should be noted that since NSGA-II provides several point solutions in instead of one solution, the average of performance metrics are considered to calculate them.

As clear from Table 13, the proposed NSGA-II algorithm delivers more desirable results in all small sized benchmarks which implies high performance of this algorithm. Also it can be observed from Table 8 that the proposed NSGA-II presents a good performance in solving the medium and large size problem which prove its validity in being utilized in ALBPs. In addition, the performance of such algorithm in SALB is slightly better than that of offered in SULB. Overall, given the complexity of the problem as well as its fuzzy nature this indicates a high quality overall performance of the developed algorithm.

8. Conclusion

In real world, determining a precise value of the objective functions for DM is a difficult task due to uncertainty and ambiguity subjected to production systems. In addition, to treat uncertainty associated with in the real world problems, fuzzy numbers have been used to represent the assembly line cycle and processing times. To deal with this problem, in this paper, along with proposing a novel fuzzy model for both *f*-SALB and *f*-SULB with objects to optimize two conflicting objectives simultaneously, TFNs are employed for task processing times. Having dealt with this problem, a novel meta-heuristic method (NSGA-II) is proposed in response to the need of appropriate method for solving the problem. Validity of the model is assessed through benchmark test problem where the obtained result demonstrated high performance opposed to an exact solution procedure employed by benchmark. Finally, in light of the obtained results, the proposed novel algorithm is capable of dealing with several conflicting objective functions associated with assembly line along with offering an efficient framework that enables the DM to consider her point of view in fuzzy environment. At last, although the attention of this study was paid to develop a single-model of ALBP but it is hoped that this direction could be followed by developing much complex problems such as Mixed-Model assembly lines by this algorithm as well as incorporating uncertainty to other model parameters in the future.

References

Ajenblit, D. A., & Wainwright, R. L. (1998). Applying genetic algorithms to the U-shaped assembly line balancing problem. Paper presented at the evolutionary computation proceedings, 1998. IEEE world congress on computational intelligence., The 1998 IEEE international conference on.

Akpınar, Sener, & Mirac Bayhan, G. (2011). A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. *Engineering Applications of Artificial Intelligence, 24*(3), 449–457. http://dx.doi.org/10.1016/j.engappai.2010.08.006.

Al-Zuheri, Atiya, Luong, Lee, & Xing, Ke (2014). Developing a multi-objective genetic optimisation approach for an operational design of a manual mixed-model assembly line with walking workers. *Journal of Intelligent Manufacturing,*, 1–17. http://dx.doi.org/10.1007/s10845-014-0934-3.

Alavidoost, M. H., Babazadeh, H., & Sayyari, S. T. (2016). An interactive fuzzy programming approach for bi-objective straight and U-shaped assembly line balancing problem. *Applied Soft Computing, 40*, 221–235.

Alavidoost, M. H., Fazel Zarandi, M. H., Tarimoradi, Mosahar, & Nemati, Yaser (2014). Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times. *Journal of Intelligent Manufacturing*, 1–24. http://dx.doi.org/10.1007/s10845-014-0978-4.

Alavidoost, M. H., & Nayeri, M. A. (2014). Proposition of a hybrid NSGA-II algorithm with fuzzy objectives for bi-objective assembly line balancing problem. Paper presented at the Tenth international industrial engineering conference.

Alavidoost, M. H., Tarimoradi, Mosahar, & Fazel Zarandi, M. H. (2015). Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems. *Applied Soft Computing, 34*(0), 655–677. http://dx.doi.org/10.1016/j.asoc.2015.06.001.

Alavidoost, M. H., Tarimoradi, M., & Zarandi, M. H. F. (2015). Bi-objective mixed-integer nonlinear programming for multi-commodity tri-echelon supply chain networks. *Journal of Intelligent Manufacturing*, 1–18.

Arcus, L. A. (1966). COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research, 4*, 25–32.

Avikal, Shwetank, Jain, Rajeev, Mishra, P. K., & Yadav, H. C. (2013). A heuristic approach for U-shaped assembly line balancing to improve labor productivity. *Computers & Industrial Engineering, 64*(4), 895–901. http://dx.doi.org/10.1016/j.cie.2013.01.001.

Baudin, Michel (2002). *Lean assembly: The nuts and bolts of making assembly operations flow.* New York, USA: Productivity Press.

Baybars, IlKER (1986a). An efficient heuristic method for the simple assembly line

balancing problem. *International Journal of Production Research, 24*(1), 149–166.
Baybars, Ilker (1986b). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science, 32*(8), 909–932.
Baykasoglu, Adil (2006). Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. *Journal of Intelligent Manufacturing, 17*(2), 217–232.
Becker, Christian, & Scholl, Armin (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715. http://dx.doi.org/10.1016/j.ejor.2004.07.023.
Bellman, Richard E., & Zadeh, Lotfi Asker (1970). Decision-making in a fuzzy environment. *Management Science, 17*(4), B-141–B-164.
Brudaru, O., & Valmar, B. (2004). Genetic algorithm with embryonic chromosomes for assembly line balancing with fuzzy processing times. Paper presented at the The 8th international research/expert conference trends in the development of machinery and associated technology, TMT 2004, Neum, Bosnia and Herzegovina.
Chang, Pei-Chann, Huang, Wei-Hsiu, & Ting, Ching-Jung (2012). Developing a varietal GA with ESMA strategy for solving the pick and place problem in printed circuit board assembly line. *Journal of Intelligent Manufacturing, 23*(5), 1589–1602. http://dx.doi.org/10.1007/s10845-010-0461-9.
Chutima, Parames, & Chimklai, Palida (2012). Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers & Industrial Engineering, 62*(1), 39–55.
Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems, vol. 5*. Springer.
Dar-El, E. M., & Rubinovitch, Y. (1979). Must—A multiple solutions technique for balancing single model assembly lines. *Management Science, 25*(11), 1105–1114. http://dx.doi.org/10.1287/mnsc.25.11.1105.
Dar-El, E. M. (1973). MALB—A heuristic technique for balancing large single-model assembly lines. *AIIE Transactions, 5*(4), 343–356.
Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197. http://dx.doi.org/10.1109/4235.996017.
Deckro, Richard F., & Rangachari, Sarangan (1990). A goal approach to assembly line balancing. *Computers & Operations Research, 17*(5), 509–521. http://dx.doi.org/10.1016/0305-0548(90)90055-C.
Delice, Yılmaz, Aydoğan, Emel Kızılkaya, Özcan, Uğur, & İlkay, MehmetSıtkı (2014). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, 1–14. http://dx.doi.org/10.1007/s10845-014-0959-7.
Dou, Jianping, Li, Jun, & Su, Chun (2013). A novel feasible task sequence-oriented discrete particle swarm algorithm for simple assembly line balancing problem of type 1. *The International Journal of Advanced Manufacturing Technology, 69*(9–12), 2445–2457. http://dx.doi.org/10.1007/s00170-013-5216-2.
El-Wahed, Abd, Waiel, F., & Lee, Sang M. (2006). Interactive fuzzy goal programming for multi-objective transportation problems. *Omega, 34*(2), 158–166.
Erel, E., & Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning and Control, 9*(5), 414–434.
Falkenauer, E., & Delchambre, A. (1992, 12–14 May 1992). A genetic algorithm for bin packing and line balancing. Paper presented at the robotics and automation, 1992. Proceedings., 1992 IEEE international conference on.
Fazel Zarandi, M. H., Tarimoradi, M., Alavidoost, M. H., & Shakeri, B. (2015). Fuzzy approximate reasoning toward Multi-Objective optimization policy: Deployment for supply chain programming. *Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015 Annual Conference of the North American* (pp. 1–6). .
Fazel Zarandi, M. H., Tarimoradi, M., Alavidoost, M. H., & Shirazi, M. A. (2015). Fuzzy Comparison Dashboard for multi-objective evolutionary applications: An implementation in supply chain planning. *Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015 Annual Conference of the North American* (pp. 1–6). .
Fonseca, D. J., Guest, C. L., Elam, M., & Karr, C. L. (2005). A fuzzy logic approach to assembly line balancing. *Mathware & Soft Computing, 12*, 57–74.
Gen, M., Tsujimura, Y., & Li, Y. (1996). Fuzzy assembly line balancing using genetic algorithms. *Computers and Industrial Engineering, 31*(3–4), 631–634.
Ghosh, Soumen, & Gagnon, Roger J. (1989). A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *The International Journal of Production Research, 27*(4), 637–670.
Gökçen, Hadi, & Ağ˘pak, Kürşad (2006Gökçen and Ağpak, 2006Gökçen and Ağpak, 2006). A goal programming approach to simple U-line balancing problem. *European Journal of Operational Research, 171*(2), 577–585.
Gutjahr, Allan L, & Nemhauser, George L. (1964). An algorithm for the line balancing problem. *Management Science, 11*(2), 308–315.
Hamzadayi, Alper, & Yildiz, Gokalp (2013). A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. *Computers & Industrial Engineering, 66*(4), 1070–1084. http://dx.doi.org/10.1016/j.cie.2013.08.008.
Hazır, Öncü, & Dolgui, Alexandre (2015). A decomposition based solution algorithm for U-type assembly line balancing with interval data. *Computers & Operations Research, 59*, 126–131. http://dx.doi.org/10.1016/j.cor.2015.01.010.
Helgeson, W. B., & Birnie, D. P. (1961). Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering, 12*(6), 394–398.
Van Hop, Nguyen (2006). A heuristic solution for fuzzy mixed-model line balancing problem. *European Journal of Operational Research, 168*(3), 798–810.
Hwang, Rea Kook, Katayama, Hiroshi, & Gen, Mitsuo (2008). U-shaped assembly line balancing problem with genetic algorithm. *International Journal of Production Research, 46*(16), 4637–4649.
Hwang, ReaKook, & Katayama, Hiroshi (2009). A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. *International Journal of Production Research, 47*(14), 3797–3822.
Jian-sha, L., Ling-ling, J., & Xiu-lin, L. (2009). Hybrid particle swarm optimization algorithm for assembly line balancing problem-2. Paper presented at the industrial engineering and engineering management, 2009. IE&EM'09. 16th international conference on.
Kalayci, Can B, & Gupta, Surendra M. (2013). A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. *The International Journal of Advanced Manufacturing Technology, 69*(1–4), 197–209.
Kao, Edward P. C. (1976). A preference order dynamic program for stochastic assembly line balancing. *Management Science, 22*(10), 1097–1104. http://dx.doi.org/10.1287/mnsc.22.10.1097.
Kara, Y., Paksoy, T., & Chang, C. T. (2009). Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. *European Journal of Operational Research, 195*(2), 335–347. http://dx.doi.org/10.1016/j.ejor.2008.01.003.
Kazemi, Seyed Mahmood, Ghodsi, Reza, Rabbani, Masoud, & Tavakkoli-Moghaddam, Reza (2011). A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. *The International Journal of Advanced Manufacturing Technology, 55*(9–12), 1111–1122.
Kim, Yeo Keun, Song, Won Seop, & Kim, Jun Hyuk (2009). A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers & Operations Research, 36*(3), 853–865.
Klir, G. J., & Yuan, B. (1995). Fuzzy sets and fuzzy logic: Theory and applications.
Kremer, O. (2010). A review of constraint-handling techniques for evolution strategies. *Applied Computational Intelligence and Soft Computing, 2010*, 11. http://dx.doi.org/10.1155/2010/185063 Article ID 185063.
La Scalia, Giada (2013). Solving type-2 assembly line balancing problem with fuzzy binary linear programming. *Journal of Intelligent and Fuzzy Systems, 25*(3), 517–524. http://dx.doi.org/10.3233/IFS-120656.
Lapierre, Sophie D, Ruiz, Angel, & Soriano, Patrick (2006). Balancing assembly lines with Tabu search. *European Journal of Operational Research, 168*(3), 826–837.
Li, Dashuang, Zhang, Chaoyong, Shao, Xinyu, & Lin, Wenwen (2014). A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. *Journal of Intelligent Manufacturing*, 1–15. http://dx.doi.org/10.1007/s10845-014-0919-2.
Manavizadeh, Neda, Hosseini, Nilufar-sadat, Rabbani, Masoud, & Jolai, Fariborz (2013). A Simulated Annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach. *Computers & Industrial Engineering, 64*(2), 669–685. http://dx.doi.org/10.1016/j.cie.2012.11.010.
Monden, Yasuhiro (2012). *Toyota production system: An integrated approach to just-in-time*. CRC Press.
Montgomery, Douglas C. (2008). *Design and analysis of experiments*. John Wiley & Sons.
Nearchou, Andreas C. (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics, 129*(2), 242–250. http://dx.doi.org/10.1016/j.ijpe.2010.10.016.
Nemati, Y., & Alavidoost, M. H. (2018). A fuzzy bi-objective MILP approach to integrate sales, production, distribution and procurement planning in a FMCG supply chain. *Soft Computing*.
Nemati, Y., Madhoshi, M., & Ghadikolaei, A. S. (2017). The effect of Sales and Operations Planning (S&OP) on supply chain's total performance: A case study in an Iranian dairy company. *Computers & Chemical Engineering, 104*, 323–338.
Nemati, Y., Madhoushi, M., & Safaei Ghadikolaei, A. (2017). Towards supply chain planning integration: uncertainty analysis using fuzzy mathematical programming approach in a plastic forming company. *Iranian Journal of Management Studies, 10*, 335–364.
Özbakır, Lale, & Tapkan, Pınar (2010). Balancing fuzzy multi-objective two-sided assembly lines via Bees Algorithm. *Journal of Intelligent and Fuzzy Systems, 21*(5), 317–329.
Özcan, Uğur, & Toklu, Bilal (2009). A Tabu search algorithm for two-sided assembly line balancing. *The International Journal of Advanced Manufacturing Technology, 43*(7–8), 822–829.
Peterson, C. (1993). A tabu search procedure for the simple assembly line balancing problem. Paper presented at the The proceedings of the decision science institute conference.
Purnomo, Hindriyanto Dwi, Wee, Hui-Ming, & Rau, Hsin (2013). Two-sided assembly lines balancing with assignment restrictions. *Mathematical and Computer Modelling, 57*(1–2), 189–199. http://dx.doi.org/10.1016/j.mcm.2011.06.010.
Ruiz, Rubén, Maroto, Concepción, & Alcaraz, Javier (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega, 34*(5), 461–476.
Sabuncuoglu, Ihsan, Erel, Erdal, & Alp, Arda (2009). Ant colony optimization for the single model U-type assembly line balancing problem. *International Journal of Production Economics, 120*(2), 287–300. http://dx.doi.org/10.1016/j.ijpe.2008.11.017.
Salveson, Melvin E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering, 6*(3), 18–25.
Scholl, Armin, & Becker, Christian (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*(3), 666–693. http://dx.doi.org/10.1016/j.ejor.2004.07.022.
Shahrasbi, A., Shamizanjani, M., Alavidoost, M. H., & Akhgar, B. (2017). An aggregated fuzzy model for the selection of a managed security service provider. *International Journal of Information Technology & Decision Making, 16*, 625–684.
Shermeh, H. E., Najafi, S. E., & Alavidoost, M. H. (2016). A novel fuzzy network SBM model for data envelopment analysis: A case study in Iran regional power companies. *Energy, 112*, 686–697.

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology, 73*(9–12), 1665–1694. http://dx.doi.org/10.1007/s00170-014-5944-y.

Taguchi, Genichi Asian, & Organization, Productivity (1986). *Introduction to quality engineering: Designing quality into products and processes.* Tokyo: Asian Productivity Organization.

Tarimoradi, Mosahar, Alavidoost, M. H., & Fazel Zarandi, M. H. (2015). Comparative Corrigendum note on papers "Fuzzy adaptive GA for multi-objective assembly line balancing" continued "Modified GA for different types of assembly line balancing with fuzzy processing times": Differences and similarities [Appl. Soft Comput. 34 (September 2015) 655–677]. *Applied Soft Computing, 35*, 786–788. http://dx.doi.org/10.1016/j.asoc.2015.07.041.

Tasan, SerenOzmehmet, & Tunali, Semra (2008). A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing, 19*(1), 49–69. http://dx.doi.org/10.1007/s10845-007-0045-5.

Toklu, Bilal, & özcan, Uğur (2008). A fuzzy goal programming model for the simple U-line balancing problem with multiple objectives. *Engineering Optimization, 40*(3), 191–204. http://dx.doi.org/10.1080/03052150701651642.

Tsujimura, Y., Gen, M., & Kubota, E. (1995). Solving fuzzy assembly-line balancing problem with genetic algorithms. *Computers and Industrial Engineering, 29*(1–4), 543–547.

Yu, Jianfeng, & Yin, Yuehong (2010). Assembly line balancing based on an adaptive genetic algorithm. *The International Journal of Advanced Manufacturing Technology, 48*(1–4), 347–354.

Yuan, Biao, Zhang, Chaoyong, & Shao, Xinyu (2013). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. *Journal of Intelligent Manufacturing,* 1–10. http://dx.doi.org/10.1007/s10845-013-0770-x.

Zacharia, P. Th., & Nearchou, Andreas C. (2012). Multi-objective fuzzy assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing, 23*(3), 615–627.

Zacharia, P. Th., & Nearchou, Andreas C. (2013). A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem. *Computers & Operations Research, 40*(12), 3033–3044. http://dx.doi.org/10.1016/j.cor.2013.07.012.

Zha, Jing, & Yu, Jian-jun (2014). A hybrid ant colony algorithm for U-line balancing and rebalancing in just-in-time production environment. *Journal of Manufacturing Systems, 33*(1), 93–102. http://dx.doi.org/10.1016/j.jmsy.2013.08.002.

Zhang, Wenqiang, & Gen, Mitsuo (2011). An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. *Journal of Intelligent Manufacturing, 22*(3), 367–378.

Zhang, Ze Qiang, & Cheng, Wen Ming (2010). Solving fuzzy U-shaped line balancing problem with exact method. *Applied Mechanics and Materials, 26*, 1046–1051.

Zhang, Z., Cheng, W., Song, L., & Yu, Q. l. (2009). A heuristic approach for fuzzy U-shaped line balancing problem. Paper presented at the sixth international conference on fuzzy systems and knowledge discovery, 2009. FSKD'09.

Zimmermann, H.-J. (1978). Fuzzy programming and linear programming with several objective functions. *Fuzzy sets and systems, 1*(1), 45–55.

Zitzler, Eckart (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63.* Shaker Ithaca.