ON SCHEDULING CYCLE SHOPS: CLASSIFICATION, COMPLEXITY AND APPROXIMATION

MARTIN MIDDENDORF† AND VADIM G. TIMKOVSKY⋆

†Institute of Applied Computer Science and Formal Description Methods, University of Karlsruhe, Karlsruhe, Germany
⋆Star Data Systems Inc., Toronto, Ontario, Canada

November 18, 1999

ABSTRACT. This paper considers problems of finding nonperiodic and periodic schedules in a cycle shop which is a special case of a job shop but an extension of a flow shop. The cycle shop means the machine environment where all jobs have to pass the machines over the same route like in a flow shop but some of the machines in the route can be met more than once. We propose a classification of cycle shops and show that recently studied reentrant flow shops, robotic flow shops, loop reentrant flowshops and V shops are special cases of cycle shops. Problems solvable in polynomial time, pseudopolynomial time, NP-hard problems and performance guarantee approximations are presented. Related earlier results are surveyed.

1. INTRODUCTION

A *cycle shop* introduced by Degtiarev and Timkovsky [DT76] (see also [T77, T86, T92, T98]) is a special case of a *job shop* but an extension of a *flow shop*. All jobs in a cycle shop have the same sequence of operations on the machines, but in contrast to a flow shop, some operations can be repeated on some machines a number of times, and this number can differ from one machine to another. Cycle-shop scheduling problems arose from the VLSI technologes research that was held in the middle seventies in Soviet electronic industry. The term "cycle shop" was chosen because the sequence of operations on the machines can be nicely depicted by a spiral *cyclogram* (see Figure 1.1) that was a favourite tool of VLSI technologies designers. Cycle shops and their special cases have been considered under different names such as a *flow line with an operator* of Livshits *et al.* [LMC74], a *reentrant flowshop* of Graves *et al.* [GMSZ83], Morton and Pentico [MP93], a *V-shop* of Lev and Adiri [LA84], a *robotic flowshop* of Asfahl [A85], a *periodic job-shop* of Serafini and Ukovich

1991 *Mathematics Subject Classification.* 68Q25, 90B35.

Key words and phrases. Parallel machines, cycle shop, job shop, robotic flow shop, scheduling, complexity, approximation.

[SU89], a *repetitive flowshop* of Ramazani and Younis [RY92], a *cyclic job shop* of Roundy [R92], a *robotic cell* of Sethi *et al.* [SSSBK92], a *loop reentrant flowshop* of Gupta [G93], a *chain-reentrant shop* of Wang *et al.* [WSV97], a *cyclic robotic flowshop* of Kats and Levner [KL97] and a *flow shop with transportation times and a single robot* of Hurink and Knust [HK98]. Recent papers on periodic scheduling surveyed by Crama *et al.* [CKKL99] pay a special attention to such an environment due to applications to robotic manufacturing. Besides, a cycle shop represents a traffic light scheduling model [SU89A].

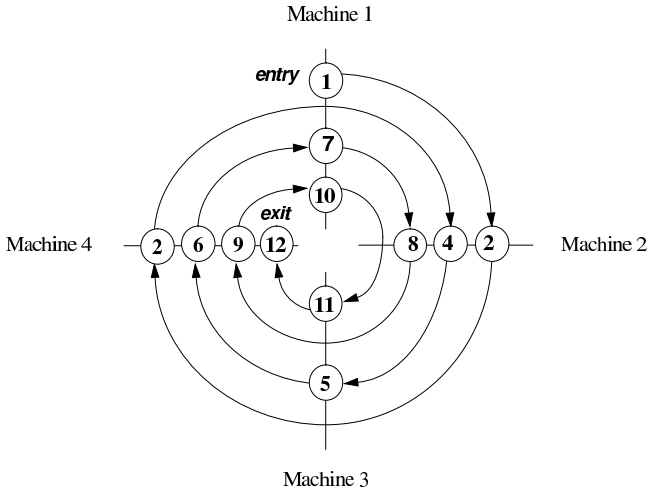

FIGURE 1.1. A cyclogram for a 4-machine cycle shop with 12-operation jobs: Machine 1 processes the 1st, 7th and 10th operation, Machine 2 processes the 2nd, 4th and 8th operation, Machine 3 processes the 5th and 11th operation, Machine 4 processes the 3rd, 6th, 9th and 12th operation of each job.

This paper presents new results on the complexity and approximation of cycle-shop scheduling. Section 2 gives a classification of cycle-shops and related scheduling problems and surveys results obtained earlier and in this paper. Section 3 presents *cycle-shopping isomorphisms* between problems on modulated identical parallel machines and problems in a perfect cycle shop, and the *cycle-shopping theorem* which is a general polynomial-time reduction of problems on identical parallel machines with equal-processing-time jobs to problems in a cycle shop with unit-time operations. Along with the *flow-shopping theorem* [BK98, T98A] and the *open-shopping theorem* [T98A] it represents one more natural derivative from the *job-shopping theorem* [T98]. Nonperiodic problems are considered in Section 4. After establishing problems solvable in polynomial time, we prove the NP-hardness of cycle-shop problems with unit-time operations that were open before by using the cycle-shopping theorem and known NP-hard problems on identical parallel machines with equal-processing-time jobs. Section 5 is devoted to periodic problems. The main result in it is a polynomial-time algorithm for a general minimum-period cycle-shop problem and the strong NP-hardness proof of the simplest minimum-flow problem with unit-time operations in cycle shops with only one repetitive machine. We also consider performance guarantee approximations for the latter problem. In conclusion we discuss open problems.

2. Claccification and previous results

The main purpose of this section is to extend the commonly known classification of nonperiodic scheduling problems [LLRKS93, B95] to robotic and periodic scheduling problems using only minor innovations. In our classification a robot is just an additional machine in any machine environment that only requires a setup after processing operations. Herein, the setup time is exactly the travel time of the empty robot move from one machine to another. Besides, we consider periodic problems as special cases of nonperiodic problems. As we can see, such an approach helps to discover the relationship between classic problems and recently studied problems in robotic and periodic shops.

All denotations will follow the notation in [LLRKS93]: $\circ$, the empty symbol; n, the *number of jobs* in a given finite job set $\mathfrak{J}$; m, the *number of machines*; C_j, m_j, r_j, d_j and w_j, the *completion time*, the *number of operations*, the *release date*, the *due date* and the *weight* of the job J_j, respectively; O_{ij}, the ith *operation* in J_j; p_{ij}, the *processing time* of O_{ij}, where $i = 1, 2, \ldots, m_j$ and $j = 1, 2, \ldots, n$. Parameters r_j, d_j, w_j and p_{ij} are considered to be integer, herein $\min_j r_j = 0$, $w_j > 0$. If not stated otherwise we assume that processing times p_{ij} are positive. Set $p_{\max} = \max_{ij} p_{ij}$, $p_{\min} = \min_{ij} p_{ij}$. Parameters $\mu_{ij}, \mu_i \in \{1, 2, \ldots, m\}$ will be used as indices of the machines $M_1, M_2, \ldots, M_m$. To represent scheduling problems we follow the three-field classification $\alpha|\beta|\gamma$, where α, β and γ specify the *machine environment*, *job characteristics* and the *minimization criterion*, respectively.

2.1. Cycle shop and related machine environments. As we use a common terminology in our classification, some of the machine environments will possibly have not the same names under which they were considered earlier.

- $\alpha = \alpha_0 \alpha_{1\varsigma}^{\epsilon} \alpha_2$, where α_0, ϵ, α_2 and ς will be defined further in this section and Section 2.3.
- $\alpha_1 \in \{P, P\,\text{mod}, \circ, J, J\,\text{per}, RJ, C, C\,\text{per}, RC, RF, V, L, F\}$.
- $\alpha_1 = P$: *identical parallel machines:* J_j consists of a single operation to be processed on any machine during time $p_j = p_{1j}$.
- $\alpha_1 = P\,\text{mod}$: *modulated identical parallel machines:* identical parallel machines, where M_i can start only at times $(i-1) \bmod m$.
- $\alpha_1 = \circ$: a *single machine:* identical parallel machines, where $m = 1$.
- $\alpha_1 = J$: a *job shop:* J_j consists of a chain of operations $O_{1j}, O_{2j}, \ldots, O_{m_j,j}$, where O_{ij} has to be processed by $M_{\mu_{ij}}$ without interruption starting on or after the completion of $O_{i-1,j}$ and $\mu_{i-1,j} \neq \mu_{ij}$ for $i > 1$.
- $\alpha_1 = J\,\text{per}$: a *perfect job shop:* a job shop, where $\mu_{ij} - 1 = (i-1) \bmod m$.
- $\alpha_1 = RJ$: a *robotic job shop:* a job shop, where m_j are odd and $m = \mu_i \neq \mu_{i-1,j} \neq \mu_{i+1,j} \neq m$ if i is even.
- $\alpha_1 = C$: a *cycle shop:* a job shop, where $m_j = \ell \geq m$, $\mu_{ij} = \mu_i$.
- $\alpha_1 = C\,\text{per}$: a *perfect cycle shop:* a cycle shop which is a perfect job shop.
- $\alpha_1 = RC$: a *robotic cycle shop:* a robotic job shop which is a cycle shop.
- $\alpha_1 = RF$: a *robotic flow shop:* a robotic cycle shop, where $\mu_i = \frac{i+1}{2}$ if i is odd.
- $\alpha_1 = V$: a *V shop:* a cycle shop, where $\ell = 2m - 1$, $\mu_i = i$ if $i \leq m$ and $\mu_i = 2m - i$ if $i > m$.
- $\alpha_1 = L$: a *loop shop:* a cycle shop, where $\ell = m + 1$, $\mu_i = i$ if $i < \ell$ and $\mu_\ell = 1$.
- $\alpha_1 = F$: a *flow shop:* a cycle shop, where $\ell = m$, $\mu_i = i$.

Jobs in a job shop are *identical* or of one *type* if $m_j = \ell$, $\mu_{ij} = \mu_i$ and $p_{ij} = p_i$. The number different *types* of jobs in $\mathfrak{J}$ will be denoted as h. So, jobs in a cycle shop are identical if $p_{ij} = p_i$. If jobs in a cycle shop are identical, O_i, p_i and S_i, where $i = 1, 2, \ldots, \ell$, will

denote operations of a single job, their processing and start times, respectively. We have to mention that originally perfect job shops were called *periodic* [T98]. The term "periodic" in this case refers only to the way of distributing operations among the machines. Here we change the term to avoid a confusion with the periodicity in time (*cf.* [SU89, LP97]).

Let M_k be the machine in a cycle shop that processes $\mathfrak{m}_k$ operations in one job. Then define $\mathfrak{m}_k$ to be the *multiplicity* of M_k. We call M_k a *unimachine* or a *multimachine* if $\mathfrak{m}_k = 1$ or $\mathfrak{m}_k > 1$, respectively. Define $\mathfrak{r}$, the number of multimachines, and $\mathfrak{m} = \max_{1 \le k \le m} \mathfrak{m}_k$ to be the (*cyclic*) *rank* and the (*cyclic*) *multiplicity* of the cycle shop, respectively.

$\alpha_0 \in \{\circ, \mathfrak{r}\}$ will be used only for cycle shops.
$\alpha_0 = \circ$: the rank of the cycle shop is not fixed.
$\alpha_0 = \mathfrak{r}$: the rank the cycle shop is fixed and equal $\mathfrak{r}$.
$\epsilon \in \{\circ, \mathfrak{m}\}$ fixes the multiplicity of cycle shops.
$\epsilon = \circ$: the multiplicity of the cycle shop is not fixed.
$\epsilon = \mathfrak{r}$: the multiplicity the cycle shop is fixed and equal $\mathfrak{m}$.
$\alpha_2 \in \{\circ, m\}$ fixes the number of machines.
$\alpha_2 = \circ$: the number of machines is not fixed.
$\alpha_2 = m$: the number of machines is fixed and equal $m + 1$ if α_1 represents a robotic shop or m otherwise. We assume that $\alpha_1 = \circ$ if and only if $\alpha_2 = 1$.

For example, a flow shop is of rank 0 and multiplicity 1, a loop shop is of rank 1 and multiplicity 2, a robotic flow shop is of rank 1 and multiplicity $m - 1$, and a two-machine cycle shop with four or more operations in jobs is of rank 2. Notice that a two-machine robotic flow shop is a three-machine flow shop. A three-machine robotic flow shop is a cycle shop of rank 1 and multiplicity 2, *i.e.*, the simplest robotic flow shop which is not a flow shop. Aslo, note that a V shop is of rank $m - 1$ and multiplicity 2. A loop shop, or a robotic flow shop has only one multimachine. In a dual way, a V shop has only one unimachine. Notwithstanding, a two-machine loop shop, a two-machine V shop and a two-machine cycle shop with three operations in jobs represent the same cycle-shop machine environment which minimally differs from a two-machine flow shop. Further we mean and consider only cycle shops of positive rank, *i.e.*, which are not flow shops.

Observe that a robotic job shop was constructed from a job shop by inserting an additional operation on an additional common machine between each two consecutive operations in each job. Hence, not only a cycle or flow shop but any special case of a job shop has a robotic counterpart (see Figure 1.2). However, we consider only robotic cycle or flow shops because we are not aware of the results on other types of robotic shops. The multimachine M_m in the robotic shops we call a *light robot*. Operations on M_m can be considered as transportations of parts between the other machines. We use the term "light" to emphasize that the empty light robot can move instantly, and all related problems can be studied without a consideration of robot moves. In the next subsection we also consider a *heavy robot* which being empty cannot move instantly, and all related problems require a consideration of robot moves. In robotic shops, P_j will denote the part related to J_j, p_{ej} and p_{oj} will denote the processing times only of even operations O_{ej}, $e = 2, 4, \ldots, m_j - 1$, on M_m and odd operations O_{oj}, $o = 1, 3, \ldots, m_j$ on $M_1, M_2, \ldots, M_{m-1}$. We assume that a light or heavy robot loads and unloads parts instantly as well because nonzero loading and unloading times can be included into processing times p_{oj} or transportation times p_{ej}.

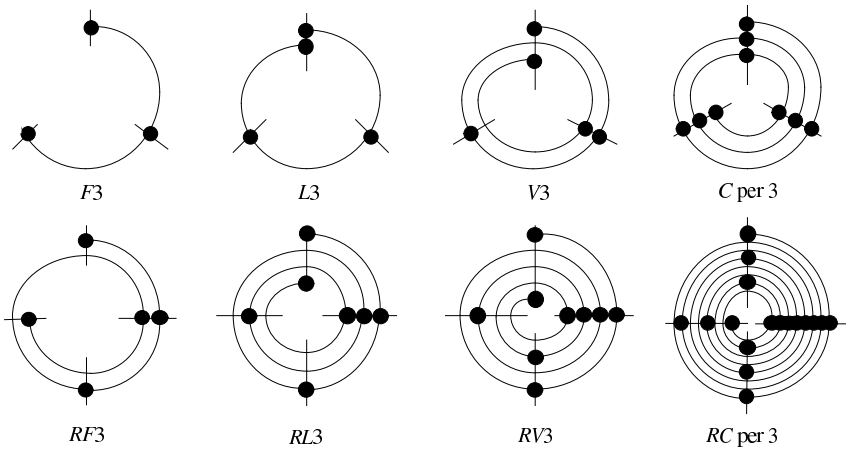

FIGURE 1.2. Cyclograms for three-machine cycle shops: $F3$, a flow shop; $L3$, a loop shop; $V3$, a V shop; $C\text{per}\,3$, a perfect cycle shop; $RF3$, a robotic flow shop; $RL3$, a robotic loop shop; $RV3$, a robotic V shop; and $RC\text{per}\,3$, a robotic perfect cycle shop.

2.2. Heavily robotic machine environments. Let us consider robotic job shops where M_1 and M_{m-1} are an *input hopper* and an *output hopper*, respectively. Formally, it means that $\mu_{1j} = 1$, $\mu_{m_j j} = m-1$ and $1 \neq \mu_{ij} \neq m-1$ for all odd $i \in \{3, 5, \ldots, m_j - 2\}$ and all j. In this case, p_{1j} and $p_{m_j j}$ are the *pick-up time* and the *drop-off time* that the robot needs to pick P_j up from the input hopper and drop P_j off onto the output hopper. In all models we know these times are not dependant on j or m_j. So, we assume that $p_{1j} = \mathfrak{p}$ and $p_{m_j j} = \mathfrak{d}$ for all j. Thus, any complications related to the robot presence reduce to the consideration of robot moves. We also assume that robot moves are *purposeful*, *i.e.*, the robot moves to and stops at the machines only with the purpose of loading, unloading or transporting parts, and that no *buffers* between or at the machines, *i.e.*, at any time any part is being handled by the robot M_m, in the input hopper M_1, in the output hopper M_{m-1}, or being processed on $M_2, M_3, \ldots, M_{m-2}$. If the parts are not allowed to stay on $M_2, M_3, \ldots, M_{m-2}$ without processing, then the *no-buffer environment* means the *no-wait* constraint for parts.

Let S_{ij} and C_{ij} be the start time and the completion time of O_{ij} in a nonpreemptive schedule σ for $\mathfrak{J}$ in a robotic job shop. Then $C_{ij} - S_{ij} = p_{ij}$. Define the *event* E_{ij} to be S_{ij} or C_{ij}. Since an empty heavy robot cannot move instantly, and loading-unloading can be executed only by the robot, any two events on $M_1, M_2, \ldots, M_{m-1}$ should differ in time. That is, a heavy robot M_m enforces the *distinctness constraint*:

$$\text{(DC)} \qquad (o, j) \neq (o', j') \quad \Longleftrightarrow \quad E_{oj} \neq E_{o'j'}.$$

If DC is satisfied, then we call σ a *distinct schedule*. It is easy to see that any distinct schedule σ defines the increasing sequence of all events in it:

$$\varepsilon = E_{o_1 j_1} < E_{o_2 j_2} < \cdots < E_{o_{k-1} j_{k-1}} < E_{o_k j_k} < \cdots < E_{o_{l-1} j_{l-1}} < E_{o_l j_l},$$

where $l = 2\sum_{j=1}^{n} \frac{m_j+1}{2} = n + \sum_{j=1}^{n} m_j$, $o_k \in \{1, 3, \ldots, m_{j_k}\}$ and each pair (o, j) appears twice in the string $(o_1, j_1)(o_2, j_2)\ldots(o_l, j_l)$ – once for S_{oj} and once for C_{oj} because each part should be loaded and unloaded only once at each machine. Besides, each part appears from the input hopper M_1 and disappears in the output hopper M_{m-1}, so $E_{o_1 j_1} = S_{1 j_1}$, $E_{o_2 j_2} = C_{1 j_1}$, $E_{o_3 j_3} = S_{3 j_1}$ and $E_{o_{l-2} j_{l-2}} = C_{m_{j_l}-2, j_l}$, $E_{o_{l-1} j_{l-1}} = S_{m_{j_l} j_l}$, $E_{o_l j_l} = C_{m_{j_l} j_l}$.

Let $\delta_\imath \in \{+, -\}$. Then the symbols $M_\imath^+$ and $M_\imath^-$ will mean that the machine $M_\imath$ starts or completes an operation, respectively. Since each event is a start time or a completion time on a machine, the sequence ε defines a unique sequence of machines,

$$\omega = M_1^+ M_1^- M_{\mu_{o_2 j_2}}^+ M_{\mu_{o_3 j_3}}^{\delta_{\mu_{o_3 j_3}}} \ldots M_{\mu_{o_{l-3} j_{l-3}}}^{\delta_{\mu_{o_{l-3} j_{l-3}}}} M_{\mu_{o_{l-2} j_{l-2}}}^- M_{m-1}^+ M_{m-1}^-,$$

that is called a *robot-move sequence*, where $M_1^{\delta_1}$ (as well as $M_{m-1}^{\delta_{m-1}}$) appears exactly n times. Define the pair $M_{\mu_{o_{k-1} j_{k-1}}}^{\delta_{\mu_{o_{k-1} j_{k-1}}}} M_{\mu_{o_k j_k}}^{\delta_{\mu_{o_k j_k}}}$ to be a *basic activity* of the robot. Since we consider only purposeful robot moves and the no-buffer environment, only the following basic activities are possible:

(1) $M_{m-1}^- M_\imath^-$ in the interval $[C_{m_j j}, C_{oj'}]$: after dropping P_j off to the output hopper M_{m-1}, empty move out of M_{m-1} to $M_\imath$, unload $P_{j'}$, where $1 \neq \imath \neq m-1$ and $j \neq j'$;
(2) $M_{m-1}^- M_1^+$ in the interval $[C_{m_j j}, S_{1 j'}]$: after dropping P_j off to the output hopper M_{m-1}, empty move out of M_{m-1} to the input hopper M_1, where $j \neq j'$;
(3) $M_\imath^- M_\jmath^+$ in the interval $[C_{e-1,j}, S_{e+1,j}]$: unload P_j from $M_\imath$, transport P_j to $M_\jmath$, load P_j onto $M_\jmath$, where $\imath \neq \jmath$;
(4) $M_\imath^+ M_\jmath^-$ in the interval $[S_{oj}, C_{o'j'}]$: load P_j onto $M_\imath$, empty move to $M_{\imath'}$, unload $P_{j'}$ from $M_\jmath$, where $1 < \imath < m-1$, $1 < \jmath < m-1$, $\imath \neq \jmath$ and $j \neq j'$;
(5) $M_\imath^+ M_\imath^-$ in the interval $[S_{oj}, C_{oj}]$: load P_j onto $M_\imath$, wait until $M_\imath$ finishes P_j, unload P_j from $M_\imath$;
(6) $M_\imath^+ M_1^+$ in the interval $[S_{oj}, S_{1j'}]$: load P_j onto $M_\imath$, empty move to the input hopper M_1, pick $P_{j'}$ up, where $1 < \imath < m-1$ and $j \neq j'$.

We consider an empty robot move from M_x to M_y to be as a *setup* for M_m with a *setup time* s_{xy}. The *setup-time constraint* is:

$$\text{(SC)} \qquad [C_{e-1,j}, S_{e+1,j}] \neq [E_{oj}, E_{o'j'}] \neq [S_{oj}, C_{oj}] \quad \Longrightarrow \quad E_{o'j'} - E_{oj} \geq s_{\mu_{oj}\mu_{o'j'}}.$$

Define a *composite activity* $A_{\mu_{e-1,j}\mu_{o'j'}}$ to be the composition of the two consecutive basic activities: a transportation in the interval $[C_{e-1,j}, S_{e+1,j}]$ and an empty move in the interval $[S_{e+1,j}, E_{o'j'}]$. Define the time of $A_{\mu_{e-1,j}\mu_{o'j'}}$ to be $t_{\mu_{e-1,j}\mu_{o'j'}} = p_e + s_{\mu_{e+1,j}\mu_{o'j'}}$. We say that the composite activity times satisfy the *triangle inequality* if

$$\text{(TI)} \qquad t_{xy} \leq t_{xz} + t_{zy} \quad \text{for} \quad x, y, z \in \{1, 2, \ldots, m-1\}.$$

Let a sequence ρ on the alphabet $\{M_1^+, M_1^-, \ldots, M_{m-1}^+, M_{m-1}^-\}$ have exactly k entries of M_1^+ and exactly k entries of M_1^-, and let ρ^l denote a repetition of ρ exactly l times. If there exists integer l such that $\omega = \rho^l$, then ρ is a *k-unit robot-move cycle*. We also say that a k-unit robot-move cycle is of *cycle capacity* k. Obviously, $n = kl$.

Define a *heavily robotic shop* to be a robotic shop, where DC and SC are satisfied and preemptions are not allowed. We call a heavily robotic shop *Euclidean* if TI is satisfied. Further we consider only heavily robotic problems with one-unit robot-move cycles because we are not aware of complexity or approximation results obtained in the case $k > 1$. Although any robotic shop has the heavily robotic counterpart, we will consider only heavily robotic cycle or flow shops. However, all results we know in the Euclidean case are devoted only to heavily robotic flow shops. The heavily robotic machine environment will be denoted by the bold letter $\boldsymbol{R}$ or $\mathbb{R}$ in the Euclidean case.

$\alpha_1 \in \{\boldsymbol{R}\alpha_3\alpha_4 C,\ \boldsymbol{R}\alpha_3\alpha_4 F,\ \mathbb{R}\alpha_3\alpha_4 F\}$.
$\alpha_1 = \boldsymbol{R}\alpha_3\alpha_4 C$: a *heavily robotic cycle shop*.
$\alpha_1 = \boldsymbol{R}\alpha_3\alpha_4 F$: a *heavily robotic flow shop*.
$\alpha_1 = \mathbb{R}\alpha_3 F\alpha_4$: a *Euclidean heavily robotic flow shop*.
$\alpha_2 = m$: the number of machines is fixed and equal $m + 3$, *i.e.*, the robot, the input hopper and the output hopper are implicit.
$\alpha_3 \in \{\circ, k\}$ fixes the cycle capacity.
$\alpha_3 = \circ$: the cycle capacity is not fixed.
$\alpha_3 = k$: the cycle capacity is fixed and equal k.
$\alpha_4 \in \{\circ, \rho\}$ fixes the robot-move sequence.
$\alpha_4 = \circ$: the robot-move sequence is not fixed.
$\alpha_4 = \rho$: the robot-move sequence is a repetition of a fixed robot-move cycle ρ.

As Sethi *et al.* [SSSBK92] show there are exactly $m!$ different one-unit robot-move cycles in a no-buffer periodic heavily robotic m-machine flow shop. In the two-machine case, where M_1 is the input hopper, and M_4 is the output hopper, the one-unit robot-move cycles are

$$\rho_1 = M_3^+ M_3^- M_4^+ M_4^- M_1^+ M_1^- M_2^+ M_2^-,$$
$$\rho_2 = M_3^+ M_1^+ M_1^- M_2^+ M_3^- M_4^+ M_4^- M_2^-,$$

In the three-machine case, where M_1 is the input hopper, and M_5 is the output hopper, the one-unit robot-move cycles are:

$$\varrho_1 = M_4^- M_5^+ M_5^- M_1^+ M_1^- M_2^+ M_2^- M_3^+ M_3^- M_4^+,$$
$$\varrho_2 = M_4^- M_5^+ M_5^- M_1^+ M_1^- M_2^+ M_3^- M_4^+ M_2^- M_3^+,$$
$$\varrho_3 = M_4^- M_5^+ M_5^- M_3^- M_4^+ M_1^+ M_1^- M_2^+ M_2^- M_3^+,$$
$$\varrho_4 = M_4^- M_5^+ M_5^- M_2^- M_3^+ M_3^- M_4^+ M_1^+ M_1^- M_2^+,$$
$$\varrho_5 = M_4^- M_5^+ M_5^- M_2^- M_3^+ M_1^+ M_1^- M_2^+ M_3^- M_4^+,$$
$$\varrho_6 = M_4^- M_5^+ M_5^- M_3^- M_4^+ M_2^- M_3^+ M_1^+ M_1^- M_2^+,$$

Further we will use the denotation $\varrho_{i,j,\ldots,k}$ assuming that $\varrho_{i,j,\ldots,k} \in \{\varrho_i, \varrho_j, \ldots, \varrho_k\}$.

2.3. Definition of periodic problems. Let σ be a schedule for a given finite job set $\mathfrak{J}$ with n jobs, and let *unlimited breeding* of σ by an infinite number of shifts by P units in time produce a feasible infinite schedule τ. Then we call the schedule τ *infinitely n-job periodic* with *period* P, the schedule σ a *generator* and the job set $\mathfrak{J}$ a *generating job set* of τ. To stress that τ is generated by σ we will use the denotation τ_σ. New results in this

paper involve only infinitely periodic schedules. We obtain the definition of a *finitely n-job periodic* schedule if we use *limited breeding* of σ by a finite number of shifts by P units in time. In what follows, if the number of jobs in $\mathfrak{J}$ is immaterial, then the term "n-job" will be omitted. Besides, if not stated otherwise, the term "periodic" will mean "infinitely periodic". Define a *periodic problem* to be a problem of finding a periodic schedule. Define a *flow time* of the periodic schedule τ_σ to be the maximum completion time in σ. One of the main points in our approach is to consider any periodic problem as a nonperiodic problem with an additional constraint.

Let O_{ij} have m_{ij} nonpreemptive parts in σ, and let O_{ijk} be the kth nonpreemptive part of O_{ij} in σ. Then O_{ij} can be considered as a chain $O_{ij1}O_{ij2}\dots O_{ijm_{ij}}$, where $m_{ij} = 1$ and $O_{ij} = O_{ij1}$ if O_{ij} is nonpreemptive. Let S_{ijk}, C_{ijk} and p_{ijk} be the start time, the completion time and the length of O_{ijk} in σ, respectively. Thus, $C_{ijk} - S_{ijk} = p_{ijk}$ and $p_{ij1} + p_{ij2} + \dots + p_{ijm_{ij}} = p_{ij}$. Obviously, τ is a feasible periodic schedule if and only if the parts O_{ijk} scheduled on $M_\imath$ do not overlap modulo P. So we have the following *periodicity constraint.*

$$\text{(PC)} \quad (i,j,k) \neq (i',j',k') \;\;\&\;\; \mu_i = \mu_{i'} \quad\Longrightarrow\quad [S_{ijk}, C_{ijk}] \cap [S_{i'j'k'}, C_{i'j'k'}] = \emptyset \bmod P$$

Any problem of finding a periodic schedule, therefore, is a problem of finding a nonperiodic schedule satisfying PC. Hence, any nonperiodic problem is a relaxation of its periodic counterpart. Obviously, $P \geq \max_{1\leq\imath\leq m} \sum_{\imath=\mu_{ij}} p_{ij}$. We call a problem *nonperiodic* if it is free of PC.

Any periodic nonpreemptive cycle-shop schedule defines a sequence $o_\imath = O_{i_1}O_{i_2}\dots O_{i_{\mathrm{m}_\imath}}$ of operations on $M_\imath$ for each $\imath = 1,2,\dots,m$, where $0 \leq S_{i_1} < S_{i_2} < \dots < S_{i_{\mathrm{m}_\imath}} < P$. The set of sequences $\{o_1, o_2, \dots, o_m\}$ is a *cyclic machine structure* [MR94] of the cycle shop.

$\varsigma \in \{\circ, \dagger\}$ fixes the cyclic machine structure.
$\varsigma = \circ$: the cyclic machine structure is not fixed.
$\varsigma = \dagger$: the cyclic machine structure is fixed.

Let $s_\imath$ and $c_\imath$ be the earliest start time and the latest completion time on $M_\imath$ in σ. Thus, $B_\imath = c_\imath - s_\imath$ is the *busy time* of $M_\imath$ in σ. Define $B_{\max}$ to be the maximum busy time in σ, *i.e.*, $B_{\max} = \max_{1\leq\imath\leq m} B_\imath$. We call a periodic schedule τ_σ with period P *trivially periodic* if $B_{\max} \leq P$. It is easy to see that this inequality implies PC, and a minimum-period trivially periodic schedule τ_σ is a result of concatenation of copies of σ, where $B_{\max} = P$. Hence, finding a trivially periodic schedule of minimum period reduces to finding a nonperiodic schedule of minimum maximum busy time. Note that the maximum busy time in heavily robotic shops is always reached on M_m, *i.e.*, on the robot, and called the robot *cycle time.* Since all the events in the sequence ε can be taken modulo P, we can get the anologous sequence of *events modulo* P in the interval $[0, P-1]$. Hence, any heavily robotic machine environment has the periodic counterpart.

Periodic problems in this paper are based on the above periodicity concept. We avoid considering periodic problems on identical parallel machines [HM95, M96] and in *recurrent job shops* [H94] with *potential* or *conjunctive constraints* [CC90, H94] originated from pipeline computing [K81].

2.4. Job characteristics and minimization criteria. In comparision with the classification of Hall *et al.* [HKS97] for periodic robotic flow-shop problems, our classification includes them as well but remains the job characteristics field the same as in the well-known classification of classic nonperiodic problems, *cf.* [LLRKS93, B95].

β $\subset \{\beta_1, \ldots, \beta_8\}$ introduces *preemption, no-wait, precedence-relation, release-date, jobs-number, operations number, processing-time* constraints, respectively.

β_1 $= pmtn$: any job can be interrupted at any time and resumed may be on another machine with a possible delay.

β_2 $= no\,wait$: delays in processing jobs are not allowed. For robotic flow or cycle shops this means that each part is being processed on a machine or held by the robot at any time during a schedule.

β_3 $\in \{prec, intree, outtree, chains\}$ denotes the general, intree, outtree or chains precedence relation on $\mathfrak{J}$. We use the denotation $\beta_3 + chains$ for a *gomeomorph* of β_3, *i.e.*, for a precedence relation obtained from β_3 by inserting chains.

β_4 $= r_j$: J_j becomes available for processing at time r_j.

β_5 $\in \{n = k, n \le k, n \ge m-2\}$: the number of jobs is fixed and equal to constant k; the number of jobs is bounded by constant k; the number of jobs is at least $m-2$.

β_6 $\in \{m_j = mh_j, \ell = mh, \ell = 2h-1, \ell = k\}$: the numbers of operations of jobs in a job or cycle shop are positive and divisible by m, where $h > 1$, odd or fixed and equal to constant k.

β_7 $\in \{p_{ij} = p_i, p_{ij} = q_{\mu_{i'}}, p_i = q_{\mu_i}/q, p_{ij} = mg_{ij}, p_{ij} = p, p_{ij} = 1, p_{ij} \in \{p, q\}, [p_i]\}$, where $i' \in \{i, i-1\}$, will mean that: the processing times do not depend on jobs; depend only on machines; are divisible by the processing time q on the machine with maximum multiplicity (only for periodic cycle shops); are divisible by m; equal; unit-time; equal p or q; *processing windows* are specified, *i.e.*, $l_i \le p_i \le u_i$ for fixed lower bound l_i and upper bound u_i (only for one-job periodic robotic flow shops), respectively. In the case of identical jobs "p_{ij}" will be replaced by "p_i".

β_x $= \circ$: the constraint β_x is removed.

β_{x-y} $= \beta_x, \beta_{x+1}, \ldots, \beta_y$, the constraint sequence, where $1 \le x < y \le 7$.

In the following minimization criteria γ we assume that C_{ij} is the completion time of O_{ij}, $C_{\max} = \max_{1 \le j \le n} C_j$, $L_{\max} = \max_{1 \le j \le n} L_j$, where $L_j = C_j - d_j$, $f_{\max} = \max_{1 \le j \le n} f_j(C_j)$, $\sum f_j = \sum_{j=1}^{n} f_j(C_j)$, where f_j are nondecreasing cost functions with values computable in constant time, $T_j = \max\{0, L_j\}$, $U_j = 0$ if $T_j = 0$ and $U_j = 1$ otherwise. Thus, C_j, L_j, T_j, U_j are special cases of $f_j(C_j)$.

$\gamma = B_{\max}$: the *maximum busy time.*
$\gamma = C_{\max}$: the *maximum completion time, i.e.*, the *schedule length.*
$\gamma = L_{\max}$: the *maximum lateness.*
$\gamma = f_{\max}$: the *maximum cost.*
$\gamma = \sum C_j$: the *total completion time.*
$\gamma = \sum C_{ij}$: the *total operational completion time.*
$\gamma = \sum U_j$: the *number of late jobs.*
$\gamma = \sum T_j$: the *total tardiness.*
$\gamma = \sum w_j C_j$: the *total weighted completion time.*
$\gamma = \sum w_j U_j$: the *weighted number of late jobs.*
$\gamma = \sum w_j T_j$: the *total weighted tardiness.*
$\gamma = \sum f_j$: the *total cost.*
$\gamma = P$: the *period.*
$\gamma = P * C_{\max}$: the *Pareto criterion* on P and $C_{\max}$.
$\gamma = aP + bC_{\max}$: the *linear combination* of P and $C_{\max}$.

2.5. Assumptions, relationship and terminology. We assume that $\gamma = P$ implies a periodic problem. In periodic problems with $B_{\max} \neq \gamma \neq P$ the period P will be considered to be given. Denotations of nonperiodic problems with $\gamma = B_{\max}$ will also denote the related trivially periodic problems since they are equivalent. To distinguish periodic problems with $\gamma \neq P$ from their nonperiodic counterparts the last letter in denotations of their machine environments will be caligraphic.

$\alpha_1 = R\mathcal{F}$: a *periodic robotic flow shop* with $B_{\max} \neq \gamma \neq P$.
$\alpha_1 = \boldsymbol{R}\alpha_3\alpha_4\mathcal{F}$: a *periodic heavily robotic flow shop* with $B_{\max} \neq \gamma \neq P$.
$\alpha_1 = \mathcal{C}$: a *periodic cycle shop with* $B_{\max} \neq \gamma \neq P$.
$\alpha_1 = \mathrm{C}$: a *finitely periodic cycle shop.*

For example, $\mathcal{C}\|C_{\max}$ and $\boldsymbol{R}\mathcal{F}\|\sum C_j$ denote periodic problems, meanwhile $C\|C_{\max}$ and $\boldsymbol{R}F\|\sum C_j$ denote their nonperiodic counterparts, respectively. $C\|P$, $C\|B_{\max}$, $\boldsymbol{R}F\|P$, $\boldsymbol{R}F\|B_{\max}$ and $RF\|P$ denote periodic problems. Note that $RF\alpha_2|\beta|\gamma$ and $RC\alpha_2|\beta|\gamma$ are special cases of $1C\alpha_2|\beta|\gamma$ and $C\alpha_2|\beta|\gamma$, respectively, and that $C\alpha_2|\beta_{1-4}, n = 1, \beta_{6-7}|\gamma$ and $J\alpha_2|\beta_{1-4}, n = 1, \beta_{6-7}|\gamma$ are identical. The same is true in the periodic case. Since the maximum busy time and the maximum completion time are always reached on M_1 in loop shops, $L\alpha_2|\beta|B_{\max}$ and $L\alpha_2|\beta|C_{\max}$ are also identical. The following relationship is already not so evident. Define a *flow component* of a cycle shop to be any maximal subsequence of adjacent unimachines in the sequence $M_{\mu_1}M_{\mu_2}\ldots M_{\mu_\ell}$. Define $\mathfrak{f}$, the *flow size* of a cycle shop, to be the length of its longest flow component.

Lemma 2.1. *Let A be $R\mathcal{F}|n = 1, p_o = q_{\mu_o}, p_e = 1|C_{\max}$, B be $1\mathcal{C}|n = 1, p_i = 1|C_{\max}$, and let P_B, $\mathfrak{f}_B$ be the given period and the flow size of the cycle shop in B, respectively. Then $A \propto B$, but $B \propto A$ only if $P_B \geq \mathfrak{f}_B$.*

Proof. Let P_A denote the period in A, and let $\mathfrak{m}_A$ and $\mathfrak{m}_B$ denote multiplicities of the cycle shops in A and B, respectively. Obviously, $P_A \geq \max\{\mathfrak{m}_A, q_{\max}\}$, where $q_{\max} = \max_\imath q_\imath$, and $P_B \geq \mathfrak{m}_B$. Without loss of generality we assume that M_1 is the multimachine in both problems. For the reduction $A \propto B$, replace each unimachine in an instance of A with processing time q by q unimachines with unit processing time and put $P_B = P_A$. This produces an instance of B which obviously has the same minimum $C_{\max}$. If $P_B \geq \mathfrak{f}_B$, the reduction $B \propto A$ can be obtained by replacing each flow component of length q in an instance of B by one unimachine with processing time q and putting $P_A = P_B$. If $\mu_1 = 1$ or/and $\mu_\ell = 1$ in B, add one artificial starting or/and finishing unit-time operation on one or two artificial unimachines, respectively. Depending on zero, one or two artificial unimachines is added, the minimum $C_{\max}$ for the constructed instance of A will equal, exceed by one or two the minimum $C_{\max}$ for the instance of B. Since $\mathfrak{f}_B = q_{\max}$ and $P_B \geq \mathfrak{m}_B = \mathfrak{m}_A$, we have $P_A \geq \max\{\mathfrak{m}_A, q_{\max}\}$. □

In the literature devoted to heavily robotic flow shops the set of parts associated with $\mathfrak{J}$ is a so called MPS, *i.e.*, a *minimal part set*, where the part types are presented in certain proportions. Also, several authors use the term "cyclic scheduling" instead of "periodic scheduling" and "cycle time" instead of "period". The latter two terms, however, are not

synonyms in our classification because they mean $B_{\max}$ and P, respectively. In several papers the cycle time has been denoted as C_t, but we do not use it to avoid an association with the completion time. Besides, $B_{\max}$ we found more informative.

Rapidly growing literature on the complexity of periodic scheduling problems is primarily devoted to infinitely periodic problems, related exact or heuristic algorithms and NP-hardness proofs. However, performance guarantee approximations for infinitely periodic problems and finitely periodic problems have been barely attracted someone's attention.

2.6. Complexity and approximation results. The following lists include only problems with positive cyclic rank: maximal solvable in polynomial or pseudopolynomial time, minimal NP-hard and having performance guarantee approximations computable in polynomial time. Two-machine robotic flow-shop problems [BK, K99] are not included since their cyclic rank is zero.

Solvable in polynomial time		
$L2\|prec, r_j, p_{ij}=1\|L_{\max}$	$O(n^2)$	[Section 4.2]
$RF3\|p_{oj}=1, p_{ej}=q_{\mu_{e-1}}\|L_{\max}$	$O(n^2)$	[K99]
$RF3\|r_j, p_{oj}=1, p_{ej}=q_{\mu_{e-1}}\|C_{\max}$	$O(n^2)$	[K99]
$RF\|n \geq m-2, p_{oj}=p, p_{ej}=q_{\mu_{e-1}}\|C_{\max}$	$O(1)$	[HK98]
$\mathbb{R}F2\|no\ wait\|B_{\max}$	$O(h \log h)$	[A99]
$\boldsymbol{R}F2\|\|B_{\max}$	$O(n \log n)$	[AK99]
$\boldsymbol{R}1\rho_2\mathcal{F}2\|\|C_{\max}$	$O(n \log n)$	[KSI91]
$\boldsymbol{R}1\rho F2\|\|B_{\max}$	$O(n \log n)$	[SSSBK92]
$\boldsymbol{R}1\varrho_{1,3,4,5}F3\|\|B_{\max}$	$O(n \log n)$	[HKS97]
$\boldsymbol{R}1\varrho_{1,4}F3\|no\ wait\|B_{\max}$	$O(n \log n)$	[AP98]
$\boldsymbol{R}1F\|no\ wait, n=1\|P$	$O(m^3 \log m)$	[KL98]
$\boldsymbol{R}1C\|no\ wait, n=1\|P$	$O(\ell^5)$	[KL97]
$\mathbb{R}1F\|n=1\|P$	$O(m^3)$	[CK97]
$C2\|p_{ij}=q_{\mu_i}\|C_{\max}$	$O(1)$	[T85]
$C2\|p_{ij}=1\|\sum C_j$	$O(1)$	[Section 4.1]
$C2\|p_{ij}=1\|\sum U_j$	$O(n^7)$	[K99A]
$C2\|no\ wait, p_{ij}=1\|\sum f_j$	$O(n^3)$	[Section 4.1]
$C2\|r_j, p_{ij}=1\|C_{\max}$	$O(n^2)$	[T97]
$C\|p_{ij}=1\|C_{\max}$	$O(\ell)$	[T85]
$\mathrm{C}\|p_{ij}=1\|P*C_{\max}$	$O(\ell)$	[T85]
$C\|prec, r_j\|P$	$O(\ell n)$	[Section 5.1]
$C\|no\ wait, n=1, p_i=1\|P$	$O(\ell^2-\mathfrak{m}\ell)$	[Section 5.2]
Solvable in pseudopolynomial time		
$RF3\|r_j, p_{oj}=p, p_{ej}=q\|C_{\max}$	$O(\ell^4 n^5)$	[M99]
$RF3\|r_j, p_{oj}=p, p_{ej}=q\|\sum C_j$	$O(\ell^4 n^5)$	[M99]
$C2\|r_j, p_{ij}=1\|\sum C_j$	$O(\ell^4 n^5)$	[M99]
$C2\|no\ wait, r_j, p_{ij}=1\|\sum C_j$	$O(\ell^4 n^5)$	[M99]
$C2\|no\ wait, r_j, p_{ij}=1\|C_{\max}$	$O(\ell^4 n^5)$	[M99]
$C2\|prec, r_j, p_{ij}=1\|L_{\max}$	$O(\ell^2 n^2)$	[Section 4.2]
$1C\|no\ wait, n=1, p_i=q_{\mu_i}/q\|P$	$O(\mathfrak{m}^2 m q_{\max})$	[Section 5.2]

NP-hard

$L2\|\|C_{\max}$		[LA84,T85]
$L2\|r_j\|C_{\max}$	strongly	[T85]
$L2\|chains\|C_{\max}$	strongly	[T85]
$RF3\|p_{oj}=1\|\sum w_jU_j$		[K99]
$RF3\|p_{oj}=1\|\sum T_j$		[K99]
$RF3\|p_{oj}=1\|\sum w_jT_j$	strongly	[K99]
$RF3\|p_{oj}=p\|L_{\max}$	strongly	[HK98]
$RF3\|r_j,p_{oj}=p\|C_{\max}$	strongly	[HK98]
$RF3\|r_j,p_{oj}=1\|L_{\max}$	strongly	[K99]
$RF3\|r_j,p_{oj}=1\|\sum C_j$	strongly	[K99]
$\boldsymbol{R}F3\|\|B_{\max}$	strongly	[HKS98]
$\boldsymbol{R}1_{\varrho_{2,6}}F3\|\|B_{\max}$	strongly	[HKS98]
$\boldsymbol{R}1_{\varrho_{2,6}}F3\|no\,wait\|B_{\max}$	strongly	[AP98]
$\boldsymbol{R}1F\|n=1\|P$	strongly	[LW89,BFK97]
$\mathbb{R}1F\|n=1,[p_o]\|P$	strongly	[CK97A]
$R\mathcal{F}\|n=1,p_o=q_{\mu_o},p_e=1\|C_{\max}$	strongly	[Section 5.3]
$C2\|p_{ij}\in\{1,2\}\|C_{\max}$	strongly	[T81]
$C2\|no\,wait,chains,p_{ij}=1\|\sum w_jC_j$	strongly	[Section 4.3]
$C2\|no\,wait,chains,p_{ij}=1\|\sum T_j$	strongly	[Section 4.3]
$C2\|no\,wait,chains,p_{ij}=1\|\sum U_j$	strongly	[Section 4.3]
$C\|no\,wait,intree,r_j,p_{ij}=1\|\sum C_j$	strongly	[Section 4.3]
$C\|no\,wait,intree,r_j,p_{ij}=1\|C_{\max}$	strongly	[Section 4.3]
$C\|no\,wait,outtree,p_{ij}=1\|L_{\max}$	strongly	[Section 4.3]
$C\|no\,wait,prec,p_{ij}=1\|C_{\max}$	strongly	[Section 4.3]
$C\|no\,wait,prec,p_{ij}=1\|\sum C_j$	strongly	[Section 4.3]
$C\|no\,wait,r_j,p_{ij}=1\|\sum U_j$	strongly	[Section 4.3]
$C\|p_{ij}=1\|\sum w_jU_j$		[Section 4.3]
$C\|r_j,p_{ij}=1\|\sum w_jC_j$	strongly	[Section 4.3]
$1\mathcal{C}\|n=1,p_i=1\|\sum C_{ij}$	strongly	[Section 5.3]
$1\mathcal{C}\|n=1,p_i=1\|C_{\max}$	strongly	[Section 5.3]
$\mathcal{C}^2_\dagger\|n=1,p_i\in\{1,q\}\|C_{\max}$	strongly	[MR94]

Polynomial-time approximations

$L2\|\|C_{\max}$	$O(n\log n)$	$\frac{C_{\max}}{C^*_{\max}}\le\frac{4}{3}$	[DS99]
$L2\|pmtn\|C_{\max}$	$O(n^2)$	$\frac{C_{\max}}{C^*_{\max}}\le\frac{3}{2}$	[HWZ98]
$C\|p_{ij}=q_{\mu_i}\|C_{\max}$	$O(\ell^2)$	$\frac{C_{\max}}{C^*_{\max}}\le 1+\frac{3m+\ell}{n}$	[T86]
$C\|\|C_{\max}$	$O(\ell^2)$	$\frac{C_{\max}}{C^*_{\max}}\le\frac{p_{\max}}{p_{\min}}\left(1+\frac{3m+\ell}{n}\right)$	[T86]
$1\mathcal{C}\|n=1,p_i=1\|C_{\max}$	$O(\mathtt{m}^2)$	$C^1_{\max}-C^*_{\max}\le\frac{\mathtt{m}^2-\mathtt{m}}{2}$	[Section 5.4]
	$O(\mathtt{m}^2)$	$C^2_{\max}-C^*_{\max}\le\frac{\mathtt{m}^2-\mathtt{m}^{3/2}/14}{2}$	[Section 5.5]

As we show further, even $J|prec,r_j|P$ can be solved in polynomial time. An $O(n\log n)$ algorithm of Aneja and Kamoun [AK99] for $\boldsymbol{R}F2\|\|B_{\max}$ is an improvement of an $O(n^4)$ algorithm of Hall *et al.* [HKS97]. An $O(m^3\log m)$ algorithm of Kats and Levner [KL98] for $\boldsymbol{R}1F|no\,wait,n=1|P$ is the best result in a series of improvements of an algorithm of Kats and Mikhailetsky [KM80], which has been reviewed by Mikhalevich *et al.* [MBM88], Kats and Levner [KL97] and Levner *et al.* [LKL97]. A *Nearest-Window-First* (NWF) algorithm

proposed by Degtiarev and Timkovsky [DT76] for approximation of $C|p_{ij}=q_{\mu_i}|C_{\max}$ solves $C|n=1,p_{ij}=q_{\mu_i}|P$ exactly [T86, T92]. Pinedo [P95, p. 265] describes almost the same algorithm for cycle shops of rank one. An absolute error bound of the NWF algorithm in application to $\mathcal{C}|n=1,p_i=q_{\mu_i}|C_{\max}$ will be derived in Section 5. An $O(\ell)$ algorithm from [T85] solves both $C|p_{ij}=1|C_{\max}$ and $\mathrm{C}|p_{ij}=1|P*C_{\max}$ producing a finite periodic schedule with period one. Note that an $O(n)$ algorithm of Hall *et al.* [HLP97] for $J|m_j\le 2|B_{\max}$ gives nothing for cycle shops of positive rank because $C|\ell=2|B_{\max}$ is actually $F2\|B_{\max}$.

As we show in Section 4.2, a polynomial-time algorithm of Bruno *et al.* [BJS80] for $F2|prec,r_j,p_{ij}=1|L_{\max}$ solves $C2|prec,r_j,p_{ij}=1|L_{\max}$ in pseudopolynomial time. Observe that this result puts some light on the complexity of $J2|r_j,p_{ij}=1|L_{\max}$, which remains open, because $C2|r_j,p_{ij}=1|L_{\max}$ turns to be its nontrivial pseudopolynomially-solvable special case with different release dates.

$L2\|C_{\max}$ remains open for the ordinary or strong NP-hardness. The NP-hardness of $L2\|C_{\max}$, $C2|p_{ij}\in\{1,2\}|C_{\max}$, $1\mathcal{C}|n=1,p_i=1|C_{\max}$ and $\mathbb{R}1F|n=1,[p_o]|P$ strengthens earlier NP-hardness results of Lenstra *et al.* [LRKB77] for $J2|m_j\le 3|C_{\max}$, Lenstra and Rinnooy Kan [LRK79] for $J2|p_{ij}\in\{1,2\}|C_{\max}$, Roundy [R92] for $\mathcal{C}|n=1|C_{\max}$, and Livshits *et al.* [LMC74] for $\boldsymbol{R}1F|n=1,[p_o]|P$, respectively. Note that the NP-hardness proof of Hall *et al.* [HLP97] for $J2|m_j=3|B_{\max}$ fits for $L2\|B_{\max}$ in fact.

All the NP-hardness results for problems $RF3|\beta_4,p_{ij}=1|\gamma$ follow from the NP-hardness of the one-machine counterparts $1\|\sum w_jU_j$, $1\|\sum T_j$, $1|r_j|L_{\max}$, $1|r_j|\sum C_j$ and $1\|\sum w_jT_j$ [K99]. All the NP-hardness results in Section 4.3 are obtained by the cycle-shopping theorem. The strong NP-hardness proofs of McCormick and Rao [MR94] for $F|no\,wait,p_i\in\{0,1\}|B_{\max}$ and $F3|no\,wait|B_{\max}$ do not fit for the related cycle-shop couterparts with positive cyclic rank. The complexity status of $C2|no\,wait|B_{\max}$ is also unknown.

The $\frac{4}{3}$-approximation of Drobouchevitch and Strusevich [DS99] for $L2\|C_{\max}$ is an improvement of the $\frac{3}{2}$-approximation obtained earlier by Gupta [G93] and, independently, Wang *et al.* [WSV97]. The $\frac{3}{2}$-approximation for $L2|pmtn|C_{\max}$ is a straightforward corollary from the same result obtained by Han *et al.* [HWZ98] for $J2|pmtn|C_{\max}$. The approximations from [T86] are obtained by the NWF algorithm which also provides performance ratio $\frac{C_{\max}}{C^*_{\max}}\le\frac{p_{\max}}{p_{\min}}\left(1+\frac{m}{n}\right)$ for $F\|C_{\max}$. Note that $C^2_{\max}$ is better than $C^1_{\max}$ only if $m>196$.

Serafini and Ukovich [SU89] describe a general model for periodic activities that covers $C|n=1|P$ and show a reduction of the problem to finding a critical circuit in a network. Roundy [R92] studies the structure of solutions of $\mathcal{C}|n=1|P*C_{\max}$. He also proposes an algorithm searching for local Pareto optimums. Rao and Jackson [RJ93] consider mixed integer linear programming models for $\mathcal{C}|n=1|\sum w_jC_j$ and $\mathcal{C}|n=1|aP+bC_{\max}$.

Note that $\mathcal{C}_\dagger|n=1|C_{\max}$ is a special case of $\mathcal{C}|n=1|C_{\max}$. So, the NP-hardness result of McCormick and Rao [MR94] for $\mathcal{C}^2_\dagger|n=1,p_i\in\{1,q\}|C_{\max}$ is another strengthening of Roundy's NP-hardness result [R92].

Timkovsky [T77] and McCormick *et al.* [MPSW91] consider the problem of scheduling *transient processes* in cycle shops when switching from one periodic schedule, representing a *steady state* of a manufacturing system, to another. A similar problem of obtaining the first steady state in three-machine heavily robotic flow shops has been studyed by Hall *el al.* [HKS98].

3. Excerptions and easy corollaries for nonperiodic problems

If processing times of jobs are divisible by m, then we refer to modulated or perfect schedules and problems as schedules and problems for modulated identical parallel machines or a perfect job shop, respectively. Our special attention will be paid to $P \bmod \alpha_2 | \beta_1, \beta_{3-5}, p_j = mg_j | \gamma$, $J \,\mathrm{per}\, \alpha_2 | \beta_{2-5}, m_j = mh_j, p_{ij} = 1 | \gamma$ and $C \,\mathrm{per}\, \alpha_2 | \beta_{2-5}, m_j = mh_j, p_{ij} = 1 | \gamma$. The following isomorphisms reveal a key connection between scheduling on identical parallel machines and unit-time job-shop scheduling.

Lemma 3.1. [Job-shopping isomorphisms][T98]

$$\begin{aligned} P \bmod \alpha_2 | pmtn, \beta_{3-5}, p_j = mg_j | \gamma &\approx J \,\mathrm{per}\, \alpha_2 | \beta_{3-5}, m_j = mg_j, p_{ij} = 1 | \gamma, \\ P \bmod \alpha_2 | \beta_{3-5}, p_j = mg_j | \gamma &\approx J \,\mathrm{per}\, \alpha_2 | no\ wait, \beta_{3-5}, m_j = mg_j, p_{ij} = 1 | \gamma. \end{aligned}$$

As it was shown in [T98], for any perfect job-shop scheduling problem with a criterion γ, which is a nondecreasing function of job completion times, it is sufficient to only consider schedules where the length of every uninterrupted part of every job is divisible by m. Hence, to determine a perfect job-shop schedule it is sufficient to indicate only start times and lengths of uninterrupted parts of jobs because the unit-time operations inside the parts are uniformly distributed among the machines. Besides, the first isomorphism in Lemma 3.1 implies that for any modulated scheduling problem on identical parellel machines, where the criterion γ has the same property, it is sufficient to only consider schedules where the length of every nonpreemptive part of every job is also divisible by m. We will use the following special case of Lemma 3.1, where $g_j = h$ for $j = 1, 2, \ldots, n$.

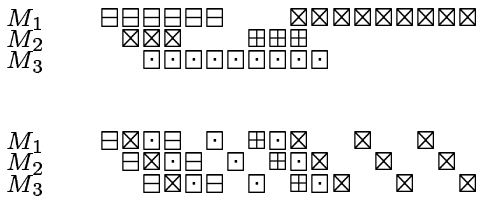

Figure 3.1. A modulated three-identical-parallel-machines schedule and the isomorphic perfect three-machine unit-time job-shop schedule.

Lemma 3.2. [Cycle-shopping isomorphisms]

$$\begin{aligned} P \bmod \alpha_2 | pmtn, \beta_{3-5}, p_j = mh | \gamma &\approx C \,\mathrm{per}\, \alpha_2 | \beta_{3-5}, \ell = mh, p_{ij} = 1 | \gamma, \\ P \bmod \alpha_2 | \beta_{3-5}, p_j = mh | \gamma &\approx C \,\mathrm{per}\, \alpha_2 | no\ wait, \beta_{3-5}, \ell = mh, p_{ij} = 1 | \gamma. \end{aligned}$$

The job-shopping theorem that provides a general polynomial-time reduction of problems on identical parallel machines to unit-time job-shop problems is based on Lemma 3.1. Here we give it in an extended form which highlights additional properties of the reduction that run out from the theorem proof.

Theorem 3.1. [Job-shopping theorem][T98]

If $\gamma \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_j C_j, \sum T_j, \sum w_j T_j, \sum U_j, \sum w_j U_j\}$, *then*

$$\begin{aligned} P\alpha_2|pmtn, \beta_{3-5}|\gamma &\propto J\alpha_2|\beta_{3-5}, p_{ij}=1|\gamma, \\ P\alpha_2|\beta_{3-5}|\gamma &\propto J\alpha_2|no\,wait, \beta_{3-5}, p_{ij}=1|\gamma, \text{ where:} \end{aligned}$$

- *the images of the reductions consist of perfect unit-time job-shop problems;*
- *the reductions conserve the number of jobs, translating the processing times* p_j *of jobs in problems on identical parallel machines into the numbers* m_j *of operations of jobs in perfect unit-time job-shop problems such that* $m_j = cp_j$ *for a common positive integer* c *divisible by* m *with* $c > m^2$.

Since the reductions in Theorem 3.1 do not change the relative length of jobs and a perfect unit-time job-shop problem with jobs of equal length is obviously a unit-time cycle shop problem, we have the following corollary for cycle shops.

Theorem 3.2. [Cycle-shopping theorem]

If $\gamma \in \{C_{\max}, L_{\max}, \sum C_j, \sum w_j C_j, \sum T_j, \sum w_j T_j, \sum U_j, \sum w_j U_j\}$, *then*

$$\begin{aligned} P\alpha_2|pmtn, \beta_{3-5}, p_j=p|\gamma &\propto C\alpha_2|\beta_{3-5}, p_{ij}=1|\gamma, \\ P\alpha_2|\beta_{3-5}, p_j=p|\gamma &\propto C\alpha_2|no\,wait, \beta_{3-5}, p_{ij}=1|\gamma, \text{ where:} \end{aligned}$$

- *the images of the reductions consist of perfect unit-time cycle-shop problems in a cycle shop of rank* m *and multiplicity more than* m;
- *the reductions conserve the number of jobs, translating processing time* p *of jobs in equal-time problems on identical parallel machines into the number* ℓ *of operations of jobs in perfect unit-time cycle-shop problems such that* $\ell = cp$ *for a positive integer* c *divisible by* m.

Note that the job-shopping involves the *preemptive–no-no-wait* correspondences and the *nonpreemptive–no-wait* correspondences (*i.e.*, the isomorphisms and the reductions). The cycle-shopping is just a contraction of these correspondences. Meanwhile, the analogous derivatives from job-shopping for flow shops [BK98, T98A] and open shops [T98A] are rather slight modifications of only the preemptive–no-no-wait correspondences that, besides, hold for all the criteria except the total completion time.

4. Nonperiodic cycle shops

4.1. Two-machine problems solvable in polynomial time. The size of $C2|r_j, p_{ij}=1|C_{\max}$ and $C2|p_{ij}=1|\sum U_j$ is $n+\log \ell+\sum_{j=1}^{n} \log r_j$[or d_j]. The size of the job-shop counterparts, $J2|r_j, p_{ij}=1|C_{\max}$ and $J2|p_{ij}=1|\sum U_j$, is $n+\sum_{j=1}^{n} \log m_j+\sum_{j=1}^{n} \log r_j$[or d_j]. Since the job-shop problems can be solved in polynomial times $O(n^2)$ [T97] and $O(n^7)$ [K99A], respectively, which are polynomial in the size of the cycle-shop problems as well, the

latter can be solved in polynomial time by the same algorithms. The *Earliest-Completion-First* (ECF) algorithm solves $J2|p_{ij}=1|\sum C_j$, which size is $n+\sum_{j=1}^{n}\log m_j$, in polynomial time $O(n\log n)$ [K94, KT96]. In application to $C2|p_{ij}=1|\sum C_j$ this time is pseudopolynomial because the size of the cycle-shop problem is $\log n+\log \ell$. To explain how to get a polynomial-time solution let us consider the ECF algorithm.

ECF algorithm. Split each job into two uninterrupted parts, where the first part includes only the first operation, the second part includes the remaining operations. And then fill the processing time of the two machines by the parts following the ECF rule, *i.e.*, at each step, schedule the part of a job which, being scheduled without interruptions, has the earliest completion time.

It is easy to check that no advantage to interruptions of jobs in application of the ECF algorithm to $C2|p_{ij}=1|\sum C_j$, and the jobs start in an ECF schedule without interruptions at times $k(\ell+1)$ and $k(\ell+1)+1$, $k=0,1,\ldots$, where $\frac{n}{2}(\ell+1)+1$ or $\frac{n-1}{2}(\ell+1)$ is the last start time if n is even or odd, respectively. Since the jobs are identical, the ECF schedule can be found with constant time and space requirements. Note that $J2|r_j,p_{ij}=1|L_{\max}$ remains open even for pseudopolynomial-time solution. In the next subsection, however, we show that the cycle-shop counterpart of this problem can be solved in pseudopolynomial-time. In the case of equal-release-date no-wait jobs we can get, however, that two-machine cycle-shop problems with more general criteria can be solved in polynomial time.

Lemma 4.1. *$P\bmod|p_j=mh|\sum f_j$ can be solved in polynomial time $O(n^3)$.*

Proof. Since f_j are nondecreasing functions, we can consider only compact optimal schedules, where each machine has no idling. Hence, the start times of jobs in any compact optimal schedule for $P\bmod|p_j=mh|\gamma$ are $0,1,\ldots,m-1,\ mh,mh+1,\ldots,2mh-1,$ $2mh,2mh+1,\ldots,3mh-1,\ \ldots$ which we denote as $S_1,S_2,\ldots,S_n$, respectively. They are all different, therefore, the problem is to find a minimum-total-cost assignment of n jobs to n start times, where the assignment of J_j to S_k costs $f_j(S_k+mh)$. But the assignment problem can be solved in polynomial time $O(n^3)$ [GJ79]. □

Lemma 4.2. *$C\,\mathrm{per}\,|no\,wait,\ell=mh,p_{ij}=1|\sum f_j$ can be solved in polynomial time $O(n^3)$.*

Proof. Follows from Lemmas 3.2 and 4.1. □

Lemma 4.3. *$C2|\beta_{2-5},\ell=2h-1,p_{ij}=1|\gamma\ \propto\ C2|\beta_{2-5},\ell=2h,p_{ij}=1|\gamma$.*

Proof. Let I be an instance of $C2|\beta_{2-5},\ell=2h-1,p_{ij}=1|\gamma$. Construct the instance I' of $C2|\beta_{2-5},\ell=2h,p_{ij}=1|\gamma$ from I by appending one operation to the beginning of each job and increasing all due dates (if they are given) by one. It is easy to make sure that each compact optimal schedule for I can be obtained from a compact optimal schedule for I' by deleting all appended operations and shifting time by one unit back. □

Theorem 4.1. *$C2|no\,wait,p_{ij}=1|\sum f_j$ can be solved in polynomial time $O(n^3)$.*

Proof. Follows from Lemma 4.2 for even ℓ because of the obvious isomorphism $C2|\beta_{2-5},\ell=2h,\beta_7|\gamma\ \approx\ C\,\mathrm{per}\,2|\beta_{2-5},\ell=2h,\beta_7|\gamma$, and Lemma 4.3 for odd ℓ. □

Corollary 4.1. *$C2|no\,wait, p_{ij}=1|\sum w_jU_j$ and $C2|no\,wait, p_{ij}=1|\sum w_jT_j$ can be solved in polynomial time $O(n^3)$.*

Note that the related job-shop counterparts are NP-hard [T85, SL86, T98] and that an $O(n^6)$ algorithm of Kravchenko [K98] for $J2|no\,wait, p_{ij}=1|\sum C_j$ solves $C2|no\,wait, p_{ij}=1|\sum C_j$ only in pseudopolynomial time because the size of the latter problem is $\log n+\log \ell$.

4.2. A two-machine problem solvable in pseudopolynomial time. Using Lemma 4.3 and a *chaining* technique introduced by Brucker and Knust [BK98] we can get the following reduction.

Theorem 4.2. *If $\gamma \neq \sum C_j$, then there is a reduction*

$$C\,\mathrm{per}\,\alpha_2|\beta_{2-5}, \ell = mh, p_{ij}=1|\gamma \;\propto\; F\alpha_2|\beta_2, \beta_3 + chains, \beta_{4-5}, p_{ij}=1|\gamma,$$

which increases the number of jobs by ℓ/m times.

Proof. Each instance of the former problem can be converted into an instance of the latter by considering jobs with processing time mh as chains of h jobs with processing time m. If J_j is a job with release date r_j, due date d_j and weight w_j of an instance of the former problem, then the new h jobs in the related chain have common release date r_j, due dates $D, D, \ldots, D, d_j$, where D is a large number bounded by a polynomial in the problem size, and weights $0, 0, \ldots, 0, w_j$, respectively. It is easy to see that the reduction is a trivial isomorphism for the criterion $C_{\max}$ which does not involve due dates and weights, and that the reduction does not hold for the criterion $\sum C_j$ due to the absence of weights. □

Corollary 4.2. *$C2|prec, r_j, p_{ij}=1|L_{\max}$ can be solved in pseudopolynomial time $O(\ell^2n^2)$.*

Proof. Follows from Lemmas 4.3 and Theorem 4.2 because $F2|prec, r_j, p_{ij}=1|L_{\max}$ can be solved in polynomial time $O(n^2)$ by an algorithm of Bruno *et al.* [BJS80]. □

Corollary 4.3. *$L2|prec, r_j, p_{ij}=1|L_{\max}$ can be solved in polynomial time $O(n^2)$.*

Proof. By Corollary 4.2, $C2|prec, r_j, \ell = k, p_{ij}=1|L_{\max}$ can be solved in polynomial time $O(n^2)$, in particular, for $k=3$. □

4.3. NP-hard problems. This section is devoted to NP-hardness proofs of nonperiodic cycle-shop scheduling problems. In the following corollary from Theorem 3.2 we take into account the trivial reduction $1|\beta_{3-4}, p_j=1|\gamma \propto P2|\beta_{3-4}, p_j=1|\gamma$ in the case $\beta_3 \neq \circ$ by introducing a second machine and a separated long enough chain of unit-time jobs with zero weights, zero release dates and an infinite due dates.

Corollary 4.4. *The following reductions prove that the cycle-shop problems at the right-hand side are NP-hard because the related problems on identical parallel machines at the left-hand side are proved to be so. Strongly NP-hard problems are marked by $*$.*

[BK99]	$P\|pmtn, p_j = p\|\sum w_jU_j$	$\propto$	$C\|p_{ij} = 1\|\sum w_jU_j$
[K99B]*	$P\|pmtn, r_j, p_j = m\|\sum U_j$	$\propto$	$C\|r_j, p_{ij} = 1\|\sum U_j$
[LY90]*	$P\|pmtn, r_j, p_j = p\|\sum w_jC_j$	$\propto$	$C\|r_j, p_{ij} = 1\|\sum w_jC_j$
[L]*	$P\|intree, r_j, p_j = 1\|\sum C_j$	$\propto$	$C\|no\,wait, intree, r_j, p_{ij} = 1\|\sum C_j$
[BGJ77]*	$P\|intree, r_j, p_j = 1\|C_{\max}$	$\propto$	$C\|no\,wait, intree, r_j, p_{ij} = 1\|C_{\max}$
[BGJ77]*	$P\|outtree, p_j = 1\|L_{\max}$	$\propto$	$C\|no\,wait, outtree, p_{ij} = 1\|L_{\max}$
[U75]*	$P\|prec, p_j = 1\|C_{\max}$	$\propto$	$C\|no\,wait, prec, p_{ij} = 1\|C_{\max}$
[LRK78]*	$P\|prec, p_j = 1\|\sum C_j$	$\propto$	$C\|no\,wait, prec, p_{ij} = 1\|\sum C_j$
[T98A]*	$P2\|chains, p_j = 1\|\sum w_jC_j$	$\propto$	$C2\|no\,wait, chains, p_{ij} = 1\|\sum w_jC_j$
[LY90A]*	$1\|chains, p_j = 1\|\sum T_j$	$\propto$	$C2\|no\,wait, chains, p_{ij} = 1\|\sum T_j$
[LRK80]*	$1\|chains, p_j = 1\|\sum U_j$	$\propto$	$C2\|no\,wait, chains, p_{ij} = 1\|\sum U_j$

The list of similar reductions could be continued. However, they lead to either unit-time cycle-shop problems which are open along with the related equal-time problems on identical parallel machines or NP-hard unit-time cycle-shop precedence-constrained problems without the no-wait constraint. NP-hardness of the latter trivially follows from the NP-hardness of their unit-time flow-shop counterparts (see the list in [BK]). For example, the strong NP-hardness of $C|prec, p_{ij} = 1|C_{\max}$ and $C2|chains, p_{ij} = 1|\sum U_j$ follows from the strong NP-hardness of $F|prec, p_{ij} = 1|C_{\max}$ [LVW84, T98A] and $F2|chains, p_{ij} = 1|\sum U_j$ [BK98], respectively. Notice that $F|r_j, p_{ij} = 1|\sum w_jC_j$ can be solved in polynomial time because even $F|r_j, p_{ij} = 1|\sum w_jU_j$ and $F|r_j, p_{ij} = 1|\sum w_jT_j$ are so [LLRKS93].

It is important to observe that the reductions in Corollary 4.5 prove the NP-hardness of no-wait unit-time problems in cycle shops of rank m and multiplicity at least m. NP-hard no-wait unit-time flow-shop counterparts could also give NP-hardness proofs but only in the case of rank 0 and multiplicity 1, *i.e.*, only for flow shops. All the no-wait unit-time flow-shop counterparts are equivalent to their relaxations without the no-wait constraint [T98A] and are strongly NP-hard except $F|intree, r_j, p_{ij} = 1|\sum C_j$ and $F|prec, p_{ij} = 1|\sum C_j$ which remain open as well as $C|intree, r_j, p_{ij} = 1|\sum C_j$ and $C|prec, p_{ij} = 1|\sum C_j$.

5. Periodic cycle shops

In this section we show that $C|prec, r_j|P$ and $C|no\,wait, n = 1, p_i = 1|P$ can be solved in polynomial time and that $1C|no\,wait, n = 1, p_i = q_{\mu_i}/q|P$ can be solved in pseudopolynomial time. Then we strengthen the NP-hardness result of Roundy for $\mathcal{C}|n = 1|C_{\max}$ [R92] showing that even $R\mathcal{F}|n = 1, p_o = q_{\mu_o}, p_e = 1|C_{\max}$ and, hence by Lemma 2.1, $1\mathcal{C}|n = 1, p_i = 1|C_{\max}$ are strongly NP-hard. We also consider absolute error bound approximations for the latter problem.

5.1. Minimum-period cycle shops and even job shops are easy. There is a simple polynomial time algorithm for $J|prec, r_j|P$ with time requirement proportional to the total number of operations in jobs. Remind that the flow time is not restricted in this problem, and the algorithm freely spends it. The first three steps solve the problem disregarding the release dates. The fourth step turns the schedule into one that satisfies given release dates.

- Construct the precedence relation R for the total set of operations in the jobs and find a linear order including R, *i.e.*, a list L of operations consistent with R.
- Set $P = \max_{1\le\imath\le m}\{\sum_{\imath=\mu_{ij}} p_{ij}\}$, *i.e.*, P is the maximum total processing time on the machines, and divide a time interval of length PT, where $T = \sum_{j=1}^{n} m_j$, *i.e.*, T is the total number of operations in jobs, into T subintervals of length P. Each subinterval will include only one operation.
- Let the first $j-1$ operations of the list L are already included in the schedule, and let they occupy the total processing times $T_\imath$ on the machines $M_\imath$, $\imath = 1, 2, \ldots, m$. Let the jth operation in L require the machine M_k. Then assign its start time on M_k to be T_k as a local time inside the jth subinterval.
- Assuming that the local start times of operations in the subintervals are fixed, move the subintervals apart so that the release dates of the jobs are satisfied and the distance between any two adjacent subintervals are divisible by P.

The algorithm obviously finds a minimum-period schedule because P is the maximum total processing time on the machines, a lower bound for periods. Special cases of the problem that cannot be solved in polynomial time by this algorithm is $J2|prec, r_j, p_{ij} = 1|P$ and even $J2|p_{ij} = 1|P$ because the size of these problems includes the number of operations in jobs under logarithms, *i.e.*, the algorithm runs only in pseudopolymonial time. We suggest that these special cases are also well solvable but they require a special consideration like $J2|p_{ij} = 1|C_{\max}$ and $J2|r_j, p_{ij} = 1|C_{\max}$ do [T97].

5.2. Trivially solvable minimum-period no-wait cycle shops. The no-wait constraint in periodic cycle shops often makes the related problems easy. Here we consider $1C|no\,wait, n = 1, p_i = q_{\mu_i}/q|P$ and $C|no\,wait, n = 1, p_i = 1|P$. Their sizes are $\mathfrak{m}\log m + \sum_{k=1}^{m}\log q_k$ and $\ell\log m$, respectively. Let us take the former problem. Since all processing times are divisible by q we will consider an equivalent form of the problem, where $q = 1$, dividing all processing times by q. Without loss of generality we assume that the multimachine in this problem is M_1. Thus, all operations on M_1 are of unit processing time, and the condition that start times on M_1 should be different modulo P becomes necessary for any feasible schedule. Let x_i be the the start time of ith operation on M_1, and let $a_i - 1$ be the total processing time of all operations between ith and $i+1$st operations on M_1 if $i > 0$ or all operations before the first operation on M_1 if $i = 0$ or all operations after the last operation on M_1 if $i = \mathfrak{m}$. Therefore, if $x_1 = 0$, then $x_i = \sum_{j=1}^{i-1} a_j$ for $i = 2, 3, \ldots, \mathfrak{m}$. Thus, the problem is to find minimum P such that $x_1, x_2, \ldots, x_{\mathfrak{m}}$ are all different modulo P. It is clear that for a fixed P this condition can be checked in time $O(\mathfrak{m})$. Remind that $P \ge Q = \max_{1\le k\le m}\mathfrak{m}_k q_k$ and note that, due to the no-wait requirement, the flow time is fixed and equal $C_{\max} = -1 + \sum_{j=0}^{\mathfrak{m}} a_j = \sum_{k=1}^{m}\mathfrak{m}_k q_k$. Thus, to solve the problem we need to check whether $x_1, x_2, \ldots, x_{\mathfrak{m}}$ are different modulo P for each period in the interval $[Q, C_{\max}-1]$. This takes time $O\left(\mathfrak{m}\left[\sum_{k=1}^{m}\mathfrak{m}_k q_k - \max_{1\le k\le m}\mathfrak{m}_k q_k\right]\right)$ which is $O\left(\mathfrak{m}^2 m q_{\max}\right)$.

Similar considerations lead to a polynomial-time algorithm for $C|no\,wait, n = 1, p_i = 1|P$. Let $x_1 < x_2 < \cdots < x_{\mathfrak{t}}$ be start times of all operations $O_{s_1}, O_{s_1}, \ldots, O_{s_{\mathfrak{t}}}$ in one job on all multimachines. Thus, $\mathfrak{t}$ is the total multiplicity of multimachines. Let $a_i - 1$ be the total processing time of all operations between O_{s_i} and $O_{s_{i+1}}$ if $i > 0$ or all operations before O_{s_1}

if $i = 0$ or all operations after $O_{s_\mathfrak{t}}$ if $i = \mathfrak{t}$. Define I_k to be the set of indices of operations on the kth multimachine, $k = 1, 2, \ldots, \mathfrak{r}$. So, $I_1, I_2, \ldots, I_\mathfrak{r}$ is a partition of $\{1, 2, \ldots, \mathfrak{t}\}$. Set $x_1 = 0$. Then $x_i = \sum_{j=1}^{i-1} a_j$ for $i = 2, 3, \ldots, \mathfrak{t}$ and the problem is to find minimum P such that start times x_i, $i \in I_k$, are different modulo P for each $k = 1, 2, \ldots, \mathfrak{r}$. For a fixed P this can be checked in time $O(\mathfrak{t})$. Since we consider the case with unit processing times, $C_{\max} = \ell$ and $Q = \mathfrak{m}$. Hence, the total time complexity is $O(\mathfrak{t}[\ell - \mathfrak{m}]) = O(\ell^2 - \mathfrak{m}\ell)$.

5.3. NP-hardness of minimum-flow cycle shops of rank one. In this subsection we consider $R\mathcal{F}|n = 1, p_o = q_{\mu_o}, p_e = 1|C_{\max}$ in an equivalent form as the one-machine problem of finding a minimum-length one-machine schedule for a chain $\mathfrak{J} = \mathrm{J}_1 \prec \mathrm{J}_2 \prec \cdots \prec \mathrm{J}_{s-1} \prec \mathrm{J}_s$ of nonpreemptive jobs with processing times $a_1, a_2, \ldots, a_{s-1}, 1$, respectively, such that the start times of the jobs are different modulo P for a given P. It is clear that $s = \frac{\ell-1}{2}$ and $a_i = 1 + p_{2i+1}$, where $i = 1, 2, \ldots, s-1$.

We use the denotation $\mathfrak{J}_1 \prec \mathfrak{J}_2$ for any two subsets $\mathfrak{J}_1$ and $\mathfrak{J}_2$ of $\mathfrak{J}$ such that $\mathrm{J}_1 \in \mathfrak{J}_1$ & $\mathrm{J}_2 \in \mathfrak{J}_2$ implies $\mathrm{J}_1 \prec \mathrm{J}_2$.

Theorem 5.1. *$R\mathcal{F}|n = 1, p_o = q_{\mu_o}, p_e = 1|C_{\max}$ is strongly NP-hard.*

Proof. We reduce 3-SAT [GJ79] to our problem. Let $\{C_1, \ldots, C_\mathfrak{m}\}$ be a set of three-literal clauses with variables $\{v_1, \ldots, v_\mathfrak{n}\}$. We construct the chain $\mathfrak{J}$ as the union of four subchains $\mathrm{A} \prec \mathrm{B} \prec \mathrm{C} \prec \mathrm{D}$ and show that a truth assignment for the variables exists if and only if there exists a schedule of length $\le T = 1 + \sum_{i=1}^{s-1} a_i + 4\mathfrak{m} + 2\mathfrak{n}$.

Let $s(\mathrm{J})$ denote the start time of the job J in a feasible schedule for $\mathfrak{J}$, and let $\mathrm{r}(\mathrm{J})$ denote the corresponding residue modulo P. Note, that we will not define P and the job processing times explicitly since we only need some specific relations between the residues of the jobs processing times. Any large enough $P > a_i$ for all i will fit. And it will be easy though tedious to define the jobs processing times accordingly. Let $e(\mathrm{J})$ denote the earliest possible start time of the job $\mathrm{J} = \mathrm{J}_i \in \mathfrak{J}$, *i.e.*, $e(\mathrm{J}) = \sum_{j=1}^{i-1} a_j$. Every time unit in a schedule between the start time of the first job and the completion time of the last job we call a *shift* if it is not used for processing a job. Thus, if $s(\mathrm{J}_{i+1}) - s(\mathrm{J}_i) = a_i + b$ then there are b shifts between J_i and J_{i+1}. For our construction we need the property that the jobs in A have to be scheduled at their earliest possible starting time in every schedule of length $\le T$. Therefore we define sets B and D such that at least $4\mathfrak{m} + 2\mathfrak{n}$ shifts occur after the first job of B in every feasible schedule. Let B and D be chains containing $4\mathfrak{m} + 2\mathfrak{n}$ jobs each such that for each $\mathrm{r} \in [1, 4\mathfrak{m} + 2\mathfrak{n}]$ there exists a job $\mathrm{J} \in \mathrm{B}$ (respectively $\mathrm{J} \in \mathrm{D}$) with $\mathrm{r} = e(\mathrm{J}) \bmod P$.

Claim 5.1. *A schedule of length $\le T$ implies that jobs $\mathrm{J} \in \mathrm{A}$ are scheduled at times $e(\mathrm{J})$.*

Proof of Claim 5.1. The claim easily follows from the definition of the sets B and D. Clearly, there must be at least $4\mathfrak{m} + 2\mathfrak{n}$ shifts between the first job in B and the last job in D. Otherwise there are jobs in $\mathrm{B} \cup \mathrm{D}$ with equal residues. Since the required total number of shifts is at most $4\mathfrak{n} + 2\mathfrak{m}$, there can be no shift before the first job in B. □

Claim 5.1 shows that $\mathrm{J} \in \mathrm{A}$ implies $\mathrm{r}(\mathrm{J}) = e(\mathrm{J}) \bmod P$ in any schedule of length $\le T$. Hence jobs in $\mathrm{J} \in \mathrm{A}$ can be used to "occupy" some residues so that they can not be used

by other jobs. Besides, Claim 5.1 allows to fix desired start times $s \in [e(\mathrm{J}), e(\mathrm{J}) + 4n + 2m$ of jobs J in C in any schedule of length $\leq T$. This can be done by occupying all residues $\mathrm{r} \in [e(\mathrm{J}) \bmod P, e(\mathrm{J}) + 4\mathfrak{n} + 2\mathfrak{m} \bmod P]$ with $\mathrm{r} \neq s \bmod P$. The number of jobs and the job processing times in the chain A will easily follow from these properties. Now define jobs $\mathrm{J}_{\mathrm{C},1} \prec \ldots \prec \mathrm{J}_{\mathrm{C},2\mathfrak{n}+\mathfrak{m}+1}$ in the chain C such that

(∗) for each $i \in [1, 2\mathfrak{n} + \mathfrak{m}]$ there are exactly 2 shifts between start times of $\mathrm{J}_{\mathrm{C},i}$ and $\mathrm{J}_{\mathrm{C},i+1}$.

For each job J (which has to be defined) with $\mathrm{J}_{\mathrm{C},i} \prec \mathrm{J} \prec \mathrm{J}_{\mathrm{C},i+1}$, $i \in [1, 2\mathfrak{n} + \mathfrak{m}]$, let $t(\mathrm{J})$ be the start time in a schedule of length $\leq T$ when there is no shift between the start times of $\mathrm{J}_{\mathrm{C},i}$ and J (and there are $2i$ shifts before $\mathrm{J}_{\mathrm{C},i}$). To obtain a truth assignment define for each variable v_j, $j \in [1, n]$ two subchains V_j and $\bar{\mathrm{V}}_j$ of jobs in C with $\mathrm{J}_{\mathrm{C},2j-1} \prec \mathrm{V}_j \prec \mathrm{J}_{\mathrm{C},2j} \prec \bar{\mathrm{V}}_j \prec \mathrm{J}_{\mathrm{C},2j+1}$ such that the following claim holds.

Claim 5.2. *A schedule of length $\leq T$ and $j \in [1, \mathfrak{n}]$ imply that either every job $\mathrm{J} \in \mathrm{V}_j$ is scheduled at time $t(\mathrm{J})$ and every job $\mathrm{J}' \in \bar{\mathrm{V}}_j$ is scheduled at time $t(\mathrm{J}) + 2$ or every job $\mathrm{J} \in \mathrm{V}_j$ is scheduled at time $t(\mathrm{J}) + 2$ and every job $\mathrm{J}' \in \bar{\mathrm{V}}_j$ is scheduled at time $t(\mathrm{J})$.*

Proof of Claim 5.2. The jobs J will be defined such that for an optimal schedule one of the following two cases holds.

(1) The two shifts between $\mathrm{J}_{\mathrm{C},2j-1}$ and $\mathrm{J}_{\mathrm{C},2j}$ are between the start times of $\mathrm{J}_{\mathrm{C},2j-1}$ and the first jobs in V_j, or the two shifts between $\mathrm{J}_{\mathrm{C},2j}$ and $\mathrm{J}_{\mathrm{C},2j+1}$ are between the start times of the last job in $\bar{\mathrm{V}}_j$ and $\mathrm{J}_{\mathrm{C},2j+1}$.
(2) The two shifts between $\mathrm{J}_{\mathrm{C},2j-1}$ and $\mathrm{J}_{\mathrm{C},2j}$ are between the start times of the last job in V_j and $\mathrm{J}_{\mathrm{C},2j}$, or the two shifts between $\mathrm{J}_{\mathrm{C},2j}$ and $\mathrm{J}_{\mathrm{C},2j+1}$ are between the start times of $\mathrm{J}_{\mathrm{C},2j}$ and the first job in $\bar{\mathrm{V}}_j$.

These conditions will be used to define a truth assignment for variables: (1) to make v_j truth and (2) to make v_j false.

Define jobs $\mathrm{J}_{f,j}$ and $\mathrm{J}_{l,j}$ to be the first and the last job in V_j, respectively. The other jobs in V_j will be defined later. Analogously, define $\bar{\mathrm{J}}_{f,j}$ and $\bar{\mathrm{J}}_{l,j}$ to be the first and the last job in $\bar{\mathrm{V}}_j$. The jobs in A will be defined such that $\mathrm{r} = t(\mathrm{J}_{f,j}) \bmod P = t(\bar{\mathrm{J}}_{f,j}) \bmod P$ and such that $\mathrm{r} + 1$ is occupied by a job from A (and all the other jobs should be defined such that they do not use the residues $\mathrm{r}, \mathrm{r} + 1$, $\mathrm{r} + 2$). This property and (∗) imply that $\mathrm{J}_{f,j}$ has to be scheduled at time $t(\mathrm{J}_{f,j})$, and $\bar{\mathrm{J}}_{f,j}$ has to be scheduled at time $t(\bar{\mathrm{J}}_{f,j}) + 2$, or vice versa (see Figure 5.1 for the first case). Analogously, define jobs in A such that $\bar{\mathrm{r}} = t(\mathrm{J}_{l,j}) \bmod P = t(\mathrm{J}_{l,j}) \bmod P$ and $\bar{\mathrm{r}} + 1$ is occupied by a job from A (and all the other jobs should be defined such they do not use the residues $\bar{\mathrm{r}}, \bar{\mathrm{r}} + 1$, $\bar{\mathrm{r}} + 2$). This property and (∗) imply that $\mathrm{J}_{l,j}$ has to be scheduled at time $t(\mathrm{J}_{l,j})$, and $\bar{\mathrm{J}}_{l,j}$ has to be scheduled at time $t(\bar{\mathrm{J}}_{l,j}) + 2$, or vice versa (see Figure 5.1 for the first case). □

Now consider a clause $C_j = \{u_h, u_k, u_l\}$, $h < k < l$, where $u_h = v_h$ if v_h occurs positive in C_j and $u_h = \bar{v}_h$ if v_h occurs negated. Define u_k and u_l analogously. Define jobs $\mathrm{J}_{j,h}, \mathrm{J}_{j,k}, \mathrm{J}_{j,l}$

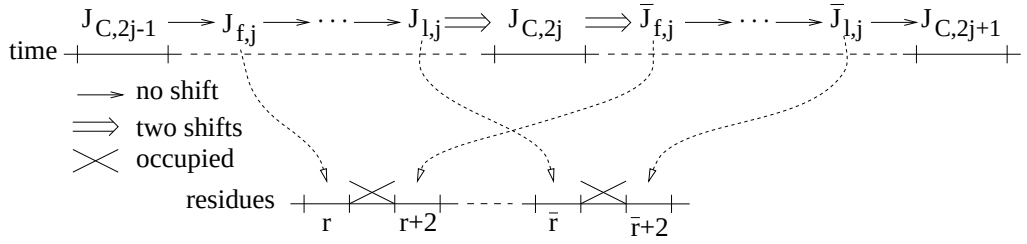

FIGURE 5.1. Shifts between the jobs

and $J_{j,*}$ as follows. If $u_h = v_h$ then $J_{j,h} \in V_h$, otherwise $J_{j,h} \in \bar{V}_h$. Analogously for u_k. But the third literal u_l in the clause should be handled differently.

Define $J_{j,l} \in \bar{V}_l$ if $u_l = v_l$, or $J_{j,h} \in V_l$ otherwise. Define $J_{j,*}$ such that $J_{C,2n+j} \prec J_{j,*} \prec J_{C,2n+j+1}$ and $J_{j,*}$ is the only job between $J_{C,2n+j}$ and $J_{C,2n+j+1}$. Property $(*)$ implies that there are three possible starting times for $J_{j,*}$. Now define processing times of the jobs in $\mathfrak{J}$ such that the following property holds (see Figure 5.2).

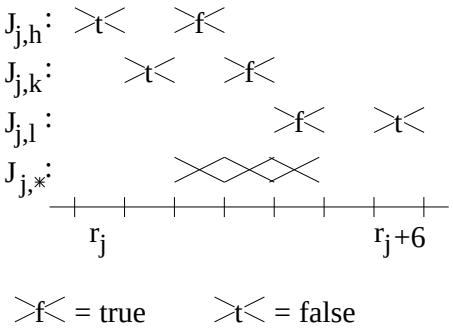

FIGURE 5.2. Residues of possible start times

$(**)$ For each clause $C_j = \{u_h, u_k, u_l\}$, $h < k < l$, $j \in [1, \mathfrak{m}]$ there exists $r_j \in [0, P-1]$ such that $r(J_{j,h}) = r_j \vee r_j+2$, $r(J_{j,k}) = r_j+1 \vee r_j+3$, $r(J_{j,l}) = r_j+4 \vee r_j+6$, $r(J_{j,*}) = r_j + 2 \vee r_j + 3 \vee r_j + 4$ and no other job uses one of the residues $r_j, r_j+1, \ldots, r_j+6$ in any schedule of length $\leq T$.

Observe that for each clause $C_j = \{u_h, u_k, u_l\}$, $h < k < l$, $j \in [1, \mathfrak{m}]$ the variable v_h,v_k or v_l is false if $r(J_{j,h}) = r_j + 2$, $r(J_{j,k}) = r_j + 3$ or $r(J_{j,l}) = r_j + 4$, respectively. Since $r(J_{j,*}) = r_j + 2 \vee r_j + 3 \vee r_j + 4$ at least one variable has to be true. Thus, a schedule of length $\leq T$ exists only if there exists a truth assignment for $\{C_1, \ldots, C_{\mathfrak{m}}\}$. The inverse statement is obvious. □

Corollary 5.1. $1\mathcal{C}|n = 1, p_i = 1|C_{\max}$ *is strongly NP-hard.*

Proof. Follows from Theorem 5.1 and Lemma 2.1. □

Corollary 5.2. *$1\mathcal{C}|n=1,p_i=1|\sum C_{ij}$ is strongly NP-hard.*

Proof. The proof is a slight extension the proof of Theorem 5.1. Let us add one job J_{s+1} with $a_s > T^2$ at the end of the chain $\mathfrak{J}$, where T was defined in the proof of Theorem 5.1. It is not a problem to enforce starting this job at time $T+1$. So, if $\mathrm{J}_1, \mathrm{J}_2, \ldots, \mathrm{J}_s$ start at times $\le T$ and J_{s+1} start at time $T+1$, then $\sum C_{ij} \le T^2 + \sum_{i=1}^{a_s}(T+i)$. On the other hand, if J_{s+1} starts at time $\ge T+2$ then $\sum C_{ij} \ge \sum_{i=1}^{a_s}(T+1+i) \ge a_s + \sum_{i=1}^{a_s}(T+i) > T^2 + \sum_{i=1}^{a_s}(T+i)$. Thus, a schedule with $\sum C_{ij} \le T^2 + \sum_{i=1}^{a_s}(T+i)$ for $\mathrm{J}_1, \mathrm{J}_2, \ldots, \mathrm{J}_s, \mathrm{J}_{s+1}$ exists if and only if a schedule with $C_{max} \le T$ for $\mathrm{J}_1, \mathrm{J}_2, \ldots, \mathrm{J}_s$ exists. □

5.4. Simplest approximation for minimum-flow-time cycle shops. In this subsection we consider a polynomial-time approximation for $1\mathcal{C}|n=1,p_i=1|C_{\max}$ and show how it can be extended to $\mathcal{C}|n=1,p_i=1|C_{\max}$. We first consider an $O\left(\mathfrak{m}^2\right)$ algorithm for the former problem obtaining the flow time $C^1_{\max}$ with

$$C^1_{\max} - C^*_{\max} \le \tfrac{\mathfrak{m}^2-\mathfrak{m}}{2},$$

where $C^*_{\max}$ denotes the minimum flow time. Let x_i be the start time of ith operation on M_1, and let $a_i - 1$ be the total processing time of all operations between ith and $i+1$st operations on M_1 if $i > 0$ or all operations before the first operation on M_1 if $i = 0$ or all operations after the last operation on M_1 if $i = \mathfrak{m}$. Therefore, $x_i \ge x_{i-1} + a_{i-1}$ for $i = 2, 3, \ldots, \mathfrak{m}$, where we set $x_1 = 0$. Thus, the problem is to find integers $x_2, \ldots, x_{\mathfrak{m}}$ with minimum $x_{\mathfrak{m}}$ such that $x_1, x_2, \ldots, x_{\mathfrak{m}}$ are all different modulo P.

The algorithm deals with the current set R of residuals modulo P engaged by the start times. At the first step set $x_1 = 0$, $R = \{0\}$. After assigning all the start times, R will contain $\mathfrak{m}$ residuals modulo P. At the ith step, $i > 1$, set x_i to be the smallest integer such that $x_i \bmod P$ is not in R and $x_i \ge x_{i-1} + a_{i-1}$, and put $x_i \bmod P$ in R. In the worst case, R is an integer interval with $i-1$ integers, so the difference $x_i - x_{i-1} - a_{i-1}$ is at most $i-1$. Summing up from $i = 2$ to $i = \mathfrak{m}$ we get that the total loss is at most $1 + 2 + \cdots + \mathfrak{m} - 1 = \frac{\mathfrak{m}^2-\mathfrak{m}}{2}$, and $x_{\mathfrak{m}}$ is going to be at most $\frac{\mathfrak{m}^2-\mathfrak{m}}{2} + \sum_{i=1}^{\mathfrak{m}-1} a_i$. On the other hand, an evident lower bound of $x_{\mathfrak{m}}$ is $\sum_{i=1}^{\mathfrak{m}-1} a_i$. So, we lose $\frac{\mathfrak{m}^2-\mathfrak{m}}{2}$ in the worst case. Obviously, the time comlexity is $O\left(\mathfrak{m}^2\right)$.

Introducing the set of residuals for each multimachine, analogous reasonings lead to two a similar approximation algorithm for $\mathcal{C}|n=1,p_i=1|C_{\max}$ with loss at most $\sum_{k=1}^{\mathfrak{r}} \frac{\mathfrak{m}_k^2-\mathfrak{m}_k}{2}$ and time complexity $O\left(\sum_{k=1}^{\mathfrak{r}} \mathfrak{m}_k^2\right)$, where $\mathfrak{m}_k$ is the multiplicity of the kth multimachine. Such an extension is exactly the NWF algorithm.

5.5. Advanced approximation. Below we describe an $O(\mathfrak{m}^2)$ algorithm for $1\mathcal{C}|n = 1, p_i = 1|C_{\max}$ that obtains a flow time $C^2_{\max}$ with

$$C^2_{\max} - C^*_{\max} \le \tfrac{\mathfrak{m}^2-\mathfrak{m}^{3/2}/14}{2}.$$

The algorithm schedules the operations $O_1, \ldots, O_{\mathfrak{m}}$ in this order. Let x_i denote the scheduled time for O_i, $i \in [1:\mathfrak{m}]$. The algorithm begins with scheduling O_1 at time $x_1 = 0$.

After $O_1, \ldots, O_{i-1}$ have been scheduled, $i \in [2, \mathfrak{m}]$ the earliest possible starting time for O_i is $x_{i-1} + a_{i-1}$. The number of time steps operation O_i is scheduled later than its earliest possible starting time is called the number of *shifts* made by the algorithm when scheduling O_i. Let $r_h = x_h \bmod P$, $h \in [1, h-1]$. A residue $t \bmod P$ is *occupied* by operation O_h if $x_h \bmod P = t \bmod P$. In the following an *interval* is a set of residues of the form $r, r+1, \ldots, r+k$ modulo P which are all occupied and where $r-1 \bmod P$ and $r+k+1 \bmod P$ are not occupied. Let $\aleph_r$ be the number of intervals minus one after scheduling O_r. In order to prevent that too many shifts are necessary when schduling operation O_i the algorithm will possibly reschedule a subset $\mathcal{S}$ of the operations $O_1, \ldots, O_{i-1}$. $\mathcal{S}$ will always be of the form $\mathcal{S} = \{O_{z+1}, \ldots, O_{i-1}\}$ for a $z \in [1, i-1]$ satisfying the following property.

(∗) There exists a (possibly empty) proper suffix of length $\leq z$ of every interval such that these suffixes contain exactly the residues $r_{z+1}, r_{z+2}, \ldots, r_{i-1}$.

Observe that if O_i is shifted by at least one time step then it is possible to reschedule all operations in $\mathcal{S}$ by one time step later.

Example illustrating the property (∗). Assume operations $O_1, \ldots, O_{11}$ have been scheduled and $z = 7$ as depicted in the upper part of Figure 5.3. Observe that the residues of operations O_j with $j > z = 7$ form suffixes of length $\leq 7 = z$ of the intervals (i.e. (∗) holds). Assume r_{12} is occupied and consider the following two cases.

(1) Residue r_{12} is occupied by an operation O_j with $j > 7 = z$ (see upper part of Figure 5.3). Then O_{12} is shifted to the first free residue behind the interval containing r_{12} (see lower part of Figure 5.3). Since the suffixes containing operations O_j with $j > z$ have length $\leq z$ at most $z = 7$ shifts are made.

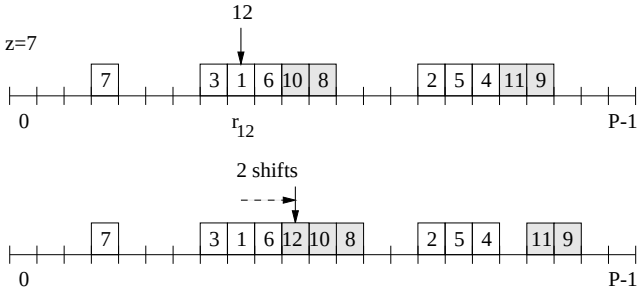

FIGURE 5.3. Residue occupied by operation O_j with $j \leq z$

(2) Residue r_{12} is occupied by an operation O_j with $j \leq 7 = z$ (see upper part of Figure 5.4). Then O_{12} is shifted until the corresponding residue r is free or is occupied by an operation O_h with $h > z$. In the last case every operation O_h with $z = 7 < h < 12 = i$ is rescheduled one time step later. Observe that O_{12} was shifted by at most z time steps and the rescheduling is possible since operation O_{12} was shifted by at least one time step.

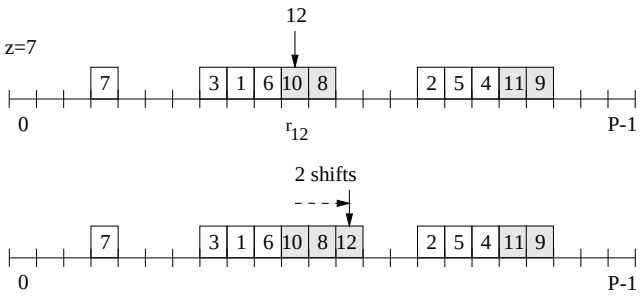

FIGURE 5.4. Residue occupied by operation O_j with $j > z$

The algorithm starts on scheduling O_1 at time 0 and setting $z = 1$ (Then $\mathcal{S} = \emptyset$ and $(*)$ holds). Let $O_1, \ldots, O_{i-1}$ be already scheduled. Then the algorithm proceedes scheduling O_i in accordance with the following steps.

Step 1. Let t be the earliest possible start time of O_{i-1}, *i.e.*, $t = x_{i-1} + a_{i-1}$. Then we have the following situations.

(a) If $t \bmod P$ is not occupied, then schedule O_i at time $x_i = t$.
(b) if $t \bmod P$ is occupied by some O_j with $j \leq z$, then let $b \geq 1$ be the smallest integer such that $t + b \bmod P$ is not occupied or is occupied by some operation O_l with $l > z$. If $t + b \bmod P$ is occupied by an operation O_l with $l > z$, then reschedule each of the operations $O_{z+1}, \ldots, O_{i-1}$ one time step later, *i.e.*, $x_h = x_h + 1$ for $h \in [z+1, i-1]$. If $t+b \bmod P$ is not occupied, then schedule O_i at time $x_i = t+b$.
(c) If $t \bmod P$ is occupied by some operation O_j with $j > z$, then schedule O_i at time $x_i = t+b$, where $b \geq 1$ is the smallest integer such that $t+b \bmod P$ is not occupied.

Step 2. If the condition $(*)$ does not hold, *i.e.*, there exists an operation O_j with $j > z$ such that $x_j + 1 \bmod P$ is occupied by an operation O_h with $h \leq z$ or there exists a suffix of an interval of length $> z$ containing only operations O_j with $j > z$ or a new interval was created, *i.e.*, there exists an operation O_j with $j > z$ such that $x_j - 1 \bmod P$ is not occupied) then set $z = i$.

To see that the algorithm works correctly assume $O_1, \ldots, O_{i-1}$ have been scheduled and $(*)$ holds for $\mathcal{S}$ defined by $z \in [1, i-1]$. Now, it is not hard to show that in Step (1) of the algorithm O_i is scheduled correctly and Step 2 guaranties that $(*)$ holds for $\mathcal{S}$ defined by the (new) value of z. To determine the flow time we make an amortized analysis. For this a counting scheme for the number of shifts made by the algorithm and a potential function Φ will be defined such that the total number of counts for the operations $O_1, \ldots, O_{\mathfrak{m}}$ does not underestimate the total number of shifts made by the algorithm plus the increase of Φ. Φ will be defined such that it becomes not negative (in fact after scheduling an operation O_i, $i \in [1, \mathfrak{m}]$ the value of Φ is at least $\frac{1}{7} i \aleph_i$. After the definition of Φ and the counting scheme we show that the desired properties hold. To define Φ and the counting scheme consider a run of the algorithm and assume a new value $z = s$ was set when scheduling operation O_s and the former value $z = r$ has been set when scheduling operation O_r, $r < s$. Consider the following three cases.

Case 1. $z = s$ was defined because a new interval was created when scheduling operation O_s, *i.e.*, there exists an operation O_j with $r < j < s$ such that $x_j - 1 \bmod P$ is not occupied. There are two possibilites: a new interval is created because O_s is not shifted and $x_s - 1 \bmod P$ is not occupied or a new interval is created during the rescheduling of operation in Step 1(b). In the following we consider only the second possibility because the first one can be handled similarly. Hence, assume there are operations $O_{i_1}, \ldots, O_{i_k}$ with $r < i_1 < \ldots < i_k < s$ such that each x_{i_j}, $j \in [1,k]$ is the start of a new interval that was created in Step 1(b) when scheduling operation O_s. Then $\Phi = \Phi + \frac{1}{7}\aleph_r$ for every operation O_h, $h \in [r+1, s]$, $h \neq i_j$, $j \in [1,k]$, and $\Phi = \Phi + \frac{1}{7}\aleph_r + \frac{1}{7}s$ for every operation O_{i_j}, $j \in [1,k]$. We count $\frac{6}{7}i_j$ shifts for operation O_{i_j}, $j \in [1,k]$, $\frac{6}{7}s$ shifts for operation O_s, $r + \frac{1}{7}(h-r)$ shifts for every operation O_h with $h \in [r+1, 2r - \frac{1}{7}r]$, $h \neq i_j$ for $j \in [1,k]$, and $r + \frac{1}{7}h$ shifts for every operation O_h with $h \in [2r - \frac{1}{7}r + 1, s-1]$, $h \neq i_j$ for $j \in [1,k]$.

Case 2. $z = s$ was defined because there exists an operation O_j with $j > r$ such that $x_j + 1 \bmod P$ is occupied by an operation O_h with $h \leq r$ and Case 1 does not hold. Then $\Phi = \Phi + \frac{1}{7}\aleph_r - \frac{1}{7}s$ when scheduling O_s, and $\Phi = \Phi + \frac{1}{7}\aleph_r$ when scheduling O_h, $h \in [r+1, s-1]$. We count $\frac{6}{7}s$ for scheduling operation O_s, r shifts for scheduling O_h, $h \in [r+1, \min\{2r - \frac{1}{7}r, s-1\}]$, and $r + \frac{1}{7}r$ shifts for scheduling O_h, $h \in [\min\{2r - \frac{1}{7}r + 1, s-1\}, s-1]$.

Case 3. $z = s$ was set because there exists a suffix of an interval of length $> r$ containing only operations O_j with $r < j$ and none of Cases 1 and 2 holds. Then $\Phi = \Phi + \frac{1}{7}\aleph_r$ for every operation O_h, $h \in [r+1, s]$. We count r for every operation O_h, $h \in [r+1, 2r - \frac{1}{7}r]$, and count $r + \frac{1}{7}\aleph_r$ for every operation O_h, $h \in [2r - \frac{1}{7}r + 1, s]$.

Lemma 5.1. *The counting scheme as defined above does not underestimate the total number of shifts plus the increase of* Φ.

Proof. Let us consider the same cases as above.

Case 1. Observe that in this case, before the rescheduling in Step 1(b) is done, $r_{i_1}, \ldots, r_{i_k}$ and r_s are elements of pairwise different intervals. Hence, altogether at most r shifts were used for scheduling the operations $O_{i_1}, \ldots, O_{i_k}$ and O_s. Of the at most r many shifts made when scheduling operations $O_{i_1}, \ldots, O_{i_k}$ and O_s we count $\frac{4}{7}r$ shifts for O_s and $\frac{3r}{7k}$ shifts for O_{i_j}, $j \in [1,k]$. For every other operation O_h , $h \in [r+1, s-1]$, $h \neq i_j$ for $j \in [1,k]$ at most $r - \min\{\frac{1}{7}r, \aleph_r\}$ shifts are made if $h \leq 2r - \frac{1}{7}r$ and at most r shifts if $h > 2r - \frac{1}{7}r$. We show that the remaining counts are more than the increase of Φ. First we add $\frac{1}{7}\aleph_r$ to Φ for each operation O_h, $h \in [r+1, s]$. After this we have the following remaining counts: For every operation O_h , $h \in [r+1, s-1]$, $h \neq i_j$ for $j \in [1,k]$ a remaining count of $\frac{1}{7}(h-r)$ and for every operation O_{i_j}, $j \in [1,k]$ a remaining count of $\frac{2}{7}i_j$. it is not hard to show that the remaing counts are enough for the increase of Φ by $\frac{1}{7}s$ for every $O_{i_j,1}$, $j \in [1,k]$.

Case 2. In this case x_h (h defined as in Case 2 in the definition of the counting scheme) was the beginning of an interval before scheduling O_s. There are two possible cases when scheduling O_s. If $x_{s-1} + a_{s-1} + 1 \bmod P$ is not occupied, then $x_s = x_{s-1} + a_{s-1} + 1$ and no shifts are made when scheduling O_s. If $x_{s-1} + a_{s-1} + 1 \bmod P$ is occupied, then at most $r - \aleph_r$ shifts are made when scheduling O_s. We count r shifts for every operation

O_h, $h \in [r+1, \min\{2r - \frac{1}{7}r, s-1\}]$. Since the number of shifts is $\leq r - \min\{r - \frac{1}{7}r, \aleph_r\}$ we can increase Φ by $\frac{1}{7}\aleph_r$ for each of these operations. Since for each operation O_h, $h \in [\min\{2r - \frac{1}{7}r + 1, s-1\}, s-1]$ we count $r + \frac{1}{7}r$. Since at most r shifts are made we can increase Φ by $\frac{1}{7}\aleph_r \leq \frac{1}{7}r$ for each of these operations. When scheduling O_s at most $r - \aleph_r$ shifts are made. Since the number of intervals in decreased by at least one we can decrease Φ by $\frac{1}{7}(s - \aleph_r)$. Hence counting $\frac{6}{7}s$ would not underestimate the number of shifts plus the increase of Φ.

Case 3. For every operation O_h, $h \in [r+1, s]$ at most $r - \min\{\frac{1}{7}r, \aleph_r\}$ shifts are made if $h \leq 2r - \frac{1}{7}r$ and at most r otherwise. Clearly, we do not underestimate the number of shifts plus the increase of Φ when counting as above. □

Lemma 5.2. *The algorithm never makes Φ negative.*

Proof. We show that the following invariant is valid during a run of the algorithm: After scheduling operation O_t $\Phi \geq \frac{1}{7}t\aleph_t$. Clearly the invariant holds after scheduling operation O_1. In Case 1, Φ is increased by $\frac{1}{7}\aleph_r$ for every operation O_h with $h \in [r+1, s]$ $h \neq i_j$ for $j \in [1, k]$. When scheduling O_s at most k new intervals are created. Since Φ is increased by $\frac{1}{7}\aleph_r + \frac{1}{7}s$ for every operation O_{i_j}, $j \in [1, k]$ the invariant remains valid in this case. In Case 2, Φ is increased by $\frac{1}{7}\aleph_r$ for every operation O_h with $h \in [r+1, s-1]$. For operation O_s the value of Φ is changed by $\frac{1}{7}\aleph_r - \frac{1}{7}s$. Since the number of intervals decreased by at least one when scheduling operation O_s the invariant remains valid. In Case 3, Φ is increased by $\frac{1}{7}\aleph_r$ for every operation O_h with $h \in [r+1, s]$. Since the number of intervals is not changed in this case when scheduling operations $O_{r+1}, \ldots, O_s$ the invariant is valid for this case. □

Theorem 5.2. *The algorithm makes at most $\frac{\mathfrak{m}^2 - \mathfrak{m}^{3/2}/14}{2}$ shifts.*

Proof. By Lemmas 5.1 and 5.2 it is enough to show that the total count is at most

$$\sum_{i=2}^{\mathfrak{m}} (i - \tfrac{1}{7}\sqrt{i}) \leq \tfrac{\mathfrak{m}^2 + \mathfrak{m}}{2} - \tfrac{1}{7}\sum_{i=2}^{\mathfrak{m}} \sqrt{i} \leq \tfrac{\mathfrak{m}^2 - \mathfrak{m}^{3/2}/14}{2},$$

where $\mathfrak{m}$ exceeds some small constant. Let $1 = z_1 < \ldots < z_l \leq \mathfrak{m}$ be the different values of z during the run of the algorithm. We assume $z_l = \mathfrak{m}$ (The case $z_l < \mathfrak{m}$ can be proved similarly). Let $r = z_i$ and $s = z_{i+1}$, $i \in [1, l-1]$ and consider again the three cases as above.

Case 1. Let $\mathfrak{O}_1 = \{O_{i_j} \mid j \in [1, k]\}$ and $\mathfrak{O}_2 = \{O_h \mid h \in [r+1, s-1], h \neq i_j, j \in [1, k]\}$. The count for each operation $O_{i_j} \in \mathfrak{O}_1$ is $\frac{6}{7}i_j$ and the count for O_s is $\frac{6}{7}s$. For each operation $O_h \in \mathfrak{O}_2$ with $h \geq 2r - \frac{1}{7}r + 1$ we have $r + \frac{1}{7}h \leq \frac{7}{13}h + \frac{1}{7}h \leq h - \frac{1}{7}\sqrt{h}$. Let $\mathfrak{O}_2' \subset \mathfrak{O}_2$ be the set of operations in $\mathfrak{O}_2$ with $h \in [r+1, 2r - \frac{1}{7}r]$. First assume that there are $q\sqrt{r}$, $q \geq 1$ operations in $\mathfrak{O}_2'$. Let $\mathfrak{O}_2' = \{O_{h_1}, \ldots, O_{h_{q\sqrt{r}}}\}$. Then the total count for these operations is at most

$$\sum_{i=1}^{q\sqrt{r}} \left(r + \tfrac{1}{7}(h_i - r)\right) = \sum_{i=1}^{q\sqrt{r}} h_i - \tfrac{6}{7}\sum_{j=1}^{q\sqrt{r}} (h_j - r) \leq \sum_{i=1}^{q\sqrt{r}} h_i - \tfrac{6}{7}\sum_{j=1}^{q\sqrt{r}} j$$
$$= \sum_{i=1}^{q\sqrt{r}} h_i - \tfrac{6}{7} \cdot \tfrac{1}{2} \cdot (q^2 r + q\sqrt{r}) \leq \sum_{i=1}^{q\sqrt{r}} \left(h_i - \tfrac{3}{7}q\sqrt{r}\right) \leq \sum_{i=1}^{q\sqrt{r}} \left(h_i - \tfrac{1}{7}\sqrt{h_i}\right).$$

The last equation holds since $h_i \leq 2r$ for $i \in [1, q\sqrt{r}]$. Now assume that there are $< \sqrt{r}$ operations in $\mathfrak{O}'_2$. Let us consider the following two cases.

(i) If $s - r > 5\sqrt{r}$ then $k \geq 4\sqrt{r}$, *i.e.*, there are at least $4\sqrt{r}$ operations O_{i_j}. Let $\mathfrak{O}'_1$ be the set of these operations. Thus for at least $3\sqrt{r}$ of these operations it is $i_j > r + \sqrt{r}$. Now we can assign to each operation $O_h \in \mathfrak{O}'_2$ a subset $\mathfrak{O}_{1,h} \subset \mathfrak{O}'_1$ of size three such that for $O_{i_j} \in \mathfrak{O}_{1,h}$ $i_j \geq \max\{h - \sqrt{r} - 2, r + \sqrt{r}\}$ holds and $\mathfrak{O}_{1,h} \cap \mathfrak{O}_{1,h'} = \emptyset$ for $h \neq h'$ (This is not hard to show). Now we change the count for each operation O_{i_j} in a set $\mathfrak{O}_{1,h}$ and count $i_j - \frac{1}{14}\sqrt{i_j}$ (instead of only $i_j - \frac{1}{7}i_j$). This allows to count for each operation O_h in $\mathfrak{O}'_2$ only

$$\begin{aligned} &r + \tfrac{1}{7}(h - r) - \tfrac{3}{14}\max\left\{h - \sqrt{r} - 2, r + \sqrt{r}\right\} \\ &\leq r - \tfrac{1}{14}\max\left\{h - \sqrt{r} - 2, r + \sqrt{r}\right\} \leq h - \tfrac{1}{7}\sqrt{h}, \end{aligned}$$

where the last inequalities hold for $r \geq 9$ and since $h \leq 2r$.

(ii) If $s - r \leq 5\sqrt{r}$ then the count for operation O_s and the operations in $\mathfrak{O}'_2$ is at most

$$\begin{aligned} &\textstyle\sum_{O_h \in \mathfrak{O}'_2}\left(r + \frac{1}{7}(h - r)\right) + \frac{6}{7}s = \sum_{O_h \in \mathfrak{O}'_2}\left(h - \frac{6}{7}(h - r)\right) + s - \frac{1}{7}s \\ &\leq \textstyle\sum_{O_h \in \mathfrak{O}'_2}\left(h - \frac{6}{7}(h - r) - \frac{1}{7}\sqrt{r}\right) + s - \frac{1}{7}\sqrt{s} \leq \sum_{O_h \in \mathfrak{O}'_2}\left(h - \frac{1}{7}\sqrt{h}\right) + s - \frac{1}{7}\sqrt{s} \end{aligned}$$

Case 2. In this case the count for scheduling O_s is $\frac{6}{7}s \leq s - \frac{1}{7}\sqrt{s}$. The count for scheduling an operation O_h with $h \in \left[\min\left\{2r - \frac{1}{7}r + 1, s - 1\right\}, s - 1\right]$ is $r + \frac{1}{7}r \leq \frac{7}{13}h + \frac{1}{7}h \leq h - \frac{1}{7}\sqrt{h}$. The count for each operation O_h with $h \in \left[r + 1, \min\left\{2r - \frac{1}{7}r, s - 1\right\}\right]$ is at most r. Let $p = \min\left\{2r - \frac{1}{7}r, s - 1\right\}$. If $p - r \geq \sqrt{s}$ then the total count for scheduling operations O_h with $h \in [r + 1, p]$ is at most

$$r(p - r) = \textstyle\sum_{i=r+1}^{p} i - \sum_{j=1}^{p-r} j \leq \sum_{i=r+1}^{p} i - \frac{1}{2}\left[(p - r)^2 + (p - r)\right] \leq \sum_{i=r+1}^{p}\left(i - \frac{1}{7}\sqrt{i}\right).$$

If, on the other hand, $p - r < \sqrt{s}$, then the total count for scheduling operations $O_{h,1}$ with $h \in [r + 1, p]$ and O_s is at most

$$\begin{aligned} &r(p - r) + \tfrac{6}{7}s \leq \textstyle\sum_{i=r+1}^{p} i + s - \frac{1}{7}s \\ &\leq \textstyle\sum_{i=r+1}^{p}\left(i - \frac{1}{7}\sqrt{s}\right) + s - \frac{1}{7}\sqrt{s} \leq \sum_{i=r+1}^{p-r}\left(i - \frac{1}{7}\sqrt{i}\right) + s - \frac{1}{7}\sqrt{s}. \end{aligned}$$

Case 3. In this case $s \geq 2r$. The count for operation O_h with $h \in \left[r + 1, 2r - \frac{1}{7}r\right]$ is r and the count is $r + \frac{1}{7}\aleph_r$ for $h \in \left[2r + \frac{1}{7}r + 1, s\right]$. Altogether the count for operations O_h, $h \in [r + 1, s]$ is at most

$$r(s - r) + \tfrac{1}{7}\aleph_r\left(s - \tfrac{13}{7}r\right) \leq \textstyle\sum_{i=r+1}^{s}\left(i - \frac{1}{7}\sqrt{i}\right),$$

where the last equation holds for $s \geq 49$. $\square$

6. Concluding remark

Studying performance guarantee approximations for cycle-shop scheduling problems are barely begun because this paper represents probably all known results on that topic. The approximation results in Subsections 5.4 and 5.5 can be easily extended to the case $p_i = q_{\mu_i}/q$. However, the absolute error bounds in this case will be increased by multiplier q.

The most interesting generalizing open question is whether the class $\alpha_0 C \alpha_2 | \beta_{1-4}, p_{ij} = q_{\mu_{i'}} | \gamma$ contains NP-hard problems. Among not unit-time problems solvable in polynomial time in it we can call only $C2|p_{ij} = q_{\mu_i}|C_{\max}$ and $RF|n \geq m-2, p_{oj} = p, p_{ej} = q_{\mu_{e-1}}|C_{\max}$. A comprehensive list of open nonperiodic robotic flow-shop problems can be found in [BK]. The following list includes some minimal open cycle-shop problems.

Open

$L2|pmtn|C_{\max}$,
$L2|pmtn|\sum C_j$,
$L2|r_j, p_{ij} = 1|\sum U_j$,
$L2|no\,wait, r_j, p_{ij} = 1|L_{\max}$,
$RF3|r_j, p_o = p, p_e = q|C_{\max}$,
$RF3|r_j, p_o = p, p_e = q|\sum C_j$,
$RF3|p_o = p, p_e = q|L_{\max}$,
$RF3|r_j, p_o = 1, p_e = q|L_{\max}$,
$RFm|p_o = 1, p_e = q|L_{\max}$,
$RF3|p_o = 1, p_e = q|\sum U_j$,
$\boldsymbol{R}F2|no\,wait|B_{\max}$,
$\boldsymbol{R}1\varrho_{3,5}F3|no\,wait|B_{\max}$,
$\boldsymbol{R}1F|no\,wait, n = 2|P$,
$\boldsymbol{R}2F|no\,wait, n = 1|P$,
$\mathbb{R}1F|n = 2|P$,
$\mathbb{R}2F|n = 1|P$,
$C2|p_{ij} = p_i|C_{\max}$,
$C2|no\,wait, p_{ij} = p_i|C_{\max}$
$C2|chains, p_{ij} = 1|C_{\max}$,
$C2|no\,wait, chains, p_{ij} = 1|C_{\max}$,
$C2|chains, p_{ij} = 1|\sum C_j$,
$C2|no\,wait, chains, p_{ij} = 1|\sum C_j$,
$C2|r_j, p_{ij} = 1|L_{\max}$,
$C2|no\,wait, r_j, p_{ij} = 1|C_{\max}$,
$C2|r_j, p_{ij} = 1|\sum C_j$,
$C2|no\,wait, r_j, p_{ij} = 1|\sum C_j$,
$C3|p_{ij} = q_{\mu_i}|C_{\max}$,
$C2|no\,wait, p_{ij} = q_{\mu_i}|C_{\max}$,
$1\mathcal{C}|n = 2, p_i = 1|C_{\max}$,
$1C|no\,wait, n = 1, p_i = q_{\mu_i}/q|P$.

Remind that $J2|r_j, p_{ij} = 1|L_{\max}$ is also open and $J2|no\,wait, r_j, p_{ij} = 1|C_{\max}$ is NP-hard [T85]. The latter problem remains open for ordinary or strong NP-hardness [T98], and $J2|no\,wait, r_j, p_{ij} = 1|\sum C_j$ is strongly NP-hard [T98].

If ℓ is even, $C2|chains, p_{ij} = 1|C_{\max}$ and $C2|chains, p_{ij} = 1|\sum C_j$ can be solved in polynomial time by job-shop algorithms for $J2|p_{ij} = 1|C_{\max}$ [T97] and $J2|p_{ij} = 1|\sum C_j$ [K94, KT96]. So, the obstacle is the case of odd ℓ. By Corollary 4.3, $C2|chains, p_{ij} = 1|C_{\max}$ can be solved in pseudopolynomial time.

It is important to say that the size of each of the four chain-like precedence-constrained equal-release-date cycle-shop problems in the above list is $\mathfrak{n} + \log \ell + \sum_{i=1}^{\mathfrak{n}} \log \mathfrak{l}_i$, where $\mathfrak{n}$ is the number of chains, and $\mathfrak{l}_i$ is the length of the ith chain. Hence, algorithms polynomial in n, *i.e.* in the number of jobs, are of exponential time for the problems. Note that the reduction $C2|pmtn|C_{\max} \propto C2|chains, p_{ij} = 1|C_{\max}$ follows from the preemption-chaining theorem [T98A]. But the reduction $C2|pmtn|\sum C_j \propto C2|chains, p_{ij} = 1|\sum C_j$ is unknown. The subproblem of $1\mathcal{C}|n = 1, p_i = 1|C_{\max}$, where the period is less than the flow size of the cycle shop, remains open. In conclusion we conjecture that all problems established here to be solvable in pseudopolynomial time can be solved indeed in polynomial time.

References

[AP98] A. Agnetis and D. Pacciarelli, *Part sequencing in three machine no-wait robotic cells*, Technical Report 10-98, Dipartmento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, 1998.

[A99] A. Agnetis, *Scheduling no-wait robotic cells with two and three machines*, EJOR (to appear).

[A85] C. R. Asfahl, *Robots and Manufacturing automation*, John Wiley and Sons, New York, 1985.

[AK99] Y. P. Aneja and H. Kamoun, *Scheduling of parts and robot activities in a two-machine robotic cell*, Computers Ops. Res. **26** (1999), no. 4, 297–312.

[BFK97] N. Brauner, G. Finke and W. Kubiak, *A proof of the Lei and Wang claim*, Technical Note, Laboratoire Leibniz, Institut Imag, Grenoble, 1997.

[B95] P. Brucker, *Scheduling algorithms*, Springer, 1995.

[BGJ77] P. Brucker, M. R. Garey, and D. S. Johnson, *Scheduling equal-length tasks under tree-like precedence constraints to minimize maximum lateness*, Math. Oper. Res. **2** (1977), 275–284.

[BK98] P. Brucker and S. Knust, *Complexity results for single-machine problems with positive finish-start time lags*, Osnabrücker Schiften zur Mathematik, Reihe P (1998).

[BK] P. Brucker and S. Knust, *Operations research: complexity results of scheduling problems*, http://www.mathematik.uni-osnabrueck.de/research/OR/class/.

[BK99] P. Brucker and S. Kravchenko, *Preemption can make parallel machine scheduling problems hard*, Osnabrücker Schiften zur Mathematik, Reihe P, Nr. 211, 1999.

[BJS80] J. Bruno, J. W. Jones, and K. So, *Deterministic scheduling with pipelined processors*, IEEE Trans. on Comput. **C-29** (1980), 308–316.

[CC90] J. Carlier and Chrétienne, *On the dominance of K-periodic schedules in cyclic scheduling*, in Proc. 2nd International Workshop on Project Management and Scheduling, Compiégne, France, 1990, pp. 181–187.

[CKKL99] Y. Crama, V. Kats, J. van de Klundert and E. Levner, *Cyclic scheduling in robotic flowshops*, Annals of Operations Research: Mathematics of Industrial Systems (1999) (to appear).

[CK97] Y. Crama and J. van de Klundert, *Cyclic scheduling of identical parts in a robotic cell*, Oper. Res. **45** (1997), 952–965.

[CK97A] Y. Crama and J. van de Klundert, *Robotic flowshop scheduling is strongly NP-complete*, in Ten Years LNMB (W. K. Klein Haneveld, O. J. Vrieze and L. C. M. Kallenberg, eds.), CWI Tract 122, Amsterdam, The Netherlands, 1997, pp. 277–286.

[DT76] Yu. I. Degtiarev and V. G. Timkovsky, *On a model of optimal planning systems of flow type*, Technics of Communication Means, Ser. Automation Control Systems (1976), no. 1, 69–77. (in Russian)

[DS99] I.G. Drobouchevitch and V.A. Strusevich, *A heuristic algorithm for two-machine re-entrant shop scheduling*, Annals of Operations Research **86** (1999), 417–439.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[GMSZ83] S. C. Graves, H. C. Meal, D. Stefek, and A. H. Zeghmi, *Scheduling of re-entrant flow shops*, J. Operations Management **3** (1983), no. 4, 197–207.

[G93] J. N. D. Gupta, *Two-stage reentrant flowshop problem with repeated processing at the first stage*, Dept. of Management, Ball State Univ., Muncie, IN, 1993.

[HKS97] N. G. Hall, H. Kamoun and C. Sriskandarajah, *Scheduling in robotic cells: classification, two and three machine cells*, Oper. Res. **45** (1997), 421–439.

[HKS98] N. G. Hall, H. Kamoun and C. Sriskandarajah, *Scheduling in robotic cells: complexity and steady state analysis*, EJOR **109** (1998), no. 1, 43–65.

[HLP97] N. G. Hall, T.–E. Lee and M. E. Posner, *Periodic shop scheduling problems*, Tech. Report, Ohio State University (1997).

[HWZ98] J. Han, J. Wen and G. Zhang, *A new approximation algorithm for UET-scheduling with chain-type precedence constraints*, Computers Ops. Res. **25** (1998), no. 9, 767–771.

[H94] C. Hannen, *Study of an NP-hard cyclic scheduling problem: the periodic recurrent job shop*, EJOR **72** (1994), 82–101.

[HM95] C. Hannen and A. Munier, *A study of the cyclic scheduling problem on parallel processors*, Discrete Appl. Math. **57** (1995), 167–192.

[HM96] C. Hannen and A. Munier, *Cyclic scheduling problems: an overview*, in Scheduling theory and applications (P. Chrétienne, E. G. Coffman, J. K. Lenstra and Z. Liu, eds.), John Wiley, New York, 1996.

[HK98] J. Hurink and S. Knust, *Flow-shop problems with transportation times and a single robot*, Osnabrücker Schriften zur Mathematik, Reihe P, Nr. 201, 1998.

[KL97] V. Kats and E. Levner, *A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling*, Oper. Res. Let. **21** (1997), 159–164.

[KL98] V. Kats and E. Levner, *Cyclic scheduling of operations for a part type in an FMS handled by a single robot: a parametric critical path approach*, Inter. J. FMS **10**, 129–138.

[KM80] V. B. Kats and Z. N. Mikhailetsky, *Exact solution of a cyclic scheduling problem*, Automation and Remote Control **4** (1980), 187–190.

[KSI91] H. Kise, T. Shioyama and T. Ibaraki, *Automated two-machine flowshop scheduling: a solvable case*, IIE Transactions **23**, no. 1, 10–16.

[K99] S. Knust, *Shop-scheduling problems with transportation*, Ph. D. thesis, Fachbereich Mathematik/Informatik, Universität Osnabrück, 1999.

[K81] P. M. Kogge, *The architecture of pipelined computers*, McGraw Hill, New York, 1981.

[K94] S. A. Kravchenko, *Minimizing total completion time in two-machine job shops with unit processing times*, Preprint N4, Institute of Technical Cybernetics, Minsk, Belorussia, 1994. (in Russian)

[K98] S. A. Kravchenko, *A polynomial time algorithm for a two-machine no-wait job-shop scheduling problem*, EJOR **106** (1998), 101–107.

[K99A] S. A. Kravchenko, *Minimizing the number of late jobs for the two-machine unit-time job-shop scheduling problem*, Discrete Appl. Math. **98** (1999), no. 3, 209–217.

[K99B] S. A. Kravchenko, *On the complexity of minimizing the number of late jobs in unit time open shops*, Discrete Appl. Math. (to appear).

[KT96] W. Kubiak and V. G. Timkovsky, *A polynomial-time algorithm for total completion time minimization in two-machine job-shop with unit-time operations*, EJOR **94** (1996), 310–320.

[LLRKS93] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *Sequencing and scheduling: algorithms and complexity*, in Handbook on Operations Research and Management Science, Volume 4: Logistics of Production and Inventory (S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, eds.), Elsevier Science Publishers B. V., Amsterdam, 1993, pp. 445–552.

[LP97] T. E. Lee and M. E. Posner, *Performance measures and schedule patterns in periodic job shops*, Oper. Res. **45** (1997), 72–91.

[LW89] L. Lei and T.–J. Wang, *A proof on the NP-completeness of the hoist scheduling problems*, GSM working paper, Rutgers university (1989).

[L] J. K. Lenstra, *Private communication.*

[LRKB77] J. K. Lenstra, A. H. G. Rinnooy Kan and P. Brucker, *Complexity of machine scheduling problems*, Annals of Discrete Math. **1** (1977), 363–362.

[LRK78] J. K. Lenstra, A. H. G. Rinnooy Kan, *Complexity of scheduling under precedence constraints*, Oper. Res. **26** (1978), 22–35.

[LRK79] J. K. Lenstra, A. H. G. Rinnooy Kan, *Computational complexity of discrete optimixation problems*, Annals of Discrete Math. **4** (1979), 121–140.

[LRK80] J. K. Lenstra, A. H. G. Rinnooy Kan, *Complexity results for scheduling chains on a single machine*, EJOR **4** (1980), 270–275.

[LVW84] J.Y.-T. Leung, O. Vornberger, and J. D. Witthoff, *On some variants of the bandwidth minimization problem*, SIAM J. Comput. **13** (1984), 650–667.

[LY90] J.Y.-T. Leung and G.H. Young, *Preemptive scheduling to minimize mean weighted flow time*, IPL **34** (1990), 47–50.

[LY90A] J.Y.-T. Leung and G.H. Young, *Minimizing Total Tardiness on a Single Machine with Precedence Constraints*, ORSA Journal on Computing **2** (1990), 346–352.

[LA84] V. Lev and I. Adiri, *V-shop scheduling*, EJOR **18** (1984), no. 1, 51–56.

[LKL97] E. Levner, V. Kats, and V. Levit, *An improved algorithm for cyclic flowshop scheduling in a robotic cell*, EJOR **97** (1997), 500–508.

[LMC74] E. M. Livshits, Z. N. Mikhailetsky and E. V. Chervyakov, *A scheduling problem in automated flow line with an automated operator*, Computational Mathematics and Computerized Systems **5** (1974), 151–155. (in Russian)

[MPSW91] S. T. McCormick, M. L. Pinedo, S. Shenker and B. Wolf, *Transient behavior in a flexible assembly system*, International Journal of FMS **3** (1991), 27–44.

[MR94] S. T. McCormick and U.S. Rao, *Some complexity results in cyclic scheduling*, Mathematical and Computer Modelling **20** (1994), 107–122.

[MBM88] V. S. Mikhalevich, S. A. Beletsky, and A. P. Monastyrev, *Optimization of multi-stage cyclic service of a production line by a transmanipulator*, Cybernetics **24** (1988), no. 4, 500–511.

[M99] M. Middendorf, *Cycle-shop scheduling problems solvable in pseudopolynomial time*, Unpublished manuscript, 1999.

[MP93] T. E. Morton and D. W. Pentico, *Heuristic scheduling systems*, John Wiley, New York, 1993.

[M96] A. Munier, *The complexity of a cyclic scheduling problem with identical machines and precedence constraints*, EJOR **91** (1996), 471–480.

[P95] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Prentice-Hall, Englwood Cliffs, NJ, 1995.

[RY93] R. Ramazini and N. Younis, *Repetitive pure flowshop problem: a permutation approarch*, Computers and Indusrtial Eng. **24** (1993), no. 1, 125–129.

[RJ93] U. S. Rao and P. L. Jackson, *Subproblems in identical jobs cyclic scheduling: properties, complexity and solution approaches*, School of Research and Industrial Engineering, Cornell University, Ithaca, NY, 1993.

[R92] R. Roundy, *Cyclic schedules for job shops with identical jobs*, Math. Oper. Res. **17** (1992), 842–865.

[SU89] P. Serafini and W. Ukovich, *A methematical model for periodic scheduling problems*, SIAM J. Disc. Math. **2** (1989), 550–581.

[SU89A] P. Serafini and W. Ukovich, *A mathematical model for the fixed-time traffic control problem*, EJOR **41** (1989), 1–14.

[SSSBK92] S. P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz and W. Kubiak, *Sequencing of parts and robot moves in a robotic cell*, International Journal of FMS **4** (1992), 331–358.

[SL86] C. Sriskandarajah and P. Ladet, *Some no-wait jobs scheduling problems: complexity results*, EJOR **24** (1986), 424–438.

[T77] V. G. Timkovsky, *On transition processes in systems of flow type*, Technics of Communication Means, Ser. Automation Control Systems (1977), no. 1(3), 46–49. (in Russian)

[T81] V. G. Timkovsky, *Computational complexity and approximation of the cycle shop scheduling problem*, in The 1st All-Union Consultation on Statistical and Discrete Analysis of Non-Numerical Information, on Expert Estimates and Discrete Optimization (Yu. N. Turin, ed.), Kazakh State University, Moscow–Alma-Ata, 1981, pp. 220–221. (in Russian)

[T85] V. G. Timkovsky, *On the complexity of scheduling an arbitrary system*, Soviet Journal of Computer and System Sciences (1885), no. 5, 46–52.

[T86] V. G. Timkovsky, *An approximation for the cycle-shop scheduling problem*, Economics and Mathematical Methods **22** (1986), no. 1, 171–174. (in Russian)

[T92] V. G. Timkovsky, *Discrete mathematics in the world of machines and parts: introduction to mathematical modelling for discrete manufacturing problems*, Nauka, Moscow, 1992. (in Russian)

[T97] V. G. Timkovsky, *A polynomial-time algorithm for the two-machine unit-time release-date job-shop schedule-length problem*, Discrete Appl. Math. **77** (1997), 185–200.

[T98] V. G. Timkovsky, *Is a unit-time job shop not easier than identical parallel machines?*, Discr. Appl. Math. **85** (1998), 149–162.

[T98A] V. G. Timkovsky, *Identical parallel machines vs. unit-time shops, preemptions vs. chains and other offsets in scheduling complexity*, Dept. of Computing and Software, McMaster University, Hamilton, Ontario, 1998.

[U75] J. D. Ullman, *NP-complete scheduling problems*, J. Comput. System Sci. **10** (1975), 384–393.

[WSV97] M. Y. Wang, S. P. Sethi and S. L. van de Velde, *Minimizing makespan in a class of re-entrant shops*, Oper. Res. **45** (1997), 702–712.

†Institute of Applied Computer Science and Formal Description Methods, University of Karlsruhe, D-76128 Karlsruhe, Germany

E-mail address: mmi@aifb.uni-karlsruhe.de

*Star Data Systems Inc., Commerce Court South, 30 Wellington St.W., Suite 300, P.O. Box 283, Toronto, Ontario M5L 1G1 Canada

E-mail address: vtimkovs@stardata.ca