Sheikha, Shaya; Komakib, G. M.; Kayvanfarc, Vahid; Teymourian, Ehsan

Article

Multi-Stage assembly flow shop with setup time and release time

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Sheikha, Shaya; Komakib, G. M.; Kayvanfarc, Vahid; Teymourian, Ehsan (2019) : Multi-Stage assembly flow shop with setup time and release time, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 6, pp. 1-15, https://doi.org/10.1016/j.orp.2019.100111

Operations Research Perspectives 6 (2019) 100111

Contents lists available at ScienceDirect

Operations Research Perspectives

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

Multi-Stage assembly flow shop with setup time and release time

Shaya Sheikh[a,*], G.M. Komaki[b], Vahid Kayvanfar[c], Ehsan Teymourian[d]

[a] Assistant Professor of Operations Management, School of Management, New York Institute of Technology, 1855 Broadway, New York, NY 10023, United States
[b] Assistant Professor of Business Analytics, Department of Marketing and Business Analytics, College of Business, Texas A&M Uni-Commerce, 2200 Campbell St, Commerce, TX 75428, USA
[c] Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Ave., 15875-4413, Tehran, Iran
[d] Department of Management Science and Information Systems, Rutgers, the state university of New Jersey, Newark & New Brunswick, NJ, USA

ARTICLE INFO

Keywords:
Scheduling
Assembly flow shop
Makespan
Completion time

ABSTRACT

This article addresses multi-stage assembly flow shop for the first time. We propose several industrial applications of this system in which the opening stage has *m* parallel machines and the following stages (e.g. assembly, transportation, painting, packaging, etc.) complete and prepare products for delivery. We develop a mixed binary linear optimization model and a constraint-free mathematical presentation for this problem. We explore performance measures of makespan and total completion time and derive polynomial optimal solutions for special cases that are likely to occur in industry. We also propose lower bounds as well as nine efficient heuristics for solving the problem with the objective of minimizing makespan. The results reveal that the proposed lower bounds is tight enough. Moreover, the optimization models and derived polynomial algorithms can be applied to assembly flow shops without setup or release time. Finally, we implement General Variable Neighborhood Search (GVNS) and Grey Wolf Optimizer to improve the heuristic solutions. Comparison of employed algorithms on randomly generated instances indicates that GVNS outperforms other heuristics. The performance of proposed heuristics and meta-heuristics are confirmed with detailed statistical analysis.

1. Introduction

During the last three decades, global competition and market demand for diverse range of products have increased the application of modular design and assembly flow shops. Researchers have explored assembly flow shops with two or three stages with a variety of performance measures under realistic assumptions. In practice, an assembly flow shop may have more than one post-processing operations after the assembly stage. These stages include but not limited to painting, finishing, polishing, inspection, packaging, etc. Given significant transportation time between stages, we can treat transportation time as a separate stage. Consideration of post-processing operations lead us to a new scheduling problem, called the multi-stage assembly flowshop (MSAF), which is the focus of this paper. Fig. 1 illustrates a schematic of multi-stage assembly flowshop. One practical example of these systems are complete knock-down kits or semi-knock-down kits that are popular in automotive, electronics, and furniture industry. These kits are shipped from suppliers that are usually located overseas and are assembled in an assembly plant close to customers' location. MSAF can also be utilized in production systems with multi-plant facilities where each plant is dedicated for processing of one or a set of components. The main processed components (a.k.a. subassemblies) are then shipped to the assembly and post-processing plant before delivery to the customer or warehouse. The processing times can be considered fixed with reasonable accuracy as majority of these processes are performed by robots. We use the average processing time for processes that are run by human (e.g. quality control inspections).

One industrial application of MSAF, which is the motivation of this paper, is the production and assembly of medical equipment such as dental chair, autoclave, and light cure. Dental chairs are sold in a variety of models with different features that addresses the needs of medical centers and dentists. The wide range of these features impacts the production time for each dental chair. The main components of a dental chair are made of aluminum, steel, and plastic. These components, together with dental tools (e.g. headpiece), and circuit boards need to be either produced in-house through casting, extrusion, machining, etc., or to be outsourced. Production of these components are achieved simultaneously in the first stage of MSAF. In the second stage, all components are assembled to make a dental chair. For simplicity, we consider one assembly stage in which all sub-assemblies such as arm system, light system, and unit stand are assembled together. Right after the assembly, dental chair goes through a series of post-processing

* Corresponding author.
E-mail addresses: Ssheik11@nyit.edu (S. Sheikh), v.kayvanfar@aut.ac.ir (V. Kayvanfar), ehsan.teymourian@rutgers.edu (E. Teymourian).

https://doi.org/10.1016/j.orp.2019.100111
Received 4 March 2018; Received in revised form 25 February 2019; Accepted 7 April 2019
Available online 17 April 2019

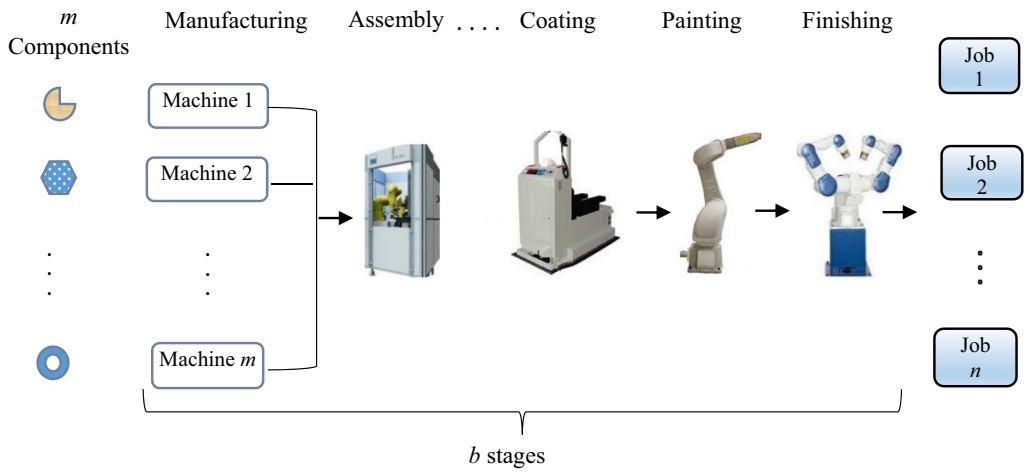

Fig. 1. Schematic presentation of multi-stage assembly flowshop problem.

stages including balancing instrument suspensions, quality inspection, and shipping preparations such as disassembly of arm and unit stand, wrapping, and packaging.

Another application of MSAF systems is in autoclave manufacturing. Autoclave is made of pressure vessel, frame, and control subassemblies. These three subassemblies are manufactured simultaneously using guillotine for cutting steel and producing panels, CNC lathe, and bending machines for punching holes and shaping corners, and welding machine for welding frames. In the second stage, the door and closure fittings are assembled to the pressure vessel to make it water tight and ready for pressure testing. Next, the pressure vessel goes under pressure testing in order to check the integrity of the welds. The amount and length of pressure varies based on the size and specific application of autoclave. The microprocessor control board is fitted along with the wiring loom to the autoclave during the assembly. In the next stage, control panel of autoclaves are set for different licensing needs. Next, the autoclave is given a full operational test where it is inspected against a checklist. Different customizations are also checked at this stage. Finally, the last stage is dedicated to packaging and making the product ready for shipping and delivery.

Majority of assembly flow shop (AF) articles assume that setup time is negligible. However, setup times could take considerable amount of time in real applications. In this article, set-up times for all stages are assumed to be separate from processing times and are sequence independent (i.e. it depends only on the processing job). MSAF systems are mainly made of multi-purpose machines that require specific setup times for preparations and fixture changes. Even though, combining these setup times with processing time may seem reasonable in the first place, the aggregate value may lead to increase in completion times and therefore, affects the productivity measure. Hence, we treat setup times separately from processing time. Importance and application of scheduling models with explicit setup times have been discussed in different studies, see Allahverdi et al. [1–5] and Komaki et al. [19].

We also consider release times for components. This is a realistic assumption as many outsourced components and raw material are provided by suppliers that are located in different geographical locations. Therefore, components are likely to be ready for production at different times.

Multi-stage assembly problems are introduced in this article for the very first time. We briefly discuss the literature review in Section 2. Problem definition and special cases are explained in Section 3. Authors would like to highlight that the derived optimal polynomial solutions in this section can be generalized for as many stages and jobs, with or without setup or release times. Proposed lower bounds are discussed in Section 4. Finally, Section 5 concludes the paper.

2. Literature review

We adopt Graham et al. [11] triple notation, $\alpha|\beta|\gamma$, to represent the characteristics of the proposed problem. Accordingly, the MSAF with release and setup time is denoted as AFb$|r_j,s_j|\gamma$ where $b > 2$. Koulamas and Kyparisis [20] considered a three-stage AF model with intermediate stage for transporting components from production stage to assembly stage. Considering transportation stage in AF model is specifically critical for multi-plant AF facilities. AF(m,1,1) with objectives of makespan or total completion time has been explored by researchers such as Koulamas & Kyparisis [20], Komaki et al. [17], Andrés and Hatami [6], and Maleki et al. [23,24]. In all these systems, the second stage is responsible for collecting the processed components from the machining stage and shipping them to the third stage where components are assembled by one assembly machine. Here, we explain the methods that have been practiced by theses researchers for solving the three stage AF problem.

Hatami et al. [12] proposed a three stage AF with objectives of mean flow time and maximum tardiness and sequence dependent setup times and transfer times. Authors proposed simulated annealing and tabu search as the solution procedures. Andrés and Hatami [6] proposed two mathematical models for three-stage AF with sequence dependent setup times on the first and third stages and the objective of minimizing total completion time. Koulamas and Kyarisis [20] investigated three-stage AF with the objective of minimizing makespan. They proposed heuristics based on Johnson rule with ratio and absolute performance guarantees. Maleki et al. [24] studied a three stage AF with sequence dependent setups at the first stage and blocking time between stages. The objective was to minimize weighted mean completion time and makespan. Authors proposed a simulated annealing to generate efficient results. Maleki et al. [23] used Taguchi technique to tune the parameters of simulated annealing and variable neighborhood search techniques. Shoaardebili and Fattahi [30] presented a three stage AF model with machine availability constraint and objectives of minimizing weighted completion times and sum of weighted tardiness and earliness. Authors applied NSGAII and multi-objective simulated annealing to solve this problem in a reasonable time.

Tajbakhsh et al. [31] proposed a mixed integer program with the objective of minimizing makespan and sum of earliness and tardiness for a three stage AF with machining, assembly, and batch processing stages. Authors utilized a hybrid algorithm that embeds advantages of genetic algorithm and particle swarm optimization. Komaki et al. [17] proposed an improved discrete cuckoo optimization algorithm for the same problem. Authors performed extensive comparisons with nine other heuristics and conclude that their proposed approach outperforms

other heuristics. Yokoyama and Santos [34] presented a three stage AF (1,1,1) model with one manufacturing machine in each of the first two stages and an assembly machine in the third stage. The objective is to minimize the weighted sum of completion times. Authors used branch and bound technique and analyzed some special cases for the presented problem. Yokoyama and Santos [34] addressed three-stage assembly flowshop scheduling problem that minimizes total completion time of products.

Another variation of multi-stage assembly flowshops is distributed assembly flowshops, DAF, with several factories at different locations and with the goal of finding the best sequence of jobs at each factory. Hatami et al. [13] presented a family of constructive heuristics and variable neighborhood descend algorithms for DAF(m,1) where the latter technique outperforms the former. Later, Hatami et al. [14] extended their model in Hatami et al. [13] by considering setup times in both machining and assembly stages. They proposed four high performance constructive heuristics as the solution procedure. The heuristic algorithms find the sequence of products and then use the rules developed by Naderi and Ruiz [27] to assign jobs to the factories. Lin et al. [22] offered back tracking search hyper heuristics as the solution procedure for DAF(m,1)||C_{max}. Gonzalez et al. [10] proposed permutation DAF(m,1) with the objective of minimizing expected makespan and with stochastic processing and assembly times. Authors offered a hybrid algorithm and a simulation technique with competitive results. Zhang et al. [36] developed DAF(m,1)||C_{max} with flexible assembly and setup times and offered hybrid variable neighborhood search, and hybrid particle swarm optimization to find efficient schedules. Li et al. [38] proposed a hybrid iterated local search with simulated annealing for distributed assembly permutation flowshop with multiple assembly factories and no-waiting time in the processing stage. Fig. 2 shows the relationship between our proposed multi-stage AF model and other AF models in the literature. In Fig. 2, PD_m represents customer order scheduling problem presented by Framinan and Gonzalez [8].

This article extends the two-stage assembly problem AF(m,1) by adding the post-assembly stages. AFb also extends the flowshop scheduling problem by allowing concurrent operations at the first stage, see Komaki et al. [19]. It is known that F3||C_{max} [9] and AF||C_{max} [21] are NP-Hard. AFb is the generalization of these problems with more complexity in terms of number of stages. Thus, the addressed problem is also NP-hard in strong sense. A summary of three stage AF models in the literature is shown in Table 1.

3. Problem description

The addressed problem in this study, AFb|r_j,s_j|γ, consists of m non-identical parallel machines at the first stage followed by $b-1$ post-processing stages with one machine per stage. The assembly stage can be any one of these $b-1$ post-processing stages. There are n jobs, each with m components and $m+b$-1 operations where the first m operations are processed in the first stage, simultaneously, and the b-1 operations in the post-processing stages with one operation at each stage. The assembly operation starts only after all components are processed in the first stage.

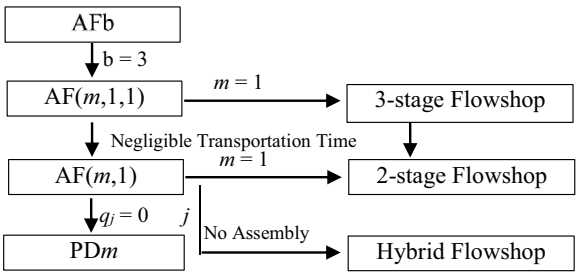

Fig. 2. Relationship between assembly flow shops.

3.1. Assumptions and notations

We consider the following assumptions for the proposed AFb|r_j,s_j|γ problem:

1) Preemption is not allowed on any machine, that is, once processing a component or product is started, it cannot be interrupted.
2) Components may not be available at time zero (non-zero release time).
3) Assembly or post processing stages begin right after all previous production processes are completed.
4) Each machine can work on one job at a time.
5) There is unlimited buffer for the jobs waiting to be processed on any machine.
6) Processing times are fixed. Alternatively, average processing times can be considered with reasonable accuracy for processes with limited variations in processing times.

Optimal solution for AF2||C_{max} has a special property in which sequence of jobs is the same for both production and assembly stages, i.e., permutation schedule is an optimal schedule for this model, see Potts et al. [28]. However, this conclusion does not necessarily hold for AF (m,1) with setup and release times. Tozkapan et al. [39] proved that permutation schedules generate superior result for AF(m,1,1) problems with minimizing total completion time and their proof can be generalized to AF(m,1,1) problems with makespan objectives. In this paper, we show that permutation schedule optimizes performance measures of AFb given that the first stage machines have no-idle condition.

Table 2 represents notations used in this article. Using the triple notation, the proposed problems in this paper are AFb|r_j,s_j|C_{max}, and AFb|r_j,s_j|TC where r_j and s_j denote the existence of positive release times and setup times. Let $j = 1, 2,.., n$ be the job index and t_{kj} be the processing time of component $k = 1,2,..,m$ of job j in the first stage.

3.2. Optimization models and special cases

The following mixed binary linear optimization model is the modified form of the AF(m,1)|s_j|C_{max} model presented Deng et al. [37] model.

Minimize C_{max}

$$\sum_{i=0,i\neq j}^{n} x_{[i,j,l]} = 1 \;\forall\, j = 1,\cdots,n;\; l = 1,\cdots,b \tag{1}$$

$$\sum_{j=0,j\neq i}^{n} x_{[i,j,l]} \leq 1 \;\forall\, i = 1,\cdots,n;\; l = 1,\cdots,b \tag{2}$$

$$x_{[i,j,l]} + x_{[j,i,l]} \leq 1 \qquad \forall\; i = 1,\ ...,n-1; j > i;\; l = 1,\ ...,b \tag{3}$$

$$A_{[k,j]} \geq A_{[k,i]} + s_{[k,j]} + t_{[k,j]} - O(1 - x_{[i,j,l]}) \qquad \forall\; i = 0,\ ...,n; j = 1,\ ...,n;\; j \neq i;\; k = 1,\ ...,m \tag{4}$$

$$A_{[k,j]} \geq r_{[k,j]} + s_{[k,j]} + t_{[k,j]} \qquad \forall\; j = 1,\ ...,n; k = 1,\ ...,m \tag{5}$$

$$C_{[l,j]} \geq C_{[l,i]} + sa_{[l,j]} + q_{[l,j]} - O(1 - x_{[i,j,l]}) \;\forall\; i = 0,\cdots,n; j = 1,\cdots,n; j \neq i; l = 2,\cdots,n \tag{6}$$

$$C_{[l,i]} \geq C_{[l-1,i]} + q_{[l,i]} \qquad \forall\; i = 1,\ ...,n; l = 3,\ ...,b \tag{7}$$

$$C_{[2,i]} \geq A_{[k,i]} + q_{[2,i]} \qquad \forall\; i = 1,\ ...,n; k = 1,\ ...,m \tag{8}$$

$$C_{max} \geq C_{[b,i]} \qquad \forall\; i = 1,\ ...,n \tag{9}$$

$$X_{[i,j,l]}\ \{0,\ 1\} \qquad \forall\; i, j = 0,\ ...,n \qquad ; l = 2,\ ...,b \tag{10}$$

$$C_{[l,j]}, A_{[k,j]} \qquad \forall\; j = 1,\ ...,n; k = 1,\ ...,m \tag{11}$$

Table 1
AF publications with three stages.

Problem Type	Reference	Model Remarks	Solution Method(s)
AF(1,m_1, m_2)| Setup Time |C_{max}	Zhang et l. [35]	MILP	Hybrid genetic algorithm
AF(m,1,1)||C_{max}	Koulamas & Kyparisis [20]	–	Heuristic
AF(m,1,1)||C_{max}	Komaki et al. [17]	–	Improved discrete cuckoo optimization
AF(m_1, m_2,1)||C_{max}	Morizava [26]	–	List based squeezing branch and bound
AF(m,1,1)| Setup Time |Σw_iC_i	Andrés & Hatami [6]	MILP	CPLEX
AF(m,1,1)| Setup Time |Σw_iC_i	Maleki et al. [24]	–	Simulated annealing
AF(m_1,1, m_2)||Σ w_iC_i	Xiong et al. [33]	MILP	Heuristics based on shortest processing time, Hybrid genetic algorithms and variable neighborhood search
AF(2,1,1)|| Tardiness	Tharumarajah et al. [32]	MILP	Heuristic
AF(m,1,1)| Setup Time | Tardiness	Campos et al. [7]	–	Generalized variable neighborhood search
AF(m,1,1)| Setup Time |(Σw_iC_i, T_{max})	Hatami et al. [12]	MO-MILP Initial solution with EDD, SPT	Simulated annealing and tabu search
AF(m,1,1)| Setup Time |(C_{max}, Σw_iC_i)	Maleki et al. [23]	–	Variable neighborhood search
AF(m,1,1)||(C_{max}, Earliness, Tardiness)	Tajbakhsh et al. [31]	MO-MINLP	Hybrid particle swarm optimizer and genetic algorithm
AF(m,1,1)||(Σw_iC_i, Earliness, Tardiness)	Shoaardebili & Fattahi [30]	MO—NLP machine availability constraint semi-resumable operations	Non-dominated sorting genetic algorithm and Multi-objective simulated annealing

$$C_{[l,0]},\ A_{[k,0]} \qquad \forall\ k = 1,\ \ldots, m;\ l = 2,\ \ldots, b \tag{12}$$

Constraints (1) in above model ensure that each job has only one predecessor in every stage. Since the precedence relationship of jobs in different stages could change, we need to consider l as index for binary variable $x_{[j,\ i,\ l]}$. Constraints (2) indicate that each job has at most one succeeding job. Constraints (3) guarantee that a job cannot be both a predecessor and a successor of another job simultaneously. Constraints (4,5) indicate the job processing precedence relationships in the first stage. Constraints (6,7) indicate the post-processing precedence relationships. Constraints (8) shows that each job cannot be assembled before all its components are completed in the first stage.

Obtaining a feasible solution for this NP-hard, large-sized optimization problem in reasonable time is extremely difficult. For instance, the smallest problem instance in Section 6 will have more than 2300 mixed integer binary or integer constraints using the above optimization model. As a result, we introduce a constraint-free format of AFb model and derive special polynomial optimal cases together with efficient heuristics for this alternative presentation.

Koulamas and Kyparisis [20] presented the makespan formula for a given sequence of products, Л, for AF(m,1,1). Nevertheless, expanding their equations to more than three stages considering setup and release times does not lead to a relatively compact structure. Thus, we extend the AF(m,1)$|r_j|\mathrm{C_{max}}$ model presented by Komaki and Keyvanfar [16] and Sheikh et al. [29] in order to derive makespan and total completion times for AFb$|r_j,s_j|\gamma$ with $b \geq 3$:

$$C_{[j]} = \max\{\ldots\max\{\max\{\max_{\forall k}\{\max\{r_{[k,j]},\ A_{[k,j-1]}\} + t_{[k,j]} + s_{[k,j]}\},\ C_{[2,j-1]} + sa_{[2,j]}\} + q_{[2,j]},\ C_{[3,j-1]} + sa_{[3,j]}\} + q_{[3,j]}\ldots, C_{[b,j-1]} + sa_{[b,j]}\} + q_{[b,j]} \tag{13}$$

$$A_{[k,j-1]} = \max\{r_{[k,j-1]},\ A_{[k,j-2]}\} + t_{[k,j-1]} + s_{[k,j-1]}\ \forall\ k = 1,\ \ldots, m\&j = 2,\ \ldots, n\&A_{[k,0]} = 0 \tag{14}$$

$$C_{[l,j-1]} = \max\{C_{[l-1,j-1]},\ C_{[l,j-2]} + sa_{[l,j-1]}\} + q_{[l,j-1]}\ \forall\ l = 2,\ \ldots, b-1\&j = 2,\ \ldots, n\&C_{[l,0]} = 0 \tag{15}$$

The performance measures are $\mathrm{C_{max}} = C_{[n]}$ and $\mathrm{TC} = \sum_{j=1}^{n} C_{[j]}$.

For $l = 2$, Eq. (15) is written as $C_{[2,j-1]} = \max\{\max_{\forall k}\{A_{[k,j-1]}\},\ C_{[2,j-2]} + sa_{[2,j-1]}\} + q_{[2,j-1]}$. In Eq. (13), $C_{[l,j\text{-}1]} + sa_{[l,j]}$ is compared with $\max_{\forall k}\{\max\{r_{[k,j]},\ A_{[k,j-1]}\} + t_{[k,j]} + s_{[k,j]}\}$.

Table 2
Notation for AFb$|r_j,\ s_j|\gamma$.

Index
- $i,\ j$: Index of jobs where i and j = 1,2,…,n
- k: Index of components where k = 1,2,…,m
- l: Index of assembly and post processing stages where l = 1,…, b

Parameters
- m: No. of machines at stage one
- n: No. of jobs
- $r_{[k,j]}$: Release time of component k of job in position j
- $t_{[k,j]}$: Processing time of component k of job in position j in stage 1
- $t_{k,j}$: Processing time of component k of job j in stage 1
- $q_{[l,\ j]}$: Processing time of job in position j in stage l where $l > 1$
- $s_{[k,j]}$: Setup time of component k of job in position j
- $sa_{[l,j]}$: Setup time of job in position j in stage l where $l > 1$
- O: A very large positive number
- $q_{l,\min}$: $\min\{q_{[l,\ j]}\}\ \forall j$ and $l > 1$
- $q_{l,\max}$: $\max\{q_{[l,j]}\}\ \forall j$ and $l > 1$
- $r_{k,\max}$: $\max\{r_{[k,j]}\}\ \forall j$
- $r_{k,\min}$: $\min\{r_{[k,j]}\}\ \forall j$
- $t_{k,\min}$: $\max\{t_{[k,j]}\}\ \forall j$
- $t_{\min}$: min $\{t_{[k,j]}\}\ \forall k, j$
- $t_{\max}$: max $\{t_{[k,j]}\}\ \forall k, j$
- $sa_{\min,\ j}$: $\min\{sa_{l,\ j}\}\ \forall l > 1$
- $s_{\max,\ j}$: $\max\{s_{k,\ j}\}\ \forall k$
- $sa_{\max,\ j}$: $\max\{sa_{l,\ j}\}\ \forall l > 1$
- $s_{\min,\ j}$: $\min\{s_{k,\ j}\}\ \forall k$

Variables
- $C_{[j]}$: Completion time of job in position j in the last stage
- $\mathrm{C_{max}}$: Completion time of the last job in the last stage (makespan)
- $T_{[j]}$: Tardiness of job in position j in the last stage
- $A_{[k,j]}$: Completion time of component k of job in position j in stage 1
- $A_{max,j}$: $\max\{A_{[k,j]}\}\ \forall k$
- $C_{[l,j]}$: Completion time of job in position j in stage l where l = 2,…, b
- R $_{[l,j]}$: Starting time of job j in stage l where $l > 1$
- L $_{[l,j]}$: Release time of job j in stage l where $l > 1$
- TC: Total completion time
- $X_{[i,j,l]}$: Binary variable; 1 if job in position j is processed immediately after job i in stage l, 0 otherwise.

Scheduling Notations
- Л: A complete schedule
- Л*: Optimal schedule
- $M_{k,1}$: Machine k in stage 1
- M_l: Machine in stage l
- $J_{[1]}$,$J_{[2]}$,…,$J_{[n]}$: Jobs' sequence

Note: Index j can be used instead of i in above parameters, both referring to job position.

This is due to the fact that if job j-1 leaves stage l before job j becomes available at this stage, the resulting idle time is used to set up stage l for job j. The same analogy also holds for Eq. (15) where $C_{[l\ -1,\ j\text{-}1]}$ is compared with $C_{[l,j\text{-}2]} + sa_{[l,j\text{-}1]}$. According to the definition, $sa_{[l,\ j]}$ and $q_{[l,\ j]}$ are equal to zero for $l = 1$.

Here, we introduce some special cases for single objective AFb$|r_j, s_j|\gamma$. We show that there are exact methods with polynomial solutions for some of these cases. We later utilize properties for these special cases to generate efficient initial solutions for the general AFb$|r_j,\ s_j|\gamma$ model.

Here, we introduce some special cases for single objective AFb$|r_j, s_j|\gamma$. We also derive exact methods with polynomial solutions for these special cases. We later utilize properties of these special cases to generate efficient initial solutions for the general AFb model.

Theorem 1. *Given* $r_{k,\max} - r_{k,\min} \le t_{\ k,\min}\ \forall k$, $q_{l,\max} \le \{t_{\min}, q_{\text{l-1,min}}\}$ $\forall l = 2,\ldots,b$, *and* $sa_{max,\ j} \le s_{min,j}\ \forall j$, *makespan and total completion times can be expressed as:*

$$C_{\max} = \max_{\forall k}\left\{r_{[k,1]} + \sum_{i=1}^{n} t_{[k,i]} + \sum_{i=1}^{n} s_{[k,i]}\right\} + \sum_{l=2}^{b} q_{[l,n]} \tag{16}$$

$$TC = \sum_{j=1}^{n} \max_{\forall k}\left\{r_{[k,1]} + \sum_{i=1}^{j} t_{[k,i]} + \sum_{i=1}^{j} s_{[k,i]}\right\} + \sum_{j=1}^{n}\sum_{l=2}^{b} q_{[l,j]} \tag{17}$$

Proof. is given in Appendix-I. Let us decode the meaning of the first condition in Theorem 1. $r_{k,\max} - r_{k,\min} \le t_{\ k,\min}$ (or $r_{k,\max} \le t_{\ k,\min} + r_{k,\min}$) $\forall k$ ensures no machine is idle in the first stage as release times, $r_{k,\max}$, are not greater than job completion times, $t_{\ k,\min} + r_{k,\min}$, in the first stage. This is equivalent to no-idle condition in stage 1 where release times, except for the first job, $r_{[k,1]}$, have no impact on makespan and total completion time according to Eqs. (16) and (17). Corollary 1 explains how the first condition of Theorem 1 reduces the search space to permutation schedules. Condition, $q_{l,\max} \le t_{\min}$, enforces the processing time in the first stage, $t_{\min}$, to be greater than or equal to processing time of product in downstream stages, $q_{l,\max}$. The third condition, $sa_{max,\ j} \le s_{min,j}$, denotes that setup times in stage 1 is larger than setups in post-processing stages. As a result, the second and third conditions lead us to first stage bottleneck condition. Moreover, it is clear that in the absence of release time and setup times, conditions $r_{k,\max} - r_{k,\min} \le t_{\ k,\min}$ and $q_{\text{l-1,min}} \ge q_{l,\ \max}$ becomes trivial which simplifies Eqs. (16) and (17). This special case can occur frequently in industry, as the production process in the first stage can take significantly longer time than other post-processing stage. Another important result is that Eqs. (16) and (17) are independent from setups times in post processing stages.

Corollary 1. If $\forall k\ r_{k,\max} - r_{k,\min} \le t_{\ k,\min}$, the optimal schedule for AF$(m,1)|r_j,s_j|C_{\max}$ is a permutation schedule.

Proof is given in Appendix-II. Corollary 2 derives a polynomial optimal algorithm for AFb$|r_j,s_j|C_{\max}$ under the conditions of Theorem 1.

Corollary 2. The optimal solution for AFb$|r_j,s_j|C_{\max}$ under the conditions of Theorem 1 is given through the following procedure:

Step 1: Find the job that minimizes the maximum of $\{r_{k,i} + \sum_{i=1}^{n} t_{k,i} + \sum_{i=1}^{n} s_{k,i}\}_{\forall k}$. Assign it as the current first job in the sequence, curr $J_{[1]}$.

Step 2: Find the job with minimum total post-processing times in stages 2 through b, $\sum_{l=2}^{b} q_{l,\ j}$. Assign it as the current last job in the sequence, curr $J_{[n]}$. Given this job is different than curr $J_{[1]}$, optimal schedule is found where $J_{[1]}$ = curr $J_{[1]}$, $J_{[n]}$ = curr $J_{[n]}$, and jobs in between can be in any order. If the job with minimum $\sum_{l=2}^{b} q_{l,\ j}$ is the same as curr $J_{[1]}$, then;

Step 3:

3.1) Find the job with the second least summation of post-processing times and assign it as curr* $J_{[n]}$. Find the value of $C_{\max}$ from Eq. (16) for curr $J_{[1]}$ and curr* $J_{[n]}$.

3.2) Find the job with the second least summation of $\max_{\forall k}\{r_{[k,1]} + \sum_{i=1}^{n} t_{[k,i]} + \sum_{i=1}^{n} s_{[k,i]}\}$ and assign it as curr* $J_{[1]}$. Find the value of $C_{\max}$ from Eq. (16) for curr* $J_{[1]}$ and curr $J_{[n]}$. The schedule with minimum $C_{\max}$ value in steps 3.1 and 3.2 represents the optimal schedule.

It is clear that the term $\sum_{i=1}^{n} t_{[k,i]} + \sum_{i=1}^{n} s_{[k,i]}$ in Eq. (15) is independent of the order of jobs which leads us to another interesting property. That is, the sequence of jobs $J_{[2]}$ through $J_{[n\text{-}1]}$ can be shuffled without impacting $C_{\max}$ value. Hence, this problem has $(n$-2)! sequence with optimal makespan. Steps 1 through 3 in above corollary requires mn, $n(b$-1), and $mn + n(b$-1) calculations in the worst case scenario, respectively. Hence complexity is $O(n(b + m))$.

There is no polynomial algorithm that provide optimal solution for AFb$|r_j,s_j|$TC under the stated condition of Theorem 1. Consequently, we rely on heuristics to find efficient solutions for it.

Theorem 2 presents another special case with polynomial solutions.

Theorem 2. *If* $r_{k,\max}$ - $r_{k,\min} \le t_{\ k,\min}\ \forall k$, $q_{l,\max} \le q_{\ l\text{-}1,\min}\ \forall l = 2,\ldots,b$, $A_{max,j} < sa_{l,\ j} \le sa_{l-1,j}\forall j,l$, *then makespan and total completion times are expressed as:*

$$C_{max} = \sum_{j=1}^{n} sa_{[2,j]} + \sum_{j=1}^{n} q[2,j] + \sum_{l=b-1}^{b} q[l,n] \tag{18}$$

$$TC = \sum_{j=1}^{n} (n - j + 1)(sa_{[2,j]} + q_{[2,j]}) + \sum_{j=1}^{n}\sum_{l=b-1}^{b} q_{[l,j]} \tag{19}$$

Proof. is given in Appendix-III. Stated conditions in Theorem II imply that the second stage is the bottleneck in this system. Fortunately, there exist polynomial optimal algorithms for AFb$|\ r_j,s_j|\ C_{\max}$ and AFb$|r_j,s_j|$TC under the conditions of Theorem 2.

Corollary 3. The optimal solution for AFb$|r_j,s_j|C_{\max}$ under the conditions of Theorem 2 is given through the following procedure:

Step 1: Find the job that minimizes the maximum of $\{r_{k,i} + t_{k,i} + s_{k,i}\}_{\forall k}$. Assign it as the first job in the sequence, $J_{[1]}$.

Step 2: Find the job that has the minimum summation of post-processing times (stages 3 through b), $\sum_{l=3}^{b} q_{l,\ j}$. If this job is different than $J_{[1]}$, assign it as the last job in the sequence, $J_{[n]}$, otherwise, find the job with the second least summation of post-processing times and assign it as $J_{[n]}$.

$\sum_{j=1}^{n} q_{[2,j]}$ in Eq. (18) is independent of the order of jobs. Also, the sequence of jobs $J_{[2]}$ through $J_{[n\text{-}1]}$ can be shuffled without impacting $C_{\max}$ value. Hence, this problem has $(n$-2)! sequence with optimal makespan. Steps 1 and 2 in above corollary requires $2n$ *and* $n(b$-2) calculations in the worst case scenario, respectively. Hence complexity is $O(nb)$. As stated earlier, the first condition in Theorem 2 establishes no-idle condition for machines in stage 1 which limits the search space to permutation schedules.

Corollary 4. The optimal solution of AFb$|\ r_j,s_j|$TC under condition of Theorem 2 is given by sorting all jobs in non-decreasing order of $sa_{[2,j]} + q_{[2,j]}$. The last term in Eq. (19), $\sum_{j=1}^{n}\sum_{l=b-1}^{b} q_{[l,j]}$, has no impact on the optimal sequence. SPT rule ensures that the first term of the objective function $(\sum_{j=1}^{n} (n - j + 1)(sa_{[2,j]} + q_{[2,j]})$ stays minimum. The optimal solution in Corollary 4 can be found using above polynomial algorithm in $O(n)$ time.

Corollary 5. Given the bottleneck condition for stage u ($q_{l,\ \max} \le q_{\ u,\min}$, $A_{\max,j} < sa_{l,\ j} \le sa_{u,\ j}\forall j$ and $\forall l$ & $u = 2,\ldots,b$ where $\forall l \ne u$) and no-idle condition for stage 1 ($r_{k,\max}$ - $r_{k,\min} \le t_{\ k,\min}\ \forall k$), the makespan and total completion times are expressed as:

$$C_{max} = \sum_{j=1}^{n} sa_{[u,j]} + \sum_{j=1}^{n} q_{[u,j]} + \sum_{l=b-1}^{b} q_{[l,n]} \quad (20)$$

$$TC = \sum_{j=1}^{n} (n-j+1)(sa_{[u,j]} + q_{[u,j]}) + \sum_{j=1}^{n} \sum_{l=b-1}^{b} q_{[l,j]} \quad (21)$$

The proof is straight forward as it can be derived directly by changing bottleneck condition of Theorem 2 for any post-processing stage. According to Eq. (20), only processing and setup times for stage u together with processing time of the last job in the last two stage is accountable for makespan. The optimal solutions for makespan and total completion times are obtained by simple modification of polynomial algorithms in Corollary (4) and (5).

Theorem 3. If $r_{k,\max} - r_{k,\min} \leq t_{k,\min}\ \forall k$ and $\{t_{\max,}\ q_{l-1,\max}\} \leq q_{l,\ \min}\ \forall l=2,\ldots,b$, and $s_{\max,j} < sa_{l-1,j} < sa_{l,\ j} < A_{\min,j}\ \forall j,l$, then makespan and total completion times can be expressed as:

$$C_{max} = \max_{\forall k}\{A_{[k,1]}\} + \sum_{l=2}^{b-1} q_{[l,1]} + \sum_{j=1}^{n} q_{[b,j]} + sa_{[b,n]} \quad (22)$$

$$TC = n \max_{\forall k}\{A_{[k,1]}\} + n\sum_{l=2}^{b-1} q_{[l,1]} + \sum_{j=1}^{n} (n-j+1) q_{[b,j]} + \sum_{j=1}^{n} sa_{[b,j]} \quad (23)$$

Proof. is given in Appendix-IV. Note that for any AFb without setup and release times, conditions 1 and 3 become trivial. The optimal solution for AFb$|r_j,s_j|C_{max}$ under the conditions of Theorem 3 is obtained through the following Corollary:

Corollary 6. The optimal solution for AFb$|r_j,s_j|C_{max}$ under the conditions of Theorem 3 is given through the following procedure:

Step 1: Find the job that minimizes the $\max_{\forall k}\{A_{[k,1]}\} + \sum_{l=2}^{b-1} q_{[l,1]}$. Assign it as the current first job in the sequence, curr $J_{[1]}$.

Step 2: Find the job that has the minimum setup time $sa_{l,j}$ in stages 2 through b.

Step 3: If the job in Step 2 is different than curr $J_{[1]}$, then:

Step 3.1: Assign it as the last job in the sequence, $J_{[n]}$ and $J_{[1]} :=$ curr $J_{[1]}$. Otherwise, find the job with the second least summation of minimum setup time $sa_{l,j}$ and temporarily assign it as $J_{[n,\alpha]}$.

Step 3.2: Find the job with the second least summation in Step 1 and temporarily assign it as $J_{[1,\alpha]}$. Assign the job found it Step 2 as $J_{[n]}$ and find the value of Eq. (22) for this assignment. The minimum value of Eq. (22) for steps 3.1 and 3.2 represents the optimal schedule.

Corollary 7. The optimal solution of AFb$|\ r_j,s_j|$TC under condition of Theorem 3 is given by the following algorithm:

Step 1: Sort all jobs in non-decreasing order of $q_{[b,\ j]}$. The sequence is shown as $\prod best := \prod SPT$ and Best $= \sum^{n}_{j=1} C_{[j]} := \sum^{n}_{j=1} C_{[j]}$. This sequence ensures that the second term of the objective function $(\sum_{j=1}^{n}(n-j+1)q_{[b,j]})$ is minimum.

Step 2: For all jobs $j = 2,\ldots, n$, generate a candidate sequence by switching $J_{[j]}$ to $J_{[1]}$. Call it $\prod current$ and set curr: $= \sum^{n}_{j=1} C_{[j]}$

Step 3: If curr < Best, set Best = curr and $\prod best := \prod current$.

The optimal solution in Corollary 5 can be found using above polynomial algorithm in O(n log n) time.

Table 3 summarizes optimal algorithms for special cases of AFb$|\ r_j,s_j|\ \gamma$ that are developed in this article.

3.2.1. Numerical example

Example 1. Consider the following example with four jobs and four stages and two machines in the first stage. The goal is to minimize the makespan. The release, processing, setup, and assembly times of jobs are given in Table 4. Two tuple format represent the release, processing, or setup times of the first and the second component of jobs in stage one, or processing and setup times in other stages. The condition of Corollary 1 holds for this example as $r_{k,\max} - r_{k,\min} \leq t_{k,\min},\ \forall k$. Therefore, the optimal solution is a permutation schedule. Besides, Theorem 3 can be applied to this example as all the corresponding conditions holds. According to Corollary 6, Job 3 minimizes the maximum of $\max_{\forall k}\{r_{k,j} + t_{k,\ j}\} + \sum_{l=2}^{b-1} q_{l,j} = 39$ and should be scheduled first. Also job 2 has minimum $sa_{b,j}$. Thus, it will be scheduled as the last job. As a result, two optimal schedule exist for this example: $J_1^* = \{J_3,J_4,J_1,J_2\}$ and $J_2^* = \{J_3,J_1,J_4,J_2\}$ is an optimal schedule with makespan of 39 + 69 + 6 = 114.

Table 3
Proposed Polynomial Optimal Algorithms with their Conditions for AFb$|\ r_j,s_j|\ \gamma$

Conditions	Optimal Algorithm for Minimizing C_{max}
$r_{k,\max} - r_{k,\min} \leq t_{k,\min}\ \forall k$ $q_{l,\max} \leq \{t_{\min,}\ q_{l-1,\min}\}\ \forall l=2,\ldots,b$ $sa_{max,\ j} \leq s_{min,j}\ \forall j$	Corollary 2
$r_{k,\max} - r_{k,\min} \leq t_{k,\min}\ \forall k$ $q_{l,\max} \leq q_{\ l-1,\min}\ \forall l=2,\ldots,b$ $A_{max,j} < sa_{l,\ j} \leq sa_{l-1,j}\ \forall j,l$	Corollary 3
$r_{k,\max} - r_{k,\min} \leq t_{k,\min}\ \forall k$ $\{t_{\max,}\ q_{l-1,\max}\} \leq q_{l,\ \min}\ \forall l=2,\ldots,b,$ $s_{\max,j} < sa_{l-1,j} < sa_{l,\ j} < A_{\min,j}\ \forall j,l$	Corollary 6
Conditions	Optimal Algorithm for Minimizing TC
$r_{k,\max} - r_{k,\min} \leq t_{k,\min}\ \forall k$ $q_{l,\max} \leq q_{\ l-1,\min}\ \forall l=2,\ldots,b$ $A_{max,j} < sa_{l,\ j} \leq sa_{l-1,j}\ \forall j,l$	Corollary 4
$r_{k,\max} - r_{k,\min} \leq t_{k,\min}\ \forall k$ $\{t_{\max,}\ q_{l-1,\max}\} \leq q_{l,\ \min}\ \forall l=2,\ldots,b,$ $s_{\max,j} < sa_{l-1,j} < sa_{l,\ j} < A_{\min,j}\ \forall j,l$	Corollary 7

Table 4
Release, processing, setup, and assembly times of jobs.

Job	1	2	3	4
$r_{k,j}$ (Stage 1)	{7,8}	{9,2}	{5,4}	{4,3}
$t_{k,j}$ (Stage 1)	{5,7}	{8,6}	{7,10}	{5,9}
$s_{k,j}$ (Stage 1)	{4,4}	{3,3}	{6,6}	{2,2}
$\{q_{2,j}, sa_{2,j}\}$ (Stage 2)	{11,5}	{12,4}	{11,7}	{13,3}
$\{q_{3,j}, sa_{3,j}\}$ (Stage 3)	{14,6}	{13,4}	{14,8}	{15,5}
$\{q_{4,j}, sa_{4,j}\}$ (Stage 4)	{15,8}	{18,6}	{19,9}	{17,10}

Example 2. Consider the following example with four jobs and four stages and two machines in the first stage. The goal is to minimize the makespan. The release, processing, setup, and assembly times of jobs are given in Table 5. The condition of Corollary 1 holds for this example as $r_{k,\max} - r_{k,\min} \leq t_{\ k,\min},\ \forall k$. Therefore, the optimal solution is a permutation schedule. Besides, Theorem 1 can be applied to this example as all the corresponding conditions holds. According to

Table 5
Release, processing, assembly, and setup times of jobs.

Job	1	2	3	4
$r_{k,j}$ (Stage 1)	{7,8}	{9,2}	{4,3}	{5,6}
$t_{k,j}$ (Stage 1)	{5,7}	{8,6}	{7,10}	{5,9}
$s_{k,j}$ (Stage 1)	{6,4}	{3,5}	{5,6}	{7,3}
$\{q_{2,j}, sa_{2,j}\}$ (Stage 2)	{4,4}	{4,3}	{5,4}	{3,2}
$\{q_{3,j}, sa_{3,j}\}$ (Stage 3)	{4,3}	{3,2}	{4,1}	{2,3}
$\{q_{4,j}, sa_{4,j}\}$ (Stage 4)	{2,2}	{1,2}	{3,5}	{2,1}

Corollary 2, Job 3 minimizes the maximum of $\{r_k + \sum_{i=1}^{n} t_{k,i} + \sum_{i=1}^{n} s_{k,i}\}$ $\forall k = 53$. Also job 4 minimizes $\sum_{l=2}^{4} q_{l,j} = 7$. Thus, Jobs 3 and 4 are scheduled as the first and the last jobs, respectively. Two optimal schedule exist for this example: $\Pi_1^* = \{J_3, J_1, J_2, J_4\}$ and $\Pi_2^* = \{J_3, J_2, J_1, J_4\}$ is an optimal schedule with makespan of $53 + 7 = 60$.

If the first stage is dominant, Corollary 2 gives the optimal result for C_{max}. If the second stage is dominant, Corollary 3 and 4 gives a polynomial algorithm for C_{max} and TC and if the last stage is dominant, Corollary 6 and 7 gives a polynomial algorithm for C_{max} and TC.

4. Proposed lower bounds for C_{max}

We propose the following lower bounds for AFb$|r_j, s_j|C_{max}$ which is the modification of lower bounds proposed by Komaki and Keyvanfar [16].

4.1. When the first stage is bottleneck

Under this condition, the first job has the minimum release time. After processing all jobs in the first stage, this stage will be idle and the last job is the job with minimum summation of post-processing times.

$$LB_1(C_{max}) = \min_{\forall k,j} r_{[k,j]} + \max_{\forall k}\left\{\sum_{j=1}^{n} t_{[k,j]} + \sum_{j=1}^{n} s_{[k,j]}\right\} + \min_{\forall j} \sum_{l=2}^{b} \{q_{[l,j]} + sa_{[l,j]}\}$$

4.2. When stage u is bottleneck (u = 2,…, b)

When stage u ($u = 2,\ldots, b$) is bottleneck, the starting time to process the first job at stage u is $\min_{\forall j}\{\max_{\forall k}\{r_{[k,j]} + t_{[k,j]} + s_{[k,j]}\} + \sum_{l=2}^{u-1}(q_{[l,j]} + sa_{[l,j]})\}$. Thereafter, the machine at stage u will be busy until all jobs are processed. After all jobs are processed in stage u, this stage will be idle and the last job is the job with minimum summation of post-processing times for stages $u + 1$ through b. LB_u shows the extreme situations where stage ($u = 2,.., b$) is bottleneck.

$$LB_u(C_{max}) = \min_{\forall j}\left\{\max_{\forall k}\{r_{[k,j]} + t_{[k,j]} + s_{[k,j]}\} + \sum_{l=2}^{u-1}(q_{[l,j]} + sa_{[l,j]})\right\} + \sum_{j=1}^{n}(q_{[u,j]} + sa_{[u,j]}) + \min_{\forall j}\sum_{l=u+1}^{b}\{q_{[l,j]} + sa_{[l,j]}\}$$

$$LB(C_{max}) = \max\{LB_1, LB_2, \cdots, LB_b\}$$

$$LB_l(C_{max}) = \sum_{j=1}^{n}\min_{\forall j}\left\{\max_{\forall k}\{r_{[k,j]} + t_{[k,j]} + s_{[k,j]}\} + \sum_{j=1}^{n}\sum_{l=2}^{u-1}(q_{[l,j]} + sa_{[l,j]})\right\} + n\sum_{j=1}^{n}(q_{[u,j]} + sa_{[u,j]}) + \sum_{j=1}^{n}\min_{\forall j}\sum_{l=u+1}^{b}\{q_{[l,j]} + sa_{[l,j]}\}$$

$$LB_{C\,max} = \max\{LB_1, LB_2, ..., LB_b\}$$

5. Solution algorithms

In this section, we develop solution algorithms for AF2b$|r_j, s_j|C_{max}$ and leave further analysis of AF2b$|r_j, s_j|$TC for future studies. This helps us to analyze C_{max} problem in more depth as C_{max} analysis can shed the light for researchers who explore solution algorithms for TC due to close relation between C_{max} and TC. The NP-hardness of AF2b$|r_j, s_j|\gamma$, prompt us to apply metaheuristics in order to generate promising solutions. In this section, we first develop simple heuristics for the addressed problem. These heuristics will be used to generate the initial solutions for metaheuristics that will be presented later in this section. We will also provide comprehensive comparison of the presented heuristics.

5.1. Constructive heuristics

We propose new constructive heuristics based on the ideas discussed in the previous section. Since AFb$|r_j, s_j|\gamma$ has never been studied before, the presented heuristics will be the first attempt to solve this NP-hard problem. Eqs. (24)–(26) are derived by adding setups times to heuristics that were proposed by Tozkapan et al. [39] and expanding them for more than two stages. $PTF_{ij} = t_{ij}$ represents processing times of job j on machine i in the first stage and $PTS_{lj} = q_{lj}$ shows the processing time of job j in stage l. In this procedure, three sequences for each job are obtained by sorting the jobs in non-decreasing order of following indices. Then, the sequence with the lowest total completion time is chosen:

$$I_1(j) = \min_{\forall k}\{t_{kj} + s_{kj}\} + \min_{\forall l}\{q_{l,j} + sa_{l,j}\} \quad (24)$$

$$I_2(j) = 1/(m + b - 1)\left[\sum_{k=1}^{m}(t_{kj} + s_{kj}) + \sum_{l=2}^{b}(q_{l,j} + sa_{l,j})\right] \quad (25)$$

$$I_3(j) = \max_{\forall k}\{t_{kj} + s_{kj}\} + \max_{\forall l}\{q_{l,j} + sa_{l,j}\} \quad (26)$$

$I_1(j)$, $I_2(j)$, and $I_3(j)$ are the minimum, average, and maximum processing times of job j, respectively. In addition, Eqs. (28)–(32) are derived from the dispatching rules proposed by Komaki and Kayvanfar [16]. These indices includes release time and setup time of jobs and expands Komaki and Kayvanfar [16]'s indices for more than two stages. Similar to the first three groups, these indices are sorted in non-decreasing order.

$$I_4(j) = \max_{\forall k}\{r_{k,j} + t_{k,j} + s_{k,j}\} + \sum_{l=2}^{b}\{q_{l,j} + sa_{k,j}\} \quad (27)$$

$$I_5(j) = \sum_{k=1}^{m}(r_{k,j} + t_{k,j} + s_{k,j})/m + \max_{\forall l}\{q_{l,j} + sa_{l,j}\} \quad (28)$$

$$I_6(j) = \sum_{k=1}^{m}(r_{k,j} + t_{k,j} + s_{k,j})/m + \sum_{l=2}^{b}\{q_{l,j} + sa_{k,j}\} \quad (29)$$

$$I_7(j) = \min_{\forall k}\{r_{k,j}\} + \max_{\forall k}\{t_{k,j} + s_{k,j}\} + \sum_{l=2}^{b}\{q_{l,j} + sa_{k,j}\} \quad (30)$$

$$I_8(j) = \max_{\forall k}\{r_{k,j}\} + t_{k*,j} + \sum_{l=2}^{b}\{q_{l,j} + sa_{k,j}\} \quad (31)$$

$$I_9(j) = \max\left\{\max_{\forall k}\{r_{k,j}\}, \max_{\forall k}\{t_{k,j} + s_{k,j}\}\right\} + \sum_{l=2}^{b}\{q_{l,j} + sa_{k,j}\} \quad (32)$$

where k^* in (31) is the first stage machine with maximum $\sum_{j=1}^{n}\{t_{k,j} + s_{k,j}\}$ $\forall I$. Note that jobs are ordered in non-decreasing ordering for all dispatching rules.

5.2. Metaheuristic algorithms

In this section, we present implementation of two metaheuristics, GVNS and Grey Wolf Optimizer, on the addressed problem. These two algorithms have been applied for solving different assembly flowshop problems. For example, Komaki et al. [15] applied GVNS for solving distributed permutation flowshop scheduling problem. Komaki and Keyvanfar [16] utilized Grey Wolf optimizer to find efficient solutions for AF(m,1)$|r_j|$ C_{max}.

5.2.1. General variable neighborhood search algorithm

The quality of heuristic solutions can be improved by allowing diversification in the solution space. In order to achieve this goal, we apply an extension of the Variable Neighborhood Search, called General Variable Neighborhood Search (GVNS), which has been successfully applied to solve various scheduling problems. The basic concept of the VNS is using local search algorithm with systematic switches between neighborhood structures in order to avoid local optima. The main difference between VNS and GVNS is that VNS uses a local search algorithm to improve the solution while GVNS uses a Variable Neighborhood Decent (VND) algorithm. Two main components of GVNS algorithm are "shaking" and "local search" where the shaking procedure perturbs the current solution to escape from the current local optima and the local search explores the neighborhood of the current solution in a systematic manner. We utilize GVNS due to its effectiveness and the ability to improve the solutions generated in Section 5.1.

GVNS starts with an initial solution x generated by heuristics in Section 5.1. Then, the shaking procedure and VND are applied to the solution in search for a better neighborhood of the current solution. The details of this algorithms including the intensity level of ω is decided in the same manner as explained by Komaki and Malakooti [18]. In a nutshell, if the solution obtained by VND algorithm outperforms the current solution, the shaking procedure will be one the lowest perturbation level (strong intensification). Otherwise, it will apply a strong perturbation. This procedure continues until all predefined neighborhoods have been explored.

5.2.2. Grey wolf optimizer algorithm

As the second metaheuristic, we utilize a swarm intelligence technique called, Grey Wolf Optimizer (GWO), to compare the quality of its solutions with GVNS. GWO algorithm is inspired by hunting behavior of wolves where each individual has a defined role with well-defined hierarchical relationship in wolf pack, see Mirjalili et al. [25]. In GWO, it is assumed that positions of leaders in a wolf pack (called alpha, beta and delta) are saved and other wolves have to update their positions based on the position of leaders. In this study, we use the sequence generated by heuristics in Section 5.1 as the starting point for GWO.

We utilize two factors, A and C, to help the algorithm escape from local minimum. Values for A are linearly decreased over the course of the algorithm while C is not. This assures that half of the iterations explore the space for possible superior solutions and the other half converges to the promising solutions. We also integrate mutation to imitate the life cycle of grey wolves.

Encircling process are modeled by Eqs. (33) and (34) where $X(t)$ and $Xp(t)$ represent the position of the prey and wolf at iteration t and $X(t+1)$ represent the new position of the wolf.

$$D = |C.X_p(t) - X(t)| \tag{33}$$

$$X(t+1) = X_p(t) - A.D \tag{34}$$

Vectors C and A in above equations are defined as $C = 2r_2$ and $A = 2ar_1\text{-}a$. r_1 and r_2 are random vectors over [0,1] where a linearly decreases over the range of [0,2] using $a = 2\text{–}2^*t/It_{max}$ and It_{max} represents maximum number of iterations in the algorithms. The attacking stage occurs when $|A| < 1$ and as the result, wolves becomes closer to the position of prey, $X_p(t)$. On the other hand, $|A| > 1$ results in wolves to explore other preys. The exact position of the prey, the optimal solution, is unknown. However, α, β, and δ, are the closest known answers to the optimal solution. Considering, X_1, X_2, and X_3 as the new positions for α, β, and δ, other wolves update their position through the following equation:

$$X(t+1) = (X_1 + X_2 + X_3)/3 \tag{35}$$

In the following, we summarize each step of the GWO algorithm.

Step 1: Consider $X_i(t)$ as the position of the wolf where $i = 1,\ldots, N_{max}$. N_{max} represents the number of wolves in the group. Each wolf represents the jobs sequence and the value generated by the best heuristic in Section 5.1 is used as the starting point.

Step 2: Fitness of the solution is determined by: $F(X_i) = \Omega - C_{max}(X_i)$ $i = 1,2,\ldots, N_{max}$, where Ω is a large positive number and $C_{max}(X_i)$ is calculated based on Eq. (13).

Step 3: We find α, β, and δ by sorting the solutions in non-decreasing order of their fitness values. Then, the first sequence is X_α and the second and the third sequences are X_β and X_δ, respectively. We apply local search on these sequences in an attempt to achieve solutions with higher quality. The proposed local search selects a job from current sequence and reinserts it in other available positions in an attempt to find sequences with improved fitness values. The orders of the X_α, X_β and X_δ are updated to reflect the improvement during the local search.

Step 4: Position of subordinate wolfs are updated through Eq. (35).

Step 5: Fitness of each wolf are recalculated using the Equation in Step 2 and α, β, and δ are updated.

Step 6: Repeat steps 4–6 until the stopping condition is met.

A wolf's fitness (solution) determines its role in the group. Updating position of wolves can impact the position and subsequently, the role of wolves in the group. For further details of adjusting GWO for AF problems, readers are referred to Komaki and Keyvanfar [16].

6. Computational experiment

The performance of proposed heuristic algorithms is measured through randomly generated instances. The number of jobs (n) are set at 20, 40, 60, and 80. The number of the machines at the first stage (m) are set at 2, 4, 6, and 8 and the number of stages (b) are set as 3, 4, 5, and 6. Also, the release, setup, processing, and assembly times are generated randomly using uniform distributions for sixteen problem sets, see Table 6. Totally, $4 \times 4 \times 4 \times 16 = 1024$ instances are generated with 30 replications of each instance, i.e., in total 30,720 test problems are run and the average performance of the algorithms are compared. All algorithms are run in MATLAB2016 on a laptop with a 2.2 GHz Intel Core i5-5200 processor and 8 GB RAM.

As is the case with heuristic methods, we limit our solution space to permutation schedules. We also make sure that none of the problem sets in Table 6 fulfills the conditions of Theorems 1 through 3. Due to probabilistic nature of GWO, each problem is run 30 times and the average of results is considered as the performance of the algorithm. We tried different parameter values to tune the metaheuristics. Population size and maximum number of iterations are reported in Table 7. The optimal values are shown in the right two columns.

Performance of the Lower Bounds: The deviation of the LB (DVL) from the best known solution of the algorithms is measured as $DVL = (Sol^* - LB_{cmax})/Sol^* * 100$ where $LB_{cmax} = \max\{LB_1, LB_2, \ldots, LB_b\}$ and Sol^* is the best obtained solution by the algorithms. Fig. 3

Table 6
Release, setup, processing, and assembly times of jobs.

Problem Set	Range of r_{ij}	Range of s_{ij} and sa_{ij}	Range of t_{ij}	Range of q_{ij}
Set 1	U[0,200]	U[0,100]	U[0,200]	U[0,200]
Set 2	U[0,100]	U[0,100]	U[0,100]	U[0,200]
Set 3	U[0,100]	U[0,100]	U[0,200]	U[0,100]
Set 4	U[0,200]	U[0,100]	U[0,100]	U[0,100]
Set 5	U[0,100]	U[0,100]	U[0,200]	U[0,200]
Set 6	U[0,200]	U[0,100]	U[0,200]	U[0,100]
Set 7	U[0,200]	U[0,100]	U[0,100]	U[0,200]
Set 8	U[0,100]	U[0,100]	U[0,100]	U[0,100]
Set 9	U[0,200]	U[0,200]	U[0,200]	U[0,200]
Set 10	U[0,100]	U[0,200]	U[0,100]	U[0,200]
Set 11	U[0,100]	U[0,200]	U[0,200]	U[0,100]
Set 12	U[0,200]	U[0,200]	U[0,100]	U[0,100]
Set 13	U[0,100]	U[0,200]	U[0,200]	U[0,200]
Set 14	U[0,200]	U[0,200]	U[0,200]	U[0,100]
Set 15	U[0,200]	U[0,200]	U[0,100]	U[0,200]
Set 16	U[0,100]	U[0,200]	U[0,100]	U[0,100]

Table 7
Optimal parameters of the meta heuristic algorithms.

Parameter	Description	Tested range	GVNS	GWO
N_{max}	Population size	100–500 with increment of 100	200	500
It_{max}	Max. of iterations	50–500 with increment of 50	450	400

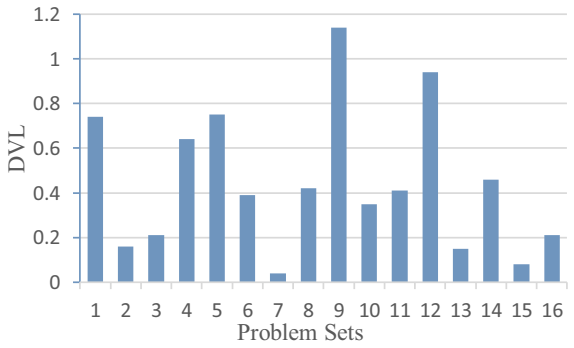

Fig. 3. Average DVL of dispatching rules and heuristic algorithms.

presents the DVL for 16 problem sets in Table 6. The best lower bound performance belongs to problem sets 7 and 15 while the worst lower bound performance belongs to sets 9 and 12. On average, the performance of the proposed LB is 0.44%.

6.1. Comparisons of the heuristic algorithms

In this section, we analyze the efficiency of heuristic proposed in Section 5.1. The heuristics proposed in this article results in very fast (less than 0.5 s) solutions. After running 30 replication of each instance, the solutions of heuristics are compared to each other. Fig. 4 shows the average and ranking of DVL for nine heuristic algorithms. It can be observed that on average, I_5 has the best performance followed by I_9 and I_8, and I_6 have the worst performance, followed by I_2 and I_1.

We use average relative percentage deviation, RPD = $(Alg_{sol} - LB)/LB \times 100$, to measure the efficiency of proposed heuristics. Fig. 5 shows the performance of each heuristic algorithm by measuring average RPD of dispatching rules based on number of jobs. I_5 and I_9 are the best performing algorithms and their performances improve as the number of jobs increases. The worst performance belongs to I_1 followed by I_6. Performance of I_1 and I_3 deteriorates by increasing the number of jobs while the performance of other algorithms improves as the number of the jobs increases. The main reason for I_1's underperformance is due to its oversimplified formula. I_1 in Eq. (24) ignores jobs' real times and only relies on the minimum of processing, and setup times over all components and stages.

Fig. 6 represents the performance of heuristics based on the number

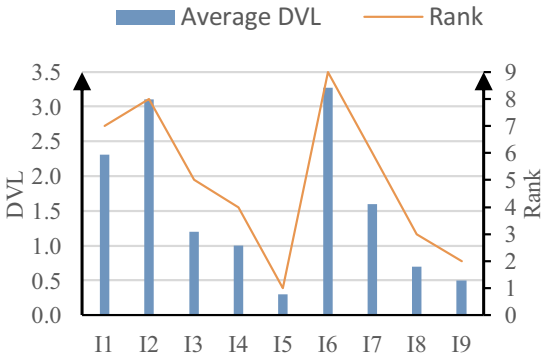

Fig. 4. Overall performance of the heuristic algorithms and their rank.

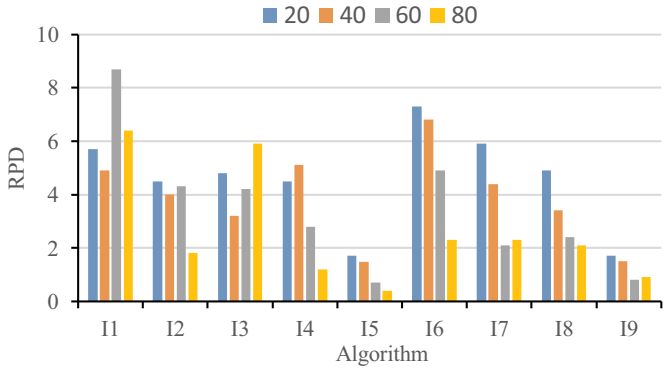

Fig. 5. Performance comparison of heuristic algorithms based on number of jobs.

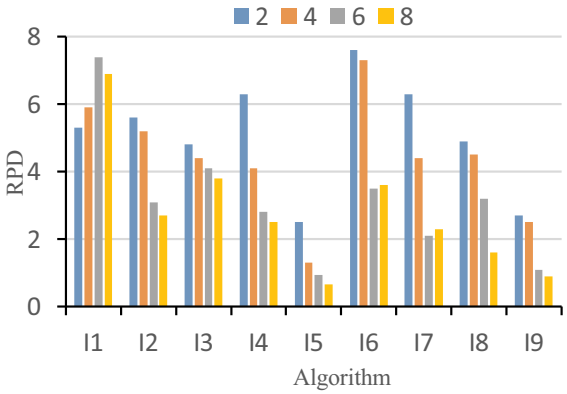

Fig. 6. Performance comparison of heuristic algorithms based on number of machines in the first stage.

of machines in the first stage. Algorithms with the best performances, I_5 and I_9, and the worst performances, I_1 and I_6, are similar to the ones in Fig. 5. As was the case in Fig. 5, the performance of heuristics except for I_1 and I_3 improves as the number of the machines increases. It can be observed that I_4 and I_7's average performances are almost identical with both having sharp increase in performance as the number of machines increases.

Fig. 7 depicts the performance of the heuristic algorithms based on number of stages. On average, I_5 followed by I_3, and I_9 yields the best performances and the worst performances belong to I_6 and I_1. In general, the performance of I_2 through I_6 and I_9 improves as number of stages increases. Comparing Figs. 5–7 reveals that performance of presented algorithms is more sensitive to the number of machines in the first stage than number of jobs or number of stages.

Fig. 8 depicts the performance of the heuristics based on the sixteen

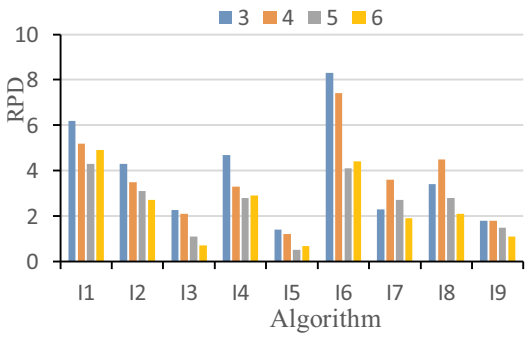

Fig. 7. Performance comparison of heuristic algorithms based on number of stages.

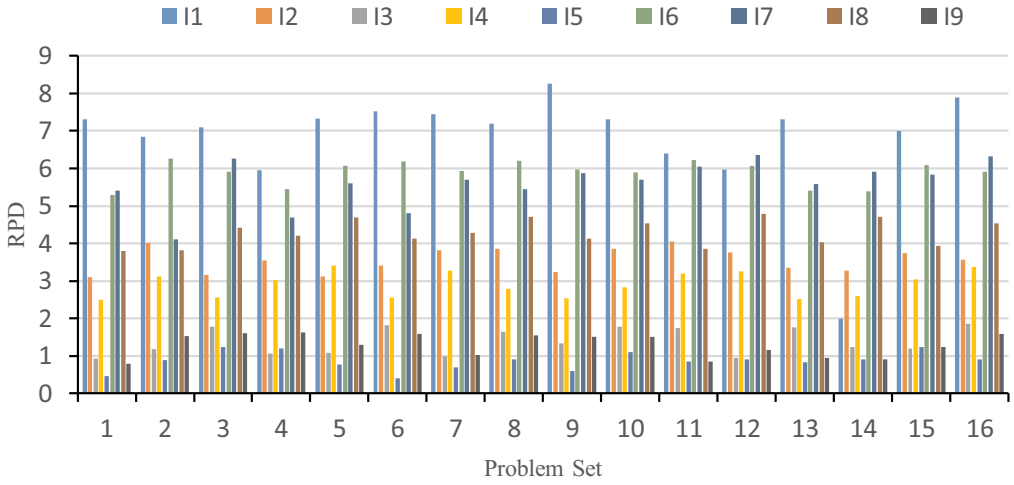

Fig. 8. Performance comparison of heuristic algorithms based on Problem Sets in Table 6.

problem sets defined in Table 6. The behavior of the heuristics confirms the ones observed in Figs. 3 through 6, with heuristic 5 yielding the best performance followed by heuristics I_9 and I_3 and heuristics I_1, I_6, I_7, and I_7 yielding the worst performances, respectively.

6.2. Statistical analysis of heuristic algorithms

In order to evaluate the significance of RPD differences in the proposed nine heuristics, we perform statistical tests of the RPD results in Fig. 8 using SPSS software. Statistical tests also help us to validate the conclusions we drew in Section 6.1. A one-way analysis of variance (ANOVA) test requires normality, independence of residuals, and homoscedasticity assumptions. First, we check the normality of obtained results through Shapiro-Wilk test. The selection of Shapiro-Wilk test is due to the limited sample size (16 samples for each heuristic). According to Table 8, p-value (sig.) for heuristics 1, 3, 4, 6, and 9 is smaller than the significance level, 0.05. Therefore, the RPD values for these heuristics are not normally distributed.

As a result, we apply a non-parametric technique such as Kruskal-Wallis one-way analysis of variance by rank to check whether the distribution of the samples is the same or not. It should be pointed out that the non-parametric tests do not require explicit conditions related to the sample of data. Rejecting null hypothesis of the test implies that at least one of the instances stochastically dominates other instances. Table 9 shows the mean rank of heuristics. The null hypothesis of the Kruskal-Wallis test is that the mean ranks of the groups are the same, while the alternative hypothesis is that at least one mean rank is different than the mean rank of heuristics.

Table 8
Shapiro-Wilk Normality Test.

Tests of Normality				
	Heuristic	Statistic	Shapiro-Wilk Df	Sig.
RPD	1	0.675	16	0.000
	2	0.924	16	0.193
	3	0.859	16	0.019
	4	0.885	16	0.046
	5	0.935	16	0.289
	6	0.848	16	0.013
	7	0.904	16	0.092
	8	0.919	16	0.164
	9	0.859	16	0.018

*This is a lower bound of the true significance.
a. Lilliefors Significance Correction.

Table 9
The ranks of groups in Kruskal-Wallis Test.

Ranks			
	Heuristic	N	Mean Rank
RPD	I1	16	130.03
	I2	16	72.94
	I3	16	32.88
	I4	16	59.06
	I5	16	12.50
	I6	16	116.75
	I7	16	110.84
	I8	16	89.38
	I9	16	28.13
	Total	144	

Table 10
Kruskal-Wallis Test.

	RPD
Kruskal-Wallis H	131.896
df	8
Asymp. Sig.	0.000

According to Table 9, I5 has the smallest (best) mean rank among other employed heuristics while I1 yields the largest (worst) result. These results are in accordance with the results observed in Fig. 8. In general, it is observed that I1, I6, and I7 have considerably larger mean rank comparing to the others. Table 10 shows the result of Kruskal-Wallis test. Table 10 shows that the P-value (sig.) is smaller than the significance level ($0.00 < 0.05$). This means that above mean rank values are not the same and the difference observed is beyond chance.

In order to figure out which heuristic(s) are statistically different, we perform pairwise Kruskal-Wallis test for all pairs of 9 heuristics. Table 11 shows the pairs with significant difference:

Table 11 tests the hypothesis that whether the RPD distributions of two heuristics are the same or not. The main trend observed is that any one of heuristics I1, I6 and I7 have significantly different mean rank (i.e. worse performance) than the rest of heuristics. These results are confirm the underperformance of I1, I6 and I7. On the other hand, Table 11 shows that heuristics I9 and I5 have significantly different (i.e. better performance) comparing with other heuristics.

Table 11
Pairwise Kruskal-Wallis Test for all Heuristics.

Sample 1-Sam...	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj.Sig.
3-1	97.156	14.748	6.588	.000	.000
4-1	70.969	14.748	4.812	.000	.000
5-1	117.531	14.748	7.970	.000	.000
3-6	-83.875	14.748	-5.687	.000	.000
9-1	101.906	14.748	6.910	.000	.000
3-7	-77.969	14.748	-5.287	.000	.000
5-6	-104.250	14.748	-7.069	.000	.000
5-7	-98.344	14.748	-6.668	.000	.000
5-8	-76.875	14.748	-5.213	.000	.000
9-6	88.625	14.748	6.009	.000	.000
9-7	82.719	14.748	5.609	.000	.000
5-2	60.438	14.748	4.098	.000	.001
9-8	61.250	14.748	4.153	.000	.001
4-6	-57.688	14.748	-3.912	.000	.003
2-1	57.094	14.748	3.871	.000	.004
3-8	-56.500	14.748	-3.831	.000	.005
4-7	-51.781	14.748	-3.511	.000	.016

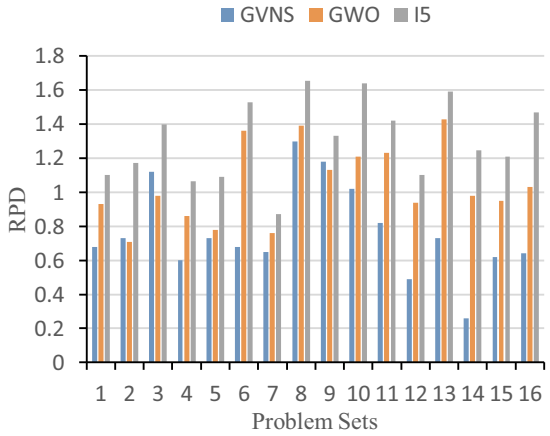

Fig. 9. (a). Average RPD based on Problem set. **(b)**. Average RPD based on number of jobs.

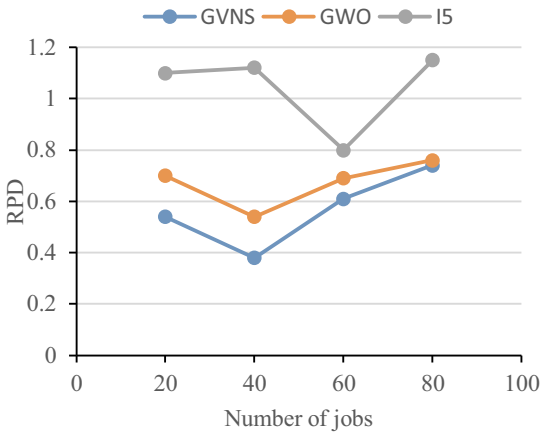

Fig. 9. (*continued*)

6.3. Comparisons of the metaheuristic algorithms

In this section, we analyze the efficiency of GVNS and GWO metaheuristics. We embed the heuristic solution of Section 5.1 as the promising starting point for GVNS and GWO. We utilize different values for the maximum number of iterations as stopping criteria in GVNS to investigate several scenarios ranging from an average computation time of less than a second to 10 s. To provide a fair comparison between GVNS and GWO, we let the algorithms run for at most 10 s.

Fig. 9(a) shows the average RPD for GVNS, GWO, and the best heuristic algorithm, I_5, based on different problems sets. On average, the performance for GVNS is 0.77% while this value for GWO and I_5 is 1.04% and 1.31%, respectively. Also, GVNS yields the best outcome in problem sets 12 and 14, while its worst performance is obtained in problem set 8. Also, GVNS is performing better than GWO in all problem sets except problem set 2.

Fig. 9(b) depicts the average RPD of algorithms based on "number of jobs" where the trends for GVNS and GWO are similar for all job sizes. This is due to the fact that these two algorithms use the results for same heuristics as starting points. It is worth noting that for all job sizes, GVNS outperforms the GWO algorithm. However, as the number of jobs increases, the average RPD for GVNS and GWO converges.

Table 12
Tests of Normality.

		Shapiro-Wilk		
	Heuristic	Statistic	df	Sig.
RPD	1.00	0.926	16	0.214
	2.00	0.940	16	0.353
	3.00	0.953	16	0.532

Table 13
Test of Homogeneity of Variance.

	Levene Statistic	df1	df2	Sig.
RPD	Based on Mean	0.047	2	45	0.954
	Based on Median	0.066	2	45	0.936
	Based on Median and with adjusted df	0.066	2	38.035	0.936
	Based on trimmed mean	0.049	2	45	0.952

Table 14
ANOVA Test.

RPD	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	2.333	2	1.167	19.651	0.000
Within Groups	2.671	45	0.059		
Total	5.005	47			

Table 15
Post Hoc for RPD.

			Subset for alpha = 0.05		
	Heuristic Name	N	1	2	3
Tukey HSD[a]	GVNS	16	0.7656		
	GWO	16		1.0419	
	I5	16			1.3056
	Sig.		1.000	1.000	1.000

Means for groups in homogeneous subsets are displayed.
[a] Uses Harmonic Mean Sample Size = 16.000.

6.4. Statistical analysis of metaheuristic algorithms and I5

Similar to heuristics comparisons in Section 6.2, we start with Shapiro-Wilk test to check the normality of obtained results in Fig. 9.

According to Table 12, there is not enough evidence to reject the normality assumption. The next step is to test the homogeneity of variances or levene's test in Table 13. According to this table, the test statistics is not significant and as a result, the group variances are homogeneous. The RPD values for these heuristics methods are generated independently from each other. Thus, the independency assumption holds.

So far, we have confirmed the assumptions for ANOVA test. ANOVA test in Table 14 shows the P-value < 0.001. Therefore, there is at least one group whose mean RPD is different than the rest.

Table 15 shows exactly which of the three heuristics are performing significantly different. Significance values have appeared on different columns of this table. Therefore, RPD value for each heuristic is significantly different than the rest of the two. This result is in accordance with the conclusion we had in Section 6.3 where the mean RPD shoed that the performance for GVNS is better than GWO and GWO performance is significantly better than I5.

7. Conclusion and future work

Multi-stage assembly flowshops (MSAF) have diverse applications in industry. This article is the first attempt in exploring assembly flowshop with more than three stages and with setup and release time. We developed a mixed binary linear optimization model and a constraint-free mathematical model for this problem. Due to the NP-hard complexity of the optimization model, we explored special cases that are likely to occur in industry and derived polynomial solutions for the constraint-free mathematical presentation. For instance, Corollary 5 proved that, under bottleneck condition for one of the stages and no-idle condition for the manufacturing stage (or negligible release time impact on makespan), there exists polynomial optimal solutions for total completion time. This is fairly likely in industry as the likelihood of having one bottleneck stage among several stages is high and facilities tend to start the production process after procuring all the necessary components. The presented models and their polynomial optimal solutions can be extended to MSAF with no release or setup time. We also proposed lower bounds (LB), nine efficient dispatching rules, and two meta-heuristic algorithms, called GVNS and GWO, for the presented MSAF. Experimental results revealed that the proposed LB is effective with average deviation of 0.44% from the best known solution. Experimental results also showed that the proposed dispatching heuristics result in fairly good solutions. Moreover, GVNS yields significantly better performance than GWO in 15 out of 16 problem sets. The performance of proposed heuristics and meta-heuristics are confirmed with detailed statistical analysis. Authors suggest exploring AFb solutions for different criteria such total tardiness and proposing new efficient LBs, heuristics, and meta-heuristics for AFb$|r_j,s_j|C_{max}$ as one path for future study. One can also consider stochastic or controllable processing, setup, and/or release times for this problem. Exploring MSAF with learning effect is another interesting concept to explore. Developing variations of efficient MILP or constraint-free optimization models for MSAF and comparing them with the results in this paper is another direction for future studies.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.orp.2019.100111.

Appendix

I-Proof of Theorem 1. Since Eq. (15) is written for AF with at least four stages, we first derive $C_{[j]}$ for AF4$|r_j,s_j|$. In the next step, we generalize the results for more than 4 stages. Eqs. (13)–(15) for AF4$|r_j,s_j|$ is written as:

$$C_{[j]} = \max\{\max\{\max\{\max_{\forall k}\{\max\{r_{[k,j]}, A_{[k,j\text{-}1]}\} + t_{[k,j]} + s_{[k,j]}\}, C_{[2,j\text{-}1]} + sa_{[2,j]}\} + q_{[2,\,j]}, C_{[3,j\text{-}1]} + sa_{[3,j]}\} + q_{[3,\,j]}, C_{[j\text{-}1]} + sa_{[4,j]}\} + q_{[4,\,j]} \quad \text{(I)}$$

$$A_{[k,j-1]} = \max\{r_{[k,j-1]}, A_{[k,j-2]}\} + t_{[k,j-1]} + s_{[k,j-1]} \;\forall\; k = 1, \ldots, m \;\&j = 2, \ldots, n \;\text{ where } A_{[k,0]} = 0 \quad \text{(II)}$$

$$C_{[l,j-1]} = \max\{C_{[l-1,j-1]}, C_{[l,j-2]} + sa_{[l,j-1]}\} + q_{[l,j-1]} \;\forall\; l = 2,\; 3 \,\&j = 2, \ldots, n \text{ where } C_{[l,0]} = 0 \quad \text{(III)}$$

For the second stage, $l = 2$, Eq. (III) is written as, $\max\{\max_{\forall k}\{A_{[k,j\text{-}1]}\}, C_{[2,j\text{-}2]} + sa_{[2,j\text{-}1]}\} + q_{[2,\,j\text{-}1]}$. Eq. (II) leads us to $A_{[k,2]} = \max\{r_{[k,2]}, r_{[k,1]} + t_{[k,1]} + s_{[k,1]}\} + t_{[k,2]} + s_{[k,2]}$. According to the first condition, for any pairs of jobs i and j; $r_{k,i} \le r_{k,j} + t_{k,j}$. For instance, for subsequent job positions, j and j-1, we have: $r_{[k,j]} \le r_{[k,j\text{-}1]} + t_{[k,j\text{-}1]}\,\forall k$ and j (IV). $s_{[k,j\text{-}1]} \ge 0\;\forall k$ and j, simplifies $A_{[k,2]}$ to: $A_{[k,2]} = r_{[k,1]} + t_{[k,1]} + t_{[k,2]} + s_{[k,1]} + s_{[k,2]}$. Thus, Eq. (II) is written as:

$$A_{[k,j-1]} = r_{[k,1]} + \sum_{i=1}^{j-1} t_{[k,i]} + \sum_{i=1}^{j-1} s_{[k,i]} \text{ or } A_{[k,j-1]} = A_{[k,j-2]} + t_{[k,j-1]} + s_{[k,j-1]} \quad \text{(V)}$$

Eq. (III) for $l = 2$ and $j = 2$ is expressed as: $C_{[2,1]} = \max\{\max_{\forall k}\{A_{[k,1]}\}, sa_{[2,1]}\} + q_{[2,1]}$ Using $sa_{max,\,j} \le s_{min,j}\;\forall j$ condition and Eq. (V) we have:

$C_{[2,1]} = max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} and C_{[2,2]} = \max\{max_{\forall k}\{A_{[k,2]}\}, C_{[2,1]} + sa_{[2,2]}\} + q_{[2,2]} = \max\{max_{\forall k}\{A_{[k,1]} + t_{[k,2]} + s_{[k,2]}\}, max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + sa_{[2,2]}\} + q_{[2,2]}$
According to the second and the third conditions in the Theorem, $C_{[2,2]}$ is simplified to:

$C_{[2,2]} = \max_{\forall k}\{A_{[k,2]}\} + q_{[2,2]}$ Thus $C_{[2,j-1]}$ is: $C_{[2,j-1]} = \max_{\forall k}\{A_{[k,j-1]}\} + q_{[2,j-1]}$ (VI)

Eq. (III) for the third stage, $l = 3$, is written as:

$C_{[3,1]} = \max\{C_{[2,1]}, sa_{[3,1]}\} + q_{[3,1]} = \max\{\max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]}, sa_{[3,1]}\} + q_{[3,1]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]}$

$C_{[3,2]} = \max\{C_{[2,\ 2]}, C_{[3,1]} + sa_{[3,2]}\} + q_{[3,2]} = \max\{\max_{\forall k}\{A_{[k,1]} + t_{[k,1]} + s_{[k,1]}\} + q_{[2,2]}, \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]} + sa_{[3,2]}\} + q_{[3,2]}$. According to the second and the third conditions, $C_{[3,2]}$ will be:

$C_{[3,2]} = \max_{\forall k}\{A_{[k,2]}\} + q_{[2,2]} + q_{[3,2]}$ and in general: $C_{[3,j-1]} = \max_{\forall k}\{A_{[k,j-1]}\} + \sum_{l=2}^{b-1} q_{[l,j-1]}$ (VII)

The first condition, $r_{k,\max} - r_{k,\min} \le t_{k,\min}$, can be expanded as:

$r_{[k,2]} \le r_{[k,1]} + t_{[k,1]}$
$\vdots$
$r_{[k,j]} \le r_{[k,j-1]} + t_{[k,j-1]}$

Adding the sides yields:

$$\sum_{i=2}^{j} r_{[k,i]} \le \sum_{i=1}^{j-1} r_{[k,i]} + \sum_{i=1}^{j-1} t_{[k,i]} \text{ or } r_{[k,j]} - r_{[k,1]} \le \sum_{i=1}^{j-1} t_{[k,i]} \text{ or } r_{[k,j]} \le A_{[k,j-1]} \quad \text{(VIII)}$$

Replacing Eqs. (V),(VI),(VII) in Eq. (1) and applying conditions (VIII), $q_{l,\max} \le \{t_{\min}, q_{l-1,\min}\}$, and $sa_{l,\ j} \le s_{\ \min}$, simplifies $C_{\ [j]}$ to:

$$C_{[j]} = \max\left\{\max_{\forall k}\left\{r_{[k,1]} + \sum_{i=1}^{j} t_{[k,i]} + \sum_{i=1}^{j} s_{[k,i]}\right\} + q_{[2,j]} + q_{[3,j]}, C_{[j-1]} + sa_{[4,j]}\right\} + q_{[4,j]}$$

For different j, $C_{\ [j]}$ is simplified to:

$$C_{[1]} = \max_{\forall k}\{r_{[k,1]} + t_{[k,1]} + s_{[k,1]}\} + \sum_{l=2}^{4} q_{[l,1]}$$

$$C_{[2]} = \max_{\forall k}\left\{r_{[k,1]} + \sum_{i=1}^{2} t_{[k,i]} + \sum_{i=1}^{2} s_{[k,i]}\right\} + \sum_{l=2}^{4} q_{[l,2]}$$

and in general, for $b > 4$: $C_{\ [j]} = \max_{\forall k}\{r_{[k,1]} + \sum_{i=1}^{j} t_{[k,i]} + \sum_{i=1}^{j} s_{[k,i]}\} + \sum_{l=2}^{b} q_{[l,j]}$. Thus C_{max} and TC in Eqs. (16) and (17) are derived accordingly. The proof is completed. **Q.E.D.**

II-Proof of Corollary 1. Suppose there is an optimal schedule, Л*, for which the order of jobs $J_{[i]}$ and $J_{[j]}$ on the first stage machine, M_i, is $J_{[i]} \rightarrow J_{[j]}$ (i.e. $J_{[i]}$ is processed immediately before $J_{[j]}$) whereas the order of these jobs on at least one of subsequent stages, let's say M_a on stage 2, is $J_{[j]} \rightarrow J_{[i]}$. Condition $r_{k,\max} - r_{k,\min} \le t_{\ k,\min}$ yields $r_{k,i} \le A_{k,i-1}\ \forall k$ and I (no idle time between consecutive jobs in the first stage). For Л* we have: $A_{[k,j]}$ (Л*) $= A_{[k,i]}$ (Л*) $+ t_{\ [k,i]} + s_{\ [k,i]} \le R_{\ [2,j]}$(Л*) $\le R_{\ [2,i]}$ (Л*).

Consider schedule Л' which is made by swapping positions of $J_{[j]}$ and $J_{[i]}$ on M_i. We have:

$A_{[k,j]}$ (Л') $\le A_{[k,j]}$ (Л*) $\le R_{[2,j]}$ (Л*)

Also, $A_{[k,i]}$ (Л') $= A_{[k,j]}$ (Л*) $\le R_{\ [2,i]}$ (Л*). Therefore, all jobs in Л' can start on M_a at the same time as they start in Л*. Thus, Л' is optimal too. Similar proof can be provided for TC objective. This corollary can be proved for any other order of jobs and any combination of processing stages considering conditions of Theorem 1. Thus, permutation schedule does not increase the optimal makespan. This result can be easily extended for AFb$|r_j|$TC. Corollary 2 derives a polynomial optimal algorithm for AFb$|r_j|C_{max}$ under the conditions of Theorem 1.

III-Proof of Theorem 2. Similar to Theorem 1, we first derive $C_{[j]}$ for AF4$|r_j,s_j|$. Then, we generalize the results for $l > 4$ stages. The first condition simplifies Eq. (II) to: $A_{[k,j\text{-}1]} = A_{[k,j\text{-}2]} + t_{[k,j\text{-}1]} + s_{[k,j\text{-}1]}$ which is the same as Eq. (V). According to the third condition, Eq. (III) for $l = 2$ is expressed as:

$C_{[2,2]} = \max\{\max_{\forall k}\{A_{[k,1]} + t_{[k,2]} + s_{[k,2]}\}, sa_{[2,1]} + q_{[2,1]} + sa_{[2,2]}\} + q_{[2,2]}$

The third condition, $A_{max,j} < sa_{\min,\ j}$, implies $s_{\max,j} < sa_{\min,\ j}$ and $t_{\max,j} < sa_{\min,\ j}$ Thus, using this condition and the second condition, we have: $C_{[2,2]} = \sum_{i=1}^{2} sa_{[2,i]} + \sum_{i=1}^{2} q_{[2,i]}$

$$C_{[2,3]} = \max\{\max_{\forall k}\{A_{[k,3]}\} C_{[2,2]} + sa_{[2,3]}\} + q_{[2,3]} = \max\{\max_{\forall k}\{A_{[k,1]} + t_{[k,2]} + s_{[k,2]} + t_{[k,3]} + s_{[k,3]}\}, sa_{[2,1]} + q_{[2,1]} + sa_{[2,2]} + q_{[2,2]} + sa_{[2,3]}\} + q_{[2,3]}$$
$$= \sum_{i=1}^{3} sa_{[2,i]} + \sum_{i=1}^{3} q_{[2,i]}$$

Thus, $C_{[2,j\text{-}1]}$ in Eq. (III) is:

$$C_{[2,j-1]} = \sum_{i=1}^{j-1} sa_{[2,i]} + \sum_{i=1}^{j-1} q_{[2,i]} \quad \text{(VIII)}$$

Eq. (15) for the third stage, $l = 3$, is as follows:

$$C_{[3,1]} = \max\{C_{[2,1]}, sa_{[3,1]}\} + q_{[3,1]} = \max\{sa_{[2,1]} + q_{[2,1]}, sa_{[3,1]}\} + q_{[3,1]} = sa_{[2,1]} + q_{[2,1]} + q_{[3,1]}$$

$$C_{[3,2]} = \max\{C_{[2,2]}, C_{[3,1]} + sa_{[3,2]}\} + q_{[3,2]} = \max\left\{\sum_{i=1}^{2} sa_{[2,i]} + \sum_{i=1}^{2} q_{[2,i]}, sa_{[2,1]} + q_{[2,1]} + q_{[3,1]} + sa_{[3,2]}\right\} + q_{[3,2]}$$

$$= \sum_{i=1}^{2} sa_{[2,i]} + \sum_{i=1}^{2} q_{[2,i]} + q_{[3,2]}$$

$$C_{[3,3]} = \max\{C_{[2,3]}, C_{[3,2]} + sa_{[3,3]}\} + q_{[3,3]} = \max\left\{\sum_{i=1}^{3} sa_{[2,i]} + \sum_{i=1}^{3} q_{[2,i]}, \sum_{i=1}^{2} sa_{[2,i]} + \sum_{i=1}^{2} q_{[2,i]} + q_{[3,2]} + sa_{[3,3]}\right\} + q_{[3,3]} = \sum_{i=1}^{3} sa_{[2,i]} + \sum_{i=1}^{3} q_{[2,i]} + q_{[3,3]}$$

and in general:

$$C_{[3,j-1]} = \sum_{i=1}^{j-1} sa_{[2,i]} + \sum_{i=1}^{j-1} q_{[2,i]} + q_{[3,j-1]} \quad \text{(IX)}$$

The third condition, $A_{max,j} < sa_{\min,\,j}$, implies $r_{k,1} < sa_{\min,\,j}$. Using this condition and replacing Eqs. (V), (VIII), and (IX) in Eq. (13) and applying conditions of Theorem 2 simplifies $C_{[j]}$ to:

$$C_{[j]} = \max\left\{\sum_{i=1}^{j} sa_{[2,i]} + \sum_{i=1}^{j} q_{[2,i]} + q_{[3,j]}, C_{[j-1]} + sa_{[4,j]}\right\} + q_{[4,j]}$$

$$C_{[2]} = \max\{\textstyle\sum_{i=1}^{2} sa_{[2,i]} + \sum_{i=1}^{2} q_{[2,i]} + q_{[3,2]}, sa_{[2,1]} + \sum_{l=2}^{b} q_{[l,1]} + sa_{[4,2]}\} + q_{[4,2]} = \sum_{i=1}^{2} sa_{[2,i]} +$$
$$\textstyle\sum_{i=1}^{2} q_{2,i]} + \sum_{l=b-1}^{b} q_{[l,2]}$$

$$C_{[3]} = \max\{\textstyle\sum_{i=1}^{3} sa_{[2,i]} + \sum_{i=1}^{3} q_{[2,i]} + q_{[3,3]}, \sum_{i=1}^{2} sa_{[2,i]} + \sum_{i=1}^{2} q_{[2,i]} + \sum_{l=b-1}^{b} q_{[l,2]} + sa_{[4,3]}\} + q_{[4,3]}$$
$$= \textstyle\sum_{i=1}^{3} sa_{[2,i]} + \sum_{i=1}^{3} q_{[2,i]} + \sum_{l=b-1}^{b} q_{[l,3]}$$

$$C_{[j]} = \sum_{i=1}^{j} sa_{[2,i]} + \sum_{i=1}^{j} q_{[2,i]} + \sum_{l=3}^{4} q_{[l,j]}$$

and, for $b > 4$: $C_{[j]} = \sum_{i=1}^{j} sa_{[2,i]} + \sum_{i=1}^{j} q_{[2,i]} + \sum_{l=b-1}^{b} q_{[l,j]}$. Thus C_{max} and TC in Eqs. (16)–(19) are derived accordingly. The proof is completed. **Q.E.D.**

IV- Proof of Theorem 3. Similar to Theorem 1, we first derive $C_{[j]}$ for AF4$|r_j,s_j|$. Then, we generalize the results for $l > 4$. The first condition simplifies Eq. (II) to: $A_{[k,j-1]} = A_{[k,j-2]} + t_{[k,j-1]} + s_{[k,j-1]}$ (V). Similarly, Eq. (III) for $l = 2$ is expressed as:

$C_{[2,1]} = \max\{\max_{\forall k}\{A_{[k,1]}\}, sa_{[2,1]}\} + q_{[2,1]} = \max_{\forall k}\{A_{[k,\,1]}\} + q_{[2,1]}$ (According to the third condition)

$$C_{[2,2]} = \max\{\max_{\forall k}\{A_{[k,1]} + t_{[k,2]} + s_{[k,2]}\}, \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + sa_{[2,2]}\} + q_{[2,2]}$$

According to the second and the third conditions, $C_{[2,2]}$ will be: $C_{[2,2]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + sa_{[2,2]} + q_{[2,2]}$

$$C_{[2,3]} = \max\{\max_{\forall k}\{A_{[k,1]} + t_{[k,2]} + s_{[k,2]} + t_{[k,3]} + s_{[k,3]}\}, \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + sa_{[2,2]} + q_{[2,2]} + sa_{[2,3]}\} + q_{[2,3]} =$$
$$\max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + sa_{[2,2]} + q_{[2,2]} + sa_{[2,3]} + q_{[2,3]}$$

Thus, $C_{[2,j-1]}$ in Eq. (III) is:

$$C_{[2,j-1]} = \max_{\forall k}\{A_{[k,1]}\} + \sum_{i=2}^{j-1} sa_{[2,i]} + \sum_{i=1}^{j-1} q_{[2,i]} \quad \text{(X)}$$

Eq. (15) for the third stage, $l = 3$, is written as follows:

$$C_{[3,1]} = \max\{\max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]}, sa_{[3,1]}\} + q_{[3,1]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]}$$

$$C_{[3,2]} = \max\{\max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]}, sa_{[2,2} + q_{[2,2]}, \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]} + sa_{[3,2]}\} + q_{[3,2]}$$

$$C_{[3,2]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]} + sa_{[3,2]} + q_{[3,2]}$$

and in general:

$$C_{[3,j-1]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + \sum_{i=1}^{j-1} q_{[3,i]} + \sum_{i=2}^{j-1} sa_{[3,i]} \quad \text{(XI)}$$

Replacing Eqs. (V) and (XI) and considering $r_{[k,\,j]} \le A_{[k,j-1]}$ in Equation (13) and applying conditions of Theorem 3, simplifies $C_{[j]}$ to:

$$C_{[j]} = \max\left\{\max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + \sum_{i=1}^{j} q_{[3,i]} + \sum_{i=2}^{j} sa_{[3,i]},\ C_{[j-1]} + sa_{[4,j]}\right\} + q_{[4,j]}$$

$$C_{[1]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]} + q_{[4,1]}$$

$$C_{[2]} = \max_{\forall k}\{A_{[k,1]}\} + q_{[2,1]} + q_{[3,1]} + q_{[4,1]} + sa_{[4,2]} + q_{[4,2]}$$

$$C_{[j]} = \max_{\forall k}\{A_{[k,1]}\} + \sum_{l=2}^{3} q_{[l,1]} + \sum_{i=1}^{j} q_{[b,i]} + sa_{[4,j]}$$

and in general, for $b > 4$: $C_{[j]} = \max_{\forall k}\{A_{[k,1]}\} + \sum_{l=2}^{b-1} q_{[l,1]} + \sum_{i=1}^{j} q_{[b,i]} + sa_{[b,j]}$. Thus C_{max} and TC in Eqs. (22) and (23) are derived accordingly. The proof is completed. **Q.E.D.**

References

[1] Allahverdi A, Gupta JND, Aldowaisan T. A review of scheduling research involving setup considerations. *OMEGA Int J Manag Sci* 1999;27:219–39.
[2] Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY. A survey of scheduling problems with setup times or costs. *Eur J Oper Res* 2008;187:985–1032.
[3] Allahverdi A, Soroush HM. The significance of reducing setup times/setup costs. *Eur J Oper Res* 2008;187:978–84.
[4] Allahverdi A. The third comprehensive survey on scheduling problems with setup times/costs. *Eur J Oper Res* 2015;246(2):345–78.
[5] Allahverdi A, Aydilek H, Aydilek A. Two-stage assembly scheduling problem for minimizing total tardiness with setup times. *Appl Math Modell* 2016;40(17-18):7796–815.
[6] Andrés C, Hatami S. The three stage assembly permutation flowshop scheduling problem. V international conference on industrial engineering and industrial management. 2011. p. 867–75.
[7] Campos SC, Arroyo JEC, Tavares RG. A general VNS heuristic for a three-stage assembly flow shop scheduling problem. International Conference on Intelligent Systems Design and Applications. Cham: Springer; 2016. p. 955–64.
[8] Framinan JM, Perez-Gonzalez P. The 2-stage assembly flowshop scheduling problem with total completion time: efficient constructive heuristic and metaheuristic. *Comput Oper Res* 2017;88:237–46.
[9] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. *Math Oper Res* 1976;1(2):117–29.
[10] Gonzalez-Neira EM, Ferone D, Hatami S, Juan AA. A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. *Simul Modell Pract Theory* 2017;79:23–36.
[11] Graham RL, Lawler EL, Lenstra JK, Kan AR. Optimization and approximation in deterministic sequencing and scheduling: a survey. In annals of discrete mathematics (Vol 5. Elsevier; 1979. p. 287–326.
[12] Hatami S, Ebrahimnejad S, Tavakkoli-Moghaddam R, Maboudian Y. Two meta-heuristics for three-stage assembly flow shop scheduling with sequence-dependent setup times. *Int J Adv Manuf Technol* 2010;50:1153–64.
[13] Hatami S, Ruiz R, Andres-Romano C. The distributed assembly permutation flowshop scheduling problem. *Int J Prod Res* 2013;51(17):5292–308.
[14] Hatami S, Ruiz R, Andrés-Romano C. Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *Int J Prod Econ* 2015;169:76–88.
[15] Komaki GM, Mobin S, Teymourian E, Sheikh S. A general variable neighborhood search algorithm to minimize makespan of the distributed permutation flowshop scheduling problem. World Acad Sci Eng and Technol Int J Soc Behav Educ Econ Bus Ind Eng 2015;9(8):2701–8.
[16] Komaki GM, Kayvanfar V. Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. *J Comput Sci* 2015;8:109–20.
[17] Komaki GM, Teymourian E, Kayvanfar V, Booyavi Z. Improved discrete Cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. *Comput Ind Eng* 2017;105:158–73.
[18] Komaki M, Malakooti B. General variable neighborhood search algorithm to minimize makespan of the distributed no-wait flow shop scheduling problem. *Prod Eng* 2017;11(3):315–29.
[19] Komaki GM, Sheikh S, Malakooti B. Flow shop scheduling problems with assembly operations: a review and new trends. *Int J Prod Res* 2018:1–30https://www.tandfonline.com/doi/abs/10.1080/00207543.2018.1550269.
[20] Koulamas C, Kyparisis GJ. The three-stage assembly flow shop scheduling problem. *Comput Oper Res* 2001;28:689–704.
[21] Lee CY, Cheng TCE, Lin BMT. Minimizing the makespan in the 3-machine assembly-type flow shop scheduling problem. Management Science 1993;39:616–25.
[22] Lin J, Wang ZJ, Li X. A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm Evolut Comput* 2017;36:124–35.
[23] Maleki-daronkolaei Aref, Seyedi I. Taguchi method for three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *J Eng Sci Technol* 2013;8(5):603–22.
[24] Maleki-Darounkolaei A, Modiri M, Tavakkoli-Moghaddam R, Seyyedi I. A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *J Ind Eng Int* 2012;8(1):1–7.
[25] Mirjalili S, Mirjalili SM, S.M., A., Lewis A. Grey wolf optimizer. *Adv Eng Softw* 2014;69:46–61. 2014.
[26] Morizawa K. A branch-and-bound based heuristic algorithm for minimizing makespan in machining-assembly flowshop scheduling. Engineering 2014;6(13):877.
[27] Naderi B, Ruiz R. The distributed permutation flowshop scheduling problem. *Comput Oper Res* 2010;37(4):754–68.
[28] Potts CN, Sevast'janov SV, Strusevich VA, Van Wassenhove LN, Zwaneveld CM. The two-stage assembly scheduling problem: complexity and approximation. Oper Res 1995;43(2):346–55.
[29] Sheikh S, Komaki GM, Kayvanfar V. Multi objective two-stage assembly flow shop with release time. *Comput Ind Eng* 2018;124:276–92.
[30] Shoaardebili N, Fattahi P. Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints. *Int J Prod Res* 2015;53(3):944–68.
[31] Tajbakhsh Z, Fattahi P, Behnamian J. Multi-objective assembly permutation flow shop scheduling problem: a mathematical model and a meta-heuristic algorithm. *J Oper Res Soc* 2014;65(10):1580–92.
[32] Tharumarajah A, Bemelman R, Welgama P, Wells A. Distributed scheduling of an assembly shop. Systems, Man, and Cybernetics. 1998 IEEE International Conference on 1. IEEE; 1998. p. 433–8.
[33] Xiong F, Xing K. Meta-heuristics for the distributed two-stage assembly scheduling problem with bi-criteria of makespan and mean completion time. *Int J Prod Res* 2014;52(9):2743–66.
[34] Yokoyama M, Santos DL. Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times. *Eur J Oper Res* 2005;161(3):754–70.
[35] Zhang Y, Zhou Z, Liu J. The production scheduling problem in a multi-page invoice printing system. *Comput Oper Res* 2010;37(10):1814–21.
[36] Zhang G, Xing K, Cao F. Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan. *Int J Prod Res* 2017:1–19.
[37] Deng J, Wang L, Wang SY, Zheng XL. A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem. *Int J Prod Res* 2016;54(12):3561–77.
[38] Li P, Yang Y, Du X, Qu X, Wang K, Liu B. Iterated Local Search for Distributed Multiple Assembly No-Wait Flowshop Scheduling. 2017 IEEE Congress on evolutionary computation (CEC) 2017:1565–71.
[39] Tozkapan A, Kirca Ö, Chung CS. A Branch and Bound Algorithm to Minimize the Total Weighted Flowtime for the Two-Stage Assembly Scheduling Problem. Computers and Operations Research 2003;30(2):309–20.