algorithms

MDPI

Article

Blocking Cyclic Job-Shop Scheduling Problems

Atabak Elmi [1], Dhananjay R. Thiruvady [1,*] and Andreas T. Ernst [2]

[1] School of Information Technology, Faculty of Science, Engineering and Built Environment, Deakin University, Geelong, VIC 3125, Australia
[2] School of Mathematics, Faculty of Science, Monash University, Melbourne, VIC 3800, Australia
* Correspondence: dhananjay.thiruvady@deakin.edu.au

Abstract: Cyclic scheduling is of vital importance in a repetitive discrete manufacturing environment. We investigate scheduling in the context of general cyclic job shops with blocking where there are no intermediate buffers between the machines. We also consider sequence-dependent setups (anticipatory and nonanticipatory), which commonly appear in different manufacturing environments. The choice of blocking condition, that is whether the sequence-dependent setups are anticipatory or not, significantly impacts the optimal schedules. We provide a novel mixed-integer programming (MIP) model for the above problem, namely blocking cyclic job-shop scheduling. Furthermore, we study the impact of sequence-dependent setups in this research. The problem is analysed in detail with respect to anticipatory and nonanticipatory setups and the efficiency of the proposed model is investigated via a computational study that is conducted on a set of randomly generated problem instances. The proposed MIP models are capable of solving small-to-medium-sized problems. Moreover, the analysis presented demonstrates that anticipatory setups directly affect blocking conditions, since intermediate buffers between the machines are not present. Hence, in systems with anticipatory setups, cycle times increase to a greater extent compared to systems with nonanticipatory setups.

Keywords: cyclic scheduling; job-shop scheduling; sequence-dependent setups; blocking conditions; mixed-integer programming

Citation: Elmi, A.; Thiruvady, D.R.; Ernst, A.T. Blocking Cyclic Job-Shop Scheduling Problems. *Algorithms* **2022**, *15*, 375. https://doi.org/10.3390/a15100375

Academic Editor: Maciej Drozdowski

Received: 1 September 2022
Accepted: 10 October 2022
Published: 14 October 2022

1. Introduction

This study is concerned with a manufacturing system, namely cyclic job shop with blocking conditions. The cyclic job-shop scheduling problem is an extension of the well-known job-shop scheduling problem (JSSP) [1,2]. The original JSSP involves scheduling jobs, each of which consist of a number of operations, and the objective is to minimise the length of the schedule, or the *makespan*. Moreover, manufacturing systems often require schedules that are cyclic in nature and leading to additional complexities [3,4]. Cyclic or periodic scheduling produce schedules that are executed repeatedly in exactly the same manner on one or more machines [5]. The objective functions mainly used in cyclic scheduling are minimising *cycle time*, *work-in-process*, and mixes of them. The cycle time is the time period between two successive occurrences of the same operations, and the work-in-process is the number of jobs processed at the same time. The aim of cyclic scheduling in this research is to find a schedule with the minimum cycle time (common in manufacturing systems) [6]. Within the system, a set of jobs, known as the minimal part set (MPS) (For example, if the requirements are 1000 parts of type A, 3000 parts of type B and 2000 parts of type C, the MPS can be chosen as (A, B, B, B, C, C) and we repeat the production of the same MPS 1000 times.) is executed cyclically, in order to meet production requirements [7,8]. Referring to parts as jobs, for each MPS, the jobs go through the machines in exactly the same pattern, and each machine thereby processes the operations in an identical way in each cycle. An alternative view is that a single MPS is sent into a production line where exactly one MPS is completed and unloaded from it. The primary performance measure is the cycle time, or equivalently its reciprocal, the throughput rate, which measures how often the MPSs are

produced in a fixed time interval. The JSSP itself is known to be among the most complex combinatorial optimisation problems [9], and the cyclic variant considered in this study incorporates further complexity.

Moreover, in several manufacturing systems, there exist production processes where no intermediate buffers exist between machines due to technical requirements or process characteristics. In the context of the JSSP, this implies that a job that completes processing on a machine has to remain on the same machine until the next machine in sequence is available for processing. Since the job cannot leave the machine that it is executing on, it effectively blocks the machine by not allowing other parts to enter it. In the literature, this is referred to as a *blocking condition*, which this study is also concerned with. Figure 1 presents an example of a blocking condition where job 1 (processing time of 30 units) needs to be processed on machine 1 followed by machine 2. We see in Figure 1a, if machine 2 is available to load and start executing job 1 when job 1 completes, then job 1 immediately moves to machine 2. Machine 1 is now available for other incoming jobs. However, if machine 2 is busy when job 1 is ready to move (Figure 1b), then job 1 stays on machine 1 thereby blocking it until machine 2 becomes free. Hence, machine 1 is also blocked for the time period that job 1 stays on it.

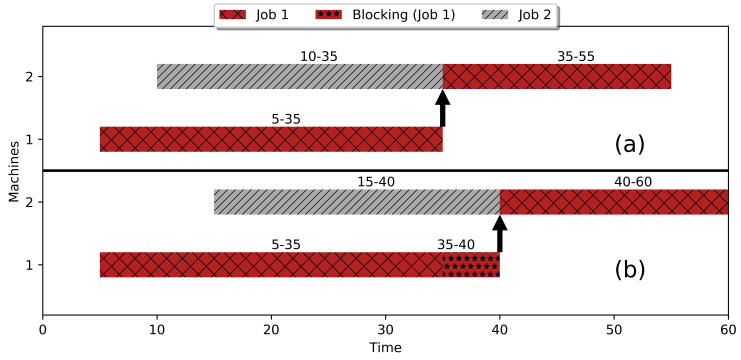

Figure 1. Blocking conditions in job-shop scheduling.

There are several well-studied scheduling problems with blocking conditions, including the manufacturing of concrete blocks [10], scheduling in the chemical industry [11], scheduling in the iron and steel industry [12], industrial waste and the manufacturing of metallic parts [13], and train scheduling [14].

A production line, structured as a job shop requires that: (i) the operations of each job of the MPS are assigned to the machines in advance and (ii) the routing (order of processing) of each job passing through the machines, not necessarily the same for each job, is known and fixed. The aim of the cyclic JSSP is to find the processing order in which the operations are repetitively processed on each machine a very large number of times. A sequence of the jobs, together with the starting times of all the operations, is referred to as a cyclic or periodical schedule. A regular time interval in which all the operations are repeated is referred to as the period or cycle time.

There have been numerous studies on cyclic scheduling [15–20]. However, there are far fewer studies on scheduling cyclic shops with blocking [21,22]. Hanen [2] developed a branch-and-bound approach to deal with a generalised version of the cyclic job shop by incorporating precedence constraints. This approach was used in the context of a computer pipeline. Song and Lee [23] investigated a scheduling problem for cyclic job shops with blocking where each machine had an input buffer of finite capacity. They developed Petri net models for the cyclic job shops with blocking. Cavory et al. [24] presented a general approach for solving the cyclic job-shop scheduling problem, based on the coupling of a genetic algorithm and a scheduler. This scheduler utilised a Petri net model of the linear precedence constraints between cyclic tasks. The goal of this genetic algorithm was to define an order of priority for jobs on the machines, to be used by the scheduler for solving

resource conflicts. Brucker and Kampmeyer [22] described a model for cyclic machine-scheduling problems with blocking. In that study, a tabu search algorithm solved the considered problem with different neighbourhood structures. Kechadi et al. [25] proposed a recurrent neural network approach for the cyclic JSSP that attempted to find the optimum solution by minimising the energy state of the network. They also extended the recurrent neural network technique by coupling it with the Lagrangian relaxation method.

A more general cyclic JSSP with special restrictions has rarely been encountered in the literature. Nonetheless, a few studies have indeed investigated variants of this problem [26–30]. The studies by Kampmeyer [26] and Brucker and Kampmeyer [27] presented a mixed linear integer program for variants of the cyclic JSSP. Following this, Brucker et al. [28] made use of the same model to solve a problem in transportation. Brucker et al. [29] studied the cyclic JSSP with blocking and transportation and proposed a branch-and-bound method to solve the problem.

The authors wish to point out that there have been a significant number of research reports on the scheduling of blocking job shops and flow shops. Furthermore, there are studies on these problems that take into account setup times and resources such as robotics and servers. However, this study examines the cyclic aspects of the scheduling problem in job shops. The literature review presents the previous studies in this area, indicating that there are not enough studies on cyclic job-shop scheduling. The following are a few studies on cyclic scheduling in order to clarify their differences from the problems investigated in this paper. Gultekin et al. [31] considered cyclic scheduling, which involves processing a single part type on machines arranged in a flow-line arrangement. Their study focused on the cyclic scheduling of transportation operations. Then, Ghadiri Nejad et al. [32] studied a real-life example of the same problem with intermediate buffers and proposed a hybrid genetic algorithm for solving it. Furthermore, Foumani et al. [33] studied stochastic optimisation of two machine robotic cells processing a single type of part. Specifically, our study focuses on the cyclic job-shop scheduling problem, in which multiple jobs are processed on a set of machines in a different order.

The aim of our study was to bridge a gap in the literature, i.e., to investigate the line of research that concerns the cyclic JSSP with blocking and sequence-dependent setups. We propose a novel mathematical modelling approach for the cyclic job-shop scheduling. To the best of the authors' knowledge, this is the first time in the literature that the proposed model schedules all the operations in a single cycle by breaking the operations that pass the cycle reference timeline. In addition, we conduct a detailed analysis of the effects of blocking conditions and the influence of anticipatory and nonanticipatory setups on the cycle times. The anticipatory setup can begin even if the job is not yet available to be processed but the machine that it is going to be processed on is idle, whereas the nonanticipatory setup can begin only if both job and machine are available.

The paper is organised as follows. Section 2 presents the cyclic JSSP together with an illustrative example. The mathematical notation and a formulation for the base problem without sequence-dependent setup times are introduced in Section 3. Section 4 discusses the extension of the problem with setup times and also provides the relevant constraints for anticipatory and nonanticipatory setups. In Section 5, we provide details of the experiments conducted and the ensuing results. Section 6 concludes the paper and discusses possibilities of future work.

2. Blocking Cyclic Job Shop Scheduling

In this section, we define the blocking cyclic job shop scheduling problem with sequence-dependent setups. We first introduce the terminology and notation that will be used throughout the remainder of this paper. We then discuss alternative ways of defining blocking during setups. To illustrate the problem and the differences between the two setup constraints, we provide a detailed example.

The problem can be defined as follows. Let $N = \{1, 2, 3, \ldots, J\}$, where J is the total number of jobs in an MPS. Each job $j \in N$ contains a set of O_j operations where

$I_j = \{1, 2, 3, \ldots, O_j\}$. For each job, the sequence of operations is given in advance. Moreover, each operation is assigned to a machine M where the set of machines are $M = \{1, 2, 3, \ldots, K\}$. The sequence $F_j = \{f_1, f_2, f_3, \ldots, f_{(O_j)}\}$ specifies the machines that the operation $i \in I_j$ of job $j \in N$ will be processed on. Thereupon, $f_i = m$ denotes that the i^{th} operation would be processed on machine $m \in M$. Moreover, the processing time of the i^{th} operation of job $j \in N$ is given by $P_{j,i}$. Each job has at most one operation on each machine. Let J_m be the set of jobs that have an operation on machine m and $o(j, m) = i$ if $f_i = m$ for job j.

The assumptions are as follows:

- Each machine has a set of operations to execute and can execute only one operation at a time.
- The operations of a job are linked by precedence constraints.
- Each job has its own unique path through the machines, independently of the other jobs.
- Each operation is assigned to one particular dedicated machine and executed with no interruption and without pre-emption for a fixed processing time.
- There are no buffers between machines, so jobs continue to occupy a machine after the end of an operation until the next machine is available.

One job of each type has to be completed per cycle. The objective is to minimise the cycle time. Until now, this has not considered the question of setup times.

2.1. Sequence-Dependent Setup Times

In the work by Panwalkar et al. [34] on task scheduling, it was found that 75% of problems occurring in practice required at least one setup which was dependent on the order of the execution of tasks. Moreover, in 15% of the problems, setup times between all pairs of tasks were required. In many real-world situations, such as those in the chemical, printing, pharmaceutical, and automobile industries, there may be several types of setup operations. For instance, between jobs, there can be cleaning or the changing of tools, and when considering a pair of jobs, there is a strong dependence on the preceding job. In the context of job shops, this type of setup is usually modelled as a sequence-dependent setup between jobs on the same machine.

In the literature, job-shop problems usually consist of two types of sequence-dependent setup times [35,36]. These are anticipatory setup times (AS) and nonanticipatory setup times (NS). For the case of AS, the setup can begin even if the job is not yet available to be processed but the machine that it is going to be processed on is idle, whereas for NS, the setup can begin only if both job and machine are available.

Due to blocking conditions often seen in limited or zero-buffered systems, we consider both types of setups separately in this research. Thereupon, in the case of AS, the setup must be done before loading the job when the corresponding machine is idle. Moreover, in the case of NS, the setup must be done only if both job and machine are available, and the job has been loaded on the machine. Figure 2 illustrates how anticipatory and nonanticipatory setups work with a small example.

Figure 3 shows the affect of setups on the CJSS with blocking conditions. Since there are no intermediate buffers between the machines, any job that completes an operation on a machine has to remain on the machine if the subsequent machine in its processing route is busy. However, if the subsequent machine is idle, the AS can commence on this machine before the job completes its operation on the current machine. The job can immediately move to the next machine after completing its operation. However, if the subsequent machine is busy and the job has completed its operation, the current machine gets blocked until the operation on that machine completes and also for AS. Figure 3 shows the differences of AS and NS, where AS affect the schedule by causing blocking conditions and NS occupy the machine before starting processing an operation.

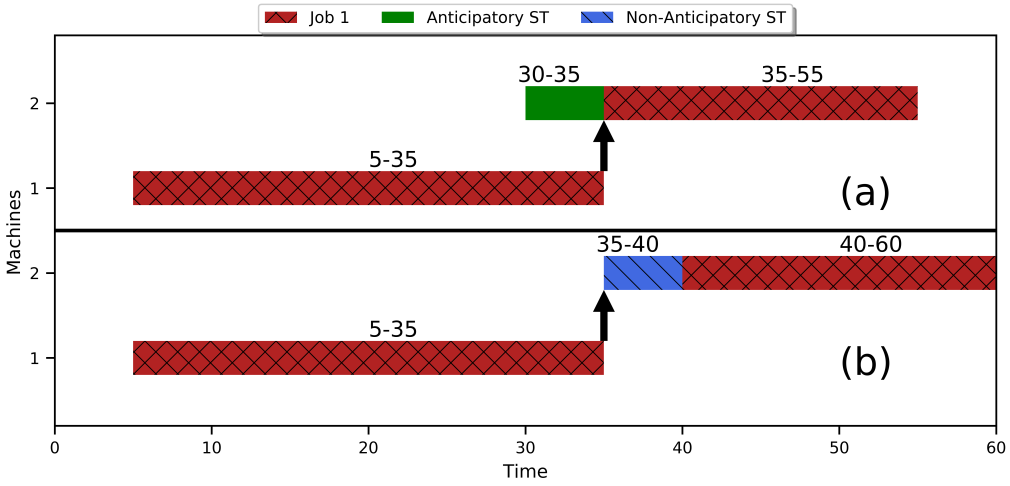

Figure 2. The anticipatory (**a**) and nonanticipatory (**b**) setups to process a job on a machine.

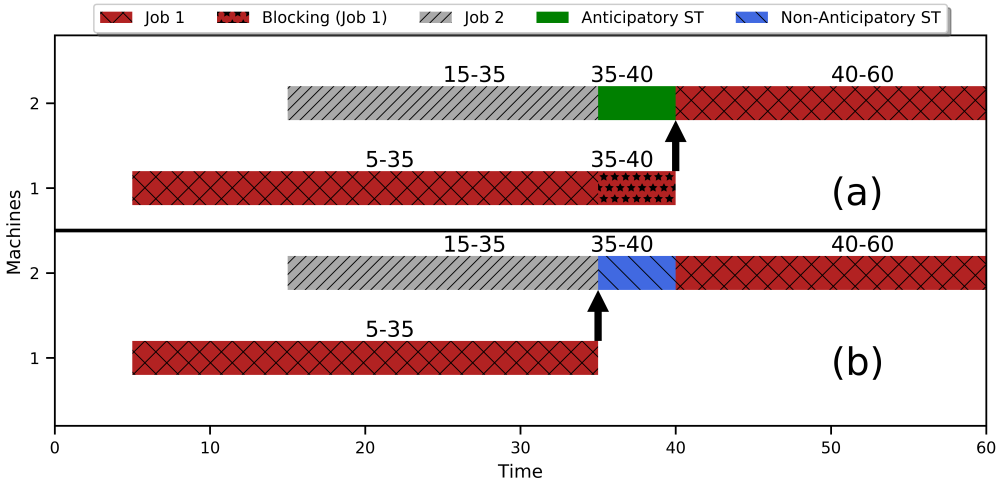

Figure 3. The effects of anticipatory (**a**) and nonanticipatory (**b**) setups on the schedule.

To the best of the authors' knowledge, the cyclic job-shop scheduling problem considering blocking conditions and sequence-dependent setup times at the same time have not been previously investigated in the literature.

2.2. The Cyclic JSSP—An Example

We now provide a detailed example of the cyclic JSSP and work through a solution to the problem. We consider a test case, which consists of three machines and three jobs. Table 1 provides a breakdown of the related data including the sequence of machines that a job must be performed on and the associated processing times. Table 2 provides the sequence-dependent setup times between the jobs on each machine. For the purposes of comparing the AS and NS, we assume the $AS_{j,j',m}$ and $NS_{j,j',m}$ are equal.

Table 1. The machine and processing time of each operation of the jobs.

Jobs	**Machines**			**Processing Times**		
1	1	3	2	63	95	28
2	1	2	3	45	41	69
3	2	3	1	39	22	44

Table 2. The sequence-dependent setup times between the pair of jobs on each machine.

	Machines								
	1			**2**			**3**		
Jobs	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
1	1	6	5	7	7	9	7	4	3
2	4	7	7	3	8	3	8	1	6
3	5	9	4	5	5	3	1	7	6

Figure 4 provides an illustration of the optimal solution found for the example above without considering setups. The start time and departure time of all operations are detailed in the figure. In addition, Figure 5 demonstrates the three consecutive cycles of the problem instance within which the cyclic job-shop production is presented. A single MPS enters the production line in each cycle and the arrows indicate the processing route of the jobs from the same MPS. It also presents the route for each job through the cyclic schedule, which is a mapping to the classic blocking job-shop scheduling problem.

In similar fashion to the previous example, Figures 6 and 7 show an optimal solution and three consecutive cycles, respectively, for the blocking CJSS problem with sequence-dependent anticipatory setups. The start time and departure time of all operations and sequence-dependent anticipatory setups are marked in black and are mentioned in the figure. We see three cycles associated with the optimal solution in Figure 7.

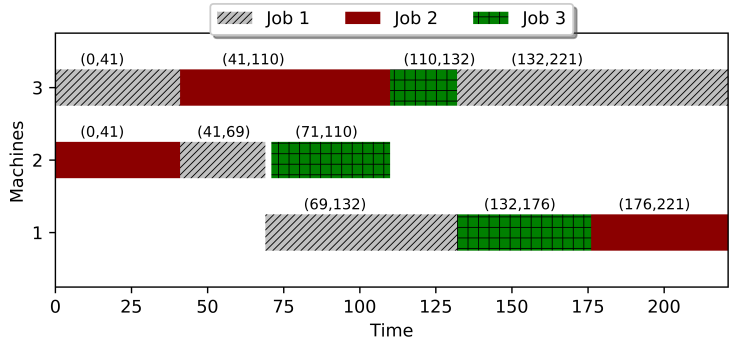

Figure 4. The optimal solution for the problem instance in Table 1; blocking CJSS problem.

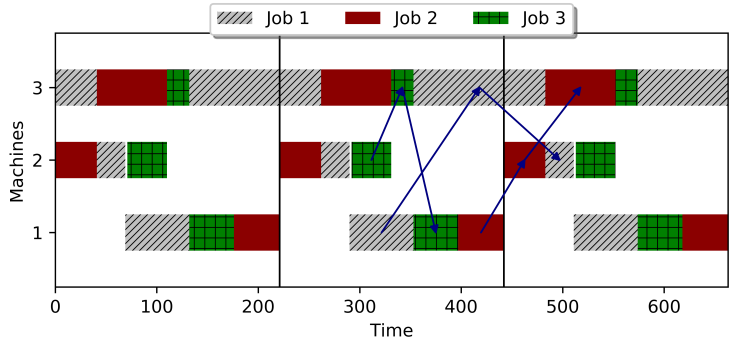

Figure 5. The Gantt chart of three consecutive cycles for the problem instance in Table 1; blocking CJSS problem.

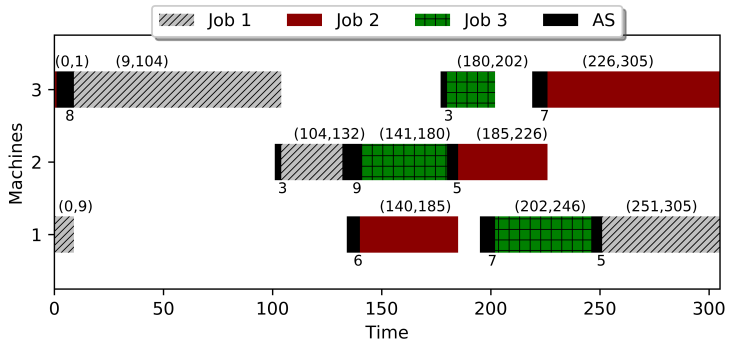

Figure 6. The detailed Gantt chart of the optimal solution found for the instance problem; blocking CJSS problem with sequence-dependent anticipatory setups (AS).

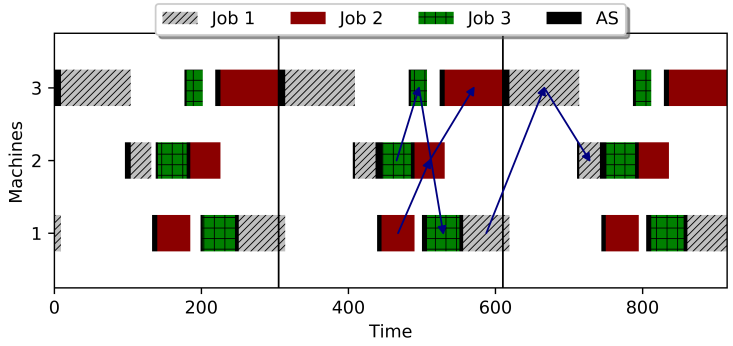

Figure 7. The Gantt chart of three consecutive cycles for the considered instance problem; blocking CJSS problem with sequence-dependent anticipatory setups (AS).

Again, Figures 8 and 9 show examples of the optimal solution and three consecutive cycles, respectively, for the blocking CJSS problem with sequence-dependent nonanticipatory setups. The start time and departure time of all operations and required nonanticipatory sequence-dependent setup times are marked in black and are mentioned in the figure. We see three cycles associated with the optimal solution in Figure 9.

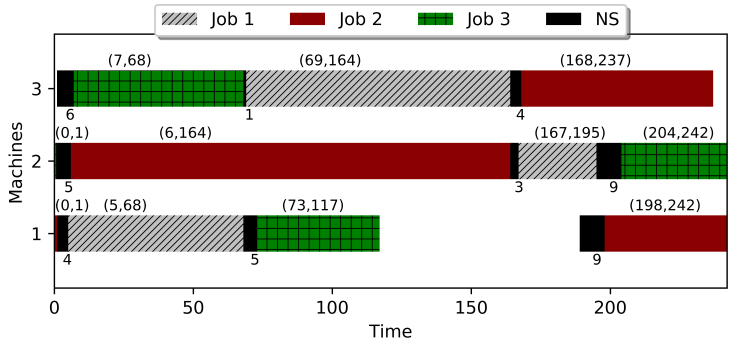

Figure 8. The detailed Gantt chart of the optimal solution found for the instance problem; blocking CJSS problem with sequence-dependent nonanticipatory setups (NS).

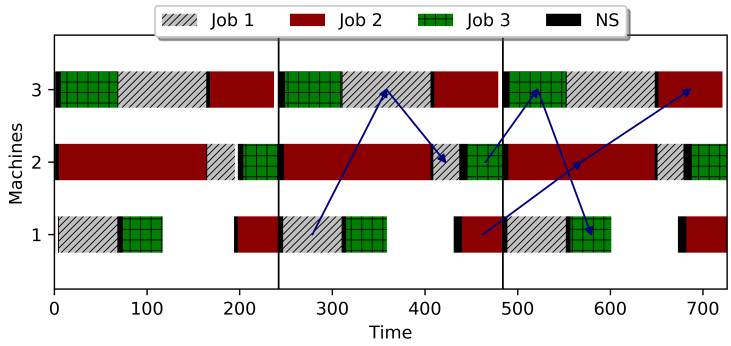

Figure 9. The Gantt chart of three consecutive cycles for the considered instance problem; blocking CJSS problem with sequence-dependent nonanticipatory setups (NS).

3. Mathematical Formulation of a Basic Model

In the following, we provide details of the mixed-integer linear programming (MILP) model for the basic cyclic job-shop scheduling (CJSS) problem without sequence-dependent setups. Due to the blocking conditions, the scheduling approach we used was based on the departure times of jobs from machines. The decision variables used in the proposed MILP model were:

$X_{j,j',m}$	1 if job j immediately precedes job j' on machine m; 0 otherwise;
$R_{j,m}$	1 if machine m is occupied by job j at the beginning of a cycle; 0 otherwise;
$B_{j,m}$	1 if job j is the first job to depart from machine m after the beginning of the cycle;
$D_{j,i}$	The departure time of job j from its i^{th} operation;
T	The cycle time;
$U_{j,m}$	An integer variable used to eliminate jobs' sequence subtours on each machine m;
W	A sufficiently large value, typically known as "Big M" in integer programming.

Note that the variable X determines the cyclic sequence of jobs on each machine. However, to do the scheduling of departure times, we need variables B to determine the first job on each machine. Similarly, R is used to capture the possibility that this first job has started processing before the start of the cycle so that the interval from the arrival of the part on the machine to the departure overlaps the cycle start time.

3.1. Occupied Machines Constraints

Scheduling all parts on all machines in a conceptual single cycle requires determining the occupied machines at the start of the cycle. Furthermore, based on the previously mentioned assumptions, each machine can process only one part at a time. The following constraint set (1) guarantees that each machine can be occupied by at most one part at the start time of the cycle.

$$\sum_{j \in N} R_{j,m} \leq 1 \qquad \forall\, m \in M \tag{1}$$

We note that the occupied machines are determined in terms of the *pressure* of the processing constraints of parts and the operational constraints of machines that are presented in the following sections.

3.2. Processing Constraints of Each Part

In classic job-shop production systems, performing any operation on a part only starts after completing its previous operation. The CJSS problem can be considered to be a specific version of the classic problem. Consider that the i^{th} operation of part j is performed

on machine m ($f_i = m$ for part j), and if it is free at the beginning of the cycle, then the processing constraints are as follows:

$$D_{j,i-1} + P_{j,i} \leq D_{j,i} \tag{2}$$

This states that the i^{th} operation of part j requires $P_{j,i}$ units of time to be elapsed. On the other hand, if machine m performing the i^{th} operation of part j is occupied at the beginning of the cycle by part j, then the related processing constraint would be as follows, due to the cyclic conditions:

$$D_{j,i-1} + P_{j,i} \leq D_{j,i} + T \tag{3}$$

The binary variable $R_{j,m}$ is used to determine if the related machine m is occupied by part j at the start of the cycle. Hence, these processing restrictions can be formulated as constraints sets (4) and (5):

$$D_{j,o(j,m)-1} + P_{j,o(j,m)} - W R_{j,m} \leq D_{j,o(j,m)} \qquad \forall\, m \in M, j \in J_m, 1 < o(j,m) \tag{4}$$

$$D_{j,o(j,m)-1} + P_{j,o(j,m)} \leq D_{j,o(j,m)} + T \qquad \forall\, m \in M, j \in J_m, 1 < o(j,m) \tag{5}$$

3.3. Parts' Sequence Determination on Machines

In addition, due to the cyclic scheduling, the relations between the cycles are established by the occupied machines at the start of the cycle. Actually, there is a cyclic sequence between the parts on each machine. To schedule the operations of all the parts in one cycle for each machine, the sequence of parts on each machine is considered to be determined based on the preceding and successor parts. The binary variable X determines the preceding and successor of all the parts on each machine. Thereupon, as illustrated in Figure 10, the first part in the next cycle is the successor of the last part in the present cycle.

To ensure the cyclic sequence of the parts, the following constraints sets (6)–(8) are used as follows:

$$\sum_{j' \in J_m} X_{j,j',m} = 1 \qquad \forall m \in M,\ j \in J_m \tag{6}$$

$$\sum_{j \in J_m} X_{j,j',m} = 1 \qquad \forall\, m \in M,\ j' \in J_m \tag{7}$$

$$U_{j,m} - U_{j',m} + 1 \leq J\,(1 - X_{j,j',m}) \qquad \forall\, m \in M,\ j \neq j' \in J_m \tag{8}$$

The last of these constraints, (8), serves to eliminate the possibility of subtours in the sequence of parts [37].

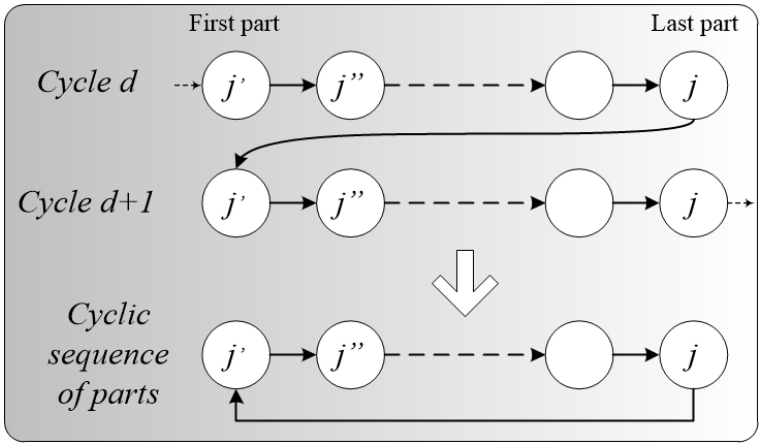

Figure 10. The cyclic sequence of parts on any machine.

3.4. Operational Constraints of Each Machine

The operational sequence constraints prevent the formation of conflicts in the usage of machines. Each machine should be unloaded before loading the next part on it and starting its processing operation. Figure 11 illustrates the operational sequence of parts on machines in the case where machine m is occupied or free at the start of the cycle.

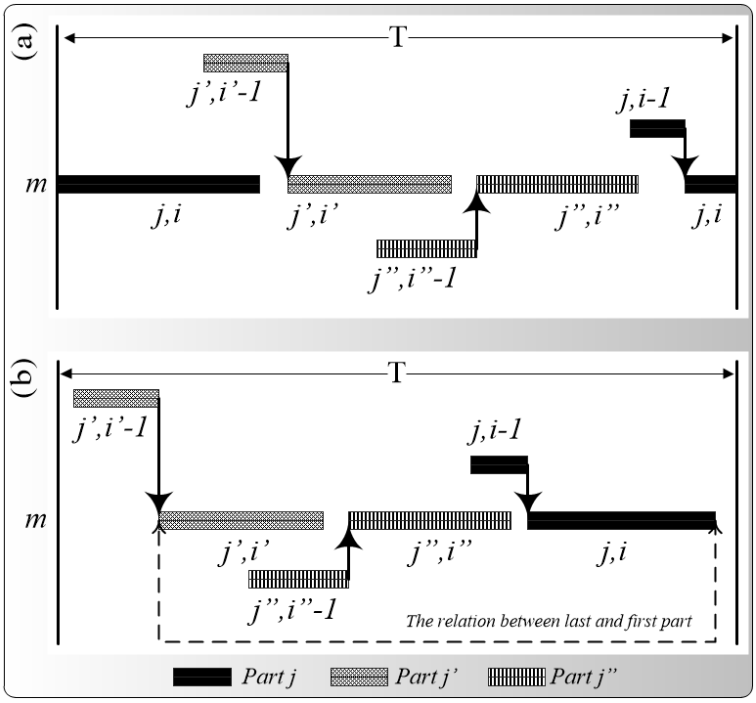

Figure 11. The operational sequencing restrictions of any machine m; (**a**,**b**) presents the restriction when m is occupied and free at the start of the cycle, respectively.

To state the relation between the last and first parts of consecutive operations, the variable B is used. Variable B determines the first part that leaves each machine during the cycle time and the following constraints set (9) guarantees that each machine will have just one.

$$\sum_{j \in J_m} B_{j,m} = 1 \qquad \forall\, m \in M \tag{9}$$

It is clear that if a machine is occupied by some part at the start of the cycle, then this part will be the first to depart, resulting in the following constraint set:

$$R_{j,m} \leq B_{j,m} \qquad \forall\, m \in M,\ j \in J_m \tag{10}$$

Due to the operational sequence of parts on machines when the part j' is the successor of part j on machine m then either

$$D_{j,i} + P_{j',i'} \leq D_{j',i'} \quad \text{or} \quad D_{j,i} + P_{j',i'} \leq D_{j',i'} + T. \tag{11}$$

The second inequality only holds for the case where j is the last part to depart from machine m before the end of the cycle and j' departs at the start of the next cycle. The following constraints sets (12) and (13) are proposed based on these inequalities:

$$D_{j,o(j,m)} + P_{j',o(j',m)} - \widehat{M}\,(1 - X_{j,j',m} + B_{j',m}) \leq D_{j',o(j',m)} \qquad \forall\, m \in M,\ j \neq j' \in J_m \tag{12}$$

$$D_{j,o(j,m)} + P_{j',o(j',m)} \leq D_{j',o(j',m)} + T \qquad \forall\, m \in M,\ j \neq j' \in J_m \tag{13}$$

It should be also noted that in the case of a zero buffer between the machines, the completed parts have to remain on the machines until the machine for performing its next operation becomes available. If part j' is the successor of part j on machine m then

$$D_{j,o(j,m)} \leq \begin{cases} D_{j',o(j',m)-1} + T & \text{if machine } m \text{ is idle at the start of the cycle with first job } j' \\ D_{j',o(j,m)-1} & \text{otherwise} \end{cases} \tag{14}$$

Note that the first case only arises if j departs m at the end of a cycle but part j' does not arrive until the start of the next cycle. The constraints set (15) is formulated based on this inequality as follows:

$$D_{j,o(j,m)} - M(1 - X_{j,j',m} + B_{j'm} - R_{j'm}) \leq D_{j',o(j',m)-1} \qquad \forall\, m \in M,\ j \neq j' \in J_m, 1 < o(j',m) \tag{15}$$

This inequality holds if (a) j' is after j on machine m so $X_{j,j',m} = 1$ and (b) we do not have $B_{j',m} - R_{j',m} = 1$ which would indicate that job j' arrived after the start of the cycle.

3.5. Proposed Mathematical Programming Model

The proposed mathematical programming model is as follows:

$$[MIP] \begin{cases} \text{minimise} & T \\ \text{subject to} & \\ & \textit{Constraints sets}: \ (1) - (15) \\ & T \geq D_{j,o(j,m)} \ \ \forall\, m \in M,\ j \in J_m \\ & X, R, B \ \in \ \{0,1\} \\ & T, D \ \in \ \mathbb{R}^+ \\ & U \ \in \ \mathbb{R} \end{cases}$$

In scheduling problems with buffers, safety times for the schedule need to be created and this increases the probability of a timely completion of the tasks. On the other hand, the problem described in this study assumes there is no buffer. In this respect, the number of feasible solutions decreases considerably, which increases the difficulty of the problem.

4. Blocking CJSS Problem with Sequence-Dependent Setup Times

We now extend the basic MIP formulation to incorporate the different types of setup times. Let $AS_{j,j',m}$ and $NS_{j,j',m}$, respectively, denote the anticipatory and nonanticipatory sequence-dependent setup times (In the following sections AS refers to the anticipatory sequence-dependent setups, and NS refers to nonanticipatory sequence-dependent setups.) required on machine m where j' is the successor job of j. The inequality Equation (11) was modified to Equation (16) for AS and Equation (17) for NS.

$$D_{j,i} + AS_{j,j',m} + P_{j',i'} \leq D_{j',i'} \quad \text{or} \quad D_{j,i} + AS_{j,j',m} + P_{j',i'} \leq D_{j',i'} + T \tag{16}$$

$$D_{j,i} + NS_{j,j',m} + P_{j',i'} \leq D_{j',i'} \quad \text{or} \quad D_{j,i} + NS_{j,j',m} + P_{j',i'} \leq D_{j',i'} + T \tag{17}$$

4.1. Blocking CJSS with Anticipatory Sequence-Dependent Setups (AS)

The MIP model was extended to consider AS. The constraints sets (12) and (13) were modified to (18) and (19) to ensure the operational sequence restrictions of jobs on machines based on the inequalities in Equation (16).

$$D_{j,o(j,m)} + AS_{j,j',m} + P_{j',o(j',m)} - \widehat{M}\,(1 - X_{j,j',m} + B_{j',m}) \leq D_{j',o(j',m)} \quad \forall\, m \in M,\ j \neq j' \in J_m \tag{18}$$

$$D_{j,o(j,m)} + AS_{j,j',m} + P_{j',o(j',m)} \leq D_{j',o(j',m)} + T \quad \forall\, m \in M,\ j \neq j' \in J_m \tag{19}$$

Additionally, a job cannot be loaded on the next machine before doing AS, even if both job and next machine are available. Therefore, as pointed in Figure 3, AS can cause blocking conditions. To ensure these restrictions, the inequalities in Equation (14) were modified as Equation (20).

$$D_{j,o(j,m)} + AS_{j,j',m} \leq \begin{cases} D_{j',o(j',m)-1} + T & \text{if machine } m \text{ idle at start of cycle with first job } j' \\ D_{j',o(j,m)-1} & \text{otherwise} \end{cases} \tag{20}$$

Due to the inequalities in Equation (20), the constraints set (15) was also modified as follows.

$$D_{j,o(j,m)} + AS_{j,j',m} - M(1 - X_{j,j',m} + B_{j'm} - R_{j'm}) \leq D_{j',o(j',m)-1} \quad \forall\, m \in M,\ j \neq j' \in J_m, 1 < o(j,m) \tag{21}$$

The proposed mathematical programming model for the CJSS problem with sequence-dependent AS was extended as follows:

$$[MIP - AS] \begin{cases} \text{minimise} & T \\ \text{subject to} & \\ & \textit{Constraints sets} : (1) - (10),\ (18) - (21) \\ & T \geq D_{j,o(j,m)} \ \ \forall\, m \in M,\ j \in J_m \\ & X, R, B \ \in \ \{0,1\} \\ & T, D \ \in \ \mathbb{R}^{+} \\ & U \ \in \ \mathbb{R} \end{cases}$$

4.2. Blocking CJSS Problem with Nonanticipatory Sequence-Dependent Setups (NS)

As mentioned previously, NS need both the job and machine to be available. The constraints sets (12) and (13) were modified as follows to ensure that the operational sequence restrictions of jobs on machines according to the inequalities in Equation (17) in the nonanticipatory case.

$$D_{j,o(j,m)} + NS_{j,j',m} + P_{j',o(j',m)} - \widehat{M}\,(1 - X_{j,j',m} + B_{j',m}) \leq D_{j',o(j',m)} \quad \forall\, m \in M,\ j \neq j' \in J_m \tag{22}$$

$$D_{j,o(j,m)} + NS_{j,j',m} + P_{j',o(j',m)} \leq D_{j',o(j',m)} + T \quad \forall\, m \in M,\ j \neq j' \in J_m \tag{23}$$

Moreover, nonanticipatory setups directly affect the processing constraints as presented in Figure 2 for job j'. The relationship between two consecutive operations of a job is stabilized considering the NS and processing time. NS need both the job and machine to be available, and hence, between two consecutive operations of a job there will be a time period to perform the NS. As a result, the NS directly affect the processing constraints. The inequalities in Equations (2) and (3) were presented for the case without any sequence-

dependent setup times. To ensure these restrictions applied between two consecutive operations of a job in CJSS with NS, the inequalities in Equations (2) and (3) were extended to Equations (24) and (25).

$$D_{j,i-1} + NS_{j',j,m} + P_{j,i} \leq D_{j,i} \tag{24}$$

$$D_{j,i-1} + NS_{j',j,m} + P_{j,i} \leq D_{j,i} + T \tag{25}$$

where job j' operates on machine m before job j, and $NS_{j',j,m}$ is the sequence-dependent NS. The variable R determined the occupying conditions, and hence, the processing constraints sets (4) and (5) were modified as constraints sets (26) and (27) based on Equations (24) and (25).

$$\begin{aligned} D_{j,o(j,m)-1} + X_{j',j,m} \times NS_{j',j,m} + P_{j,o(j,m)} - W\,R_{j,m} \leq D_{j,o(j,m)} \\ \forall\, m \in M, j \neq j' \in J_m, 1 < o(j,m) \end{aligned} \tag{26}$$

$$\begin{aligned} D_{j,o(j,m)-1} + X_{j',j,m} \times NS_{j',j,m} + P_{j,o(j,m)} \leq D_{j,o(j,m)} + T \\ \forall\, m \in M, j \neq j' \in J_m, 1 < o(j,m) \end{aligned} \tag{27}$$

Thus, the proposed MILP model for the CJSS problem with sequence-dependent nonanticipatory setups was extended as follows:

$$[MIP-NS] \begin{cases} \text{minimise} & T \\ \text{subject to} & \\ & \textit{Constraints sets}: \ (1),\ (6)-(10),\ (15),\ (22)-(27) \\ & T \geq D_{j,o(j,m)} \ \ \forall\, m \in M,\ j \in J_m \\ & X, R, B \ \in \ \{0,1\} \\ & T, D \ \in \ \mathbb{R}^+ \\ & U \ \in \ \mathbb{R} \end{cases}$$

The size of the basic formulation, as a function of an instance size with J jobs and K machines, is $\mathcal{O}(J^2K)$ variables and $\mathcal{O}(J^2K)$ constraints. The size is dominated by the $X_{j,j',m}$ variables with corresponding separation constraints (12)–(15) for each such X variable. The anticipatory and nonanticipatory variants simply modify the separation constraints but add neither new variables nor additional constraints.

5. Experimental Settings and Results

In the literature, there is no benchmark dataset specifically designed for the cyclic JSS problem. Cyclic production systems have minimal part sets, which makes the problem significantly different to classic JSS. This leads to two major differences. First, the objective of the classic JSS is minimising the makespan, compared to cycle time in cyclic JSS. While both objectives have similarities, the resulting (optimal) schedules can be substantially different. Second, in the cyclic problem, there is less variety among the products but higher volumes relative to classic job shops. This leads to the size of the minimal part sets in cyclic JSS being smaller than the number of jobs in classic JSS. Due to the above reasons, for this study, we designed a number of benchmark problem instances (see Appendix A, which provides details of the algorithm for generating the instances) motivated by practical requirements and using typical characteristics seen in JSS.

The MIPs developed in this study were programmed in Python 3.6 and all experiments were conducted on MonARCH, the campus cluster at Monash University. Each run was given four cores and a limit of 10 GB memory. The MIP models were solved using Gurobi 9.0.1 (http://www.gurobi.com/ (accessed on 30 August 2022)). Each run was given one hour of execution time.

Table 3 shows the results of the MIP models for the blocking CJSS problem without setups, with sequence-dependent anticipatory and nonanticipatory setups. There were 16 problem instances with different combinations of machines (K) and jobs (J), which were generated by the method described above. The results were split in three jobs: (1) cyclic job shop without sequence-dependent setups, (2) cyclic job shop with anticipatory sequence-dependent setups, and (3) cyclic job shop with nonanticipatory sequence-dependent setups. For each MIP, the details provided include the number of constraints (**#Cons**), number of variables (**#Vars**), the objective value of the best feasible solution (**BF**), the lower bound (**LB**), the gap between the upper and lower bounds $\left(Gap = \frac{(BF-LB)}{LB}\right)$, and the total time taken. If no feasible solution was found, the corresponding **BF** and **Gap** columns have a dash (-).

Table 3. The performance of proposed MIP approaches on the test problem instances (#*Cons: number of constraints; #Var: number of variables; BF: best found; LB: lower bound*).

K	J	#Cons	#Var	Cyclic JSS: Zero Setups				Anticipatory Setups				Nonanticipatory Setups			
				BF	**LB**	**Gap**	**Time**	**BF**	**LB**	**Gap**	**Time**	**BF**	**LB**	**Gap**	**Time**
9	**3**	733	190	257	257	0	7.35	390	390	0	18.6	337	337	0	17.3
	4	1313	289	270	270	0	5.09	441	441	0	6.92	335	335	0	7.77
	5	2157	406	535	164	69.27	3600	701	207	70.42	3600	660	166	74.80	3600
	6	3319	541	566	391	30.92	3600	-	444	-	3600	-	437	-	3600
12	**3**	988	253	258	258	0	14.94	373	373	0	14.53	354	354	0	22.83
	4	1769	385	335	335	0	308.59	468	468	0	684.56	431	431	0	386.49
	5	2904	541	529	189	64.21	3600	695	258	62.88	3600	741	214	71.08	3600
	6	4465	721	-	190	-	3600	-	223	-	3600	-	206	-	3600
15	**3**	1243	316	313	313	0	11.07	449	449	0	9.15	411	411	0	18.75
	4	2225	481	344	213	38.08	3600	457	251	44.99	3600	462	242	47.62	3600
	5	3651	676	568	330	41.90	3600	761	385	49.41	3600	-	285	-	3600
	6	5611	901	-	171	-	3600	-	200	-	3600	-	195	-	3600
18	**3**	1498	379	275	275	0	17.42	365	365	0	26.56	360	360	0	26.95
	4	2681	577	364	201	44.78	3600	519	315	39.36	3600	484	180	62.76	3600
	5	4398	811	600	171	71.55	3600	-	210	-	3600	721	175	75.79	3600
	6	6757	1081	-	184	-	3600	-	198	-	3600	-	193	-	3600

First, we considered cyclic JSS on its own. We see that for problems with a small number of jobs (three) the optimal solution was always found, and the time required was often quite small (under 30 s) relative to the time limit. Additionally, for a small number of machines (9, 12) and four jobs, the optimal solutions could also be found. For a larger number of jobs (five), feasible solutions were found but often with large gaps. For problems with six jobs, no feasible solutions were found irrespective of the number of machines. Nonetheless, the MIP always found lower bounds (the quality of which could not be guaranteed).

Considering cyclic JSS with anticipatory setups, we see a similar pattern. However, in this case, the problems proved more difficult to solve with one more problem not being solved (18 machines and 5 jobs) compared to Cyclic JSS. Additionally, when solutions were found, if they were not optimal, the gaps were larger. This is not surprising, since the overhead of the anticipatory setup times makes the problem more complex.

Finally, considering cyclic JSS with nonanticipatory setups, we see that it became even harder (compared to cyclic JSS and cyclic JSS with AS) to find solutions within the allowed time limits. The nonanticipatory setup required a job moving between machines to immediately leave the machine that it was currently on, and this meant that the MIP model struggled more when attempting to find a feasible solution.

In order to further understand the impact of sequence-dependent setup times, we considered a constant setup time with the following two options: (a) **MIN**: use the minimum set up time in a problem instance as the standard setup time between all jobs and (b) **MAX**: use the maximum set up time in a problem instance as the standard setup time between all jobs. Furthermore, to account for random effects in the data, four different datasets were generated with the same problem sizes. Table 4 shows the results of the comparisons with these changes for five different datasets for each problem size. The first three columns show replication (**Rep**), machines (**K**), and jobs (**J**), and the following columns are split by the best feasible solution, lower bounds, and gaps. Within each, the results are split by the type of setup, i.e., no setups (**ZS**), anticipatory, and nonanticipatory. For the case with setups, the results are further split by sequence-dependent setups (**SD**), standard maximum setups (**MAX**), and standard minimum setups (**MIN**).

The results showed, as expected, that cycle times increased with increasing amounts of setup time. Furthermore, the MIP models for which setups were used led to more instances where solutions could not be found. For example, **MAX** compared to **ZS** clearly showed two instances where solutions were not found. Interestingly, **MIN** was able to find two solutions where no solution was found in the case of **ZS**.

The differences in cycle times in the systems with sequence-dependent and standard setups demonstrated the importance of considering the sequence-dependent setups in blocking cyclic job-shop systems. Moreover, the cycle times of problems with no setups and minimum nonanticipatory setups were very similar, whereas the minimum anticipatory setups were rather different. In previous sections, we discussed that anticipatory setups could cause blocking conditions on the machines. Table 4 demonstrates that the cycle times were smaller in the systems with nonanticipatory setups comparing to the systems with anticipatory setups. These points present the importance of anticipatory setups in blocking job-shop systems.

Figure 12 presents the performance of the proposed MIP modelling approach for CJSS and demonstrates the **Gap** for different size of test instances. The plots demonstrate that the proposed approach was able to solve all the instances with a small number of jobs (solved optimally or close to optimality) and show that it was an efficient approach for cyclic production systems with fewer product varieties. Moreover, the MIP approach could find solutions for the medium-size test instances with four jobs (considering all machine) and five jobs (with 9 and 12 machines) but struggled to prove optimality. For instances with a large number of jobs and machines, the MIP approach rarely found solutions (plots on the right of the figure). The plots for the instances with five jobs indicate that the complexity of CJSS problem with maximum anticipatory setup (*Amax*) was higher compared to that of the CJSS problem with maximum nonanticipatory setup (*Nmax*). Overall, Figure 12 shows that the MIP approach could be used efficiently in production systems with a lower range of product varieties.

Table 4. A comparison of cycle times of different anticipatory and nonanticipatory setups for five repetitions of all MIP models. *ZS: zero setups, SD: sequence-dependent, Max: standard maximum setup, Min: standard minimum setup.*

Rep	K	J	Best Found							Lower Bound							Gap						
			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory		
				Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max
1	9	3	257	314	390	464	262	337	417	257	314	390	464	262	337	417	0	0	0	0	0	0	0
		4	270	365	441	526	275	335	447	270	365	441	526	275	335	447	0	0	0	0	0	0	0
		5	535	524	701	954	546	660	813	164	195	207	238	175	166	232	69.27	62.73	70.42	75.09	67.89	74.80	71.43
		6	566	-	-	-	-	-	-	391	397	444	583	397	437	583	30.92	-	-	-	-	-	-
	12	3	258	316	373	446	264	354	433	258	316	373	446	264	354	433	0	0	0	0	0	0	0
		4	335	393	468	559	340	431	523	335	393	468	559	340	431	523	0	0	0	0	0	0	0
		5	529	527	695	-	508	741	990	189	206	258	166	189	214	244	64.21	60.92	62.88	-	62.73	71.08	75.34
		6	-	-	-	-	-	-	-	190	205	223	211	182	206	245	-	-	-	-	-	-	-
	15	3	313	385	449	458	319	411	462	313	385	449	458	319	411	462	0	0	0	0	0	0	0
		4	344	375	457	584	354	462	540	213	217	251	341	217	242	341	38.08	42.13	44.99	41.61	38.70	47.62	36.85
		5	568	522	761	-	648	-	-	330	335	385	490	297	285	375	41.90	35.82	49.41	-	54.13	-	-
		6	-	-	-	-	-	-	-	171	194	200	206	182	195	233	-	-	-	-	-	-	-
	18	3	275	296	365	447	279	360	423	275	296	365	447	279	360	423	0	0	0	0	0	0	0
		4	364	407	519	663	399	484	654	201	264	315	384	189	180	283	44.78	35.25	39.36	42.13	52.63	62.76	56.73
		5	600	-	-	-	-	721	871	171	165	210	207	155	175	272	71.55	-	-	-	-	75.79	68.80
		6	-	-	-	-	-	-	-	184	195	198	267	177	193	177	-	-	-	-	-	-	-
2	9	3	258	260	344	428	262	329	406	258	260	344	428	262	329	406	0	0	0	0	0	0	0
		4	334	366	455	543	339	430	501	334	366	455	543	339	430	501	0	0	0	0	0	0	0
		5	412	421	552	743	419	553	653	412	421	552	563	419	553	594	0	0	0	24.23	0	0	8.96
		6	586	-	-	-	-	-	-	180	191	166	161	168	174	226	69.26	-	-	-	-	-	-
	12	3	289	291	358	386	293	339	423	289	291	358	386	293	339	423	0	0	0	0	0	0	0
		4	325	368	442	542	332	428	500	325	368	442	542	332	428	500	0	0	0	0	0	0	0
		5	500	613	553	-	421	662	776	295	301	396	459	300	369	455	41	50.85	28.39	-	28.74	44.26	41.37
		6	776	-	-	-	-	-	-	387	432	475	618	335	393	490	50.13	-	-	-	-	-	-

Table 4. *Cont.*

Rep	K	J	Best Found							Lower Bound							Gap						
			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory		
				Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max
3	15	3	292	372	392	462	297	383	452	292	372	392	462	297	383	452	0	0	0	0	0	0	0
		4	367	411	539	671	373	571	599	314	280	339	404	373	339	404	14.44	31.87	37.11	39.79	0	40.63	32.55
		5	614	-	-	-	-	-	-	175	184	227	249	160	189	422	71.49	-	-	-	-	-	-
		6	-	-	-	-	-	-	-	231	329	426	571	174	197	232	-	-	-	-	-	-	-
	18	3	244	312	354	417	249	330	404	244	312	354	417	249	330	404	0	0	0	0	0	0	0
		4	340	421	507	635	363	559	600	340	303	344	427	303	335	354	0	28.03	32.15	32.76	16.53	40.07	41
		5	573	-	-	-	-	-	-	221	201	209	326	170	365	301	61.43	-	-	-	-	-	-
		6	-	-	-	-	-	-	-	199	211	212	266	189	202	242	-	-	-	-	-	-	-
3	9	3	207	302	345	398	213	297	361	207	302	345	398	213	297	361	0	0	0	0	0	0	0
		4	376	394	475	569	382	470	554	376	394	475	569	382	470	554	0	0	0	0	0	0	0
		5	433	552	718	863	439	554	679	166	191	209	230	161	190	230	61.66	65.35	70.87	73.36	63.33	65.78	66.15
		6	-	-	-	-	-	-	-	349	364	401	550	353	401	550	-	-	-	-	-	-	-
	12	3	265	269	338	415	269	324	402	265	269	338	415	269	324	402	0	0	0	0	0	0	0
		4	362	460	524	583	368	487	566	362	275	524	583	368	356	566	0	40.22	0	0	0	26.90	0
		5	555	615	699	930	629	669	908	177	192	210	227	173	193	284	68.06	68.81	69.97	75.64	72.47	71.18	68.72
		6	714	-	-	-	-	-	-	203	203	216	340	194	240	297	71.56	-	-	-	-	-	-
	15	3	229	239	300	390	235	295	363	229	239	300	390	235	295	363	0	0	0	0	0	0	0
		4	410	451	580	638	384	556	631	209	238	252	362	211	210	335	48.96	47.23	56.55	43.26	45.12	62.30	46.91
		5	729	-	-	-	-	-	-	187	196	193	201	174	191	170	74.31	-	-	-	-	-	-
		6	-	-	-	-	-	-	-	179	195	207	202	168	184	215	-	-	-	-	-	-	-
	18	3	286	323	402	459	290	355	429	286	323	402	459	290	355	429	0	0	0	0	0	0	0
		4	382	439	519	674	420	516	688	153	153	217	299	152	217	205	59.94	65.11	58.19	55.64	63.93	57.95	70.19
		5	630	-	-	-	-	-	-	204	354	393	509	176	303	383	67.62	-	-	-	-	-	-
		6	792	-	-	-	-	-	-	187	186	178	186	169	184	213	76.39	-	-	-	-	-	-

Table 4. *Cont.*

Rep	K	J	Best Found							Lower Bound							Gap						
			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory		
				Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max
4	9	3	268	333	384	463	273	341	422	268	333	384	463	273	341	422	0	0	0	0	0	0	0
		4	270	377	431	538	274	363	455	270	377	431	538	274	363	455	0	0	0	0	0	0	0
		5	447	465	594	747	455	590	690	447	465	594	575	455	493	573	0	0	0	23.03	0	16.44	16.96
		6	633	-	-	-	-	-	-	196	184	199	216	179	197	233	69.03	-	-	-	-	-	-
	12	3	250	253	313	392	253	307	373	250	253	313	392	253	307	373	0	0	0	0	0	0	0
		4	358	440	495	644	390	481	599	144	166	197	284	141	197	177	59.78	62.37	60.20	55.90	63.72	59.04	70.48
		5	494	588	677	-	-	-	-	242	296	279	469	173	209	469	51.05	49.63	58.79	-	-	-	-
		6	-	-	-	-	-	-	-	400	406	440	592	406	440	592	-	-	-	-	-	-	-
	15	3	244	307	350	426	249	327	392	244	307	350	426	249	327	392	0	0	0	0	0	0	0
		4	304	363	512	598	309	411	529	304	236	274	350	248	273	350	0	34.99	46.48	41.47	19.87	33.58	33.84
		5	620	-	-	922	-	787	991	244	332	367	487	179	313	413	60.59	-	-	47.18	-	60.23	58.32
		6	-	-	-	-	-	-	-	205	210	221	263	185	201	245	-	-	-	-	-	-	-
	18	3	263	265	345	411	268	329	405	263	265	345	411	268	329	405	0	0	0	0	0	0	0
		4	382	450	518	687	390	484	680	168	168	193	295	170	185	294	56.08	62.67	62.74	57.06	56.41	61.80	56.76
		5	673	632	-	-	-	-	-	168	201	233	263	163	177	195	75.07	68.18	-	-	-	-	-
		6	-	-	-	-	-	-	-	247	226	339	417	196	203	333	-	-	-	-	-	-	-
5	9	3	267	389	433	451	271	344	411	267	389	433	451	271	344	411	0	0	0	0	0	0	0
		4	342	350	451	548	348	434	517	342	350	451	548	348	434	517	0	0	0	0	0	0	0
		5	449	515	631	768	457	595	713	449	359	425	510	457	400	510	0	30.29	32.65	33.59	0	32.77	28.47
		6	-	-	-	-	-	761	-	185	205	216	227	185	193	222	-	-	-	-	-	74.64	-
	12	3	292	408	438	479	298	375	455	292	408	438	479	298	375	455	0	0	0	0	0	0	0
		4	371	432	518	645	377	481	582	371	261	315	375	251	313	375	0	39.58	39.19	41.86	33.42	34.93	35.57
		5	525	687	752	835	481	791	906	394	399	466	554	399	466	554	24.95	41.92	38.03	33.65	17.05	41.09	38.85
		6	609	-	-	-	-	-	-	171	194	201	222	163	180	220	71.98	-	-	-	-	-	-

Table 4. *Cont.*

Rep	K	J	Best Found							Lower Bound							Gap						
			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory			ZS	Anticipatory			Nonanticipatory		
				Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max		Min	SD	Max	Min	SD	Max
	15	3	313	369	419	480	317	381	459	313	369	419	480	317	381	459	0	0	0	0	0	0	0
		4	386	414	469	597	393	485	590	269	273	374	397	273	331	397	30.31	34.06	20.26	33.50	30.53	31.75	32.71
		5	586	-	-	-	-	-	-	186	208	173	337	174	191	237	68.26	-	-	-	-	-	-
		6	-	-	-	-	-	-	-	186	196	201	203	177	184	245	-	-	-	-	-	-	-
	18	3	282	306	382	441	286	358	439	282	306	382	441	286	358	439	0	0	0	0	0	0	0
		4	446	486	565	684	460	518	729	158	167	205	317	154	205	289	64.67	65.71	63.72	53.66	66.60	60.46	60.36
		5	562	-	-	-	-	-	-	192	188	230	264	177	205	261	65.76	-	-	-	-	-	-
		6	-	-	-	-	-	-	-	203	208	224	283	187	211	252	-	-	-	-	-	-	-

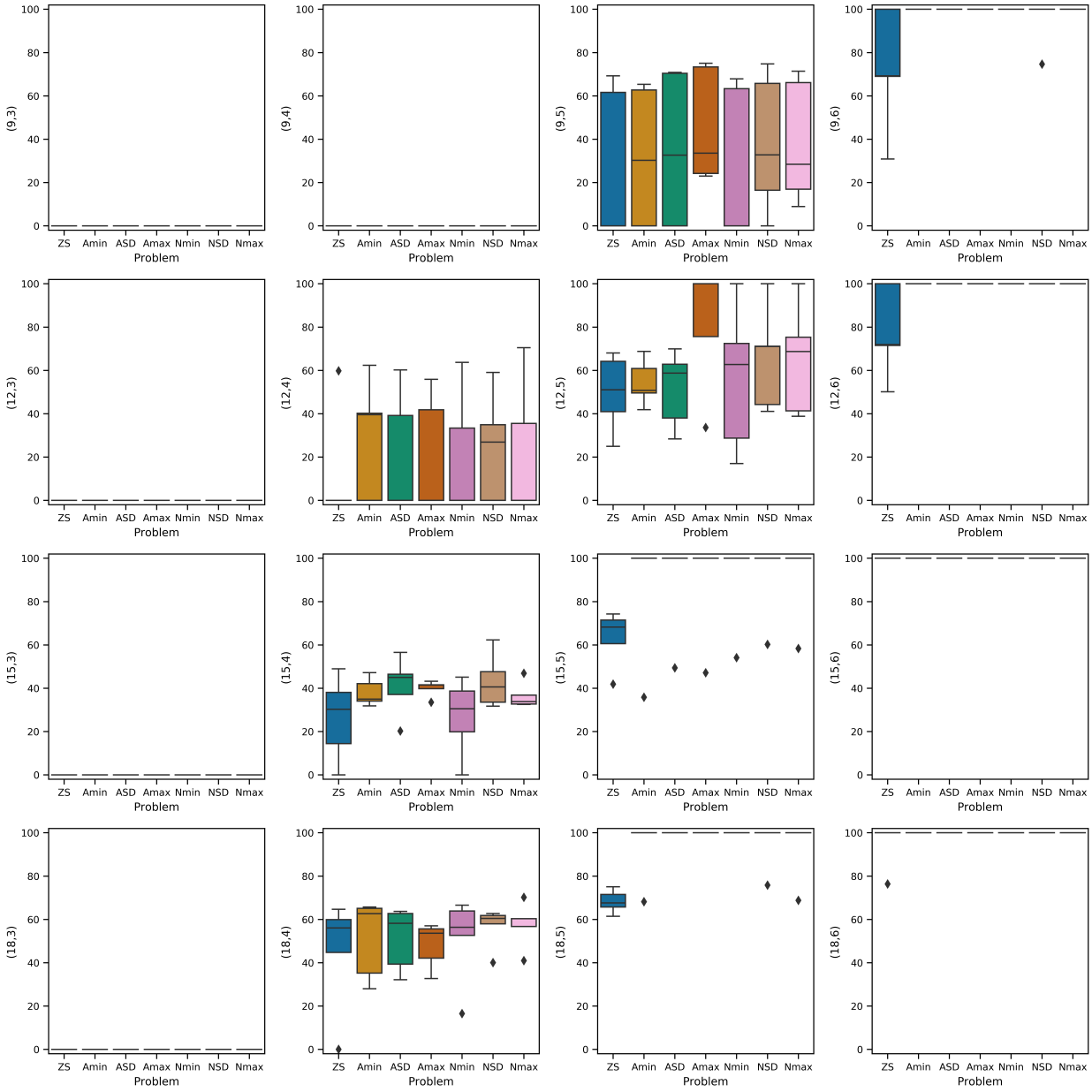

Figure 12. Box plots of the gaps in Table 4 (with $J = 3, \ldots, 6$ for the four columns and $K = 9, 12, 15, 18$ for the rows of graphs).

6. Conclusions

The cyclic job-shop scheduling problem was investigated in this paper. A novel mixed-integer programming model was proposed for the problem based on the characteristics of cyclic scheduling. Thereupon, the operations were scheduled within a single cycle. The occupying machines at the start time of the cycle had an important effect on solution quality due to the characteristics of the job-shop problem. Technological manufacturing cells, such as robotic cells, were attempted in order to eliminate or decrease the number of buffers. Thereupon, manufacturing systems without buffers were encountered in real situations. This study investigated the problem where the machines had no buffers that caused the blocking conditions, which rapidly decreased the number of feasible solutions and, therefore, made it a lot harder to find those feasible solutions.

Moreover, the sequence-dependent setups were considered in this research. There were two types of setups, anticipatory and nonanticipatory, which were included before and after loading the jobs on the machines, respectively. Thereupon, the anticipatory and

nonanticipatory sequence-dependent setups were investigated in this research separately. The advantages of this study and the achievements can be outlined as follows:

- Previous studies on modelling the cyclic job-shop scheduling problem were based on performing operations in iterative cycles, whereas the proposed model in this research scheduled all the operations within a single cycle.
- Due to the scheduling within a single cycle, the proposed model could be simply extended to different resource constrained variants.
- Two kinds of sequence-dependent setups were considered based on anticipatory and nonanticipatory concepts.
- The nonanticipatory setups did not affect the blocking condition as the related job did not have to wait for the previous machine because of the related setup.
- The anticipatory setups directly affected the blocking conditions, as any part might need to wait on the previous machine until the related setup was completed.

As future research, the proposed model could be applied to cyclic job-shop robotic cells. Moreover, this research could be extended to manufacturing systems with servers performing the setups.

The difficulty of solving these problems with MIP solvers observed in this paper indicate that further research on exact methods is required, perhaps based on decomposition methods, in order to solve large-scale cyclic job-shop scheduling problems. Moreover, the proposed model could be applied to cyclic job-shop scheduling problems in resource-constrained systems where robotic cells and servers perform the setups.

Author Contributions: Conceptualization, A.E.; Data curation, A.E.; Investigation, A.E., D.R.T. and A.T.E.; Methodology, A.E., D.R.T. and A.T.E.; Software, A.E.; Validation, A.E. and D.R.T.; Visualization, A.E.; Writing—original draft, A.E. and D.R.T.; Writing—review & editing, A.E., D.R.T. and A.T.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Generating Problem Instances

Algorithm A1 was used to generate the test problems for each benchmark with a specific number of jobs and machines based on the benchmark generation method reported by Taillard [38]. The procedure starts by taking as input the number of jobs, number of machines, the ranges for processing times, and sequence-dependent setup times. Lines 2–5 set lists for recording the processing route of each job. The processing time of each operation of each job is randomly generated within lines 6–7. Lines 8–9 generate the required sequence-dependent setup times between each pair of jobs. This is followed by generating the processing route for each job in lines 10–15, where the list Ψ has the set of machines that each job should be processed on. For each operation (line 12), a machine is selected from the set of machines and then Ψ is updated by eliminating the selected machine. Finally, a machine from the set of remaining machines is selected for the next operation in the queue.

- The operation times of jobs on the machines are generated from the uniform distribution $[a, b] = [1, 99]$.
- The sequence-dependent setup times are generated from the uniform distribution $[a, c] = [1, 33]$.
- The standard maximum and minimum setup times are equal to the maximum and minimum of sequence-dependent setup times, respectively.

Algorithm A1 Generate Problems.

procedure FUNCTION(J, K, a, b, c)
  for $j \in \{1, \ldots, J\}$ **do**
    $O_j \leftarrow K$
    $I_j \leftarrow \{1, \ldots, O_j\}$
    $F_j = \{\}$
  for $(j, i) : j \in \{1, \ldots, J\}, i \in I_j$ **do**
    $P_{j,i} \leftarrow U[a, b]$
  for $(j, j', m) : j, j' \in \{1, \ldots, J\}, j \neq j', m \in \{1, \ldots, K\}$ **do**
    $AS_{j,j',m} = NS_{j,j',m} \leftarrow U[a, c]$
  for $j : j \in \{1, \ldots, J\}$ **do**
    $\Psi = \{1, \ldots, K\}$
    for $i : i \in I_j$ **do**
      $f_i \leftarrow$ randomly select from Ψ
      $\Psi \leftarrow \Psi - f_i$
      $F_j \leftarrow F_j \cup f_i$
  return $(P,\ AS,\ NS,\ I_j,\ F_j)$

References

1. Brucker, P.; Kampmeyer, T. Tabu search algorithms for cyclic machine scheduling problems. *J. Sched.* **2005**, *8*, 303–322. [CrossRef]
2. Hanen, C. Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. *Eur. J. Oper. Res.* **1994**, *72*, 82–101. [CrossRef]
3. Brucker, P. Multiprocessor tasks. In *Scheduling Algorithms*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 313–341.
4. Wójcik, R.; Pempera, J. Designing cyclic schedules for streaming repetitive job-shop manufacturing systems with blocking and no-wait constraints. *IFAC-PapersOnLine* **2019**, *52*, 73–78. [CrossRef]
5. Levner, E.; Kats, V.; De Pablo, D.A.L.; Cheng, T.E. Complexity of cyclic scheduling problems: A state-of-the-art survey. *Comput. Ind. Eng.* **2010**, *59*, 352–361.
6. Šůcha, P.; Hanzálek, Z. A cyclic scheduling problem with an undetermined number of parallel identical processors. *Comput. Optim. Appl.* **2011**, *48*, 71–90. [CrossRef]
7. Brauner, N.; Finke, G.; Kubiak, W. Complexity of one-cycle robotic flow-shops. *J. Sched.* **2003**, *6*, 355–372. [CrossRef]
8. Lee, T.E.; Seo, J. Stochastic cyclic flow lines: Non-blocking, Markovian models. *J. Oper. Res. Soc.* **1998**, *49*, 537–548. [CrossRef]
9. Xing, L.N.; Chen, Y.W.; Yang, K.W. Multi-population interactive coevolutionary algorithm for flexible job shop scheduling problems. *Comput. Optim. Appl.* **2011**, *48*, 139–155. [CrossRef]
10. Grabowski, J.; Pempera, J. Sequencing of jobs in some production system. *Eur. J. Oper. Res.* **2000**, *125*, 535–550. [CrossRef]
11. Ronconi, D.P. A note on constructive heuristics for the flowshop problem with blocking. *Int. J. Prod. Econ.* **2004**, *87*, 39–48. [CrossRef]
12. Gong, H.; Tang, L.; Duin, C. A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. *Comput. Oper. Res.* **2010**, *37*, 960–969. [CrossRef]
13. Martinez, S.; Dauzère-Pérès, S.; Gueret, C.; Mati, Y.; Sauer, N. Complexity of flowshop scheduling problems with a new blocking constraint. *Eur. J. Oper. Res.* **2006**, *169*, 855–864. [CrossRef]
14. Lange, J.; Werner, F. Approaches to modeling train scheduling problems as job-shop problems with blocking constraints. *J. Sched.* **2018**, *21*, 191–207. [CrossRef]
15. Kats, V.; Levner, E. Cyclic scheduling in a robotic production line. *J. Sched.* **2002**, *5*, 23–41. [CrossRef]
16. Dawande, M.W.; Geismar, H.N.; Sethi, S.P.; Sriskandarajah, C. *Throughput Optimization in Robotic Cells*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007; Volume 101.
17. Ayala, M.; Benabid, A.; Artigues, C.; Hanen, C. The resource-constrained modulo scheduling problem: An experimental study. *Comput. Optim. Appl.* **2013**, *54*, 645–673. [CrossRef]
18. Pempera, J.; Smutnicki, C. Open shop cyclic scheduling. *Eur. J. Oper. Res.* **2018**, *269*, 773–781. [CrossRef]
19. Wang, J.; Pan, C.; Hu, H.; Li, L.; Zhou, Y. A cyclic scheduling approach to single-arm cluster tools with multiple wafer types and residency time constraints. *IEEE Trans. Autom. Sci. Eng.* **2018**, *16*, 1373–1386. [CrossRef]
20. Elmi, A.; Nazari, A.; Thiruvady, D.; Durmusoglu, A. Cyclic Flow Shop Robotic Cell Scheduling Problem With Multiple Part Types. *IEEE Trans. Eng. Manag.* **2020**. [CrossRef]
21. Bożejko, W.; Uchroński, M.; Wodecki, M. Block approach to the cyclic flow shop scheduling. *Comput. Ind. Eng.* **2015**, *81*, 158–166. [CrossRef]
22. Brucker, P.; Kampmeyer, T. Cyclic job shop scheduling problems with blocking. *Ann. Oper. Res.* **2008**, *159*, 161–181. [CrossRef]

23. Song, J.S.; Lee, T.E. Petri net modeling and scheduling for cyclic job shops with blocking. *Comput. Ind. Eng.* **1998**, *34*, 281–295. [CrossRef]
24. Cavory, G.; Dupas, R.; Goncalves, G. A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints. *Eur. J. Oper. Res.* **2005**, *161*, 73–85. [CrossRef]
25. Kechadi, M.T.; Low, K.S.; Goncalves, G. Recurrent neural network approach for cyclic job shop scheduling problem. *J. Manuf. Syst.* **2013**, *32*, 689–699. [CrossRef]
26. Kampmeyer, T. Cyclic Scheduling Problems. Ph.D. Thesis, Universität Osnabrück, Osnabrück, Germany, 2006.
27. Brucker, P.; Kampmeyer, T. A general model for cyclic machine scheduling problems. *Discret. Appl. Math.* **2008**, *156*, 2561–2572. [CrossRef]
28. Brucker, P.; Burke, E.K.; Groenemeyer, S. A mixed integer programming model for the cyclic job-shop problem with transportation. *Discret. Appl. Math.* **2012**, *160*, 1924–1935. [CrossRef]
29. Brucker, P.; Burke, E.K.; Groenemeyer, S. A branch and bound algorithm for the cyclic job-shop problem with transportation. *Comput. Oper. Res.* **2012**, *39*, 3200–3214. [CrossRef]
30. Quinton, F.; Hamaz, I.; Houssin, L. A mixed integer linear programming modelling for the flexible cyclic jobshop problem. *Ann. Oper. Res.* **2020**, *285*, 335–352. [CrossRef]
31. Gultekin, H.; Dalgıç, Ö.O.; Akturk, M.S. Pure cycles in two-machine dual-gripper robotic cells. *Robot. Comput.-Integr. Manuf.* **2017**, *48*, 121–131. [CrossRef]
32. Ghadiri Nejad, M.; Shavarani, S.M.; Güden, H.; Barenji, R.V. Process sequencing for a pick-and-place robot in a real-life flexible robotic cell. *Int. J. Adv. Manuf. Technol.* **2019**, *103*, 3613–3627. [CrossRef]
33. Foumani, M.; Razeghi, A.; Smith-Miles, K. Stochastic optimization of two-machine flow shop robotic cells with controllable inspection times: From theory toward practice. *Robot. Comput.-Integr. Manuf.* **2020**, *61*, 101822. [CrossRef]
34. Panwalkar, S.; Dudek, R.; Smith, M. Sequencing research and the industrial scheduling problem. In *Proceedings of the Symposium on the Theory of Scheduling and Its Applications*; Springer: Berlin/Heidelberg, Germany, 1973; pp. 29–38.
35. Defersha, F.M.; Chen, M. A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. *Int. J. Adv. Manuf. Technol.* **2010**, *49*, 263–279. [CrossRef]
36. Framinan, J.M.; Leisten, R.; García, R.R. Manufacturing scheduling systems. In *An integrated view on Models, Methods and Tools*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 51–63.
37. Miller, C.E.; Tucker, A.W.; Zemlin, R.A. Integer programming formulation of traveling salesman problems. *J. ACM (JACM)* **1960**, *7*, 326–329. [CrossRef]
38. Taillard, E. Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **1993**, *64*, 278–285. [CrossRef]