Annals of Operations Research 57(1995)217–232

Scheduling a two-machine robotic cell: A solvable case

Eugene Levner
Department of Automated Systems, Centre for Technological Education, Affiliated with Tel-Aviv University, 52 Golomb St., Holon 58102, Israel
E-mail: msegen@pluto.mscc.huji.ac.il

Konstantin Kogan
Department of Industrial Engineering, Tel-Aviv University, Ramat-Aviv, 69978 Tel-Aviv, Israel

Ilya Levin
Department of Automated Systems, Centre for Technological Education, Affiliated with Tel-Aviv University, 52 Golomb St., Holon 58102, Israel

The paper deals with the scheduling of a robotic cell in which jobs are processed on two tandem machines. The job transportation between the machines is done by a transportation robot. The robotic cell has limitations on the intermediate space between the machines for storing the work-in-process. What complicates the scheduling problem is that the loading/unloading operation times are non-negligible. Given the total number of operations n, an optimal $O(n \log n)$-time algorithm is proposed together with the proof of optimality.

Keywords: Scheduling, robotic cell, tandem machines, optimization.

1. Introduction

Small-scale robotic cells with two tandem CNC machines for job processing and robots serving for transportation of jobs and tools are quite commonly found in real applications. They are the simplest automated blocks, the "islands of automation", in modern computer-integrated manufacturing. In order to efficiently utilize these automated facilities, diverse scheduling problems are to be solved (see, among others, Blazewicz et al. [2], Kusiak [5], Stecke and Suri [9]).

Since the introduction of the first analytical flowshop scheduling model by Johnson [3], a number of subsequent works on different versions of flowshop scheduling have appeared (see, for example, the surveys by Belen'kii and Levner [1] and Lawler et al. [6]). For recent examples, we can refer to the papers by Kise et al. [4], Stern

and Vitner [10], and Panwalkar [8], in which they investigate robotic cells with limitations on the intermediate space between the machines for storing the work-in-process (WIP).

Kise et al. [4] have considered a two-machine, one-robot, no-WIP-storage scheduling problem with a centralized storage and solved it efficiently. Another modification of the two-machine, one-robot flowshop scheduling problem, with job-dependent robot's transportation times has been considered by Stern and Vitner [10], who claimed the NP-hardness of the problem and suggested an approximate $O(n \log n)$-time algorithm (n = the number of jobs to be processed).

Panwalkar [8] has considered another version of the no-WIP-storage cell configuration, with no space after the first machine for storing the completed jobs, but having the space for storing the WIP before the second machine, and has solved the problem in quadratic time using the Johnson algorithm as a subroutine. In the latter work, the loading/unloading operation times are assumed to be negligible and the transportation times are job independent.

In the present paper, we intend to study the two-machine robotic cell in the same geometric configuration as that in the Panwalkar [8] model. Proceeding from practical considerations, we complicate the model by allowing the loading/unloading operation times to be non-negligible.

In the following section, we describe the robotic cell that motivated this study and formulate the dynamics of the system. Then we describe the scheduling problem and its graph representation. Starting from the analysis of critical path properties in graphs, we suggest an $O(n \log n)$-time algorithm for finding an optimal sequence that minimizes the completion time of n jobs, and prove its optimality. In conclusion, we refer to some possible directions for further research.

2. Description of the robotic cell

Consider the robotic cell that motivated this study. It is a component of the computer-integrated manufacturing system CIM-2000 MECHATRONICS produced by DEGEM SYSTEM, Ltd (Israel) which is up and running in the Holon Centre for Technological Education.

The cell contains two tandem CNC machines: milling machine Denford TRIAC VMC (machine MA), and lathe DEGEM NCL 2000 (machine MB). The cell also contains:

- one transporting robot DEGEM PS 2000 (robot TR);
- two robotic loaders, Mitsubishi HM01 (robots RA and RB);
- input automatic storage/retrieval station (AS/RS) for storing blanks and tools, and output AS/RS for storing finished parts, DEGEM ST-2000.

Cell configuration and capacity limitations of the input AS/RS do not allow it to be used for storing jobs completed on machine MA. At the same time, the output

AS/RS may be used for storing the jobs completed on MA; thus, the latter storage station serves as the integrated output-WIP AS/RS. The cell layout is depicted in figure 1.

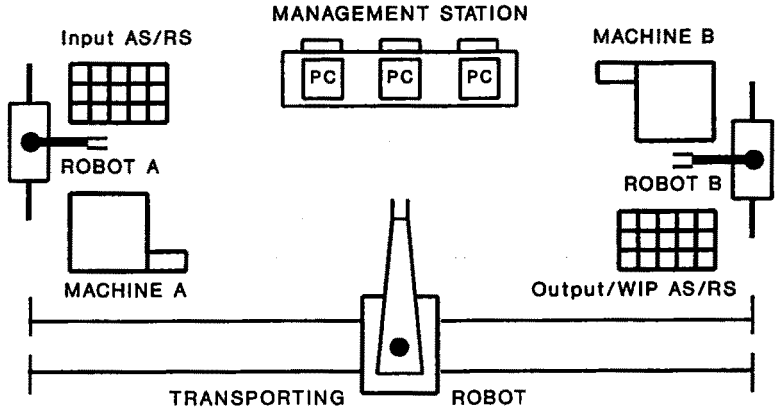

Figure 1. Layout of the robotic cell.

At time zero, there are n jobs (parts) available in the input AS/RS; they are to be processed in turn on machines MA and MB.

The part's route in the cell is as follows: first, a due part is moved by the robot RA from the input AS/RS to machine MA; next, the transporting robot TR picks up the part finished on MA and delivers it to the output/WIP station where a stacker places the part into a predetermined cell, within the service range of robot RB. Next, RB loads the part on machine MB (if and when it is empty) and, after finishing the part on MB, puts the latter on the output AS/RS.

The problem is to schedule jobs on the machines so as to minimize the makespan M under the following standard restrictions:

(i) Every machine can perform at most one job at a time, and each job must be performed only on one machine at a time.

(ii) The same job sequence occurs on each machine.

(iii) The space of the output-WIP AS/RS is sufficiently large for storing all the required parts finished on MA.

The following non-standard conditions complicate the problem in comparison with the Johnson [3] and Panwalkar [8] flowshop models:

(1) the loading/unloading operation times are considered to be non-negligible, as is often the case in industry;

(2) the times for placing jobs into appropriate cells of the output-WIP storage are allowed to be job dependent.

3. Dynamics of the system

Let $P = (j_1,\ldots,j_n)$ be an arbitrary sequence of given jobs, where j_k stands for the kth job to be processed in the sequence. The jobs are processed in sequence P (the same on two machines) according to the following technological rules:

Rule 1 (Controlling operations related to machine MA):

(a) Robot RA starts its life cycle by taking the first job j_1 from the input AS/RS and loading it on machine MA.

(b) Next, machine MA starts processing job j_1.

(c) After completing the processing of j_1 on MA, transporting robot TR unloads the latter job from MA in order to move it to the WIP AS/RS before machine MB. Concurrently, robot RA starts its next life cycle by turning to the input AS/RS for the next job j_2 scheduled to be processed.

(d) Next, all other jobs $j_2\ldots,j_n$ are loaded, processed on machine MA and unloaded by transporting robot TR, one after another. The unloading of job j_k, $2 \le k \le n$, by robot TR starts either at the time when j_k is finished on MA, or at the time when TR returns to MA after delivering j_{k-1} to the MB's area, whichever is larger.

Rule 2 (Controlling the moves of the transporting robot):

(a) After job j_1 is unloaded from MA, robot TR carries it from MA to the area of machine MB and places it into the output-WIP AS/RS.

(b) Immediately after that, TR moves back to machine MA.

(c) Concurrently, the stacker attached to the output-WIP AS/RS places job j_1 into a predetermined storage cell (within the service range of robot RB).

Other jobs $j_2\ldots,j_n$ are transported by TR and the stacker in a similar way; after delivering the last job, TR stops near the WIP AS/RS.

Rule 3 (Controlling operations related to machine MB):

(a) After job j_1 is delivered by the stacker, robot RB (which originally waits in an initial position, near MB) reloads it to machine MB.

(b) After that, machine MB starts processing job j_1.

(c) After j_1 is completed on MB, robot RB unloads it and delivers it to the output AS/RS. Then RB returns to its initial position, from where it starts its new life cycle by taking the next job from the WIP AS/RS (if and when it is available).

All other jobs $j_2,\ldots,j_n$ are performed on MB one after another, in a similar way as job j_1, until the last job is finished on MB and delivered to the output AS/RS.

4. Mathematical description of the system

4.1. NOTATION

n = the given number of jobs;

For $i = 1,\ldots,n$:

$L_A(i)$ = "loading time" for job i on machine MA (required to perform the job handling operations described in rule 1(a));

$P_A(i)$ = processing time for job i on machine MA;

$U_A(i)$ = time required for robot TR to unload job i from MA;

$L_B(i)$ and $P_B(i)$ are, respectively, job-loading and processing times on machine MB (described above in rules 3(a) and 3(b));

$U_B(i)$ = time required for RB to unload job i from MB, deliver it to the output AS/RS, and return to MB (see rule 3(c));

T = (job independent) travel time required for TR to move any job from MA to the WIP storage and drop it there;

R = "return time" required for robot TR to return from the WIP-output AS/RS to machine MA;

$W(i)$ = "placement time" required for the stacker to place job i into the appropriate cell of the WIP storage.

Define also the key time instants of a schedule $P = (j_1, j_2,\ldots,j_n)$ as follows:

$s^A(i)$: the time instant when RA starts its ith life cycle in P by turning from its initial position for loading job i on MA;

$s^B(i)$: the time instant when robot RB starts its ith life cycle by reloading job i from the WIP storage to machine MB;

$u^A(i)$: the time instant when TR starts unloading job i from MA;

$f^B(i)$: the time instant when job i is released at the output AS/RS, that is, the completion time of servicing job i in the cell in the sequence P; $i = j_1, j_2,\ldots,j_n$.

4.2. GRAPH PRESENTATION OF THE TECHNOLOGICAL PROCESS

The above technological process may be portrayed with the help of an oriented graph $G(P)$ (see figure 2(a)), the arcs of which represent operations and precedence relations, and the nodes represent events occurring in time, that is, the moments of starting and finishing the operations. Operations duration are assigned to the corresponding arcs and called the arc lengths.

Node S of the graph denotes the start of processing of jobs and node F denotes the completion of the whole process.

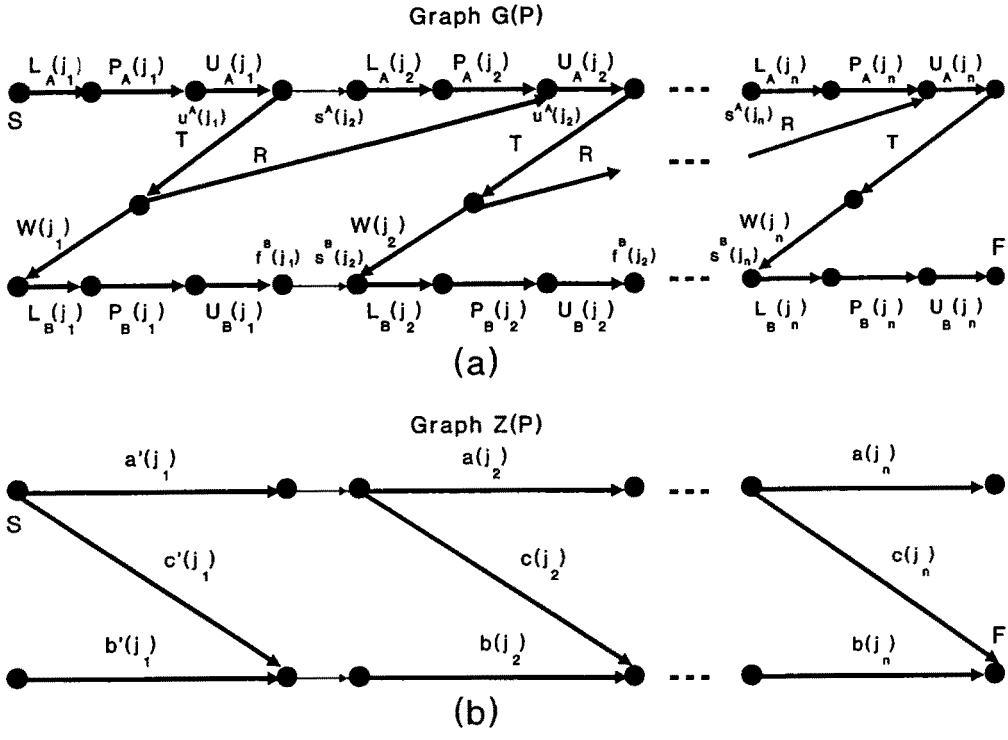

Figure 2. (a) Graph $G(P)$ of the initial problem and (b) graph $Z(P)$ of the equivalent non-uniform (a, b, c) problem.

The thin-line arcs, called dummies, show precedence relations only and do not represent real operations; they have zero length. In particular, the dummies in figure 2 show that the jobs are to be processed on the machines in turn, one after another.

4.3. THE MAKESPAN AND CRITICAL PATHS

Given a sequence of n jobs, $P = (j_1, j_2, \ldots, j_n)$, the $s^A(i)$, $s^B(i)$, $u^A(i)$ and $f^B(i)$ values (where $i = j_1, j_2, \ldots, j_n$) can be computed as follows.

According to rules 1(a) and 1(b),

$$s^A(j_1) = 0, \quad u^A(j_1) = L_A(j_1) + P_A(j_1). \tag{1}$$

According to rules 1, 2(a) and 2(b),

$$\begin{aligned} s^A(j_k) &= u^A(j_{k-1}) + U_A(j_{k-1}), \qquad k = 2, \ldots, n, \\ u^A(j_k) &= s^A(j_k) + \max(T + R, L_A(j_k) + P_A(j_k)), \quad k = 2, \ldots, n. \end{aligned} \tag{2}$$

According to rules 2 and 3,

$$
\begin{aligned}
&s^B(j_1) = u^A(j_1) + U_A(j_1) + T + W(j_1);\\
&f^B(j_1) = s^B(j_1) + L_B(j_1) + P_B(j_1) + U_B(j_1);\\
&s^B(j_k) = \max(s^B(j_{k-1}) + L_B(j_{k-1}) + P_B(j_{k-1}) + U_B(j_{k-1}),\\
&\qquad u^A(j_k) + U_A(j_k) + T + W(j_k));\\
&f^B(j_k) = s^B(j_k) + L_B(j_k) + P_B(j_k) + U_B(j_k), \quad k = 2,\ldots,n.
\end{aligned}
\tag{3}
$$

Note that if the loading/unloading times are negligible for all jobs, i.e. $L_A(i) = U_A(i) = W(i) = 0$ for all i, $1 \le i \le n$, then the above relations (1)–(3) degenerate into those of the Panwalkar [8] model.

Denote by d an arbitrary directed path from S to F in graph $G(P)$ and let $L(d)$ be the length of d, the latter being defined as the sum of lengths of all arcs constituting d.

It is well known in the literature (see, for example, Johnson [3], Kise et al. [4], Levner [7]) that in various scheduling problems the makespan $M(P)$, for any fixed P, equal the length $L_{\max}(P)$ of the critical (longest) path in $G(P)$. The following lemma states that this fact is also valid for the problem considered.

LEMMA

For the considered two-machine scheduling problem,

$$
M(P) = L_{\max}(P) = \max_{d \in \Pi(P)} L(d), \tag{4}
$$

where $\Pi(P)$ denotes the set of all directed paths from S to F in graph $G(P)$.

The proof is by induction on n, using (1)–(3) and the fact that $f^B(j_n) = M(P)$. It is omitted here as being similar to the proofs presented in Kise et al. [4] and Levner [7].

In the next section, we introduce an auxiliary flowshop problem, called "the (a, b, c)-problem", which will serve below to efficiently solve the initial scheduling problem.

5. Analysis of scheduling problems

5.1. THE (a, b, c)-PROBLEM

Consider a two-machine, n-job Johnson-type problem in which processing each job is characterized by three parameters only (without any loading and transportation times whatsoever): $a(i)$, $b(i)$ and $c(i)$. Two of these are the same as in the Johnson [3] problem:

$a(i)$ = processing time of job i on machine MA.

$b(i)$ = processing time of job i on machine MB,

while the third parameter is defined as follows:

$c(i)$ = the minimal time interval that has to elapse between the start of processing job i on machine MA and the finish of processing that job on machine MB, $i = 1,\ldots,n$.

Let us call this problem "the uniform (a, b, c)-problem". Its technological graph G is depicted in figure 3.

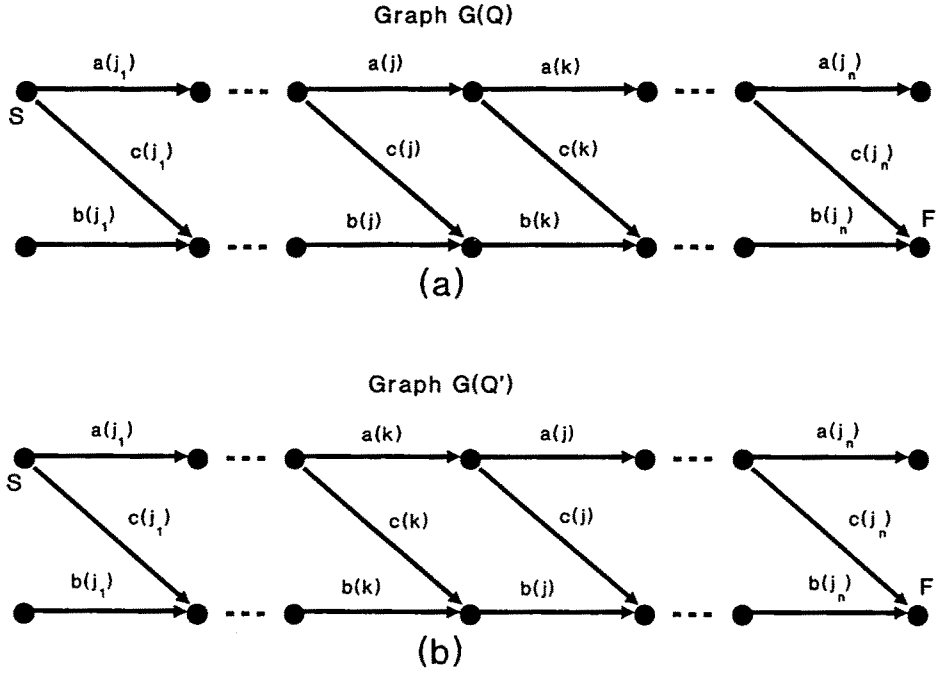

Figure 3. Graph G of the uniform (a, b, c)-problem for two adjacent sequences, (a) Q and (b) Q'.

Together with the uniform (a, b, c)-problem, we shall also treat the so-called "non-uniform (a, b, c)-problem", obtained from the uniform (a, b, c)-problem as follows: for every job i, $1 \le i \le n$, together with three numbers, $a(i)$, $b(i)$, and $c(i)$ defined above, we are given three (generally speaking, other) time parameters $a'(i)$, $b'(i)$, $c'(i)$ such that whenever job i is processed first in any sequence, it acquires new parameters $a'(i)$, $b'(i)$, $c'(i)$, while all other jobs in the sequence possess the same $a(i)$, $b(i)$, and $c(i)$ values as in the initial uniform (a, b, c)-problem. The main idea behind the definition is that in the algorithm below we will try each job with $a'(i)$,

$b'(i)$, and $c'(i)$ values as an initial job, treating all other jobs as elements of a uniform (a, b, c)-problem. Graph $Z(P)$, corresponding to a non-uniform (a, b, c)-problem, is depicted in figure 2(b).

5.2. OPTIMAL SOLUTION

The following theorem motivates introduction of the (a, b, c)-problem.

THEOREM 1

The initial n-job two-machine flowshop problem is reducible to an n-job non-uniform (a, b, c)-problem.

Proof

Consider an arbitrary permutation of n jobs P and its corresponding graph $G(P)$ (see figure 2(a)).

Let us construct an instance of the non-uniform (a, b, c)-problem equivalent to the initial scheduling problem. For this purpose, we replace $G(P)$ by an equivalent graph $Z(P)$ corresponding to an (a, b, c)-problem (see figure 2(b)).

In order to make graphs $G(P)$ and $Z(P)$ equivalent, we define the auxiliary equivalent non-uniform (a, b, c)-problem as follows:

$$\begin{aligned} a'(i) &= L_A(i) + P_A(i) + U_A(i); \\ b'(i) &= L_B(i) + P_B(i) + U_B(i); \\ c'(i) &= a'(i) + b'(i) + T + W(i); \\ a(i) &= \max(L_A(i) + P_A(i);\ T + R) + U_A(i); \\ b(i) &= L_B(i) + P_B(i) + U_B(i); \\ c(i) &= a(i) + b(i) + T + W(i), \quad i = j_1, j_2, \ldots, j_n. \end{aligned} \tag{5}$$

Due to (5), length $L_{\max}(P)$ of the critical path from S to F in graph $G(P)$ equals length $L'_{\max}(P)$ of the critical path from S to F in $Z(P)$ as long as for each i, $i = j_1, \ldots, j_n$, the following is valid:

(1) The length of the path leading from the beginning of the arc (operation) labelled $L_A(i)$ to the end of the arc labelled $U_A(i)$ in $G(P)$ equals the length ($a'(i)$ or ($a(i)$) of the corresponding arc in $Z(P)$.

(2) The length of the path leading from the beginning of arc (operation) $L_B(i)$ to the end of arc $U_B(i)$ in $G(P)$ equals the length ($b'(i)$ or $b(i)$) of the corresponding arc in $Z(P)$.

(3) The length of the path leading from the beginning of arc (operation) $L_A(i)$ to the end of arc $U_B(i)$ in $G(P)$ equals the length ($c'(i)$ or $c(i)$) of the corresponding arc in $Z(P)$ (see figures 2(a, b)).

Then, taking (4) into account, we obtain that the makespan $M(P)$ is the same for both problems. Therefore, the minimal makespan for the described (a, b, c)-problem will provide, at the same time, the minimal makespan for the original scheduling problem. □

The following theorem shows that the n-job uniform (a, b, c)-problem is solvable in polynomial time.

THEOREM 2

A sequence for the two-machine uniform (a, b, c)-problem is optimal whenever first the jobs i for which $a(i) \le b(i)$ are processed, arranged so that the $c(i) - b(i)$ values are ordered from the smallest to the largest, and these jobs are followed by the remaining ones (i.e. such that $a(i) > b(i)$) arranged so that the $c(i) - a(i)$ values are ordered from the largest to the smallest.

Proof

Let P^* be the sequence in which jobs are arranged according to the rule formulated in the theorem, and assume that it is not optimal. Then there exists an optimal sequence, say Q, containing two jobs, say j and k, with j immediately preceding k, for which the condition of the theorem does not hold.

The latter circumstance is possible in the following cases:

(i) $a(j) \le b(j)$, $a(k) \le b(k)$, and $c(j) - b(j) > c(k) - b(k)$;

(ii) $a(j) > b(j)$, $a(k) > b(k)$, and $c(j) - a(j) < c(k) - a(k)$;

(iii) $a(j) > b(j)$, and $a(k) \le b(k)$.

Due to the above assumption on the optimality of Q,

$$M(Q) < M(P^*). \tag{6}$$

Consider the sequence $Q' = (\ldots k, j \ldots)$, obtained from Q by interchanging jobs j and k. We intend to show that Q' is not worse that Q, that is $M(Q') \le M(Q)$. For this purpose, we will use equation (4) as related to the considered (a, b, c)-problem and to its graph depicted in figure 3.

Compare the critical paths in graphs $G(Q)$ and $G(Q')$ (see figure 3), and show that for all the above cases (i) – (iii), the critical path in $G(Q')$ is not larger than the critical path in $G(Q)$, or

$$\max\big(c(k) + b(j), a(k) + c(j)\big) \le \max\big(c(j) + b(k), a(j) + c(k)\big). \tag{7}$$

1. Let case (i) take place. Since $c(k) + b(j) < c(j) + b(k)$, and $a(j) \le b(j)$, then $a(j) + c(k) \le c(k) + b(j) < c(j) + b(k)$, and therefore $\max(c(j) + b(k), a(j) + c(k)) = c(j) + b(k)$.

Consider two subcases:

(1a) If $\max(c(k) + b(j), a(k) + c(j)) = c(k) + b(j)$ (that is, $c(k) + b(j) \ge a(k) + c(j)$), then by virtue of $c(j) - b(j) > c(k) - b(k)$, we have that $c(j) + b(k) > c(k) + b(j)$ and, therefore, (7) holds.

(1b) If $\max(c(k) + b(j), a(k) + c(j)) = a(k) + c(j)$ (that is, $c(k) + b(j) \le a(k) + c(j)$), then by virtue of $a(k) \le b(k)$, we have that $c(j) + b(k) \ge a(k) + c(j)$ and, therefore, in case (i) inequality (7) holds.

2. Let case (ii) take place. Since $a(j) + c(k) > a(k) + c(j)$ and $a(j) > b(j)$, then $a(j) + c(k) > a(k) + c(j) > c(j) + b(k)$, and therefore $\max(c(j) + b(k), a(j) + c(k)) = a(j) + c(k)$. Consider two subcases:

(2a) If $\max(c(k) + b(j), a(k) + c(j)) = c(k) + b(j)$ then by virtue of $a(j) > b(j)$, we have that $a(j) + c(k) > c(k) + b(j)$ and, therefore, inequality (7) holds.

(2b) If $\max(c(k) + b(j), a(k) + c(j)) = a(k) + c(j)$, then by virtue of $c(j) - a(j) < c(k) - a(k)$, we have that $a(j) + c(k) > a(k) + c(j)$ and, therefore, in case (ii) inequality (7) holds.

3. Let case (iii) take place. Consider four subcases:

(3a) $\max(c(k) + b(j), a(k) + c(j)) = c(k) + b(j)$, and
$\max(c(j) + b(k), a(j) + c(k)) = c(j) + b(k)$.

In this subcase, by virtue of $a(j) > b(j)$, we have that $c(j) + b(k) \ge a(j) + c(k) > b(j) + c(k)$, that is, (7) holds.

(3b) $\max(c(k) + b(j), a(k) + c(j)) = c(k) + b(j)$, and
$\max(c(j) + b(k), a(j) + c(k)) = a(j) + c(k)$.

In this subcase, by virtue of $a(j) > b(j)$, we have that $c(k) + b(j) < c(k) + a(j)$, that is, (7) holds.

(3c) $\max(c(k) + b(j), a(k) + c(j)) = a(k) + c(j)$, and
$\max(c(j) + b(k), a(j) + c(k)) = c(j) + b(k)$.

By virtue of $a(k) \le b(k)$, we have that $a(k) + c(j) \le b(k) + c(j)$, that is, (7) holds.

(3d) $\max(c(k) + b(j), a(k) + c(j)) = a(k) + c(j)$, and
$\max(c(j) + b(k), a(j) + c(k)) = a(j) + c(k)$.

By virtue of $a(k) \le b(k)$, we have that $a(j) + c(k) \ge c(j) + b(k) \ge c(j) + a(k)$, that is, in case (iii) inequality (7) holds.

By virtue of (7), we can conclude that the critical path in $G(Q')$ is not larger than the critical path in $G(Q)$, and taking (4) into account, we derive that the makespan cannot increase in reversing the order of jobs j and k: $M(Q') \le M(Q)$.

If $Q' = P^*$, we have a contradiction to (6).

If $Q' \neq P^*$, we may find in Q' a new pair of jobs, say l and m, with l immediately preceding m, for which the condition of the theorem does not hold. Consider the sequence $Q'' = (\ldots m, l \ldots)$, obtained from Q' by interchanging jobs l and m. Applying again the above arguments, we derive: $M(Q'') \le M(Q')$. After a finite number of such pairwise interchanges, sequence P^* can be obtained from Q, with $M(P^*) \le \ldots \le M(Q'') \le M(Q') \le M(Q)$. This is a contradiction to (6), which completes the proof. □

The following corollaries can be easily derived.

COROLLARY 1

The n-job uniform (a, b, c)-problem is solvable in $O(n \log n)$ time.

COROLLARY 2

Consider the n-job non-uniform (a, b, c)-problem defined above, and let a certain job, say i, be specified as the first one in any schedule. Then, if the remaining jobs $\{1,\ldots,n\} \setminus i$ are arranged in the same order as they appear in the optimal solution of the initial n-job uniform (a, b, c)-problem (that is, according to the rule stated in theorem 2), we shall obtain an optimal sequence for the non-uniform (a, b, c)-problem considered.

Based on the above analysis, we can now describe an algorithm for solving the initial scheduling problem.

6. Sequencing algorithm

6.1. ALGORITHM DESCRIPTION

Input: For all i, $1 \le i \le n$, $L_A(i)$, $P_A(i)$, $U_A(i)$, $L_B(i)$, $P_B(i)$, $U_B(i)$, T, R, and $W(i)$.

Output: An optimal solution P^* of the initial n-job, two-machine problem, and the minimal makespan $M^* = M(P^*) = \min_P M(P)$.

Step 1. [Define the auxiliary uniform and non-uniform (a, b, c)-problems].
Let

$$a'(i) = L_A(i) + P_A(i) + U_A(i); \quad b'(i) = L_B(i) + P_B(i) + U_B(i);$$

$$c'(i) = a'(i) + b'(i) + T + W(i); \quad a(i) = \max(L_A(i) + P_A(i), T + R) + U_A(i);$$

$$b(i) = L_B(i) + P_B(i) + U_B(i); \quad c(i) = a(i) + b(i) + T + W(i); \quad i = 1,\ldots,n.$$

Step 2. [Find an optimal sequence for the uniform (a, b, c)-problem].

Arrange first the jobs i for which $a(i) \le b(i)$, in non-decreasing order of $c(i) - b(i)$ values, and then arrange the remaining jobs in non-increasing order of $c(i) - a(i)$ values.

Denote the sequence obtained by $P^0 = (j_1^0, j_2^0, \ldots, j_n^0)$; $M^0 = M(P^0)$.

Step 3. [Try each job i as an initial, with all the remaining jobs being arranged as in P^0].

Move job $i = j_k^0$ $(2 \le k \le n)$ in P^0 to the first position and shift the first $k-1$ jobs one position to the right; denote the sequence obtained by P_k, $2 \le k \le n$. Take P^0 as P_1. Calculate the makespan $M_k = M(P_k)$ for each of the obtained sequences P_k, $1 \le k \le n$.

Step 4. [Find the minimal makespan and an optimal sequence].

Find $M^* = \min_{1 \le k \le n} M(P_k)$; then M^* is the minimal makespan and $P^* = \arg\min_k M(P_k)$ is an optimal sequence.

6.2. ALGORITHM COMPLEXITY

Let us verify that the algorithm described above runs in $O(n \log n)$ time.

In step 1, in order to determine the auxiliary uniform and non-uniform (a, b, c)-problems, we need $O(n)$ operations.

In step 2, in order to find the optimal sequence P^0 for the uniform (a, b, c)-problem, we need $O(n \log n)$ time (see corollary 1).

Let us show that $O(n)$ time is sufficient to run step 3.

Given P^0, let us introduce, for each i, $i = 1, \ldots, n$, six auxiliary parameters, which we call "potentials". In order to simplify the notation, let us re-enumerate the jobs so that $P^0 = (1, 2, \ldots, n)$. The potentials for this P^0 are defined as follows:

$$p^{(1)}(k) = p^{(2)}(k) = 0, \qquad \text{for } k = 1;$$

$$p^{(1)}(k) = \sum_{i=1}^{k-1} a(i); \quad p^{(2)}(k) = \sum_{i=1}^{k-1} b(i), \qquad \text{for } 2 \le k \le n;$$

$p^{(3)}(k)$ = the makespan (i.e. the completion time) to fulfill jobs $1, 2, \ldots, k-1$ on two machines in this order, $2 \le k \le n$;

$$p^{(4)}(k) = p^{(5)}(k) = 0, \qquad \text{for } k = n;$$

$$p^{(4)}(k) = \sum_{i=k+1}^{n} a(i); \quad p^{(5)}(k) = \sum_{i=k+1}^{n} b(i), \quad \text{for } 1 \le k \le n-1;$$

$p^{(6)}(k)$ = the makespan to fulfill jobs $k+1, k+2, \ldots, n$ on two machines in this order, $k = 1, \ldots, n-1$.

Potentials $p^{(3)}(k)$ and $p^{(6)}(k)$ can be calculated recursively as follows:

$$p^{(3)}(1) = 0,\ p^{(3)}(2) = \max(b(1), c(1));$$

$$p^{(3)}(k) = \max(p^{(1)}(k) + c(k-1),\ p^{(3)}(k-2) + b(k-1)),\ \text{for } k = 3,\ldots n;$$

$$p^{(6)}(n) = 0,\ p^{(6)}(n-1) = \max(a(n), c(n));$$

$$p^{(6)}(k) = \max(p^{(5)}(k) + c(k+1),\ p^{(6)}(k+2) + a(k+1)),\ \text{for } k = n-2, n-3, \ldots, 1.$$

The above formulas show that $O(n)$ time is sufficient for finding all the potentials. The main idea behind the definition of potentials is that, for each k, $1 \le k \le n$, they permit us to determine the makespan $M(P_k)$ in $O(1)$ time.

Indeed, consider sequence P_k in the corresponding non-uniform (a, b, c)-problem treated in step 3, and aggregate the jobs appearing before job $i = k$ $(k = 2,\ldots, n)$ in P^0 into an equivalent "job" J_k having the following a, b, c-parameters:

$$a(J_k) = p^{(1)}(k);\ b(J_k) = p^{(2)}(k);\ c(J_k) = p^{(3)}(k).$$

In a similar way, aggregate the jobs appearing after job $i = k$ $(k = 1,\ldots, n-1)$ in P^0 into an equivalent "job" J^k having the following parameters:

$$a(J^k) = p^{(4)}(k);\ b(J^k) = p^{(5)}(k);\ c(J^k) = p^{(6)}(k).$$

Consider now a sequence P_k as a sequence of three jobs: $i = k$ (standing in the first position), J_k and J^k: $P_k = (k, J_k, J^k)$.

The makespan $M(P_k)$ due to (4), for any fixed $i = k$, can be calculated in $O(1)$ time: $M(P_k) = \max(a'(k) + a(J_k) + a(J^k),\ a'(k) + a(J_k) + c(J^k),\ a'(k) + c(J_k) + b(J^k),\ c'(k) + b(J_k) + b(J^k),\ b'(k) + b(J_k) + b(J^k))$.

Therefore, in order to calculate all n makespan values in step 3, we need $O(n)$ operations.

Step 4 requires $O(n)$ operations.

Hence, $O(n \log n)$ time is sufficient to optimally solve the scheduling problem considered.

Remark

As we have mentioned above, the problem considered is a generalization of the Panwalkar problem; hence, incorporating the above potentials into the (quadratic) Panwalkar [8] algorithm, we can derive a modification of the latter running in $O(n \log n)$ time.

7. A numerical example

In the following example, travel times T and R equal 5 units and loading times $L_A(i)$, $L_B(i)$, $U_B(i)$ are zero for all i, $1 \le i \le n$. The other parameters (processing

Table 1

Input parameters of the initial problem.

Job	$P_A(i)$	$P_B(i)$	$W(i)$	$U_A(i)$
1	12	24	18	23
2	20	33	8	7
3	15	35	10	3
4	5	10	12	15
5	1	5	3	1
6	18	13	17	12
7	10	25	11	13

Table 2

Parameters of the (a, b, c)-problems.

Job	$a(i)$	$b(i)$	$c(i)$	$a'(i)$	$b'(i)$	$c'(i)$
1	35	24	82	35	24	82
2	27	33	73	27	33	73
3	18	35	68	18	35	68
4	25	10	52	20	10	47
5	11	5	24	2	5	15
6	30	13	65	30	13	65
7	23	25	64	23	25	64

times $P_A(i)$ and $P_B(i)$, placement time $W(i)$, and unloading time $U_A(i)$) are presented in table 1.

Step 1 of the algorithm will give the parameters of the uniform and non-uniform (a, b, c)-problems shown in table 2.

Step 2 gives an optimal solution of the uniform (a, b, c)-problem, $P^0 = (3, 7, 2, 1, 6, 4, 5)$.

Steps 3 and 4 result in the optimal solution of the initial problem, $P^* = (5, 3, 7, 2, 1, 6, 4)$, with the minimal makespan $M(P^*) = 187$.

8. Concluding remarks

Using a graph representation of the two-machine flowshop scheduling problem, we have incorporated non-zero loading/unloading times into a model of a robotic cell and derived an optimal scheduling algorithm running in $O(n \log n)$ time.

The model discussed in the paper may be generalized in several practice-oriented directions: (a) each job may be replaced by a PERT network of interrelated

operations; (b) the transporting robot may be used for delivering tools and fixtures, with delivery times being job dependent; (c) more than one transporting robot and more than two machines may be treated; (d) the limited capacity of the output-WIP AS/RS may not be sufficient for storing all the required WIP jobs. To the best of our knowledge, under assumptions (a) and (b) the scheduling problem retains the property of polynomial solvability, while in cases (c) and (d) it becomes NP-hard.

References

[1] A.S. Belen'kii and E.V. Levner, Scheduling models and methods in optimal freight transportation planning, Auto. Remote Contr. 10(1991)1–56.

[2] T. Blazewicz, G. Finke, R. Haupt and G. Schmidt, New trends in machine scheduling, Euro. J. Oper. Res. 37(1988)303–317.

[3] S.M. Johnson, Optimal two- and three-stage production scheduling with setup times included, Naval. Res. Log. Quart. 1(1954)61–68.

[4] H. Kise, T. Shioyama and T. Ibaraki, Automated two-machine flowshop scheduling: A solvable case, IIE Trans. 23(1991)10–16.

[5] A. Kusiak, Scheduling flexible machining and assembly systems, in: *Flexible Manufacturing Systems. Operations Research Models and Applications*, ed. K.E. Stecke and R. Suri (Elsevier, New York, 1986) pp. 521–532.

[6] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity, in: *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science, Vol. 4, ed. S. Graves, A. Rinnooy Kan and P. Zipkin (North-Holland, New York, 1993).

[7] E.V. Levner, Optimal planning of part's machining on a number of machines, Auto. Remote Contr. 11(1969)1972–1979.

[8] S.S. Panwalkar, Scheduling of a two-machine flowshop with travel time between machines, J. Oper. Res. Soc. 42(1991)609–613.

[9] R.E. Stecke and R. Suri (eds.), *Flexible Manufacturing Systems: Operations Research Models and Applications* (Elsevier, New York, 1986).

[10] H.I. Stern and G. Vitner, Scheduling parts in a combined production–transportation work cell, J. Oper. Res. Soc. 41(1990)625–632.