Computers & Industrial Engineering 100 (2016) 72–87

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

# Robot move sequence determining and multiple part-type scheduling in hybrid flexible flow shop robotic cells

G. Didem Batur [a,*], Serpil Erol [a], Oya Ekin Karasan [b]

[a] Department of Industrial Engineering, Gazi University, 06570 Ankara, Turkey
[b] Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey

ARTICLE INFO

*Article history:*
Received 4 March 2015
Received in revised form 10 August 2016
Accepted 11 August 2016
Available online 17 August 2016

*Keywords:*
Parallel machine scheduling
Hybrid flow shops
Robotic systems
Mathematical modeling
Simulated annealing

ABSTRACT

We focus on the scheduling problem arising in hybrid flexible flow shops which repeatedly produce a set of multiple part-types and where the transportation of the parts between the machines is performed by a robot. The cycle time of the cell is affected by the robot move sequence, part/machine assignments and part sequences. In a hybrid flexible flow shop in which there exist one machine in the first and two machines in the second stage, the problem of determining the best cycle time is modeled as a traveling salesman problem. In order to provide a solution methodology for realistic problem instances, a Simulated Annealing based heuristic is constructed and the problem is solved using two different neighborhood structures. The results are also compared against an effective proposed lower bound value.



## 1. Introduction

The point of origin of this research is the increase in the level of automation in manufacturing industries. As pointed out by Kamoun, Hall, and Sriskandarajah (1999), since setup times are reduced to improve flexibility, material handling time and cost become bottleneck, and efficient material handling becomes very important. A robotic cell is a manufacturing cell which consists of a number of machines and a material handling robot. In such systems, a part is taken from the input buffer, carried to the related machine and left to the output buffer after its processing is completed.

Robots are commonly used in many different cell formations. In this study, we focus on the robot usage in hybrid flow shops (HFSs). In a classical flow shop, all jobs are processed by the same set of machines in a linear fashion, from the first to the last stage and one machine performs all the processing for each stage. In order to extend the capacity of a single stage, additional parallel machines may be purchased. This extension of a flow shop to allow multiple (usually identical) machines in stages transforms the flow shop into a HFS (Kurz & Aşkın, 2003).

In the literature there are many studies under the interrelated topics of 'flow shops with parallel machines', 'hybrid flow shops' and 'flexible flow shops'. These research fields contain both similar and different aspects. The studies on hybrid flow shops (HFSs) usually focus on non-identical jobs and identical parallel machines, whereas the interest of studies on flow shops with parallel machines is in the identical job and uniform parallel machine environment (Dessouky, Dessouky, & Verma, 1998). Hybrid flexible flow shop problem which is the focus of this study, is obtained from classical flow shop with parallel machines problem by introducing a few specific additional assumptions (Nowicki & Smutnicki, 1998). In hybrid cells, it is possible to produce parts of different types and a job might skip any number of stages provided it is processed in at least one of them, whereas, in flexible flow shops, all jobs are processed following the same production flow: stage 1,stage 2,..., stage $m$ (Kurz & Aşkın, 2003).

Though the underlying optimization problems in a HFS are challenging, they have received a lot of attention in the literature due to the practical relevance of the inherent problems. Vignier, Billaut, and Proust (2010), Linn and Zhang (1999), Wang (2005) and Quadt and Kuhn (2007), and more recently Ruiz and Vazquez-Rodriguez (2010) present reviews on HFS problems.

Under the topics of hybrid flow shops or flexible flow lines there are many studies considering setup operations which are similar to the robot operations considered in our study.

Yaurima, Burtseva, and Tchernykh (2009) focus on hybrid flow shops with unrelated machines and sequence-dependent setup time. Taking the availability constraints and limited buffers into account, they present a genetic algorithm. Jabbarizadeh, Zandieh, and Talebi (2009) also consider sequence-dependent setup times and machine availability constraints on hybrid flexible flow shops.

* Corresponding author.
*E-mail address:* dbatur@gazi.edu.tr (G.D. Batur).

http://dx.doi.org/10.1016/j.cie.2016.08.006
0360-8352/

They propose heuristic methods and present computational experiments to evaluate the efficiencies of the algorithms.

Zandieh and Karimi (2011) consider a multi-objective group scheduling problem in a hybrid flexible flow shop setting with sequence-dependent setup times by minimizing the total weighted tardiness and the maximum completion time, simultaneously. They propose a multi-population genetic algorithm for the problem and compare it with the multi-objective genetic algorithm and the non-dominated sorting genetic algorithm.

Sawik (2012) provides mixed-integer programming models for cyclic or batch scheduling of a flexible flow shop with finite in-process buffers and continuous or limited machine availability, and compares the cyclic and batch scheduling modes. The computational experiments reported in the paper indicate that when setup times are negligible, cyclic scheduling outperforms batch scheduling for both continuous and limited machine availability. Based on these promising results, we also focus on cyclic schedules in our study.

As manufacturers implement larger and more complex robotic cells, more challenging optimization problems arise in handling such systems. To face this challenge, there have been many studies as early as dating back to late 1970s. We would like to draw the attention of the interested reader to surveys such as Crama, Kats, Van de Klundert, and Levner (2000), Lee, Lin, and Ying (2010), Dawande, Geismar, Sethi, and Sriskandarajah (2005) and Brauner (2008).

In robotic cells, different types of parts can be processed in lots. Parts typically differ from each other by having different processing times on a given machine. In multiple part type scheduling, a Minimal Part Set (MPS), i.e., the smallest possible set of parts having the same proportions as the overall production target, is produced repetitively. During an MPS cycle, all the parts in an MPS are taken from the input, get processed on appropriate machines and leave the system in its original starting state. Considering this cyclic production environment, the objective in multiple part type scheduling is to minimize the average time to produce one MPS. Multiple part-type problems are harder than their identical part-type counter problems even for small number of machines.

Machines in our robotic cell have the ability to handle a mixture of operations, which is defined as the process flexibility together with the ability to interchange the ordering of several operations for each part type, which is defined as the operational flexibility. Therefore, studies taking these definitions into account also play an important role in our study.

Gültekin, Aktürk, and Karaşan (2006) consider a robotic cell scheduling problem with two machines. Due to tooling constraints, some operations of identical parts can only be processed on certain machines. They find the allocation of the flexible operations to the two machines and the robot move cycle in order to minimize the cycle time.

The scope in Gültekin, Karaşan, and Aktürk (2009) is an $m$-machine flexible robotic manufacturing cell consisting of CNC machines. Using the advantage of the flexibility of the machines, the authors define a class of robot move cycles, namely pure cycles, and prove that, in most of the regions, one of these cycles is optimal.

Kamalabadi, Gholami, and Mirzaei (2007) consider multiple part type 3-machine robotic cells possessing operational flexibility that allow the operations to be performed in any order. They develop a mathematical model which is based on Petri nets and then, due to the difficulty of obtaining optimal solutions in reasonable computational times, they implement the particle swarm optimization heuristic for solving the problem.

Batur, Karaşan, and Aktürk (2012) focus on the scheduling problem arising in 2-machine flexible robotic cells which repeatedly produce a set of multiple part-types. As a result of the flexibility properties of the system, they try to find the robot move sequence as well as the processing times of the parts on each machine that minimize the cycle time.

The study of Elmi and Topaloglu (2013) is similar to ours. They deal with using multiple robots in hybrid flow shop robotic cells. A mixed integer linear programming model minimizing the makespan is proposed along with a simulated annealing based heuristic as solution methodologies. Although the cell formation considered in this study is close to the robotic cell in this study, our system becomes more complex when taking flexibility and sequencing of the robots moves into account. Additionally, we are able to provide a very effective lower bound value for our problem.

In hybrid flow shop scheduling, there are two inherent problems that have to be jointly solved, namely, the sequencing of parts on the stages and the allocation of parts to the different machines at each stage (Gupta, 1988). Considering these two problems together with the robotic cells, there are three main problems of this study; part input sequencing, part/machine allocation for each stage and the robot move sequencing.

In this study we focus on the scheduling problem observed in hybrid flexible flow shops where multiple part-types are produced and the transportation of these parts is performed by the help of a robot. Such systems necessitate multi-stage environments which may contain more than one machine and are commonly used in industries such as food processing, chemical, textile, metallurgical, printed circuit board and automobile manufacturing. Within our scope is an in-line robotic cell formation in which the first stage has only one machine whereas the second stage has two identical machines.

In the following section, the notation and basic assumptions pertinent to this study are introduced. Section 3 presents the proposed mathematical model. In Section 4, a simulated annealing approach is proposed as a solution methodology and two different neighborhood structures are distinguished. In Section 5 the results of the heuristic methodology are compared against a proposed lower bound value. Section 6 summarizes the contributions and concluding remarks of this study.

## 2. Notation and assumptions

As is mentioned before, in the literature, there are studies under the topics of 'flow shops with parallel machines', 'hybrid flow shops' and 'flexible flow shops'. Although there are differences, following characteristics are in common:

(1) The number of processing stages $k$ is at least 2.
(2) Stage $k$ has $m_k \geqslant 1$ machines in parallel and in at least one of the stages $m_k > 1$.
(3) All jobs are processed following the same production flow: stage 1, stage 2, ..., stage $m$. A job might skip any number of stages provided it is processed in at least one of them.

In this study we have used the following assumptions which are also used in the "standard" form of the HFS problem (Ruiz & Vazquez-Rodriguez, 2010);

(1) All jobs and machines are available at time zero.
(2) Machines at a given stage are identical.
(3) Any machine can process only one operation at a time and any job can be processed by only one machine at a time.
(4) Setup times are negligible.
(5) Preemption is not allowed.
(6) The capacity of buffers between stages is unlimited.
(7) Problem data is deterministic and known in advance.

Throughout the production schedule, parts are taken from the input buffer, carried to related machine(s) according to the machine availability and the system conditions, and when their processing is completed carried to the output buffer by the robot. All of the parts move on the same direction from the first stage to the second one, but they do not have to visit both of the stages. In a 2-stage flexible flow shop, there are three types of parts with respect to the stages on which they are going to be processed. Parts are allowed to follow one of the three routes: 1 (just first stage), 2 (just second stage), and 1–2 (both stages). When a part is to be processed on the second stage that has two identical machines, the question is to determine which one of these two machines will process the part.

In this study, we consider constant travel time robotic cells. In such cells, the robot moves with varying acceleration and deceleration, and the robot travel-times between pairs of machines are roughly equal. Two main assumptions related to the robot travel times between machines, namely, additive travel-time and constant travel time, are both commonly used in industries. However, constant travel time cells seem to be more appropriate to certain robotic cell formations (Dawande, Sriskandarajah, & Sethi, 2002).

Fig. 1 depicts the cell under consideration. There are 2 blocks corresponding to 2 stages, in which the first block has only 1 machine whereas the second one contains 2 machines.

Against this background, the objective is to minimize the long run average cycle time required for the repetitive production of a minimal part set. Throughout this study each part in the MPS is treated independently since two identical parts belonging to the same part-type might have different allocations. The accompanying notation will be formally defined in the forthcoming sections.

To the best of the authors' knowledge, the current study is a first attempt to consider hybrid flexible flow shops in the context of robotic cells. Considering multiple part-types necessitates different processing requirements and ultimately makes our problem quite challenging.

## 3. Mathematical model

Batur et al. (2012) is a study of close kinship to ours. In their study, there are two identical machines that could work either in a flow shop manner or in parallel mode. Each machine has the capacity to perform all the operations of each part. The model developed decides for each part whether to distribute its operations to both machines or whether to handle all the operations in a single machine. In contrast, within the scope of the current study, the system works in a hybrid flow shop manner including the properties of both the flow shop and parallel models, and for each part, the stage process requirement is known in advance.

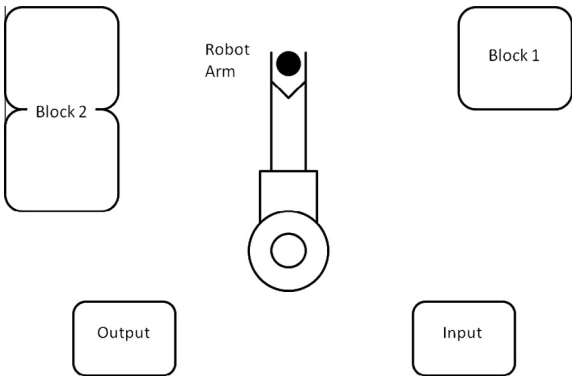


**Fig. 1.** Cell formation. The cell has input and output storages together with the two stages in which the first stage has only 1 machine and the second one has 2 identical machines.

Respecting these requirements, the decisions of part input sequences and part assignments to the identical machines on the stages are made.

This problem is modeled as a special traveling salesman problem (TSP) in which the distance matrix consists of decision variables as well as parameters. The proposed model is an adaptation of the mathematical model given by Batur et al. (2012). A basic definition used in this formulation is the following one:

**Definition 1.** The epoch that part $i$ is on station $q$ is identified by node $i_q$. The input buffer is denoted as station 1, machine in the first stage is denoted as station 2, first and second machines in the second stage are denoted as stations 3 and 4, respectively, and the output buffer is denoted as station 5. During an MPS cycle, the machines may need to be visited twice, one for loading and one for unloading of the same part since the robot may perform some other activities rather than to wait in front of the machine during the time that the part is being processed. Therefore, stations 6, 7 and 8 are also created as the copies of stations 2, 3 and 4, respectively, in order to account for any potential cyclic solution. For $q \in \{2,3,4\}$, we let $q' \in \{6,7,8\}$ denote its corresponding copy.

For the TSP formulation, we have an arc set $A$ and a node set $N$, which are represented as follows:

$$N = \{i_q :\ q = 1,\ldots,8,\ i = 1,\ldots,n\},$$
$$A = \{(i_q, j_r) :\ i_q, j_r \in N \text{ and the movement from node } i_q,$$
where part $i$ is at station $q$, to node $j_r$, where part $j$ is at station $r$, is possible$\}$.

The parameters and variables that are used in the mathematical model representation are given in Table 1. Variables $l2_{i_q}$, $l3_{i_q}$ and $l4_{i_q}$ will take values from the set $\{0,1,\ldots,n\}$. When the variable equals to 0, it means that the corresponding machine is empty at

**Table 1**
Parameters and decision variables used in the mathematical model representation.

| | |
|---|---|
| *Parameters* | |
| $n$ | Total number of parts to be produced in the MPS |
| $k$ | Number of processing stages |
| $m_k$ | Number of machines at stage $k$ |
| $p_{i_k}$ | Processing times of part-types $i$ to be produced in stage $k$, $i = 1,\ldots,n$, $k = 1,2$ |
| $\epsilon$ | Load/unload times of machines by the robot. (Consistent with the literature we assume that loading/unloading times for all machines are the same.) |
| $\delta$ | Time taken by the robot to travel between two machines. (The robot travel time is assumed to be constant.) |
| $time_{i_q j_r}$ | Total time needed for the movement prescribed by the robot activity from where part $i$ is at station $q$ to where part $j$ is at station $r$ |
| $wait_{i_q j_r}$ | 1, if there exists a potential waiting time for the movement of the robot from node $i_q$ to node $j_r$; and 0, otherwise |
| *Decision variables* | |
| $y_{i_q j_r}$ | 1, if robot goes from node $i_q$ to node $j_r$; and 0, otherwise |
| $p'_{i_q}$ | Processing time of part $i$ on station $q$, $q = 2,3,4,6,7,8$ |
| $s_{i_q}$ | Time that processing of part $i$ is started on station $q$, $q = 2,3,4,6,7,8$ |
| $c_{i_q}$ | Time that robot completed the activity related to node $i_q$, $q = 2,3,4,6,7,8$ |
| $w_{i_q}$ | Robot waiting time for part $i$ on station $q$, $q = 2,3,4,6,7,8$ |
| $v_{i_q}$ | Total activity time of the robot in between just after loading the machine corresponding to station $q$ by part $i$ and arriving in front of the same machine to unload it, $q = 2,3,4,6,7,8$ |
| $z_{i_q}$ | 1, if start time of processing of part $i$ on station $q$ is considered before its completion time within a cycle; and 0, otherwise. |
| $l2_{i_q}$ | Loaded part on station 2, when robot is at node $i_q$, $q = 2,3,4,6,7,8$ |
| $l3_{i_q}$ | Loaded part on station 3, when robot is at node $i_q$, $q = 2,3,4,6,7,8$ |
| $l4_{i_q}$ | Loaded part on station 4, when robot is at node $i_q$, $q = 2,3,4,6,7,8$ |
| $C$ | Cycle time value |

that moment; whereas its equivalence to any value between 1 and $n$ means that the machine is loaded with that particular part at node $i_q$.

The total time spent in going from node $i_q$ to node $j_r$ is shown by $time_{i_q j_r}$ values. For example, a movement from node $i_1$ to node $i_2$ corresponds to the situation that the robot takes a part from the input buffer ($\epsilon$), carries it to the second station (the first machine of the first stage) ($\delta$) and loads the part ($\epsilon$); making a total of $2\epsilon+\delta$ time units. For our problem, related $time_{i_q j_r}$ values are shown in Table 2, where the first part corresponds to the states on which $i=j$ and the second one corresponds to the states on which $i\neq j$. Moreover, the movements with costs marked by $X$'s cannot be performed due to the feasibility conditions. Besides, some movements are unreasonable; for example, a part cannot be taken from the input buffer and left to the output buffer without any processing, cannot be taken from the output buffer, or cannot be left on the input buffer. For example, $y_{i_2 i_1}$ necessitates part $i$ to be taken from station 2 and carried to station 1 which is the input buffer, however, such a movement cannot be performed.

As can be seen from Table 2, some movements are considered to be possible owing to the flexibility property of the system. In flexible flow shop environments, a part is allowed to enter the system at second stage instead of the first stage. Therefore, the robot movement from station 1 to stations 3(7) or 4(8) is possible together with the movement from station 1 to station 2(6). Similarly, any part for which all the processing is completed in the first stage, can be carried directly to the output buffer, instead of visiting the second stage; which causes the movement from station 2 (6) to station 5 to be also possible.

Waiting time, which occurs when the robot is ready to unload but when the processing of the loaded part has not been completed yet, is represented as follows:

$$w_{i_q} = max\left\{0, p'_{i_q} - v_{i_q}\right\},\ \forall\ i,q \tag{1}$$

where $i$ is the loaded part on station $q$ and $v_{i_q}$ is the total activity time of the robot in between just after loading the machine corresponding to station $q$ by part $i$ and arriving in front of the same machine to unload it.

$wait_{i_q j_r}$ parameters are used to determine for which movements there is a potential waiting time. Clearly movements corresponding to different types of parts do not necessitate any waiting time value; so these parameters are defined only for the situations associated with the same parts. Waiting is possible for part $i$ among the movements for which the corresponding cost values are underlined in Table 2; so $wait_{i_q j_r}$ equals to 1 for these movements. For example, the arc from node $i_2$ to node $i_3$ corresponds to unloading part $i$ from station 2 and loading it onto station 3 and this movement may cause some waiting in front of station 2 until the processing of part $i$ is completed.

Now, we proceed with our mathematical model. Our first set of constraints are the assignment constraints. Eq. (2) guarantees that when there is an incoming arc to a node, there must also be an outgoing arc from it.

$$\sum_{(j_r):\ (j_r,i_q)\in A} y_{j_r i_q} = \sum_{(j_r):\ (i_q j_r)\in A} y_{i_q j_r},\ \forall\ i_q \in N \tag{2}$$

Since some nodes are allowed not be visited, the assignment constraints of a TSP become the following:

$$\sum_{(j_r):(i_q j_r)\in A} y_{i_q j_r} \leqslant 1,\ \forall\ i_q \in N,\ \text{and}\ \sum_{(j_r):(j_r,i_q)\in A} y_{j_r,i_q} \leqslant 1,\ \forall\ j_r \in N \tag{3}$$

We need to ensure that all the parts are going to be taken from the input buffer and left to the output buffer exactly once. This fact is guaranteed by Eq. (4).

$$\sum_{(j_r)} y_{i_1 j_r} = 1,\ \forall\ i \text{ and } \sum_{(j_r)} y_{j_r i_5} = 1,\ \forall\ i \tag{4}$$

The considered robotic cell consists of 2-stages and owing to its flexibility property, a part can be carried from the input buffer to any of these stages and also to the output buffer from any of the stages. The following equations enforce only viable movement options.

$$\begin{aligned} &y_{i_1,i_2} + y_{i_1,i_6} + y_{i_1,i_3} + y_{i_1,i_7} + y_{i_1,i_4} + y_{i_1,i_8} = 1,\ \forall\ i \\ &y_{i_2,i_5} + y_{i_3,i_5} + y_{i_4,i_5} + y_{i_6,i_5} + y_{i_7,i_5} + y_{i_8,i_5} = 1,\ \forall\ i \end{aligned} \tag{5}$$

Eq. (6) is used to define the movements between the stages. As it is mentioned before, a job might skip any number of stages provided it is processed in at least one of them. Following this explanation, not every part has to visit both of the two stages and Eq. (6) corresponds to this situation.

$$y_{i_2,i_3} + y_{i_2,i_7} + y_{i_6,i_3} + y_{i_6,i_7} + y_{i_2,i_4} + y_{i_2,i_8} + y_{i_6,i_4} + y_{i_6,i_8} \leqslant 1,\ \forall\ i \tag{6}$$

Without loss of generality, the system is assumed to start when the robot is in front of the input buffer and ready to take part 1, i.e., at node $1_1$, as is defined by Eq. (7).

$$c_{1_1} = 0 \tag{7}$$

Eq. (8) is used in order to calculate the completion times of nodes, according to the movements, costs and waiting time values, i.e., the classical Miller-Tucker-Zemlin constraints for TSP (Miller, Tucker, & Zemlin, 1960).

$$c_{j_r} \geqslant c_{i_q} + time_{i_q j_r} \cdot y_{i_q j_r} - M(1 - y_{i_q j_r}) + wait_{i_q j_r} \cdot w_{i_q}\ \forall\ i_q, j_r \in N \tag{8}$$

Another set of equations are the processing time related constraints. Eq. (9) indicates that all the processing of any part at any stage must be completed. Moreover, they force that the processing times at stations and their duplicates are identical.

$$p'_{i_2} = p_{i_1} \text{ and } p'_{i_3} + p'_{i_4} = p_{i_2} \text{ where } p'_{i_2} = p'_{i_6}, p'_{i_3} = p'_{i_7} \text{ and } p'_{i_4} = p'_{i_8},\ \forall\ i \tag{9}$$

Equation set (10) is used to define that when node $i_q$ is visited on any path, a process with a time value of at most the total processing time of part $i$ can be performed on station $q$. Besides, if node $i_q$ is not visited, no process will be performed on the station.

$$\begin{aligned} &p'_{i_q} \leqslant p_{i_1} \cdot \sum_{(j_r):(i_q j_r)\in A \text{ or } (i_{q'} j_r)\in A} (y_{i_q j_r} + y_{i_{q'} j_r}),\ i_q \in N \text{ s.t. } q=\{2\}\ q'=\{6\} \\ &p'_{i_q} \leqslant p_{i_2} \cdot \sum_{(j_r):(i_q j_r)\in A \text{ or } (i_{q'} j_r)\in A} (y_{i_q j_r} + y_{i_{q'} j_r}),\ i_q \in N \text{ s.t. } q=\{3,4\}, q'=\{7,8\} \end{aligned} \tag{10}$$

**Table 2**
Total times of movements performed. $i$ and $j$ values denote the nodes.

| | $(j_1)$ | $(j_2)/(j_6)$ | $(j_3)/(j_7)$ | $(j_4)/(j_8)$ | $(j_5)$ | $(j_1)$ | $(j_2)/(j_6)$ | $(j_3)/(j_7)$ | $(j_4)/(j_8)$ | $(j_5)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $(i_1)$ | X | $2\epsilon+\delta$ | $2\epsilon+\delta$ | $2\epsilon+\delta$ | X | X | X | X | X | X |
| $(i_2)/(i_6)$ | X | X | $\underline{2\epsilon+\delta}$ | $\underline{2\epsilon+\delta}$ | $\underline{2\epsilon+\delta}$ | $\delta$ | X | $\delta$ | $\delta$ | X |
| $(i_3)/(i_7)$ | $\delta$ | X | X | X | $\underline{2\epsilon+\delta}$ | $\delta$ | $\delta$ | X | $\delta$ | X |
| $(i_4)/(i_8)$ | $\delta$ | X | X | X | $\underline{2\epsilon+\delta}$ | $\delta$ | $\delta$ | $\delta$ | X | X |
| $(i_5)$ | $\delta$ | X | X | X | X | $\delta$ | $\delta$ | $\delta$ | $\delta$ | X |

$s_{i_q}$ and $c_{i_q}$ variables are defined as the beginning of the processing on a station and the time that the related movement is performed on a node, respectively. As is given by Equation set (11), the beginning time of the processing is calculated according to the time that it is loaded on the station. Since stations $q$ and $q'$ are the copies of each other, this value is represented by either $s_{i_q}$ or $s_{i'_q}$.

$$\begin{aligned}
&s_{i_q} = s_{i_{q'}} \ \forall\, i_q \in N, \text{ where } q = 2,3,4\\
&s_{i_r} \geqslant c_{i_r} - M(1 - yi_q, i_r) \ \forall\, i_q \in N, \text{ where } q = 1,2,3,4,\ r \neq 5,\ q \neq r\\
&s_{i_r} \geqslant c_{i_{r'}} - M(1 - yi_q, i_{r'}) \ \forall\, i_q \in N, \text{ where } q = 1,2,3,4,\ r \neq 5,\ q \neq r,\\
&q' \neq r',\ q \neq r'
\end{aligned} \tag{11}$$

This set of equations shows that if there is an arc from node $i_q$ to node $i_r$ or $i_{r'}$, meaning that part $i$ is unloaded from station $q$ and loaded on station $r$ or the copy station $r'$, then its processing on machine corresponding to station $r$ or $r'$ starts at the time when the part is left on the machine which is equal to the time when the movement is performed. Since we cannot take any part from the output buffer, $q$ cannot be equal to station 5.

In order to calculate the waiting time values, $z_{i_q}$ variables, which are defined for all of the nodes, are used. This variable helps us to understand whether the load or unload of a part is performed sooner in a given cycle. $z$ values are determined using Equation set (12).

$$\begin{aligned}
&c_{i_q} \geqslant s_{i_q} + p'_{i_q} - M(1 - z_{i_q}),\ \forall\, i_q \in N\\
&s_{i_q} \geqslant c_{i_q} + 4\epsilon + 3\delta - Mz_{i_q},\ \forall\, i_q \in N
\end{aligned} \tag{12}$$

In the second situation, throughout a cycle, an already loaded part needs to be unloaded and loaded again. Thus, in the second equation of Equation set (12), $4\epsilon + 3\delta$ is used to define the minimum time between the machine's unload and its next loading.

The waiting time value, which is observed when a part's processing is not completed until the robot arrives the machine to unload it, is calculated using the starting and completion times ($s_{i_q}$ and $c_{i_q}$ values) together with $z_{i_q}$ variables.

$$\begin{aligned}
&w_{i_q} \geqslant p'_{i_q} - (c_{i_q} - s_{i_q}) - M(1 - z_{i_q}),\ \forall\, i_q \in N\\
&w_{i_q} \geqslant p'_{i_q} - (C - s_{i_q} + c_{i_q}) - Mz_{i_q},\ \forall\, i_q \in N
\end{aligned} \tag{13}$$

We also have "loaded parts vs. movements" related constraints. This set of equations are used in order to identify the parts loaded on the stations when a movement is performed. Considering the movement $(i_q, j_r)$, we have two cases, which are explained as follows:

- Movement is related to different parts, i.e., $i \neq j$, meaning that part $i$ is loaded on station $q$ and the robot travels to station $r$ for part $j$. Such a movement does not cause any difference in terms of the loaded parts for the machines and the following equations are used for this type of movements:

$$-l2_{i_q} + m2_{j_r} \leqslant n(1 - y_{i_q j_r}), \quad l2_{i_q} - m2_{j_r} \leqslant n(1 - y_{i_q j_r}) \tag{14}$$

$$-l3_{i_q} + m3_{j_r} \leqslant n(1 - y_{i_q j_r}), \quad l3_{i_q} - m3_{j_r} \leqslant n(1 - y_{i_q j_r}) \tag{15}$$

$$-l4_{i_q} + m4_{j_r} \leqslant n(1 - y_{i_q j_r}), \quad l4_{i_q} - m4_{j_r} \leqslant n(1 - y_{i_q j_r}) \tag{16}$$

- Movement is related to the same part, i.e., $i = j$, meaning that part $i$ is taken from station $q$ and carried to station $r$. Such a movement needs the following conditions to be met:
  - Machine corresponding to the station $r$ needs to be empty at node $i_q$.

$$l2_{i_q} \leqslant n(1 - y_{i_q, i_r}),\ l3_{i_q} \leqslant n(1 - y_{i_q, i_r}),\ \text{or } l4_{i_q} \leqslant n(1 - y_{i_q, i_r}) \tag{17}$$

  - Machine corresponding to the station $q$ will become empty at node $i_q$.

$$l2_{i_r} \leqslant n(1 - y_{i_q, i_r}),\ l3_{i_r} \leqslant n(1 - y_{i_q, i_r}),\ \text{or } l4_{i_r} \leqslant n(1 - y_{i_q, i_r}) \tag{18}$$

  - This type of a movement results with no difference in terms of the loaded parts for the machines corresponding to the stations other than $q$, $q'$ and $r$, $r'$; and Eqs. (14)–(16) are used for this situation, as in the first case.

Because of the considered cell formation, some movements may block some others. For example, part carriage between stations 3 and 4 is not allowed, since all of the processing of any part at any stage should be completed by only one of the machines at this stage. Equation set (19) is used to exclude such movements.

$$\begin{aligned}
&y_{i_q, i_r} + y_{i_r, i_q} + y_{i_q, i_{r'}} + y_{i_r, i_{q'}} + y_{i_{q'}, i_r} + y_{i_{r'}, i_q} + y_{i_{q'}, i_{r'}} + y_{i_{r'}, i_{q'}} = 0\ q, q', r,\\
&\quad r' = \{3,4,7,8\}\\
&\sum_{j_s} (y_{j_s, i_q} + y_{j_s, i_r}) \leqslant 1\ \forall\, i,\ q, q', r, r' = \{3,4,7,8\},\ q \neq r,\ q \neq r',\\
&\quad q' \neq r,\ q' \neq r'
\end{aligned} \tag{19}$$

It is known that, due to the flexibility property of the system, when a part does not have any operation to be performed on the first stage but only on the second stage, it can directly be carried to the second stage from the input buffer; similarly, when a part does not have any operation to be performed on the second stage but only on the first stage, it can directly be carried to the output buffer from the first stage. This fact is guaranteed by the following equations:

$$\begin{aligned}
&y_{i_1, i_2} + y_{i_1, i_6} \leqslant p_{i_1},\ \forall\, i\\
&y_{i_1, i_2} + y_{i_1, i_6} \geqslant p_{i_1}/M,\ \forall\, i\\
&y_{i_3, i_5} + y_{i_4, i_5} + y_{i_7, i_5} + y_{i_8, i_5} \leqslant p_{i_2},\ \forall\, i\\
&y_{i_3, i_5} + y_{i_4, i_5} + y_{i_7, i_5} + y_{i_8, i_5} \geqslant p_{i_2}/M,\ \forall\, i
\end{aligned} \tag{20}$$

The first two equations in set (20) avoid the movement from input buffer to the stations 2(6) in case the processing time at the first stage, i.e., $p_{i_1}$ is zero, whereas the same movement is guaranteed to occur when it is more than zero. Similarly, the other two equations avoid the movements from the stations 3(7) or 4(8) to output buffer if the processing time at the second stage is zero, and this movement is guaranteed when it is more than zero. For these four equations in Equation set (20), $M$ represents a large enough number for which both the values $p_{i_1}/M$ and $p_{i_2}/M$ are at most 1. In order to guarantee this fact, $M = \max_i\{\max\{p_{i_1}, p_{i_2}\}\}$ can be used.

In our model, the objective is to find the minimum cycle time value and Eq. (21) is used in order to reach this goal. Using this constraint, the completion time of the last node, which gives the completion time of the cycle, is determined.

$$C \geqslant c_{i_q} + y_{i_q, 1_1} \cdot time_{i_q, 1_1} \tag{21}$$

Against all, we propose the following mixed integer linear programming formulation:

$$\begin{aligned}
&\min\ C\\
&\text{Subject to } (2) - (21)
\end{aligned}$$

We use the following example to describe the proposed mathematical model.

**Example 1.** Assume that we have 3 part to be completed having the processing values of $p_{1_1} = 17$, $p_{2_1} = 20$, $p_{3_1} = 0$, $p_{1_2} = 30$, $p_{2_2} = 34$, $p_{3_2} = 27$. $\epsilon$ and $\delta$ are 1 and 2 time units, respectively.

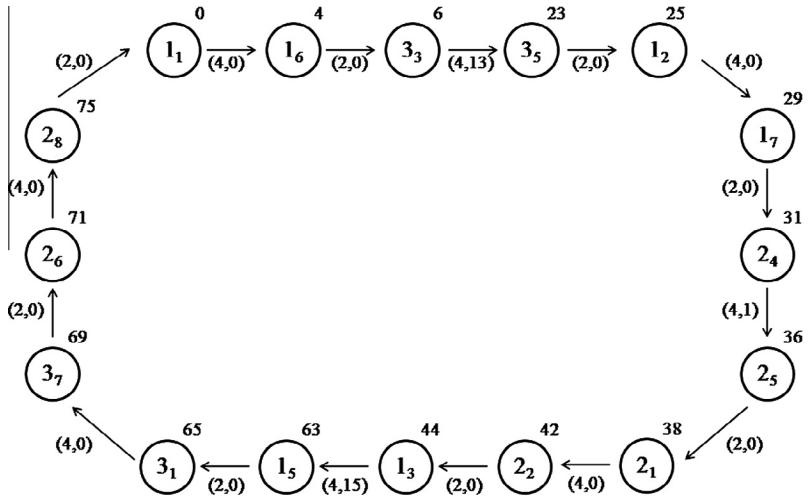


**Fig. 2.** TSP representation of Example 1. Circles represent the nodes and arcs represent the movements of the robot.

Optimal tour that correspond to the TSP formulation is given by Fig. 2. The values written on the arcs represent the travel and waiting time values between the nodes respectively. For example, in order to move from node $3_3$ to node $3_5$, a travel of 4 units of time is needed together with a waiting of 13 units, which gives a total of 17 units of time.

The Gantt-chart of the solution obtained by the model is also given in Fig. 3. As can be followed from the figure, at time $t = 0$, 3rd and 4th stations are loaded whereas the 2nd station is empty, and robot takes part 1 from the input buffer in order to load it on this empty station. This movement includes the robot taking the part from the input buffer ($\epsilon$), traveling to the 2nd station ($\delta$) and loading the part on the station ($\epsilon$), which gives a total of $2\epsilon + \delta$ units of time. Further steps can also be observed in the same manner. As can be seen from the figure, the cycle is completed at time $t = 77$ and at that moment all the stations returns back to their initial situations at time $t = 0$.

TSP is a well known NP-Hard problem. The formulation above is more general than the classical TSP formulation and requires a great amount of computational effort even if the number of machines in the cell is small. Nevertheless, it is possible to extend the model considering a structure with more stages or machines. For this aim, it will be enough to add machine related decision variables together with extra constraints defining the possible movements between the new stations. During our preliminary computations, we were able to get optimal solutions to instances with very small dimensions. Nonetheless, we opted to provide an exact solution methodology for our problem for completeness sake. Consequently, we focussed our attention on heuristic approaches and composed the Simulated Annealing based algorithm introduced in Section 4.

## 4. Heuristic algorithm

Simulated Annealing (SA) was introduced by Metropolis, Rosenbluth, and Resenbluth (1953) and popularized by Kirkpatrick, Gelatt, and Vecchi (1983) as a competitive method to solve combinatorial optimization problems (Zegordi, Itoh, & Enkawa, 1995). It is believed to be a successful metaheuristic when solving scheduling problems with various objectives like make-span, flow time, idle time, work-in-process and tardiness, etc. (Hooda & Dhingra, 2011). Kim, Kim, Jang, and Chen (2002), Low, Yeh, and Huang (2004), Lee et al. (2010), Behnamian, Zandieh, and Fatemi Ghomi (2009) and Zhang and Wu (2010) are researchers that have utilized SA competitively in their optimization. Our computational results will attest to this finding as well.

SA starts working with a high temperature. The search travels from a current solution to one of its neighboring solutions. A new solution may be accepted even if its objective function value is worse than that of the previous one depending on a predefined probability function. The probability function is defined as follows:

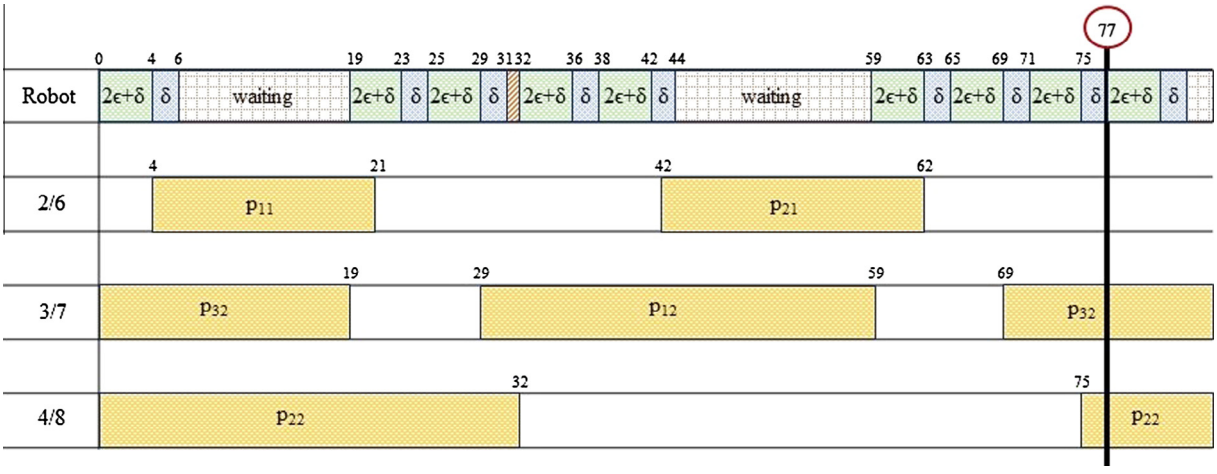


**Fig. 3.** Gantt-chart of Example 1. The first row corresponds to the robot's movements and the others to the stations.

$P(\Delta E) = e^{-\Delta E/T}$, where $\Delta E$ indicates the rate of change in the objective function and $T$ indicates the current temperature. For this control process, a random value is selected from the interval (0,1); if this randomly chosen value is less than or equal to the probability value the new solution is accepted, otherwise it is rejected. Allowing for moves that worsen the objective function values aid in avoiding getting stuck at local optima. To reach the thermal equilibrium of SA, the process is repeated $L$ times at each temperature, where $L$ is a control parameter referred to as the length of the Markov chain. The search continues until the termination criterion is reached, decreasing the temperature value according to a predetermined cooling function.

In order to apply SA procedure to a combinatorial optimization problem, there are some critical decisions to be made. The starting temperature must be hot enough to allow a move to almost any neighborhood state. However, if the temperature starts at too high a value, then the search can move to any neighbor and thus transform the search (at least in the early stages) into a random search. When deciding on the final temperature, it is typical to let the temperature decrease until it reaches zero. Some implementations keep decreasing the temperature until some other condition is met; for example, no change in the best state for a certain period of time. In our study, these two temperature values are taken from the paper by Karaoglan, Altıparmak, Kara, and Dengiz (2011). In their study, the initial temperature is taken as 665 in which an inferior solution (inferior by 70% relative to the current solution) is accepted with a probability of 0.90 and the final temperature is taken as 0.15 such that a solution which is inferior by 1% relative to current solution is accepted with a probability of 0.1%. The following equations which are developed according to the previously defined probability function, are used to determine these values:

$$P(70) = e^{-70/T} = 0.90 \rightarrow ln(0.90) = -70/T \rightarrow T = -70/ln(0.90) = 665$$
$$P(1) = e^{-1/T} = 0.001 \rightarrow ln(0.001) = -1/T \rightarrow T = -1/ln(0.001) = 0.15 \quad (22)$$

In the light of the given equations, the termination temperature value is taken to be 0.15 whereas different probability values are tested for the starting temperature. In addition to four values which are determined using Eq. (22) and given in Table 3, the value 100 which is another commonly used initial temperature value in the literature, is taken into account; i.e., we have used five different initial temperatures which are 100, 168, 196, 313 and 665.

The way in which we decrement our temperature is critical for the success of the algorithm. We use a geometric decrement where $T = T \times (DecRatio)$, where $DecRatio < 1$. Experience has shown that $DecRatio$ should be between 0.8 and 0.99, with better results being found in the higher end of the range. In this study the temperature decrement is taken as $DecRatio = 0.99$.

Another crucial decision to make is the number of iterations to perform at each temperature. A constant number of iterations at each temperature is a typical scheme. We assumed the iteration number to be equal to the given number of parts of the considered problem.

For this problem, solutions should provide the part sequences and the part/machine allocations for stages containing more than one machine. In order to represent the solution, a vector of length $n + (m \times n)$ is used, where $n$ is the part number and $m$ is the number of stages in the cell. This vector composes of $m + 1$ segments, each of length $n$. The first segment identifies the part sequence and the remaining m segments identify the machines for each of the stages, on which these sequenced parts are to be processed. According to the sequence and allocations identified by this vector representation, the robot takes the parts from the input buffer, delivers them to allocated machines and carries them to the output buffer when their processing is completed.

**Table 3**
Initial temperature values, which are determined using Eq. (22).

| $\Delta E$ (%) | Acceptance probability | Initial temperature |
|---|---|---|
| −60 | 0.70 | 168 |
| −70 | 0.70 | 196 |
| −70 | 0.80 | 313 |
| −70 | 0.90 | 665 |

The quality of a specific solution vector is measured by our objective function which is the minimization of the cycle time that passes between the time that the first part is taken from the input buffer and the time that the last part is left to the output buffer. Once the sequence and allocations are provided by a solution vector, calculation of the cycle time can be easily performed.

The initial solution vector of the problem is constructed randomly. Throughout our search, two different neighborhood strategies are used each resulting in $n$ new solutions at each iteration. The neighbor solution is picked as the best one among these new solutions and is accepted/rejected according to the previously defined probability function.

*Neighborhood 1*. The first neighborhood constructs feasible solutions that can be attained through random swap operations. More specifically, a random value ($r$) is selected from the range $(0, 1)$ and the following conditions are checked:

- if $r > 0.5$ then part sequence of two randomly selected parts are changed,
- if $r \leq 0.5$ then part/machine allocation of a randomly selected part for the second stage which has two machines, is changed.

*Neighborhood 2*. In the second neighborhood, in order to get rid of the randomness of the first neighborhood structure, some components of the current solution are maintained while constructing the new solution. First, the idle time and the waiting time values pertaining to the current solution are calculated. Analyzing these two values, the bottleneck parts of the current solution, for which improvement can be observed, are distinguished. Once the idle and waiting time values are determined for all the stages and machines in the cell, a neighbor solution is constructed according to the rules given below. The flow chart of Neighborhood 2 is depicted in Fig. 4.

*Rule 1*. If the machine with the largest idle time value (say $j$) and the machine with the largest waiting time value in the same stage (say $k$) are the same, the following $X_A$ value is calculated and a new solution is obtained by changing the order of the part causing the largest waiting time (say $i$) with the part which has the processing time value closest to the value $X_A$.

$P_{ik}$ processing time of the part causing the largest waiting time at stage $k$.
$w_{ik}$ largest waiting time value.

$$X_A = P_{ik} - w_{ik} \quad (23)$$

Using this rule, a part which is thought to cause little or no waiting is swapped with the part with the largest waiting time value. By escaping the waiting time values which increase the cycle time value significantly, the cycle time might decrease. Changing the parts causing waiting times with the parts causing idle times, it is tried to balance the idle and waiting times throughout the system; besides filling the idle time intervals, the waiting times will decrease, ultimately decreasing the cycle time.

*Rule 2*. If the machine with the largest idle time value (say $j$) and the machine with the largest waiting time value in the same stage (say $k$) are different, the following $X_B$ value for machine $j$ and $X_C$ value for machine $k$ are calculated. The new solution is obtained by changing the part/machine allocations of the part causing the

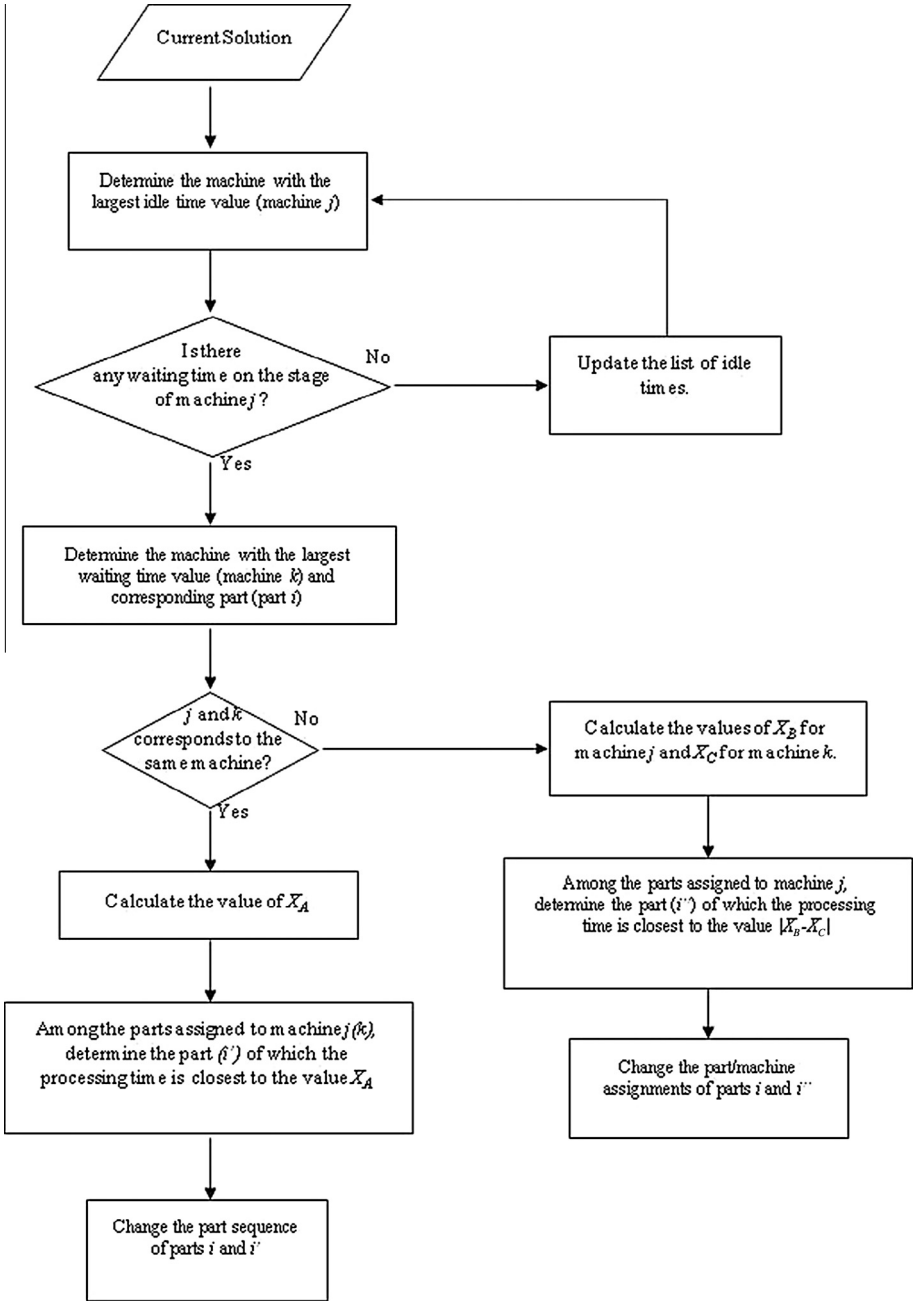


**Fig. 4.** Flowchart of Neighborhood 2. The values of $X_A$, $X_B$ and $X_C$ are calculated using the Eqs. (23) and (24), respectively.

largest waiting time on machine $k$ with the part on machine $j$ having the closest processing time value to the value $|X_B - X_C|$.

$P_{ik}$ processing time of the part causing the largest waiting time at stage $k$.
$b_{j(k)}$ idle time value on machine $j(k)$.

$$X_B = b_j$$
$$X_C = b_k + P_{ik} \quad (24)$$

This rule also helps to balance the idle and waiting times in the system. When the part with the largest waiting time is not assigned to the related machine, there occurs extra space (which is equal to the processing time of the part) in addition to the existing idle time at this machine. If a part of which the processing time is close enough to the difference between the current idle time at the machine and the largest idle time is found, then it is predicted that there may occur an improvement in the cycle time value.

One of the main steps of the solution procedure is the calculation of the cycle time value. The notations used in this section are given in Table 4. Besides, the pseudo-code of the algorithm is given in Algorithm 1.

**Table 4**
Notations used in the algorithm, different from the mathematical model.

| | |
|---|---|
| $V$ | Solution vector |
| $S$ | Event list |
| $S'$ | Possible event list |
| $Z_j$ | Indicates whether part $j$ is in the input buffer or not |
| $mc_j$ | Current machine for part $j$ |
| $mn_j$ | Next machine for part $j$ |
| $D_m$ | Loaded part on machine $m$ |
| $Q_m$ | Priority of machine $m$, which is determined according to the total workload of related machine |

**Algorithm 1.** Cycle Time Value.

```
1: Input: V, ϵ, δ, p_11, ..., p_nm.
2: Output: S, C.
3: S = ∅,
4: for i = 0, i < n, i++ do
5:    Z_i = 1,
6: end for
7: OutputDepot = Σ_{i=1}^{m} m_i + 1
8: i = Vec[0]
9: S = S ∪ {i}
10: while CycleTimeLoop do
11:   while TryAgain do
12:     for i = 0, i < n_s, i++ do
13:       j = S[i]
14:       if Z_j = 1 then
15:           if D_{mn_j} = 0 then
16:             A_1 = 1
17:           else
18:             if Z_j = 0 then
19:               if mn_j = OutputDepot then
20:                 A_2 = 1
21:               else
22:                 if D_{mn_j} = 0 then
23:                   A_3 = 1
24:                 end if
25:               end if
26:             end if
27:           end if
28:         end if
29:       if A_1 = 1 or A_2 = 1 or A_3 = 1 and t_j ⩽ t then
30:         S' = S' ∪ j
31:       end if
32:     end for
33:     if S' = ∅ then
34:       for i = 0, i < n_S, i++ do
35:         if Z_i = 0 and t_i < t_min then
36:           if mn_i = OutputDepot or D_{mn_i} = 0 then
37:             t_min = t_i
38:           end if
39:         end if
40:       end for
41:       t = t_min
42:         TryAgain = true
43:     else
44:       if n_{S'} = 1 then
45:         j = S'[0]
46:       end if
47:     else
48:       for i = 0, i < n_s, i++ do
49:         k = S'[i]
50:         if Q[mn_k] < Q then
51:           Q = Q[mn_k]
52:           j = S'[i]
53:         end if
54:       end for
55:     end if
56:   end while
57:   if Z_j = 1
58:     Z_j = 0
59:     Update (mn_j, mc_j, t, t_j)
60:     if j = S[1] then
61:       Y_j = FirstPart
62:       l = l + 1
63:       L_{S(l)} = t
64:       C_Tp = L_{S(l−1)} − L_{S(l−2)}
65:       C_Tc = L_{S(l)} − L_{S(l−1)}
66:       if C_Tp = C_Tc then
67:         CycleTimeLoop = false
68:       else
69:         CycleTimeLoop = true
70:       end if
71:     end if
72:     i = Vec[j]
73:     S = S ∪ {j}
74:   else
75:     if Z_j = 0 then
76:       if mn_j = OutputDepot then
77:         Update(mn_j, mc_j, t, t_j)
78:       else
79:         D_{mc_j} = 0
80:         Update(mn_j, mc_j, t, t_j)
81:       end if
82:     end if
83:   end if
84: end while
```

Algorithm works as follows:

*Step 1.* All the parts are loaded on the input buffer (line 4).

*Step 2.* The first part of the solution vector is added to the event list (line 9).

*Step 3.* Possible event list is determined according to the parts in the event list.

If the first part of the event list is in the input buffer (as in line 14) and if the next machine for the part is empty, then $A_1 = 1$.

If not,

If the next machine for the part is empty, $A_3 = 1$.

If $A_1 = 1$ or $A_2 = 1$ or $A_3 = 1$ and processing of at least one part is completed (line 29), related part is added to the possible event list.

*Step 4.* When all of the parts in the event list are checked, if there are no parts in the possible event list (line 33), update the system time according to the earliest completed part's completion time and go to Step 3.

If not,

If there is only one part in the possible event list (line 44), choose that part.

If not,

If there are more than one parts in the list, choose the part according to the machine priorities (lines 50–53).

*Step 5.* If the selected part is in the input buffer (line 57) corresponding machine and time updates are performed and the part is carried and loaded on the related machine.

If the chosen part is the first part in the solution vector, the difference between the times at which this movement is performed in the last two cycles is calculated and this value is saved as the current cycle time. In case of equivalence of the last two cycle time values, the variable *CycleTimeLoop* is updated which indicates that the cycle time value is found.

If not,

If the part is going to be carried to the output buffer or any other machine, the part is unloaded and carried to the output buffer or corresponding machine, performing the necessary updates of this movement.

*Step 6.* Add the next part of the solution vector to the event list and go to *Step 3* (line 72).
*Step 7.* When all the parts of the solution vector are taken from the input buffer exactly once, go to *Step 1* and load a new set of parts on the input buffer. Repeat *Step 1* to *Step 6* until two successive equal cycle time values are found which is defined by the variable *CycleTimeLoop*. When such two cycles are determined, the corresponding cycle time value is taken as the final cycle time value of the problem.

## 5. Lower bound and performance analysis

In order to evaluate the performance of our SA algorithm, we propose the following lower bound (*L.B.*) value for our problem.

**Theorem.** *The following is a lower bound value for the cycle time of the scheduling problem in constant travel time hybrid flow shop robotic cells producing multiple part-types:*

$$LowerBound(L.B.) = \max_k \left\{ \frac{\sum_j p_{jk} + n_k \times (4\epsilon + 3\delta)}{m_k} \right. \quad (25)$$

*where $p_{jk}$, $n_k$, $4\epsilon + 3\delta$ and $m_k$ define the processing time of part j at stage k, the number of parts to be processed at stage k, the total time needed to take a part from the input buffer or any previous stage to the next stage or the output buffer and number of machines at stage k, respectively.*

**Proof.** In order to achieve this lower bound value, consider the following smallest possible cell formation in the context of a hybrid flow shop formation in single stage single machine robotic cells.

*Case 1.* In a single stage single machine system, for which a gant-chart is given in Fig. 5, the robot takes the part from the input buffer ($\epsilon$), travels to the machine ($\delta$), loads the part ($\epsilon$), waits until its processing is completed ($w_1 = p_1$), takes the part of which the processing is completed ($\epsilon$), travels to the output buffer ($\delta$), unloads ($\epsilon$) and goes back to the input buffer to take a new part ($\delta$). Total time needed for any part $i$ equals to $p_i + 4\epsilon + 3\delta$. This scheme of movements is the same for all of the parts, thus the cycle time of this cell equals to $\sum_i w_i + n(4\epsilon + 3\delta) = \sum_i p_i + n(4\epsilon + 3\delta)$. In Fig. 5 this time can be traced between the arrows.
*Case 2.* In the second step, we consider the robotic cell with single stage multi-machines, for which a gant-chart is given in Fig. 6. In this situation, total time needed for each of the parts is again $p_i + 4\epsilon + 3\delta$, but since the robot will perform for more than one machine instead of working on a single machine, there will occur idle and waiting times for the machines. In this case, the total time for any machine $j$ equals to $\sum_i p_{ij} + 4\epsilon + 3\delta + \sum_i d_{ij}$, where $p_{ij}$ defines the processing time of part $i$ on machine $j$ and $d_{ij}$ defines the idle time value observed for part $i$ on machine $j$. Since the idle times change according to the robot's movements, when calculating the lower bound value we take the smallest possible idle time value which is equal to zero, thus the total time needed for machine $j$ is at least $\sum_i p_{ij} + 4\epsilon + 3\delta$. It is obvious that when there are more than one machine to perform, the best results are obtained when the machines are equally loaded. Thus, total time needed for all the parts to be processed equals to $\sum_j \sum_i p_{ij} + n(4\epsilon + 3\delta)$ where $n$ is the number of the parts. Allocating this time among the machines equally, the lower bound of the cycle time value is found to be $(\sum_j \sum_i p_{ij} + n(4\epsilon + 3\delta))/m$ where $m$ is the number of machines.
*Case 3.* When there is more than one stage and only one machine at each of these stages, as can be seen from Fig. 7,

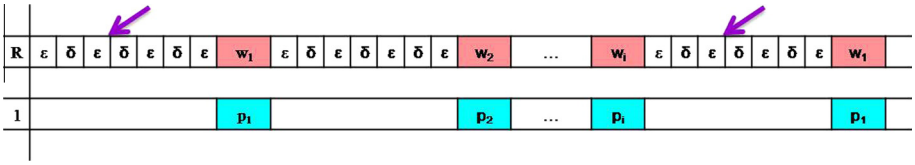


**Fig. 5.** Cycle time for a single stage single machine cell.

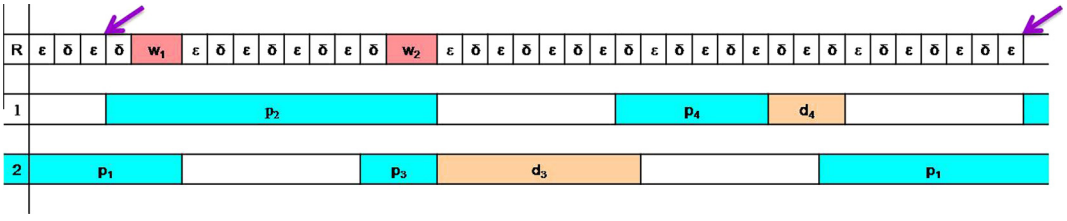


**Fig. 6.** Cycle time for a single stage multi-machine cell.

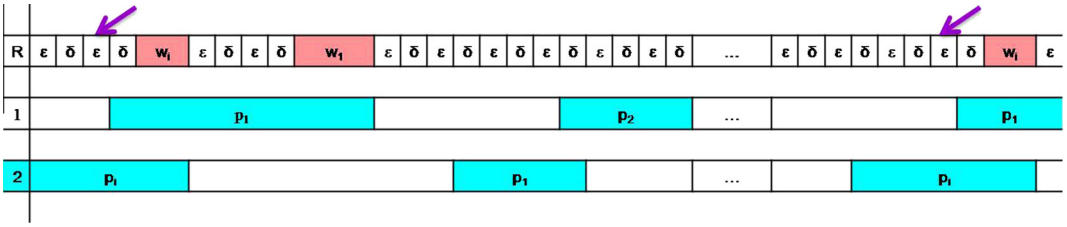


**Fig. 7.** Cycle time for a multi-stage single machine cell.

**Table 5**
Parameter values which are constructed according to the observations.

| | |
|---|---|
| Number of parts | 4–5–10–15–20–25–50–100–150 |
| Average processing times | 0–200, 0–300, 0–500 units of time |
| Load/unload times | 1%, 6%, 10% over the average processing time values |
| Travel times | 1%, 6%, 10% over the average processing time values |
| Initial temperatures | 100, 168, 196, 313, 665 |

the robot has to deal with more than one machine and this causes idle times to occur in the system. Similar to Case 2, the total time needed for each machine equals to $\sum_i p_{ik} + 4\epsilon + 3\delta + \sum_i d_{ik}$ and taking the total idle time to be zero, the lower bound for each machine becomes $\sum_i p_{ik} + 4\epsilon + 3\delta$. In multi-stage environments, all the parts are processed at all of the stages and each stage is responsible for its own operations, therefore there is no processing time sharing between the machines, instead, the cycle time value is determined according to the latest completed stage's completion time value. So, the lower bound value equals to $\max_k\{\sum_i p_{ik} + n(4\epsilon + 3\delta)\}$. *Case 4.* Finally, when we consider multi-stage cells with more than one machine at least one of the stages. It is obvious that this system is a combination of the two previous cases. Taking the stages into account one by one, the total time needed for the stage with a single machine is $\sum_i p_{ik} + 4\epsilon + 3\delta$, whereas for the stages with more than one machine it becomes $(\sum_k \sum_i p_{ik} + n(4\epsilon + 3\delta))/m$. Since we know that in multi-stage systems, the cycle time is determined according to the latest completed stage's completion time value, it can easily be said that in our system the cycle time will be at least $\max_k\{(\sum_i p_{ik} + n(4\epsilon + 3\delta))/m_k\}$, where $k$ defines the stages and $m_k$ defines the number of machines at stages. Since our cell has the flexibility property which allows some parts not to be processed on some stages, we use the notation $n_k$ to represent the part number to be processed on stage $k$, instead of $n$. Thus, the lower bound value for this system becomes equal to the value $\max_k\{(\sum_i p_{ik} + n_k(4\epsilon + 3\delta))/m_k\}$. □

In order to test the algorithms performance, we designed an extensive experimental setting. Hall, Kamoun, and Sriskandarajah (1997) consider multiple part-type scheduling problems in robotic cells and observe that the system returns to its initial state after one or two MPSs. Similarly, in our study, a cyclic production plan is obtained after two or three MPSs.

The considered problem has some basic parameters, namely, the number of parts, the average processing time, the load/unload time, the travel time, the number of stages and the number of machines at each of these stages. In this study, we focus on 2-stage robotic cells, at which there is only 1 machine in the first stage and 2 machines in the second one. In Table 5, the remaining factors which affect the complexity of the problems along with their associated levels are given. These levels are constructed according to the observations obtained throughout the preliminary trials of the proposed algorithm. According to the given parameters, we produced 10 instances for each type of problem classes and obtained a total of 360 instances. We execute the algorithm for 5 times for each of these instances and take the best result into consideration.

The algorithm is executed on Microsoft Visual Studio 2010 using the programming language of C++ on a Pentium V 2.80 GHz. 1 GB computer. We have obtained satisfying solutions

| | | P = 0-200 | | | | | P = 0-300 | | | | | P = 0-500 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Temperature | | 100 | 168 | 196 | 313 | 665 | 100 | 168 | 196 | 313 | 665 | 100 | 168 | 196 | 313 | 665 |
| Number of Parts | 4 | 5.13% | 5.13% | 5.13% | 5.13% | 5.13% | 6.27% | 6.27% | 6.27% | 6.27% | 6.27% | 8.09% | 8.03% | 8.03% | 8.09% | 8.09% |
| | 5 | 4.73% | 4.73% | 4.73% | 4.73% | 4.71% | 9.97% | 10.00% | 9.97% | 9.97% | 9.97% | 8.69% | 8.56% | 8.56% | 8.56% | 8.56% |
| | 10 | 4.26% | 4.22% | 4.18% | 4.15% | 4.11% | 2.66% | 2.56% | 2.46% | 2.39% | 2.32% | 3.95% | 3.91% | 3.84% | 3.82% | 3.78% |
| | 15 | 2.96% | 2.91% | 2.84% | 2.79% | 2.70% | 2.05% | 1.97% | 1.90% | 1.87% | 1.80% | 4.48% | 4.40% | 4.35% | 4.28% | 4.21% |
| | 20 | 1.85% | 1.82% | 1.79% | 1.73% | 1.69% | 3.59% | 3.55% | 3.50% | 3.46% | 3.38% | 3.37% | 3.32% | 3.26% | 3.24% | 3.20% |
| | 25 | 2.32% | 2.27% | 2.22% | 2.17% | 2.14% | 0.92% | 0.87% | 0.83% | 0.79% | 0.77% | 3.48% | 3.44% | 3.40% | 3.36% | 3.28% |
| | 50 | 2.24% | 2.21% | 2.19% | 2.16% | 2.12% | 2.84% | 2.82% | 2.80% | 2.78% | 2.75% | 3.61% | 3.57% | 3.54% | 3.52% | 3.48% |
| | 100 | 2.83% | 2.81% | 2.81% | 2.80% | 2.77% | 1.82% | 1.81% | 1.81% | 1.80% | 1.79% | 4.84% | 4.83% | 4.82% | 4.81% | 4.80% |
| | 150 | 3.10% | 3.08% | 3.06% | 3.05% | 3.04% | 0.88% | 0.87% | 0.87% | 0.86% | 0.85% | 2.76% | 2.75% | 2.75% | 2.74% | 2.73% |
| Load / Unload Time (Epsilon) | low | 1.52% | 1.49% | 1.47% | 1.45% | 1.41% | 1.70% | 1.67% | 1.61% | 1.49% | 1.54% | 2.60% | 2.57% | 2.54% | 2.52% | 2.49% |
| | medium | 3.14% | 3.12% | 3.08% | 3.04% | 3.00% | 3.84% | 3.80% | 3.77% | 3.74% | 3.71% | 5.11% | 5.07% | 5.04% | 5.01% | 4.97% |
| | high | 5.15% | 5.12% | 5.10% | 5.08% | 5.05% | 4.80% | 4.77% | 4.75% | 4.74% | 4.72% | 6.71% | 6.63% | 6.61% | 6.61% | 6.59% |
| Travel Time (Delta) | low | 1.50% | 1.47% | 1.45% | 1.43% | 1.39% | 2.33% | 2.29% | 2.24% | 2.21% | 2.17% | 2.44% | 2.34% | 2.31% | 2.31% | 2.27% |
| | medium | 3.24% | 3.23% | 3.19% | 3.17% | 3.14% | 2.67% | 2.65% | 2.60% | 2.59% | 2.55% | 3.96% | 3.93% | 3.90% | 3.87% | 3.84% |
| | high | 5.05% | 5.03% | 5.01% | 4.97% | 4.94% | 5.33% | 5.31% | 5.30% | 5.27% | 5.24% | 8.02% | 8.00% | 7.98% | 7.96% | 7.93% |
| Epsilon & Delta | l+l | 0.79% | 0.76% | 0.74% | 0.71% | 0.69% | 1.74% | 1.71% | 1.61% | 1.58% | 1.53% | 1.93% | 1.89% | 1.86% | 1.84% | 1.81% |
| | l+m | 1.55% | 1.55% | 1.52% | 1.51% | 1.46% | 1.55% | 1.52% | 1.46% | 1.44% | 1.40% | 2.38% | 2.35% | 2.33% | 2.32% | 2.29% |
| | l+h | 2.20% | 2.16% | 2.14% | 2.12% | 2.09% | 1.81% | 1.78% | 1.77% | 1.73% | 1.70% | 3.49% | 3.47% | 3.43% | 3.41% | 3.36% |
| | m+l | 1.70% | 1.67% | 1.64% | 1.63% | 1.57% | 2.54% | 2.47% | 2.44% | 2.40% | 2.38% | 2.69% | 2.66% | 2.63% | 2.60% | 2.54% |
| | m+m | 2.94% | 2.92% | 2.87% | 2.82% | 2.79% | 2.94% | 2.92% | 2.87% | 2.86% | 2.81% | 3.49% | 3.43% | 3.39% | 3.36% | 3.31% |
| | m+h | 4.78% | 4.76% | 4.74% | 4.67% | 4.63% | 6.03% | 6.01% | 6.00% | 5.97% | 5.95% | 9.14% | 9.12% | 9.09% | 9.07% | 9.06% |
| | h+l | 2.01% | 2.00% | 1.96% | 1.94% | 1.90% | 2.71% | 2.67% | 2.66% | 2.65% | 2.62% | 2.69% | 2.48% | 2.45% | 2.48% | 2.45% |
| | h+m | 5.25% | 5.21% | 5.20% | 5.18% | 5.16% | 3.53% | 3.50% | 3.47% | 3.47% | 3.45% | 6.02% | 6.00% | 5.97% | 5.94% | 5.92% |
| | h+h | 8.19% | 8.15% | 8.14% | 8.13% | 8.11% | 8.16% | 8.14% | 8.13% | 8.11% | 8.08% | 11.43% | 11.42% | 11.40% | 11.40% | 11.38% |
| Average | | 3.27% | 3.24% | 3.22% | 3.19% | 3.16% | 3.45% | 3.42% | 3.38% | 3.36% | 3.32% | 4.81% | 4.76% | 4.73% | 4.71% | 4.68% |

**Fig. 8.** Observations for Neighborhood 1 (random neighborhood). Solutions are grouped according to their values and marked by blue, pink and yellow colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

| | | P= 0-200 | | | | | P= 0-300 | | | | | P= 0-500 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Temperature | | 100 | 168 | 196 | 313 | 665 | 100 | 168 | 196 | 313 | 665 | 100 | 168 | 196 | 313 | 665 |
| Number of Parts | 4 | 5.05% | 5.05% | 5.05% | 5.05% | 5.05% | 6.42% | 6.42% | 6.42% | 6.42% | 6.42% | 7.91% | 7.91% | 7.91% | 7.91% | 7.91% |
| | 5 | 4.71% | 4.71% | 4.71% | 4.71% | 4.71% | 9.88% | 9.88% | 9.88% | 9.88% | 9.88% | 8.20% | 8.13% | 8.13% | 8.12% | 8.12% |
| | 10 | 4.26% | 4.18% | 4.14% | 4.09% | 4.07% | 2.70% | 2.57% | 2.50% | 2.40% | 2.36% | 3.86% | 3.83% | 3.82% | 3.80% | 3.75% |
| | 15 | 3.07% | 2.96% | 2.90% | 2.85% | 2.80% | 1.96% | 1.88% | 1.83% | 1.78% | 1.74% | 4.23% | 4.18% | 4.15% | 4.09% | 4.04% |
| | 20 | 1.87% | 1.85% | 1.81% | 1.77% | 1.71% | 3.55% | 3.51% | 3.48% | 3.44% | 3.35% | 3.39% | 3.35% | 3.32% | 3.26% | 3.19% |
| | 25 | 2.38% | 2.30% | 2.25% | 2.22% | 2.16% | 0.85% | 0.80% | 0.78% | 0.76% | 0.71% | 3.53% | 3.48% | 3.45% | 3.39% | 3.33% |
| | 50 | 2.23% | 2.21% | 2.18% | 2.16% | 2.13% | 2.87% | 2.85% | 2.83% | 2.81% | 2.80% | 3.62% | 3.58% | 3.56% | 3.54% | 3.51% |
| | 100 | 2.83% | 2.80% | 2.78% | 2.77% | 2.75% | 1.83% | 1.81% | 1.80% | 1.77% | 1.76% | 4.88% | 4.86% | 4.85% | 4.85% | 4.83% |
| | 150 | 3.08% | 3.08% | 3.06% | 3.05% | 3.03% | 0.87% | 0.86% | 0.86% | 0.85% | 0.83% | 2.78% | 2.77% | 2.76% | 2.75% | 2.74% |
| Load / Unload Time (Epsilon) | low | 1.52% | 1.48% | 1.46% | 1.43% | 1.41% | 1.66% | 1.63% | 1.60% | 1.52% | 1.53% | 2.61% | 2.55% | 2.54% | 2.51% | 2.48% |
| | medium | 3.16% | 3.12% | 3.08% | 3.05% | 3.01% | 3.86% | 3.81% | 3.79% | 3.76% | 3.74% | 5.07% | 5.05% | 5.03% | 5.00% | 4.97% |
| | high | 5.14% | 5.11% | 5.09% | 5.08% | 5.05% | 4.78% | 4.75% | 4.74% | 4.71% | 4.69% | 6.45% | 6.43% | 6.41% | 6.39% | 6.36% |
| Travel Time (Delta) | low | 1.50% | 1.46% | 1.44% | 1.42% | 1.39% | 2.35% | 2.31% | 2.28% | 2.23% | 2.20% | 2.37% | 2.32% | 2.30% | 2.27% | 2.22% |
| | medium | 3.27% | 3.22% | 3.19% | 3.17% | 3.14% | 2.63% | 2.59% | 2.56% | 2.55% | 2.52% | 3.97% | 3.94% | 3.93% | 3.90% | 3.88% |
| | high | 5.06% | 5.03% | 5.00% | 4.97% | 4.94% | 5.32% | 5.30% | 5.29% | 5.26% | 5.23% | 7.79% | 7.77% | 7.76% | 7.73% | 7.70% |
| Epsilon & Delta | l+l | 0.80% | 0.76% | 0.74% | 0.72% | 0.69% | 1.75% | 1.73% | 1.67% | 1.62% | 1.57% | 2.00% | 1.91% | 1.88% | 1.85% | 1.80% |
| | l+m | 1.57% | 1.50% | 1.48% | 1.46% | 1.44% | 1.49% | 1.44% | 1.42% | 1.40% | 1.39% | 2.35% | 2.33% | 2.32% | 2.30% | 2.28% |
| | l+h | 2.21% | 2.17% | 2.15% | 2.12% | 2.11% | 1.74% | 1.73% | 1.71% | 1.69% | 1.63% | 3.47% | 3.42% | 3.42% | 3.39% | 3.35% |
| | m+l | 1.69% | 1.65% | 1.63% | 1.61% | 1.56% | 2.57% | 2.50% | 2.48% | 2.44% | 2.42% | 2.56% | 2.55% | 2.53% | 2.51% | 2.46% |
| | m+m | 2.98% | 2.94% | 2.90% | 2.87% | 2.82% | 2.93% | 2.89% | 2.86% | 2.84% | 2.79% | 3.49% | 3.46% | 3.43% | 3.40% | 3.39% |
| | m+h | 4.80% | 4.76% | 4.72% | 4.67% | 4.64% | 6.08% | 6.06% | 6.03% | 6.00% | 5.99% | 9.16% | 9.14% | 9.13% | 9.10% | 9.06% |
| | h+l | 2.02% | 1.98% | 1.95% | 1.93% | 1.92% | 2.73% | 2.69% | 2.68% | 2.63% | 2.62% | 2.54% | 2.51% | 2.48% | 2.45% | 2.41% |
| | h+m | 5.25% | 5.21% | 5.18% | 5.18% | 5.16% | 3.46% | 3.43% | 3.41% | 3.40% | 3.38% | 6.07% | 6.03% | 6.03% | 6.00% | 5.98% |
| | h+h | 8.17% | 8.15% | 8.14% | 8.12% | 8.09% | 8.15% | 8.13% | 8.12% | 8.09% | 8.07% | 10.74% | 10.74% | 10.73% | 10.72% | 10.70% |
| Average | | 3.28% | 3.24% | 3.21% | 3.19% | 3.16% | 3.43% | 3.40% | 3.38% | 3.35% | 3.32% | 4.71% | 4.68% | 4.66% | 4.63% | 4.60% |

**Fig. 9.** Observations for Neighborhood 2 (improved neighborhood). Solutions are grouped according to their values and marked by blue, pink and yellow colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

very quickly even for large sized problems. For the largest problem sizes of 150 parts, the average execution time is observed to be around 60 min. In order to evaluate the quality of the obtained solutions, we have compared our method with the proposed lower bound value using the following equation in order to calculate the deviation:

$$\%\ deviation = \frac{L.B. - C(Algorithm)}{C(Algorithm)} \times 100 \tag{26}$$

The results are depicted in Figs. 8 and 9. In Figs. 8 and 9, letters '*l*', '*m*' and '*h*' define low, medium and high parameter levels, respectively. The terms '$l+l$', '$l+m$', etc. given under the heading of

| | | Ti=100 | Ti=168 | Ti=196 | Ti=313 | Ti=665 |
|---|---|---|---|---|---|---|
| Processing Time | Low | 3.27% | 3.24% | 3.22% | 3.19% | 3.16% |
| | Medium | 3.45% | 3.42% | 3.38% | 3.36% | 3.32% |
| | High | 4.81% | 4.76% | 4.73% | 4.71% | 4.68% |
| Epsilon | Low | 1.94% | 1.91% | 1.87% | 1.82% | 1.81% |
| | Medium | 4.03% | 4.00% | 3.96% | 3.93% | 3.89% |
| | High | 5.55% | 5.51% | 5.49% | 5.48% | 5.45% |
| Delta | Low | 2.09% | 2.03% | 2.00% | 1.98% | 1.94% |
| | Medium | 3.29% | 3.27% | 3.23% | 3.21% | 3.18% |
| | High | 6.14% | 6.11% | 6.09% | 6.07% | 6.04% |
| Epsilon & Delta | Low | 1.87% | 1.84% | 1.80% | 1.78% | 1.74% |
| | Medium | 2.70% | 2.65% | 2.62% | 2.60% | 2.56% |
| | High | 6.95% | 6.92% | 6.90% | 6.88% | 6.86% |

**Fig. 10.** Parameter based results for Neighborhood 1.

| | | Ti=100 | Ti=168 | Ti=196 | Ti=313 | Ti=665 |
|---|---|---|---|---|---|---|
| Processing Time | Low | 3.28% | 3.24% | 3.21% | 3.19% | 3.16% |
| | Medium | 3.43% | 3.40% | 3.38% | 3.35% | 3.32% |
| | High | 4.71% | 4.68% | 4.66% | 4.63% | 4.60% |
| Epsilon | Low | 1.93% | 1.89% | 1.86% | 1.82% | 1.81% |
| | Medium | 4.03% | 3.99% | 3.97% | 3.94% | 3.90% |
| | High | 5.46% | 5.43% | 5.41% | 5.39% | 5.37% |
| Delta | Low | 2.07% | 2.03% | 2.00% | 1.97% | 1.94% |
| | Medium | 3.29% | 3.25% | 3.23% | 3.20% | 3.18% |
| | High | 6.06% | 6.03% | 6.02% | 5.99% | 5.96% |
| Epsilon & Delta | Low | 1.86% | 1.82% | 1.79% | 1.77% | 1.73% |
| | Medium | 2.68% | 2.64% | 2.62% | 2.59% | 2.56% |
| | High | 6.88% | 6.85% | 6.83% | 6.81% | 6.79% |

**Fig. 11.** Parameter based results for Neighborhood 2.

"*Epsilon&Delta*" indicate the situations when these two parameters (load/unload and travel times) are simultaneously considered.

As can be seen from these tables, average deviation values are observed to be around 3.2% for low levels of processing time values for both of the two algorithms, whereas it becomes around 3.3% for medium levels and around 4.6–4.7% for high levels. The increase in the values arises from the fact that sequencing and scheduling problem, and also reaching to the defined lower bound value, gets harder as the difference between the processing time values gets higher.

Comparisons are performed with respect to different levels of the previously defined parameter values; problem size, load/unload and travel times. These observations are given in Figs. 10 and 11. It can be seen from the tables that the two crucial parameters affecting the performances are the load/unload and travel time values. In cases where the sum of these two parameters is high, the deviation gets also higher. Considering the two parameters individually, the travel time seems to be more important. This result is also expected since the total number of load/unload movements is independent from the algorithm whereas the travel time value is not, since throughout the solution procedure, it is aimed to find the best possible robot move sequence having smallest travel times.

Neighborhood structures do not make big differences in the average deviation values. However, especially for the case with high processing time, high load/unload time and high travel times, this value seems to be around 0.08%. The reason for the improvement in the deviation values in this case is that the second neigh-

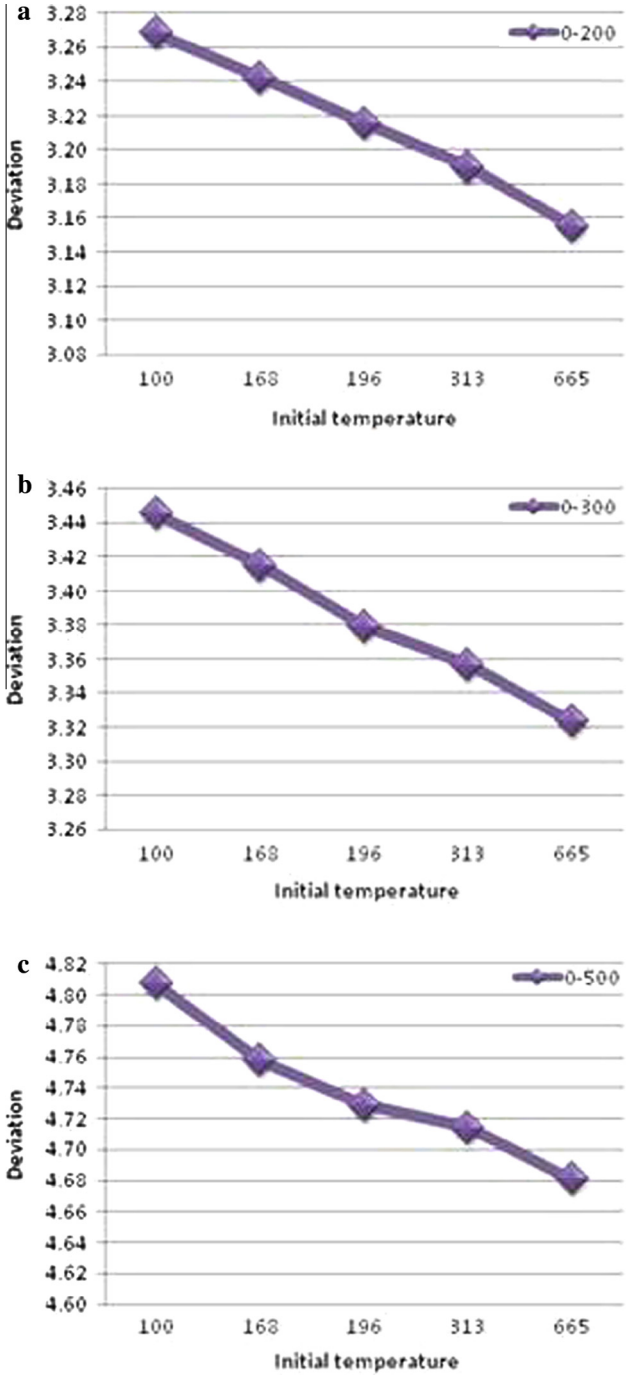


**Fig. 12.** Initial temperature based results for Neighborhood 1.

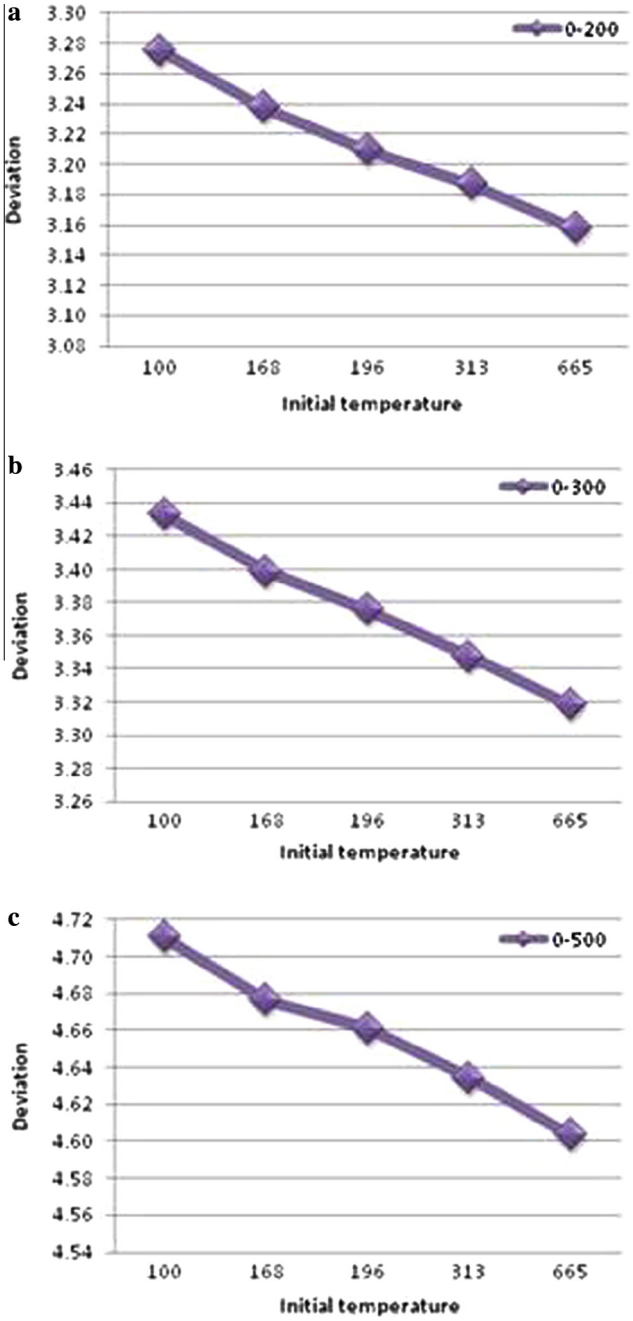


**Fig. 13.** Initial temperature based results for Neighborhood 2.

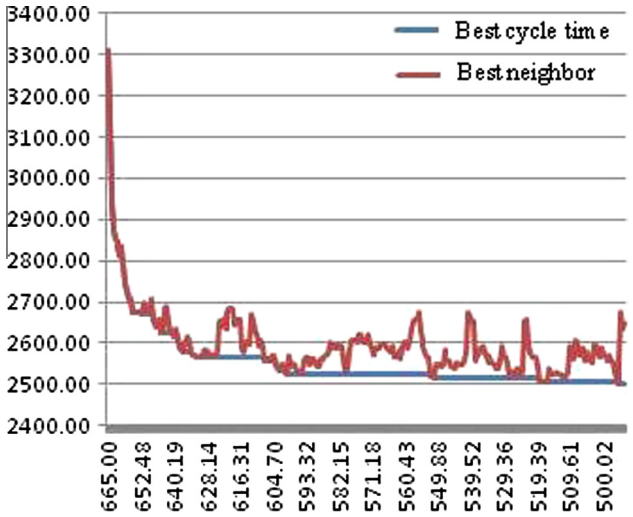


**Fig. 14.** Best cycle time and best neighbor results in terms of iterations for early temperature values. The horizontal axis represents the temperature values whereas the vertical corresponds to the cycle times.

borhood, which is the improved one, takes the processing times into account more attentively.

Comparing the results obtained for different starting temperatures, it is observed that for high temperatures the deviations are much more smaller. This is also an expected result as depicted in Figs. 12 and 13.

It is observed that the results related to the initial temperature values are not so different. In order to take a closer look on the improvements obtained with respect to the temperature value, a randomly chosen instance is solved using the initial temperature of $T = 665$, which is the largest value considered, and the 'best cycle time' solutions obtained at each iteration are recorded together with the 'best neighbors' constructed. The results of this experiment are provided in Appendix A. In Fig. 14, the first part of the figure given in Appendix A is enlarged in order to see the high rate of change observed during the earlier stages of the algorithm.

As can be seen from Fig. 14, in the beginning, the rate of change over the objective function values is extremely high, whereas in the later steps no significant difference is observed. Besides, consistent with the SA logic, as the temperature decreases, worse solutions are no longer allowed to be chosen. For example, when the temperature is around 140.47–125.71, neighbors with the objective function value of 2750 may be accepted, but when the temperature gets lower even the biggest cycle time solutions are below the value of 2700.

When we take a closer look on the best solutions obtained throughout the solution procedure, we observe that the change ratios over the cycle time values are high in the early iterations whereas this ratio gets smaller in the later temperatures. This difference can be followed from the cycle time values given in Fig. 15 together with the ratios given in Fig. 16.

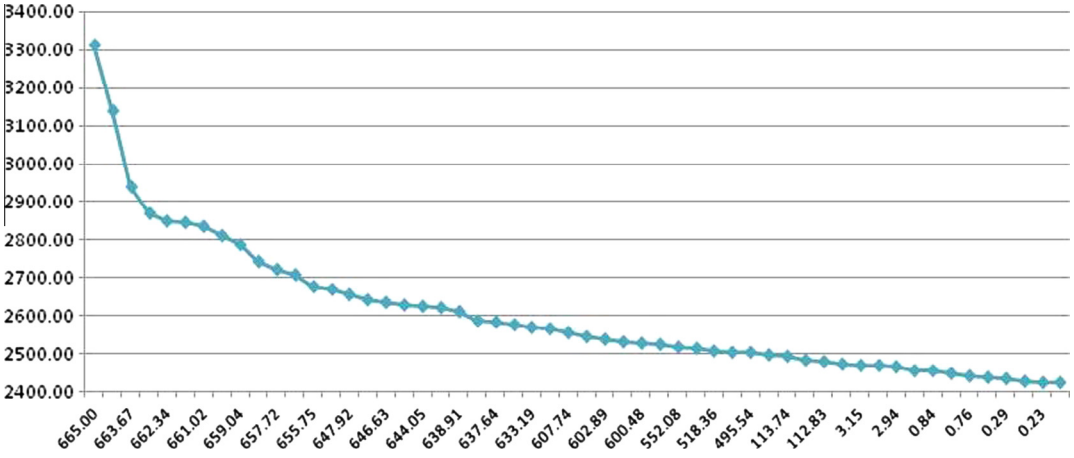


**Fig. 15.** Iteration based change points for cycle time values. The horizontal axis represents the temperature values whereas the vertical corresponds to the cycle times.

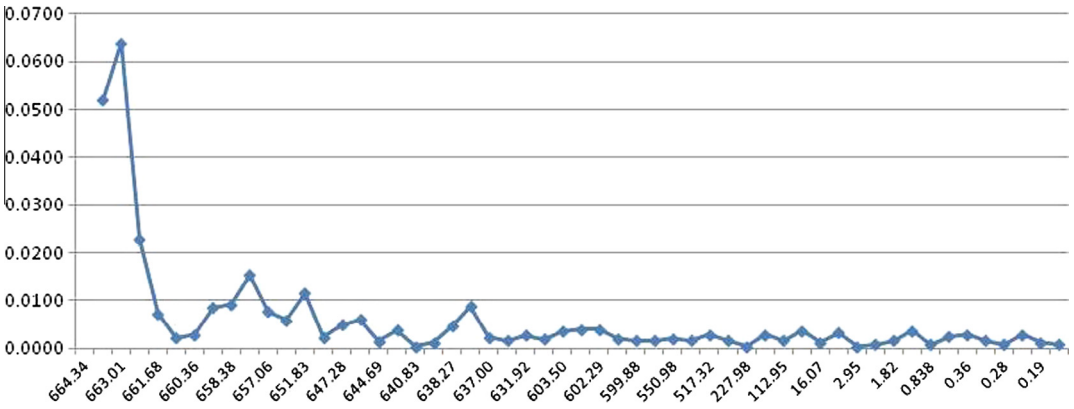


**Fig. 16.** Iteration based change ratios for cycle time values. The horizontal axis represents the temperature values whereas the vertical corresponds to the cycle time improvements.

## 6. Conclusion

In this study, we focus on the scheduling problem arising in constant travel time hybrid flexible flow shops in which lots that contain different types of parts are repeatedly processed and the transportation of the parts between the machines is performed by a robot. Consistent with the definition of hybrid flow shops, in this system there are at least two production stages and more than one machine for at least one of these stages. Consistent with multiple part-type scheduling, we minimize the average time to produce one MPS in a cyclic environment. The challenge is threefold: (a) determining a part sequence, (b) assigning the parts to the machines, (c) choosing a robot move sequence. As an additional challenge, due to the flexibility property of the system, some parts are allowed not to be processed on some of the stages.

The solution that we look for defines the movements of the robot exactly, giving the part to be carried/loaded/unloaded together with the related machine. The problem is first modeled as a TSP with a distance matrix which contains decision variables as well as problem parameters. This model is more general than the classical TSP and requires a great amount of computational effort even for the small sized problems. Consequently, we focused our attention on heuristic approaches and proposed an SA based heuristic algorithm. In order to evaluate our algorithm, we constructed a lower bound value which can be used regardless of the problem parameters, i.e., number of stages/machines/parts, etc. We solved randomly produced test problems using the constructed algorithm, compared the results with this lower bound value, and concluded that our algorithm is very competitive, providing results with very small optimality gaps for almost all the examples in our experimental design. Sensitivity analyzes which confirmed the robustness of our heuristic methodology are conducted.

As far as the authors know, this is the first study to consider single robot usage and multiple part-type production in hybrid flow shop scheduling literature. There is a room for improvement and future research potential of this study. One possible extension may be to consider a structure with each stage containing at least two machines. In such a case, the complexity of the problem will no doubt increase dramatically. Another potential future research direction is to consider the technological differences between the machines, studying the non-identical machine case for this problem. Having non-identical machines at stages, any machine may not be able to perform all the processing of a part and some specific operations of the parts will have to be assigned to certain machines. One may also consider to use more than one robot in order to decrease the idle time values albeit with increasing the number of possible movements in the system.

## Appendix A. Numerical example for the case of initial temperature value of 665

In order to see the change on the 'best cycle time' and 'best neighbor' results obtained through the iterations of the algorithm, a randomly chosen instance is solved using the initial temperature of $T = 665$, which is the largest value considered. As can be seen from Fig. A.1, there is a high rate of change during the earlier stages of the algorithm, whereas it gets smoother as the temperature decreases.

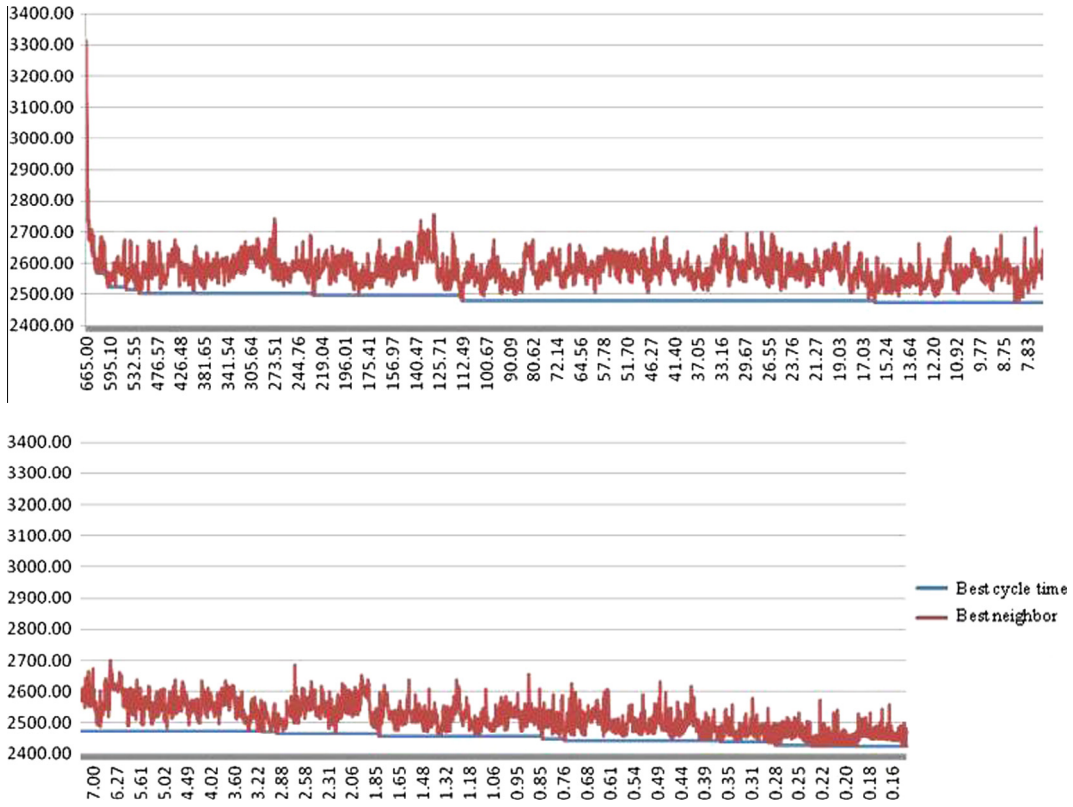


**Fig. A.1.** Best cycle time and best neighbor results in terms of iterations. The horizontal axis represents the temperature values whereas the vertical corresponds to the cycle times.

## References


Batur, G. D., Karaşan, O., & Aktürk, M. S. (2012). Multiple part-type scheduling in flexible robotic cells. *International Journal of Production Economics, 135*(2), 726–740.

Behnamian, J., Zandieh, M., & Fatemi Ghomi, S. M. T. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. *Expert Systems with Applications, 36*, 9637–9644.

Brauner, N. (2008). Identical part production in cyclic robotic cells: Concepts, overview and open questions. *Discrete Applied Mathematics, 156*, 2480–2492.

Crama, Y., Kats, V., Van de Klundert, J., & Levner, E. (2000). Cyclic scheduling in robotic flowshops. *Annals of Operations Research, 96*, 97–124.

Dawande, M., Geismar, H. N., Sethi, S. P., & Sriskandarajah, C. (2005). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling, 8*(5), 387–426.

Dawande, M., Sriskandarajah, C., & Sethi, S. P. (2002). On throughput maximization in constant travel-time robotic cells. *Manufacturing and Service Operations Management, 4*(4), 296–312.

Dessouky, M. M., Dessouky, M. I., & Verma, S. K. (1998). Flowshop scheduling with identical jobs and uniform parallel machines. *European Journal of Operational Research, 109*, 620–631.

Elmi, A., & Topaloglu, S. (2013). A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. *Computers and Operations Research, 40*, 2543–2555.

Gültekin, H., Aktürk, M. S., & Karaşan, O. E. (2006). Robotic cell scheduling with tooling constraints. *European Journal of Operational Research, 174*, 777–796.

Gültekin, H., Karaşan, O. E., & Aktürk, M. S. (2009). Pure cycles in flexible robotic cells. *Computers and Operations Research, 36*(2), 329–344.

Gupta, J. N. D. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society, 39*(4), 359–364.

Hall, N. G., Kamoun, H., & Sriskandarajah, C. (1997). Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research, 109*(1), 43–65.

Hooda, N., & Dhingra, A. K. (2011). Flow shop scheduling using simulated annealing: A review. *International Journal of Applied Engineering Research, 2*(1), 234–249.

Jabbarizadeh, F., Zandieh, M., & Talebi, D. (2009). Hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints. *Computers and Industrial Engineering, 57*(3), 949–957.

Kamalabadi, I. N., Gholami, S., & Mirzaei, A. H. (2007). Considering a cyclic multiple-part type three-machine robotic cell problem. In *IEEM, IEEE international conference* (pp. 704–708).

Kamoun, H., Hall, N. G., & Sriskandarajah, C. (1999). Scheduling in robotic cells: Heuristics and cell design. *Operations Research, 47*(6), 821–835.

Karaoglan, I., Altıparmak, F., Kara, I., & Dengiz, B. (2011). A branch and cut algorithm for the location-routing problem with simultaneous pickup and delivery. *European Journal of Operational Research, 211*, 318–332.

Kim, D. W., Kim, K. H., Jang, W., & Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer Integrated Manufacturing, 18*, 223–231.

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Kurz, M. E., & Aşkın, R. G. (2003). Comparing scheduling rules for flexible flow lines. *International Journal of Production Economics, 85*, 371–388.

Lee, Z. J., Lin, A. W., & Ying, K. C. (2010). Scheduling jobs on dynamic parallel machines with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology, 47*, 773–781.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering, 37*(1–2), 57–61.

Low, C., Yeh, J. Y., & Huang, K. I. (2004). A robust simulated annealing heuristic for flow shop scheduling problems. *International Journal of Advanced Manufacturing Technology, 23*, 762–767.

Metropolis, N., Rosenbluth, A., & Resenbluth, M. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*, 1087–1092.

Miller, C., Tucker, A., & Zemlin, R. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM), 7*(4), 326–329.

Nowicki, E., & Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research, 106*, 226–253.

Quadt, D., & Kuhn, D. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research, 178*(3), 686–698.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*, 1–18.

Sawik, T. (2012). Batch versus cyclic scheduling of flexible flow shops by mixed-integer programming. *International Journal of Production Research, 50*(18), 5017–5034.

Vignier, A., Billaut, J. C., & Proust, C. (2010). Les problémes dordonnancement de type flow-shop hybride. état de lart. *RAIRO Operations Research, 33*(2), 117–183.

Wang, H. (2005). Flexible flow shop scheduling: Optimum, heuristics and artificial intelligence solutions. *Expert Systems, 22*(2), 78–85.

Yaurima, V., Burtseva, L., & Tchernykh, A. (2009). Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Computers and Industrial Engineering, 56*(4), 1452–1463.

Zandieh, M., & Karimi, N. (2011). An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times. *Journal of Intelligent Manufacturing, 22*, 979–989.

Zegordi, A. H., Itoh, K., & Enkawa, T. (1995). Minimizing makespan for flow shop scheduling by combining simulated annealing with sequencing knowledge. *European Journal of Operational Research, 85*, 515–531.

Zhang, R., & Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing, 10*, 79–89.