European Journal of Operational Research 209 (2011) 51–56

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A faster algorithm for 2-cyclic robotic scheduling with a fixed robot route and interval processing times

Vladimir Kats [a,1], Eugene Levner [b,c,*]

[a] Institute for Industrial Mathematics, 57/10 Klazkin St., Beer-Sheva 84641, Israel
[b] Holon Institute of Technology, Holon 58102, Israel
[c] Bar Ilan University, Ramat Gan 52900, Israel

ARTICLE INFO

Article history:
Received 31 May 2010
Accepted 2 October 2010
Available online 29 October 2010

Keywords:
Efficient algorithms
Graph-theoretic models
Cyclic scheduling
Polynomial models
Robotic scheduling

ABSTRACT

Consider an m-machine production line for processing identical parts served by a mobile robot. The problem is to find the minimum cycle time for 2-cyclic schedules, in which exactly two parts enter and two parts leave the production line during each cycle. This work treats a special case of the 2-cyclic robot scheduling problem when the robot route is given and the operation durations are to be chosen from prescribed intervals. The problem was previously proved to be polynomially solvable in $O(m^8 \log m)$ time. This paper proposes an improved algorithm with reduced complexity $O(m^4)$.

1. Introduction

Consider a robotic flowshop line consisting of m workstations ("machines") processing identical parts, in which a single robot transports the parts between the stations and loads/unloads the parts. The robot repeats its moves periodically, such a production process is called *cyclic*. The periodic sequence of operations performed by the robot is called a *schedule* and its duration is the *cycle time*.

A cyclic sequence in which during each cycle exactly K parts enter the line and K parts leave the line is called a *K-cyclic* or *K-degree schedule*. Notice that each robot's move from one workstation to another appears K times during each cycle. At the end of the cycle the line returns to its original state. The *mean cycle time* is the cycle time divided by K. The *throughput rate* of the production process is the average number of finished parts produced per unit of time; it is the inverse of the mean cycle time. The cyclic scheduling problem is to specify a sequence of robot moves and the processing times so that to maximize the throughput rate, or, equivalently, to minimize the mean cycle time.

Cyclic robot scheduling problems have been intensely studied over the past decades. Several surveys (see e.g. Crama et al., 2000; Dawande et al., 2005; Hall, 1999) and a recent monograph by Dawande et al. (2007) bring together main algorithmic results. The multi-cyclic schedules may have a higher throughput rate than the best 1-cyclic ones, as has been reported by many authors, see e.g. Blazewicz et al. (1989), Levner et al. (1996), Crama and Klundert van de (1997), Finke and Brauner (1999), Che et al. (2002, 2003), Chu (2006), Che and Chu (2009). Several fast algorithms have been developed for the 2-cyclic scheduling problem with *constant processing times* (Che et al., 2002, 2003; Chu, 2006; Kats and Levner, 2009). In a more general situation, when processing times are chosen from prescribed intervals, the methods derived for constant times are not applicable. We consider the 2-cyclic scheduling problem with *interval data* in the case when the robot route is given in advance. Such a problem type – in which the robot route is fixed, has theoretical and practical importance on its own; moreover, it can be used as an estimating sub-problem in branch-and-bound computational schemes for solving a more general scheduling problem – in which the best robot route is to be found. The interested reader can find more information on practical applications of this problem in Lei (1993), Ioachim and Soumis (1995), Chen et al. (1998), and Kats et al. (2008).

The case of interval data is important in practice because in such a case constraints of the production model are more flexible. In the present paper, we propose a new algorithm of complexity $O(m^4)$ for this problem, thus improving the earlier $O(m^8 \log m)$-time geometric algorithm developed in Kats and Levner (2010). In the next section we describe the problem. In Section 3 we reduce it to a parametric critical path problem in a graph. In Sections 4

* Corresponding author at: Holon Institute of Technology, Holon 58102, Israel. Tel.: +972 5 46246757; fax: +972 3 7384039.

E-mail addresses: vkats@iimath.com (V. Kats), elevner@ull.es, levner@hit.ac.il, levnere@mail.biu.ac.il (E. Levner).

[1] Tel.: +972 8 6421885.

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.10.002

and 5 we present the parametric sub-algorithm and analyze its properties. In Section 6 we describe a new algorithm and estimate its complexity. Section 7 concludes the paper.

2. Problem description

A robotic production line has two types of resources - workstations and robots. Consider a flow-line having m sequential workstations $S_1,\ldots,S_m$. Let S_0 and S_{m+1} denote the input and output stations, respectively. The line processes identical parts. The order of operations and movements of each part along the stations is represented by the sequence $S = (S_0,S_1,\ldots,S_m,S_{m+1})$, where the kth component indicates that the kth operation is performed in station S_k, $k = 0,\ldots,m + 1$.

A robot is used to transport parts between various stations and load/unload them. As said above, we assume that the robot's route is given. In the 2-cyclic schedules, the identical parts are loaded into the line at time $\ldots\ -rT, -rT + T_1,\ldots, -2T, -2T + T_1, -T, -T + T_1, 0, T_1, T, T + T_1, 2T,\ldots, rT + T_1, (r + 1)T,\ldots$, where $T_1 < T$ and r is an integer. The parts loaded at time rT (respectively, at time $rT + T_1$), $r = 0,\pm1,\pm2,\ldots$ are called the *parts of Class 1* (respectively, the *parts of Class 2*). Exactly two parts enter and two parts leave the line during each cycle. T is called the *cycle time*, T_1 is called the *semi-cycle time*; they are the unknowns of our scheduling problem whose optimal values have to be found.

Let us describe a robot cyclic route, denoted by R. Since the loaded robot always moves from a station at which it takes a part to the next station in the sequence S, the route R can be presented compactly as a $2(m + 1)$-long sequence: $R = (S_{R(0)},S_{R(1)},S_{R(2)},\ldots, S_{R(m)},S_{R(m+1)},\ldots,S_{R(2m+1)})$, or, simply, $R = ([0],[1],\ldots,[m],[m+1],\ldots,[2m+1])$, using, briefly, $[k]$ instead of $S_{R(k)}$.

Consider interval $[0,T)$. Within this interval the robot performs twice all its operations; it starts at time $t = 0$ with loading a part from the input station S_0, so $[0] = 0$. The sequence $(0,[1],\ldots, [m],\ldots,[2m+1])$ is a permutation of the workstation numbers $0,1,\ldots,m$ in which each number is repeated twice.

The scheduling problem must satisfy the following conditions:

- ($C1$) After being loaded at workstation S_k, a part stays there for time p_k, where $L_k \leqslant p_k \leqslant U_k$, and L_k and U_k are given constants, $k = 1,\ldots,m$. Time p_k is called *processing time*; L_k and U_k define interval $[L_k, U_k]$ of possible values of p_k, $k = 1,\ldots,m$.
- ($C2$) The loaded robot requires time d_k to move a part from station S_k to S_{k+1}, $k = 0,\ldots,m$.
- ($C3$) The unloaded robot requires time r_{ij} to run from S_i to S_j, $1 \leqslant i \leqslant m+1$, $0 \leqslant j \leqslant m$.
- ($C4$) After a part is processed in a workstation, it must be unloaded and moved immediately by the robot to the next workstation in S and then be processed on it without pause; this requirement is called the *no-wait condition*. In this case, *processing time* and *residence time* (the latter term denotes time spent by a part on a machine) coincide. This and other types of the no-wait robotic cells are discussed in detail in Dawande et al. (2007).

The decision variables in the problem are:

p_k = part's processing time at station S_k, for $k = 1,\ldots,m$;
t_k (and, respectively, t'_k) = the completion time of the operation on workstation S_k within the cycle $[0,T)$ for a part of Class 1 (respectively, for a part of Class 2), $k = 0,\ldots,m$;
T = the cycle time, and T_1 = the semi-cycle time.

Given processing sequence S, robot's route R, real numbers d_j, r_{ij}, L_k, and U_k ($k = 1,\ldots,m$; $i = 1,\ldots,m+1$; $j = 0,\ldots,m$), a *feasible schedule* is a set of operation completion times t_k, t'_k and processing times p_k ($k = 1,\ldots,m$) lying within prescribed intervals $[L_k,U_k]$ and providing that the empty robot has sufficient time to travel between the workstations, for some fixed pair of values T and T_1. Such parameter values T and T_1 will be called *feasible*. The scheduling problem is to find a feasible schedule $\{\{t_k\},\{p_k\}\}$ and a feasible pair (T,T_1) so that cycle time T is minimum, and, thus, the throughput rate is maximum.

3. The critical path reformulation

For a given periodically repeated robot route $R = ([0],[1],\ldots, [m],[m+1],\ldots,[2m+1])$, the completion times t_k and t'_k satisfy the following chain of inequalities:

$$0 = t^*_{[0]} < t^*_{[1]} < t^*_{[2]} \cdots < t^*_{[2m+1]} < T, \qquad (1)$$

where $t^*_{[q]}$ denotes either t_k or t'_k.

The symbol * in $t^*_{[q]}$ is used to distinguish whether the robot transports a part of Class 1 or Class 2 during its $[q]$-th move; namely, $t^*_{[q]}$ is t_k, for some k, if the part is of Class 1 and $t^*_{[q]}$ is t'_k if the part is of Class 2. This information is given by the sequence R. For example, if $m = 4$ and robot route $R_0 = (0,4,2',1,3',2,0',4',3,1')$, then i (respectively, i') denotes that a part of Class 1 (respectively, of Class 2) is moved by the robot from workstation i, where $i = 0,1,2,3,4$.

The chain (1) in this case is the following:

$$0 = t_0 < t_4 < t'_2 < t_1 < t'_3 < t_2 < t'_0 < t'_4 < t_3 < t'_1 < T.$$

We will study the scheduling problem in time period $[0,T)$ and assume that the robot route R and, therefore, the inequalities (1) are given. We distinguish the case where $t_k > t_{k-1}$ in (1) (we call it "Case A") from the case where $t_k < t_{k-1}$ in (1) ("Case B"). The cases will be displayed in the constraints 4 and 5 below. For instance, for the route R_0considered in the above example, operations on workstations $\{1,2,3\}$ belong to the Case A while operation $\{4\}$ is in Case B. Also, we need to distinguish the case where $t_k < t'_k$ from the case $t_k > t'_k$ in (1) as is displayed in the constraints 7 and 8 below.

Then, taking into account the chain of inequalities (1), the scheduling problem in interval $[0,T)$ can be formulated as the following linear program:

$$\textit{Problem } P: \text{Find } T^* = \text{minimum } T, \qquad (2)$$

$$\text{subject to} \quad L_k \leqslant p_k \leqslant U_k, \qquad (3)$$

$$p_k = t_k - t_{k-1} - d_{k-1}, \quad \textit{if } t_k > t_{k-1} \quad \textit{in}(1), \qquad (4)$$

$$p_k = T + t_k - t_{k-1} - d_{k-1}, \quad \textit{if } t_k < t_{k-1} \quad \textit{in}(1), \qquad (5)$$

$$t^*_{[q]} + d_{[q]} + r_{[q]+1,[q+1]} \leqslant t^*_{[q+1]}, \qquad (6)$$

$$t'_k = t_k + T_1, \quad \textit{if } t_k < t'_k \quad \textit{in}(1), \qquad (7)$$

$$t'_k = t_k + T_1 - T, \quad \textit{if } t_k > t'_k \quad \textit{in}(1), \qquad (8)$$

$$t_0 = 0, \qquad (9)$$

where $k = 1,\ldots,m$; $q = 0,1,2,\ldots,2m + 1$, $t^*_{[2m+2]} \equiv T$, $r_{[2m+1]+1,[2m+2]} \equiv r_{[2m+1]+1,[0]}$, $t'_0 = T_1$.

Relation (3) describes the two-sided constraints on the processing times. Relations 4 and 5, depending on the mutual position of t_k and t_{k-1}in chain (1), express the no-wait condition, i.e., immediately after being completed on machine $k - 1$ (at moment t_{k-1}) a part is transported by the robot to machine k (which requires d_{k-1} units of time) and then without delay is processed at that machine during p_k units of time. Inequalities (6) guarantee that the robot has enough time to arrive at each machine ($S_{[k+1]}$) before the corresponding processing operation on this machine is finished. Eqs. (7) and (8) describe relations between variables of Class 1, denoted by t_k, and those of Class 2, denoted by t'_k, depending on their mutual position in (1). These equations guarantee that the parts of