Tabu search algorithm for single and multi-model line Balancing problems

Mohamed Amine Abdeljaouad[1] and Nathalie Klement[2]

[1] CEA Tech Hauts-de-France, 165 Avenue de Bretagne, Lille, 59000 France
mohamed-amine.abdeljaouad@cea.fr
[2] Arts et Métiers Institute of Technology, LISPEN, HESAM Université, Lille, France
Nathalie.klement@ensam.eu

Abstract. This paper deals with the assembly line balancing issue. The considered objective is to minimize the weighted sum of products' cycle times. The originality of this objective is that it is the generalization of the cycle time minimization used in single-model lines (SALBP) to the multi-model case (MALBP). An optimization algorithm made of a heuristic and a tabu-search method is presented and evaluated through an experimental study carried out on several and various randomly generated instances for both the single and multi-product cases. The returned solutions are compared to optimal solutions given by a mathematical model from the literature and to a proposed lower bound inspired from the classical SALBP bound. The results show that the algorithm is high performing as the average relative gap between them is quite low for both problems.

Keywords: Line balancing, optimization, tabu search. SALBP, MALBP

1 Introduction

Since their first use in 1913 as part of the Ford automotive manufacturing process, assembly lines have spread all over the world. They are now one of the main forms of industrial production, as they significantly reduce the manufacturing time, increasing thus the productivity. They also reduce the needed workforce and therefore the production costs.

The performance of an assembly line depends on various factors. Finding a good balancing of the workload over the line's stations is one of the most important. Several line-balancing problems have been dealt with in the literature. A review on the issue can be found in [18]. Assembly line balancing problems can first be classified according to the number of models (types of product) that the line can produce: we distinguish between single-model assembly line balancing problems (SALBP), where a unique type of product is produced, and the mixed/multi-model assembly line balancing problems (MALBP) where the outcomes are products of different types. Line balancing problems can also be classified according to the type of the line (serial line, line with parallel workstations…) and the durations of the operations (deterministic or stochastic).

Different objective functions and constraints have been considered by researchers. The most known objective functions for single-model lines are the minimization of the number of workstations for a given cycle time (type SALBP-1) and the minimization of the cycle time (type SALBP-2) for a given number of workstations [17]. Of course, this list is not exhaustive. In MALB problems, the objectives are more sophisticated. To name a few, we can mention the reduction of the assembly cost [20], the labor cost [22], the line idle time [16] or the smoothness index [15]. As for the other operational research problems, the methods used for the balancing of assembly lines can be exact or approximate. Exact methods range from mathematical programming [6, 9] to branch and bound [12, 14], among others. However, as line balancing problems are usually NP-hard, exact methods cannot solve big-sized instances in a reasonable time. The alternative is therefore to use heuristic methods, which can quickly find satisfying solutions. Various kind of heuristics have been proposed for SALBP and MALBP [7, 11, 21]. Most of the time, the heuristic methods are combined with a metaheuristic that allows to visit more solutions and thus to improve the quality of the heuristic's one [8]. Different kind of metaheuristics have been used; we can mention simulated annealing [4, 13], tabu search [3, 19], genetic algorithms [2, 5] or ant colony optimization [1].

In this paper, we deal with a line balancing problem where the objective is to minimize the weighted sum of products' cycle times (i.e. the weighted sum of the maximum time each type of product spends on a workstation), where the weight of each product represents its ratio among all the demanded quantity. This objective, introduced by [22] as part of a multi-objective MALB optimization problem, can be applied for both the MALBP and the SALBP (where it will be equivalent to the minimization of cycle time, i.e. the maximum of the sum of task times assigned to each workstation). Therefore, we will treat both cases in this work, and provide an optimization algorithm that will be tested on single and multi-models instances. The rest of the constraints of the problems are the following: n operations have to be assigned to m serial workstations. Each operation can be assigned to any workstation but the precedence constraints should be respected: an operation i cannot be assigned to a station k_1 if one of its predecessors j is assigned to a workstation k_2 such that index $k_1 < k_2$. We consider that the durations of the operations are deterministic. Apart from the workstations, all the other resources needed for the processing of the operations (such as workers or tools) are assumed available. This issue is NP-hard [10].

The rest of the paper is structured as follows: In section 2, an optimization algorithm, based on a heuristic and a tabu-search metaheuristic is presented. In section 3, we carry out an experimental study to assess the performance of our algorithm and we conclude the paper in section 4 with our remark and perspectives.

2 Tabu search algorithm

The proposed solving method is made of a heuristic and a metaheuristic. The heuristic leads to an initial feasible solution and is based on the computation of a lower bound. This bound is inspired from the classical bound of SALBP-2 given in equation (1), with J being the number of product types, m the number of workstations, n_j the number of

operations required by product type j and p_i the processing time of operation i. The latter is calculated for each product model and the lower bound for our problem is thus equal to the weighted sum of these cycle time bounds, as specified in equation (2), where w_j is the demand proportion for product model j.

$$LB_j = \max\left(max_{i=1,\dots n_j}(p_i), \quad \sum_{i=1}^{n_j} \frac{p_i}{m} \right) \qquad \forall j = 1, \dots J \qquad (1)$$

$$Lower\ bound = \sum_{j=1}^{J} w_j LB_j \qquad (2)$$

The heuristic starts by sorting the operations on the basis of their precedence constraints. It puts them into groups as follows: group 1 contains the operations that do not have a predecessor, group 2 contains the operations whose predecessors are in group 1, group 3 contains the operations whose predecessors are in groups 1 and 2, and so on. The operations within each group are then sorted in the decreasing order of the number of their successors and, in case of a tie, in the decreasing order of the sum of their successors' processing times.

The heuristic then browses the operations group by group, in that order, and assigns them to the workstations, starting by station $k = 1$. At each iteration i, it selects the first non-assigned operation o from the group and assigns it to a workstation, according to the rule given below, with S_{jk} being the sum of the operations' processing times of product type j that are already assigned to station k at iteration i:

- If assigning operation o to station k reduce the gap between the value $\sum_{j=1}^{J} w_j S_{jk}$ and the $Lower\ bound$, or if $k = m$, then assign o to station k.
- Otherwise: $k \leftarrow k + 1$

The solution returned by the heuristic is then used as input for a tabu search algorithm that tries to improve it. The latter's main steps are given in Algorithm 1.

Algorithm 1: ***Tabu search***

Input, data: Initial solution S, stopping criterion c, size of the tabu list ts;
Initialization: *Best solution* ← S; *Current solution* ← S; $i \leftarrow 0$; *Tabu List* ← ∅;
while $i < c$ **do**
  for each product type j **do**
    Select station k on which j has the longest processing time
    Randomly select one of the operations i of j on k with $i \notin$ *Tabu List*
    Move i to either station $k - 1$ and $k + 1$ (the less loaded one if the precedence constraints are respected)
    Compute the new solution S_j
  Current solution ← $min_{j=1,\dots J}(S_j)$;
  Add the moved operation i to *Tabu List* for ts iterations;
  if *Current solution* < *Best solution* **then**
    Best solution ← *Current solution*
  else $i \leftarrow i + 1$
return (*Best Solution*)

The idea behind choosing a tabu search method is to create a solution neighborhood structure for each product type: At each iteration and for each product type, the tabu search algorithm selects the workstation on which the selected type has its longest processing time and moves one of its operations to another station, by respecting the precedence constraints. The new obtained solutions are then compared and the best one is kept. The moved operation is added to a tabu list during a certain number of iterations, to prevent the algorithm from being stuck in the same movements' loop.

A neighborhood structure based on the different product types seems useful since the considered objective directly depends on the cycle time of each model. This structure can ease the search process of efficient and high-quality solutions. To the best of our knowledge, this kind of neighborhood was never used before for multi-model balancing optimization. The algorithm repeats this process and stops after a certain number of consecutive iterations without improving the solution.

3 Experimental study

Although there are many benchmark instances from line balancing literature, only few of them are made for MALB problems. Moreover, to the best of our knowledge, there is no comparative study dealing with the same objective function. Therefore, we choose to test the proposed algorithm on several randomly generated instances, where the durations, the precedence constraints and the repartition of the operations over the product types are generated given some probability parameters. We generated two families of instances: small-sized and big-sized. For the small-sized ones, we compared the algorithm's solutions to optimal solutions, obtained with a mathematical model inspired from the formulation presented in [22]. These small-sized instances are made of {10, 20, 50} operations, {3, 5, 6} workstations and {1, 3, 5} product types. The big-sized ones are made of {100, 200} operations, {6, 10, 15, 20} workstations and {1, 3, 5} product types. For these latter instances, the returned solutions are compared to the lower bound, since the optimal solving of these instances with the mathematical model cannot be done in a reasonable time. For each size, 5 instances are tested, for a total of 225 tests. The durations of the operations are generated with a uniform distribution in [1, 99]. The precedence relationship between the operations and the repartition of the operations over the different product types are generated based on data from a real life assembly line manufacturing pneumatic cylinders.

All the instances were quickly solved by our algorithm. Those with 200 operations and multiple products required between 1 and 3 minutes to be solved, while the other instances took only a few seconds. As it is expected from an approximate approach, the proposed method thus shows its ability to solve rather quickly big-sized instances of this NP-hard problem, for which the mathematical solving may take hours. Table 1 and Table 2 display the results obtained for the small-sized and big-sized instances, respectively. In Table 1, the 'GLB' column gives the average relative gap between the lower bound calculated in equation (2) and the optimal solutions, while the 'GSol' column gives the average relative gap between the algorithm's solutions and the

optimal ones. In Table 2, the 'Products' columns display the average relative gap between the obtained solutions and the lower bound.

As we can see from Table 1, our algorithm is high performing for both the single and multiple products instances, with a 0.99% overall average relative gap to the optimal solutions. This is confirmed by a 42.2% rate of optimal solutions among those returned by the algorithm. Table 1 also shows that the algorithm's solutions are closer to the optimal ones than the lower bound, whose average relative gap is higher (about 6.3%). This gap between the bound and the optimal solutions helps to explain the behavior of our algorithm for the big-sized instances: Indeed, even if the average relative gap between the algorithm's solutions and the lower bound is a bit higher in Table 2, this may be due to the gap between the bound itself and the optimal solutions. The results of our algorithm are therefore also promising for the big-sized instances.

Table 1. Results for the small-sized instances.

Operations	**Stations**	**Products**					
		1		**3**		**5**	
		GLB	GSol	GLB	GSol	GLB	GSol
10	3	2.26	0.76	12.47	0	16.07	0
	5	12.89	0	13.33	0.29	7.5	0.19
	6	7.24	1.89	8.28	0	0.21	0
20	3	1	0	5.29	0.16	8.93	0.26
	5	3.58	0.62	9.47	1.43	13.99	1.4
	6	2.77	1.94	11.36	1.8	13.93	2.74
50	3	0	0.07	0.77	0.9	1.88	0.74
	5	0	0.53	2.12	2.44	3.63	2.4
	6	0	0.84	2.51	2.4	8.98	3.2
Average		**3.3%**	**0.73%**	**7.28%**	**1.04%**	**8.34%**	**1.21%**

Table 2. Results for the big-sized instances

Operations	**Stations**	**Products**		
		1	**3**	**5**
100	6	0.78	4.34	5.04
	10	1.5	7.46	9.64
	15	2.81	12.42	15.01
200	6	0.17	2.22	2.87
	10	0.56	4.43	6.26
	20	1.29	10.44	12.81
Average		**1.18%**	**6.88%**	**8.6%**

4 Conclusion

In this paper, an optimization algorithm based on a tabu search procedure is presented to solve a line-balancing problem. The objective is to minimize a weighted sum of the

product cycle times, where the weights represent the demand-ratio for each product. The algorithm is tested on both single-model and multi-model problems and the results obtained show that the outcomes are very satisfying in both cases, with a close average relative gap to the optimal solutions and to the lower bound.

The next step of this work is to test the algorithm for other objective functions for the multi-model line-balancing problems. As a perspective, the balancing obtained for these different objective functions can also be compared through a simulation for example, in order to analyze their impact on the production, considering various constraints. Future work also includes the experimentations of other heuristics to provide the initial solutions for the tabu search algorithm and testing the method on a real assembly line. A case-study line producing several models of pneumatic cylinders has already been linked with this project.

References

1. Akpinar S., Bayhan G.M., Baykasoglu A. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Applied Soft Computing, 13, 574–589 (2013)
2. Alavidoost M.H., Fazel Zarandi M.H., Tarimoradi M., Nemati Y.: Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times. Journal of Intelligent Manufacturing, 28, 313-336 (2017).
3. Arikan M.: Type-2 assembly line balancing with workload smoothing objective: A reactive tabu search algorithm. Gazi University Journal of Science, 34(1), 162-178 (2021).
4. Baykasoglu A.: Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. Journal of Intelligent Manufacturing 17:217–232 (2006)
5. Chong K.E., Omar, M.K., Baker N.A. Solving assembly line balancing problem using genetic algorithm with heuristic treated initial population. Proceedings of the World Congress on Engineering, 978-988 (2008)
6. Çil Z.A., Li Z., Mete S., Özceylan E.: Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human-robot collaboration.
7. Chutima P, Olanviwatchai P Mixed model U shaped assembly line balancing problem with coincidence memetic algorithm. Journal of Software Engineering and Applications, 3, 347–363 (2010)
8. Fatini D.M., Mohammad F.R., Mohd Z.Z.: A review on hybrid metaheuristics in solving assembly line balancing problem. AIP Conference Proceedings, 2138(1) (2019).
9. Gokcen H, Erel E. Binary integer formulation for mixed model assembly line balancing problem. Computers & Industrial Engineering, 34(2): 451–461 (1998)
10. Gutjahr, A.L., Nemhauser, G.L.: An algorithm for the line balancing problem. Management Science, 11(2), 308-315 (1964).
11. Kilincci O.: A Petri net-based heuristic for simple assembly line balancing problem of type 2 The international Journal of Advanced Manufacturing Technology 46, 329– 338 (2010)
12. Klein, R., Scholl, A.: Maximizing the production rate in simple assembly line balancing — A branch and bound procedure. European Journal of Operational Research, 91(2), 367-385 (1996).
13. Lalaoui M., El Afia, A.: A fuzzy generalized simulated annealing for a simple assembly line balancing problem. IFAC-PapersOnLine, 51(32), 600-605 (2018)

14. Li Z., Kucukkoc I., Zhang Z.: Branch, bound and remember algorithm for U-shaped assembly line balancing problem. Computers & Industrial Engineering, 124, 24-35 (2018).
15. Roshani A., Roshani A., Roshani A., Salehi M., Esfandyari A.: A simulated annealing algorithm for multi-manned assembly line balancing problem. Journal of Manufacturing Systems, 32, 238– 247 (2013)
16. Sarker B.R., Pan H. Designing a mixed-model assembly line to minimize the costs of idle and utility times. Computers and Industrial Engineering, 34(3), 609-628 (2001)
17. Scholl, A., Becker, C. State-of-the-art Exact and Heuristic Solution Procedures for Simple Assembly Line Balancing. European Journal of Operational Research, 168(3), 666-693 (2006).
18. Sivasankaran P., Shahabudeen P.: Literature review of assembly line balancing problems. The international Journal of Advanced Manufacturing Technology, 73, 1665-1694 (2014).
19. Suwannarongsri S, Limnararat S.: A hybrid tabu search method for assembly line balancing. Proceedings of the 7th international conference on simulation (modelling and optimization) China (2007).
20. Tseng Y.J., Chen J.Y., Huang F.Y.: A multi-plant assembly sequence planning model with integrated assembly sequence planning and plant assignment using GA. The international Journal of Advanced Manufacturing Technology, 48, 333-345 (2010).
21. Yeh D.H., Kao H.H.: A new bidirectional heuristic for the assembly line balancing problem. Computers & Industrial Engineering, 57, 1155–1160 (2009)
22. Zhang W., Gen, M.: An efficient multi-objective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. Journal of Intelligent Manufacturing, 22, 367–78 (2011).