10.1515/acsc-2017-0021

Archives of Control Sciences
Volume 27(LXIII), 2017
No. 2, *pages 323–330*

An exact block algorithm for no-idle RPQ problem

JAROSLAW PEMPERA

In the work a single-machine scheduling problem is being considered, in which all tasks have a fixed availability (release) and delivery time. In the analyzed variant no-idle time is allowed on a machine. The purpose of optimization is to determine such order of tasks that minimizes the makespan, i.e. the time of execution of all the tasks. There is also a number of properties of the problem presented, in particular there are formulated block eliminating properties for no-idle constraint. There was an exact B&B algorithm based on the block properties proposed.

Key words: scheduling, single machine, no-idle, B&B algorithm.

1. Introduction

In a single-machine problem with jobs' release time and delivery time (RPQ) there must be a number of tasks executed on one machine. No idle time means that since the commencement of the first task the machine must perform the next task without interruption until the completion of the last task. Therefore, there must be a schedule of tasks minimizing delivery time of all the tasks determined.

No-idle constraint occurs in production systems, in which there are thermal or chemical reaction processes. In such systems, machine downtimes can not only cause failure of machinery, deterioration in the quality of products but can also generate additional costs related to energy expenditure. No-idle constraint can also be used in planning of outsourcing services, in which costs are dependent on the length of the contracts or in case of renting of expensive equipment for the execution of orders. Undoubtedly, both RPQ and RPQ no-idle problems are NP-hard. In case of RPQ problem there are efficient algorithms based on Branch-and-Bound (B&B) method proposed by Carlier [1] and Grabowski et al. [4]. In the first algorithm the node in the B&B search tree method is created in the time $O(n \ln n)$ using the algorithm Schrage [10], whereas in the second algorithm the nodes are created on the basis of the concept of a block task. Due to short duration of execution of these algorithms (a few seconds for 50–10000 tasks) they can be used to estimate lower bounds for the construction of multi-machine problems with

The Author is with Department of Automatics, Mechatronics and Control Systems, Faculty of Electronics, Wrocław University of Technology, Janiszewskiego str. 11-17, 50-372 Wrocław, Poland, e-mail: jaroslaw.pempera@pwr.edu.pl

Received 9.11.2016. Revised 16.05.2017.

the criterion of minimizing the completion time of all tasks such as: flow shop problems, job shop problems, flexible flow shop and job shop problems.

Construction of algorithms for simple scheduling problems is important in scientific context because the properties of these problems can be used in the construction of exact and heuristic algorithms for multi-machine scheduling problems. Discovered by Grabowski et al. block properties have been successfully used not only in the construction of exact algorithms for flow shop and job shop problems (the exact solution of small size), but also in the construction of one of the most effective heuristic algorithms for the following problems: flow shop [3, 7], job shop [8], no-store (blocking) flow shop [9] etc.

Study on task scheduling with no-idle constraint on machines mainly relate to the flow shop problem. There were proposed exact algorithms [11] and heuristic algorithms based on the construction [2], [5] and local search methods [6]. The paper presents the new properties of the problem with no-idle time on machine. Due to the significant similarity to the block properties of RPQ problem they will also be called block properties. Using these properties there has been an exact algorithm proposed based on the B&B method.

2. Problem description

On machine there must be executed n task from the set $J = \{1, \ldots, n\}$. Each task has a known execution time $p_j > 0$, $j \in J$ and cannot start earlier than the release time $r_j \geqslant 0$. The machine cannot perform more than one task at a time. The tasks on the machine must be carried out without any interruption. Once the task is completed the finished product is delivered to the customer in time $q_j \geqslant 0$. Let S_j, C_j and D_j, be respectively the starting moment, the completion time and delivery moment of task j, $j \in J$. There must be determined the tasks schedule minimizing the moment of all the tasks delivery.

Let π be a permutation defined on the set J describing the sequence of tasks execution on one machine. Feasible schedule for tasks execution in the order of π must meet the following restrictions:

$$S_j \geqslant r_j \quad \text{for } j \in J, \tag{1}$$

$$C_j = S_j + p_j \quad \text{for } j \in J, \tag{2}$$

$$D_j = C_j + q_j \quad \text{for } j \in J, \tag{3}$$

$$S_{\pi(j)} = C_{\pi(j-1)} \quad \text{for } j = 2, \ldots, n. \tag{4}$$

Constraints (1–3) are obvious, while equation (4) imposes the performance of tasks on the machine in order π with no idle time.

Let $D_{\max} = \max_{j \in J} D_j$ be the moment of all the tasks delivery. We want to find the sequence of tasks on the machine π^*, such that

$$D_{\max}(\pi^*) = \min_{\pi \in \Pi} D_{\max}(\pi), \tag{5}$$

where Π is the set of all permutations defined on the set J.

2.1. Example

There must be $n = 10$ tasks performed on a machine. The release times, execution times and delivery times of tasks are summarized in Table 1. In Fig. 1 there are three schedules for tasks execution presented in the Gantt chart form. Schedules (A) and (B) have been designated for the permutations generated by the use of Schrage algorithm (A-RPQ, B-RPQ no-idle) while schedule (C) is the optimal schedule for the RPQ no-idle problem. The delivery moment of all tasks is: (A)–42, (B)–46, (C)–43.

Table 1: Test data

Parametr	1	2	3	4	5	6	7	8	9	10
release time – r_i	9	16	15	27	6	4	26	12	3	11
execution time – p_i	4	1	1	2	1	1	2	4	1	1
delivery time – q_i	26	13	22	12	28	1	5	17	20	0

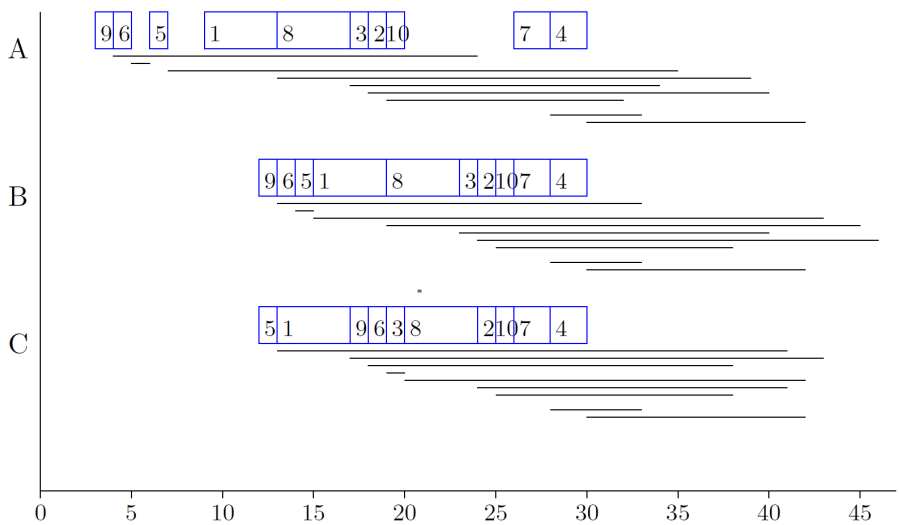

Figure 1: Gantt chart

3. Graph model

Before the presentation of graph model it must be noted that the equation (4) can be replaced by two inequalities:

$$S_{\pi(j)} \geqslant S_{\pi(j-1)} + p_{\pi(j-1)} \quad \text{dla } j = 2,\ldots,n, \tag{6}$$

$$S_{\pi(j-1)} \geqslant S_{\pi(j)} - p_{\pi(j-1)} \quad \text{dla } j = 2,\ldots,n. \tag{7}$$

For fixed processing order π, there is a graph $G(\pi) = (V, E(\pi))$ defined with the set of nodes V and a set of directed arcs $E(\pi)$. A set of nodes $V = J \cup \{s,t\}$ consists of n nodes representing tasks and two fictitious nodes s (initial, source) and t (final, target). The node representing the task j is burdened with weight equal to the duration of the task p_j, $j \in J$, while the fictitious nodes with weight equal to zero.

A set of arcs $E(\pi)$ consists of four subsets of arcs:

$$E^s = \{(s,j) : j \in J\}, \tag{8}$$

arc $(s,j) \in E^s$ corresponds to the constraint (1) and has weight r_j,

$$E^t = \{(j,t) : j \in J\}, \tag{9}$$

arc $(j,t) \in E^t$ corresponds to the constraint (3) and has weight q_j,

$$E^1(\pi) = \{(\pi(j-1),\pi(j)) : j = 2,\ldots n\}, \tag{10}$$

arc $(\pi(j-1),\pi(j)) \in E^1(\pi)$ corresponds to the constraint (6) and has weight 0. Arcs from the set $E^1(\pi)$ corresponds to the processing order of tasks π and will be called (in short) machine arcs. The last subset constitute arcs resulting from the no-idle constraint

$$E^2(\pi) = \{(\pi(j),\pi(j-1)) : j = 2,\ldots n\}, \tag{11}$$

arc $(\pi(j),\pi(j-1)) \in E^2(\pi)$ corresponds to the constraint (7) and has weight $-p_{\pi(j)} - p_{\pi(j-1)}$, which results from transformation of (7) to

$$S_{\pi(j-1)} \geqslant S_{\pi(j)} + p_{\pi(j)} - p_{\pi(j)} - p_{\pi(j-1)} \quad \text{for } j = 2,\ldots,n. \tag{12}$$

Propertie 1 *For fixed processing order of tasks execution π on a single machine, the earliest starting moment for the task j, $j \in J$ is equal to the length of the longest path to the node representing this task (without weight of node) in the graph $G(\pi)$.*

Propertie 2 *For fixed processing order π, the earliest delivery moment of all the tasks is equal to the length of the longest path to a node t in the graph $G(\pi)$.*

It follows from the construction of a graph $G(\pi)$ that every path begins in the source node s, what is more, the longest path in the graph $G(\pi)$ to node t is also the longest path in this graph. This path will be called *critical path*.

The set of arcs of the graph $G(\pi)$ consists of arcs with negative weighs, therefore, determination of the length of the longest paths requires the use of the Bellman-Forda algorithm with complexity of $|V| \cdot |E|$.

Propertie 3 *The lengths of the longest paths in $G(\pi)$ can be determined in time $O(n)$.*

Proof. For the proof, it is enough to note that the lengths of the longest paths have a fixed value after execution of a relaxation of arcs in the following order: (*i*) arcs from the set E^s, (*ii*) arcs from the set $E^1(\pi)$ in the order of $(\pi(1),\pi(2)),\ldots,(\pi(n-1),\pi(n))$, (*iii*) arcs from the set $E^2(\pi)$ in the order of $(\pi(n),\pi(n-1)),\ldots,(\pi(2),\pi(1))$, (*iv*) arcs from the set E^t.

Any critical path in the graph $G(\pi)$ begins with an arc belonging to E^s and ending with an arc belonging to E^t. Therefore, it will be described with the pair (a,b), where $(s,a) \in E^s$ i $(b,t) \in E^t$. For $a = b$ the solution π is the optimal solution. Let us consider two cases: (*i*) $a < b$ and (*ii*) $a > b$.

In case of (*i*) a sequence $B_{a,b} = (\pi(a),\pi(a+1),\ldots,\pi(b))$ will be called *block of tasks*. The block theorem [4] can be formulated as follows:

Propertie 4 *If $B_{a,b}$ is a block in π and $\beta \in \Pi$ is any permutation such that $D_{\max}(\beta) < D_{\max}(\pi)$, there is a task $\pi(k) \in \{\pi(a+1),\ldots,\pi(b)\}$ (or $\pi(k) \in \{\pi(a),\ldots,\pi(b-1)\}$) such that in the permutation β task $\pi(k)$ precedes all other tasks in a block (it is processed after all other tasks in a block).*

In case (*ii*) the sequence $B_{a,b} = (\pi(a),\ldots,\pi(n),\pi(1),\ldots,\pi(b))$ will be called *i-block of tasks*. In a way analogous to 4 property, the following property can be proved for the *i*-block $B_{a,b}$:

Propertie 5 *If $B_{a,b}$ is i-block in π and $\beta \in \Pi$ is any permutation such that $D_{\max}(\beta) < D_{\max}(\pi)$, there is a task $\pi(k) \in \{\pi(a+1),\ldots,\pi(n)\}$ (or $\pi(k) \in \{\pi(1),\ldots,\pi(b-1)\}$) such that in the permutation β task $\pi(k)$ precedes all other tasks in the set $\{\pi(a),\ldots,\pi(n)\}$ (it is processed after all other tasks from the set $\{\pi(1),\ldots,\pi(b)\}$).*

4. Exact block algorithm

With each node of Branch and Bound method there is associated a permutation determining the order of tasks π, block or *i*-block of tasks $B_{a,b}$ and the relation of partial order of R. The relation R implies a constraint " if $(i,j) \in R$, then task j can begin only after the task i is completed". The space of solutions is divided into subspaces (branching) on the basis of tasks in the block (or *i*-block) $B_{a,b}$.

A new node in a search tree of B&B algorithm is created by generating a new permutation and adding constraints to relation R. For $k = a+1,\ldots,x$ ($x = b$ for the block and $x = n$ for *i*-block) a new permutation is generated by shifting task $\pi(k)$ on position a in

π. To the relation R there are added constraints $\{(\pi(k),\pi(j)) : j \neq k,\ j = a,\ldots,x\}$, which means that the task $\pi(k)$ must be executed before all other tasks from the set $\{a,\ldots,x\}$. But for $k = x,\ldots,b-1$ ($x = a$ for the block and $x = 1$ for i-block) a new permutation is generated by shifting task $\pi(k)$ on position b w π. To the relation R there are added constraints $\{(\pi(j),\pi(k)) : j \neq k,\ j = x,\ldots,b\}$ which means that the task $\pi(k)$ must be executed after all other tasks from the set $\{x,\ldots,b\}$.

The search node of tree is cut off if the relation R is contradictory. Similarly as in the exact block algorithm for RPQ problem, the node generated by shifting task $\pi(k)$ is cut off, if $r_{\pi(k)} > r_{\pi(a)}$ in case of inserting into the position a and if $q_{\pi(k)} > q_{\pi(b)}$ in case of inserting into the position b.

In Fig. 2 there was the course of the proposed algorithm for data from example presented. Each node is described in a separate rectangle, in which there is a permutation, node number, and $D_{\max}$ (in the lower right corner) presented. Tasks forming block (i-blok) were linked with the use of thick line, in addition, the tasks $\pi(a)$ and $\pi(b)$ were highlighted in bold.

Transfers of tasks resulting from the division strategy were illustrated by arcs, wherein the arcs leading to the nodes were indicated by dashed lines. The initial permutation π (Node 1) was generated by Schrage algorithm. In the permutations π there is i-block $B_{9,6}$. In the case of tasks 4 and 5 there are conditions $r_4 > r_7$ and $q_5 > q_3$ and $q_1 > q_3$ fulfilled, therefore the corresponding nodes will be cut off. Finally, as a result of the division there are three child nodes created by shifting: (*i*) task 8 to position 6 (node 2), (*ii*) task 6 to position 6 (node 7), (*iii*) task 9 to position 6 (node 12). Child nodes are connected to a parent node with an arc with the assigned number of a task shifted as a result of generating of a new permutations. In any child the number of restrictions remembered in relation of partial order increases. It is the main mechanism of cutting off child nodes. This can be seen on the example of node 11, wherein the child nodes generated by shifting tasks 5, 1, 3 and 8 were cut off because of the restrictions added while creating node 10, i.e. shifting tasks 9, which generates restriction: task 9 must be performed after tasks 5,1,8 and 3. The algorithm found the optimal solution in node 4 with a value of 43 and generated a total of 14 nodes (permutations) in reference to 10! all possible. There were tests conducted on the effectiveness of the block algorithm on multiple instances of data generated at random of size from 10 to 80 tasks. In most cases the number of nodes generated by algorithm did not exceed a few dozen and duration of action did not exceed a few hundred milliseconds. In addition, during the operation of the algorithm, i-blocks were generated relatively rarely. This means that for many instances of RPQ problem, in exact solution the removal of idle-time of machine does not increase the time of delivery of all tasks.

5. Summary

In the work, for single-machine scheduling problem with release and delivery dates with no-idle time on machines there was the notion of i-block of tasks formulated which

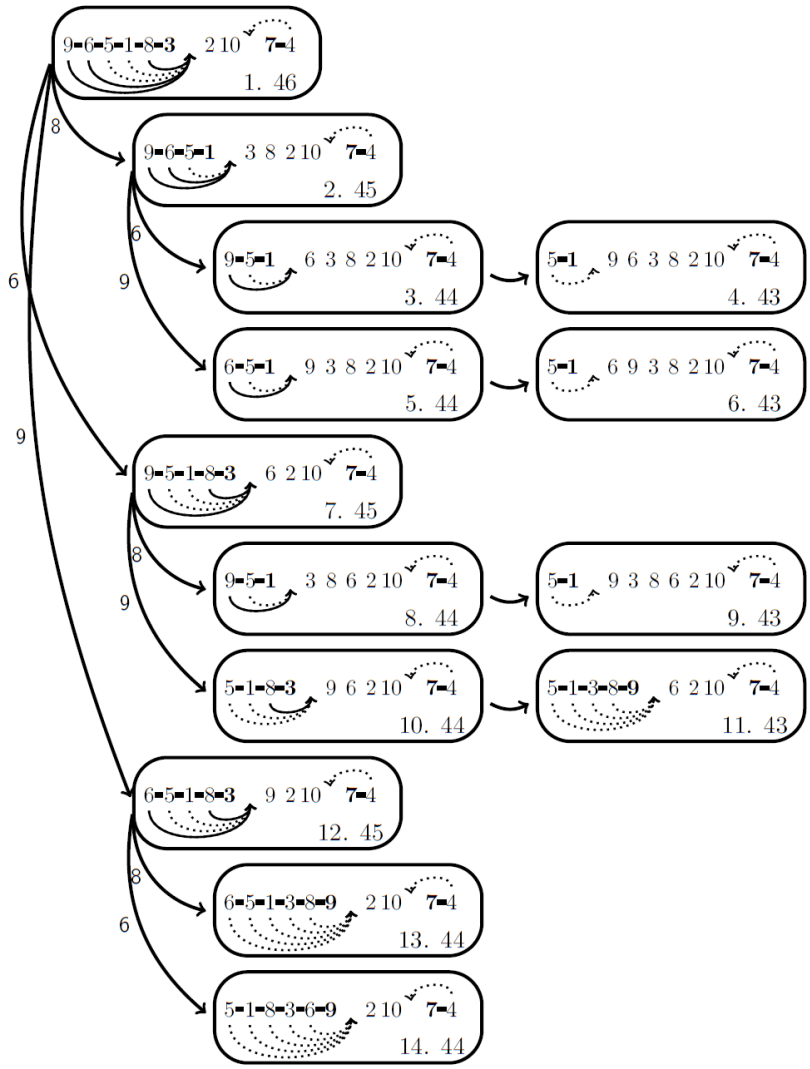

Figure 2: The calculation process for the data from example

is an extension of the well-known block of tasks. There was an exact algorithm proposed based on B&B method in which block properties have been used in the branch and bound strategy. It follows from the experimental studies on algorithms that solution to a significant number of instances of size of dozens of tasks takes no more than 1 second on a PC. The author of the paper believes that, based on *i*-blocks of tasks there can be efficient local search algorithms constricted for multi-machine tasks scheduling problems.

References

[1] J. CARLIER: The one-machine sequencing problem. *European J. of Operational Research*, **11**(1), (1982), 42-47.

[2] O. CEPEK, M. OKADA and M. VLACH: Note: On the two-machine no-idle flow-shop problem. *Naval Research Logistics*, **47**(4), (2000), 353-358.

[3] J. GRABOWSKI and J. PEMPERA: New block properties for the permutation flow shop problem with application in tabu search. *The Journal of the Operational Research Society*, **52**(2), (2001), 210-220.

[4] J. GRABOWSKI, E. NOWICKI and S. ZDRZAŞKA: A block approach for single-machine scheduling with release dates and due dates. *European J. of Operational Research*, **26** (1986), 278-285.

[5] P.J. KALCZYNSKI and J. KAMBUROWSKI: A heuristic for minimizing the makespan in no-idle permutation flow shops. *Computers & Industrial Engineering*, **45** (2005), 146-154.

[6] B. LIU, L. WANG and Y. JIN: An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. e, Computers & Operations Research, **35**(9), (2008), 2791-2806.

[7] E. NOWICKI and C. SMUTNICKI: A fast taboo search algorithm for the job shop problem. *Management Science*, **24**(5), (1996), 530-534.

[8] E. NOWICKI and C. SMUTNICKI: A fast tabu search algorithm for the permutation flow-shop problem. *European J. of Operational Research*, **91**(1), (1996), 160-175.

[9] Q.-K. PAN and L. WANG: No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. In press at *Int. J. of Advanced Manufacturing Technology*.

[10] L. SCHRAGE: Obtaining optimal solution to resource constrained network scheduling problem. Unpublished manuscript, 1971.

[11] P. VACHAJITPAN: Job sequencing with continuous machine operation. *Computers & Industrial Engineering*, **6**(3), (1982), 255-259.

[12] R.E. KALMAN: Mathematical description of linear dynamical system. *SIAM J. Control*, **1**(2), (1963), 152-192.

[13] F.C. SHWEPPE: Uncertain Dynamic Systems. Prentice-Hall, Englewood Cliffs, N.J., 1970.