J Intell Manuf (2019) 30:557–573
https://doi.org/10.1007/s10845-016-1262-6

A multiple-rule based constructive randomized search algorithm for solving assembly line worker assignment and balancing problem

Sebnem Demirkol Akyol[1] · **Adil Baykasoğlu**[1]

Received: 16 February 2016 / Accepted: 12 September 2016 / Published online: 6 October 2016

Abstract This paper proposes a constructive heuristic approach for the assembly line worker assignment and balancing problem (ALWABP). ALWABP arises when the operation time for every task differs according to the worker who executes the task. Since the operation times of tasks vary due to the workers, the problem requires a simultaneous solution to the double assignment problem. Tasks must be assigned to workers and workers to stations, concurrently. This problem is especially proposed in sheltered work centers for the disabled. However, it is not only important for the assembly lines with the disabled, but also for manually operated assembly lines with high labor turnover. In this paper, a multiple-rule based constructive randomized search (MRBCRS) algorithm is proposed in order to solve the ALWABP. Thirty nine task priority rules and four worker priority rules are defined. Performance of the proposed MRBCRS is compared with the relevant literature on benchmark data. Experimental results show that the proposed MRBCRS is very effective for benchmark problems. The results show that the algorithm improves upon the best-performing methods from the literature in terms of solution quality and time.

Keywords Assembly line balancing · Worker assignment · Heuristic approaches · Variable task times · Combinatorial optimization

✉ Adil Baykasoğlu
adil.baykasoglu@deu.edu.tr

[1] Department of Industrial Engineering, Dokuz Eylül University, Izmir, Turkey

Introduction

Assembly lines are a special case of flow-line production systems that manufacture standardized commodities in large amounts. In an assembly process, subassemblies and components are put together on the semi-finished assembly which moves from station to station where the parts are added in sequence until the final assembly is produced. The tasks can be executed by workers, robots or both of them. The assembly line concept has been present in various types of industries such as automobiles and other transportation vehicles, household appliances, electronic goods, computers, engines, and etc. In today's highly competitive market trends, companies need to meet the consumers' expectations in short time with minimum acceptable costs in order to be sustainable. This is why assembly line concept is so popular among all manufacturing systems. In such environment, an important decision problem, assembly line balancing problem (ALBP) arises.

ALBP is relevant for the allocation of the tasks, each having an operation processing time and a set of precedence relations, among workstations so that a given objective function is optimized and the precedence relations are satisfied. The most common problems used in the literature are Type-1 and Type-2 problems and each type of them belong to the NP-hard class of the combinatorial optimization problems (Karp 1972). Type-1 problems try to minimize the number of workstations hence; the cycle time must be predetermined. In industrial life this type of balancing problems is generally occurred when the designing phase of a new assembly line. On the other hand, Type-2 problems try to minimize the cycle time for a fixed number of workstations. So, this type of balancing problems is more appropriate when redesigning an existing line. Moreover, Type-E problems try to maximize the line efficiency by simultaneously

minimizing the cycle time and the number of workstations. Type-F problems seek for a feasible assembly line exists or not for predetermined cycle time and number of workstations. In classic ALBP, it is assumed that every task has a fixed operation time. However, in recent years the traditional ALBP has evolved and it is understood that the fixed operation time assumption is inconvenient for the real life manufacturing systems. Because every worker has unique characteristics such as skill, experience, ability, etc., especially in labor intensive assembly lines a task operation time differs depending on the worker who executes the task. In order to close this gap between the traditional ALBP and real life assembly line systems, a special case of ALBP which is called assembly line worker assignment and balancing problem (ALWABP) is introduced to the assembly line literature.

ALWABP arises when operation times of tasks vary due to the worker who executes the task, and some task-worker incompatibilities are occurred. Since task times are dependent on the worker, the concept of assigning tasks to workers is occurred in additional to the ALBP. In other words, ALWABP is a double assignment problem which includes assigning tasks to workers and workers to stations, simultaneously. Even the traditional ALBP is NP-hard; the ALWABP is also NP-hard. Since the ALWABP is a NP-hard hot research topic, many researchers have proposed various solution techniques to solve the problem. However, none of the proposed solution strategies is proven to be optimal for every benchmark test instance, and the problem is still attractive for researchers. The aim of this paper is to introduce an efficient solution approach for the ALWABP, which tries to minimize the cycle time of the line, in order to contribute satisfactory results to the literature.

The remainder of the paper is organized as follows. In "Literature review" section, the relevant literature is briefly reviewed. In "Formal problem definition and mathematical model" section, the ALWABP is defined and mathematical model of the problem is presented. Later in "A multiple-rule based constructive randomized search algorithm for solving ALWABP-2" section, the proposed constructive heuristic approach is introduced. In "Computational study" section, computational study and results of experiments are stated. Finally, "Conclusions" section concludes the paper and suggests some future work.

Literature review

Miralles et al. (2007) introduced the ALWABP for the first time to the assembly line literature. They proposed the "Sheltered Workcenters for Disabled" (SWD) concept and presented a case study in an assembly line that have a fixed number of workstations. They expressed that disabled workers can be changed with normal operators without any production loss by a logical assignment.

Chaves et al. (2007) applied a clustering search approach and tested it on the generated ALWABP-2 data. Since then, the proposed benchmark data sets, which are composed of four families (Roszieg, Heskia, Tonge and Wee-Mag), have been used in every ALWABP-2 research study. One year later, Miralles et al. (2008) developed a branch-and-bound algorithm for the ALWABP, enabling the solution of small-sized instances. They randomly generated data based on the Jackson problem from the SALBP (Hoffmann 1990) and solved the problem via branch and bound. Then, they integrated a heuristic approach to the branch and bound and applied the proposed method to an industrial case. Because of the problem complexity and the need to solve larger instances, the literature has since then shifted its efforts to heuristic methods.

Guo et al. (2008) proposed a novel flexible assembly line balancing problem with work sharing and workstation revisiting restrictions. They described a two-level solution procedure that uses both genetic algorithm and heuristic method. Solution procedure was applied to industrial data and experimental results showed the effectiveness of the proposed model. Costa and Miralles (2009) incorporates job rotation to the ALWABP. They integrated the training aspects associated with job rotation and ALWABP and offered a mixed integer linear model and a heuristic method in order to cope with to the proposed problem. The results showed that even in very complex contexts such as in SWD, it is possible to improve the welfare of workers by applying job rotation, without important losses in productivity.

Chaves et al. (2009) hybridized clustering search algorithm with the iterated local search in order to solve the ALWABP-2 benchmark data. They obtained good results in reasonable computation times. Moreira and Costa (2009) developed a minimalist tabu search algorithm that tries to be successful at simplicity, flexibility, accuracy and speed. Blum and Miralles (2011) proposed an iterated beam search algorithm for the ALWABP-2 and obtained the best results in the previous literature. Moreira et al. (2012) proposed a simple constructive heuristic approach that was based on 16 task priority rules and 3 worker priority rules and hybridized it with the genetic algorithm. They obtained the fastest results found so far. Zaman et al. (2012) proposed an operator assignment problem in an assembly line with fixed number of workstations and developed a genetic algorithm procedure to tackle with it. Araujo et al. (2012) introduced two new versions of the classic ALWABP which are parallelization and collaboration between different workers. In order to solve the proposed problem, the authors developed an integer programming model and hybridized it with a constructed heuristic algorithm. Mutlu et al. (2013) proposed an iterative genetic algorithm for the ALWABP-2 and obtained satisfactory solu-

tions in short CPU times. Borba and Ritt (2014) developed an interval probabilistic beam search and hybridized it with the branch and bound procedure. Except the Wee-Mag family, they almost achieved the best results. Vila and Pereira (2014) developed a branch and bound procedure with three different remember algorithms by a time constraint of 60, 600 s and with no time constraints. The time constraint of 600 s and no time constraint versions of their approaches acquired the best solutions in the relevant literature.

Sungur and Yavuz (2014) presented a new assembly line configuration that consists of workers whose qualification levels are ranked hierarchically. They proposed a model in which a lower qualified worker can be substituted by higher qualified ones with respect to a cost. They proposed an integer linear programming model in order to tackle with the problem and obtained good computational results. Polat et al. (2015) proposed a two-phase variable neighbourhood search algorithm for solving the ALWABP. They implemented the proposed method both on the benchmark data and a real life problem. The results showed the efficiency of the proposed procedure. Ritt et al. (2015) enhanced the ALWABP by adding stochastic worker availability. The authors presented a two-stage mixed integer program and local search heuristics for dealing with the proposed problem. As a result of computational experiments, they showed that stochastic modeling is a good idea for improving the line's efficiency and obtained good results for instances of small-sized problems.

Moreira et al. (2015) introduced the Assembly Line Worker Integration and Balancing Problem (ALWBP) which contains assembly lines with normal operators and disabled workers simultaneously. They introduced benchmark data and presented mathematical models and heuristic methodologies in order to solve the newly proposed problem. Guo et al. (2015) proposed a novel harmony search-based memetic optimization model for integrated production and transportation scheduling. They converted the proposed problem into an order assignment problem by using the heuristic approach. The experimental results proved the efficiency of the proposed solution method. More recently, Zacharia and Nearchou (2016) are presented a multi-objective evolutionary algorithm in order to solve the bi-criteria ALWABP. The results showed that the proposed solution procedure as a very satisfactory performance in terms of solution quality.

As it is mentioned above, ALWABP is a relatively new type of the traditional ALBP. It has been studied for almost a decade. There are many research attempts to cope with ALWABP, but none of the related work have obtained optimal solutions for all of the test instances. The aim of this study is to make a contribution to the relevant literature by proposing a new rule based solution procedure in order to obtain good results in short computational times.

Formal problem definition and mathematical model

In this section, we present a formal definition of the ALWABP-2. In analogy with the traditional ALBP; ALWABP also have the same types of problems such as; -1, -2, -E, and -F. In ALWABP-1 the number of workstations is tried to be minimized for a predetermined cycle time; and in ALWABP-2 the cycle time is tried to be minimized for a fixed number of workstations. In the relevant literature, single model of ALWABP-2 with paced line is the most common situation. The following notation is used in the remainder of this paper:

i, j: task index,
h: worker index,
s: workstation index,
N: set of tasks,
H: set of available workers,
W: set of workstations,
c: cycle time,
t_{hi}: processing time of i when worker h executes it,
IP_i: set of immediate predecessors of i,
x_{shi}: binary variable equal to 1 only if task i is assigned to worker h in station s,
y_{sh}: binary variable equal to 1 only when worker h is assigned to station s.

A mathematical model that was proposed by Miralles et al. (2008) can be written as follows:

$$\text{Min} \quad c \tag{1}$$

$$\text{subject to:} \quad \sum_{h\in H}\sum_{s\in W} x_{shi} = 1 \quad \forall i \in N, \tag{2}$$

$$\sum_{s\in W} y_{sh} \le 1 \quad \forall h \in H, \tag{3}$$

$$\sum_{h\in H} y_{sh} \le 1 \quad \forall s \in W, \tag{4}$$

$$\sum_{h\in H}\sum_{s\in W} s \cdot x_{shi} \le \sum_{h\in H}\sum_{s\in W} s \cdot x_{shj} \quad \forall i, j/i \in IP_j, \tag{5}$$

$$\sum_{i\in N} t_{hi} \cdot x_{shi} \le c \quad \forall h \in H; \forall s \in W, \tag{6}$$

$$\sum_{i\in N} x_{shi} \le M \cdot y_{sh} \quad \forall h \in H; \forall s \in W$$

$$\text{with} \quad y_{sh} \in [0,1] \quad \forall s \in W, h \in H$$

$$x_{shi} \in [0,1] \quad \forall s \in W, h \in H, i \in N$$

$$M > \sum_{h\in H}\sum_{i\in N} t_{hi} \tag{7}$$

The objective function given in (1) minimizes the cycle time. The constraint given in (2) ensures that every task i

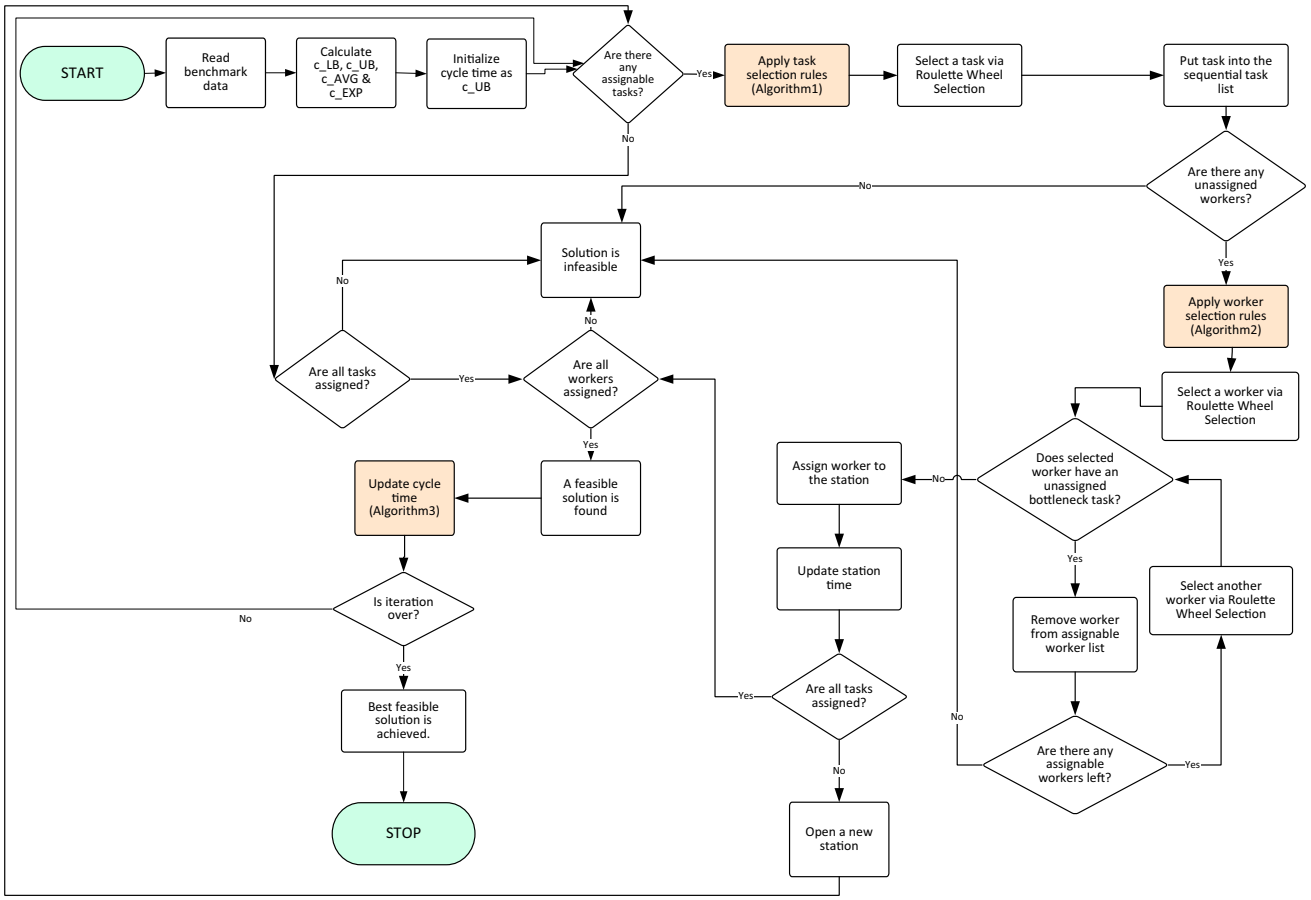

Fig. 1 Flowchart of the proposed RBCRS heuristic for the ALWABP-2

is assigned to a single station s and worker h. Constraints sets given in (3) and (4) expresses that every worker can be assigned to only one station; and in every station there is only one worker, respectively. The set of constraints given in (5) states the precedence relations between tasks i and j, where i is predecessor of j. Constraints sets given in (6) and (7) ensures that every worker h assigned to station s can have more than one task, whenever given cycle time c is not achieved. As cycle time c and y_{sh} are both variables, (6) and (7) are defined separately in order to maintain the model linearity.

A multiple-rule based constructive randomized search algorithm for solving ALWABP-2

In this study a multiple-rule based constructive randomized search (MRBCRS) heuristic is developed in order to solve the ALWABP-2. The heuristic prioritizes tasks and sequentially assigns tasks to the current station in order and then assigns the best suitable worker to that station. This is called station-oriented assignment procedure in the relevant literature (Scholl and Voß 1996). The flowchart of the proposed algorithm is given in Fig. 1.

Initially, benchmark data is read and by using the data cycle time values (lower bound, upper bound, average and expected) are calculated. Assignable tasks are determined according to the precedence relations and then rule based search algorithm, which will be explained in detail later, is applied. After prioritizing tasks, a task is chosen randomly via roulette wheel selection. All assignable tasks are chosen one by one and are put in order sequentially. Then, worker selection rules are applied to the unassigned workers and one of them is selected.

After determining the worker, it must be checked that, is there any task which can be operated by only the selected worker, which is called *bottleneck task* in this work. If such a situation exists, it must be controlled that if the task is executed on the current station or not. If not, then the solution is infeasible, because none of the remaining workers can operate the bottleneck task. If this is not the case then, the selected worker and tasks, which are a subset of the sequential task list, are assigned to the current station. While choosing tasks from the sequential list, two important points must be considered:

- If there is a task in the list that cannot be executed by the current worker, eliminate that one and go on from the sequential task list.
- Sum of operation times of the assigned tasks, namely *station time*, cannot exceed the cycle time.

When the feasible solution is obtained, then the current cycle time is reduced and the whole process is applied again. By this way, at every feasible solution cycle time is diminished and finally one feasible solution with the minimum cycle time is gathered when maximum iteration number is achieved.

Rule based task and worker selection strategy

In this study, in order to select one task/worker among tasks/workers in an intelligent manner, we apply 39 rules to the tasks and 4 to the workers. Most of the following rules are collected from Baykasoğlu (2006) and Moreira et al. (2012). Although we will use the notation given above, there are some additional ones in order to express the proposed algorithm:

IS_i: set of immediate successors of i,
S_i: set of all successors of i,
P_i: set of all predecessors of i,
UB_i: upper bound on the station to which i may be assigned, $UB_i = N+1-\left[\left(t_{hi}+\sum_{j\in S_i} t_{hj}\right)/c\right]^+$,
LB_i: lower bound on the station to which i may be assigned, $LB_i = \left[\left(t_{hi}+\sum_{j\in P_i} t_{hj}\right)/c\right]^+$.

Some rules are based on the cycle time. Because of the varying operation times, we calculate the expected cycle time (c_{exp}) and use it instead of cycle time in the relevant rules. The upper bound (c_{UB}), lower bound (c_{LB}), average (c_{avg}) and expected (c_{exp}) values of cycle time are calculated as follows:

$$c_{UB} = \max\left\{\sum \frac{\max\{t_{hi}\}}{W}, \max\{t_{hi}\}\right\} \quad \forall h \in H, \forall i \in N \tag{8}$$

$$c_{LB} = \max\left\{\sum \frac{\min\{t_{hi}\}}{W}, \min\{t_{hi}\}\right\} \quad \forall h \in H, \forall i \in N \tag{9}$$

$$c_{avg} = \sum (avg\{t_{hi}\})/W \quad \forall h \in H \tag{10}$$

$$c_{\exp} = \frac{1}{6}\left(c_{UB} + 4 * c_{avg} + c_{LB}\right) \tag{11}$$

where $\max\{t_{hi}\}$, $\min\{t_{hi}\}$, $\text{avg}\{t_{hi}\}$ are the maximum, minimum and average values of the operation time of task i executed by worker h, respectively.

Task selection rules

Most of the priority rules are based on the task execution times, which is a parameter that depends on the worker assigned to each workstation in our problem. So, to overcome this complication we consider a task's minimum, average and maximum operation times ($t_i^{\min}, t_i^{avg}, t_i^{\max}$). Task priority rules used in the proposed approach are listed in the following:

(1) Maximum Longest Processing Time ($\text{LPT}_{\max}$), $t_i^{\max}$;
(2) Minimum Longest Processing Time ($\text{LPT}_{\min}$), $t_i^{\min}$;
(3) Average Longest Processing Time (LPT_{avg}), t_i^{avg};
(4) Maximum Shortest Processing Time ($\text{SPT}_{\max}$), $t_i^{\max}$;
(5) Minimum Shortest Processing Time ($\text{SPT}_{\min}$), $t_i^{\min}$;
(6) Average Shortest Processing Time (SPT_{avg}), t_i^{avg};
(7) Greatest Number of Immediate Successors (GNIS), $|IS_i|$;
(8) Greatest Number of Successors (GNS), $|S_i|$;
(9) Greatest Number of Immediate Predecessors (GNIP), $|IP_i|$;
(10) Greatest Number of Predecessors (GNIP), $|P_i|$;
(11) Random priority (R);
(12) Smallest Task Number (STN), i;
(13) Maximum Greatest Ranked Positional Weight (GRPW), $t_i^{\max} + \sum_{i\in S_i}^{i} t_j^{\max}$;
(14) Minimum Greatest Ranked Positional Weight (GRPW), $t_i^{\min} + \sum_{i\in S_i}^{i} t_j^{\min}$;
(15) Average Greatest Ranked Positional Weight (GRPW), $t_i^{avg} + \sum_{i\in S_i}^{i} t_j^{avg}$;
(16) Maximum Greatest Average Ranked Positional Weight (GARPW), $\left(t_i^{\max} + \sum_{j\in S_i}^{t_j^{\max}}\right)/(|S_i|+1)$;
(17) Minimum Greatest Average Ranked Positional Weight (GARPW), $\left(t_i^{\min} + \sum_{j\in S_i}^{t_j^{\min}}\right)/(|S_i|+1)$;
(18) Average Greatest Average Ranked Positional Weight (GARPW), $\left(t_i^{avg} + \sum_{j\in S_i}^{t_j^{avg}}\right)/(|S_i|+1)$;
(19) Maximum Smallest Upper Bound (SUB), $N+1-\left[\left(t_i^{\max}+\sum_{j\in S_i} t_j^{\max}\right)/c_{\exp}\right]^+$;
(20) Minimum Smallest Upper Bound (SUB), $N+1-\left[\left(t_i^{\min}+\sum_{j\in S_i} t_j^{\min}\right)/c_{\exp}\right]^+$;
(21) Average Smallest Upper Bound (SUB), $N+1-\left[\left(t_i^{avg}+\sum_{j\in S_i} t_j^{avg}\right)/c_{\exp}\right]^+$;
(22) Maximum Smallest Upper Bound Divided by the Number of Successors, (S_UB_NS), $UB_i^{\max}/(|S_i|+1)$;
(23) Minimum Smallest Upper Bound Divided by the Number of Successors, (S_UB_NS), $UB_i^{\min}/(|S_i|+1)$;
(24) Average Smallest Upper Bound Divided by the Number of Successors, (S_UB_NS), $UB_i^{avg}/(|S_i|+1)$;
(25) Maximum Greatest Processing Time Divided by the Upper Bound, (G_PT_UB), $t_i^{\max}/UB_i^{\max}$;

Algorithm 1. Pseudo code of the proposed rule-based adaptive task search algorithm

```
procedure: Rule based adaptive search method
input: assignable tasks, assignable workers
output: task to assign
BEGIN
    FOR each rule
        apply rule to assignable tasks;
        add points to the appropriate task(s);
    ENDFOR
    create a task probability list;
    FOR each task in assignable tasks
        give probability to task directly proportional to task points;
        add task probability to task probability list;
    ENDFOR
    Select randomly one task from the task probability list;
END
```

(26) Minimum Greatest Processing Time Divided by the Upper Bound (G_PT_UB), $t_i^{\min}/UB_i^{\min}$;
(27) Average Greatest Processing Time Divided by the Upper Bound (G_PT_UB), t_i^{avg}/UB_i^{avg};
(28) Maximum Smallest Lower Bound (SLB), $\left[\left(t_i^{\max}+\sum_{j\in P_i} t_j^{\max}\right)/c_{\exp}\right]^+$;
(29) Minimum Smallest Lower Bound (SLB), $\left[\left(t_i^{\min}+\sum_{j\in P_i} t_j^{\min}\right)/c_{\exp}\right]^+$;
(30) Average Smallest Lower Bound (SLB), $\left[\left(t_i^{avg}+\sum_{j\in P_i} t_j^{avg}\right)/c_{\exp}\right]^+$;
(31) Maximum Smallest Slack (SSLK), $UB_i^{\max}-LB_i^{\max}$;
(32) Minimum Smallest Slack (SSLK), $UB_i^{\min}-LB_i^{\min}$;
(33) Average Smallest Slack (SSLK), $UB_i^{avg}-LB_i^{avg}$;
(34) Maximum Smallest Number of Successors Divided by Task Slack, (S_NS_SLK), $S_i^{/}\left(UB_i^{\max}-LB_i^{\max}\right)$;
(35) Minimum Smallest Number of Successors Divided by Task Slack, (S_NS_SLK), $S_i^{/}\left(UB_i^{\min}-LB_i^{\min}\right)$;
(36) Average Smallest Number of Successors Divided by Task Slack, (S_NS_SLK), $S_i^{/}\left(UB_i^{avg}-LB_i^{avg}\right)$;
(37) Maximum Greatest Task Time Divided by Task Slack (TT_SLK), $t_i^{\max}/\left(UB_i^{\max}-LB_i^{\max}\right)$;
(38) Minimum Greatest Task Time Divided by Task Slack (TT_SLK), $t_i^{\min}/\left(UB_i^{\min}-LB_i^{\min}\right)$;
(39) Average Greatest Task Time Divided by Task Slack (TT_SLK), $t_i^{avg}/\left(UB_i^{avg}-LB_i^{avg}\right)$. Maximum Longest Processing Time (LPT_{max}), $t_i^{\max}$.

Worker selection rules

Four worker selection rules are applied in this study:

(1) Greatest Number of Tasks that can be Executed (GNTE), $\sum N_i$;
(2) Greatest Number of Tasks that can be Executed in Minimum Time (GNTEMT), $\sum t_i / \sum N_i$;
(3) Maximum Utilization (MU);
(4) Random priority (R).

Assignable tasks (workers) are prioritized by using the above rules. Then, a task (worker) is randomly selected within assignable tasks (workers) such that the task (worker) with the higher priority takes the higher selection probability. This strategy is called *roulette wheel selection* in the literature. The pseudo codes of the proposed rule-based adaptive task search and worker selection mechanisms are illustrated in Algorithms 1 and 2, respectively.

Illustrative example

In order to explain the proposed MRBCRS approach more clearly, a simple numeric example is given in this section. Data used in the illustrative example is collected from a harness production company which manufactures cable networks for automotive sector. The harness production company has two main production areas; cable cutting area and final assembly area. This simple example is gathered from the final assembly area of the most basic product of the company. The assembly line is composed of 3 workstations, 3 workers and 8 tasks. The task operation times per worker and precedence relations are given in Table 1 and Fig. 2, respectively. "X" in Table 1 represents that there are some task-worker incompatibles (i.e. worker 1 and tasks 2 and 4).

Cycle time values are found by Eqs. (8)–(11). $c_{UB} = \max\{64, 42\} = 64$; $c_{LB} = \max\{38, 22\} = 38$; $c_{avg} = 51.3$; $c_{\exp} = 51.2$. For the first iteration we set cycle time equal to thec_{UB}. In the beginning there are 2 assignable tasks according to the precedence relations {1, 5}. We apply task

Algorithm 2. Pseudo code of the proposed rule-based worker selection algorithm

```
procedure: Rule based worker selection
input: unassigned workers, sequential task list
output: selected worker
BEGIN
   FOR each rule
        apply rule to unassigned workers;
        add points to the appropriate worker(s);
   ENDFOR
   create a worker probability list;
   FOR each worker in unassigned workers
       give probability to worker directly proportional to worker points;
       add worker probability to worker probability list;
   ENDFOR
   Select randomly one task from the worker probability list;
END
```

Table 1 Task execution times per worker

Tasks	Workers		
	1	2	3
1	8	6	10
2	X	20	22
3	10	X	30
4	X	20	15
5	8	6	X
6	22	32	42
7	25	20	30
8	30	25	15

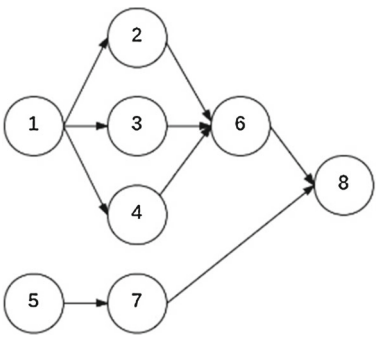

Fig. 2 Precedence relations of the numeric example

Table 2 The prioritized values of the candidate tasks

	Tasks			Tasks	
Rules	1	5	Rules	1	5
1	**10**	8	21	**6**	8
2	**6**	**6**	22	**0.83**	2.67
3	**8**	7	23	**1.17**	2.67
4	10	**8**	24	**1**	2.67
5	**6**	**6**	25	**2**	1
6	8	**7**	26	**0.86**	0.75
7	**3**	1	27	**1.33**	1.17
8	**5**	2	28	**1**	**1**
9	**0**	**0**	29	**1**	**1**
10	**0**	**0**	30	**1**	**1**
11	*		31	**4**	7
12	**1**	5	32	**6**	7
13	**154**	38	33	**5**	7
14	**88**	26	34	1.25	**0.29**
15	**121.83**	32	35	0.71	**0.29**
16	**25.67**	12.67	36	0.83	**0.29**
17	**20.31**	10.67	37	**2.5**	1.14
18	**14.67**	8.67	38	**0.86**	0.86
19	**5**	8	39	**1.33**	1
20	**7**	8			

priority rules to the candidate tasks. The prioritized values of the two candidates are listed in Table 2.

The bold parts are the superior ones for the corresponding rule. Among the all 39 rules, Task 1 overperforms for the 34 rules and Task 5 overperforms for 11 rules. So, there will be a selection with the probabilities: $p_1 = 34/(34 + 11) = 0.76$; $p_5 = 11/(34 + 11) = 0.24$. We pick a random number between [0, 1] that is 0.65 which corresponds to the Task 1. We put Task 1 to the *sequential task list* as the first element. All workers can execute Task 1 and the operation times of the Workers 1, 2 and 3 are 8, 6 and 10, respectively. Now, Tasks 2, 3, 4 and 5 are assignable. We apply the priority rules again for the new tasks adaptively and then select one of them randomly by considering the probabilities. Let the newly selected task be the Task 5. Then, the sequential task list contains 1 and 5. Note that Worker 3 cannot execute Task 5. So, at the end of the iteration if Worker 3 is assigned to the current station (station 1), Task 5 should be eliminated

Table 3 Task-worker assignment for the first station

Tasks	1	5	3	7	2	4	6	8
Station time (worker 1)	8	16	26	51	X	X	-	–
Station time (worker 2)	6	12	X	32	52	**72**	–	–
Station time (worker 3)	10	X	40	-	62	**77**	**119**	–
Station	1	1	1	1	1	1	1	1

Table 4 The prioritized values of the candidate workers

Rules	Workers		
	1	2	3
1	6	**7**	**7**
2	17.17	18.43	**23.43**
3	0.8	0.81	**0.97**
4	*		

and not assigned to the station. Through the iteration, we continue to select tasks according to the rule based method and update the station time. At the end, we obtain the results as shown in Table 3.

Table 3 represents task and worker assignments to the first station. First row shows the sequence of tasks (sequential task list). Rows 2, 3 and 4 illustrates the station times if Worker 1, 2 or 3 are selected, respectively. The sign 'X' represents that the current worker cannot execute the corresponding task and the '–' sign means that the current worker cannot operate relevant task because of the precedence relations. Moreover, bold numbers illustrate that the cycle time is exceeded. For example, row 4 of Table 3 represents that if Worker 3 is selected for station 1, according to the sequential task list Task 1 must be assigned firstly. Then, task 5 cannot be assigned because Worker 3 is incapable of executing it ("X"). The next task which is Task 3 should be assigned from the sequential task list. Now, the next task in the sequential task list is Task 7. However, it cannot be selected because of the precedence relations. Since Task 7 is an immediate successor of Task 5 and Task 5 is not assigned yet, Task 7 could not be executed ("-"). Then, Task 2 must be assigned according to the list and the station time becomes 62 s. Since the cycle time is 64 s, there are not any assignable tasks in the sequential task list and station must be closed.

Then, in order to select a worker we apply priority rules to the unassigned workers (see Table 4). According to the Table 4 Workers 1 and 2 are superior for one rule and Worker 3 is superior for three rules. So, the probabilities of selection of the workers are 0.2, 0.2 and 0.6 for Workers 1, 2 and 3, respectively. Let the random number be 0.11, then Worker 1 is selected for the first station. Tasks 1, 5, 3 and 7 and Worker 1 are assigned to the station 1, for the first iteration. After assigning the first station, the whole process is repeated for the remaining stations. At the end of the first iteration an initial solution is obtained. Table 5 illustrates the solution obtained by the end of the first iteration.

Table 5 Initial solution of the numeric example

Task	1	5	3	7	2	4	6	8
Worker	1	1	1	1	3	3	2	2
Station	1	1	1	1	2	2	3	3
Operation time	8	8	10	25	22	15	32	25
Station time	8	16	26	**51**	22	**37**	32	**57**

Table 5 indicates that Tasks 1, 5, 3 and 7 and Worker 1 are assigned to Station 1; Tasks 2 and 4 and Worker 3 are assigned to Station 2 and lastly, Tasks 6 and 8 and Worker 2 are assigned to Station 3. Station times of the Stations 1, 2 and 3 are 51, 37 and 57 s, respectively. Next, the cycle time has to be decreased and a new solution will be gathered by using new cycle time. This process will be continued until the maximum iteration number, which is 1000 in our case, is achieved.

Algorithm 3. Pseudo code of the cycle time update algorithm

```
procedure: Cycle time update method
input: cycle time, c_UB
output: new cycle time
BEGIN
IF (cycle time / c_UB) > 0.75 THEN
        new cycle time = cycle time * 0.75
ELSE IF (currentCycleTime / c_UB) > 0.6 THEN
        new cycle time = cycle time * 0.85
ELSE
        new cycle time = cycle time * 0.99
END IF
END
```

Table 6 Test instance characteristics

Family	Number of tasks (N)	Number of workers (W)	Order strength (OS)
Roszieg	25 (low)	4 (groups 1–4) or 6 (groups 5–8)	71.67 (high)
Heskia	28 (low)	4 (groups 1–4) or 7 (groups 5–8)	22.49 (low)
Tonge	70 (high)	10 (groups 1–4) or 17 (groups 5–8)	59.42 (high)
Wee-Mag	75 (high)	11 (groups 1–4) or 19 (groups 5–8)	22.67 (low)

Table 7 Test groups characteristics

Factor	Low	High
Relation between tasks and operators	1, 2, 3, 4	5, 6, 7, 8
Variability of operation times	1, 2, 5, 6	3, 4, 7, 8
Percentage of task-worker incompatibilities	1, 3, 5, 7	2, 4, 6, 8

Cycle time update

The algorithm whose pseudo code is illustrated in Algorithm 3 tries to find the minimum cycle time in the feasible search space. Inertia weight is used in order to decrease cycle time efficiently. In the relevant literature in order to obtain fast convergence, several cycle time update mechanism are applied (Simaria and Pedro 2004; Akyol and Bayhan 2011; Bansal et al. 2011; Mutlu et al. 2013). In this study, the ratio of the current cycle time to the upper bound of the cycle time is used. If this ratio is greater than 75 %, it means that the current cycle time is far away from the minimum cycle time in the feasible search space and it should be decreased drastically. If the ratio is between 75–60 %, then the current cycle time is nearer to the minimum cycle time and the moves should be made more slightly when finding the new cycle time. Lastly, if the ratio is smaller than 0.60, the new cycle time should be obtained by decreasing the current cycle time in very small steps in order not to miss the minimum cycle time.

Computational study

In this section, a computational study of the proposed MRBCRS heuristic is presented. The proposed heuristic is implemented in Microsoft Visual Studio Premium 2012 C# version 11.0.1. The tests were run on a Core 2 Duo i7 2.2 GHz processor and 6 GB main memory running the Windows 7 operating system.

Benchmark data is generated from the corresponding SALBP benchmark data set (Chaves et al. 2007). Test instances are derived by using five experimental factors at a low and a high level. These factors are the number of tasks, the number of workers, the order strength (OS), the variability of the task execution time, and the number of infeasible task-worker pairs (OS = the number of precedence relations/ $[N*(N-1)]$). The details of these characteristics are given in Tables 6 and 7. There are 320 test instances which are grouped into four families: Heskia, Roszieg, Tonge and Wee-Mag. Each one of the families contains 80 instances. There are 32 task groups that each of them contains 10 test instances. Each task group is defined by the family name and a number between 1 and 8.

Because the proposed algorithm is a heuristic method, in order to obtain accurate results we run every single instance 10 times (10 replications) for 1000 iterations. It means that we run our program for 3200 times and every run has a maximum iteration number of 1000. Table 8 reports the detailed results of the comparison of our proposed algorithm to the relevant literature.

The first column represents the benchmark data family, while the second one represents the group number of that family. As it is stated previously, each family group consists of 10 instances, so the table shows the average results of 10 instances for each group of each family. The third column reports the optimum results found so far in the relevant literature under no running time constraints. The rest of the columns represent the referred papers and their solution techniques and results. CPU times of the solution procedures are reported in Table 9. The CPU times of Vila and Pereira (2014) for the Roszieg and Heskia families are not given, because they are not reported in the corresponding paper. Authors claimed that their proposed algorithm solves every instance from the Heskia and Roszieg groups in less than one second. According to Tables 8 and 9, the results show that our proposed MRBCRS algorithm is superior to the all other techniques in general, but a detailed examination may indicate the following findings:

- We obtain the best results for the 75 % of the 320 test instances for the ALWABP-2.
- For the Roszieg family, we report the best feasible solutions for all of the test instances. None of the referred papers have found results close to ours yet.
- For the Heskia family, we obtain the very best results found so far for 6 of the 8 test groups (1, 2, 4, 5, 6 and 7).

Table 8 Comparison of the proposed MRBCRS approach to the relevant literature

Family	Group	Best so far	Demirkol Akyol &Baykasoğlu		Chaves et al. (2009)				Moreira and Costa (2009)		Blum and Miralles (2011)	
			MRBCRS		ILS		CS		TS		IBS	
			Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
Roszieg	1	20.1	**16.1**	**16.1**	20.1	20.4	20.1	20.2	20.1	20.1	20.1	20.1
	2	31.5	**16.8**	**16.8**	32.4	38.9	31.5	32.5	31.5	31.5	31.5	31.5
	3	28.1	**24.5**	**24.5**	28.2	28.7	28.1	28.5	28.1	28.1	28.1	28.1
	4	28.0	**25.5**	**25.5**	28.0	28.1	28.0	28.0	28.0	28.0	28.0	28.0
	5	9.7	**8.2**	**8.2**	10.3	11.8	9.7	10.7	9.7	9.7	9.7	9.7
	6	11.0	**9.3**	**9.3**	11.5	14.3	11.0	12.1	11.0	11.0	11.0	11.0
	7	16.0	**13.4**	**13.4**	16.5	18.7	16.0	16.9	16.0	16.0	16.0	16.0
	8	15.1	**12.7**	**12.7**	15.3	17.7	15.1	15.6	15.1	15.2	15.1	15.1
	Avg.	19.9	**15.8**	**15.8**	20.3	22.3	19.9	20.6	19.9	20.0	19.9	19.9
Heskiea	1	102.3	**99.4**	**99.4**	102.3	103.0	102.3	102.8	102.3	102.8	102.3	102.3
	2	122.6	**111.9**	**111.9**	122.7	124.2	122.6	123.8	122.6	122.8	122.6	122.6
	3	172.5	**172.5**	**172.5**	172.6	176.4	**172.5**	**175.5**	**172.5**	172.6	**172.5**	**172.5**
	4	171.2	**160.7**	**160.7**	171.3	171.8	171.2	171.7	171.2	171.8	171.2	171.2
	5	34.9	**34.5**	**34.5**	35.3	38.6	34.9	37.8	35.0	36.8	34.9	34.9
	6	42.6	**39.7**	**39.7**	43.6	45.7	42.6	44.7	43.1	44.6	42.6	42.6
	7	75.2	**74.2**	**74.2**	76.7	78.6	75.2	77.7	75.2	76.6	75.2	75.2
	8	67.2	**67.2**	**67.2**	68.1	72.4	67.2	70.7	67.3	68.4	**67.2**	**67.2**
	Avg.	98.6	**95.0**	**95.0**	99.1	101.3	98.6	100.6	98.7	99.6	98.6	98.6

Table 8 continued

Family	Group	Best so far	Demirkol Akyol &Baykasoğlu		Chaves et al. 2009				Moreira and Costa (2009)		Blum and Miralles (2011)	
			MRBCRS		ILS		CS		TS		IBS	
			Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
Tonge	1	90.6	**90.6**	**90.6**	120.0	135.3	96.7	116.6	100.1	108.3	94.9	96.7
	2	106.7	**106.7**	**106.7**	151.8	174.8	116.0	141.8	117.7	128.0	110.2	111.5
	3	159.3	**159.3**	**159.3**	214.6	236.5	167.7	199.4	171.5	187.4	165.0	168.0
	4	163.9	**160.7**	**160.7**	220.7	244.3	174.0	206.0	178.3	194.1	170.0	171.4
	5	31.6	**31.6**	**31.6**	64.2	71.1	41.3	51.3	41.7	47.6	33.1	34.2
	6	36.9	**36.9**	**36.9**	74.2	87.4	48.5	61.6	47.7	57.6	40.0	41.0
	7	63.2	**63.2**	**63.2**	113.7	129.3	77.8	93.0	75.5	82.1	66.4	67.9
	8	61.2	**61.2**	**61.2**	116.1	131.3	77.9	95.6	75.8	86.1	64.7	66.6
	Avg.	89.2	**88.8**	**88.8**	134.4	151.3	100.0	120.7	101.0	111.4	93.0	94.7
Wee-Mag	1	26.1	43.0	43.0	31.2	35.5	29.0	32.7	35.3	38.4	28.7	29.7
	2	31.2	49.8	49.8	37.4	41.0	34.6	38.4	41.1	45.3	33.6	34.9
	3	45.8	78.6	78.6	54.3	61.3	50.8	56.7	58.4	64.8	50.1	51.6
	4	44.3	77.9	77.9	52.7	58.9	49.6	55.6	56.1	61.3	48.6	50.1
	5	9.6	15.0	15.0	16.7	45.4	13.1	20.9	19.9	23.8	10.3	10.7
	6	11.2	19.4	19.4	18.7	23.7	14.6	18.2	23.3	28.6	11.9	12.4
	7	17.1	26.9	26.9	25.1	31.1	21.2	27.1	32.6	39.2	18.2	19.0
	8	17.2	27.0	27.0	24.9	33.7	21.6	26.8	31.7	39.1	18.1	18.9
	Avg.	25.3	42.2	42.2	32.6	41.3	29.3	34.6	37.3	42.6	27.4	28.4

Table 8 continued

Family	Group	Best so far	Moreira et al. (2012)		Mutlu et al. (2013)		Borba and Ritt (2014)			Vila and Pereira (2014)		
			HGA		IGA		IPBS		BB	BB&R 60s	BB&R 600s	BB&R no time limit
			Best	Avg.	Best	Avg.	Best	Avg.	Best	Best	Best	Best
Roszieg	1	20.1	20.1	20.1	20.1	20.1	20.1	20.1	20.1	20.1	20.1	20.1
	2	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5	31.5
	3	28.1	28.1	28.1	28.1	28.1	28.1	28.1	28.1	28.1	28.1	28.1
	4	28.0	28.0	28.0	28.0	28.0	28.0	28.0	28.0	28.0	28.0	28.0
	5	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7
	6	11.0	11.1	11.1	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0
	7	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0
	8	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1	15.1
	Avg.	19.9	20.0	20.0	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9
Heskiea	1	102.3	102.3	102.3	102.3	102.3	102.3	102.3	102.3	102.3	102.3	102.3
	2	122.6	122.7	122.7	122.6	122.6	122.6	122.6	122.6	122.6	122.6	122.6
	3	172.5	**172.5**	**172.5**	**172.5**	**172.5**	**172.5**	**172.5**	**172.5**	**172.5**	**172.5**	**172.5**
	4	171.2	171.2	171.7	171.2	171.3	171.2	171.3	171.2	171.2	171.2	171.2
	5	34.9	34.9	35.1	34.9	34.9	34.9	34.9	34.9	34.9	34.9	34.9
	6	42.6	42.6	42.8	42.6	42.6	42.7	42.8	42.7	42.7	42.7	42.7
	7	75.2	75.2	75.4	75.2	75.2	75.2	75.2	75.2	75.2	75.2	75.2
	8	67.2	**67.2**	**67.6**	**67.2**	**67.2**	**67.2**	**67.2**	**67.2**	**67.2**	**67.2**	**67.2**
	Avg.	98.6	98.6	98.8	98.6	98.6	98.6	98.6	98.6	98.6	98.6	98.6

Table 8 continued

Family	Group	Best so far	Moreira et al. (2012)		Mutlu et al. (2013)		Borba and Ritt (2014)			Vila and Pereira (2014)		
			HGA		IGA		IPBS		BB	BB&R 60s	BB&R 600s	BB&R no time limit
			Best	Avg.	Best	Avg.	Best	Avg.	Best	Best	Best	Best
Tonge	1	90.6	92.8	95.9	93.0	94.1	91.3	92.2	**90.6**	**90.6**	**90.6**	**90.6**
	2	106.7	109.3	111.2	109.3	110.2	107.8	109.1	**106.7**	**106.7**	**106.7**	**106.7**
	3	159.3	162.2	166.3	162.4	165.2	160.8	161.7	**159.3**	160.3	**159.3**	**159.3**
	4	163.9	168.4	171.0	168.4	170.1	165.9	167.5	163.9	165.9	163.9	163.9
	5	31.6	34.1	35.2	33.1	33.1	32.2	32.5	**31.6**	31.8	31.7	31.7
	6	36.9	40.2	42.0	40.1	40.4	38.9	39.4	**36.9**	37.1	**36.9**	**36.9**
	7	63.2	66.6	69.6	66.4	66.4	64.5	64.9	63.4	63.9	**63.2**	**63.2**
	8	61.2	65.8	67.5	64.6	64.8	63.1	63.9	**61.2**	62.0	**61.2**	**61.2**
	Avg.	89.2	92.4	94.8	92.2	93.0	90.6	91.4	89.2	89.8	89.2	89.2
Wee-Mag	1	26.1	26.7	27.8	26.7	27.4	27.1	27.6	26.8	26.6	**26.1**	**26.1**
	2	31.2	32.2	33.5	32.3	32.7	32.1	32.6	32.2	31.7	**31.2**	**31.2**
	3	45.8	47.6	49.9	47.6	48.2	47.5	48.4	47.5	46.6	46.1	45.8
	4	44.3	45.6	47.8	45.8	46.0	45.4	46.2	45.2	44.8	**44.3**	**44.3**
	5	9.6	10.5	11.1	10.3	10.4	9.9	10.0	9.9	9.8	**9.6**	**9.6**
	6	11.2	12.3	12.9	12.1	12.1	11.4	11.6	11.4	**11.2**	**11.2**	**11.2**
	7	17.1	18.6	19.7	18.2	18.5	17.7	17.9	17.6	17.4	17.2	17.2
	8	17.2	18.4	19.3	18.0	18.4	17.7	17.7	17.6	17.4	**17.2**	**17.2**
	Avg.	25.3	26.5	27.8	26.4	26.7	26.1	26.5	26.0	25.7	25.4	25.3

ILS iterated local search, *CS* clustering search, *TS* tabu search, *IBS* iterated beam search, *HGA* hybrid genetic algorithm, *IGA* iterative genetic algorithm, *IPBS* interval probabilistic beam search, *BB* branch and bound, *BB&R* ranch and bound and remember algorithm

Table 9 Comparison of the CPU times

Family	Group	Demirkol Akyol &Baykasoğlu	Chaves et al. (2009)		Moreira and Costa (2009)	Blum and Miralles (2011)	Moreira et al. (2012)	Mutlu et al. (2013)	Borba and Ritt (2014)		Vila and Pereira (2014)
		RBCRS	ILS	CS	TS	IBS	HGA	IGA	IPBS	BB	BB&R
Roszieg	1	0.0	3.4	0.8	0.2	0.0	0.0	0.3	0.0	0.1	–
	2	0.0	3.2	0.9	0.4	0.1	4.5	0.3	0.0	0.1	–
	3	0.0	3.3	0.6	0.2	0.1	4.0	0.3	0.0	0.2	–
	4	0.0	3.3	0.2	0.0	0.0	3.4	0.2	0.0	0.1	–
	5	0.0	4.6	1.3	0.6	0.0	3.6	0.5	0.0	0.4	–
	6	0.0	4.5	1.4	0.6	0.0	4.0	0.5	0.1	0.3	–
	7	0.0	4.6	1.5	0.2	0.0	4.5	0.5	0.0	0.5	–
	8	0.0	4.6	1.9	0.7	0.0	4.5	0.5	0.0	0.4	–
Heskiea	1	0.0	6.3	1.3	0.6	8.2	6.9	1.7	0.1	0.2	–
	2	0.0	6.3	1.4	0.4	3.0	9.3	0.9	0.1	0.2	–
	3	0.0	6.4	1.7	0.8	5.7	9.2	1.7	0.1	0.2	–
	4	0.0	6.4	1.4	0.5	5.2	9.5	1.4	0.2	0.2	–
	5	0.1	7.5	4.4	1.7	1.1	8.0	0.9	0.2	1.2	–
	6	0.1	7.5	3.4	0.6	2.5	7.4	1.1	0.2	1.4	–
	7	0.2	7.5	2.9	0.4	1.7	6.6	0.9	0.1	1.2	–
	8	0.1	7.5	3.6	2.7	2.5	9.2	1.4	0.2	1.4	–

Table 9 continued

Family	Group	Demirkol Akyol & Baykasoğlu	Chaves et al. (2009)		Moreira and Costa (2009)	Blum and Miralles (2011)	Moreira et al. (2012)	Mutlu et al. (2013)	Borba and Ritt (2014)		Vila and Pereira (2014)
		RBCRS	ILS	CS	TS	IBS	HGA	IGA	IPBS	BB	BB&R
Tonge	1	5.3	102.5	64.0	21.3	86.4	205.7	47.4	19.7	165.4	266.9
	2	3.3	101.7	64.6	26.4	92.2	241.2	40.5	14.8	134.2	84.0
	3	2.1	102.6	66.2	28.9	150.3	391.0	70.8	26.8	362.1	278.8
	4	5.6	102.6	65.7	17.7	149.5	347.6	59.4	21.4	233.9	238.7
	5	15.3	151.1	101.1	27.5	88.0	296.9	78.0	26.2	789.9	443.5
	6	16.0	151.4	105.3	32.1	70.5	300.0	68.4	19.4	822.5	405.5
	7	18.7	151.4	100.1	32.3	124.3	446.7	68.1	31.8	1438.2	464.3
	8	11.8	151.5	100.3	37.5	156.4	469.4	78.0	31.5	1294.8	464.6
Wee-Mag	1	1.2	51.0	94.3	39.9	104.9	136.9	65.7	9.4	3316.6	858.3
	2	4.5	51.2	91.4	35.7	84.9	158.8	61.8	7.7	3534.2	1233.7
	3	2.9	51.0	96.0	29.7	106.3	248.5	92.7	12.4	3295.5	1198.8
	4	5.5	51.2	103.9	35.8	143.3	245.9	81.9	14.2	2929.8	852.6
	5	3.4	64.8	141.2	43.8	57.1	213.9	67.2	11.6	3504.6	228.1
	6	4.1	65.0	155.2	40.1	60.2	225.6	67.2	10.4	2727.8	428.1
	7	4.2	64.6	148.0	27.0	71.3	283.7	68.1	11.0	2251.9	396.8
	8	7.4	64.7	140.6	47.9	90.0	288.1	77.4	11.8	2677.1	664.3

Also, we get the best results for the remaining groups 3 and 8. These results are pioneer in the relevant literature.

- For the Tonge family, we report the minimum cycle time value that have ever found for the group number 4; and for the rest of the family we obtain the best values found so far in the recent literature. As it can be seen from Table 8, only the last two references could have report the best values for some of the test groups. Except our algorithm, none of the above methods can get optimum solutions for all Tonge groups.
- We could not obtain the best results for all of the Wee-Mag family this may be related to problem size. However, for group 5 our solution is 15.0 which is much more less than the solutions obtained by Chaves et al. (2009) and Moreira and Costa (2009). Also, we obtained better results than Moreira and Costa (2009) for the groups 6, 7 and 8.
- Order strength of the Roszieg family is pretty high, but it contains a limited number of tasks (see Table 6). Similarly, the number of tasks of the Heskia family is also low. From this point of view, it can be claimed that the proposed solution procedure is capable of obtaining much better results for small sized test instances.
- Tonge is one of the large-scaled problem families with order strength of 59.42 and task number of 70. The proposed algorithm performs very well on Tonge family; it finds the very best result so far on test group 4 and obtained the best results on rest of them. None of the studies in the relevant literature could obtain the best results for all groups of the Tonge family. Our work not only gets the best results for all of the groups, but also found a new best solution for the group 4. Moreover, test group 4 includes high percentage of task-worker incompatibilities and high operation times variability (see Table 7). Hence, it can be concluded that our algorithm works well on high task-worker incompatibilities and operation time varibilities.
- Table 9 compares the CPU times of our proposed algorithm to the other approaches in the literature. Note that all of the algorithms were run on different computers with different operating systems and performances. Thus, it is not logical to compare them in terms of the computational time. However, it can be stated that our proposed algorithm performs better results in limited times and obtains minimum CPU times at every data family.

Conclusions

Human factors play an important role in labor intensive assembly lines. Since every worker has his own characteristics, such as skill, ability, morale, experience, etc., it is inappropriate to consider workers as unique. However, in classical assembly line balancing literature, this aspect is disregarded and all workers are assumed to be unique. Consequently, task operation times are assumed to be fixed and do not depend on the workers. But this assumption does not represent the real life assembly systems. In order to relax fix operation times assumption, ALWABP is introduced to the assembly line literature, recently.

ALWABP is a decision problem that occurs when operation times of tasks differs according to the operator. Although the operation time of a task is assumed to be fixed in classical ALBP, it depends on the operator who executes the task in ALWABP. Even though ALWABP is a relatively new problem, it has attracted the scientists' interest. Many research studies have been made to solve ALWABP in recent years, but one optimum solution method cannot be found so far. Minimizing the cycle time is commonly used as a primary objective in ALWABP literature (ALWABP-2). Even traditional ALBP is NP-hard, this much more complex ALWABP-2 is NP-hard, of course. Because of the complex problem nature, optimum seeking methods are not capable of solving it. So, in this paper a rule based constructive heuristic approach was proposed in order to solve ALWABP-2. In the proposed MRBCRS heuristic, 39 task priority rules and 4 worker priority rules were used to sequence tasks and select workers. In order to evaluate the performance of the proposed solution procedure, it is tested on ALWABP-2 benchmark data which consists of four families, each having 80 test instances. Every test instance was run for 10 replications, and our proposed heuristic was executed for 3200 times, in total. Experimental results showed that our proposed algorithm outperforms all others in the relevant literature. Concisely, the contribution of this study to the relevant literature is that a novel solution procedure for solving ALWABP-2 was presented and the proposed solution procedure was proven to be efficient in terms of solution quality.

Possible future research directions for both the problem and the solution method include; the proposed solution procedure could be applied to a real life assembly line; some other task/worker selection rules may be added to the heuristic approach; and the proposed heuristic could be extended to solve different assembly line balancing problems with different line configurations such as U-shaped line, stochastic worker availability, multi/mixed model assembly lines.

References

Akyol, S. D., & Bayhan, G. M. (2011). 'A particle swarm optimization algorithm for maximizing production rate and workload smoothness. In *Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, IEEE, October, Spain (pp. 44–49).

Araujo, F. F. B., Costa, A. M., & Miralles, C. (2012). Two extensions for the ALWABP: Parallel stations and collaborative approach.

International Journal of Production Economics, *140*(1), 483–495. doi:10.1016/j.ijpe.2012.06.032.

Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S. & Abraham, A. (2011). 'Inertia weight strategies in particle swarm optimization. In *Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, IEEE, October, Spain (pp. 633–640).

Baykasoğlu, A. (2006). Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. *Journal of Intelligent Manufacturing*, *17*(2), 217–232. doi:10.1007/s10845-005-6638-y.

Blum, C., & Miralles, C. (2011). On solving the assembly line worker assignment and balancing problem via beam search. *Computers and Operations Research*, *38*(1), 328–39. doi:10.1016/j.cor.2010.05.008.

Borba, L., & Ritt, M. (2014). A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. *Computers and Operations Research*, *45*(1), 87–96. doi:10.1016/j.cor.2013.12.002.

Chaves, A. A., Miralles, C., & Lorena, L. A. N. (2007). Clustering search approach for the assembly line worker assignment and balancing problem. In *Proceedings of the 37th International Conference on Computers and Industrial Engineering* (pp. 1469–1478).

Chaves, A. A., Lorena, L. A. N., & Miralles, C. (2009). Hybrid metaheuristic for the assembly line worker assignment and balancing problem. *Lecture Notes in Computer Science*, *5818*, 1–14.

Costa, A. M., & Miralles, C. (2009). Job rotation in assembly lines employing disabled workers. *International Journal of Production Economics*, *120*(2), 625–632.

Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T., et al. (2008). A genetic-algorithm-based optimization model for solving the flexible assembly line balancing problem with work sharing and workstation revisiting. *IEEE Transactions on Systems, Man and Cybernetics Part C—Applications and Reviews*, *38*(2), 218–228.

Guo, Z. X., Shi, L., Chen, L., & Liang, Y. (2015). A harmony search-based memetic optimization model for integrated production and transportation scheduling in MTO manufacturing. In *OMEGA*. doi:10.1016/j.omega.2015.10.012 (in press)

Hoffmann, T. R. (1990). Assembly line balancing: A set of challenging problems. *International Journal of Production Research*, *28*, 1807–1815.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Thatcher (Eds.), *Complexity of computer applications* (pp. 85–104). New York: Plenum Press.

Miralles, C., Garcia-Sabater, J. P., Andres, C., & Cardos, M. (2007). Advantages of assembly lines in sheltered work centres for disabled. A case study. *International Journal of Production Economics*, *110*(1), 187–197. doi:10.1016/j.ijpe.2007.02.023.

Miralles, C., Garcia-Sabater, J. P., Andres, C., & Cardos, M. (2008). Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled. *Discrete Applied Mathematics*, *156*(3), 352–367. doi:10.1016/j.dam.2005.12.012.

Moreira, M. C. O., & Costa, A. M. (2009). A minimalist yet efficient tabu search for balancing assembly lines with disabled workers. In *Anais do XLI Simposio Brasileiro de Pesquisa Operacional*, Porto Seguro, Brazil (pp. 660–671).

Moreira, M. C. O., Ritt, M., Costa, A. M., & Chaves, A. A. (2012). Simple heuristics for the assembly line worker assignment and balancing problem. *Journal of Heuristics*, *18*(3), 505–524. doi:10.1007/s10732-012-9195-5.

Moreira, M. C. O., Miralles, C., & Costa, A. M. (2015). Model and heuristics for the assembly line worker integration and balancing problem. *Computers and Operations Research*, *54*(1), 64–73. doi:10.1016/j.cor.2014.08.021.

Mutlu, O., Polat, O., & Supciller, A. A. (2013). An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. *Computers and Operations Research*, *40*(1), 418–26. doi:10.1016/j.cor.2012.07.010.

Polat, O., Kalayci, C. B., Mutlu, O., & Gupta, S. M. (2015). A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-II: an industrial case study. *International Journal of Production Research*, *54*(3), 722–741. doi:10.1080/00207543.2015.1055344.

Ritt, M., Costa, A. M., & Miralles, C. (2015). The assembly line worker assignment and balancing problem with stochastic worker availability. *International Journal of Production Research*, *54*(3), 907–922. doi:10.1080/00207543.2015.1108534.

Scholl, A., & Voß, S. (1996). Simple assembly line balancing–heuristic approaches. *Journal of Heuristics*, *2*(3), 217–244. doi:10.1007/BF00127358.

Simaria, A. S., & Pedro, M. V. (2004). A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. *Computers and Industrial Engineering*, *47*(4), 391–407.

Sungur, B., & Yavuz, Y. (2014). Assembly line balancing with hierarchical worker assignment. *Journal of Manufacturing Systems*, *37*, 290–298. doi:10.1016/j.jmsy.2014.08.004.

Vila, M., & Pereira, J. (2014). A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers and Operations Research*, *44*(1), 105–114. doi:10.1016/j.cor.2013.10.016.

Zacharia, P Th, & Nearchou, A. C. (2016). A population-based algorithm for the bi-objective assembly line worker assignment and balancing problem. *Engineering Applications of Artificial Intelligence*, *49*, 1–9.

Zaman, T., Paul, S. K., & Azeem, A. (2012). Sustainable operator assignment in an assembly line using genetic algorithm. *International Journal of Production Research*, *50*(18), 5077–5084.