Journal of Manufacturing Systems 37 (2015) 672–682

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm

Can B. Kalayci[a,1], Arif Hancilar[a], Askiner Gungor[a,*], Surendra M. Gupta[b,2]

[a] Department of Industrial Engineering, Pamukkale University, Kinikli Kampusu, 20070 Denizli, Turkey
[b] 334 SN, Department of Mechanical and Industrial Engineering, Northeastern University, 360 Huntington Avenue, Boston, MA 02115, USA

ARTICLE INFO

Article history:
Received 27 May 2014
Received in revised form 8 October 2014
Accepted 24 November 2014
Available online 18 December 2014

Keywords:
Fuzzy disassembly line balancing
Combinatorial optimization
Metaheuristics
Artificial bee colony
Variable neighborhood search

ABSTRACT

This paper presents a fuzzy extension of the disassembly line balancing problem (DLBP) with fuzzy task processing times since uncertainty is the main character of real-world disassembly systems. The processing times of tasks are formulated by triangular fuzzy membership functions. The balance measure function is modified according to fuzzy characteristics of the disassembly line. A hybrid discrete artificial bee colony algorithm is proposed to solve the problem whose performance is studied over a well-known test problem taken from open literature and over a new data set introduced in this study. Furthermore, the influence of the fuzziness on the computational complexity of HDABC is evaluated and the solution quality of the proposed algorithm is compared against discrete and traditional versions of the artificial bee colony algorithm. Computational comparisons demonstrate the superiority of the proposed algorithm.

1. Introduction

At a time when economic and ecological resources are rapidly depleted, product recovery that aims to minimize the amount of waste sent to landfills is gaining a lot of importance. Comprehensive reviews of environmentally conscious manufacturing and product recovery can be found in [1,2]. Product recovery consists of several steps in which the first crucial step is disassembly [3]. Disassembly is a methodical extraction of valuable parts [4] from discarded products through a series of operations. Some of the objectives of disassembly are given as follows [5]: recovery of valuable parts and subassemblies, removal of hazardous parts, extraction of parts from the remainder of the product which can be sent to inventory for future use, achieving environmentally friendly manufacturing standards, retrieval of parts or subassemblies of discontinued products to satisfy a sudden demand for these parts, decreasing the amount of residue to be sent to landfills. Thus the design of an efficient disassembly line has a considerable industrial and environmental importance.

Disassembly Line Balancing Problem (DLBP) is a multi-objective problem as described in [6] and has been mathematically proven to be NP-complete in [7] making the goal to achieve the optimal balance computationally expensive. NP-complete or NP-hard terms are the ways of showing that certain classes of problems are not solvable in realistic time [8]. Exponential time complexity of exhaustive search limits its application to large sized instances although it works well in obtaining optimal solutions for small sized instances. An efficient search method, therefore, needs to be employed to attain an (near) optimal solution. The ways of approaching to this combinatorial optimization problem in the literature can be divided into two categories; mathematical programming techniques [9–12] and metaheuristics [7,13–24]. Metaheuristics are more popular since the problem quickly becomes unsolvable with mathematical programming techniques for a practical sized problem. See [25] for more information on DLBP.

Disassembly operations have unique characteristics and cannot be considered as the reverse of assembly operations. Please see [17] for details. Although there are significant differences between assembly line balancing and disassembly line balancing problems, disassembly line balancing literature has been building up on top of the assembly line balancing literature since there are also some similarities between them. The literature of fuzziness in assembly line balancing is of primary interest of this study. The fuzziness in the literature of assembly line balancing was first mentioned in [26,27] by solving a simple fuzzy assembly line balancing

* Corresponding author. Tel.: +90 258 296 3141; fax: +90 258 296 3262.
E-mail addresses: cbkalayci@pau.edu.tr (C.B. Kalayci), arif.hancilar@gmail.com (A. Hancilar), askiner@pau.edu.tr (A. Gungor), gupta@neu.edu (S.M. Gupta).
[1] Tel.: +90 258 296 3209; fax: +90 258 296 3262.
[2] Tel.: +1 617 373 4846; fax: +1 617 373 2921.

http://dx.doi.org/10.1016/j.jmsy.2014.11.015
0278-6125/

problem applied to a single product example. A fuzzy mixed model line balancing problem with a fuzzy binary linear programming model was considered in [28] using a heuristic solution approach that deals with fuzzy processing times. Fuzzy multi objective two-sided assembly line balancing problem was solved using a bees algorithm in [29] with three fuzzy goals: maximize the work slackness index, minimize the total balance delay and maximize the line efficiency. Genetic algorithms were used to solve multi objective fuzzy assembly line balancing problem type 2 [30] and fuzzy assembly line balancing problem type E [31].

In disassembly, there is a high degree of uncertainty in the structure and the quality of the returned products. The structural unknowns of end-of-life (EOL) products may create other uncertainties, such as estimation of disassembly task time of each part due to both machine and human factors. To the best of our knowledge, there is only one previously published work [32] that deals with fuzziness in mixed model disassembly line balancing problem using binary fuzzy goal programming. Yet, in this study, in order to deal with imprecise data, fuzzy numbers are introduced to represent the processing time of each job. Thus, a fuzzy disassembly line balancing model is obtained. The consideration of fuzziness for the solution of disassembly line balancing problem is of immense interest since the data obtained from more realistic situations are imprecise and uncertain. Aiming to fill this gap, this paper introduces a new hybrid discrete artificial bee colony algorithm (HDABC) for solving the multi objective fuzzy disassembly line balancing problem (FDLBP). The fuzzy processing times of disassembly tasks are represented by triangular fuzzy membership functions. Two different evaluation mechanisms are used in the proposed algorithm: lexicographic method that focuses on each objective according to predefined priorities and fixed weighted method that tries to optimize each conflicting objective concurrently with the hope of constructing an efficient Pareto frontier. The performance of HDABC is compared against discrete and traditional versions of the artificial bee colony algorithm on two different cases. Moreover, the influence of the fuzziness on the computational complexity of HDABC is evaluated.

2. Fuzzy disassembly line balancing model

2.1. Notation

π	a solution in VNS
$\widetilde{BD}$	fuzzy balance delay time
$\widetilde{BE}$	fuzzy balance efficiency
$c_{\max}$	maximum cycle time
$\tilde{c}$	fuzzy cycle time
d_i	demand quantity of part i requested
f_s	the nectar amount (i.e., the fitness value) of a food source (s) in the ABC algorithm
g	current iteration (generation) number of the algorithm
h_i	binary value; 1 if part i is hazardous, else 0.
i	part identification, task count (1, . . ., n)
imp	non-improved iteration count
IP	set (i, j) of parts such that task i must precede task j
j	part identification, task count (1, . . ., n)
k	workstation count (1,. . .,m)
LB_j	lower bound for dimension j
m	number of workstations required for a given solution sequence
n	number of parts for removal (dimension of the problem)
P_s	the selection probability of a food source (s)
ps	population size of the ABC algorithm
PS_i	ith part in a solution sequence
$r_{0,1}$	a uniformly distributed real number in [0,1]
$r_{-1,1}$	a uniformly distributed real number in [−1, 1]
s	a solution (bee) count (1, . . ., ps)
$\widetilde{SI}$	smoothness index
$\widetilde{ST_j}$	total fuzzy station time; the sum of processing time at workstation j
t_i	part removal time of part i
UB_j	upper bound for dimension j
x_{jk}	binary value; 1 if part j is assigned to station k, 0 otherwise.
y_{ij}	binary value; 1 if task i is executed after task j, 0 otherwise.

2.2. Problem formulation

Assumptions of multi objective fuzzy disassembly line balancing problem are given as follows: there is only one type of product in the disassembly line; complete disassembly is performed; part removal times are known and assumed to be fuzzy integer; hazardous parts are known in advance; and demand is also known in advance and its quantity is deterministic.

Mathematical formulation of the multi-objective DLBP was first introduced in [7] as a deterministic model. Based on the main concepts of [7], a fuzzy extension of the problem is formulized considering fuzzy part removal times and fuzzy cycle time as follows:

$$\min f_1 = m \tag{1}$$

$$\min f_2 = \sqrt{\sum_{k=1}^{m} (\widetilde{c} - \widetilde{ST_k})} \tag{2}$$

$$\min f_3 = \sum_{i=1}^{n} i \times h_{PS_i}, \quad h_{PS_i} = \begin{cases} 1 & \text{hazardous} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\min f_4 = \sum_{i=1}^{n} i \times d_{PS_i}, \quad d_{PS_i} \in N, \quad \forall PS_i \tag{4}$$

Subject to:

$$\sum_{k=1}^{m} x_{jk} = 1, \quad j = 1, \ldots, n \tag{5}$$

$$\left\lceil \frac{\sum_{i=1}^{n} t_i}{\widetilde{c}} \right\rceil \le n \tag{6}$$

$$\sum_{k=1}^{m} \widetilde{ST} \le \widetilde{c} \tag{7}$$

$$x_{ik} \le \sum_{k=1}^{m} x_{jk}, \quad \forall (i, j) \in IP \tag{8}$$

Objective functions; (1) aims to minimize the number of workstations, (2) balances the smoothness index ($\widetilde{SI}$) which shows the balance of workload distributed among the stations, (3) aims to remove hazardous component as early as possible whereas (4) aims to handle a high demand component as early as possible. Equation (5), guarantees that all tasks are assigned to exactly one workstation. Constraints; (6) guarantees that the number of work stations with a workload does not exceed the permitted number, (7) guarantees that the work content of workstation cannot exceed the cycle time, (8) guarantees that all the disassembly precedence relationship between parts should be satisfied.

In addition to the main objectives, we evaluate the performance of disassembly line in a fuzzy environment using two measures. The first measure is called *the balance efficiency*, i.e., $\widetilde{BE}$ as in Eq. (9). Another measure is the *balance delay time*, i.e., $\widetilde{BD}$ as in Eq. (10). Balance delay time ($\widetilde{BD}$) shows the unused capacity of the line. This measure is obtained by summing the idle time of all the stations.

$$\widetilde{BE} = \frac{\sum_{k=1}^{m} \widetilde{ST_k}}{\mathrm{NWS} \times \widetilde{c}} \tag{9}$$

$$\widetilde{BD} = \sum_{k=1}^{m} (\widetilde{c} - \widetilde{ST_k}) \tag{10}$$

2.3. Fuzzification of the disassembly line balancing problem

In this study, the purpose of using fuzzy data approach is to represent more realistic situations, in other words, to deal with uncertainty. The fuzziness of data is represented by triangular fuzzy numbers as in [30,31]. Triangular fuzzy numbers arithmetic is performed as follows where α and β represent fuzzy numbers:

$$\begin{aligned}
&\widetilde{A} + \widetilde{B} = (\alpha_1 + \beta_1, \alpha_2 + \beta_2, \alpha_3 + \beta_3) \\
&\widetilde{A} - \widetilde{B} = (\alpha_1 - \beta_3, \alpha_2 - \beta_2, \alpha_3 - \beta_1) \\
&\widetilde{A} \times \widetilde{B} = (\alpha_1 \cdot \beta_1, \alpha_2 \cdot \beta_2, \alpha_3 \cdot \beta_3) \\
&\frac{\widetilde{A}}{\widetilde{B}} = \left(\frac{\alpha_1}{\beta_3}, \frac{\alpha_2}{\beta_2}, \frac{\alpha_3}{\beta_1}\right)
\end{aligned} \tag{11}$$

To compare the fuzzy numbers, the greatest associate ordinary number ranking method is used as follows:

$$F(\widetilde{A}) = \frac{\alpha_1 + 2\alpha_2 + \alpha_3}{4} \tag{12}$$

The objective is to determine the set of Pareto optimal solutions based on the ranked values of the fuzzy criteria.

3. Artificial bee colony algorithm

Swarm intelligence is a research branch of artificial intelligence that aims to bring solutions to the problems by modeling the population of interacting agents or swarms that are able to self-organize. The agents are able to exchange information in order to share experiences, and the performance of the overall swarm emerges from the collection of the simple agents' interactions and actions [33]. An ant colony, a flock of birds, flying particles, immune system or a bee colony are typical examples of a swarm system. Artificial Bee Colony (ABC) Algorithm that was originally proposed for continuous function optimization problems by [34] is an optimization algorithm based on the intelligent behavior of honey bee swarm. The main steps of the basic ABC algorithm based on [35] are shown in Table 1.

Table 1
Main steps of the basic ABC algorithm.

Step	Description
1	Initialize the population
2	Evaluate the solutions
3	Employed bees phase
4	Calculate probabilities for onlooker bees
5	Onlooker bees phase
6	Scout bees phase
7	Memorize the best solution achieved so far
8	If maximum cycle number is not reached, go to Step 3
9	Stop

A possible optimum solution to the problem is represented by a food source and the quality of the solution is defined by the nectar amount found in that food source. There are three steps in each cycle as follows: releasing the employed and onlooker bees onto the food sources, calculating their nectar amounts and sending a determined number of scout bees randomly onto the possible food sources [36]. In other words, employed, onlooker and scout bees are the main mechanisms of the ABC algorithm to generate a set of feasible solutions, exploit these solutions and explore new solutions, respectively. If a solution is not improved by a pre-set number of trials, then employed bee leaves that food source and the employed bee becomes a scout in order to explore new areas of the search space. In the ABC algorithm, onlookers and employed bees carry out the exploitation process in the search space while the scouts control the exploration process. For further details of the basic algorithm the readers unfamiliar with the ABC algorithm are referred to [34] which introduced the original algorithm. Based on [35], the detailed steps of the basic ABC algorithm are given as follows:

At the first step, ABC generates a randomly distributed initial food source positions. The initial population of solutions is filled with ps number of randomly generated n-dimensional real-valued vectors that corresponds to food source positions. Let X_i $\{x_{i1}, x_{i2}, \ldots, x_{in}\}$ represent the ith food source in the population and each food source is generated according to Eq. (14) as follows:

$$x_{sj} = LB_j + (UB_j - LB_j) \times r_{-1,1} \quad \text{for} \quad j = 1, 2, \ldots n \quad \text{and} \quad s = 1, 2, \ldots \text{ps} \tag{14}$$

In the employed bees phase, each employed bee generates a new food source in the neighborhood of its present position according to equation (15) as follows:

$$x_{\text{new}(j)} = x_{sj} + (x_{sj} - x_{pj}) \times r_{-1,1} \quad \text{for} \quad p = 1, 2, \ldots, \text{ps} \quad \text{and} \quad j = 1, 2, \ldots, n \tag{15}$$

Once a new solution is obtained, it will be evaluated and compared to the previous solution. If the fitness of the new solution is equal to or better than that of previous solution, new solution will replace the previous solution and become a new member of the population; otherwise previous solution is retained. Thus, a greedy selection mechanism is employed between the old and new candidate solutions.

In the onlooker bee phase, the nectar information gathered from employed bee phase is evaluated by an onlooker bee and a food source is then selected according to its probability value calculated using the following equation.

$$p_s = \frac{f_s}{\sum_{s=1}^{ps} f_s} \tag{16}$$

In short, a roulette wheel selection mechanism is applied to select a food source. Once the onlooker has selected the food source, it produces a modification using Eq. (15). As in the case of the employed bee phase, a replacement of the food source takes place if the modified food source has a better or equal nectar amount than the previous one, and it finally becomes a new member in the population.If a solution is not improved by a pre-set number of trials limit, the food source is left out, and the corresponding employed bee becomes a scout. The scout produces a new solution randomly according to the following equation.

$$x_{sj} = LB_j + (UB_j - LB_j) \times r_{0,1} \quad \text{for} \quad j = 1, 2, \ldots n \quad \text{and} \quad s = 1, 2, \ldots \text{ps} \tag{17}$$

4. The proposed HDABC algorithm

4.1. Solution representation

The original ABC algorithm is designed for continuous optimization problems. However, standard continuous encoding scheme of ABC cannot be directly used to solve the FDLBP. In order to apply ABC to FDLBP, constructing a direct relationship between the task sequence and the vector of individuals in ABC is required. The permutation based representation is used as in [26] to encode each solution in HDABC algorithm.

4.2. Population initialization and solution construction strategy

Initial population is generated randomly for the HDABC algorithm. A solution construction is started with opening the first workstation. A station oriented assignment procedure based on [14] is used to fill next workstations with remaining tasks. In this procedure, first, a set of candidate tasks is identified. A task is inserted into the set if it satisfies three criteria: first, it has not already been assigned to one or more of the previous workstations, second, its predecessors has already been completed and third, its fuzzy task time should be less than or equal to the remaining available fuzzy time of the workstation under consideration. Then, a task is randomly selected from the candidate task set and assigned to the workstation. If there is no candidate task for that workstation, a new workstation is opened with a fuzzy cycle time. This process continues until all tasks are assigned to a number of workstations.

4.3. Neighborhood structures

In this study, a number of neighborhood structures [insert, swap, two point right, one point right and one point left operators] are employed. Each neighborhood structure changes the initial configuration. Insert operator (N1) randomly selects a position of a task and inserts it to another randomly selected position in the disassembly sequence. Swap operator (N2) randomly selects two tasks in the line sequence, swaps their positions and returns a new feasible solution. Two point right operator (N3) randomly selects two cut positions in the disassembly sequence, reconstructs a new sub-sequence within these cut positions while not changing the positions of tasks located out of the two selected cut positions. One point right operator (N4) randomly selects a cut position in the disassembly sequence, reconstructs a new sequence starting from this position according to the solution construction strategy while keeping the positions of tasks as already assigned before the cut position. One point left operator (N5) randomly selects a cut point in the disassembly sequence and keeps the positions of the tasks which are assigned after this cut point while reconstructing a new sub-sequence before the cut position according to the solution construction strategy.

4.4. VNS approach

A variable neighborhood (VNS) based approach [37] is employed in order to further exploit the solutions that HDABC cannot improve. VNS uses the idea of systematically changing the neighborhoods in order to improve the current solution and aims to further explore the solution space, which may not be explored by a simple local search technique [38]. Shaking and local search operators are used in the implementation of the VNS. The shaking operator decides the search direction of the VNS from a set of neighborhoods. Therefore, each solution obtained by the shaking operator is further evaluated with the local search operator in order to explore new promising neighborhoods of the current solution. If there is an improvement, then the shaking operations returns to the first operator, otherwise, shaking continues with the next operation. After reaching the maximum number of shaking operations, the search procedure continues with the first operation in the new iteration until VNS iteration limit is reached. The VNS procedure is given in Table 2.

4.5. Evaluation mechanism

The evaluation mechanism is concerned with the computation of the objective function for each solution. In this study, the solutions are evaluated with a lexicographic method in which objectives are evaluated under the control of a decision maker in a hierarchical manner according to predefined priorities. Objectives are considered with a higher priority in the following order: first, second, third and fourth objectives. Therefore, with this lexicographic evaluation method, the algorithm tries to optimize the first objective by ignoring the others. Then, the algorithm tries to optimize second, third and fourth objectives, respectively. It is not possible to construct an efficient Pareto set since lexicographic evaluation method makes the algorithm ignore some better solutions according to other objectives. However, the algorithm will probably catch the best solution for the first objective since the lexicographic evaluation method focuses on the first objective with the highest priority. In fact, the second objective should also reach best values among others since the second objective behaves parallel to the first objective. However, third and fourth objectives are not likely to reach their best values since they conflict with the first and second objectives.

Another approach is to assign weights to each objective in order to obtain a single objective optimization problem. Thus, the algorithm can focus on a single objective and an efficient frontier may be found. In general, there are three different methods to assign these weights to each objective: fixed, random and adaptive weighted methods. In our preliminary experiments, fixed weighted method outperformed others. Therefore, second evaluation method that was investigated is fixed weighted method. In this method, multi objective optimization problem is solved with equal weights associated with each objective. Thus, an additional objective function is obtained using the following equation.

$$\min f_5 = \sum_{i=1}^{4} w_i \times f_i \quad \text{where} \quad \sum_{i=1}^{4} w_i = 1 \tag{18}$$

4.6. Pareto filter mechanism

Real world problems generally require simultaneous optimization of multiple (often conflicting) objectives. It is not possible to find one solution that perfectly satisfies all of our needs. In this case, a decision maker has to make a choice regarding which objective has the highest priority. As mentioned in the introduction section, the FDLBP is a multi-objective optimization problem where a number of objectives need to be minimized concurrently.

On the contrary to the optimization problem with a single objective, in a multi-objective optimization problem, there is a trade-off among objectives of a solution which is known as Pareto optimal set which contains all non-dominated solutions [39]. In this Pareto optimal set, a solution is improved in terms of a prioritized objective by lowering the importance of at least one of the other conflicting objectives. Multi-objective optimization enables decision maker to select an optimal solution in the form of Pareto optimal set based on his priority among objectives of the solution for FDLBP. To guarantee that the solutions obtained are non-dominated, a simple Pareto filtering approach is embedded into HDABC algorithm. The Pareto set is iteratively updated to get closer to an efficient Pareto-optimal front. Once the algorithm generates a solution that dominates any

Table 2
The VNS procedure.

Step	Description
1	Take initial solution π^0 from HDABC algorithm as an input
2	Accept initial solution π^0as the best solution π^1 to start shaking procedure
3	If VNS iteration limit is reached, go to Step 11
4	Apply a shaking operator randomly to the current best solution π^1 and obtain a π^2 solution in the neighborhood of the current best solution π^1
5	If solution π^2 is not better than current best solution π^1, go to Step 7
6	Save solution π^2 as current best solution π^1 and go back to Step 3
7	Accept solution π^2 as solution π^3 for starting local search procedure
8	Apply a local search operator to the solution π^3 and obtain a solution π^4
9	If solution π^4 is better than current best solution π^1, save π^4 as current best solution π^1 and go to Step 3
10	If VNS iteration limit is not reached, go to Step 8
11	Stop VNS procedure and send best found solution π^1 to the HDABC algorithm

Table 3
The procedure of HDABC.

Step	Description
1	Start
2	Initialize input data, precedence relationships and the predefined parameters (*ps, eb, ob, timelimit, ilimit*)
3	Generate initial population according to station based task assignment procedure
4	Evaluate initial population and construct *POP* matrix that keeps each solution and associated fitness values (f_1, f_2, f_3, f_4, f_5).
5	Set *NI* vector which holds the non-improved iteration number for each member of the population to a zero vector
6	Apply Pareto filter mechanism and save *POS* matrix that keeps Pareto-optimal solutions from initial population
8	Save best so far solution, apply Pareto filter mechanism and update *POS* matrix that keeps Pareto-optimal solutions according to current population
9	Release all employed bees to food sources as follows:
9.1	Apply randomly selected neighborhood structure to each solution in the current population and obtain new solutions
9.1.1	If an improvement is not achieved as a result of the neighborhood operator, go to Step 9.1.3
9.1.2	Save new solution and replace old solution in the current population, go to Step 9.1.4
9.1.3	Increase related position of *NI* vector to save non-improved iteration count
10	Release each onlooker bee as follows:
10.1	Apply roulette wheel selection mechanism and select a food source for next onlooker bee.
10.2	Apply randomly selected neighborhood structure to the selected solution and obtain a new solution
10.3	If an improvement is not achieved as a result of the neighborhood operator, go to Step 10.5
10.4	Save new solution and replace the selected old solution in the current population, go to 10.6
10.5	Increase related positioned member of *NI* vector to save non-improved iteration count
10.6	If there are any remaining onlookers at hand, go to Step 10.1
11	Start scout bees phase as follows:
11.1	A boolean statement is checked for each member in the population as follows:
11.1.1	If non-improved iteration count (*imp*) is less than predefined iteration limit (*ilimit*), go to Step 11.1.7
11.1.2	Generate a random number (r) in [0,1]
11.1.3	If $r \leq 0.5$, go to Step 12.1.5
11.1.4	Apply VNS algorithm to the solution
11.1.5	Generate a random solution
11.1.6	An improvement is achieved, set the related positioned member of *NI* vector to zero value
11.1.7	If any unchecked member is present, go to Step 11.1.1
12	Check time limit, if not exceeded go to Step 8
13	Save global statistics and print final results
14	Stop

solution in the Pareto set, it replaces the solution(s) which were dominated in the set.

4.7. HDABC procedure

The procedure of HDABC is given in Table 3.

5. Numerical results

The proposed algorithm was coded in C++ and tested on an Intel Xeon 2.5 GHz processor with 8 GB RAM. The performance of the HDABC algorithm was evaluated over two cases; a well-known data set (25-part mobile phone example) [40] and a new data set (47-part laptop example) introduced in this study. Considering fuzziness for the processing times, fuzzy data are represented by triangular fuzzy membership functions. The data for the two scenarios are given in Tables 4 and 5, respectively.

In order to show the effectiveness of the algorithm, fixed weighted and lexicographic evaluation mechanism is tested on DLBP before solving FDLBP. The numerical results of proposed algorithm are compared with other studies from the literature in Table 6. As a result, HDABC algorithm performed better within a limited time frame of 10 s. After extended experimentation, the following settings for HDABC algorithm's control parameters were found to be the best: number of employed bees, number of onlooker bees and number of tasks are equal to each other, iteration limit to release scout bees is equal to 100 while variable neighborhood search limit is equal to 10.

In order to understand how fuzziness influences the computational complexity, we investigated the behavior of the algorithm in a single run for different cases: 25-part and 47-part DLBP instances against 25-part and 47-part FDLBP instances. The algorithm was run for 100 s and the cumulative number of solutions obtained by the algorithm was observed. As depicted in Fig. 1, the cumulative number of feasible solutions found by the algorithm decreased when the problem size increased and/or fuzziness was considered. It can be drawn from Fig. 1 that fuzziness increases the complexity of the algorithm because it reduces the feasible number of solutions obtained in a given time since when the number of feasible solutions increase, the time to reach (near) optimum solutions is more likely to decrease. A further observation can be made from Fig. 1 that when the number of parts increases the influence of fuzziness

Table 4
Data of mobile phone instance.

Task no.	Description	Time	Demand	Hazardous	Predecessor
1	Antenna	(1, 3, 8)	4	Yes	0
2	Battery	(1, 2, 6)	7	Yes	0
3	Antenna guide	(2, 3, 4)	1	No	1, 2
4	Bolt (Type 1) A	(8, 10, 12)	1	No	0
5	Bolt (Type 1) B	(8, 10, 12)	1	No	0
6	Bolt (Type 2) 1	(12, 15, 18)	1	No	2
7	Bolt (Type 2) 2	(12, 15, 18)	1	No	2
8	Bolt (Type 2) 3	(12, 15, 18)	1	No	2
9	Bolt (Type 2) 4	(12, 15, 18)	1	No	3
10	Clip	(1, 2, 3)	2	No	4,5
11	Rubber seal	(1, 2, 3)	1	No	10
12	Speaker	(1, 2, 6)	4	Yes	11
13	White cable	(1, 2, 3)	1	No	6, 7, 8, 9
14	Red/blue cable	(1, 2, 3)	1	No	6, 7, 8, 9
15	Orange cable	(1, 2, 3)	1	No	6, 7, 8, 9
16	Metal top	(1, 2, 3)	1	No	6, 7, 8, 9
17	Front cover	(1, 2, 3)	2	No	13, 14
18	Back cover	(2, 3, 4)	2	No	15
19	Circuit board	(10, 12, 18)	8	Yes	13, 14, 16, 18
20	Plastic screen	(4, 5, 6)	1	No	17
21	Keyboard	(1, 2, 3)	4	No	17
22	LCD	(4, 5, 6)	6	No	21
23	Sub keyboard	(10, 12, 18)	7	Yes	16, 22
24	Internal IC	(1, 2, 3)	1	No	19, 23
25	Microphone	(1, 2, 6)	4	Yes	21

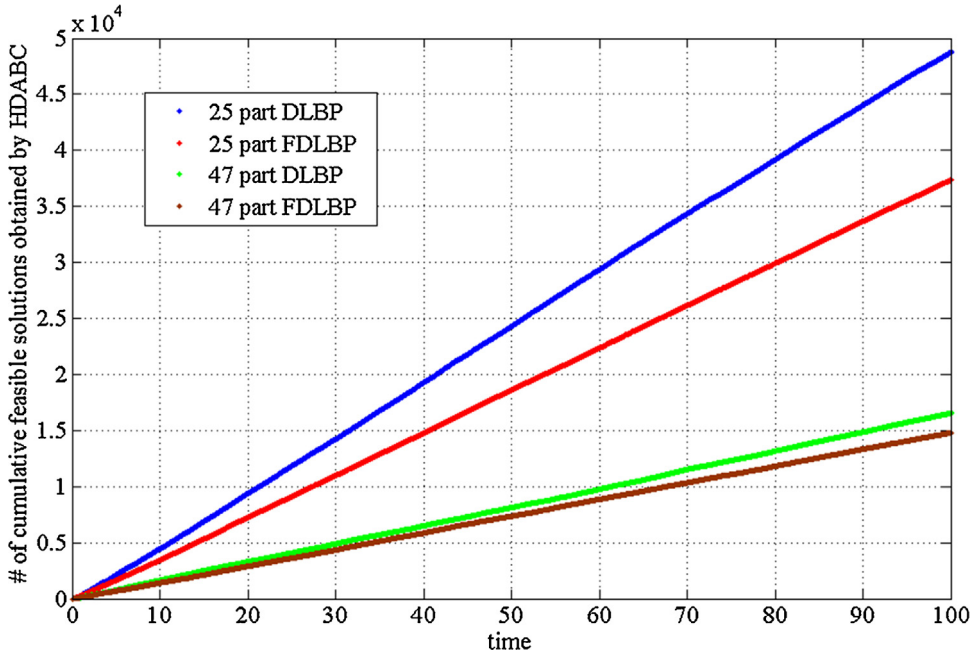

Fig. 1. Computational complexity comparison of DLBP and FDLBP for 25 part and 47 part instances.

on computational complexity is not as high as compared to smaller sized problems.

After the computational complexity analysis, 30 replications, starting each time from a different number seed, on the problem instance is performed within a time frame of 100 s to solve FDLBP instances. The algorithm is terminated on each replication after this time limit is reached. The best solutions obtained using lexicographic evaluation mechanism are presented in Table 7.

A typical Pareto optimal set obtained by HDABC algorithm with fixed weighted evaluation mechanism for mobile phone example and laptop example is given in Tables 8 and 9, respectively.

The results obtained showed that both methods have advantages against each other. Fixed weighted method was able to construct a more diverse Pareto frontier while lexicographic method was more successful to optimize the objective with the highest priority since lexicographic method may disregard some Pareto optimal solutions because of the predefined priorities with a biased focus on certain objectives. As discussed earlier, a decision maker is able to compare the solutions and select the most desirable one according to his/her needs. For example, for both product cases, the decision maker chooses solution 1, if smoothness index and number of workstation is more critical for the company. On the other hand, s/he may choose solution 2 if removing hazardous and high demanded components are more important.

In order to show the performance of HDABC, we compared HDABC with traditional artificial bee colony (ABC) algorithm and discrete version of artificial bee colony (DABC) algorithm. We tested each algorithm separately on both FDLBP instances.

Table 5
Data of laptop instance.

Task no.	Description	Time	Demand	Hazardous	Predecessor
1	2 bolt group	(14, 16, 18)	1	No	0
2	4 bolt group	(28, 32, 36)	1	No	1
3	Hard drive	(3, 4, 9)	3	Yes	2
4	Hard drive cover	(2, 4, 6)	1	No	2
5	Battery	(3, 4, 9)	5	Yes	0
6	Battery cover	(4, 6, 8)	1	No	5
7	Notebook feet	(8, 10, 12)	1	No	0
8	2 bolt group	(12, 16, 20)	1	No	0
9	ZIF connector	(4, 5, 6)	2	No	8
10	4 bolt group	(28, 32, 36)	1	No	8
11	CD driver	(3, 4, 9)	4	Yes	9, 10
12	CD driver cover	(4, 6, 8)	1	No	10
13	1 bolt group	(6, 8, 10)	1	No	0
14	RAM cover	(1, 2, 3)	2	No	13
15	3 bolt group	(20, 24, 28)	1	No	0
16	Strip cover	(5, 7, 9)	1	No	15
17	4 bolt group	(28, 32, 36)	1	No	16
18	Keyboard	(4, 6, 8)	2	No	17
19	RAM (bottom)	(3, 4, 9)	7	Yes	14
20	2 bolt group	(12, 16, 20)	1	No	18
21	Modem	(3, 4, 9)	6	Yes	20
22	PCI card	(3, 4, 9)	7	Yes	18
23	4 bolt group	(28, 32, 36)	1	No	22
24	4 bolt group	(28, 32, 36)	1	No	22
25	Monitor	(12, 16, 20)	6	Yes	23, 24
26	11 bolt group	(76, 88, 100)	1	No	19
27	Heat sink cover	(6, 8, 10)	2	No	26
28	4 bolt group	(28, 32, 36)	1	No	27
29	Heat sink	(3, 6, 11)	4	Yes	28
30	1 bolt group	(6, 8, 10)	1	No	29
31	Processor	(3, 4, 9)	7	Yes	30
32	13 bolt group	(98, 104, 110)	1	No	1, 5, 9, 18
33	Top cover	(14, 18, 22)	2	No	32
34	Cable	(2, 4, 6)	2	No	33
35	1 bolt group	(6, 8, 10)	1	No	33
36	Speaker	(7, 8, 12)	4	Yes	34, 35
37	9 bolt group	(60, 72, 84)	1	No	36
38	System board	(6, 8, 10)	1	No	37
39	9 bolt group	(60, 72, 84)	1	No	38
40	Fan	(8, 10,12)	3	No	39
41	2 bolt group	(12, 16, 20)	1	No	40
42	Audio board	(3, 4, 9)	3	Yes	41
43	2 bolt group	(12, 16, 20)	1	No	36
44	LED board	(3, 4, 9)	4	Yes	43
45	4 bolt group	(28, 32, 36)	1	No	44
46	Touchpad	(2, 4, 6)	3	No	45
47	RAM (top)	(3, 4, 9)	7	Yes	18

Table 6
Best results for DLBP.

Research work	f_1	f_2	f_3	f_4
MOACO [14]	9	9	–	883
RL [41]	9	9	97	862
HDABC with fixed weighted evaluation mechanism	9	9	80	824
HDABC with lexicographic evaluation mechanism	9	9	76	825

Table 7
Best results of HDABC with lexicographic evaluation mechanism for FDLBP.

Example	f_1	f_2	f_3	f_4	$\widetilde{BD}$	$\widetilde{SI}$	$\widetilde{BE}$
Mobile phone	12	14,048	85	919	(0, 33, 107)	(0, 12.042, 32.109)	(0.505, 0.810, 1)
Laptop	10	62,376	336	2840	(0, 184, 388)	(0, 61.595, 126.317)	(0.647, 0.823, 1)

Table 8
Pareto optimal solutions found by HDABC with fixed weighted evaluation mechanism for FDLBP with 25-part mobile phone instance.

Solution	$\widetilde{BD}$	$\widetilde{SI}$	$\widetilde{BE}$	f_1	f_2	f_3	f_4
1	(0, 33, 107)	(0, 12.12, 32.17)	(0.50, 0.81, 1)	**12**	**14,10**	86	935
2	(11, 48, 125)	(8.54, 19.84, 37.32)	(0.46, 0.75, 1)	13	21,39	**74**	**814**

Bold entries shows better solutions in terms of objective function values.

Table 9
Pareto optimal solutions found by HDABC with fixed weighted evaluation mechanism for FDLBP with 47-part laptop instance.

Solution	$\widetilde{BD}$	$\widetilde{SI}$	$\widetilde{BE}$	f_1	f_2	f_3	f_4
1	(0, 184, 388)	(0, 62.21, 126.55)	(0.65, 0.82, 1)	**10**	**62.74**	322	2678
2	(83, 288, 498)	(44.76, 103.37, 160.85)	(0.59, 0.75, 0.97)	11	103.09	**288**	**2471**

Bold entries shows better solutions in terms of objective function values.

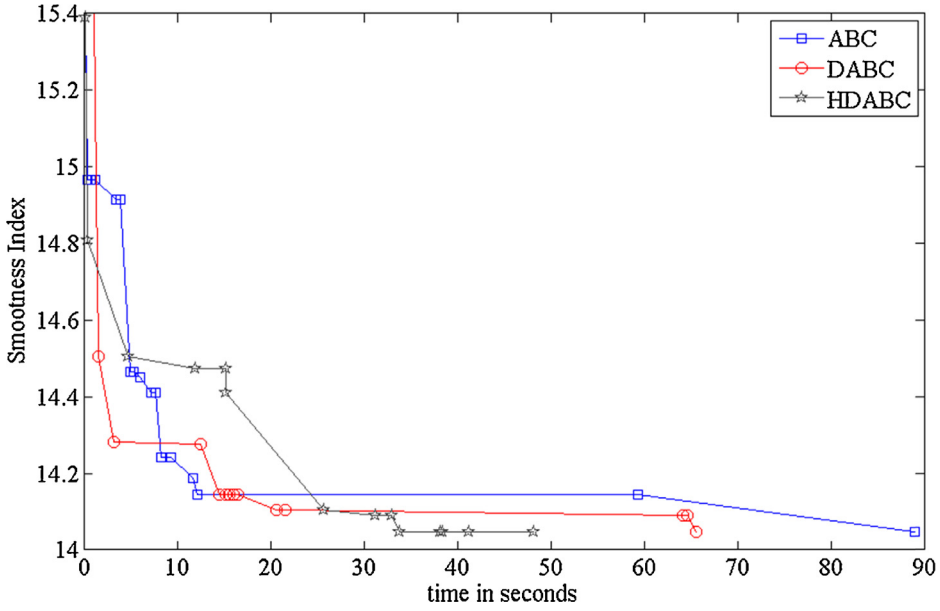

Fig. 2. Improvement of the solutions over iterations with each algorithm's best performance for 25 part FDLBP problem instance.

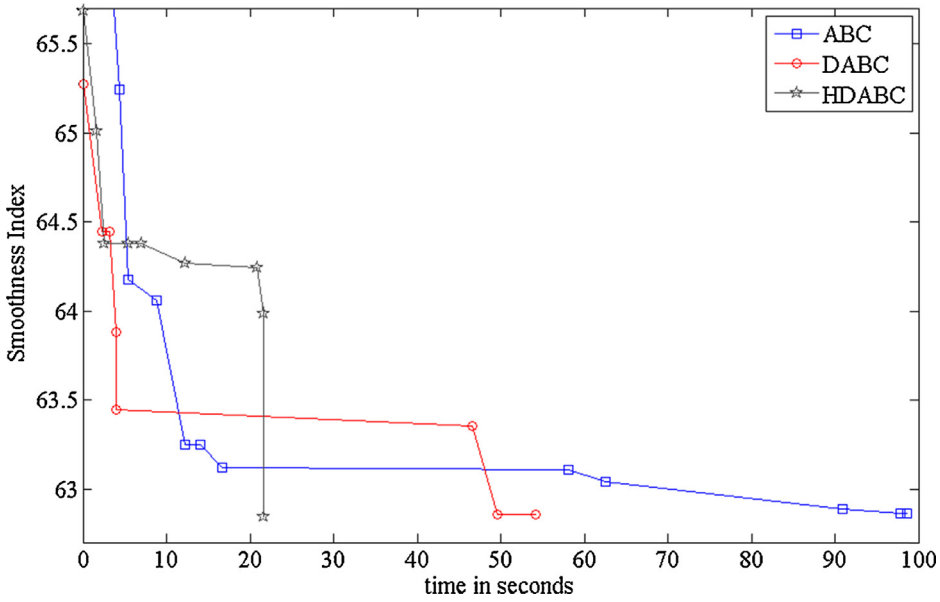

Fig. 3. Improvement of the solutions over iterations with each algorithm's best performance for 47 part FDLBP problem instance.

Figs. 2 and 3 demonstrate the improvement of the solutions over iterations within a time limit of 100 s for 25-part and 47-part FDLBP instances, respectively. All algorithms compared reached the same solutions within the time frame, however, HDABC algorithm achieves the solution faster than the others. Figs. 4 and 5 depict the best solutions reached by all algorithms on 30 sample runs of 25-part and 47-part FDLBP instances, respectively. In Figs. 4 and 5, it can be seen that HDABC was more robust since it was able to reach better solutions in many of sample runs. For 47-part instance, the performance difference is more obvious. Fig. 6 presents a statistical analysis for 30 sample run results indicating that within 95% of confidence interval HDABC is more robust and efficient.

In order to understand the influence of integrated VNS approach on HDABC algorithm, a comparison between DABC and HDABC was done in terms of number of improvements achieved within 100 s of time frame. As can be seen in Tables 10 and 11, HDABC algorithm was able to achieve 286 (and 287 for 47-part FDLBP instance) improvements with integrated VNS mechanism while DABC was able to achieve 197 improvements.

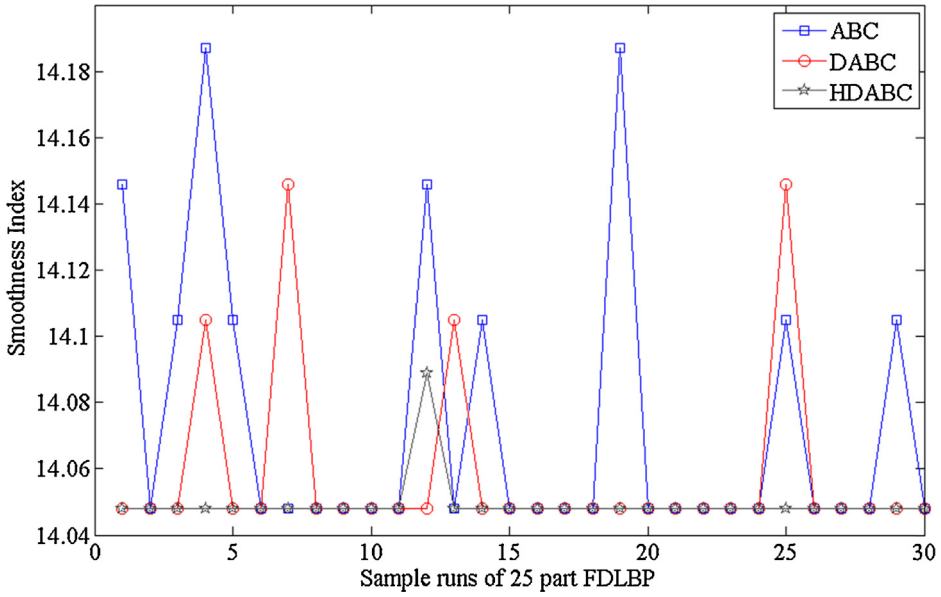

Fig. 4. 30 sample runs of 25 part FLDBP instance with each algorithm.

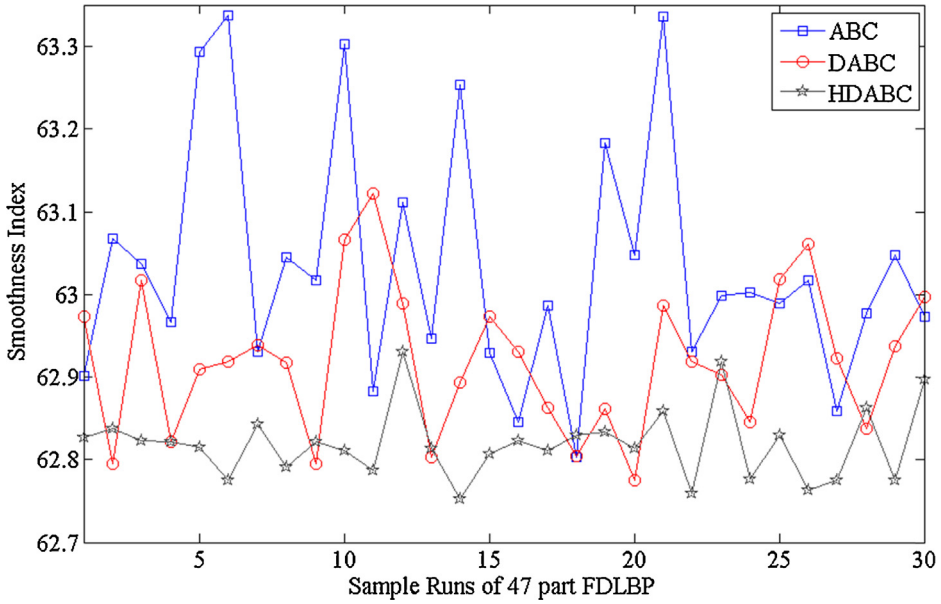

Fig. 5. 30 sample runs of 47 part FLDBP instance with each algorithm.

Table 10
Number of improvements achieved by neighborhood structures in 100 s (25 part FDLBP problem instance): a sample run of HDABC algorithm.

	VNS local search structures						
	Pure (DABC alone)	N1	N2	N3	N4	N5	Sum
		With local search					
VNS shaking structures							
N1	51	18	6	1	1	0	77
N2	61	24	5	3	2	0	95
N3	14	3	1	3	1	0	22
N4	51	8	4	2	1	0	66
N5	20	4	1	1	0	0	26
Sum	197	57	17	10	5	0	286

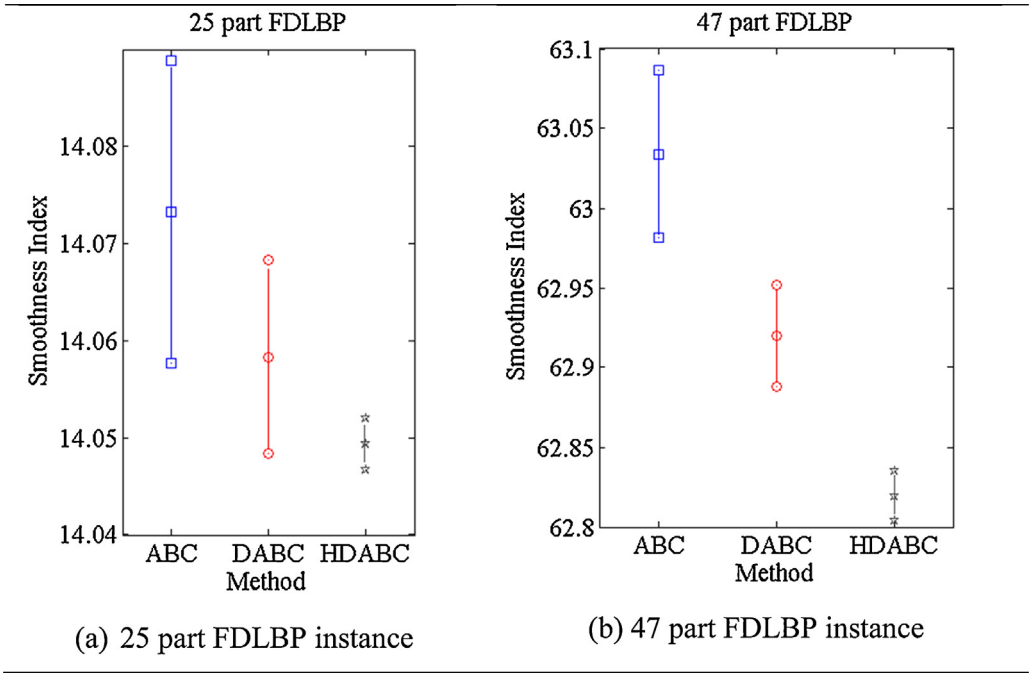

Fig. 6. Performance comparison of methods within 95% confidence interval for FDLBP problem instances.

Table 11
Number of improvements achieved by neighborhood structures in 100 s (47 part FDLBP problem instance): a sample run of HDABC algorithm.

	VNS local search structures						
	Pure (DABC alone)	N1	N2	N3	N4	N5	Sum
		With local search					
VNS shaking structures							
N1	47	27	5	3	2	0	84
N2	92	17	4	1	0	0	114
N3	16	7	4	0	1	0	28
N4	20	9	0	1	0	0	30
N5	22	6	1	1	1	0	31
Sum	197	66	14	6	4	0	287

6. Conclusions

This study contributes to the literature by presenting a fuzzy extension of the disassembly line balancing problem (DLBP) with fuzzy task processing times using a hybrid discrete artificial bee colony (HDABC) algorithm as a solution approach for the first time. The algorithm aims to solve the fuzzy disassembly line balancing problem with objectives of minimizing the number of workstations, optimizing fuzzy smoothness index of the line, maximizing the earliest removal of hazardous components and maximizing the earliest removal of high demanded components. Balance delay and balance efficiency of the disassembly line was also measured. The effectiveness of proposed algorithm is tested using a mobile phone instance with 25 parts from the literature and a laptop instance with 47 parts introduced in this study. The total cost function is formulated according to lexicographic and fixed weighted mechanism. In the fixed weighted method, the total cost function is calculated with equally weighted sum of multiple objectives. As for the lexicographic method, it was assumed that a manager already decided the priorities of the objectives. The influences of both mechanisms on the performance of the proposed HDABC were examined. The examination to understand the influence of fuzziness on the complexity of the algorithm indicated that fuzziness increases the complexity because it reduces the feasible number of solutions obtained in a given time since when the number of feasible solutions increase, the time to reach (near) optimum solutions is more likely to decrease. Further analysis suggested that the performance of HDABC compared to traditional artificial bee colony (ABC) algorithm and discrete version of artificial bee colony (DABC) algorithm was the best in terms solution quality and robustness. This work is limited to single model FDLBP and considers a complete disassembly of certain products. As for further studies, more complex DLBPs should be considered. An idea is to address mixed model or partial DLBP with fuzzy job processing times.

Acknowledgment

This work is partially supported by the Scientific Research Projects Fund of Pamukkale University under the Grant No. 2010FBE051.

References

[1] Gungor A, Gupta SM. Issues in environmentally conscious manufacturing and product recovery: a survey. Comput Ind Eng 1999;36:811–53.
[2] Ilgin MA, Gupta SM. Environmentally conscious manufacturing and product recovery (ECMPRO): a review of the state of the art. J Environ Manage 2010;91:563–91.
[3] Tang Y, Zhou M, Zussman E, Caudill R. Disassembly modeling, planning, and application. J Manuf Syst 2002;21:200–17.
[4] Rickli JL, Camelio JA. Multi-objective partial disassembly optimization based on sequence feasibility. J Manuf Syst 2013;32:281–93.

[5] Gungor A, Gupta SM. A solution approach to the disassembly line balancing problem in the presence of task failures. Int J Prod Res 2001;39:1427–67.
[6] Gungor A, Gupta SM. Disassembly line in product recovery. Int J Prod Res 2002;40:2569–89.
[7] McGovern SM, Gupta SM. A balancing method and genetic algorithm for disassembly line balancing. Eur J Oper Res 2007;179:692–708.
[8] Tovey CA. Tutorial on computational complexity. Interfaces 2002;32:30–61.
[9] Altekin FT, Akkan C. Task-failure-driven rebalancing of disassembly lines. Int J Prod Res 2012;50:4955–76.
[10] Altekin FT, Kandiller L, Ozdemirel NE. Profit-oriented disassembly-line balancing. Int J Prod Res 2008;46:2675–93.
[11] Bentaha ML, Battaïa O, Dolgui A. A sample average approximation method for disassembly line balancing problem under uncertainty. Comput Oper Res 2014;51:111–22.
[12] Koc A, Sabuncuoglu I, Erel E. Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. IIE Trans 2009;41:866–81.
[13] Agrawal S, Tiwari MK. A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. Int J Prod Res 2008;46:1405–29.
[14] Ding L-P, Feng Y-X, Tan J-R, Gao Y-C. A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. Int J Adv Manuf Technol 2010;48:761–71.
[15] Kalayci CB, Gupta SM. Ant colony optimization for sequence-dependent disassembly line balancing problem. J Manuf Technol Manage 2013;24:413–27.
[16] Kalayci CB, Gupta SM. River formation dynamics approach for sequence-dependent disassembly line balancing problem. In: Reverse supply chains. Boca Raton, FL: CRC Press; 2013. p. 289–312.
[17] Kalayci CB, Gupta SM. Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. Expert Syst Appl 2013;40:7231–41.
[18] Kalayci CB, Gupta SM. A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. Int J Adv Manuf Technol 2013;69:197–209.
[19] Kalayci CB, Gupta SM. Simulated annealing algorithm for solving sequence-dependent disassembly line balancing problem. In: IFAC conference on manufacturing modelling, management, and control. 2013. p. 93–8.
[20] Kalayci CB, Gupta SM. A tabu search algorithm for balancing a sequence-dependent disassembly line. Prod Plan Control 2013;25:149–60.
[21] McGovern SM, Gupta SM. Ant colony optimization for disassembly sequencing with multiple objectives. Int J Adv Manuf Technol 2005;30:481–96.
[22] McGovern SM, Gupta SM. Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. Int J Prod Res 2007;45:4485–511.
[23] Zha J, Yu J-j. A hybrid ant colony algorithm for U-line balancing and rebalancing in just-in-time production environment. J Manuf Syst 2014;33:93–102.
[24] Kalayci CB, Polat O, Gupta SM. A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. Ann Oper Res 2014:1–34 (in press).
[25] McGovern SM, Gupta SM. The disassembly line: balancing and modeling. New York: McGraw Hill; 2011.
[26] Tsujimura Y, Gen M, Kubota E. Solving fuzzy assembly-line balancing problem with genetic algorithms. Comput Ind Eng 1995;29:543–7.
[27] Gen M, Tsujimura Y, Li Y. Fuzzy assembly line balancing using genetic algorithms. Comput Ind Eng 1996;31:631–4.
[28] Hop NV. A heuristic solution for fuzzy mixed-model line balancing problem. Eur J Oper Res 2006;168:798–810.
[29] Tapkan P, Özbakır L, Baykasoğlu A. Bees Algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem. Optim Lett 2012;6:1039–49.
[30] Zacharia PT, Nearchou A. Multi-objective fuzzy assembly line balancing using genetic algorithms. J Intell Manuf 2012;23:615–27.
[31] Zacharia PT, Nearchou AC. A meta-heuristic algorithm for the fuzzy assembly line balancing type-E problem. Comput Oper Res 2013;40:3033–44.
[32] Paksoy T, Güngör A, Özceylan E, Hancilar A. Mixed model disassembly line balancing problem with fuzzy goals. Int J Prod Res 2013;51:6082–96.
[33] Anghinolfi D, Paolucci M. A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. Eur J Oper Res 2009;193:73–85.
[34] Karaboga D. An idea based on honey bee swarm for numerical optimization. Kayseri: Erciyes University, Engineering Faculty, Computer Engineering Department; 2005.
[35] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Appl Math Comput 2009;214:108–32.
[36] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 2008;8:687–97.
[37] Mladenović N, Hansen P. Variable neighborhood search. Comput Oper Res 1997;24:1097–100.
[38] Hansen P, Mladenović N, Moreno Pérez J. Variable neighbourhood search: methods and applications. Ann Oper Res 2010;175:367–407.
[39] Delorme X, Battaïa O, Dolgui A. Multi-objective approaches for design of assembly lines. In: Benyoucef L, Hennet J-C, Tiwari MK, editors. Applications of multi-criteria and game theory approaches. London: Springer; 2014. p. 31–56.
[40] Gupta SM, Erbis E, McGovern SM. Disassembly sequencing problem: a case study of a cell phone. In: Gupta SM, editor. Environmentally conscious manufacturing IV. Bellingham: SPIE-International Society for Optical Engineering; 2004. p. 43–52.
[41] Tuncel E, Zeid A, Kamarthi S. Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. J Intell Manuf 2014;25:647–59.