Robotic Cell Design Through Operation Assignment Integrated with Sequencing of Robot Moves

Hakan Gultekin
Mechanical and Industrial Engineering Department
Sultan Qaboos University
Muscat, Oman
hgultekin@squ.edu.om

Abstract

In this study, we consider the mass production of identical parts consisting of a specific number of operations in a robotic manufacturing cell. The throughput of the cell depends on the robot move sequence, which in turn depends on the assignment of these operations to the machines. Despite this relation, the robot move sequencing and line balancing problems discussed in the literature are not integrated, but they are instead solved sequentially. This study is one of the very first to attempt this integration. We formulated the problem as a mixed-integer linear programming formulation. However, in order to be able to solve real-sized problem instances of this NP-hard problem, a parametric heuristic algorithm is developed. The performance of the developed solution methods is tested with an extensive computational study. The results indicate that the heuristic algorithm performs efficiently.

Keywords
Robotic cell scheduling, line balancing, operation assignment, mathematical programming, heuristics

1. Introduction
The number of manufacturing firms using automated material handling systems is increasing rapidly. Firms that want to be sufficiently flexible to satisfy varying customer needs and be competitive in terms of cost, speed, and quality deploy such systems. Some of the most widely used equipment for material handling are industrial robots. This study considers a widely used application of such robots. A manufacturing cell consisting of a number of machines and a material handling robot that is responsible for loading/unloading the machines and the part transfers between the machines is called a robotic cell. The cell works as a flow shop in which the machines are located serially. Identical parts are assumed to be produced indefinitely. Each part passes through each machine in the same sequence. There is an input buffer that always has parts waiting for processing and an output buffer that always has a place for the produced parts. Other than these two, there are no intermediate buffers between the machines. This study considers the design problem of these robotic cells.

The processing times of the parts on the machines and the sequence of robot moves affect the throughput of the robotic cells, which is the most common objective in the literature. The processing times result from the assignment of the operations to the machines. Most studies assume that the processing times are assumed to be given parameters (see the reviews by Crama et al. 2000, Geismar et al. 2005, and Brauner 2008). Then, the problem is to determine the robot move sequence corresponding to these processing times, which are determined priorly by solving the Assembly Line Balancing (ALB) problem. In ALB, the usual objective is to balance the processing times on the machines. However, as will be shown in the following sections, balanced processing times do not necessarily maximize the throughput rate in robotic cells. Therefore, in this study, we integrate these two problems and solve them simultaneously. In a different framework, Karabatı and Sayın (2003) and Öztürk et al. (2015) integrated the task assignment and model sequencing problems in mixed model assembly lines and showed that solving these problems simultaneously is crucial for effective line management. Similarly, we show that the throughput rate can be increased significantly by integrating the operation assignment and robot move sequencing problems. Note that, the maximization of the throughput rate is equivalent to the minimization of the cycle time, which is defined as the long run average time to produce a single part.

In this study, we assume that the parts to be processed consist of a fixed number of operations to be performed on the machines. The robot move sequence with the minimum cycle time depends on the processing times on the machines, and these processing times are determined by the assignment of the operations to the machines. An operation can only be assigned to one of the machines that is capable of performing it.

This study considers 1-unit robot move cycles consistent with most studies in the literature. In a 1-unit cycle, one part enters the system, each machine is loaded and unloaded once and one part is dropped to the output buffer. 1-unit cycles are not only simple and practical for application in real industrial settings but also proven to be optimal for 2- and 3-machine robotic cells (Sethi et al. 1992, Crama and van de Klundert 1999) and perform efficiently in the remaining cases (Dawande et al. 2005, Brauner 2008, Geismar et al. 2005). As a consequence, the problem is to determine the assignment of operations to the machines and the corresponding 1-unit robot move cycle that jointly minimize the cycle time.

Crama and van de Klundert (1997) developed an $O(m^3)$ time algorithm to find the best 1-unit cycle in an m-machine cell when the processing times are known. We make use of this algorithm to develop a solution method for the current problem. Akturk et al. (2005) and Gultekin et al. (2006) considered the 2-machine robotic cell scheduling problem together with the operational flexibility. This led to an operation assignment problem as if each part has two operations to be assigned to the machines. Gultekin et al. (2010) considered the operation assignment problem in a 2-machine robotic cell. They assumed the machine speeds to be adjustable, which resulted in controllable processing times.

The current problem is closely related with ALB problems, which can be reviewed from the works of Baybars (1986), Becker and Scholl (2006), and Scholl and Becker (2006). The current study is the first time that an ALB problem is considered in a robotic cell. There is another stream of research in which the task allocation problem is solved for assembly lines with multiple processing robots instead of a single material handling robot (see e.g. Gultekin et al. 2016). However, the problem structure is completely different in these two problem types.

It is known that even the most basic form of line balancing problems (called Simple Assembly Line Balancing Problems SALBPs) is NP-hard (Baybars 1986). Therefore, a large number of heuristic procedures has been developed to solve these problems. The largest candidate rule (Moodie and Young 1965), the ranked positional weight method (Helgeson and Birnie 1961), Hoffmann's algorithm (Hoffmann 1963), the COMSOAL method (Arcus 1965), and the immediate update first fit method (Hackman et al. 1989) are some well-known examples of construction-type heuristics that are developed for SALBPs.

Generalized Assembly Line Balancing Problems GALBPs are derived from SALBPs by relaxing or changing one or more of their characteristics. GALBPs include practical constraints such as machines with different capabilities or a different layout of the line (i.e., U-type assembly lines). The existence of the material handling robot and the restriction of the "assignability" of the operations to the machines differ our problem from SALBPs.

In the next section, we introduce the notation used, define the problem, and develop mathematical programming formulations. In Section 3, a heuristic algorithm is developed. In Section 4, an extensive computational for testing the performance of the mathematical models and the heuristic algorithm is presented, and the results are reported. Section 5 is devoted to the concluding remarks and future research directions.

2. Problem Definition and Mathematical Model

We consider an m-machine robotic cell in which identical parts are to be processed indefinitely. Each part has a set of operations ($N = \{1,2, \dots, n\}$) to be performed on the machines. The processing time of operation j is denoted as oj. There are m machines ($M = \{M_1, M_2, \dots, M_n\}$) that are serially located and capable of performing different operations. The 0-1 parameter a_{ij} denotes whether operation j can be assigned to machine i or not. The total processing time of the part on M_i is denoted by p_i, which is the sum of the individual processing times of the operations assigned to that machine. There are precedence relationships between the operations denoted by the 0-1 parameter b_{kj}. If operation k is a predecessor of j then $b_{kj} = 1$. In this case, k must be performed earlier than j either on a former machine or on the same machine.

In a 1-unit cycle, the robot takes one part from the input buffer, loads and unloads each machine exactly once, and drops one part onto the output buffer. In order for this cycle to be repeatable, the starting and final positions of the

robot must be identical. Sethi et al. (1992) proved that there are $m!$ 1-unit cycles in an m machine cell. In order to represent these cycles, Crama and van de Klundert (1997) defined a robot activity A_i as the set of robot moves consisting of unloading M_i, transporting the part to M_{i+1} , and loading this machine. The input and output buffers are denoted by M_0 and M_{m+1}, respectively. In a 1-unit cycle, the robot performs each of these activities exactly once. Each permutation of the activities yields a feasible 1-unit cycle. Without loss of generality, it may be assumed that all cycles start with the activity A_0. We denote the loading/unloading time of a machine by ε and the transportation time of the robot between two adjacent machines by δ. The transportation time is additive. As a consequence, the transportation time between any two machines, M_i and M_h, is calculated as $|i-h|\delta$. w_i denotes the necessary waiting time of the robot for M_i to complete processing before unloading it. After loading a part to M_i, if the robot waits in front of the machine until the processing is completed, then the waiting time is equal to the processing time of the machine, $w_i = p_i$. Instead of waiting in front of the machine, if the robot performs some other activities and later returns back to the machine to unload it, the waiting time is equal to the remaining processing time of the part. If the processing completes before the robot arrives in front of the machine, the part waits on the machine, and the robot unloads the part as soon as it arrives. In this case, the waiting time is zero.

In the traditional ALB literature, the optimal operation assignment is obtained when the machine processing times are balanced. This is achieved by minimizing the largest processing time over all machines. However, in a robotic cell a solution with balanced processing times may not provide the optimal cycle time.

We develop an MIP formulation that solves the operation assignment and robot move sequencing problems simultaneously. To achieve this, the activities ($A = \{A_0, A_1, \dots, A_m\}$) are assigned to the set of respective positions ($P = \{1,2, \dots, m+1\}$). As already mentioned, all cycles are assumed to start with the activity A_0. Therefore, the remaining m activities are assigned to the remaining m positions. The following MIP decides the sequence of these activities by assigning them to the positions and determines the assignment of the activities to the machines that jointly minimize the cycle time. For this purpose, we define the following decision variables:

$$y_{pk} = \begin{cases} 1, & \text{If robot activity } k \text{ is assigned to position } p \\ 0, & \text{Otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{If operation } j \text{ is assigned to machine } i \\ 0, & \text{Otherwise} \end{cases}$$

$T_k =$ starting time of activity k

$C =$ cycle time

$$Min \quad C \tag{1}$$

$$\text{Subject To:} \quad C \geq T_k + 2\varepsilon + (k+2)\delta, \quad k \in A \tag{2}$$

$$\sum_{k \in A} y_{pk} = 1, \quad p \in P \tag{3}$$

$$\sum_{p \in P} y_{pk} = 1, \quad k \in A \tag{4}$$

$$\sum_{i \in M} x_{ij} = 1 \quad j \in N \tag{5}$$

$$x_{ij} \leq a_{ij}, \quad i \in M, j \in N \tag{6}$$

$$x_{il} b_{il} \leq \sum_{h \leq i} x_{hj}, \quad i \in M, \quad l, j \in N \tag{7}$$

$$T_k \geq T_{k-1} + 2\varepsilon + \delta + \sum_{j \in N} x_{kj} o_j - B\left(1 - y_{p(k-1)} + \sum_{h=1}^{p-1} y_{hk}\right) \qquad k \in A\backslash\{0\}, \qquad p \in P \tag{8}$$

$$T_k + C \geq T_{k-1} + 2\varepsilon + \delta + \sum_{j \in N} x_{kj} o_j - B\left(1 - y_{p(k-1)} + \sum_{h=p+1}^{m+1} y_{hk}\right) \qquad k \in A\backslash\{0\}, \qquad p \in P \tag{9}$$

$$T_k \geq T_l + 2\varepsilon + (|l + 1 - k| + 1)\delta - B\left(2 - y_{pk} - y_{(p-1)l}\right) \qquad k, l \in A: k \neq l, \qquad p \in P\backslash\{1\} \tag{10}$$

$$x_{01} = 1 \tag{11}$$

$$T_0 = 0 \tag{12}$$

$$x_{ij}, y_{pk} \in \{0,1\} \qquad i \in M, j \in N, p \in P, k \in A \tag{13}$$

$$T_k, C \qquad i \in M, k \in A \tag{14}$$

In this model Equation (1) is the objective function which minimizes the cycle time. Constraint (2) satisfies the cycle time value, which results from the linearization of the objective function. Constraint (3) and (4) determine the sequence of the activities by ensuring that one activity is assigned to each position and each activity is assigned to one position, respectively. Constraint (5) satisfies the assignment of operations by enforcing each operation to be assigned to only one machine. Constraint (6) ensures that every operation is assigned to a machine that is able to undertake it. Constraint (7) satisfies the precedence relationships between the operations.

The activity A_k cannot start unless the processing of the part is completed on M_k. The processing on M_k starts after A_{k-1} is completed. Since we are considering cyclic scheduling, in an activity sequence either A_{k-1} precedes A_k or vice versa. These two cases are handled separately in Constraints (8) and (9). Additionally, A_k cannot start unless the robot is ready in front of M_k as formulated in Constraint (10). Constraints (11) and (12) ensure that the activity in the first position is A_0, and it starts at time 0. Constraint (13) defines the binary decision variables and Constraint (14) defines the sign restrictions.

We know that even simple ALBPs are known to be NP-hard (Baybars 1986), and the robotic operation allocation is NP-hard even for 2-machine cells (Gultekin et al. 2010), which means that the considered problem is also NP-hard. Therefore, we develop a heuristic algorithm in the next section.

3. Heuristic Algorithm

In this section, we develop a heuristic algorithm for the problem described in Section 2. As mentioned earlier, Crama and van de Klundert (1997) developed a dynamic programming algorithm that determines the optimal 1-unit cycle in an m-machine robotic cell when the machine processing times are known. In order to make use of this efficient algorithm, the operation assignment must be determined. However, the traditional ALB algorithms that try to balance the processing times need not be optimal for the current problem. Additionally, there is no specific characteristic for the optimal processing time values. Some robot cycles work well when the processing times are balanced, whereas some others may require different assignment schemes and different characteristics. Since there is a large number of 1-unit cycles, it is not possible to characterize the optimal processing time values for each of them. Therefore, we develop a heuristic that searches different assignments. For each assignment, the corresponding best 1-unit cycle is determined using the Crama and van de Klundert (1997) algorithm.

In order to assign the operations, a priority-based method is used. Such methods are well known and widely used in the ALB literature. The operations are ranked according to their priorities. At each iteration the operation with the highest priority that has no unassigned predecessors is chosen and assigned to the first available machine. In this study, the operations are prioritized according to their processing times as in the largest candidate rule (Moodie and Young 1965). In such methods, an upper bound is calculated for the machine processing times, and the operations are assigned

so that the processing times do not exceed this upper bound. Since the objective is to balance all processing times in classical ALB, the same upper bound is used for all machines. However, in order to allow for variation in the processing times and to systemically search for different assignment schemes, we use different upper bounds for each machine in this study. The alteration of these upper bounds results in different assignments and thus different processing time values. Therefore, solutions with unbalanced processing times can also be searched.

Let U_i be the upper bound of the processing time on machine i and R be the average processing time per machine: $R = \sum_j \frac{o_j}{m}$. In order to systematically search the solution space, we calculated the upper bounds on the machines using the following parametric sinusoidal function:

$$\boldsymbol{U_i = R\left(1 + \left(\frac{\alpha}{\alpha_{up} - \alpha_{lo}}\right)\sin\left(\beta + i\gamma\right)\right)}$$

In this equation, α, β, and γ are parameters that are used to search for different solutions. Each of these parameters has their own lower, upper, and increment values. For example, if the lower, upper, and increment values for α are $\alpha_{lo} = -2$, $\alpha_{up} = 2$, and $\alpha_{inc} = 1$, respectively, then all of the values in the set $\{-2,-1,0,1,2\}$ are utilized one at a time during the algorithm. Similarly, different values are utilized for β and γ. The best lower, upper, and increment values for each of these parameters are determined through a computational study.

Such a parametric function can be used to search for many different realizations of processing times, as can be seen in Error! Reference source not found.. By using different α, β, and γ values, each of the realizations depicted in this figure can be attained. The *sin* function fluctuates in the range [-1, 1]. As a consequence, the values of U_i fluctuate around the average processing time per machine, R. The α parameter refers to the amount of deviation in the processing time of each machine from the average processing time (vertical deviation). β determines how the sinusoidal shape starts with the first machine(See Figure 1), whereas γ refers to the deviation between the upper bounds of adjacent machines (horizontal deviation). If γ is small, adjacent machines will have closer upper bound values.

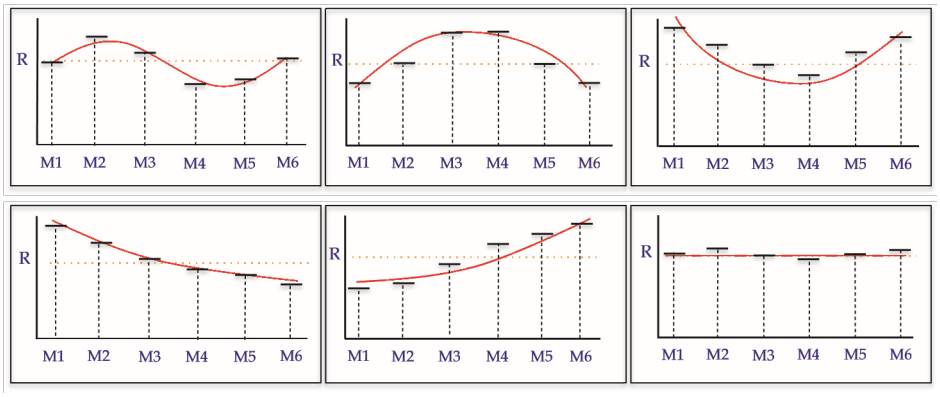

Figure 1. Examples of some sinusoidal upper bounds and possible processing time realizations

The developed heuristic first calculates the U_i values with one of the methods mentioned above. The initial assignment of operations is determined using these upper bounds. The best 1-unit cycle corresponding to the resulting processing times is determined using the dynamic programming algorithm of Crama and van de Klundert (1997). Then, the upper bounds are altered, and the procedure is repeated. The upper bounds are calculated by altering the values of α, β, and γ one at a time. The algorithm ends after all of the predetermined values for α, β, and γ are utilized. The steps of the algorithm can be seen in Figure 2.

In this algorithm, the minimum cycle time is set to a very large value, the operations are ordered with respect to their processing times, and the initial assignments are reset. First, the α, β, and γ values are set and the corresponding upper bounds of the processing times for all machines are determined. Then, by considering each machine in the sequence

separately, the ordered operations are assigned to the machines one at a time by checking the following conditions. The considered operation must be unassigned, it must not have any unassigned predecessors, it must be assignable to the considered machine, and either the processing time of the machine must not exceed its upper bound or this machine must be the last one in the sequence that this operation can be assigned to. After the assignments are made, the corresponding best 1-unit cycle, which is denoted as RMC in Figure 2, is determined using the Crama and van de Klundert (1997) algorithm. Then, the minimum cycle time is updated. After all values of α, β, and γ are utilized, the best cycle time is returned as a result of the algorithm.

To determine the ranges (lo, up) and increment values for the α, β, and γ parameters, a computational study of a 4-machine robotic cell is performed. For these tests, we used the SALBP-2 test problems from the online library for ALB problems (www.alb.mansci.de). In this problem set, there are a total of 17 different instances. For each problem, the number of operations, their processing times, and the precedence relations are provided. However, the assignability of the operations, ε, and the δ values are specific to the current problem and do not exist in this library. We generated these parameters randomly. We tested different range and increment combinations for the α, β, and γ parameters. All 17 problems were solved with each combination. Among the large set of combinations, the most outstanding ones are listed in Table 1 in which the lower and upper values for the parameters are represented as a range (lo, up). The results attained from each combination are compared with the optimal solutions of the mathematical model. Among these nine alternatives, two of them are highlighted in boldface. Combination 1 has the shortest CPU time (0.33 s with a 4.94% error), and combination 9 has the smallest average percent error (2.06% with a 0.43 s CPU time). As expected, the solution time increases as the range increases and the increment value decreases. Since the problem can be considered as a design problem, we decide to use combination 9 that minimizes the average deviation and has a fairly good solution time performance.

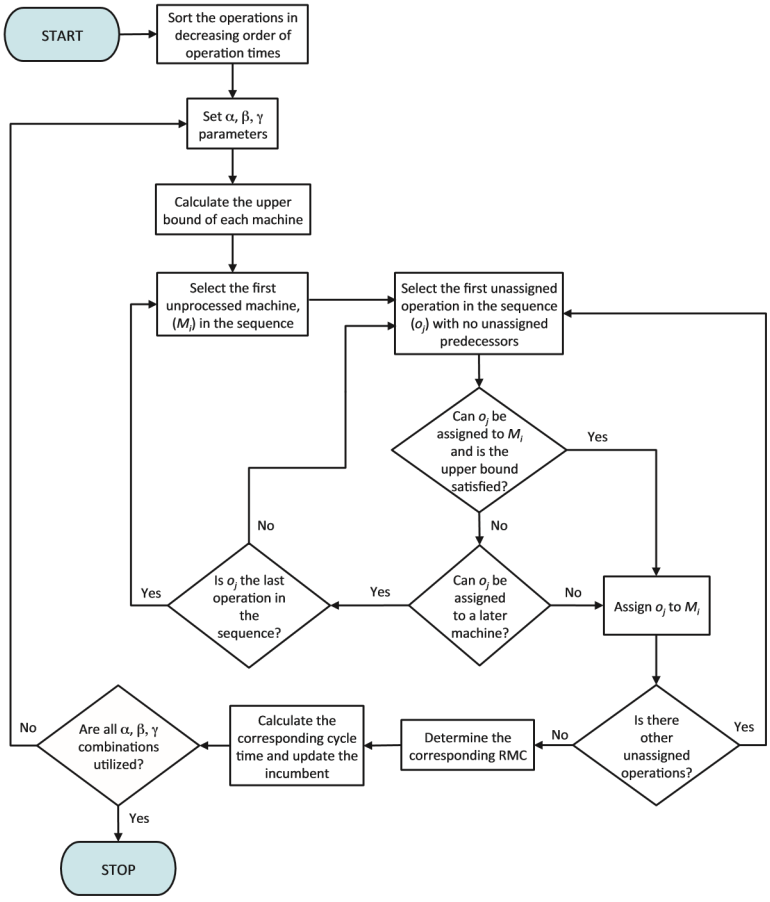

Figure 2. Flowchart of the Robotic Cell Line Balancing Algorithm (ROCLIB)

4. Computational Study

In order to test the performance of the MIP formulation and the ROCLIB heuristic, we performed an extensive computational study. For this purpose, we used the 17 test problems in the SALBP-2 test set for ALB problems (www.alb.mansci.de). Table 2 provides the number of operations in each of these test problems. We considered 5, 7, 10, 20, and 30 machine robotic cells. We used all 17 problems with 5, 7, and 10 machines. However, for larger number of machines, in order the problem to be meaningful, we used the problems that have more than 70 operations. Therefore, a total of 8 problems are considered for 20 and 30 machine instances (we used one of the two problems with 89 operations and 148 operations). ε, δ, and the assignability matrix, a_{ij}, are specific to this study and do not exist in the test problems. Therefore, we generated them randomly. Since all machines are loaded and unloaded exactly once in a 1-unit cycle, the ε parameter does not have a specific effect on the optimal 1-unit cycle and performance. Therefore, ε is not included in the experimental design. In order to generate the assignability matrices, we used f as a parameter to indicate the ratio of the number of ones to the total number of elements in the matrix, as in the work by (Dar-El 1975). We considered two cases for the assignability; one is full assignability where $f = 1$, and the other is partial assignability where f is less than 1. We randomly generated the f values for the partial assignability case from the interval [0.3, 0.4]. The parameters used in the experimental design are listed in Table 3.

Table 1. Outstanding parameter combinations used in the calibration study

	α		β		γ	
Combination	Range	Inc.	Range	Inc.	Range	Inc.
1	**(3, 13)**	**1**	**(-40, 200)**	**40**	**(15, 120)**	**15**
2	(5, 20)	1	(-40, 200)	40	(15, 120)	15
3	(3, 13)	1	(-40, 240)	20	(15, 120)	15
4	(3, 13)	1	(-45, 225)	45	(15, 120)	15
5	(3, 13)	1	(-40, 200)	40	(10, 170)	10
6	(3, 13)	1	(-40, 200)	40	(15, 90)	15
7	(3, 13)	1	(-40, 200)	40	(15, 165)	15
8	(3, 13)	1	(-40, 200)	40	(15, 345)	15
9	**(5, 14)**	**1**	**(-40, 200)**	**40**	**(0, 180)**	**15**

Table 2. Number of operations in each the test problem

Problem No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
# of Operations	29	35	45	32	89	30	70	83	111	53	58	75	89	94	148	148	297

Table 3. Design parameters for the experimental study

Levels	m	δ	f
L1	5	$U\left(max\left\{0, \frac{R}{100(m-2)}\right\}, \frac{R}{50(m-2)}\right)$	1
L2	7	$U\left(max\left\{0, \frac{R}{50(m-2)}\right\}, \frac{R}{25(m-2)}\right)$	$\in [0.3, 0.4]$
L3	10	$U\left(max\left\{0, \frac{R}{25(m-2)}\right\}, \frac{R}{15(m-2)}\right)$	
L4	20		
L5	30		

Each of the 17 problems is solved with each parameter combination, and five replications are performed for each problem for 5, 7, and 10 machines. This makes 17 × 3 × 3 × 2 × 5 = 1530 problem instances. For 20 and 30 machine cases we considered 8 problems with the same parameter combinations that makes 480 instances. In total, 2010 problem instances are generated. The mathematical models are solved using the GAMS CPLEX 12.6 solver. The heuristic algorithm is coded in C++. Both the MIP model and the heuristic algorithm are solved on a computer with an Intel Xeon 2 × 2.4 GHz CPU and 12 GB of RAM. The MIP model is executed with a time limit of 3 h. However, for 20 and 30 machine instances, even a feasible integer solution could not be found within 3 h time limit. Therefore, in order to have a comparison basis for the heuristics, we fed the solution provided by the ROCLIB heuristic into the CPLEX as an initial solution, and run the mathematical model with an additional 15 minutes time limit. However, the optimal solution could not be attained in none of the instances in this case either. CPLEX could improve the provided initial solution in several instances but output the same initial solution in the rest of the cases.

Table 4 summarizes the results of all runs. This table reports the optimality gap, CPU times, and the number of instances in which the optimal solution is found by ROCLIB and MIP. When the number of machines is 5 or 7, MIP could find the optimal solutions for all instances. When m = 10, MIP could not find the optimal solution in seven instances in 3-h time limit. The average gap for MIP is 0.098% in 510 instances. However, the maximum gap is 14.5%. When the number of machines is 20 or 30, MIP could not find the optimal solution in none of the instances (TL denotes Time Limit in this table). The average and maximum gaps are approximately 20% and 35% in these cases when CPLEX starts with an initial solution.

Table 4. Summary of the results

	Gap (%)		# of optimal solutions		CPU Time (s)	
m	MIP	ROCLIB	MIP	ROCLIB	MIP	ROCLIB
5	0	1.35	510	328	33.88	0.44
7	0	2	510	238	61.31	0.46
10	0.098	5.17	503	55	613.44	0.49
20	22.37	1.57	0	116	TL	0.82
30	19.35	1.34	0	86	TL	1.20

TL: Time limit

The average gap for the ROCLIB heuristic appeared to be 1.35%, 2%, and 5.17% for 5, 7, and 10 machines, respectively. The maximum deviations are 14.9%, 17.5%, and 15.6%, respectively. In only 15 of the 510 instances, the gap for ROCLIB was greater than 10% when m = 5. This number is 3 for m = 7 and 15 for m = 10. ROCLIB found the optimal solutions in 328, 238, and 55 instances for each these machines. For 20 and 30 machine instances the average gaps are 1.57% and 1.34%, and the maximum gaps are 11.9% and 10.93%. The number of instances in which the gap was greater than 10% is 4 for 20 machines and it is 1 for 30 machines. In 116 instances of 20 machines and 86 instances of 30 machines the CPLEX could not improve the ROCLIB solution.

The CPU time of the MIP increases drastically as the number of machines increases from 7 to 10. In all instances of 20 and 30 machines the mathematical models stopped because of time limit. On the other hand, the CPU time for ROCLIB increases very slightly with respect to the number of machines.

Table 5 lists the results with respect to different values of the f parameter. It can be seen that the solution times of MIP are larger when f is small. All seven instances in which MIP stopped because of the time limit when $m = 10$ occurred when f is small. From this table, it is evident that the solution time and quality for ROCLIB do not depend on the f parameter.

Table 5. Results with respect to different δ values

		Gap (%)		# of optimal solutions		CPU Time (s)	
m	*δ*	MIP	ROCLIB	MIP	ROCLIB	MIP	ROCLIB
5	L1	0	2.75	170	83	70.51	0.44
	L2	0	0.79	170	105	29.2	0.44
	L3	0	0.5	170	140	1.94	0.44
7	L1	0	1.95	170	81	58.57	0.46
	L2	0	2.69	170	67	62.63	0.46
	L3	0	1.34	170	90	62.74	0.45
10	L1	0.296	4.07	163	40	1194.91	0.50
	L2	0	5.99	170	6	363.77	0.49
	L3	0	5.45	170	9	281.65	0.46
20	L1	25.46	1.06	0	50	TL	0.85
	L2	23.94	1.30	0	34	TL	0.82
	L3	17.71	2.31	0	32	TL	0.79
30	L1	22.47	1.44	0	24	TL	1.26
	L2	21.29	1.28	0	29	TL	1.19
	L3	14.28	1.30	0	33	TL	1.13

Table 6 provides the deviation of ROCLIB from the optimal solution for each of the 17 instances in the test set. For 20 and 30 machines 8 problems are considered as explained before. Although the deviation depends on the number of machines, there is no direct correlation between the solution quality and the number of operations in the instance. The average solution time for each of the 17 test problems are also provided in the same table. It can be argued that the solution time increases linearly with respect to the number of operations. The number of machines have a little effect on the solution time.

5. Conclusion

In this study, we considered a robotic cell design problem that includes line balancing and robot move sequencing for the first time to the best of the author's knowledge. We developed a mathematical programming-based exact solution procedure and a heuristic algorithm to solve this NP-hard problem. We calibrated the parameters of the heuristic that uses a sinusoidal function through an initial computational study. Then we tested the performance of the solution procedures through an extensive computational study. The results suggest that, when the number of machines is smaller, the exact procedures can determine the optimal solutions. However, for large values of m, the solution time of MIP increases drastically. The developed ROCLIB heuristic provides high-quality solutions within reasonable CPU times.

As a future study, instead of identical parts, mixed or multi-model lines can be considered, where the part sequencing must also be solved. In the current study, the number of machines is assumed to be given, and the objective was to minimize the cycle time (SALBP-2). As a future research direction, the problem can be changed to minimize the number of machines subject to a given cycle time (SALBP-1). Furthermore, in the current study, the robot is assumed to have a single gripper that can hold a single part at any instance. Another future research direction is to consider a dual-gripper robot that can hold two parts simultaneously (see, e.g. Gultekin et al. 2017). The productivity gain that can be attained by using a dual-gripper robot instead of a single-gripper robot can be analyzed so that the additional investment cost of using a dual-gripper robot can be justified. Another future research is to consider assembly line balancing problem together with robot speed optimization as considered in Gürel et al. (2019), and Gultekin et al. (2021).

Table 6. Performance of ROCLIB for all test problems

# of Operations	*m:*	Gap (%)					CPU Time (s)				
		5	7	10	20	30	5	7	10	20	30
29		2.62	1.62	4.63	-	-	0.08	0.10	0.13	-	-
30		2.18	2.15	4.72	-	-	0.10	0.12	0.14	-	-
32		1.21	1.66	6.42	-	-	0.14	0.16	0.18	-	-
35		1.60	2.87	4.89	-	-	0.09	0.11	0.13	-	-
45		1.91	1.89	6.77	-	-	0.32	0.34	0.37	-	-
53		0.27	1.38	4.72	-	-	0.08	0.10	0.13	-	-
58		2.43	2.62	5.64	-	-	0.24	0.27	0.28	-	-
70		1.37	2.49	4.83	1.71	1.08	0.31	0.34	0.34	0.39	0.67
75		0.52	2.23	5.34	2.04	0.33	0.53	0.57	0.56	0.41	0.80
83		0.64	1.89	4.56	1.42	1.23	0.17	0.20	0.22	0.46	0.74
89		1.70	3.13	5.23	1.02	1.09	0.18	0.21	0.24	0.49	0.85
89		1.40	1.30	4.60	-	-	0.24	0.27	0.29	-	-
94		1.15	1.95	5.66	0.52	1.40	0.35	0.38	0.41	0.54	0.82
111		0.67	1.52	5.46	1.53	1.51	0.40	0.43	0.46	0.67	0.99
148		1.38	1.70	4.82	2.04	2.53	0.91	0.83	0.88	0.98	1.37
148		1.27	1.67	4.96	-	-	0.87	0.79	0.84	-	-
297		0.63	1.84	4.66	2.17	1.53	2.44	2.54	2.64	2.63	3.32

References

Akturk, M. S., Gultekin, H., and Karasan,O.E., Robotic Cell Scheduling with Operational Flexibility, *Discrete Applied Mathematics*, vol. 145, no. 3, pp. 334-48, 2005.

Arcus, A. L., A Computer Method of Sequencing Operations for Assembly Lines, *International Journal of Production Research*, vol. 4, no. 4, pp. 259–77, 1965.

Baybars, I., A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem, *Management Science*, vol. 32, pp. 909–32, 1986.

Becker, C., and Scholl, A., A Survey on Problems and Methods in Generalized Assembly Line Balancing, *European Journal of Operational Research*, vol. 168, no. 3, pp. 694–715, 2006.

Brauner, N., Identical Part Production in Cyclic Robotic Cells: Concepts, Overview and Open Questions, *Discrete Applied Mathematics*, vol. 156, no. 13, pp. 2480–92, 2008.

Crama, Y., Kats, V., van de Klundert, J., and Levner, E., Cyclic Scheduling in Robotic Flowshops, *Annals of Operations Research*, vol. 96, no. 1–4, pp. 97–124, 2000.

Crama, Y., and van de Klundert, J., Cyclic Scheduling of Identical Parts in a Robotic Cell, *Operations Research*, vol. 45, no. 6, pp.952–65, 1997.

Crama, Y., and van de Klundert, J., Cyclic Scheduling in 3-Machine Robotic Flow Shops, *Journal of Scheduling*, vol. 2, no. 1, pp. 35–54, 1999.

Dar-El, E. M., Solving Large Single-Model Assembly Line Balancing Problems-A Comparative Study, *AIIE Transactions*, vol. 7, no. 3, pp. 302–310, 1975.

Dawande, M., Geismar, H. N., Sethi, S. P., and Sriskandarajah, C., Sequencing and Scheduling in Robotic Cells: Recent Developments, *Journal of Scheduling*, vol. 8, no. 5, pp. 387–426, 2005.

Geismar, H. N., Dawande, M., and Sriskandarajah, C., Approximation Algorithms for K-Unit Cyclic Solutions in Robotic Cells, *European Journal of Operational Research*, vol. 162, no. 2, pp. 291–309, 2005.

Geismar, H. N., Dawande, M., and Sethi, S. P, Dominance of Cyclic Solutions and Challenges in the Scheduling of Robotic Cells, *SIAM Review*, vol. 47, no. 4, pp. 709–21, 2005.

Gultekin, H., Akturk, M. S., and Karasan,O.E., Cyclic Scheduling of a 2-Machine Robotic Cell with Tooling Constraints, *European Journal of Operational Research*, vol. 174, no. 2, pp.777–96, 2006.

Gultekin, H., Akturk, M. S., and Karasan,O.E., Bicriteria Robotic Operation Allocation in a Flexible Manufacturing Cell, *Computers & Operations Research*, vol. 37, no. 4, pp. 779–89, 2010.

Gultekin, H., Dalgıç, Ö.O., Akturk, M.S., Pure cycles in two-machine dual-gripper robotic cells, *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 121-131, 2017.

Gultekin, H., Gürel, S., Taspinar, R., Bicriteria scheduling of a material handling robot in an m-machine cell to minimize the energy consumption of the robot and the cycle time, *Robotics and Computer-Integrated Manufacturing*, vol. 72, 102207, 2021.

Gultekin, H., Tula, A., and Akturk, M.S, Automated robotic assembly line design with unavailability periods and tool changes, *European Journal of Industrial Engineering,* vol. 10, no. 4, pp. 499-526, 2016.

Gürel, S., Gultekin, H., Eghbal Akhlaghi, V., Energy conscious scheduling of a material handling robot in a manufacturing cell, *Robotics and Computer-Integrated Manufacturing,* vol. 58, pp. 97-108, 2019.

Hackman, S. T., Magazine, M. J., and Wee, T. S, Fast, Effective Algorithms for Simple Assembly Line Balancing Problems, *Operations Research*, vol. 37, no. 6, pp. 916–24, 1989.

Helgeson, W. B., and Birnie, D. P., Assembly Line Balancing Using Ranked Positional Weight Technique, *Journal of Industrial Engineering*, vol. 12, pp. 394–98, 1961.

Hoffmann, T. R., Assembly Line Balancing with a Precedence Matrix, *Management Science*, vol. 9, no. 4, pp. 551–562, 1963.

Karabatı, S., and Sayın, S., Assembly Line Balancing in a Mixed-Model Sequencing Environment with Synchronous Transfers, *European Journal of Operational Research*, vol. 149, no. 2, pp. 417–429, 2003.

Moodie, C. L., and Young, H.H., A Heuristic Method of Assembly Line Balancing for Assumptions of Constant or Variable Work Element Times, *Journal of Industrial Engineering*, vol. 16, no. 3, pp. 23–29, 1965.

Öztürk, C., Tunalı, S., Hnich, B., and Örnek, A.. Cyclic Scheduling of Flexible Mixed Model Assembly Lines with Parallel Stations, *Journal of Manufacturing Systems*, vol. 36 pp. 147–58, 2015.

Scholl, A., and Becker. C., State-of-the-Art Exact and Heuristic Solution Procedures for Simple Assembly Line Balancing, *European Journal of Operational Research*, vol. 168, no. 3, pp. 666–93, 2006.

Sethi, S. P., Sriskandarajah, C., Sorger, G., and Kubiak, W., Sequencing of Parts and Robot Moves in a Robotic Cell, *International Journal of Flexible Manufacturing Systems*, vol. 4, pp. 331–358, 1992.

Biography

Hakan Gultekin is an Associate Professor at the Department of Mechanical and Industrial Engineering, Sultan Qaboos University, Muscat, Oman since 2018. He holds BSc, MSc, and Ph.D. degrees on Industrial Engineering from Bilkent University, Turkey. He has been to the University of Liege as a postdoc researcher before joining TOBB University of Economics and Technology, Turkey in 2007. He worked there as an assistant professor between 2007-2013 and as an associate professor between 2013-2018. His research interests include scheduling, optimization modeling, and exact and heuristic algorithm development, especially for problems arising in modern manufacturing systems, energy systems, transportation and logistics, and wireless sensor networks.