Journal of Intelligent Manufacturing (2021) 32:2121–2124
https://doi.org/10.1007/s10845-020-01632-8

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

Adalberto Sato Michels[1] · **Alysson M. Costa**[1]

Received: 30 June 2020 / Accepted: 13 July 2020 / Published online: 22 July 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract
Some of the solutions reported in a recent paper (Akyol and Baykasoglu in J Intell Manuf 30(2):557–573, 2019. https://doi.org/10.1007/s10845-016-1262-6) are infeasible, jeopardising the conclusion that "best results for the 75% of the 320 test instances for the ALWABP-2" were obtained. In this note, we explore the contributions of the paper, show that some of the solutions are indeed infeasible and point out possible inconsistencies in the employed heuristic that may be causing this issue.

Keywords Assembly line balancing · Worker assignment · Mixed integer linear programming

Introduction

The assembly line worker assignment and balancing problem (ALWABP) is a generalised type of assembly. It has been firstly proposed by Miralles et al. (2007, 2008) and consists in simultaneously deciding assignments of tasks and workers to stations. Workers might have different processing times for each task, and some task-worker incompatibilities may also be defined. Several solution methods have been developed to solve the problem. Among them, Vilà and Pereira (2014) and Borba and Ritt (2014) respectively implemented branch, bound and remember (BB&R) and branch-and-bound (B&B) algorithms. In combination, these algorithms were able to find optimal solutions for 319 instances out of a 320-instance dataset (Chaves et al. 2007).

Recently, Akyol and Baykasoglu (2019) proposed a multiple-rule based constructive randomized search (MRBCRS) heuristic. The MRBCRS heuristic follows a station-oriented assignment procedure, i.e. it prioritises and sequentially assigns tasks and a worker to the station under evaluation. In order to decide on the sequence of workers and tasks to be assigned, priority rules are created. This variety of heuristic had been previously proposed for the ALWABP by Moreira et al. (2012) who extended the work of Scholl and Voß (1997) and defined 16 task priority and 3 worker priority rules. The main contribution in Akyol and Baykasoglu (2019) is the definition and implementation of 39 task and 4 worker priority rules. This contribution has been recognised by several paper citing Akyol and Baykasoglu (2019)'s work (Akyol and Baykasoğlu 2019; Janardhanan et al. 2019; Zhang et al. 2020; Emre and Bulent 2020).

In Akyol and Baykasoglu (2019), the authors claimed they have obtained better or comparable results for 75% of the benchmark upon the best-performing methods from the literature in terms of solution quality and time. A great part of these improvements, however, were due to reporting better results than the known, proven optimal solutions for instance families with a low number of tasks, which implies that these reported solutions are, in fact, infeasible. Although a mixed integer linear programming (MILP) model had been presented, the authors did not apply it to the dataset, although it has been shown that this MILP model, originally proposed by Miralles et al. (2008), is capable of solving to optimality at least small-size instances (Borba and Ritt 2014).

In this note, the mathematical model presented in Akyol and Baykasoglu (2019) is reproduced in (1)–(9), which was written following the same notation and variables used in the original paper (Tables 1 and 2). *Italicised* fonts are used to refer to figures, tables, and algorithms of Akyol and Baykasoglu (2019)'s article, while regular fonts are used to refer to this note. In order to show the incorrectness of results

✉ Alysson M. Costa
alysson.costa@unimelb.edu.au

Adalberto Sato Michels
a.satomichels@unimelb.edu.au

[1] School of Mathematics and Statistics, University of Melbourne, Melbourne, Australia

Table 1 Notation used in Akyol and Baykasoglu (2019)

i, j	Task indexes
h	Worker index
s	Station index
N	Number of tasks
H	Number of workers
W	Number of stations
M	A sufficiently large positive number
P	Set of precedence relations $(i, j) \in N \mid i \prec j$
t_{hi}	Processing time of task i when executed by worker h
M	A big-M value defined as $\sum_{h \in H} \sum_{i \in N} t_{hi}$

in Akyol and Baykasoglu (2019), Sect. 2 presents a counterexample for a group of 10 instances. It demonstrates feasible (and optimal) solutions with explicit task and worker assignment decisions for such group. Then, Sect. 3 discusses possible inconsistencies in the previously employed MRBCRS heuristic that may be causing this issue of allowing infeasible solutions to be accepted as valid. Finally, Sect. 4 ends this note with concluding remarks.

Min c,

subject to: (1)

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

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

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

$$\sum_{h \in H} \sum_{s \in W} s \cdot x_{shi} \leq \sum_{h \in H} \sum_{s \in W} s \cdot x_{shj}, \quad \forall (i, j) \in P, \tag{5}$$

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

$$\sum_{i \in N} x_{shi} \leq M \cdot y_{sh}, \quad \forall h \in H; s \in W, \tag{7}$$

with

$$x_{shi} \in [0, 1], \quad \forall s \in W; h \in H; i \in N, \tag{8}$$

$$y_{sh} \in [0, 1], \quad \forall s \in W; h \in H. \tag{9}$$

The goal is to minimise the cycle time c (1), while assigning each of the tasks to one station (2). The model allows a worker to be assigned to a single station (3) and a station to receive a single worker (4). Constraints (5) impose the precedence constraints for task execution, while (6) enforce c as the value of the line cycle time and (7) relate task and worker assignment variables. Note that the linear relaxation of this model can be improved by disaggregating Constraints (7) and by using stronger precedence constraints such as proposed by Ritt and Costa (2018). For consistency, we use in the following the model without these modifications.

Counterexample

The largest discrepancy between the presented results and known optimal values has been noted for instances in the Roszieg family (low number of tasks and high order strength). The authors state that "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 example, in group 2 (10 instance with low relation between tasks and operators, low variability of operation times, and high percentage of task-worker incompatibilities), the reported average result for cycle time was 16.8, whilst the average for proven optimal solutions is 31.5 (*Table 8*).

Table 3 presents optimal solutions for each instance in the aforementioned group (Roszieg-2). It explicitly shows 25 task and 4 worker assignment decisions to 4 stations for each case. All instances were obtained by solving model (1)–(9) with Gurobi 8.1 on four threads of a 64-bit Intel Core™ i5-7200U CPU (2.5 GHz) with 8GB of RAM. Not surprisingly, optimality was reached for all instances in less than 0.3 seconds. Additionally, cycle times for each individual station are reported, where **bold** values represent the most loaded station, which also corresponds to the line's optimal cycle time. As expected, the average of these bold values is 31.5, which is the best known and proven optimal value (Vilà and Pereira 2014; Borba and Ritt 2014). Hence, by using this counterexample, it is evidenced that the average result (16.8) found by Akyol and Baykasoglu (2019) is lower than the cycle time of any individual instance in this group, thus showing that the solutions obtained by the MRBCRS heuristic may generally be inconsistent and infeasible.

Discussion

In this section, possible flaws in the MRBCRS heuristic are discussed, namely problems with (i) the inability of detecting infeasible solutions and (ii) the cycle time update algorithm loop. The heuristic's flowchart can be observed in *Figure 1* of the original paper. According to this flowchart, the heuristic starts by reading a benchmark data and calculating, among other parameters, an upper bound value for the cycle time (c_{UB}), which is given by equation (10). This upper bound definition itself is incorrect: due to conceivable idle times caused by precedence relations and incompatibilities, the cycle time might actually be worse than a simple sum of the worst processing times divided by the number of available stations. Besides that, authors claim that the heuristic is initialised with such value: "for the first iteration, we set

Table 2 Variables

x_{shi}	Binary variable equal to 1 if task i is assigned to worker h in station s
y_{sh}	Binary variable equal to 1 if worker h is assigned to station s
c	Cycle time

Table 3 Task and worker assignments for instances from the Roszieg family, group 2

Instance		Station 1	Station 2	Station 3	Station 4
1	Worker	4	3	1	2
	Tasks	1,2,3,4,5,6,7,12,15	8,9,10	11,13,14,16,17,23	18,19,20,21,22,24,25
	Cycle time	29	4	**30**	18
2	Worker	4	2	1	3
	Tasks	1,2,3,4,5,6	7,8,9,12,15	11,13,14,17,23	10,16,18,19,20,21,22,24,25
	Cycle time	14	17	**27**	21
3	Worker	3	1	4	2
	Tasks	1,2	3,4,5,6,7,8,9,11,13,14,20,21,24	12,15,16	10,17,18,19,22,23,25
	Cycle time	6	**76**	7	23
4	Worker	2	4	1	3
	Tasks	1,2,3,4,8,9	5,6,7,10,11,12	13,14,15,20,21	16,17,18,19,22,23,24,25
	Cycle time	15	19	23	**25**
5	Worker	1	3	4	2
	Tasks	1,2,3,4	5,6,7,8,9,11,13,14,20	10,12,15,16,17,18,21,23,24	19,22,25
	Cycle time	21	**26**	24	10
6	Worker	4	1	2	3
	Tasks	1,2,3	4,5,8	6,7,9,10,11,12,13,15,17,23	14,16,18,19,20,21,22,24,25
	Cycle time	11	21	**22**	**22**
7	Worker	4	1	3	2
	Tasks	1,2,3,4,8,9	5,6,7,12	11,13,14,15,20,21,24	10,16,17,18,19,22,23,25
	Cycle time	18	**22**	21	19
8	Worker	4	3	1	2
	Tasks	1,2,3,4,5,6,7	8,9,11,12,13,15,17	10,14,19,20,21,23	16,18,22,24,25
	Cycle time	**20**	18	19	**20**
9	Worker	2	1	3	4
	Tasks	1,2,3,4,8	5,6,7,12,15	9,11,13,17,23	10,14,16,18,19,20,21,22,24,25
	Cycle time	19	**27**	18	**27**
10	Worker	4	2	3	1
	Tasks	1,2,3	4,5,8,9	6,7,10,11,12,13,15,17,23	14,16,18,19,20,21,22,24,25
	Cycle time	6	14	22	**40**

cycle time equal to the c_{UB}". If c_{UB} is calculated for Roszieg-2 instances, as given in Expression 10, one will realise that $c_{UB} = 32$ for all of them, which is contradictory to instances 3 and 10 optimal values (76 and 40, respectively). Therefore, we suspect that the heuristic is incapable of properly evaluating and detecting infeasible solutions, since the flowchart shows that the heuristic would stop in the case of encountering one.

$$c_{UB} = \max\left\{ \sum \frac{\max\{t_{hi}\}}{|W|}, \max\{t_{hi}\} \right\}, \quad \forall h \in H; i \in N. \tag{10}$$

Another possible inconsistency may be uncovered in the cycle time update loop, whose pseudo code is illustrated in *Algorithm 3*. Whenever an allegedly feasible solution is found, the heuristic updates the cycle time limit for the next iteration, multiplying the current one for a given rate (< 1) depending on the ratio of the current cycle time to the upper bound of the cycle time. In their study, "the ratio of the cur-

rent 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 (0.75). 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 (0.85) 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 (0.99) in order not to miss the minimum cycle time". The problem here is that the heuristic's flowchart in *Figure 1* does not present any step unassigning tasks from stations, which would lead the algorithm to not even testing the new cycle time limit and deeming it as feasible, indefinitely reducing the cycle time until the stopping criteria on the number of iterations is reached. Finally, even in the circumstances that it is not the case and the new cycle time is in fact evaluated, the heuristic does not seem to be able to notice most infeasible assignment decision. Hence, it would keep accepting lower cycle time values in either situation.

Conclusion

Akyol and Baykasoglu (2019) propose a heuristic for the assembly line worker assignment and balancing problem (ALWABP). The heuristic is based on new priority rules for task and worker assignment. In this note, we point out some inconsistencies in the reported results (i.e. infeasible solutions). We discuss some features of the MRBCRS heuristic that may be causing the issues, including the use of an incorrect upper bound for the problem.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Akyol, S. D., & Baykasoglu, A. (2019). A multiple-rule based constructive randomized search algorithm for solving assembly line worker assignment and balancing problem. *Journal of Intelligent Manufacturing*, *30*(2), 557–573. https://doi.org/10.1007/s10845-016-1262-6.

Akyol, S. D., & Baykasoğlu, A. (2019). ErgoALWABP: A multiple-rule based constructive randomized search algorithm for solving assembly line worker assignment and balancing problem under ergonomic risk factors. *Journal of Intelligent Manufacturing*, *30*(1), 291–302. https://doi.org/10.1007/s10845-016-1246-6.

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*, 87–96. https://doi.org/10.1016/j.cor.2013.12.002.

Chaves AA, Lorena LAN, Miralles C (2007) Clustering search approach for the assembly line worker assignment and balancing problem

Emre, C., & Bulent, D. M. (2020). A novel optimization approach for segmented rabbit chase oriented U-type assembly line design: an application from lighting industry. *Assembly Automation*, *40*(3), 483–510. https://doi.org/10.1108/AA-02-2019-0020.

Janardhanan, M. N., Li, Z., & Nielsen, P. (2019). Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem. *Soft Computing*, *23*(21), 11263–11276. https://doi.org/10.1007/s00500-018-03684-8.

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*(1–2), 187–197.

Miralles, C., García-Sabater, J. P., Andrés, C., & Cardós, 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. https://doi.org/10.1016/j.dam.2005.12.012.

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. https://doi.org/10.1007/s10732-012-9195-5.

Ritt, M., & Costa, A. M. (2018). Improved integer programming models for simple assembly line balancing and related problems. *International Transactions in Operational Research*, *25*(4), 1345–1359.

Scholl, A., & Voß, S. (1997). Simple assembly line balancing-heuristic approaches. *Journal of Heuristics*, *2*(3), 217–244.

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

Zhang, Z., Tang, Q. H., Ruiz, R., & Zhang, L. (2020). Ergonomic risk and cycle time minimization for the U-shaped worker assignment assembly line balancing problem: A multi-objective approach. *Computers and Operations Research*, *118*, 104905. https://doi.org/10.1016/j.cor.2020.104905.

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