Taylor & Francis
Taylor & Francis Group

Production & Manufacturing Research

An Open Access Journal

ISSN: (Print) 2169-3277 (Online) Journal homepage: http://www.tandfonline.com/loi/tpmr20

Heuristics for solving a multi-model robotic assembly line balancing problem

Mads Kammer Christensen, Mukund Nilakantan Janardhanan & Peter Nielsen

To cite this article: Mads Kammer Christensen, Mukund Nilakantan Janardhanan & Peter Nielsen (2017) Heuristics for solving a multi-model robotic assembly line balancing problem, Production & Manufacturing Research, 5:1, 410-424, DOI: 10.1080/21693277.2017.1403977

To link to this article: https://doi.org/10.1080/21693277.2017.1403977

Published online: 27 Nov 2017.

PRODUCTION & MANUFACTURING RESEARCH, 2017
VOL. 5, NO. 1, 410–424
https://doi.org/10.1080/21693277.2017.1403977

Taylor & Francis
Taylor & Francis Group

Heuristics for solving a multi-model robotic assembly line balancing problem

Mads Kammer Christensen[a], Mukund Nilakantan Janardhanan[b] and Peter Nielsen[b]

[a]Department of Mathematical Sciences, Aalborg University, Aalborg, Denmark; [b]Department of Materials and Production, Aalborg University, Aalborg, Denmark

ABSTRACT

Topic of balancing assembly lines is of great interest for researchers and industry practitioners due to the significant impact it has on increasing productivity and efficiency of manufacturing systems. Robots are widely applied in manufacturing industries for assembly processes. Wide literature has been reported on balancing of robotic assembly lines with single and mixed models. Researchers have extensively used heuristics and metaheuristics to solve these problems due to their NP-hard nature. However, no work has been reported on how to balance a robotic assembly line with multiple models (MuRALB) with batch production. This problem is highly relevant for large-scale assembly of products found, e.g. the automotive industry. To authors' knowledge, this is the first attempt to solve this problem. This research proposes a novel heuristic to solve type II MuRALB problem. Type II problem deals with minimizing the cycle time for a fixed set of robots. Heuristic is implemented, and method for scheduling batched production with related setup times for a robotic assembly line is presented, and based on the analysis conducted, advantage of batching is presented. Proposed heuristic is tested on a set of new five datasets, and performance of this heuristic and batching is presented in detail.

ARTICLE HISTORY
Received 25 September 2017
Accepted 6 November 2017

KEYWORDS
Heuristics; assembly line balancing; robots; batch production; multi-model

1. Introduction

Assembly lines are considered a critical part of most production systems that assemble a wide variety of products (Hu et al., 2011). Balancing of assembly lines arises as an important problem when a new assembly line is designed, or an existing assembly line needs to be redesigned/reconfigured (Battaïa & Dolgui, 2013; Liao, 2014). The process of balancing lines of various configurations has an extensive history in the literature and is still considered a significant field of study with new variations – such as systems allowing for rapid reconfiguration – still gaining in interest (Gyulai, Kádár, Kovács, & Monostori, 2014). Due to the dynamic nature of the demand for different product variants that are required to be assembled, it is very necessary to adjust the assembly line setup regularly, which results in

CONTACT Mukund Nilakantan Janardhanan mnj@m-tech.aau.dk

increased assembly line operation cost (Altemeier, Helmdach, Koberstein, & Dangelmaier, 2010).

Assembly lines comprise a set of workstations connected by a conveyor system or similar type of material transport system. The assembly of a product is divided into different tasks, and each of these tasks is executed on the available workstations (Sivasankaran & Shahabudeen, 2014) based on a predetermined precedence relationship. The simple assembly line balancing (ALB) problem primarily aims at allocating tasks equally to all workstations in such a manner that all tasks are completed without violating the precedence relationships (Becker & Scholl, 2006; Nilakantan, Ponnambalam, Jawahar, & Kanagaraj, 2015). An assembly line taxonomy has been proposed and classified according to five elements (Battaïa & Dolgui, 2013):

- Number of lines to be balanced,
- Task attributes,
- Workstation attributes,
- Constraints to be respected by a feasible solution,
- Criteria used to distinguish better or when possible the best (optimal) solutions.

The size and complexity of the problem are influenced mainly by the first three elements, while the latter two influence the basic nature of the solution method. In addition to these classification elements, we propose to add one related to whether humans or robots perform the assembly tasks. Due to the specialization in capabilities for robots, this distinction seems highly relevant (Levitin, Rubinovitz, & Shnits, 2006). This single-model taxonomy is seamlessly integrated with the classification system based on the number of different models produced, which distinguishes between single-, mixed-, and multi-model assembly line balancing (Liao, 2014; Sivasankaran & Shahabudeen, 2014). In single-model assembly lines, the assembly of a single product is considered, whereas mixed- and multi-model assembly lines deal with the assembly of more than one product at the same time. In mixed-model assembly lines, the assembly of different products is carried out in a chosen intermixed sequence with a lot size of one (Bukchin, Dar-El, & Rubinovitz, 2002). In the case of multi-model assembly lines, the assembly of products in a sequence of batches with intermediate setup operations is carried out (van Zante-de Fokkert & de Kok, 1997).

A selection of different high-quality products is one of the criteria widely considered when implementing robotic assembly lines (Çil, Mete, & Ağpak, 2016a; Gao, Sun, Wang, & Gen, 2009). The emphasis on product diversity makes mixed- and multi-model assembly line problems highly relevant for the industry (Boysen, Fliedner, & Scholl, 2008). Multi-model assembly lines are used primarily to assemble a variety of similar products which differ in the task precedence sequence. Different products are assembled in batches, and before each batch is assembled, a setup takes place, thereby incurring a setup time (Liao, 2014). A detailed review of mixed- and multi-model assembly line balancing problems is found in (van Zante-de Fokkert & de Kok, 1997) who also present a detailed comparison of heuristics used for balancing. Updated reviews of line balancing problems are found in (Battaïa & Dolgui, 2013; Boysen et al., 2008). Due to the increasing demand for product variety from customers, industry has had to increase the flexibility of assembly line systems. Thus, robots are extensively replacing human labor in assembly lines to reduce the increasing costs of greater customizability, and such systems are referred to as robotic assembly lines (RAL). Robots can perform a variety of tasks without tiring or requiring breaks and

can be equipped with different tools to perform different tasks. Assembly lines with robots improve the production rate as well as the quality of the products assembled (Levitin et al., 2006). Allocating the best fit robot is one of the important questions addressed in an RAL (Çil, Mete, & Ağpak, 2016b). Balancing of the robotic assembly line (RALB) mainly deals with assigning tasks to workstations and allocating the robot best fit to perform these tasks (Rubinovitz, Bukchin, & Lenz, 1993). The literature on balancing a manual single-, mixed-, or multi-model assembly line considers various assumptions to facilitate the process's ease of execution or its likeness to reality, as described in Sivasankaran and Shahabudeen (2014). One of the most common and inaccurate assumptions for classic assembly line balancing (ALB) is that of deterministic production and setup times, an assumption being more prudent when considering robotic assembly line balancing (RALB) (Rubinovitz et al., 1993).

With the advent of RALB and continuous improvements within the field of robotics, the assumption of deterministic production and setup times becomes even more plausible. As assembly line balancing procedures reduce their variance by implementing robotics (Rubinovitz et al., 1993), the relevance of good balancing methods increases. Over the years, literature based on RALB with respect to different objectives has grown substantially. The literature has addressed two main types of RALB problems extensively. They are the type I RALB problem which deals mainly with minimizing the number of workstations for a given cycle time and the type II RALB problem dealing with minimizing the cycle time for a given number of workstations (Nilakantan et al., 2015). However, most of the focus has been on single- and mixed-model robotic assembly lines and the type II RALB problem (Rabbani, Mousavi, & Farrokhi-Asl, 2016). Over the years, researchers have addressed different objectives, such as minimizing energy consumption and assembly line cost of RALB (Li, Janardhanan, Tang, & Nielsen, 2016, in press; Nilakantan, Li, Tang, & Nielsen, 2017; Nilakantan, Ponnambalam, & Jawahar, 2016). A limited number of contributions are reported for robotic assembly line mixed-model setup, and most of those focused on minimizing the cycle time (Çil et al., 2016a).

Although both mixed- and multi-model ALB are covered widely in the literature, multi-model assembly lines with robots have not been treated in any considerable manner, and thus, we face a gap in production scheduling. With the increasing interest in low-volume construction of customized products, it seems ever more prudent to consider production of low-volume batches (Scholl & Becker, 2006). The primary difference between batched and individual production runs is found when considering the need for changing the setup of the robots when changing their tasks, which is a key focus point of the RALB problem (Gao et al., 2009). By allowing multiple identical products to undergo the same set of assembly processes without having to reconfigure the robots between single process executions, we intend to reduce the time needed to produce each product, while still allowing the production of a range of different products. Based on the workflow of the production line, RALB problems are categorized in the literature as either straight (traditional) types or U-shaped (Mukund Nilakantan & Ponnambalam, 2016).

Due to the complexity of assembly line balancing problems being NP-hard, optimizing such problems is computationally difficult for large problem instances (Becker & Scholl, 2006). In last few decades, heuristics and exact solution procedures have been widely applied to solve the ALB and RALB problems (Scholl & Voß, 1997). The advent of metaheuristics and its ability to arrive quickly at acceptable solutions to complex problems enabled the timely solution of the NP-hard assembly line balancing problems (Nilakantan et al., 2015). We

likewise know from literature that the typical company must aggregate and batch (Eriksen & Nielsen, 2016) demand for products to achieve a stable input to their manufacturing system. The need to aggregate and thus batch products in production context has long been known to be a challenging problem (Axsäter, 1981; Wijngaard, 1982). The topic is highly significant but has not received much attention in recent research. However, it is well known that to stabilize input to manufacturing systems, one wants to reduce the demand variation. Demand variation occurs naturally, but batching demand can reduce the impact of variation while simultaneously reducing the number of setups and thus indirectly increasing line efficiency. From the literature study, it can be seen that no work has been reported for balancing a robotic assembly line with multi-models and batching. This paper mainly presents the following contributions to the field of RALB.

(1) A mathematical model is developed to minimize cycle time of a multi-model robotic assembly line, hereafter referred to as (type II MuRALB).
(2) A novel heuristic is developed to solve the proposed problem.
(3) A set of benchmark problems is developed to test the proposed heuristic, and computational performance of the heuristic has been tested.

The remainder of this paper is structured as follows: Section 2 describes the problem assumption and mathematical model for the proposed MuRALB problem. Section 3 presents the heuristic proposed to solve the problem along with pseudocode of the proposed algorithm. Section 4 illustrates the solving procedure for MuRALB problem using a small-sized problem. Section 5 presents the detailed experimental study and the results obtained for the proposed problem with different types of dataset problems generated and presents the details on how the proposed algorithm handles problems of increased sizes. Section 6 summarizes the findings of the paper with possible future research directions.

2. Problem formulation and assumptions

An assembly line setup with R robots available to perform T tasks is considered. In this paper, we are interested in developing a scheduling heuristic for assembling product batches of size B. In this paper, we assume the assembly line to be arranged in a straight line to allow full mobility of the robots in question, and precedence relationship *(prc)* of the tasks is considered. For our deliberations on batched production runs in robotic assembly line balancing, we consider a single-objective type II MuRALB problem. We are concerned with minimizing the total time required to produce a given number of different batches of identical products. We consider both setup and production times for performing different tasks for different robots. Sharing of the tasks among the workstation is not considered. Robots used are free to move between the assembly tasks. Spatial restrictions are disregarded in this paper for the development of the heuristic for assigning batched assembly tasks. One of the major issues while considering production batches instead of single product production is the tracking when the first product of the batch has passed through a process and when the entire batch has passed. In this paper, we are concerned with enabling the intermixing of tasks for different product batches in order to attain the fastest possible production of a given number of batches of a certain size.

2.1. Assumptions

The following assumptions for the MuRALB heuristic are based on those made by Rabbani et al. (2016); however, it is different with respect to the robot availability assumption, and the assumption with regard to spacing is not considered. Since this study is the first attempt to solve a multi-model robotic assembly line balancing problem, and it is well known that problem of this nature is NP-hard Rabbani et al. (2016)and computationally hard to solve. This study will act as a starting platform for further researches in this area. Hence, due to additional complexities involved by adding these constraints, they are not included in this paper.

- Assembly tasks cannot be subdivided. The precedence relations among the tasks are distinctive and constant. The precedence graph describes the precedence relations.
- The processing time of an assembly task is deterministic and depends on the assigned robot.
- Setup times between tasks are deterministic and depend on the assigned robot.
- Robots are fully mobile, eliminating the need to consider transport time between tasks.
- The line is balanced for multiple different products.
- The line is balanced for batches of identical products.
- The same robot performs the same task in each cycle.
- One assembly task per product per robot is undertaken at a time.
- Only one of each type of robot is available for task allocation.

2.2. Mathematical model

The mathematical formulation for the MuRALB problem is developed with the aim of minimizing cycle time, and the mathematical formulation is presented. The underlying mathematical considerations of the MuRALB heuristic are described by Equations (1), (2), (3), and (4), with notation presented.

- Indices

  t : Set of assembly tasks $t = 1, 2, \ldots, T$

  rt : Set of required tasks $rt = 0$, $rt < t$

  r : Set of robots $r = 1, 2 \ldots, R$

- Parameters

  Prc : T × T binary matrix of predecessors for tasks t

  $s_{r,t}$: Setup time of robot r for processing task t

  $p_{r,t}$: Processing time of task i by robot r

  B : Product batch size

- Decision variables

  $x_{r,t}$: 1, if the common task I is assigned to robot r, 0, otherwise

  C_t : Total cycle time for all products

Objective:

$$\min C = \sum_{t=1}^{T} \sum_{r=1}^{R} x_{t,r} \cdot (s_{t,r} + B \times p_{t,r}) \tag{1}$$

Subject to:

$$\sum_{r=1}^{R} x_{s,r} - \sum_{r=1}^{R} x_{t,r} \leq 0, \qquad \forall s \in Prc(t), \tag{2}$$

$$\sum_{r=1}^{R} x_{t,r} \qquad \forall t \in T, \tag{3}$$

$$x_{t,r} \in \{0, 1\} \tag{4}$$

Equation (1) minimizes the cycle time for all models. Equation (2) is the precedence constraints. Equation (3) ensures that each robot is assigned only one task at a time. Equation (4) indicates the binary assignment variables.

3. Heuristic Algorithm for MuRALB problem

Heuristic is defined as a technique which seeks good (i.e. near optimal) solutions at reasonable computational cost without being able to guarantee either feasibility or optimality (Reeves, 1993). Over the years, heuristics have been used quite often to solve complex real-world optimization problems. Heuristics can also help to create a solution or improve an existing solution by exploring the neighboring solutions based on certain rules or strategies (Erel & Sarin, 1998). Scholl and Voß (1997) proposed two types of heuristics for simple assembly line balancing problems. Over the years, different researchers have applied heuristics for solving assembly line balancing problems and problems of similar type (Hosseini Nasab, Tavana, & Yousefi, 2014). Applying heuristics provides a quick feedback or solution in a design process. This helps industry practitioners to modify the required design in a reasonable amount of time. The results from heuristic can act a benchmark for a problem that is proposed for the first time in the literature. These are the few reasons why heuristics have been selected to solve the problem of this nature (Jawahar, Ponnambalam, Sivakumar, & Thangadurai, 2014). The assembly line balancing problem has been classified as an NP-hard problem (Becker & Scholl, 2006), and the problem considered in this paper also falls under this category due to additional constraints included because of the multi-batch production and robots used for the assembly. To solve the NP-hard MuRALB problem within a reasonable computation time, we develop a heuristic to solve the problem. Algorithm 1 and Algorithm 2 present the two functions of the proposed heuristic algorithm.

Algorithm 1 presents the function *OPTIMISER* that takes the following inputs: binary precedence matrix, a matrix of setup times, and one with the processing times for each task for each robot, the size of the production batches, the number of different tasks involved in the production, and the number of assignable robots. The *OPTIMISER* loops until all tasks are assigned, incrementing the time index by one for each loop. In the loop, the function cycles through all robots for all tasks, checking whether the task in question is available for assignment and afterward checking whether any robots are available to complete a task batch

within the time specified by the time counter updated in the outer loop. If a suitable robot is found, the robot assignment matrix is updated with the time of completion for the batch; the counters for number of assigned tasks, occupancy of the robot, and task readiness are updated as well. When all tasks are assigned a robot, the function returns the assignment matrix of time of completion.

Algorithm 2 presents the *ELIGIBILITY CHECKER* function that receives the same inputs as the *OPTIMISER* function and also the trackers for task number, robot number, the robot assignments, and the task readiness tracker. The function checks if the precedence requirements for the task are met and, if so, proceeds to check if the task can be completed in time by the robot in question; this includes considerations of whether the first product of the batch has its precedence requirements met and whether the last product of the batch will before undergoing the task.

Changing the sequence of products produced is a simple matter of editing the precedence matrix, for example, by requiring the last task of product one to be assigned before assigning the first task of product two. If we wish to consider a small setup time between identical products, this can be included in our processing times matrix. In the case of certain robots being unable to handle a specific assembly task, this can be accounted for by assigning an unreasonably high value to the relevant processing time matrix entry. The proposed MuRALB heuristic algorithm is coded in C++ and examines the performance by implementing it in R using the Rcpp package.

Algorithm 1: Optimiser Function

```
function OPTIMISER (Precedence requirements, Setup times, Processing times
Batch size, No. of robots, No. of tasks)
   while Assigned tasks ! Tasks do Increment time
      for first task,…., last task do
         for first robot,…..., last robot do
            if task is eligible then
               if robot is available at 'time' then
                  Update robot assignment entry to time of completion
                  Increment the 'Assigned tasks' counter
                  Update the robot occupancy
                  Update when the task will be ready for further processing
               end if
            end if
         end for
      end for
   end while
return Robot assignment matrix
end function
```

Algorithm 2: Eligibility Checker Function

```
function ELIGIBILITY CHECKER (task, robot, time, No. of robots, No. of tasks, Robot assignment, Precedence matrix, Task
  readiness tracker, Batch Size, Setup times, Processing times)
   Construct a vector of precedence requirements from the precedence matrix
   Construct a vector of assigned tasks from the robot assignment matrix
   if Not all necessary tasks have been assigned previously then return False
   end if
    if Time to complete task in question by robot in question exceeds value of time
  variable then return False
   end if
 return True
 end function
```

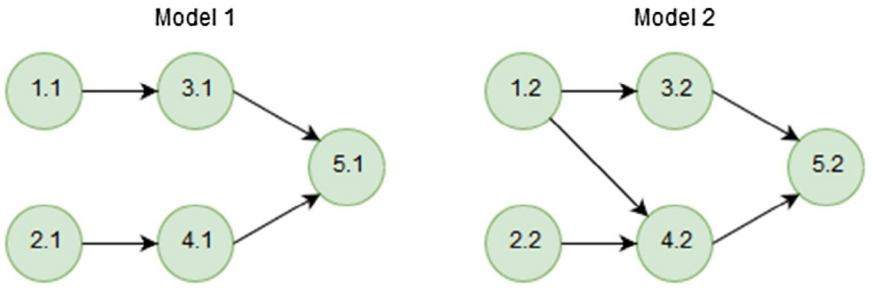

Figure 1. Precedence relations for the illustrative example.

Table 1. Setup time for each robot and task.

	Task				
Robot	1	2	3	4	5
1	4	8	4	8	4
2	7	6	5	8	7
3	8	5	6	3	3

4. Illustrative example

To illustrate the procedure of assigning robots to tasks, and tasks to workstations, we consider two models assembled by three robots. Each model requires five tasks to complete, and their precedence relationships are as shown in Figure 1. All the tasks t are considered unique to account for whichever small differences may arise from handling different products. Thus, no combined precedence graphs are possible. In this example, we consider a fixed batch size of $B = 5$. The data we use for our illustrative example are adapted from the data used by Rabbani et al. (2016). Data for large size problems are generated using R code presented in Appendix 1 since there is no literature available. Table 2 presents the processing times for each task by each robot, and Table 1 shows the setup times for each new task.

Table 3 presents the assignment of batched tasks to robots which Figure 2 illustrates. The yellow blocks represent setup time for the assembly tasks represented as blue blocks which belong to product one. The red blocks represent the setup time for the assembly tasks for product two represented as green blocks. A significant benefit of going forward with the batching of robots is the reduction in time spent reconfiguring the robots between tasks. The temporal benefit of producing in batches decreases when the batch size increases, which is illustrated in Figure 3 in the next section. Figure 3 shows the average production time for one product when using the robot specifications of Tables 1 and 2, dependent on the batch size. The bottom blue line indicates the average production time for one product when disregarding the setup time. As can be understood from Figure 3, the cycle-time benefit of batching production arises from the resulting decrease in setup time, thereby approaching the simple RALB situation already described in (Gao et al., 2009; Levitin et al., 2006). The decrease in time per product is not entirely smooth, which may arise from the reduced flexibility introduced by increasing the batch size. The exact benefit of increasing the batch size depends on the specific setup and production times for the tasks and robots in question, but the general reduction in cycle time per production set is evident.

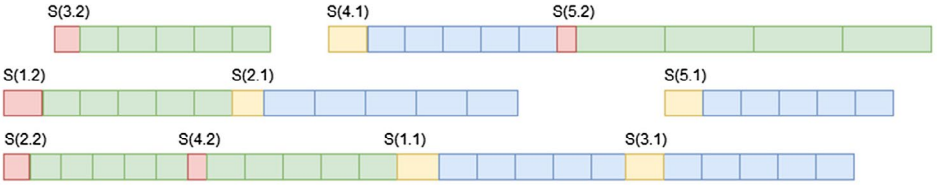

Figure 2. Solution for the illustrative example.

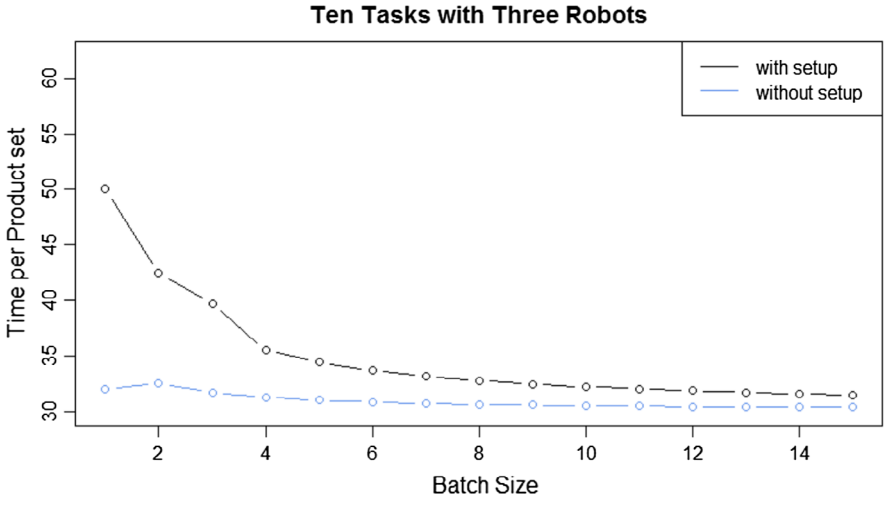

Figure 3. Average production time for one of each product based on batch size.

Table 2. Processing times for the tasks by robots.

	Task model									
Robot	1.1	1.2	2.1	2.2	3.1	3.2	4.1	4.2	5.1	5.2
1	13	7	13	7	14	6	5	6	9	14
2	14	8	10	12	12	6	11	13	6	14
3	7	13	11	5	6	7	7	6	9	14

Table 3. Assignment of tasks to robots based on Algorithm 1.

Robot	Task				Production time
1	3.2 × B	4.1 × B	5.2 × B		172
2	1.2 × B	2.1 × B	5.1 × B		
3	2.2 × B	4.2 × B	1.1 × B	3.1 × B	

5. Experimental study

The computational experiments test the performance of the proposed MuRALB heuristic. Since the literature survey showed there is not much work reported, different datasets of different sizes are generated to test the heuristic. Different datasets ranging from 10 to 50 tasks are generated with different combinations of robots available for the assembly process.

The experiments show how the heuristic performs when the problem size changes. The study proves (refer Figures 4–8) that the proposed heuristic MuRALB approach remains applicable even when the problem size increases. Different sets of precedence relations, setup times, and the processing times used for testing are generated randomly using the statistical programming language R. In the data generation, setup times are considered higher than the production times. Appendix 1 shows the code used for generating Figure 4. Figures 5, 6, and 7 are also generated using the same procedure. When generating the precedence graphs, we require that no batch has tasks from other batches as precedents, the first task has no precedents, the final task has all previous tasks as precedents, and no task has the final task as precedent. The proposed heuristic is tested on different problem sizes such as a 20-task problem, 30-task problem, 40-task problem, and 50-task problem. These problems are generated with different combinations of products and robots used for the assembly. The first problem consists of 20 tasks for two different products assembled by five robots, the second problem consists of 30 tasks for three different products assembled by seven robots, the third problem consists of 40 tasks for three different products assembled by nine robots, and the last problem consists of 50 tasks for four different products assembled by ten robots.

The effect of batch size on production time per product set appears similar across the four production setups, with the benefit of batching decreasing as the batch size increases. This tendency seems reasonable, as the influence of the setup time decreases when the fraction of time spent on setting up becomes smaller relative to the total cycle time. If we were to increase the batch size even further, the average set cycle time would tend toward what it would be if we disregard the setup time, notwithstanding any obstructions that might occur due to the lack of flexibility, as in the case shown in Figure 7 with batch size 4 and 15, Figure 6 with batch size 9, and Figure 5 with batch size 6. Table 4 displays the performance data for batch sizes 1, 4, and 8, with the batch sizes chosen to reflect the greatest variety in the set cycle time. The decrease in set cycle time is substantial, which is related to the choice of using high setup times, but still underlines the prudence of using MuRALB instead of mixed robotic assembly line balancing when striving to minimize cycle time. Table 4 also displays the computational times for an Intel(R) Core(TM) i7–4810MQ CPU at 2.80 GHz.

Table 4. An overview of the performance of the MuRALB heuristic.

Number of tasks	Size of batches	Number of models	Number of robots	Avg.set cycle time	Computational time, ms
10	1	2	3	50	<1
	4			35.5	2
	8			32.75	3
20	1	2	5	792	35
	4			386	69
	8			329	117
30	1	3	7	988	99
	4			535	219
	8			447	357
40	1	3	9	1098	166
	4			539.5	326
	8			430	529
50	1	4	10	1067	263
	4			500.5	508
	8			428	869

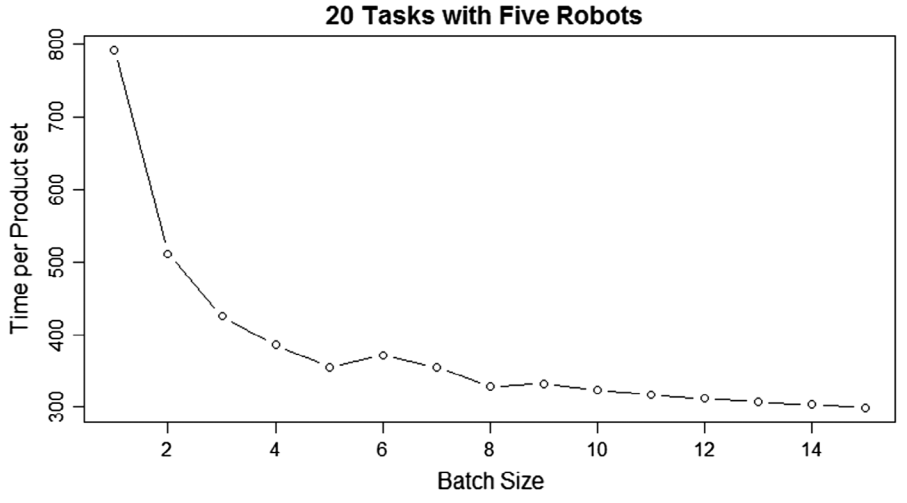

Figure 4. Average production time for 20 tasks, five robots.

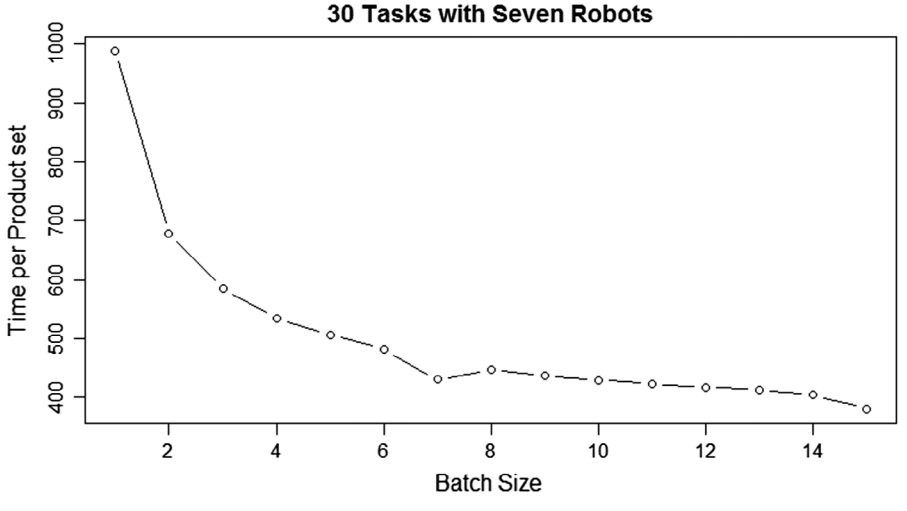

Figure 5. Average production time for 30 tasks, seven robots.

The computational times listed in Table 4 indicate a substantial increase when increasing the size of the problem, which, based on Algorithm 1, is related to the increase in total cycle time, as the algorithm loops for each time unit increase.

6. Conclusion and managerial implications

In this paper, we study the type II MuRALB problem, for which we strive to minimize the total cycle time for the set of different products. The cycle-time minimization is based on reducing the number of times a robot must be configured for a new task on a new product. We present the mathematical framework for the cycle-time minimization and convert it to a conceptual algorithm. This paper proposes a heuristic algorithm for solving the multi-model

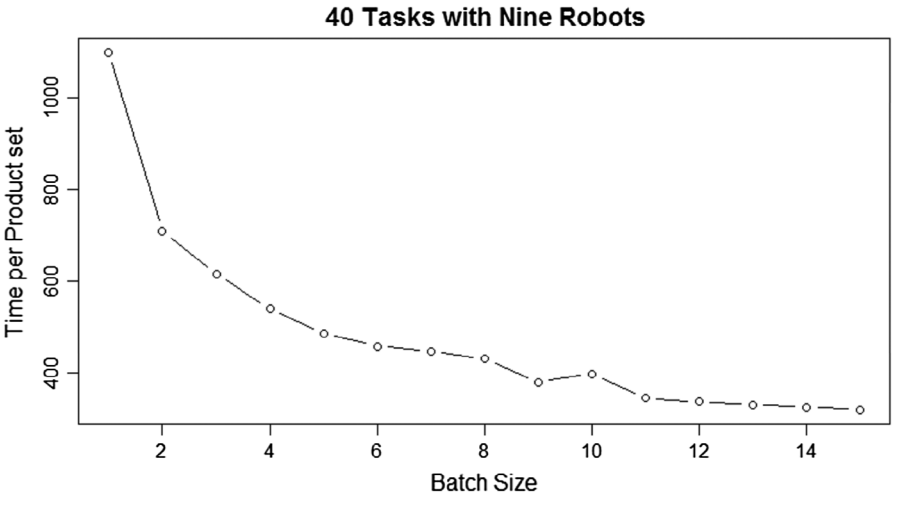

Figure 6. Average production time for 40 tasks, nine robots.

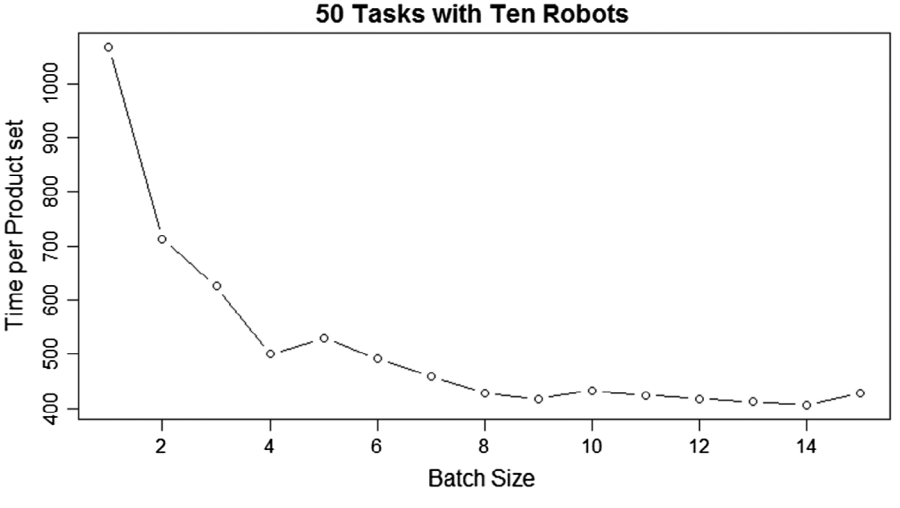

Figure 7. Average production time for 50 tasks, ten robots.

assembly line balancing problem and has reported how the proposed heuristic algorithm works for robotic assembly lines with 10, 20, 30, 40, and 50 tasks with different combinations of batch sizes and robots. The algorithm is implemented in C++ and is applied to solve the considered datasets which are generated randomly. This problem serves to visualize the concept of configuring a robotic assembly line for batched production. The MuRALB heuristic proves to work well on problems beyond illustrative size and provides a significant decrease in the average cycle time for one product set when batching a few tasks. The benefit of batching tasks fades as the batch size increases, which relates to the setup times becoming a smaller fraction of the total cycle time. The MuRALB heuristic performs well while solving large problems within a practical time frame but does not consider either spatial requirements or reduction in idle time for the robots.

Using the MuRALB heuristic algorithm allows production managers to schedule mass production of batches of customized products, which gives rise to better-tailored product lines to specific consumer groups. The drawbacks of implementing a batched production are the increased spatial requirements and the reduced flexibility at the production site. The increase in spatial requirements stems from the reduced flexibility, as the products may tend to move in bulk, dependent on the setup and production times. We recommend the application of metaheuristic algorithms or hybrid optimization approaches (Relich & Pawlewski, 2017; Sitek & Wikarek, 2016) to solve the proposed MuRALB problem, as this may further improve the efficiency of batched production. Moreover, to account for any spatial needs in connection to the batched production, it may prove prudent to add further objectives or constraints to the problem at hand.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Altemeier, S., Helmdach, M., Koberstein, A., & Dangelmaier, W. (2010). Reconfiguration of assembly lines under the influence of high product variety in the automotive industry–a decision support system. *International Journal of Production Research, 48*(21), 6235–6256.

Axsäter, S. (1981). Aggregation of product data for hierarchical production planning. *Operations Research, 29*(4), 744–756.

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solutionapproaches. *International Journal of Production Economics, 142*(2), 259–277.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715.

Boysen, N., Fliedner, M., & Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics, 111*(2), 509–528.

Bukchin, J., Dar-El, E. M., & Rubinovitz, J. (2002). Mixed model assembly line design in a make-to-order environment. *Computers & Industrial Engineering, 41*(4), 405–421.

Çil, Z. A., Mete, S., & Ağpak, K. (2016a). Analysis of the type II robotic mixed-model assembly line balancing problem. *Engineering Optimization, 49*(6), 990–1009.

Çil, Z. A., Mete, S., & Ağpak, K. (2016b). A Goal Programming Approach for Robotic Assembly Line Balancing Problem. *IFAC-PapersOnLine, 49*(12), 938–942.

Erel, E., & Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control, 9*(5), 414–434.

Eriksen, P. S., & Nielsen, P. (2016). Order quantity distributions: Estimating an adequate aggregation horizon. *Management and Production Engineering Review, 7*(3), 39–48.

Gao, J., Sun, L., Wang, L., & Gen, M. (2009). An efficient approach for type II robotic assembly line balancing problems. *Computers & Industrial Engineering, 56*(3), 1065–1080.

Gyulai, D., Kádár, B., Kovács, A., & Monostori, L. (2014). Capacity management for assembly systems with dedicated and reconfigurable resources. *CIRP Annals-Manufacturing Technology, 63*(1), 457–460.

Hosseini Nasab, H., Tavana, M., & Yousefi, M. (2014). A new heuristic algorithm for the planar minimum covering circle problem. *Production & Manufacturing Research, 2*(1), 142–155.

Hu, S. J., Ko, J., Weyand, L., ElMaraghy, H., Lien, T., Koren, Y., … Shpitalni, M. (2011). Assembly system design and operations for product variety. *CIRP Annals-Manufacturing Technology, 60*(2), 715–733.

Jawahar, N., Ponnambalam, S., Sivakumar, K., & Thangadurai, V. (2014). Heuristics for multiobjective optimization of two-sided assembly line systems. *The Scientific World Journal, 2014*, 1–16.

Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*(3), 811–825.

Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (2016). Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Advances in Mechanical Engineering, 8*(9), 1687814016667907.

Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (in press). Mathematical model and metaheuristics for simultaneous balancing and sequencing of a robotic mixed-model assembly line. *Engineering Optimization*. doi:10.1080/0305215X.0302017.1351963

Liao, L.-M. (2014). Construction and comparison of multi-model and mixed-model assembly lines balancing problems with bi-objective. *Journal of Industrial and Production Engineering, 31*(8), 483–490.

Mukund Nilakantan, J., & Ponnambalam, S. (2016). Robotic U-shaped assembly line balancing using particle swarm optimization. *Engineering Optimization, 48*(2), 231–252.

Nilakantan, J. M., Ponnambalam, S., Jawahar, N., & Kanagaraj, G. (2015). Bio-inspired search algorithms to solve robotic assembly line balancing problems. *Neural Computing and Applications, 26*(6), 1379–1393.

Nilakantan, J. M., Li, Z., Tang, Q., & Nielsen, P. (2017). Multi-objective co-operative co-evolutionary algorithm for minimizing carbon footprint and maximizing line efficiency in robotic assembly line systems. *Journal of Cleaner Production, 156*, 124–136.

Nilakantan, M. J., Ponnambalam, S., & Jawahar, N. (2016). Design of energy efficient RAL system using evolutionary algorithms. *Engineering Computations, 33*(2), 580–602.

Rabbani, M., Mousavi, Z., & Farrokhi-Asl, H. (2016). Multi-objective metaheuristics for solving a type II robotic mixed-model assembly line balancing problem. *Journal of Industrial and Production Engineering, 33*(7), 472–484.

Reeves, C. R. (1993). *Modern heuristic techniques for combinatorial problems*. New York, NY: John Wiley & Sons, Inc.

Relich, M., & Pawlewski, P. (2017). A fuzzy weighted average approach for selecting portfolio of new product development projects. *Neurocomputing, 231*, 19–27.

Rubinovitz, J., Bukchin, J., & Lenz, E. (1993). RALB–A heuristic algorithm for design and balancing of robotic assembly lines. *CIRP Annals-Manufacturing Technology, 42*(1), 497–500.

Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*(3), 666–693.

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

Sitek, P., & Wikarek, J. (2016). A hybrid programming framework for modeling and solving constraint satisfaction and optimization problems. *Scientific Programming, 2016*, 1–13.

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology, 73*(9-12), 1665–1694.

Wijngaard, J. (1982). On aggregation in production planning. *Engineering Costs and Production Economics, 6*, 259–265.

van Zante-de Fokkert, J. I., & de Kok, T. G. (1997). The mixed and multi model line balancing problem: A comparison. *European Journal of Operational Research, 100*(3), 399–412.

Appendix 1. Larger Example Data Sets

```
#MuRALB for 20 tasks with Five Robots ####
set. Seed (40)

Precedence10.1<- rbinom (100, c (0, 1), c (0.9, 0.5))
Matrix 10.1 <- matrix (Precedence10.1, nrow= 10, ncol= 10)
for (i in 1:10) {
   Matrix10.1 [i, 1] <- 0
   Matrix10.1 [i, 10] <- 1
   Matrix10.1 [10, i] <- 0
   Matrix10.1 [i, i] <- 0
}
Precedence10.2 <- rbinom (100, c (0, 1), c (0.9, 0.5))
Matrix 10.2 <- matrix (Precedence10.2, nrow= 10, ncol= 10)
for (i in 1:10) {
   Matrix10.2 [i, 1] <- 0
   Matrix10.2 [i, 10] <- 1
   Matrix10.2 [10, i] <- 0
   Matrix10.2 [i, i] <- 0
}
Matrix10.0 <- matrix (0, nrow= 10, ncol= 10)

Precedence2 <- as.vector (rbind (cbind (Matrix10.1, Matrix10.0),
cbind (Matrix 10.0, Matrix 10.2)))
Tasks2 <- 20
Robots2 <- 5
Solution2 <- 1:15
microbenchmark. solution2 <- microbenchmark (for (Batch2 in 1:15) {
   Solution2 [Batch2] <- max (Optimizer (Precedence2, Production2, Setup2, Batch2, Tasks2,
Robots2))
} )
For (Batch2 in 1:15) {
   Solution2 [Batch2] <- max (Optimizer (Precedence2, Produciton2, Setup2, Batch2, Tasks2,
Robots2))
}

Normalized2 <- Solution2 [1:15] / (1:15)

plot (Normalized2, type = "b", main= "20␣tasks ␣ with␣Five␣Robots",
xlab = "Batch␣Size", ylab ="Time ␣ per ␣ product ␣ set " )
```