



A comparative analysis of meta-heuristic methods on disassembly line balancing problem with stochastic time

Süleyman Mete¹ · Faruk Serin² · Zeynel Abidin Çil³ · Erkan Çelik⁴ ·
Eren Özceylan¹

Accepted: 2 August 2022 / Published online: 4 October 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

The balancing of the disassembly line directly affects the productivity of the disassembly process. The disassembly line balancing (DLB) problem can be determined as assigning the tasks to serial workstations to optimize some performance measures like number of workstations, cycle time, removing hazardous parts earlier, etc. The aim of the paper is to develop an efficient heuristic algorithm to minimize the number of workstations under a pre-known cycle time. In this paper, a genetic algorithm (GA) and a constructive heuristic based on the Dijkstra algorithm is proposed to solve the DLB problem with stochastic task times that is caused by the nature of disassembly operation. The proposed algorithms are tested on benchmark problems and compared with the results of the piecewise-linear model (PLM) and simulated annealing (SA). The average relative percentage deviation is applied to transfer the obtained number of workstations. The results obtained by GA are clearly superior in all tests problem according to average relative percentage deviation. Moreover, the proposed constructive heuristic based on the Dijkstra algorithm is also superior to PLM and SA algorithm with respect to number of workstations and the computational times. The proposed approaches can be a very competitive and promising tool for further research in DLB literature and real cases in industries according to test results. Disassembly lines which need less time or number of workstations for balancing may be simply designed by the proposed techniques.

Keywords Genetic algorithm, Dijkstra algorithm · Disassembly line balancing · Piecewise linear · Simulated annealing · Stochastic task times

✉ Eren Özceylan
ereozceylan@gmail.com

¹ Department of Industrial Engineering, Gaziantep University, Gaziantep, Turkey

² Department of Computer Engineering, Mersin University, Mersin, Turkey

³ Department of Industrial Engineering, Izmir Democracy University, Izmir, Turkey

⁴ Department of Transportation and Logistics, Istanbul University, Istanbul, Turkey

1 Introduction

The product life cycle has significantly decreased in recent years despite developing life quality and innovations. Therefore, the manufacturers adapt the production system to mass customization and rapid product realization. These factors bring the new challenges for waste management. Product recovery operations are considered as a difficulty for improving value added activities by many manufacturing and remanufacturing firms. However, recycling and remanufacturing processes have become popular in the manufacturing environment due to a sequence of the legislative action, environmental issue and public pressure (Ketzenberg et al., 2003). Remanufacturing needs material and components acquisition of unused goods for minimizing the amount of waste sent to landfills. Therefore, certain steps are available to improve product recovery. Ilgin and Gupta (2010) provide much more information about product recovery, remanufacturing, and recycling.

One of the important steps of the product recovery/remanufacturing is disassembly, which is a systematic separation of appreciated parts from the used products. Therefore, disassembly line has attracted much more attention to increase productivity and flexibility of disassembly automation (Gungor and Gupta 1999). The topic of balancing of disassembly line has become important to obtain valued component and increase process capability. DLB problem can be defined that each task is assigned to a workstation with considering precedence relations and workstation capacity (cycle time) to minimize/maximize some performance measures. Unlike assembly line, disassembly environment and operations cannot be standardized easily because there is an uncertainty of product conditions. Factors such as the conditions of use of the product, the amount of abrasion of the parts, workers' performance, and environmental effects can be considered for this. Hence, uncertainty in the quality of the used products can cause large time variations for the same task on the disassembly line (Altekin et al., 2016). In the literature, certain research has been interested in DLB problem under the stochastic task times variation (Agrawal & Tiwari, 2008; Aydemir et al. 2013; Bentaha et al., 2014; Altekin et al., 2016). In these studies, task times are generally assumed that disassembly task times are normally distributed with known mean and variance. Recently, the mean and covariance matrix of task processing times with partial uncertain knowledge for the DLB problem is considered by Liu et al (2020a). He et al. (2021) also studied a green-oriented bi-objective DLB problem with stochastic task time.

Due to the NP-hard nature of DLBP proved by McGovern and Gupta (2007a, 2007b) heuristic methods become popular to obtain high-quality solutions in acceptable computational time. In our study, we mainly focus to improve an efficient genetic algorithm. Also, three techniques with different perspectives, which are simulated annealing, piecewise linear model, and Dijkstra's shortest path, were compared with the proposed method to show superior performance of the genetic algorithm. Genetic algorithm, simulated annealing, piecewise linear model, and Dijkstra's shortest path are evolutionary, probabilistic, linear, and graph-based techniques, respectively. In this study, these four techniques with different perspectives were selected to benchmark the effectiveness and efficiency of different solution approaches. The main contributions of the paper can be given as follows: (i) a novel GA algorithm is presented, (ii) a new recombination and mutation operators are developed based on precedence relations because the length of each chromosome can be different from each other, (iii) a constructive heuristic algorithm is developed for DLB problem in order to use graph structure, (iv) stochastic test problems for DLB problem are generated for comparative analysis, and (v) the results are compared with SA algorithm and PLM.

The rest of the paper is organized as follows. The previous research of the DLB problem is given in Sect. 2. In Sect. 3, problem definition and mathematical model for DLB is explained. In Sect. 4, the proposed algorithms are given in detail. Computational experiments are shown and discussed for stochastic DLB problem in Sect. 5. Conclusions and directions for future research are argued in the last section. Abbreviations and acronyms repeatedly used in this paper are listed in Table 1.

2 Literature review

The first study about the problem of DLB is defined by Gungor and Gupta (1999) since that time research related DLB problem literature is increased day by day. There are exact solution methods and heuristic approaches to solve the problem in the literature. Hence, this section is divided as two sub-sections. Firstly, mathematical modelling solution approaches are analysed, then heuristic-based solution approach studies are examined.

First mathematical formulation is developed by Altekin et al. (2008) to solve DLB problem. Altekin and Akkan (2012) formulated the problem of rebalancing disassembly lines with task failure by allowing partial disassembly. Koc et al. (2009) developed two mathematical formulations (dynamic and integer programming) of DLB problem with deterministic task times under complete disassembly. The sample average approximation method with stochastic linear mixed integer programming is proposed by Bentaha et al. (2014) for solving the problem as efficiently way. Mixed integer linear programming for DLB problem with joint precedence graph generation are developed by Riggs et al. (2015). Ilgin et al. (2017) proposed linear physical programming to solve the problem. Kalaycilar et al. (2016) presented integer programming approach for solving partial DLB problem under fixed number of workstations. A novel mathematical model for joint design assembly and disassembly line balancing problem is introduced by Mete et al. (2018). Recently, Mete et al. (2019) proposed a model for supply-driven rebalancing of DLB problem. Li et al. (2020) proposed an exact solution approach based on branch, bound, and remember algorithm to solve the problem. Liu et al. (2020b) introduced cutting plane algorithm to solve stochastic DLB problem. Recently, Çil et al. (2022) proposed constraint programming model for two-sided DLB problem with sequence dependent setup time. Yin et al. (2022) developed exact mixed-integer programming model multi-product partial DLB problem with multi-robot workstations.

On the other hand, different heuristic and metaheuristic approaches are proposed in the literature for the DLB problem, due to the nature of the problem is NP-hard (Mcgovern & Gupta, 2007a). These solution methods can be exemplified as follows: Genetic algorithm (Aydemir-Karadag & Turkbey, 2013; McGovern & Gupta, 2007b; Kalaycilar et al., 2016), particle swarm optimisation (Kalayci & Gupta, 2013a; Xiao et al., 2017), ant colony algorithm (Agrawal & Tiwari, 2008; Ding et al., 2009; Kalayci & Gupta, 2013b; Zhu et al., 2014), greedy algorithm (McGovern & Gupta, 2007a), artificial fish swarm algorithm (Zhang et al., 2017), reinforcement learning technique (Tuncel et al., 2014), beam search algorithm (Mete et al., 2016), network-based shortest route model (Hezer and Kara, 2015), gravitational search algorithm (Ren et al., 2017), 2-opt Algorithm (Ren et al., 2018) genetic simulated annealing (Wang et al., 2019) and iterated local search (ILS) algorithm (Li et al., 2019). Kazancoglu and Ozturkoglu (2018) addressed to business and green issues in DLB problem, and a hybrid approach was developed to solve the problem. In recent years, Çil et al. (2020) proposed a mathematical model and ant colony algorithm for robotic DLB problem. A Pareto-discrete hummingbird algorithm is developed by Yin et al. (2021) for partial sequence-dependent

Table 1 Abbreviations and acronyms

	DLB	Disassembly line balancing
GA		Genetic algorithm
SA		Simulated annealing
PLM		Piecewise-linear model
TAOG		Transformed AND/OR graph
TPA		Dijkstra-based shortest path approach
ARPD		Average relative percentage deviation
NW		Number of workstations
μ_i		Means
σ^2		Variances
α		A confidence level
P		probability

DLB problems with tool requirements. Zhang et al. (2022) improved whale optimisation algorithm for two-sided DLB problem with part characteristic indexes.

The literature on the DLB problem and its variants is rich, and the reader is referred to the comprehensive survey by Özceylan et al. (2019), and Deniz and Ozcelik (2019) for a recent coverage of the state of the art on models and solution algorithms. On the other hand, Gungor and Gupta (2001) generated shortest directed paths between the source and the final nodes of weighted state network using the Dijkstra's shortest path algorithm for DLB problem with task failures. In addition to related literature, a constructive heuristic algorithm for DLB problem under stochastic task times with complete disassembly is developed in this study.

3 Problem formulation

DLB problem assigns the set of tasks to each workstation for each product to be disassembled by considering cycle time constraints and precedence relationships. The objective function of this study is to minimize the number of workstations. The task precedence diagram proposed by Koç et al. (2009), which is Transformed AND/OR graph (TAOG), is considered for formulated the problem. TAOG contains certain knowledge about precedence relationships among disassembly tasks on whole the disassembly trees. Each node in the TAOG corresponding to a subassembly is symbolized by an artificial node (A_k). A disassembly task is symbolized by normal node (B_i) in the graph. The example for TAOG graph of toy car can be examined in Fig. 1, which contains 43 artificial (grey) nodes and 97 normal (white) nodes.

The mathematical model for the deterministic DLB problem is proposed by Koc et al. (2009). The model is revised for stochastic DLB problem with change of cycle time constraint in this paper. The model assumptions and notations are given as follows:

- Complete disassembly is permitted, and one type of product is disassembled.
- TAOG is used for task precedence relations.
- Tasks splitting are not allowed.
- Single model and straight disassembly line are considered.

3.1 Notations

I	, is task index $i = 1, 2, \dots, I$;
j	, is workstation index, $j = 1, 2, \dots, J$;
d_{Bi}	, is the disassembly task time of (B_i);
C	, is the cycle time;
$P(A_k)$, $P(B_i)$, is the immediate predecessor set of artificial node A_k , B_i , respectively;
$S(A_k)$, $S(B_i)$, is the immediate successor set of artificial node A_k , B_i , respectively;

$Z_{1-\alpha}$, Probability of cumulative standard normal distribution for $1 - \alpha$.

μ_i , is the mean time of disassembly task i

σ_i , is the standard deviation of disassembly task i

3.2 Decision variables

If task B_i is assigned to workstation j , X_{ij} is 1; otherwise, it is 0;

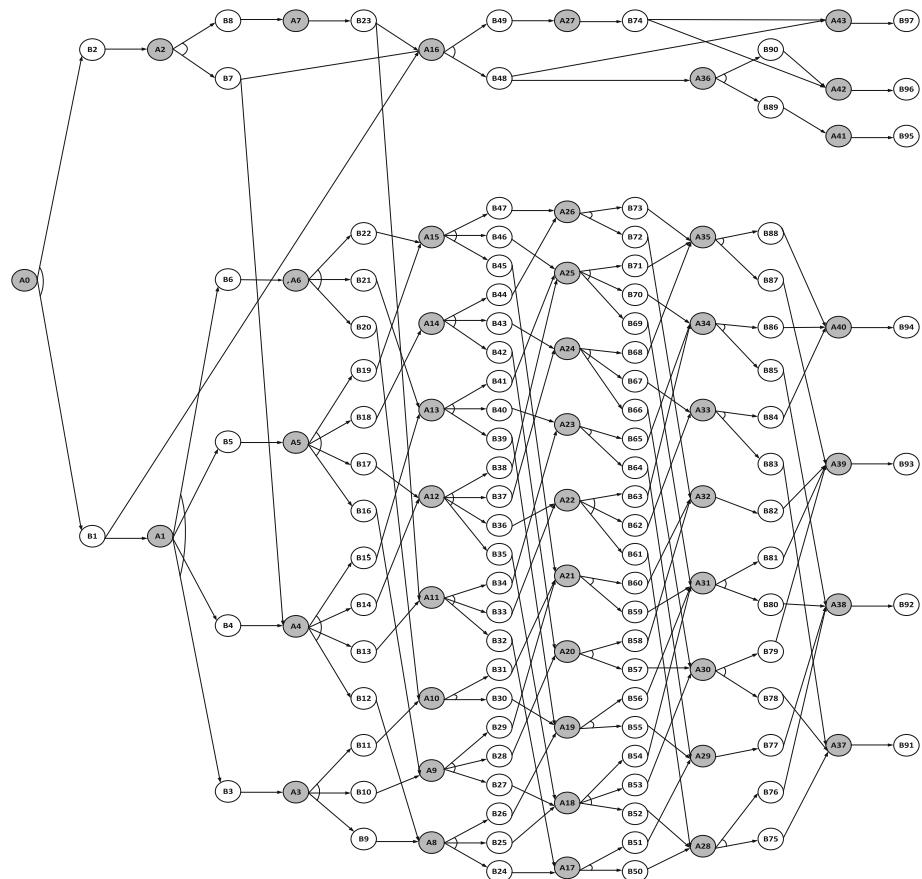


Fig. 1 An example of TAOG graph for toy car (Mete et al., 2018)

If workstation j is opened, F_j is 1; otherwise, it is 0;

If task B_i is performed, Z_i is 1; otherwise, it is 0;

3.3 Objective function

The objective function (1) is to minimize the number of disassembly workstations under a pre-known cycle time.

$$\min \sum_{j=1}^J j \cdot F_j \quad (1)$$

3.4 Subject To

Equations (2) and (3) ensure that accurately one of the OR-successors is chosen. By this way, these two equations push the solution to contain the feasible set of tasks.

$$\sum_{i:B_i \in S(A_k)} Z_i = 1 \quad \forall k=0 \quad (2)$$

$$\sum_{i:B_i \in S(A_k)} Z_i = \sum_{i:B_i \in P(A_k)} Z_i \quad \forall k=1,2,\dots,h \quad (3)$$

Equation (4) provides that a selected task is assigned to one of the workstations.

$$\sum_{j=1}^J X_{ij} = Z_i \quad \forall i=1,2,\dots,I \quad (4)$$

Inequality (5) deals with the precedence relationships among the disassembly tasks: Since exactly one of the OR predecessors and one of the OR successors of an artificial node will be selected.

$$\sum_{i:B_i \in P(A_k)} \sum_{j=1}^v X_{ij} \leq \sum_{i:B_i \in S(A_k)} X_{iv} \quad \forall k=1,2,\dots,K \quad (5)$$

Constraint (6) ensures that completion probability of tasks in workstation within the cycle time is greater or equal than $(1 - \alpha)$ probability level.

$$p \left\{ \sum_i .X_{ij} + Z_1 - \sqrt{\sum \sigma_i^2 .X_{ij}} \leq C \right\} \geq 1 - \alpha \quad \forall j=1,2,\dots,J \quad (6)$$

Constraint set (7) indicate binary restrictions

$$X_{ij}, F_j, Z_i \in \{0, 1\} \quad \forall i=1,2,\dots,I; j=1,2,\dots,J \quad (7)$$

4 Proposed approaches

The following sub-sections describe the applied approaches respectively.

4.1 The proposed constructive heuristic algorithm

The pseudo-codes of the Dijkstra-based shortest path approach are given in Algorithm 1 and Algorithm 2. TAOG is converted to weighted directed graph defined as $G(V, E)$ an example of which is shown in Fig. 2. V of G is vertex list of the modified graph G , and refers to tasks. E is edge connecting vertex v_i to v_j , and denoted as $E(v_i, v_j, w)$ where weight w is processing time of task v_j . There are two vertices start vertex s and finish (end) vertex e , which are not task. In graph construction, Edges generated from s to all vertices having no predecessor task, and from all vertices having no successor to e . Weight from any vertex s to v_a is equal to processing time of task v_a , and weight from e to any vertex is zero. After construction of the graph G , the main procedure in Algorithm 1 is run. As seen in Algorithm 1, firstly, the path of minimum total task processing time is calculated by using Dijkstra-based shortest

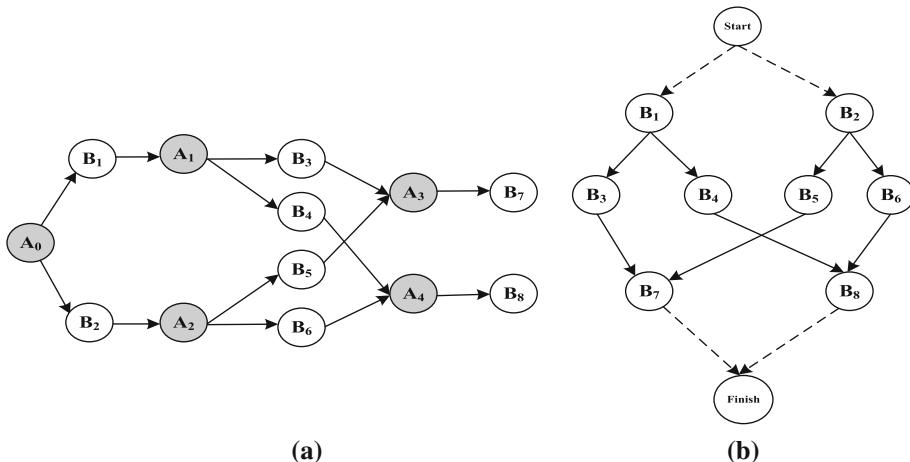


Fig. 2 **a** Example of TAOG precedence diagram, **b** modified graph for heuristic algorithm

path approach as seen in Algorithm 2. Then, the number of workstations is calculated as stochastically as in Algorithm 3. Task assignability value is calculated as in Algorithm 4.

Algorithm 1. The pseudo-code of the Dijkstra-based approach

Name:

main(G, T, V, D, c, s, e)

Input Variables:

G: Graph

T: Task processing time list

V: Task variance list

D: Cumulative normal distribution table

c: Cycle time

s: Start task (vertex)

e: End task (vertex)

Operation Variables:

P: Vertex list of tasks on path

Operations:

$P = \text{TPA_min_cost_path}(G, s)$

$ns = \text{calculate_stochastic_workstation}(V, D, P, T, c, s, e)$

Output:

$ns: \text{number of workstations}$

Algorithm 2. The procedure of the task selection

Name:
 $TPA_min_cost_path(G,s)$

Input Variables:

G : Graph

s : source vertex

Operation Variables:

P : Predecessors vertex list in path from source

D : Path distance from source to source

Q : The unsettled vertices of graph

W : Weight of edge between vertex u and vertex v

u : A vertex

t : Temporary distance

Operations:

for each vertex v in G

$D[v] = \infty$

$P[v] = \emptyset$

end for

$D[s] = 0$;

Q = the set of all vertex in G

while Q is not empty

u = vertex in Q with smallest distance D

 remove u from Q

for each successor v of u

$t = D[u] + W(u, v)$

if $t < D[v]$ **then**

$D[v] = t$

$P[v] = u$

end if

end for

end while

Output:

P

4.2 The proposed genetic algorithm

The GA is a search heuristic for solving constrained optimization problems. The components of a genetic algorithm are population, chromosomes (individuals) and genes (Goldberg, 1989). A population consists of individuals while an individual consists of genes. A chromosome gives a solution for the problems. The algorithm repeatedly modifies the population while searching better solution (Serin et al., 2019). Selection, crossover, and mutation processes are the searching operation of algorithm. The graph of the problem is constructed initially to apply the proposed algorithm. In the graph, the edge weight of two sequential nodes (tasks) is equal to the task processing time. Genetic algorithms are widely used in solving problems that can be represented as graph.

The number of workstations is calculated stochastically. The stochastic calculation procedure is shown in Algorithm 3. In stochastic calculation, task assignability value is calculated as in Algorithm 4, and this value must not exceed the confidence level. The stochastic calculation procedure and task assignability value are the same for Dijkstra-based shortest path approach and proposed genetic algorithm.

Algorithm 3. Procedure of the stochastic case**Name:***calculate_stochastic_workstation (V, D, P, T, c, s, e)***Input Variables:***T*: Task processing time list*V*: Task variance list*D*: Cumulative normal distribution table*c*: Cycle time*s*: Start task*e*: End task*a*: Confidence level**Operation Variables:***W*: vertices assigned to the workstation*n*: Number of workstations**Operations:***n* = 1*p*=0*v*=*P*[*e*]*W* = \emptyset **while** *v* is not *s* *W* = *W* \cup *v* *p*=task_assignability_value(*W*, *T*, *V*, *D*, *c*) **if** *p*>*a* **then** *W* = \emptyset *W* = *W* \cup *v* *n*=*n*+1 **end if** *v*=*P*[*v*]**end while***p*=task_assignability_value(*W*, *T*, *V*, *D*, *c*)**if** *p*>*a* **then** *n*=*n*+1**end if****Output:***n***Algorithm 4.** Procedure of task assignment**Name:***task_assignability_value(*W*, *T*, *V*, *D*, *c*)***Input Variables:***W*: vertices assigned to the workstation*T*: Task processing time list*V*: Task variance list*D*: Cumulative normal distribution table*c*: Cycle time**Operation Variables:***sm, sv, z***Operations:***sm*=0*sv*=0**for each** vertex *v* of *W* *sm* =*sm*+*T*[*v*] *sv*=*sv*+*V*[*v*]**end for***z*= $(c-sm)^2/\sqrt{sv}$ **Output:**1-*D*[*z*]

In the proposed genetic algorithm, a gene is a task while an individual is an assigned task list as shown in Fig. 3 where T_i is a gene (task). The first individuals of the population are

Fig. 3 An individual

T_1	T_2	T_3	...	T_{n-2}	T_{n-1}	T_n
-------	-------	-------	-----	-----------	-----------	-------

generated randomly. In this study, several improvement tools are applied to GA to increase the capability to find the best solution and reduce the problem's time complexity. The structure of the TAOG graph is different from the standard precedence relationship. Thus, different crossover and mutation techniques are applied in this paper.

There are different types of crossovers such as single point crossover, two-point crossover, uniform crossover. In single point crossover, a crossover point of the individual chromosomes is chosen. Then, the genes beyond the selected point are swapped between the two parent individuals. In two-point crossover, two points of the individual chromosomes are selected and then the genes between the chosen points are exchanged between the two parent individuals. In uniform crossover, each gene of child chromosomes is chosen randomly from the corresponding genes of either parent. These common crossovers frequently produce illegal offspring since the produced chromosome is not feasible task assignments in many times for this problem. This may result in too much iteration for a successful solution or an unsuccessful solution. To avoid this situation, instead of choosing a random or static point, if a common gene exists in both chromosomes, the approach of crossing over this common gene point is adopted. In the crossover tool, if two individuals have a common point (gene), they can be selected for crossover. Otherwise, the algorithm will continue looking for new matching. In other words, in crossover, a mutual task is searched in two individuals, and two individuals are swapped in these points as in Fig. 4 where T_2^a and T_{k-2}^b are the same task. (T , x , and i denote gene (task), any chromosome, and i^{th} gene of the chromosome x for T_i^x respectively).

In mutation operation, simply, a random point of the individual is selected, and updated. However, this mutation commonly produce illegal offspring since the produced chromosome is not feasible task assignments in many times for this problem. Thus, in this study, the individual is regenerated from this mutation as in Fig. 5, where T_3 is mutation point, to avoid from infeasible solutions. This means all genes after mutation point are also updated instead of only updating the gene of mutation point. Finally, the best individual is selected by applying fitness function. Fitness function calculates number of required workstations to complete the process. Then, the tasks are assigned to the workstations as in Fig. 6.

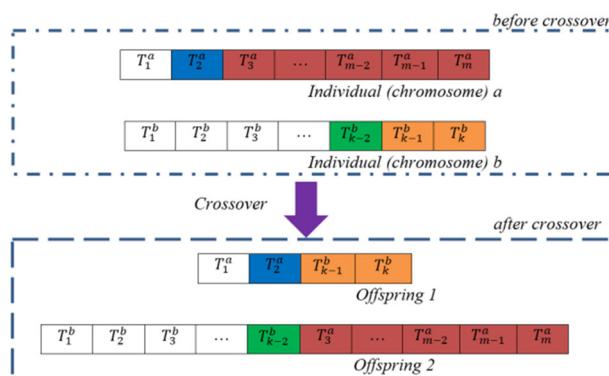
**Fig. 4** Crossover operation

Fig. 5 Mutation operation

T ₁	T ₂	T ₃	...	T _{h-2}	T _{h-1}	T _h
Before mutation						
T ₁	T ₂	T ₃	...	T _{r-2}	T _{r-1}	T _r
After mutation						

Fig. 6 Workstations of tasks for an individual

T ₁	T ₂	T ₃	...	T _{n-2}	T _{n-1}	T _n
W ₁	W ₁	W ₂	...	W _{q-1}	W _{q-1}	W _q

5 Computational analysis

The computational analysis section is divided into two parts. In the first part, a numerical example is to show problem structure and proposed methods procedure. Then, computational experiment results of the algorithms and models are illustrated in detail.

5.1 Numerical example

In a DLB problem, the uncertainty of the task times arises from the instability of human operators in terms of non-uniform work rates, skill, fatigue etc. (Agrawal & Tiwari, 2008). In the literature, the task times are assumed to be normally distributed with known means (μ_i) and variances (σ_i^2) (Agrawal & Tiwari, 2008; Altekin, 2016; Bentaha et al., 2014). The all assigned tasks to a workstation could be completed within cycle time with a confidence level (α). That means, decision maker(s) wants the probability (P) that all tasks in a workstation can be completed within cycle time, should be equal to or greater than $1 - \alpha$ (Ağpak & Gökçen, 2007; Chiang & Urban, 2006). In literature, P values are assumed in most studies as 0.90, 0.95 and 0.975 (Bagher et al., 2011; Baykasoglu & Özbakir, 2007; Celik et al., 2014; Chiang & Urban, 2006; Urban & Chiang, 2006), that is also considered in same manner in this study. For example, the solution of the P1(3, 2, 20) problem is illustrated here. The mean task time and variance of the task time are presented in Table 2.

For this example, the cycle time is 30; the coefficient of variance is 0.10, and the confidence level 0.025. The problem is solved in 2 ms. The number of the workstation is founded as 6. The solution of the problem is presented in Fig. 7. Moreover, the mean and variance of each workstation is shown on Fig. 8 for the same example.

The probability of cumulative standard normal distribution for 0.975 is presented and all is bigger than 0.975. The number of workstations is obtained 6 and the probability of the last workstation is 0.9998 that is bigger than the predetermined probability.

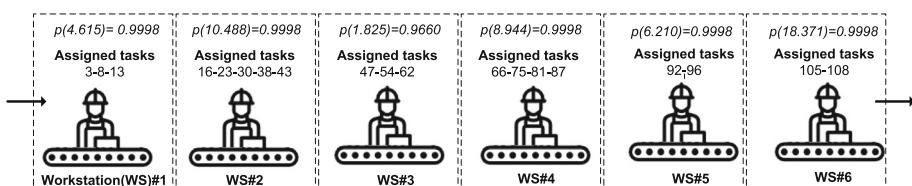
In the same manner, all test problems for small, medium, and large data set are solved that is presented in ‘Appendix 1’, ‘Appendix 2’, and ‘Appendix 3’, respectively. The performance of the proposed approaches is evaluated with benchmark test problem from the literature. The detailed information how these test problems were generated can be found in Koc et al. (2009). The variation of the task times should be change by reason of the worker. Hence, the generated deterministic task times of DLB problems are used as means (μ_i) to produce stochastic task times. The variances of the task times are acquired by multiplying a specified coefficient of variance (CV) with μ_i . Hence, we used 0.10 and 0.25 for low and high variance that is proposed by Silverman and Carter (1986). Therefore, there are 576 (576 = 96testproblems × 3confidencelevel() × 2coefficientofvariance) solutions for

Table 2 The mean and variance of task time and for P1(3, 2, 20) problem

#Task (i)	μ_i	σ_i^2	#Task (i)	μ_i	σ_i^2	#Task (i)	μ_i	σ_i^2	##Task (i)	μ_i	σ_i^2
1	15	1.5	28	12	1.2	55	17	1.7	82	8	0.8
2	17	1.7	29	9	0.9	56	4	0.4	83	2	0.2
3	3	0.3	30	3	0.3	57	9	0.9	84	16	1.6
4	16	1.6	31	14	1.4	58	19	1.9	85	12	1.2
5	12	1.2	32	1	0.1	59	11	1.1	86	15	1.5
6	9	0.9	33	20	2	60	8	0.8	87	7	0.7
7	12	1.2	34	10	1	61	18	1.8	88	17	1.7
8	12	1.2	35	4	0.4	62	10	1	89	16	1.6
9	13	1.3	36	20	2	63	10	1	90	7	0.7
10	12	1.2	37	3	0.3	64	11	1.1	91	20	2
11	10	1	38	1	0.1	65	9	0.9	92	15	1.5
12	15	1.5	39	14	1.4	66	6	0.6	93	9	0.9
13	8	0.8	40	9	0.9	67	10	1	94	11	1.1
14	18	1.8	41	15	1.5	68	3	0.3	95	9	0.9
15	9	0.9	42	16	1.6	69	18	1.8	96	6	0.6
16	6	0.6	43	4	0.4	70	19	1.9	97	12	1.2
17	19	1.9	44	13	1.3	71	20	2	98	17	1.7
18	2	0.2	45	16	1.6	72	13	1.3	99	18	1.8
19	12	1.2	46	4	0.4	73	15	1.5	100	10	1
20	15	1.5	47	13	1.3	74	13	1.3	101	14	1.4
21	13	1.3	48	8	0.8	75	4	0.4	102	20	2

Table 2 (continued)

#Task (i)	μ_i	σ_i^2	#Task (i)	μ_i	σ_i^2	#Task (i)	μ_i	σ_i^2	##Task (i)	μ_i	σ_i^2
22	13	1.3	49	3	0.3	76	5	0.5	103	10	1
23	5	0.5	50	9	0.9	77	16	1.6	104	15	1.5
24	20	2	51	6	0.6	78	3	0.3	105	13	1.3
25	19	1.9	52	15	1.5	79	9	0.9	106	1	0.1
26	9	0.9	53	9	0.9	80	6	0.6	107	20	2
27	20	2	54	4	0.4	81	1	0.1	108	2	0.2

**Fig. 7** The assignment of the P1 (3, 2, 20) problem

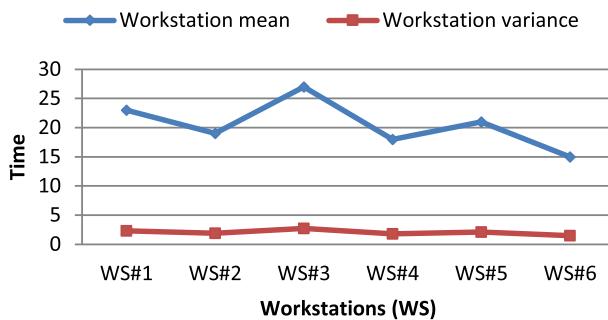


Fig. 8 Mean and variance for the P1 (3, 2, 20) problem

all test problems. We have evaluated for number of workstation and computational time (as milliseconds). The computational results present that the computational time taken is acceptable even for the large-size test problems.

5.2 Computational results

In this part, the results of the GA and constructive heuristic algorithm based on Dijkstra shortest path approach are compared with results of PLM and SA algorithm. The GA provides several advantages for solving the problem. First, it is a populated-based algorithm, so it is not dependent on the initial solution's quality. Secondly, recombination and mutation operators provide to search different areas in solution space. Thirdly, GA gives a higher chance of moving the better solutions to the next generation. In other words, it provides to increase diversity in the population to keep global best solution (Yang, 2020). Dijkstra's algorithm also provides a feasible solution efficiently enough for even relatively large problems. However, it could cause time waste due to its blind search, and it cannot handle opposing edges. On the other hand, the PLM is to prove superior performance of the graph-based algorithm and populated-based method for the DLB problem. The model cannot find a feasible solution for the large size problem with acceptable computational time. Due to the high combinatorial nature of the problem, especially for the large-size problems, GA is used. The SA algorithm provides certain advantages in terms of solution quality: (i) The general application of the algorithm is easy to build for line balancing problems. Also, it is one of the most popular methods to solve this type of problem. (ii) The number of control parameters (including T, iteration number) is limited to be easily optimized. (iii) Acceptance of worse solutions based on probability can provide to escape local optimal. (iv) On the other hand, the finding best solution depending on the initial solution and quality of the neighbouring strategy. Besides, finding the global best solution can take a lot of time (Fang et al., 2020).

First, the GA is selected because it shows relatively high performance for solving combinatorial optimization problems. Also, there are some successful implementations of DLBPs (McGovern & Gupta, 2007a; Pistolesi et al., 2017). Simulated annealing, piecewise linear model, Dijkstra algorithms are applied to show the proposed GA's superior performance. The Dijkstra algorithm is a type of greedy search technique. It is a suitable AND/OR graph structure so it can reach fair results in a short time. The simulated annealing algorithm is a probabilistic heuristic algorithm, whereas the GA is a population-based algorithm. The SA also is used for many DLBPs and ALBPs (Fang et al., 2020; Roshani & Giglio, 2017), and

it provides effective results for both problem types. Regarding the piecewise linear model, Altekin (2016) presented this technique to increase the mathematical model's computational efficiency. Considering the reasons above, these techniques, having different perspectives, were selected to compare different solution approaches' effectiveness and efficiency. We presented the cycle time constraint for stochastic DLB in Eq. (6). It ensures that completion probability of tasks in workstation within the cycle time is greater or equal than probability level. The model is linearized using most recently proposed approach of Altekin (2016) which is piecewise linear approximation. The same approach is considered in these computational results, and the results are compared with our proposed approaches. The PLM was coded in GAMS/CPLEX solver, then, the small and medium test problems are solved with 3600 s time limitation. The results of the large test problems could not present because of the complexity of the problem.

There are selection, crossover, and mutation operations in searching operation of the proposed genetic algorithm. The values of parameters for genetic algorithm were selected according to the best solution results of exhaustive search in manually determined ranges (for example: from 100 to 10,000 increasing by 10 for the number of iterations). The parameters of proposed GA used in this paper are given as follows: the population size is determined as 50 individuals, the number of iterations is selected as 1000, mutation rate was selected as 0.5 and uniform rate was determined as 0.5. On the other hand, the proposed constructive heuristic and genetic algorithms, and SA have been compiled in JavaTM Platform, Standard Edition Development Kit 9 (JDKTM). All runs are performed on personnel computer with Intel Core i7, 2.3 GHz processor, and 16 GB 1600 MHz DDR3. Each test problems are solved 10 times and the minimum number of workstations is reported. All the results are presented in '[Appendix 1](#)', '[Appendix 2](#)', and '[Appendix 3](#)'. The complexity of the proposed algorithms was computed, and the total complexity of GA is $O(|V|^2 \cdot \log(|V|))$ and the total complexity of TPA is $O(|V|^2 \cdot \log(|V|))$ found. The details of calculating the complexity of GA given as an example, where V is defined for a number of nodes.

Time complexity of the single individual generation $O(|V|^2)$

Time complexity of the population generation $O(|V|^2)$

Time complexity of the crossover $O(|V|)$

Time complexity of the mutation $O(|V|^2 \cdot \log(|V|))$

Total complexity of GA $O(|V|^2 \cdot \log(|V|))$

To design an appropriate comparison, the results of proposed algorithms including CPU times are obtained. The average relative percentage deviation (ARPD) is applied to transfer the obtained number of workstations. As different instances are solved, the ARPD is applied to transfer the obtained number of workstations utilizing average:

$$ARPD = 100 * (NW_{some} - NW_{Best}) / NW_{Best}$$

where NW_{some} is the achieved number of workstations by one algorithm and NW_{Best} is the minimum number of workstations by one algorithm. The results show that the proposed GA algorithm outperform from SA, PLM, and Dijkstra-based shortest path approach (TPA) for all test problems according to number of workstations (see Table 3). We also compared the number of workstations and CPU times for each test problems that is '[Appendix 1](#)', '[Appendix 2](#)', and '[Appendix 3](#)'. We summarized in Table 4 for small, medium, and large

Table 3 The ARPD values for small, medium, and large data set instances

Algorithms	Small data (%)	Medium data (%)	Large data (%)
PLM	21.5	—	—
SA	68.0	74.4	71.3
GA	0	0	0
TPA	24.9	23.8	20.1

Table 4 The comparative study with respect to number of workstations for medium test problems

	PLM = SA	PLM is better than SA	PLM is worse than SA
PLM vs SA	—	91	101
	TPA = PLM	TPA is better than PLM	TPA is worse than PLM
TPA vs PLM	69	97	26
	TPA = SA	TPA is better than SA	TPA is worse than SA
TPA vs SA	—	192	—

data test problems according to ARPD values.

Regarding medium data set shown in '[Appendix 2](#)', the GA outperforms TPA, PLM and SA algorithm in terms of number of workstations, finding the minimum solutions. Moreover, TPA also outperforms PLM and SA algorithm in terms of number of workstation and computational times. According to test results, proposed genetic algorithm is better than proposed TPA algorithm, PLM, and SA algorithm. On the other hand, we analyze in detail and compared the proposed TPA with PLM and SA algorithm results for medium data set with comparative manner (see Table 4 without GA results). While TPA gives better solution than PLM for 97 out of 192 test problems, TPA also gives better solution than SA for all medium data sets according to the number of workstations. In addition, TPA and PLM obtained same results for 69 medium test problems. However, TPA is worse than PLM for 26 out of 192 test problems.

As we stated above, we could not obtain the any solution for large data set with PLM. Therefore, we only compared the results of GA, TPA and SA. In this analysis, GA outperforms for all test problems in large data set that is presented in '[Appendix 3](#)'.

We also applied paired-t test to examine difference between results of the TPA, SA and PLM in Table 5. GA results are not considered in statistics test due to GA is clearly superior in all tests problem shown in Table 4. Therefore, our second proposed algorithm TPA is whether superior to SA and PLM or not. First column shows paired TPA, SA and PLM for three data sets (small, medium, and large) and mean difference is given in second column. Sample size and degree of freedom and test statistic (p -value) are demonstrated with third, fourth and

Table 5 The result of the paired t-test for number of workstations

Paired	Mean difference	N	df	p-value
TPA-PLM small data set	.04787	188	187	0.029
TPA-SA small data set	− .95213	188	187	.000
PLM-SA small data set	− 1.00000	188	817	.000
TPA-SA for medium data set	− 2.59896	192	191	.000
TPA-SA for large data set	− 4.25000	192	191	.000

last column, respectively. The sample size is 188 for the first third paired t-test because of 4 unobtained results of PLM. As a result, Table 5 presents that there is a significant difference between TPA algorithm with SA and PLM for small data set according to the number of workstation ($p < 0.05$). On the other hand, there is a significant difference into TPA and SA for both medium and large data set due to p -value (0.000) lesser than 0.05. There is also a significant difference between TPA algorithm with SA and PLM for all data set according to the CPU times ($p < 0.05$). Finally, the proposed algorithm is superior to PLM and SA algorithm with respect to number of workstations and the computational times.

6 Conclusion

In recent years, the importance of product recovery has considerably increased due to legal regulations, public awareness and so on. Therefore, DLB problem is getting more attention in theoretical and practical cases. The problem is considered under stochastic task times because an uncertainty of product conditions can cause large time variation on the same task. In this paper, DLB problem is considered in order to propose efficient solution method. Genetic algorithm and a constructive heuristic algorithm based on a shortest task time is proposed to solve the large-size problems. The proposed approaches are compared with PLM and SA to measure the performance.

The results illustrate that the proposed GA is very efficient and effective with respect to solution quality. Moreover, constructive heuristic algorithm is also effective according to computational time. To the best knowledge of the authors, there is no benchmark data set to solve stochastic DLB problem using TAOG graph under complete disassembly. Hence, a new data set is generated which is based on study of Mete et al. (2016).

The main aim of this study is to compare four different techniques such as GA, SA, constrictive heuristic, TPM, and PLM. Therefore, the heuristic algorithms are not compared to the existing literature on different problem structures. To overcome the limitation, most widely used meta-heuristic algorithms can be employed for comparison purposes, such as bee colony and particle swarm optimization algorithms. Moreover, developing new lower bounds may help the researchers to compare the proposed solution method results. Besides, the proposed algorithm can be applied to other type of disassembly lines such as U-type and parallel lines. Also, reinforcement learning methods are used as a heuristic algorithm in the literature. It can provide effective results for DLB problems due to graph structure. It should be evaluated in the future research.

Appendix 1

See Tables 6.

Table 6 The results of the stochastic DLB for small data set

$P(a, t, N)$				PLM		SA		GA		TPA	
	CT	CV	1- α	NS	CPU(s)	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P1 (3, 2, 20)	30	0.1	90	6	56	6	6	4	1351	6	1
		95	6	22		6	6	4	1344	6	1
		97.5	6	50		6	10	5	1328	6	2
		0.25	90	6	41	6	13	5	1409	6	3
		95	6	38		7	8	5	1465	6	2
	40	0.1	90	4	1.6	4	6	3	1328	4	1
		95	4	3.1		4	9	3	1290	4	1
		97.5	4	2		4	14	3	1300	4	1
		0.25	90	4	40.3	4	6	3	1367	4	1
		95	4	40		4	7	4	1332	4	1
P4 (3, 3, 20)	30	0.1	90	5	22.4	4	7	4	1324	5	1
		95	4	2		5	9	3	1948	4	1
		97.5	4	2		5	9	3	1928	4	2
		0.25	90	4	3	5	19	3	1948	4	2
		95	4	8		5	8	3	2034	4	6
	40	0.1	90	4	21	5	8	3	2030	4	2
		95	4	46		5	8	3	2017	4	2
		97.5	4	46		3	9	2	1902	3	1
		0.25	90	3	1.5	3	9	2	1898	3	1
		95	3	1		4	9	2	1892	3	2
P7 (3, 5, 20)	30	0.1	90	3	1	3	8	2	1955	3	1
		95	3	1		3	8	2	1934	3	2
		97.5	3	2.9		4	8	2	1970	3	1
	40	0.1	90	3	2	4	16	2	2707	3	2
		95	3	2		4	14	2	2740	3	2
		97.5	3	2		4	13	2	2719	3	4
	0.25	90	3	3		4	14	2	2768	3	3
		95	3	6		4	12	2	2759	3	2

Table 6 (continued)

$P(a, t, N)$				PLM		SA		GA		TPA	
	CT	CV	1- α	NS	CPU(s)	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P10 (3, 10, 20)	40	0.1	97.5	3	6	4	17	2	2790	3	2
			90	2	3	3	14	2	2688	2	3
		0.25	95	2	1.2	3	12	2	2680	2	2
			97.5	2	2.8	3	12	2	2665	2	3
			90	2	1.9	3	13	2	2789	2	4
	30	0.1	95	2	2	3	12	2	2731	2	3
			97.5	2	1	3	12	2	2805	2	2
		0.25	90	2	2	4	21	2	6954	2	7
			95	2	1	4	21	2	6836	2	6
			97.5	2	2	4	29	2	6891	2	14
P13 (4, 2, 20)	40	0.1	90	2	1	5	24	2	6901	2	16
			95	2	1	4	48	2	7002	2	7
		0.25	97.5	2	2	5	25	2	6923	2	18
			90	2	1.9	3	24	2	6830	2	7
			95	2	1.5	3	21	2	6794	2	9
	30	0.1	97.5	2	1.1	3	25	2	6921	2	9
			90	2	2.9	3	21	2	6833	2	7
		0.25	95	2	2	3	23	2	6917	2	12
			97.5	2	1.1	3	25	2	6841	2	7
			90	4	4	4	8	4	1408	4	2
P16 (4, 3, 20)	40	0.1	95	4	3	4	8	4	1406	4	1
			97.5	4	1	5	9	4	1400	4	1
		0.25	90	4	20	5	19	4	1441	4	2
			95	4	5	5	11	4	1487	4	2
			97.5	4	67	6	10	4	1468	4	2
	30	0.1	90	3	1.1	3	7	3	1341	3	1
			95	3	2.7	4	11	3	1350	3	1
		0.25	97.5	3	2.1	4	19	3	1319	3	1
			90	3	4.3	4	8	3	1393	3	1
			95	3	13.6	4	9	3	1376	3	2
		0.25	97.5	3	5.5	4	9	3	1405	3	3
			90	3	2	4	9	3	1983	3	1
			95	3	65	4	9	3	2049	3	1
		0.25	97.5	4	11	4	8	3	1985	3	3
			90	2	0.8	3	10	2	2046	4	1
			95	4	11	4	8	2	2005	3	2

Table 6 (continued)

$P(a, t, N)$	CT	CV	1- α	PLM		SA		GA		TPA	
				NS	CPU(s)	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P19 (4, 5, 20)	30	0.1	95	2	1.1	3	8	2	2011	3	1
			97.5	2	1.2	3	8	2	2000	3	1
			0.25	90	3	8.1	3	8	2	2019	3
		0.25	95	3	8.7	3	12	2	2070	3	1
			97.5	3	22.8	3	9	2	2095	3	2
			90	2	2	3	12	2	3434	2	4
	40	0.1	95	2	2	3	13	2	3416	2	3
			97.5	2	3	3	12	2	3430	3	5
			90	3	8	3	14	2	3435	3	7
		0.25	95	3	21	4	18	2	3473	3	9
			97.5	3*	15	4	14	2	3450	3	3
			95	2	21	4	18	2	3473	3	9
P22 (4, 10, 20)	30	0.1	97.5	2	15	4	14	2	3450	3	3
			90	3	22.8	3	12	2	3434	2	4
			95	2	1.6	2	13	2	3364	2	3
		0.25	95	2	1.5	3	13	2	3332	2	5
			97.5	2	1.8	2	12	2	3345	2	3
			90	2	2.2	2	12	2	3445	2	3
	40	0.1	95	2	2	2	11	2	3477	2	5
			97.5	2	2.5	3	13	2	3368	2	4
			90	2	2	3	24	1	5959	2	19
		0.25	95	2	2	3	26	2	5867	2	11
			97.5	2	3	3	23	2	5903	2	10
			90	2	2	3	32	1	6129	2	23
P25 (5, 2, 20)	30	0.1	95	2	1	3	85	2	6251	2	10
			97.5	2	3	3	24	2	5984	2	19
			90	1	1	2	56	1	5930	1	9
		0.25	95	1	1.2	2	25	1	5789	1	9
			97.5	1	1.3	2	23	1	5863	1	10
			90	1	1	2	50	1	5888	1	11
	40	0.1	95	1	1	2	23	1	5937	1	11
			97.5	1	1.4	2	24	1	6181	1	21
			90	4	3	5	8	4	1657	4	2
		0.25	95	4	3	5	7	4	1655	4	1
			97.5	4	8	6	17	4	1693	4	1
			90	4	7	5	7	4	1671	4	3
		0.25	95	4	13	6	24	4	1725	4	1
			97.5	5	72	6	8	4	1823	4	2
			90	3	1.7	4	7	3	1630	3	1

Table 6 (continued)

$P(a, t, N)$	CT	CV	1- α	PLM		SA		GA		TPA		
				NS	CPU(s)	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)	
P28 (5, 3, 20)	30	0.1	95	3	1.4	4	7	3	1632	3	1	
			97.5	3	3.8	4	7	3	1626	3	2	
			0.25	90	3	2.4	4	7	3	1649	3	1
			95	3	7	4	7	3	1713	3	1	
			97.5	4	39.7	4	7	3	1743	3	4	
	40	0.1	90	4	3	5	10	3	2735	4	2	
			95	4	5	5	10	3	2714	4	2	
			97.5	4	3	5	22	2	2684	4	4	
			0.25	90	4	12	5	9	3	2718	4	5
			95	4	107	5	9	3	2747	4	2	
P31 (5, 5, 20)	30	0.1	97.5	4	132	5	8	3	2826	4	2	
			90	3	1.8	3	9	2	2624	3	2	
			95	3	1.2	3	11	2	2654	3	2	
			97.5	3	2.1	3	9	2	2651	3	2	
			0.25	90	3	33	3	10	2	2676	3	2
	40	0.1	95	3	13.8	3	10	2	2726	3	2	
			97.5	3	15.6	3	9	2	2706	3	2	
			90	2	4	4	13	2	4260	3	6	
			95	2	2	4	14	2	4274	3	4	
			97.5	2	6	4	28	2	4180	3	8	
P34 (5, 10, 20)	30	0.1	0.25	90	3	115	4	12	2	4323	3	7
			95	3	35	4	13	2	4271	3	10	
			97.5	3*	28	4	13	2	4380	3	4	
			90	2	1.9	3	13	2	4268	2	5	
			95	2	2.5	3	15	2	4191	2	6	
	40	0.1	97.5	2	2.6	3	12	2	4215	2	4	
			0.25	90	2	1.8	3	14	2	4276	2	4
			95	2	3.9	3	13	2	4218	2	4	
			97.5	2	7.3	3	13	2	4304	2	6	
			90	2	2	3	35	2	7768	2	13	
			95	2	3	3	24	2	7856	2	11	
			97.5	2	3	3	24	2	7874	2	11	
			0.25	90	2	3	3	29	2	8595	2	15
			95	2	2	4	57	2	8481	2	12	

Table 6 (continued)

$P(a, t, N)$				PLM		SA		GA		TPA	
	CT	CV	1- α	NS	CPU(s)	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P37 (10, 2, 20)	40	0.1	97.5	2	2	3	24	2	8035	2	13
			90	2	2.6	2	52	1	8151	2	14
		0.25	95	2	2.4	2	28	1	7842	2	12
			97.5	2	2.3	2	20	2	7835	2	13
			90	2	2.7	2	34	1	7905	2	14
	30	0.1	95	2	2.7	2	24	1	7916	2	12
			97.5	2	2.2	3	34	1	7890	2	25
		0.25	90	5	44	7	8	5	2741	5	5
			95	5	100	7	8	5	2735	5	5
			97.5	5	70	8	10	4	2682	5	5
P40 (10, 3, 20)	40	0.1	90	5	28	8	8	5	2685	5	5
			95	7*	+ 3600	8	8	5	2905	5	6
		0.25	97.5	6	3139	9	10	5	2778	5	6
			90	4	10.8	5	18	3	2653	4	2
			95	4	16	5	17	3	2647	4	2
	30	0.1	97.5	4	22.4	5	12	3	2654	4	3
			90	4	31.6	6	8	3	2616	4	3
		0.25	95	4	29	5	9	3	2574	4	2
			97.5	4	235	6	8	3	2753	4	2
			90	3	9	5	9	2	3677	3	4
	40	0.1	95	3	5	5	9	2	3671	3	4
			97.5	3	11	5	10	3	3529	3	9
		0.25	90	3	41	5	23	3	3535	3	104
			95	3	164	5	12	3	3555	3	6
			97.5	3	86	5	20	3	3619	3	5

Table 6 (continued)

$P(a, t, N)$				PLM		SA		GA		TPA		
	CT	CV	1- α	NS	CPU(s)	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)	
P43 (10, 5, 20)	30	0.1	97.5	2	5.8	4	10	2	3650	2	4	
			90	2	7	4	16	2	5881	3	8	
			95	2	8	4	14	2	5961	3	9	
			97.5	2	10	5	14	2	6003	3	14	
			0.25	90	3	87	4	14	2	5937	3	10
	40	0.1	95	3	58	4	16	2	5972	3	9	
			97.5	3	97	5	14	2	5880	3	8	
			90	2	11.3	3	20	2	5899	2	11	
			95	2	8.9	3	14	1	5859	2	9	
			97.5	2	9.2	3	15	1	5864	2	7	
P46 (10, 10, 20)	30	0.1	0.25	90	2	6.7	3	14	1	5991	2	8
			95	2	6.7		14	2	5898	2	7	
			97.5	2	6	3	14	1	5816	2	7	
			90	2	7	3	28	2	18,024	2	52	
			95	2	11	3	33	2	17,797	2	32	
	40	0.1	97.5	2	8	3	28	2	17,530	2	28	
			0.25	90	2	8	3	28	2	17,419	2	55
			95	2	9	3	28	2	17,574	2	33	
			97.5	2	35	3	48	2	17,782	2	49	
			90	1	5.1	3	69	1	17,594	2	31	

Appendix 2

See Table 7.

Table 7 The results of the stochastic DLB for medium data set

P (a, t, N)	CT	CV	1- α	PLM		SA		GA		TPA		
				NS	CPU(s)	NS	CPU(ms)))	NS	CPU(ms)	NS	CPU(ms)	
P2 (3, 2, 50)	30	0.1	90	10	2100	15	22	9	4165	12	3	
			95	10	1333	15	21	9	4127	12	3	
			97.5	**	+ 3600	16	21	9	4022	13	2	
			0.25	90	**	+ 3600	16	21	9	4189	13	2
			95	**	+ 3600	16	23	10	4192	13	2	
	40	0.1	97.5	**	+ 3600	16	21	9	4233	13	3	
			90	8	3472	11	46	6	4046	9	3	
			95	8	3113	12	21	6	4088	9	2	
			97.5	8	3553	12	22	7	4071	9	2	
			0.25	90	**	+ 3600	12	20	7	4127	9	2
P5 (3, 3, 50)	30	0.1	95	**	+ 3600	12	21	7	4172	9	2	
			97.5	**	+ 3600	12	20	7	4095	9	5	
			90	8	2205	12	24	7	5948	9	3	
			95	8	261	12	26	7	6003	9	3	
			97.5	8	850	13	25	7	5898	9	5	
	40	0.1	0.25	90	**	+ 3600	13	25	7	5922	9	8
			95	**	+ 3600	13	26	8	6056	9	3	
			97.5	**	+ 3600	13	32	9	5990	9	3	
			90	6	2181	9	45	5	5809	7	3	
			95	6	386	9	25	6	5768	7	4	
P8 (3, 5, 50)	30	0.1	97.5	6	127	9	27	6	5845	7	3	
			0.25	90	7	2283	9	25	5	5863	7	3
			95	7	3217	10	25	6	5782	7	3	
			97.5	**	+ 3600	9	30	6	5916	7	3	
			90	6	751	8	52	6	10,348	7	6	
	40	0.1	95	6	1903	8	62	6	10,317	7	6	
			97.5	7*	+ 3600	9	36	6	10,256	7	12	
			0.25	90	7*	+ 3600	8	36	6	10,387	7	12
			95	**	+ 3600	9	60	6	10,513	8	13	
			97.5	**	+ 3600	9	36	6	10,470	8	7	

Table 7 (continued)

P (a , t , N)	CT	CV	1- α	PLM		SA		GA		TPA		
				NS	CPU(s)	NS	CPU(ms)))	NS	CPU(ms)	NS	CPU(ms)	
P11 (3, 10, 50)	30	0.1	97.5	5	253	6	46	4	10,178	5	8	
			0.25	90	** + 3600	6	75	4	10,498	5	6	
			95	6*	+ 3600	6	54	5	10,318	5	7	
			97.5	5	704	6	54	5	10,322	5	6	
			90	4	103	6	124	4	25,506	4	40	
	40	0.1	95	4	35	6	76	4	27,111	4	27	
			97.5	4	95	6	85	4	26,024	4	29	
			0.25	90	4	306	6	73	4	26,416	4	46
			95	4	459	6	66	4	26,960	4	29	
			97.5	4	446	6	66	4	26,951	4	38	
P14 (4, 2, 50)	30	0.1	90	3	49	4	83	3	25,389	3	33	
			95	3	29	4	73	3	25,961	3	27	
			97.5	3	53	4	58	3	25,744	3	26	
			0.25	90	3	41	4	59	3	25,405	3	28
			95	3	23	4	73	3	26,896	3	28	
	40	0.1	97.5	3	125	4	58	3	25,832	3	51	
			90	**	+ 3600	16	23	7	5141	11	2	
			95	**	+ 3600	16	23	8	5165	11	3	
			97.5	**	+ 3600	16	23	9	5047	12	5	
			0.25	90	**	+ 3600	16	50	8	5259	12	14
P17 (4, 3, 50)	30	0.1	95	**	+ 3600	16	24	8	5457	12	2	
			97.5	**	+ 3600	17	24	8	5123	12	4	
			90	8	1016	10	26	6	5049	8	2	
			95	8	3255	10	54	6	5008	8	3	
			97.5	**	+ 3600	10	24	7	4932	8	3	
	40	0.1	90	**	+ 3600	10	35	6	5122	8	3	
			95	**	+ 3600	11	27	6	5103	9	3	
			97.5	**	+ 3600	11	24	7	5120	9	3	
			90	**	+ 3600	10	23	7	7574	9	4	
			95	8	963	11	26	7	7681	9	4	
			97.5	8	3353	12	29	7	7598	9	11	
			0.25	90	**	+ 3600	12	25	6	7666	9	21
			95	**	+ 3600	12	74	7	7667	9	8	
			97.5	**	+ 3600	13	61	8	7659	9	4	

Table 7 (continued)

P (a , t , N)	CT	CV	1- α	PLM		SA		GA		TPA	
				NS	CPU(s)	NS	CPU(ms)))	NS	CPU(ms)	NS	CPU(ms)
P20 (4, 5, 50)	40	0.1	90	6	641	8	28	5	7414	6	5
			95	6	169	8	63	5	7532	6	4
			97.5	6	327	8	26	5	7503	6	4
		0.25	90	**	+ 3600	8	25	5	7717	6	4
			95	**	+ 3600	8	65	5	7624	7	6
			97.5	**	+ 3600	8	36	6	7589	7	6
	30	0.1	90	6	355	9	36	5	12,998	7	8
			95	6	1576	9	33	5	13,074	7	8
			97.5	6	943	9	71	6	12,948	7	18
		0.25	90	7	3543	9	52	6	13,034	7	14
			95	**	+ 3600	9	45	6	13,355	8	9
			97.5	**	+ 3600	9	69	6	13,031	8	9
P23 (4, 10, 50)	40	0.1	90	5	393	7	72	4	12,842	5	12
			95	5	1831	7	61	4	12,814	5	9
			97.5	5	2131	7	71	4	12,797	5	9
		0.25	90	5	3370	7	37	4	12,939	5	11
			95	5	3430	7	42	4	12,886	5	10
			97.5	**	+ 3600	7	44	5	12,995	5	9
	30	0.1	90	4	81	7	97	4	32,696	4	30
			95	4	605	6	92	4	33,372	4	35
			97.5	4	323	7	95	4	32,668	4	57
		0.25	90	4	1638	7	143	4	35,079	4	80
			95	4	1359	7	105	4	31,950	4	36
			97.5	4	1702	10	137	4	33,189	4	37
P26 (5, 2, 50)	40	0.1	90	3	10	5	75	3	32,590	3	36
			95	3	87	5	72	3	33,139	3	37
			97.5	3	305	5	87	3	33,331	3	37
		0.25	90	3	966	5	78	3	33,488	3	36
			95	3	161	5	80	3	33,584	3	51
			97.5	3	190	5	80	3	34,112	3	38
	30	0.1	90	**	+ 3600	16	22	9	6204	11	3
			95	**	+ 3600	16	20	8	6085	11	3
			97.5	**	+ 3600	16	20	10	6023	12	5
		0.25	90	**	+ 3600	16	20	9	5972	12	7
			95	**	+ 3600	16	20	9	6224	13	4

Table 7 (continued)

P (a , t , N)	CT	CV	1- α	PLM		SA		GA		TPA	
				NS	CPU(s)	NS	CPU(ms)))	NS	CPU(ms)	NS	CPU(ms)
P29 (5, 3, 50)	40	0.1	97.5	**	+ 3600	17	46	9	6091	13	5
			90	8	1039	10	31	7	5907	8	3
		0.25	95	8	1439	11	20	7	5838	9	3
			97.5	8	216	11	22	7	5934	9	3
			90	**	+ 3600	11	20	6	5981	9	3
	30	0.1	95	**	+ 3600	11	22	6	5931	9	3
			97.5	**	+ 3600	12	29	8	5977	9	3
			90	7	277	11	33	7	9073	7	5
		0.25	95	7	1662	12	27	6	9032	7	5
			97.5	7	272	11	31	6	8937	7	10
P32 (5, 5, 50)	40	0.1	90	**	+ 3600	10	31	6	8974	7	11
			95	**	+ 3600	12	30	6	9029	8	9
			97.5	**	+ 3600	12	37	7	9127	8	6
		0.25	90	5	78	7	31	5	8810	6	6
			95	5	518	8	36	5	8955	6	6
	30	0.1	97.5	5	309	8	30	5	8862	6	8
			90	6	1332	8	27	5	9000	6	5
			95	6	1782	8	27	5	8936	6	7
		0.25	97.5	**	+ 3600	9	28	5	9061	6	5
			90	6	1630	9	69	5	17,836	6	14
	P32 (5, 5, 50)	40	95	6	1128	9	47	5	17,778	6	10
			97.5	6	1398	9	47	5	17,538	7	24
		0.1	90	**	+ 3600	9	113	5	18,138	7	15
			95	**	+ 3600	10	47	5	18,079	7	12
			97.5	**	+ 3600	10	76	6	18,289	7	12
		30	90	4	693	7	45	4	17,647	5	15
			95	5	1193	7	51	4	17,753	5	13
			97.5	5	1554	6	51	4	17,973	5	13
		0.25	90	5	2061	6	40	4	17,711	5	12
			95	6*	+ 3600	7	46	4	17,874	5	12
			97.5	**	+ 3600	7	47	4	18,176	5	12

Table 7 (continued)

P (a , t , N)	CT	CV	1- α	PLM		SA		GA		TPA		
				NS	CPU(s)	NS	CPU(ms)))	NS	CPU(ms)	NS	CPU(ms)	
P35 (5, 10, 50)	30	0.1	90	4	268	6	125	4	42,851	4	50	
			95	4	688	7	141	4	42,988	4	44	
			97.5	**	+ 3600	6	169	4	44,420	4	45	
			0.25	90	4	1764	6	103	4	42,819	4	42
			95	4	1023	6	128	4	42,696	5	45	
	40	0.1	97.5	5*	+ 3600	7	145	4	42,438	5	42	
			90	3	265	4	87	3	42,260	3	42	
			95	3	250	5	87	3	43,748	3	39	
			97.5	3	43	4	127	3	42,631	3	46	
			0.25	90	3	1029	4	86	3	43,772	3	41
P38 (10, 2, 50)	30	0.1	95	3	358	5	80	3	42,997	3	46	
			97.5	3	1245	5	75	3	43,605	3	69	
			95	**	+ 3600	14	50	8	9993	11	12	
			97.5	**	+ 3600	15	26	8	10,003	11	7	
			0.25	90	**	+ 3600	16	33	8	9922	12	6
	40	0.1	95	**	+ 3600	17	30	7	10,006	12	7	
			97.5	**	+ 3600	19	22	8	10,004	12	7	
			95	**	+ 3600	19	55	8	9960	12	13	
			97.5	**	+ 3600	10	23	6	9810	8	7	
			95	**	+ 3600	11	25	6	9832	8	6	
P41 (10, 3, 50)	30	0.1	97.5	8*	+ 3600	11	32	6	9823	8	6	
			90	**	+ 3600	11	22	6	9966	8	6	
			95	**	+ 3600	12	22	6	9899	8	6	
			97.5	**	+ 3600	12	22	6	10,116	9	17	
			90	**	+ 3600	13	41	5	16,602	9	12	

Table 7 (continued)

P (a , t , N)	CT	CV	1- α	PLM		SA		GA		TPA	
				NS	CPU(s)	NS	CPU(ms)))	NS	CPU(ms)	NS	CPU(ms)
40	0.1	0.25	95	**	+ 3600	13	29	6	16,604	9	10
			97.5	**	+ 3600	13	33	6	16,555	9	18
			90	**	+ 3600	13	31	6	16,716	9	21
		0.25	95	**	+ 3600	15	35	6	16,666	9	10
			97.5	**	+ 3600	15	35	6	16,643	9	13
			90	**	+ 3600	9	32	4	16,430	6	12
	0.25	0.25	95	**	+ 3600	10	33	5	16,375	6	13
			97.5	**	+ 3600	10	36	4	16,554	6	12
			90	**	+ 3600	10	26	5	16,575	6	10
		0.1	95	**	+ 3600	10	29	4	16,574	7	12
			97.5	**	+ 3600	10	29	5	16,532	7	12
			90	**	+ 3600	9	70	4	31,658	5	28
P44 (10, 5, 50)	0.1	0.25	95	**	+ 3600	8	43	4	31,850	5	28
			97.5	**	+ 3600	9	48	4	31,592	5	46
			90	**	+ 3600	9	47	4	32,358	5	37
		0.25	95	**	+ 3600	10	65	4	32,352	6	56
			97.5	**	+ 3600	9	52	4	31,925	6	23
			90	6*	+ 3600	6	94	3	31,807	4	33
	0.25	0.1	95	4	746	6	51	3	32,290	4	28
			97.5	**	+ 3600	6	55	3	33,116	4	28
			90	**	+ 3600	6	47	3	32,393	4	30
		0.25	95	**	+ 3600	6	47	4	32,149	4	30
			97.5	**	+ 3600	7	47	4	31,667	4	30
			90	**	+ 3600	6	88	3	83,036	4	96
P47 (10, 10, 50)	0.1	0.25	95	**	+ 3600	6	97	3	76,878	4	93
			97.5	**	+ 3600	7	80	3	82,116	4	100
			90	**	+ 3600	6	142	3	76,967	4	113
		0.25	95	**	+ 3600	7	99	4	83,415	4	102
			97.5	**	+ 3600	7	122	4	83,648	4	98
			90	3	631	4	151	3	83,461	3	103
	0.25	0.1	95	3	276	5	96	3	81,461	3	103
			97.5	3	243	5	150	3	77,472	3	98
			90	**	+ 3600	5	96	3	81,936	3	92
		0.25	95	**	+ 3600	5	84	3	82,577	3	99
			97.5	**	+ 3600	5	95	3	80,841	3	162

Appendix 3

See Table 8.

Table 8 The results of the stochastic DLB for large data set

$P(a, t, N)$	CT	CV	1- α	SA		GA		TPA	
				NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P3 (3, 2, 80)	30	0.1	90	27	56	18	6885	24	6
			95	27	33	20	7009	24	3
			97.5	30	32	20	6774	25	3
			0.25	90	30	34	21	7092	25
			95	31	47	21	7388	25	5
	40	0.1	97.5	31	37	20	7249	25	3
			90	19	38	13	6840	17	3
			95	20	38	13	6771	17	3
			97.5	20	40	14	6876	17	3
			0.25	90	20	33	13	7078	17
P6 (3, 3, 80)	30	0.1	95	21	35	13	7014	18	4
			97.5	21	34	14	6975	18	5
			90	18	85	12	10,178	15	5
			95	19	90	12	9986	15	5
			97.5	19	44	11	9995	15	9
	40	0.1	0.25	90	19	48	12	10,054	15
			95	21	72	14	11,096	15	7
			97.5	21	43	14	10,157	15	5
			90	13	78	8	9857	10	6
			95	13	48	9	9813	11	5
P9 (3, 5, 80)	30	0.1	97.5	13	44	9	9928	11	6
			90	13	49	8	9921	11	5
			95	14	44	9	9956	11	4
			97.5	15	96	10	10,000	11	6
			90	14	70	9	17,988	11	12
	40	0.1	95	14	70	10	17,704	11	12
			97.5	15	63	10	17,809	11	23
			0.25	90	15	127	10	17,592	11
			95	16	121	11	17,885	11	25
			97.5	15	93	11	17,885	11	12

Table 8 (continued)

$P(a, t, N)$			SA		GA		TPA			
	CT	CV	1- α	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)	
P12 (3, 10, 80)	30	0.1	97.5	11	77	7	17,787	8	17	
			0.25	90	10	70	7	18,006	8	12
			95	11	70	8	17,871	8	27	
		0.25	97.5	11	133	8	17,868	9	10	
			90	9	140	5	41,259	6	60	
			95	9	187	5	42,263	6	41	
	40	0.1	97.5	10	120	6	42,389	6	43	
			0.25	90	9	178	6	43,345	6	52
			95	10	195	6	42,827	6	42	
		0.25	97.5	10	169	6	42,190	6	55	
			90	6	127	4	42,627	4	45	
			95	7	115	4	42,702	5	46	
P15 (4, 2, 80)	30	0.1	97.5	7	126	4	42,305	5	46	
			0.25	90	9	123	4	42,671	5	43
			95	7	132	4	42,924	5	44	
		0.25	97.5	7	247	4	44,108	5	79	
			90	25	41	14	8713	20	6	
			95	26	40	13	8696	20	5	
	40	0.1	97.5	27	47	14	8861	21	5	
			0.25	90	27	44	14	8859	21	7
			95	27	44	15	8964	21	4	
		0.25	97.5	27	84	15	8860	21	5	
			90	18	54	10	8605	13	5	
			95	18	67	11	8538	14	4	
P18 (4, 3, 80)	30	0.1	97.5	19	50	11	8692	14	4	
			0.25	90	19	40	10	8718	14	4
			95	19	43	11	8747	14	4	
		0.25	97.5	19	39	12	8740	15	12	
			90	18	48	11	13,239	13	8	
			95	18	48	11	13,167	13	7	
	40	0.1	97.5	19	44	11	13,233	14	12	
			0.25	90	19	56	11	13,149	14	11
			95	19	87	12	15,997	15	13	
		0.25	97.5	19	56	12	13,505	15	6	
			90	13	48	8	12,936	10	9	

Table 8 (continued)

$P(a, t, N)$			SA		GA		TPA			
	CT	CV	1- α	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)	
P21 (4, 5, 80)	30	0.1	95	13	48	8	12,981	10	7	
			97.5	13	51	8	13,034	10	7	
			0.25	90	13	54	8	13,362	10	8
		0.25	95	14	92	9	13,264	10	6	
			97.5	14	53	8	13,257	10	8	
			95	17	110	8	22,270	10	17	
	40	0.1	97.5	17	68	8	22,238	10	15	
			0.25	90	18	83	8	22,166	10	24
			95	18	68	9	22,014	11	16	
		0.25	97.5	18	120	8	22,587	11	19	
			95	12	63	6	21,924	7	19	
			97.5	12	58	6	22,022	7	16	
P24 (4, 10, 80)	30	0.1	97.5	12	82	6	21,855	7	17	
			0.25	90	11	67	6	22,140	7	14
			95	13	69	6	22,206	7	21	
		0.25	97.5	13	68	7	22,146	8	62	
			95	10	144	6	59,736	7	60	
			97.5	10	256	6	60,437	7	55	
	40	0.1	97.5	10	169	6	59,427	8	74	
			0.25	90	10	137	7	59,215	8	77
			95	11	235	7	60,003	8	61	
		0.25	97.5	11	203	7	60,800	8	56	
			95	7	192	5	60,473	5	71	
			97.5	8	142	5	59,227	6	63	
P27 (5, 2, 80)	30	0.1	97.5	8	194	5	60,376	6	65	
			0.25	90	7	125	5	60,000	6	63
			95	8	264	5	62,944	6	75	
		0.25	97.5	8	175	5	62,835	6	18	
			95	21	33	13	10,433	16	6	
			97.5	21	36	13	10,480	16	5	
	40	0.1	97.5	22	34	14	10,395	17	6	
			0.25	90	21	40	16	11,106	17	11
			95	23	35	14	10,421	18	5	
		0.25	97.5	23	78	14	10,381	18	5	
			95	15	59	10	10,219	12	5	

Table 8 (continued)

$P(a, t, N)$			SA		GA		TPA		
	CT	CV	1- α	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P30 (5, 3, 80)	30	0.1	95	15	35	10	10,237	13	7
			97.5	15	38	10	10,312	13	5
			0.25	90	15	32	10	10,369	13
			95	15	35	10	10,380	13	5
			97.5	17	33	11	10,373	13	6
	40	0.1	90	20	55	10	16,147	13	9
			95	19	49	10	16,125	13	8
			97.5	20	50	11	15,977	13	18
			0.25	90	19	51	12	16,168	13
			95	20	60	12	16,367	14	17
P33 (5, 5, 80)	30	0.1	97.5	20	57	11	16,267	14	8
			0.25	90	19	51	12	16,168	13
			95	20	60	12	16,367	14	17
			97.5	20	57	11	16,267	14	8
			40	0.1	90	14	49	8	16,027
	40	0.25	95	14	47	9	16,018	10	10
			97.5	14	50	9	15,909	10	10
			90	14	47	9	16,189	10	10
			95	14	47	9	16,211	10	11
			97.5	15	47	9	16,163	10	12
P36 (5, 10, 80)	30	0.1	90	13	86	8	29,781	8	22
			95	13	83	8	29,714	8	25
			97.5	13	158	8	30,333	9	34
			0.25	90	13	80	8	29,750	9
			95	14	224	8	30,075	9	20
	40	0.1	97.5	15	98	8	30,091	9	21
			90	9	66	6	29,611	6	24
			95	9	95	6	29,687	6	23
			97.5	9	79	6	29,606	6	24
			0.25	90	9	79	6	29,799	6
	30	0.25	95	10	125	6	30,501	7	23
			97.5	10	124	6	29,991	7	23
			90	10	169	6	74,242	6	127
			95	9	196	6	75,733	6	77
			97.5	10	129	6	76,948	6	82

Table 8 (continued)

$P(a, t, N)$				SA		GA		TPA	
	CT	CV	1- α	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)
P39 (10, 2, 80)	40	0.1	97.5	10	168	6	76,299	7	143
			90	7	223	4	73,989	5	88
			95	7	113	4	73,173	5	86
		0.25	97.5	7	188	4	75,051	5	80
			90	7	114	4	78,670	5	75
	30	0.1	95	7	154	5	74,772	5	84
			97.5	7	235	5	78,679	5	153
			90	30	36	13	18,999	17	20
		0.25	95	30	36	16	19,406	17	10
			97.5	32	40	15	19,223	19	12
P42 (10, 3, 80)	40	0.1	90	31	52	15	19,382	19	25
			95	32	55	14	19,107	22	13
			97.5	32	51	16	19,109	22	26
		0.25	90	21	37	11	19,044	13	13
			95	22	86	10	18,572	14	11
	30	0.1	97.5	22	47	10	19,125	14	11
			90	21	35	12	18,997	14	11
			95	22	53	11	18,992	14	13
		0.25	97.5	23	44	11	19,223	14	30
			90	16	42	9	28,572	12	18
		0.1	95	17	46	9	28,394	12	22
			97.5	17	45	8	28,567	12	35
			90	17	96	9	28,800	12	38
		0.25	95	18	78	9	32,002	12	22
			97.5	18	48	9	29,030	12	23
			90	12	59	6	28,282	8	27
		0.25	95	12	78	6	28,728	8	20
			97.5	12	46	6	28,580	9	21
			90	12	88	7	28,361	9	23
		0.1	95	12	53	7	29,238	9	21

Table 8 (continued)

$P(a, t, N)$			SA		GA		TPA			
	CT	CV	1- α	NS	CPU(ms)	NS	CPU(ms)	NS	CPU(ms)	
P45 (10, 5, 80)	30	0.1	97.5	12	44	8	28,989	9	22	
			90	13	84	6	62,152	8	50	
			95	13	86	7	58,550	8	49	
			97.5	13	148	7	58,580	8	81	
			0.25	90	13	84	6	58,746	8	152
	40	0.1	95	14	96	7	59,700	9	83	
			97.5	14	85	7	59,073	9	52	
			90	9	76	5	60,130	6	56	
			95	9	134	5	61,506	6	52	
			97.5	10	100	5	60,384	6	62	
P48 (10, 10, 80)	30	0.1	0.25	90	9	87	5	59,701	6	51
			95	10	99	5	61,531	6	67	
			97.5	10	86	5	58,595	6	51	
			90	9	185	5	155,361	5	167	
			95	9	174	5	148,174	5	178	
	40	0.1	97.5	10	182	5	151,031	6	263	
			0.25	90	10	342	5	163,915	6	187
			95	9	308	5	163,812	6	171	
			97.5	10	299	5	150,497	6	166	
			90	6	139	4	154,175	4	180	

References

- Ağpak, K., & Gökçen, H. (2007). A chance-constrained approach to stochastic line balancing problem. *European Journal of Operational Research*, 180(3), 098–1115.
- Agrawal, S., & Tiwari, M. K. (2008). A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research*, 46(6), 1405–1429.
- Altekin, F. T. (2016). A piecewise linear model for stochastic disassembly line balancing. *IFAC-Papers on Line*, 49(12), 932–937.
- Altekin, F. T., & Akkan, C. (2012). Task-failure-driven rebalancing of disassembly lines. *International Journal of Production Research*, 50(18), 4955–4976.
- Altekin, F. T., Bayindir, Z. P., & Gümüşkaya, V. (2016). Remedial actions for disassembly lines with stochastic task times. *Computers and Industrial Engineering*, 99, 78–96.
- Altekin, F. T., Kandiller, L., & Ozdemirel, N. E. (2008). Profit-oriented disassembly line balancing. *International Journal of Production Research*, 46(10), 2675–2693.
- Aydemir-Karadag, A., & Turkbey, O. (2013). Multi-objective optimization of stochastic disassembly line balancing with station paralleling. *Computers & Industrial Engineering*, 65, 413–425.
- Bagher, M., Zandieh, M., & Farsijani, H. (2011). Balancing of stochastic U-type assembly lines: An imperialist competitive algorithm. *International Journal of Advanced Manufacturing Technology*, 54, 271–285.
- Baykasoglu, A., & Özbakır, L. (2007). Stochastic U-line balancing using genetic algorithms. *International Journal of Advanced Manufacturing Technology*, 32, 139–147.
- Bentaha, M. L., Battaila, O., & Dolgui, A. (2014). A sample average approximation method for disassembly line balancing problem under uncertainty. *Computers and Operations Research*, 51, 111–122.
- Celik, E., Kara, Y., & Atasagun, Y. (2014). A new approach for rebalancing of U-lines with stochastic task times using ant colony optimization algorithm. *International Journal of Production Research*, 52(24), 7262–7275.
- Chiang, W. C., & Urban, T. L. (2006). The stochastic u-line balancing problem: A heuristic procedure. *European Journal of Operational Research*, 175(3), 1767–1781.
- Çil, Z. A., Kızılıy, D., Li, Z., & Öztop, H. (2022). Two-sided disassembly line balancing problem with sequence-dependent setup time: A constraint programming model and artificial bee colony algorithm. *Expert Systems with Applications*, 203, 117529.
- Çil, Z. A., Mete, S., & Serin, F. (2020). Robotic disassembly line balancing problem: A mathematical model and ant colony optimization approach. *Applied Mathematical Modelling*, 86, 335–348.
- Deniz, N., & Ozcelik, F. (2019). An extended review on disassembly line balancing with bibliometric & social network and future study realization analysis. *Journal of Cleaner Production*, 225(1), 697–715.
- Ding, L.-P., Tan, J.-R., Feng, Y.-X., & Gao, Y.-C. (2009). Multi objective optimization for disassembly line balancing based on pareto ant colony algorithm. *Computer Integrated Manufacturing Systems*, 15(7), 1406–1413.
- Fang, Y., Ming, H., Li, M., Liu, Q., & Pham, D. T. (2020). Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time. *International Journal of Production Research*, 58(3), 846–862.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.
- Güngör, A., and Gupta, S.M., 1999, Disassembly line balancing, Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute, 193–19.
- Güngör, A., & Gupta, S. M. (2001). A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*, 39(7), 142–1467.
- He, J., Chu, F., Zheng, F., & Liu, M. (2021). A green-oriented bi-objective disassembly line balancing problem with stochastic task processing times. *Annals of Operations Research*, 296, 71–93.
- Hezer, S., & Kara, Y. (2015). A network-based shortest route model for parallel disassembly line balancing problem. *International Journal of Production Research*, 53(6), 1849–1865.
- Ilgin, M. A., Akçay, H., & Araz, C. (2017). Disassembly line balancing using linear physical programming. *International Journal of Production Research*, 55(20), 6108–6119.
- Ilgin, M. A., & Gupta, S. M. (2010). Environmentally conscious manufacturing and product recovery (ECM-PRO): A review of the state of the art. *Journal of Environmental Management*, 91, 563–591.
- Kalayci, C. B., & Gupta, S. M. (2013a). 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.
- Kalayci, C. B., & Gupta, S. M. (2013b). Ant colony optimization for sequence-dependent disassembly line balancing problem. *Journal of Manufacturing Technology Management*, 24(3), 413–427.

- Kalaycilar, E. G., Azizoğlu, M., & Yeralan, S. (2016). A disassembly line balancing problem with fixed number of workstations. *European Journal of Operational Research*, 249(2), 592–604.
- Kazancoglu, Y., & Ozturkoglu, Y. (2018). Integrated framework of disassembly line balancing with Green and business objectives using a mixed MCDM. *Journal of Cleaner Production*, 191, 179–191.
- Ketzenberg, M. E., Souza, G. C., & Guide, V. D. R. (2003). Mixed assembly and disassembly operations for remanufacturing'. *Production and Operations Management*, 12(3), 320–335.
- Koç, A., Sabuncuoğlu, I., & Erel, E. (2009). Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. *IIE Transactions*, 41(10), 866–881.
- Li, Z., Çil, Z. A., Mete, S., & Kucukkoc, I. (2020). A fast branch, bound and remember algorithm for disassembly line balancing problem. *International Journal of Production Research*, 58(11), 3220–3234.
- Li, Z., Kucukkoc, I., & Zhang, Z. (2019). Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem. *Computers & Industrial Engineering*. <https://doi.org/10.1016/j.cie.2019.106056>
- Liu, M., Liu, X., Chu, F., Zheng, F., & Chu, C. 2020b. An exact method for disassembly line balancing problem with limited distributional information. *International Journal of Production Research*, 1–18.
- Liu, M., Liu, X., Chu, F., Zheng, F., & Chu, C. (2020a). Robust disassembly line balancing with ambiguous task processing times. *International Journal of Production Research*, 58(19), 5806–5835.
- McGovern, S. M., & Gupta, S. M. (2007a). A balancing method and genetic algorithm for disassembly line balancing. *European Journal of Operational Research*, 179(3), 692–708.
- McGovern, S. M., & Gupta, S. M. (2007b). Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. *International Journal of Production Research*, 45(18–19), 4485–4511.
- Mete, S., Çil, Z. A., Ağpak, K., Özceylan, E., & Dolgui, A. (2016). A solution approach based on beam search algorithm for disassembly line balancing problem. *Journal of Manufacturing Systems*, 41(1), 188–200.
- Mete, S., Çil, Z. A., Celik, E., & Ozceylan, E. (2019). Supply-driven rebalancing of disassembly lines: A novel mathematical model approach. *Journal of Cleaner Production*, 213, 1157–1164.
- Mete, S., Çil, Z. A., Özceylan, E., Ağpak, K., & Battalı, O. (2018). An optimisation support for the design of hybrid production lines including assembly and disassembly tasks. *International Journal of Production Research*, 56(24), 7375–7389.
- Özceylan, E., Kalayci, C. B., Güngör, A., & Gupta, S. M. (2019). Disassembly line balancing problem: A review of the state of the art and future directions. *International Journal of Production Research*, 57(15–16), 4805–4827.
- Pistolesi, F., Lazzerini, B., Dalle Mura, M., & Dini, G. (2017). EMOGA: A hybrid genetic algorithm with extremal optimization core for multi objective disassembly line balancing. *IEEE Transactions on Industrial Informatics*, 14(3), 1089–1098.
- Ren, Y., Yu, D., Zhang, C., Tian, G., Meng, L., & Zhou, X. (2017). An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. *International Journal of Production Research*, 55(24), 7302–7316.
- Ren, Y., Zhang, C., Zhao, F., Tian, G., Lin, W., Meng, L., & Li, H. (2018). Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-optimal algorithm. *Journal of Cleaner Production*, 174, 1475–1486.
- Riggs, R. J., Battalı, O., & Hu, S. J. (2015). Disassembly line balancing under high variety of end of life states using a joint precedence graph approach. *Journal of Manufacturing Systems*, 37(3), 638–648.
- Roshani, A., & Giglio, D. (2017). Simulated annealing algorithms for the multi-manned assembly line balancing problem: Minimising cycle time. *International Journal of Production Research*, 55(10), 2731–2751.
- Serin, F., Mete, S., & Çelik, E. (2019). An efficient algorithm for U-type assembly line re-balancing problem with stochastic task times. *Assembly Automation*, 39(4), 581–595.
- Silverman, F. N., & Carter, J. C. (1986). A cost-based methodology for stochastic line balancing with intermittent line stoppages. *Management Science*, 32(4), 455–463.
- Tuncel, E., Zeid, A., & Kamarthi, S. (2014). Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *Journal of Intelligent Manufacturing*, 25(4), 647–659.
- Urban, T. L., & Chiang, W. C. (2006). An optimal piecewise-linear program for the u-line balancing problem with stochastic task times. *European Journal of Operational Research*, 168, 771–782.
- Wang, K., Li, X., & Gao, L. (2019). Modelling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit. *Journal of Cleaner Production*, 211, 115–133.
- Xiao, S., Wang, Y., Yu, H., & Nie, S. (2017). An entropy-based adaptive hybrid particle swarm optimization for disassembly line balancing problems. *Entropy*, 19(11), 1–14.
- Yang, X. S. (2020). *Nature-inspired optimization algorithms*. Academic Press.
- Yin, T., Zhang, Z., & Jiang, J. (2021). A Pareto-discrete hummingbird algorithm for partial sequence-dependent disassembly line balancing problem considering tool requirements. *Journal of Manufacturing Systems*, 60, 406–428.

- Yin, T., Zhang, Z., Zhang, Y., Wu, T., & Liang, W. (2022). Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations. *Robotics and Computer-Integrated Manufacturing*, 73, 102251.
- Zhang, Y., Zhang, Z., Guan, C., & Xu, P. (2022). Improved whale optimisation algorithm for two-sided disassembly line balancing problems considering part characteristic indexes. *International Journal of Production Research*, 60(8), 2553–2571.
- Zhang, Z., Wang, K., Zhu, L., & Wang, Y. (2017). A pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem. *Expert Systems with Applications*, 86(1), 165–176.
- Zhu, X., Zhang, Z., & Hu, J. (2014). An ant colony optimization algorithm for multi-objective disassembly line balancing problem. *China Mechanical Engineering*, 25(8), 1075–1079.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.