This may be the author's version of a work that was submitted/accepted for publication in the following source:

Yan, Pengyu, Liu, Shi-Qiang, Yang, Cheng-Hu, & Masoud, Mahmoud (2019)
A comparative study on three graph-based constructive algorithms for multi-stage scheduling with blocking.
Journal of Industrial and Management Optimization, 15(1), pp. 221-233.

This file was downloaded from: https://eprints.qut.edu.au/118192/

© 2018 American Institute of Mathematical Sciences

This work is covered by copyright. Unless the document is being made available under a Creative Commons Licence, you must assume that re-use is limited to personal use and that permission from the copyright owner must be obtained for all other uses. If the document is available under a Creative Commons License (or other specified license) then refer to the Licence for details of permitted re-use. It is a condition of access that users recognise and abide by the legal requirements associated with these rights. If you believe that this work infringes copyright please provide details by email to qut.copyright@qut.edu.au

Notice: Please note that this document may not be the Version of Record (i.e. published version) of the work. Author manuscript versions (as Submitted for peer review or as Accepted for publication after peer review) can be identified by an absence of publisher branding and/or typeset appearance. If there is any doubt, please refer to the published source.

https://doi.org/10.3934/jimo.2018040
A COMPARATIVE STUDY ON THREE GRAPH-BASED
CONSTRUCTIVE ALGORITHMS FOR MULTI-STAGE
SCHEDULING WITH BLOCKING

PENGYU YAN
School of Economics and Management
University of Electronic Science and Technology of China, Chengdu 611731, China

SHI QIANG LIU* and CHENG-HU YANG
School of Economics and Management, Fuzhou University
Fujian Province 350108, China

MAHMOUD MASoud
School of Mathematical Sciences, Queensland University of Technology
2 George St, Brisbane Queensland 4001, Australia

(Communicated by Xiaoqiang Cai)

ABSTRACT. In this paper, the blocking conditions are investigated in permutation flow shop, general flow shop and job shop environments, in which there are no buffer storages between any pair of machines. Based on an alternative graph that is an extension of classical disjunctive graph, a new and generic polynomial-time algorithm is proposed to construct a feasible schedule with a given job processing sequence, especially for satisfying complex blocking constraints in multi-stage scheduling environments. To highlight the state-of-the-art of the proposed algorithm, a comparative analysis is conducted in comparison to two other constructive algorithms in the literature. The comparison shows that the proposed algorithm has the following advantages: i) it is more adaptive because it can be applied to three different types of scheduling problems (i.e., permutation flow-shop, general flow-shop and job-shop) without any modifications; ii) it is able to quickly evaluate whether a schedule is feasible (acyclic) or infeasible (cyclic) through checking the availability of the topological order in a directed alternative graph model; iii) it is able to determine the critical path which is useful to design the neighborhood moves in the development of metaheuristics.

1. Introduction.

1.1. Background. Motivated by real-world applications, investigation of blocking constraints in multi-stage scheduling systems creates a relatively new research direction in scheduling theory [45, 16, 42, 4, 37, 38, 23, 17, 24, 29, 30, 50]. For example, in the chemical industry, partially-processed chemical products have to be kept in the processing machine because of temperature requirement and the lack of safe

2010 Mathematics Subject Classification. Primary: 58F15, 58F17; Secondary: 53C35.
Key words and phrases. Constructive algorithm, multi-stage scheduling, blocking, alternative graph.

This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 71571032, 71531003, 71572156 and 71572030.
* Corresponding author: Shi Qiang Liu.
storage space [39]. The other example occurs in the railway industry where a train may have to hold-while-wait a railway track section until the next section on the routing becomes available [37, 38, 17, 31, 12, 36]. Another typical example arises in the healthcare industry where a patient has to remain in the pre-operative holding area (e.g., a ward bed) until the peri-operative service unit (e.g., an operating theater) becomes available [42]. One recent implementation of job shop scheduling with blocking constraints is to analyse and model the aircraft scheduling problem at busy European airport terminals [13, 49].

In multi-stage scheduling systems, there are two well-known types: job shop and flow shop. Furthermore, a flow shop is divided into two sub-types: a permutation flow shop if the same job sequence is allowed on every machine and a general (non-permutation) flow shop otherwise [32, 33, 44, 40, 11, 22, 51, 14, 2]. In a flow shop system, the operations of each job are processed on a set of machines in the same unidirectional order [2]. In comparison, the processing routes of jobs may be different in a job shop system [10, 56]. In this study, the permutation flow shop, the general flow shop and the job shop scheduling problems with blocking constraints are respectively called BPFSS, BGFSS and BJSS for ease of presentation.

1.2. Related literature. In comparison with a substantial body of knowledge in the traditional flow shop and job shop scheduling problems that neglect any buffering conditions, little research works addressed the BPFSS, BGFSS and BJSS problems. The related literature review on the BPFSS, BGFSS and BJSS problems is given as follows. McCormick et al. [39] developed a directed-graph-procedure method (see Section 3) for BPFSS based on the equivalent maximum flow and the longest path techniques. Abadi et al. [1] proposed a method to determine the makespan of a given BPFSS schedule, which was further used by Caraffa et al. [8] to develop a genetic algorithm to search the optimal makespan. Ronconi [47] improved the well-known NEH algorithm [41] to solve the BPFSS by replacing the first steps LTP rule of NEH algorithm with the MinMax and Profile Fitting rules. Ronconi [46] devised a branch-and-bound algorithm using a lower-bound determination method that exploits the blocking characteristics of BPFSS. Grabowski and Pempera [15] developed a tabu search metaheuristic algorithm to find the optimal schedule for BJSS, based on a so-called recursive-procedure method (see Section 3) for determining the makespan of a given BJSS schedule. Mascis and Pacciarelli [34] introduced some heuristic methods based on the alternative graph for BJSS by investigating dispatching rules such as Avoid Maximum Current Machine, Select Most Critical Pair, Select Most Balanced Pair, and etc. Gröflin and Klinkert [16] proposed an advanced tabu search metaheuristic algorithm for BJSS based on a generalised disjunctive graph model. Brucker and Kampmeyer [4] designed a tabu search metaheuristic algorithm for a cyclic BJSS problem. Pranzo and Pacciarelli [45] recently developed the greedy heuristics based on the alternative graph model to solve BJSS without swap-allowed and the BJSS with swap-allowed in deadlock situations.

1.3. Contribution and innovation. In the literature, most research works for the classical multi-stage scheduling problems are based on the disjunctive graph model [48, 21, 43, 3, 52, 26]. As a matter of fact, constructing a feasible solution of a multi-stage scheduling problem with blocking constraints is a challenging task. To the best of our knowledge, the development of efficient graph-based constructive algorithms for multi-stage scheduling with blocking has rarely been found in
the literature. To fill this gap, this paper aims to propose a more advanced and more generic alternative-graph-based constructive algorithm for solving the BPFSS, BGFSS and BJSS problems. To validate the state-of-the-art of the proposed approach, a comparative study is conducted by comparing two other constructive algorithms in the literature.

The remainder of this paper is outlined as follows. In Section 2, we describe this new constructive algorithm based on the alternative graph model. In Section 3, we conduct a comparative study and illustrate the results using a numerical example. The concluding remarks of this study are given in the last section.

2. An alternative-graph-based constructive algorithm.

2.1. Alternative graph. In the scheduling literature, the disjunctive graph model [48] has been extensively studied to model, analyse and solve many classical multi-stage scheduling problems. However, a strong limitation of the disjunctive graph model is that it disregards the capacities and conditions of intermediate buffers (i.e., temporary storage units) between two successive machines. To overcome this limitation, the alternative graph as an improvement of the disjunctive graph was introduced to take into account the blocking constraints [34]. In the alternative graph model, there are a set of $n$ jobs that consist of $N$ nodes (operations) $o = \{o_0, o_1, \ldots, o_n\}$ to be performed on a set of $m$ machines, where $N$ equals $n \times m + 2$. Nodes $o_0$ and $o_N$, respectively called source and sink, are dummy operations with zero processing time. There is a set of precedence relationships among each pair of operations. Different from the disjunctive graph model, the precedence relations are divided into two sets: conjunctive arcs and alternative arcs that replace disjunctive arcs. Associating a node to each operation, a multi-stage scheduling problem with blocking can be represented by the triple $G = \{O, C, A\}$ that denotes the alternative graph, which consists of a set of nodes $O$, a set of fixed-direction conjunctive arcs $C$ and a set of alternative arcs $A$. If $((o_i \rightarrow o_j), (o_h \rightarrow o_k)) \in A$, we say that $(o_i \rightarrow o_j)$ is the alternative of $(o_h \rightarrow o_k)$ and vice versa.

Two types of operations, namely ideal operation and blocking operation, should be distinguished in the alternative graph. An ideal operation is processed on a machine from its starting time to its completion time and then is able to leave this machine immediately. On the contrary, a blocking operation may have to remain on a machine even after its completion time, thus blocking this machine.

The difference between the disjunctive graph and the alternative graph is analysed in Figure 1, where

(a) The pair of disjunctive arcs $((o_i \rightarrow o_j), (o_j \rightarrow o_i))$ if both $o_i$ and $o_j$ are ideal.
(b) The pair of alternative arcs $((o_{SJ[i]} \rightarrow o_j), (o_j \rightarrow o_i))$ if $o_i$ is blocking and $o_j$ is ideal.
(c) The pair of alternative arcs $((o_i \rightarrow o_j), (o_{SJ[i]} \rightarrow o_i))$ if $o_i$ is ideal and $o_j$ is blocking.
(d) The pair of alternative arcs $((o_{SJ[i]} \rightarrow o_j), (o_{SJ[j]} \rightarrow o_i))$ if both $o_i$ and $o_j$ are blocking.

A blocking operation has the following three important attributes:

(i) a blocking operation $o_i$ of a job is primarily associated with its the same-job successor denoted as $o_{SJ[i]}$;

(ii) a blocking operation $o_i$ completed on machine $M(i)$ remains on it until the next machine (denoted as $M(SJ[i])$) becomes available for processing $o_{SJ[i]}$;
iii) the blocking time of operation \( o_i \) is no-zero if the starting time of \( o_{SJ[i]} \) is greater than the completion time of \( o_i \).

2.2. Algorithm procedure. As far as we know, efficient constructive algorithms for the BPMFSS, BPGFSS, and BJSS have rarely been explored. In this study, a new alternative-graph-based constructive algorithm is developed to construct a feasible schedule based on a given directed alternative graph, i.e., the determination of the timing information of all operations in the topological order.

The main procedure of the proposed alternative-graph-based constructive algorithm is described below:

Step 1. Construct a directed alternative graph model based on a given sequence of jobs (or a move after swapping two operations in neighborhood search if this constructive algorithm is embedded into a metaheuristic algorithm).

Step 2. Based on a directed alternative graph, set the list of predecessors and successors for each node.

Step 3. Based on the list of predecessors and successors for each node, compute the in-count value (i.e., the number of predecessors) of each node in the alternative graph.

Step 4. Based on the in-count values, determine the topological order:

Step 4.1 Select node 0 as the first node in the topological order.

Step 4.2 Decrease the in-count value for each of the immediate successor nodes of the selected node by one.

Step 4.3 Select any of the unselected nodes having a zero in-count value and put this node as the next node in the topological order. If none of the unselected nodes have a zero in-count value, stop running the algorithm as the given alternative graph is cyclic (infeasible).

Step 4.4 Repeat Steps 4.2 and 4.3 until all nodes are selected.

Step 5. For each node in the topological order:

Step 5.1 Get the index of a node (operation \( o_i \)) and the indices of its associated operations such as the same-job successor (operation \( o_{SJ[i]} \)) or the same-machine predecessor (operation \( o_{PM[i]} \)).

Step 5.2 If operation \( o_i \) is the first operation of a job, the starting time \( e_i \) of operation \( o_i \) is determined by \( e_i \leftarrow \max(0, e_{SJ[PM[i]]}) \).

Step 5.3 If operation \( o_i \) is processed on the last machine, the starting time \( e_i \) of operation \( o_i \) is determined by \( e_i \leftarrow \max(e_{PM[i]} + p_{PM[i]}, e_{PJ[i]} + p_{PJ[i]}) \).

Step 5.4 Otherwise, the starting time \( e_i \) of operation \( o_i \) is determined by \( e_i \leftarrow \max(\min(SJ[PM[i]], PJ[i]) + p_{PJ[i]}) \).

Step 6. For each node (excluding the last node) in the topological order:
Step 6.1 Get the indices of the node (operation $o_i$) and its same-job successor
(operation $o_{SJ[i]}$).

Step 6.2 Determine the completion time $c_i$ of operation $o_i$ by $c_i \leftarrow (e_i + p_i)$.

Step 6.3 Determine the blocking time $b_i$ of operation $o_i$ by $b_i \leftarrow \max(0, e_{SJ[i]} - c_i, 0)$.

Note that the proposed alternative-graph-based algorithm is a constructive algo-
rithm which aims at determining the timing information of all operations in terms
of the topological order based on a given directed alternative graph model. The
computational complexity of the alternative-graph-based algorithm is $O(N)$, where
$N$ is the number of total nodes in the graph model. Due to its efficiency, the
proposed alternative-graph-based constructive algorithm has been embedded in the
metaheuristic algorithms to solve several realistic scheduling problems with addi-
tional constraints (e.g., different release dates, setup times, buffering requirements
and etc.) in real-world applications [18, 19, 25, 27, 28].

3. A comparative study. In the literature, there are two constructive algorithms
called the directed-graph [39] and recursive-procedure [47] that can only be applied
to construct a feasible BPFSS schedule. In this section, a comparative analysis is
conducted to compare the proposed algorithm with these two approaches based on
a small-size numerical example.

Table 1. The processing times of four jobs in a numerical example

|       | $M_1$ | $M_2$ | $M_3$ |
|-------|-------|-------|-------|
| $J_1$ | $p_1=1$ | $p_8=3$ | $p_9=3$ |
| $J_2$ | $p_2=1$ | $p_9=2$ | $p_{10}=2$ |
| $J_3$ | $p_3=4$ | $p_{10}=1$ | $p_{11}=4$ |
| $J_4$ | $p_4=2$ | $p_{11}=2$ | $p_{12}=2$ |

3.1. A numerical example. The data of a BPFSS instance with four jobs and
three machines is given in Table 1. It is assumed that four jobs in this instance
have the same processing route, namely, $M_1 \rightarrow M_2 \rightarrow M_3$. Note that the processing
times of BPFSS will be kept the same to establish the data of the corresponding
BGFSS and BJSS instances.

3.2. Alternative-graph result. If the sequence of jobs is assumed as $J_2 \rightarrow J_4 \rightarrow
J_1 \rightarrow J_3$, then a directed alternative graph model for this BPFSS instance can be
drawn in Figure 2.

The detailed calculation procedure of applying the proposed constructive algo-
rithm to solve the directed alternative graph shown in Figure 1 is explained as
follows.

- In Steps 2-4, the topological order of total 14 nodes (twelve real operations plus
two dummy ones) is obtained as \{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0\}.

- In terms of the obtained topological order, the starting time of each node can
be determined sequentially in Step 5. For example, for node 3 that is the first
operation of job 3, its starting time is determined by $e_3 = \max(0, e_{SJ[PM[3]]})$ =
1. For node 9 that is processed on the last machine, its starting time is determined by $e_9 = \max(e_{PM}[9], e_{PJ}[9], p_{PJ[9]} + p_{PP[9]}) = 10$. For node 5 that is not processed on the last machine and not the first operation of a job, its starting time is determined by $e_5 = \max(e_{SJ[PM[5]]}, e_{PJ[5]}, p_{PJ[5]}) = 6$.

- Any blocking operation can be found with the determination of its non-zero blocking time in Step 6. For example, for node 8 that is a blocking operation, its blocking time is determined in Step 6.3, i.e., $b_8 = \max(e_{SJ[8]} - c_8, 0) = 1$.

3.3. Directed-graph result. In the literature, there is a so-called directed-graph-based constructive algorithm [39] that can be applied to determine the makespan of the BPFSS. In the below, the corresponding directed graph for this numerical example is drawn in Figure 3(a)-3(b), in which this approach is able to determine the longest path through a single machine or a node.

In comparison, the notations used in the directed-graph-based constructive algorithm are more complex. In the directed graph, the given permutation sequence of jobs has to be denoted as $\{\pi_1, \pi_2, \ldots, \pi_n\}$. In addition, each node is denoted as $(\pi_j, k)$ that associates the departure time $d_{\pi_j, k}$ of an operation $o_{\pi_j, k}$ which belongs to job $\pi_j$ and is processed on machine $k \in \{0, 1, \ldots, m\}$ where machine 0 is added as a dummy machine. Most nodes have two outgoing arcs: one arc is corresponding to node $(\pi_j, k)$ with a weight equal to the processing time of $p_{\pi_j, k}$; the other arc is denoted as $(\pi_j+1, k - 1)$ with a zero weight. In addition, only three specific nodes, namely $(\pi_j, 0)$, $(\pi_j, m)$ and $(\pi_n, k)$ have one outgoing arc that respectively connects the nodes $(\pi_j, 1)$, $(\pi_j+1, m - 1)$ and $(\pi_n, k + 1)$ with weights of $p_{\pi_j, 1}$, zero and $p_{\pi_n, k+1}$.

In the directed-graph-based constructive algorithm, the length of the longest path through Machine $k$ is computed by the following equation:

$$PL(k) = \sum_{q=1}^{k} p_{\pi_1, q} + \sum_{r=2}^{n} \max(p_{\pi_r, k}, p_{\pi_{r-1}, k+1}) + \sum_{q=k+1}^{m} p_{\pi_n, q}.$$ (1)
Similarly, the length of the longest path through an operation $o_{\pi_j,k}$ is determined by the following equation:

$$PL(\pi_j,k) = \sum_{q=1}^{k} p_{\pi_1,q} + \sum_{r=2}^{j} \max(p_{\pi_r,k},p_{\pi_{r-1},k+1}) + \sum_{r=j+1}^{m} \max(p_{\pi_r,k+1},p_{\pi_{r-1},k+2}) + \sum_{q=k+2}^{m} p_{\pi_n,q}.$$  

(2)

As shown in Figure 3(a), for example, the longest path through Machine 2 is highlighted by red dot line with the length of 14 time units. In Figure 3(b), the length of the longest path through node (1, 1) is 12 time units.

$PL(k) = (p_{2,1} + p_{2,2}) + (\max(p_{3,2},p_{2,3}) + \max(p_{1,2},p_{3,3}) + \max(p_{4,2},p_{1,3})) + p_{4,3}$
advanced and more generalised constructive algorithm based on a combination of blocking constraints (i.e., BPFSS). Thus, this gap inspires us to develop a more algorithm which are just for the permutation flow shop scheduling problem with 3.5.

Three comparisons.

Step 4. Determine the blocking time of $J_j$

Step 3. Based on the obtained time information of $J_j$

Step 1. Based on the directed graph model, get the permutation sequence of jobs $\pi_1 \rightarrow \pi_2 \rightarrow \pi_3 \rightarrow J_4$, and then set $d_2,0 = 0$.

Step 2. Determine the departure times of $J_2$: $d_{2,1} = 1$; $d_{2,2} = 3, d_{2,3} = 5$.

Step 3. Based on the obtained time information of $J_2$, determine the departure times of $J_3$: $d_{3,0} = d_{2,1} = 1$; $d_{3,1} = \max(d_{3,0} + p_{3,1}, d_{2,2}) = 5$; $d_{3,2} = \max(d_{3,1} + p_{3,2}, d_{3,3}) = 6$; $d_{3,3} = d_{3,2} + p_{3,3} = 10$.

Based on the obtained time information of $J_3$, determine the departure times of $J_1$: $d_{1,0} = d_{3,1} = 5$; $d_{1,1} = \max(d_{1,0} + p_{1,1}, d_{3,2}) = 6$; $d_{1,2} = \max(d_{1,1} + p_{1,2}, d_{3,3}) = 10$; $d_{1,3} = d_{1,2} + p_{1,3} = 13$.

Based on the obtained time information of $J_1$, determine the departure times of $J_4$: $d_{4,0} = d_{1,1} = 6$; $d_{4,1} = \max(d_{4,0} + p_{4,1}, d_{1,2}) = 10$; $d_{4,2} = \max(d_{4,1} + p_{4,2}, d_{1,3}) = 13$; $d_{4,3} = d_{4,2} + p_{4,3} = 15$.

Step 4. Determine the blocking time of $J_3$: $b_{3,1} = \max(0, d_{2,2} - d_{3,0} - p_{3,3}) = 0$; $b_{3,2} = \max(0, d_{2,3} - d_{3,1} - p_{3,2}) = 0$; $b_{3,3} = \max(0, d_{2,2} - d_{3,1} - p_{3,3}) = 0$.

Determine the blocking time of $J_1$: $b_{1,1} = \max(0, d_{4,2} - d_{1,0} - p_{1,1}) = 0$; $b_{1,2} = \max(0, d_{3,1} - d_{1,1} - p_{1,2}) = 1$; $b_{1,3} = \max(0, d_{3,1} - d_{1,1} - p_{1,2}) = 1$.

Determine the blocking time of $J_4$: $b_{4,1} = \max(0, d_{1,2} - d_{4,0} - p_{4,1}) = 2$; $b_{4,2} = \max(0, d_{1,3} - d_{4,1} - p_{4,2}) = 1$.

3.4. Recursive-procedure result. Using the same notations in the directed graph, another approach is called the recursive-procedure constructive algorithm, as it recursively calculates the departure times of each job on each machine in terms of the following sequential expressions [47].

The main steps of the recursive-procedure constructive algorithm are presented below.

Recursive-procedure Constructive Algorithm

Step 1. $d_{\pi_1,0} = 0$.

Step 2. $d_{\pi_1,k} = \sum_{i=1}^{k} p_{\pi_1,i}, \forall k = 1,2,\ldots,m$.

Step 3. For $d_{\pi_1,k}$, $\forall j = 2,\ldots,n$.

Step 3.1. $d_{\pi_1,0} = d_{\pi_1-1,1}$.

Step 3.2. $d_{\pi_1,k} = \max(d_{\pi_1,k-1} + p_{\pi_1,k}, d_{\pi_1-1,k+1}), \forall k = 1,\ldots,m-1$.

Step 3.3. $d_{\pi_1,m} = d_{\pi_1,m-1} + p_{\pi_1,m}, \forall j = 1,\ldots,n$.

Step 4. $b_{\pi_1,k} = \max(0, d_{\pi_1-1,k+1} - d_{\pi_1,k-1} - p_{\pi_1,k}), \forall j = 2,\ldots,n; k = 1,\ldots,m-1$.

As above, the so-called departure time (the completion time plus blocking time) of operations in the first job on every machine are calculated first, then the second job, and so on until the last job. The makespan of a BPFSS schedule is equal to the departure time of the last node, $d_{j,m}$.

For better understanding, the detailed calculation at each step is illustrated as follows.

Step 1. Based on the directed graph model, get the permutation sequence of jobs on each machine: $J_2 \rightarrow J_3 \rightarrow J_1 \rightarrow J_4$, and then set $d_{2,0} = 0$.

Step 2. Determine the departure times of $J_2$: $d_{2,1} = 1$; $d_{2,2} = 3, d_{2,3} = 5$.

Step 3. Based on the obtained time information of $J_2$, determine the departure times of $J_3$: $d_{3,0} = d_{2,1} = 1$; $d_{3,1} = \max(d_{3,0} + p_{3,1}, d_{2,2}) = 5$; $d_{3,2} = \max(d_{3,1} + p_{3,2}, d_{3,3}) = 6$; $d_{3,3} = d_{3,2} + p_{3,3} = 10$.

Based on the obtained time information of $J_3$, determine the departure times of $J_1$: $d_{1,0} = d_{3,1} = 5$; $d_{1,1} = \max(d_{1,0} + p_{1,1}, d_{3,2}) = 6$; $d_{1,2} = \max(d_{1,1} + p_{1,2}, d_{3,3}) = 10$; $d_{1,3} = d_{1,2} + p_{1,3} = 13$.

Based on the obtained time information of $J_1$, determine the departure times of $J_4$: $d_{4,0} = d_{1,1} = 6$; $d_{4,1} = \max(d_{4,0} + p_{4,1}, d_{1,2}) = 10$; $d_{4,2} = \max(d_{4,1} + p_{4,2}, d_{1,3}) = 13$; $d_{4,3} = d_{4,2} + p_{4,3} = 15$.

Step 4. Determine the blocking time of $J_3$: $b_{3,1} = \max(0, d_{2,2} - d_{3,0} - p_{3,3}) = 0$; $b_{3,2} = \max(0, d_{2,3} - d_{3,1} - p_{3,2}) = 0$; $b_{3,3} = \max(0, d_{2,2} - d_{3,1} - p_{3,3}) = 0$.

Determine the blocking time of $J_1$: $b_{1,1} = \max(0, d_{4,2} - d_{1,0} - p_{1,1}) = 0$; $b_{1,2} = \max(0, d_{3,3} - d_{1,1} - p_{1,2}) = 1$; $b_{1,3} = \max(0, d_{3,3} - d_{1,1} - p_{1,2}) = 1$.

Determine the blocking time of $J_4$: $b_{4,1} = \max(0, d_{1,2} - d_{4,0} - p_{4,1}) = 2$; $b_{4,2} = \max(0, d_{1,3} - d_{4,1} - p_{4,2}) = 1$.

3.5. Three comparisons. There are only two above constructive algorithms (i.e., directed-graph-based constructive algorithm and recursive-procedure constructive algorithm) which are just for the permutation flow shop scheduling problem with blocking constraints (i.e., BPFSS). Thus, this gap inspires us to develop a more advanced and more generalised constructive algorithm based on a combination of
alternative graph theory and the topological-sequence algorithm. The proposed algorithm is targeted to solve three types of multi-stage scheduling problems (including non-permutation flowshop and job shop) with blocking constraints, i.e., BPFSS, BGFSS, and BJSS. Using a numerical example, three comparisons are analysed to illustrate theoretic advances of the proposed algorithm in comparison to these two constructive algorithms in the literature.

**Comparison 1** The other two constructive algorithms (i.e., the directed-graph-based constructive algorithm and the recursive-procedure constructive algorithm) are only applicable to construct the feasible schedule of the BPFSS problem. In comparison, our proposed alternative-graph-based constructive algorithm is more adaptive, because it can be applied to not only the PMFSS but also the PGFSS and the BJSS. The evidence based on a numerical example to validate this advantage is given below.

**Evidence 1** Using the same data of this BPFSS example, the BPFSS problem is changed to the corresponding BJSS problem by adjusting the machine sequence of job $J_4$ (i.e., $o_4 \rightarrow o_8 \rightarrow o_{12}$ becomes $o_4 \rightarrow o_{12} \rightarrow o_8$), as highlighted by red dotted lines. Thus, a new directed alternative graph model for this BJSS example is displayed in Figure 4. After applying the proposed alternative-graph method without any modification in the algorithm procedure, the timing information of all operations in a BJSS schedule can still be efficiently determined.

**Comparison 2** The proposed alternative-graph-based constructive algorithm is able to rapidly evaluate whether the given schedule represented by a directed alternative graph model is feasible or infeasible.

**Evidence 2** For illustration, an infeasible BGFSS schedule is drawn as a directed alternative graph in Figure 5. In Step 4 of the alternative-graph-based constructive algorithm, it is able to find that the immediate successors of operations $o_6$ and $o_3$ are respectively $o_{10}$ and $o_7$, of which the in-count values still remain non-zero after decreasing one. In this case, a deadlock situation that none of the nodes contain zero in-count value occurs, which implies that this alternative graph is cyclic (see $o_{10} \rightarrow o_7 \rightarrow o_{11} \rightarrow o_{10}$) or the schedule is infeasible.
Comparison 3 The proposed alternative-graph-based constructive algorithm can determine the critical path in the graph model. In the directed-graph-based constructive algorithm, the length values of the longest paths through a machine or a node may be smaller than the makespan, which means that these longest paths cannot guarantee to be the critical path.

Evidence 3 Note that the length of a critical path should be equal to the makespan value. As shown in Figure 3(a)-3(b), the lengths of two longest paths though Machine 2 and Node (1, 1) are 14 and 12, respectively. However, the makespan of this 4-job and 3-machine BPFSS instance is 15 indeed. For ease of comparison, the critical path of this BPFSS instance is highlighted by red dot line, as shown in Figure 6.
4. Conclusions. Contribution and innovation of this paper are concluded as follows. A new alternative-graph-based constructive algorithm is proposed to model, analyse and solve the permutation flow shop, general flow shop, and job shop scheduling with blocking constraints. The proposed alternative-graph constructive algorithm is a fundamental tool to deal with real-world multi-stage scheduling systems that must consider buffering conditions. According to this comparative study, it is summarised that the proposed alternative-graph-based constructive algorithm has the following three advantages.

i) The proposed algorithm is more adaptive because it can be applied to solve the BPFSS, BGFSS and BJSS problems without any modification in the algorithm procedure. In comparison, the other two constructive algorithms can only be used to solve the BPFSS problem.

ii) The proposed algorithm can efficiently determine the timing information of all operations as well as the critical path of which the length is equal to the makespan value. In comparison, for the directed-graph-based constructive algorithm, the length of the longest path through a machine or a node in the directed graph may be smaller than the makespan, which means that this approach cannot guarantee to find the critical path.

iii) The proposed algorithm is capable of quickly assessing whether a BGFSS or BJSS schedule is infeasible during the algorithm procedure, based on the given alternative graph model.

Regarding the future research, an important extension of this study is to embed this alternative-graph constructive algorithm into a metaheuristic algorithm with the design of specific neighborhood moves to find the near-optimal (or optimal) multi-stage schedule while satisfying the blocking constraints. More computational experiments based on the benchmark instances will be conducted to test the effectiveness and efficiency of the proposed constructive and metaheuristic algorithms, by comparing with the results obtained by the exact MIP solver such as IBM ILOG-CPLEX. The proposed constructive algorithm is promising to be applied to deal with the real-world robotic cell, railway, mining, aviation and hospital scheduling problems that actually should consider intermediate buffering conditions [24, 12, 36, 13, 49, 27, 35, 53, 54, 55, 20, 5, 9]. Moreover, the proposed approach will be extended to deal with the stochastic-version scheduling problems to minimise the discounted holding cost due to blocking conditions [6, 7].

Acknowledgments. The authors are very grateful to the editors and the anonymous reviewers for their constructive comments and suggestions that have led to an improved version of this paper.

REFERENCES

[1] I. N. K. Abadi, N. G. Hall and C. Sriskandarajah, Minimizing cycle time in a blocking flow shop, Oper. Res., 48 (2000), 177–180.
[2] D. Bai, M Tang, Z. H. Zhang and E. D. R. Santibanez-Gonzalez, Flow shop learning effect scheduling problem with release dates, Omega, (2017).
[3] K. R. Baker and T. Dan, Principles of Sequencing and Scheduling, Wiley Publishing, 2009.
[4] P. Brucker and T. Kampmeyer, Cyclic job shop scheduling problems with blocking, Annals of Operations Research, 159 (2008), 161–181.
[5] R. L. Burdett and E. Kozan, An integrated approach for scheduling health care activities in a hospital, European Journal of Operational Research, 264 (2018), 756–773.
[6] X. Cai, X. Wu and X. Zhou, Optimal Stochastic Scheduling, Springer, CA, USA, 2014.
[7] X. Cai, X. Wu and X. Zhou, Stochastic scheduling on parallel machines to minimize discounted holding costs, *Journal of Scheduling*, 12 (2009), 375–388.

[8] V. Caraffa, S. Ianes, T. P. Bagchi and C. Sriskandarajah, Minimizing makespan in a blocking flowshop using genetic algorithms, *International Journal of Production Economics*, 70 (2001), 101–115.

[9] A. Che, V. Kats and E. Levner, An efficient bicriteria algorithm for stable robotic flow shop scheduling, *European Journal of Operational Research*, 260 (2017), 964–971.

[10] T. C. E. Cheng, B. Peng and Z. Lü, A hybrid evolutionary algorithm to solve the job shop scheduling problem, *Annals of Operations Research*, 242 (2016), 223–237.

[11] D. Cinar, J. A. Oliveira, Y. I. Topcu and P. M. Pardalos, A priority-based genetic algorithm for a flexible job shop scheduling problem, *Journal of Industrial and Management Optimization*, 12 (2016), 1391–1415.

[12] A. D’Ariano, D. Pacciarelli and M. Pistelli, A branch and bound algorithm for scheduling trains in a railway network, *European Journal of Operational Research*, 183 (2007), 643–657.

[13] A. D’Ariano, D. Pacciarelli and M. Pistelli, Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area, *Networks*, 65 (2015), 212–227.

[14] B. Q. Fan and T. C. E. Cheng, Two-agent scheduling in a flowshop, *European Journal of Operational Research*, 252 (2016), 376–384.

[15] J. Grabowski and J. Pempera, The permutation flow shop problem with blocking. A tabu search approach, *Omega*, 35 (2007), 302–311.

[16] H. Gröflin and A. Klinkert, A new neighborhood and tabu search for the Blocking Job Shop, *Discrete Applied Mathematics*, 157 (2009), 3646–3655.

[17] E. Kozan and S. Q. Liu, A demand-responsive decision support system for coal transportation, *Decision Support Systems*, 54 (2012), 665–680.

[18] E. Kozan and S. Q. Liu, An operational-level multi-stage mine production timetabling model for optimally synchronising drilling, blasting and excavating operations, *International Journal of Mining Reclamation and Environment*, 31 (2017), 457–474.

[19] E. Kozan and S. Q. Liu, A new open-pit multi-stage mine production timetabling model for drilling, blasting and excavating operations, *Mining Technology*, 125 (2016), 47–53.

[20] J. Lange and F. Werner, Approaches to modeling train scheduling problems as job-shop problems with blocking constraints, *Journal of Scheduling*, 21 (2018), 191–207.

[21] J. Y. T. Leung, L. Kelly and J. H. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press, USA, 2004.

[22] Y. K. Lin and C. S. Chong, A tabu search algorithm to minimize total weighted tardiness for the job shop scheduling problem, *Journal of Industrial and Management Optimization*, 12 (2016), 703–717.

[23] S. Q. Liu and E. Kozan, Parallel-identical-machine job-shop scheduling with different stage-dependent buffering requirements, *Computers and Operations Research*, 74 (2016), 31–41.

[24] S. Q. Liu and E. Kozan, Optimising a coal rail network under capacity constraints, *Flexible Services and Manufacturing Journal*, 23 (2011), 90–110.

[25] S. Q. Liu and E. Kozan, International Journal of Systems Science: Operations and Logistics, *International Journal of Systems Science: Operations and Logistics*, (2017), 1–14.

[26] S. Q. Liu and E. Kozan, Scheduling a flow shop with combined buffer conditions, *International Journal of Production Economics*, 117 (2009), 371–389.

[27] S. Q. Liu and E. Kozan, A hybrid metaheuristic algorithm to optimise a real-world robotic cell, *Computers and Operations Research*, 84 (2017), 188–194.

[28] S. Q. Liu, E. Kozan, Y. Zhang and F. T. S. Chan, Job shop scheduling with a combination of four buffering constraints, *International Journal of Production Research*, (2017), 1–20.

[29] S. Q. Liu and E. Kozan, Optimising a coal rail network under capacity constraints, *Flexible Services and Manufacturing Journal*, 23 (2011), 90–110.

[30] S. Q. Liu and E. Kozan, Scheduling a flow shop with combined buffer conditions, *International Journal of Production Economics*, 117 (2009), 371–380.

[31] S. Q. Liu and E. Kozan, Scheduling trains as a blocking parallel-machine job shop scheduling problem, *Computers and Operations Research*, 36 (2009), 2840–2852.

[32] S. Q. Liu and H. L. Ong, Metaheuristics for the mixed shop scheduling problem, *Asia-Pacific Journal of Operational Research*, 21 (2004), 97–115.

[33] S. Q. Liu and H. L. Ong, A comparative study of algorithms for the flowshop scheduling problem, *Asia-Pacific Journal of Operational Research*, 19 (2002), 205–222.
[34] A. Mascis and D. Pacciarelli, Job-shop scheduling with blocking and no-wait constraints, *European Journal of Operational Research*, 143 (2002), 498–517.

[35] M. Masoud, E. Kozen, G. Kent and S. Q. Liu, Experimental dataset for optimising the freight rail operations, *Data in Brief*, 9 (2016), 492–500.

[36] M. Masoud, G. Kent, E. Kozen and S. Q. Liu, A new multi-objective model to optimize rail transport scheduler, *Journal of Transportation Technologies*, 6 (2016), 86–98.

[37] M. Masoud, E. Kozen, G. Kent and S. Q. Liu, A new constraint programming approach for optimising a coal rail system, *Optimization Letters*, 11 (2017), 725–738.

[38] M. Masoud, E. Kozen, G. Kent and S. Q. Liu, An integrated approach to optimise sugarcane rail operations, *Computers and Industrial Engineering*, 98 (2016), 211–220.

[39] S. T. McCormick, M. L. Pinedo, S. Shenker and B. Wolf, Sequencing in an assembly line with blocking to minimize cycle time, *Operations Research*, 37 (1989), 925–935.

[40] E. Mokotoff, Algorithms for bicriteria minimization in the permutation flow shop scheduling problem, *Journal of Industrial and Management Optimization*, 7 (2011), 253–282.

[41] M. Nawaz and E. E. Ensore, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, 11 (1983), 91–95.

[42] D.-N. Pham and A. Klinkert, Surgical case scheduling as a generalized job shop scheduling problem, *European Journal of Operational Research*, 185 (2008), 1011–1025.

[43] M. L. Pinedo, *Planning and Scheduling in Manufacturing and Services*, Springer, New York, USA, 2005.

[44] C. N. Potts, D. B. Shmoys and D. P. Williamson, Permutation vs. non-permutation flow shop schedules, *Operations Research Letters*, 10 (1991), 281–284.

[45] M. Pranzo and D. Pacciarelli, An iterated greedy metaheuristic for the blocking job shop scheduling problem, *Journal of Heuristics*, 22 (2016), 587–611.

[46] D. Ronconi, A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking, *Annals of Operations Research*, 138 (2005), 53–65.

[47] D. P. Ronconi, A note on constructive heuristics for the flowshop problem with blocking, *International Journal of Production Economics*, 87 (2004), 39–48.

[48] D. Roy and B. Sussmann, *Les Problèmes D’ordonnancement avec Contraintes Disjontives*, Note DS No, Montrouge, 1964.

[49] M. Samà, A. D’Ariano, P. D’Ariano and D. Pacciarelli, Scheduling models for optimal aircraft traffic control at busy airports: Tardiness, priorities, equity and violations considerations, *Omega*, 67 (2017), 81–98.

[50] Y. W. Shin and D. H. Moon, Throughput of flow lines with unreliable parallel-machine workstations and blocking, *Journal of Industrial and Management Optimization*, 13 (2017), 901–916.

[51] J. K. Winch, X. Cai and G. L. Vairaktarakis, Cyclic job scheduling in paced assembly lines with cross-trained workers, *International journal of production research*, 45 (2007), 803–828.

[52] P. Yan, A. Che, E. Levner and S. Q. Liu, A heuristic for inserting randomly arriving jobs into an existing hoist schedule, *IEEE Transactions on Automation Science and Engineering*, PP (2017), 1–8.

[53] P. Yan, A. Che, N. Yang and C. Chu, A tabu search algorithm with solution space partition and repairing procedure for cyclic robotic cell scheduling problem, *International Journal of Production Research*, 50 (2012), 6403–6418.

[54] P. Yan, C. Chu, N. Yang and A. Che, A branch and bound algorithm for optimal cyclic scheduling in a robotic cell with processing time windows, *International Journal of Production Research*, 48 (2010), 6461–6480.

[55] P. Yan, A. Che, X. Cai and X. Tang, Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance, *Computers and Operations Research*, 50 (2014), 128–140.

[56] C. Y. Zhang, P. Li, Y. Rao and Z. Guan, Minimizing cycle time in a blocking flowshop, *Operations Research*, 48 (2000), 177–180.