Avrupa Bilim ve Teknoloji Dergisi
Sayı 23, S. 126-148, Nisan 2021
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 23, pp. 126-148, April 2021
Copyright © 2021 EJOSAT
Research Article

A Hybrid Benders Decomposition Algorithm and New Models for the Distributed Permutation Flowshop Scheduling Problem

Hanifi Okan Isguder[1*], Alper Hamzadayı[2]

[1*] Dokuz Eylul University, Faculty of Science, Departmant of Statistics, Izmir, Turkey, (ORCID: 0000-0002-5188-856X), okan.isguder@deu.edu.tr
[2] Van Yuzuncu Yil University, Engineering Faculty, Departmant of Industrial Engineering, Van, Turkey, (ORCID: 0000-0003-4035-2775), alperhamzadayi@yyu.edu.tr

(First received 21 October 2020 and in final form 24 February 2021)

(**DOI:** 10.31590/ejosat.814129)

ATIF/REFERENCE: Isguder, H. O., & Hamzadayi, A. (2021). A Hybrid Benders Decomposition Algorithm and Models for the Distributed Permutation Flowshop Scheduling Problem. *European Journal of Science and Technology*, (23), 126-148.

Abstract

The distributed permutation flowshop scheduling problem (DPFSP) is a generalization of the regular flowshop scheduling problem where several factories are accessible for processing the jobs. In this paper, two new mathematical models are developed by deriving inspiration from the formulations developed for the multiple-traveling salesman problem (mTSP), and six different pure Benders decomposition algorithms are developed based on different mathematical model formulations. In addition, a hybrid Benders decomposition algorithm is developed through the best performed mathematical. Nine newly developed exact methods are compared in detail with each other, the best mathematical models given by Naderi and Ruiz (2010) and an automatic Benders decomposition algorithm by using the 84 problem instances available in the literature. The consequences of the experiment performed for the comparison of all existing and new exact algorithms have revealed that the proposed hybrid Benders decomposition algorithm has outperformed considerably when compared to the other methods. In this paper, 4 new best solutions are identified for the DPFSP.

Keywords: Distributed flowshop problem, Mixed integer linear programming, Benders decomposition algorithm, LS3 local search procedure.

Dağıtılmış Permütasyon Akış Tipi Atölye Çizelgeleme Problemi için Hibrit Benders Ayrıştırma Algoritması ve Yeni Modeller

Öz

Dağıtılmış permütasyon akış tipi çizelgeleme problemi (DPATÇP), işleri işlemek için birkaç fabrikanın mevcut olduğu akış tipi çizelgeleme probleminin bir genellemesidir. Bu çalışmada, çoklu gezgin satıcı problemi (ÇGSP) için geliştirilen modellerden esinlenilerek iki yeni matematiksel model ve farklı matematiksel modellere dayalı olarak altı farklı saf Benders ayrıştırma algoritmaları geliştirilmiştir. Ayrıca, en iyi performansı sağlayan matematiksel model aracılığıyla hibrit bir Benders ayrıştırma algoritması geliştirilmiştir. Yeni geliştirilen dokuz kesin çözüm yöntemi, Naderi ve Ruiz (2010) tarafından önerilen en iyi matematiksel modeller ve otomatik Benders ayrıştırma algoritması ile literatürde mevcut olan 84 problem seti kullanılarak karşılaştırılmıştır. Tüm mevcut ve yeni kesin çözüm algoritmaların karşılaştırılması için gerçekleştirilen deneyin sonuçları, önerilen hibrit Benders ayrıştırma algoritmasının diğer yöntemlere kıyasla önemli ölçüde daha iyi performans gösterdiğini ortaya koymuştur. Bu makalede, DPATÇP için 4 yeni en iyi çözüm saptanmıştır.

Anahtar Kelimeler: Dağıtılmış akış tipi problem, Karışık tamsayı doğrusal programlama, Benders ayrıştırma algoritması, LS3 yerel arama prosedürü.

[*] Corresponding Author: okan.isguder@deu.edu.tr

1. Introduction

Machine scheduling problems have been extensively studied in the literature for more than 60 years beginning from Johnson's first study being conducted (Johnson (1954)). Framinan, Leisten, and Ruiz (2014), McKay, Pinedo and Webster (2002) and Pinedo (2016) emphasized and discussed the importance of the optimized scheduling in detail and in depth. The flowshop scheduling problem is one of the most studied versions of the scheduling problem in the literature. In a flowshop problem, the machines on the production floor are arranged in series and the jobs go through all machines in the order determined similar to the mentioned way. Each job has a known amount of processing time at each machine, meaning a job cannot be processed on the next machine unless processed completely on the previous one. The machines cannot process more than one job at the same time and no preemption is allowed, for example, it is not possible to interrupt the jobs once started at any machine. The most commonly studied objective in the flowshop literature is the minimization of the maximum completion time called makespan. Detailed literature reviews on flowshop scheduling problem can be found in Fernandez-Viagas, Ruiz, and Framinan (2017), Framinan, Gupta, and Leisten (2004), Gupta and Stafford (2006), Hejazi and Saghafian (2005), Reisman, Kumar, and Motwani (1997), and Ruiz and Maroto (2005). The flowshop problem of a makespan criterion is NP-Complete in the strong sense (Garey, Johnson, and Sethi (1976)).

The multi factory environment has a critical significance in today's centralized globalized economy (Chan et al. (2006); Deng and Wang (2017); Jia et al. (2007)). Consequently, the multi factory production scheduling environment, the so-called distributed scheduling problem, has drawn increasingly more attention in recent years (Giovanni and Pezzella (2010); Gupta and Stafford (2006); Ying et al. (2017)). The distributed permutation flow shop scheduling problem (DPFSP), one of the distributed scheduling problem types, is a generalization of the conventional permutation flow shop scheduling problem (PFSP). In the DPFSP, a set of jobs must be processed at a number of identical factories, and each factory is equipped with a series of identical machines arranged as a flowshop. Which job will be produced and the order of the jobs to be produced at each factory must be decided simultaneously according to a given performance measure.

The primary study for the DPFSP was carried out by Naderi and Ruiz (2010) and no other study is available as a precise solution method in the current literature. In the mentioned study, Naderi and Ruiz (2010) presented six different linear programming models. The best 2 performances were shown by the minimal sequence-based distributed permutation flowshop scheduling model and position-based distributed permutation flowshop scheduling model, respectively, among these 6 different linear programming models. In this study, Naderi and Ruiz (2010) also proposed two factory assignment rules together with 14 heuristics based on dispatching rules, NEH method (Nawaz, Enscore, and Ham (1983)), and a variable neighborhood decent method (VND). As revealed by computational and statistical analysis, the NEH based heuristics with two factory assignment rules (denoted by NEH1 and NEH2, respectively) and the VND with two acceptance criteria (referred to as VNDa and VNDb, respectively) were the top four effective heuristics. Numerous heuristics and metaheuristics have been also proposed for solving the DPFSP by minimizing the makespan criterion since the first study conducted by Naderi and Ruiz (2010). Liu and Gao (2010) presented an electromagnetism metaheuristic (EM) by combining numerous local search neighborhoods. They were able to improve 151 best-known solutions out of 720 large instances presented in Naderi and Ruiz (2010) but involving a significantly larger CPU time. An improved version of NEH2 of Naderi and Ruiz (2010) heuristic by using a novel insertion rule was presented by Gao and Chen (2011a). In the same year, a hybrid genetic algorithm (HGA) with an enhanced local search method was put forward by Gao and Chen (2011b). A revised VND by hybridizing the VND method and improved NEH heuristic was proposed by Gao et al. (2012). A tabu search (TS) algorithm based on exchanging sub-sequences between factories to generate neighboring solutions was presented by Gao, Chen, and Deng (2013). In the same year, a modified iterated greedy (MIG) method was put forward by Lin, Ying, and Huang (2013). They obtained much better results than the HGA and TS with greatly reduced CPU time by applying MIG. Once more, in the same year, an estimation of distribution algorithm (EDA) that uses explicit probability distributions in the search process was proposed by Wang et al. (2013). A scatter search algorithm (called SSNR) was presented by Naderi and Ruiz (2014). They deduced that the SSNR was a clear winner against the HGA, MIG, EM, TS, VNDa, VNDb, and VND (B&B). A hybrid immune algorithm (HIA) was proposed by Xu et al. (2014). The authors claimed that the HIA improved 585 out of 720 instances of Naderi and Ruiz (2010). A bound-search iterated greedy (BSIG) that incorporates several different local search procedures was proposed by Fernandez-Viagas and Framinan (2015). It was compared to the EDA, IG, and best methods presented in Naderi and Ruiz (2010). The results provided show a clear superiority of the BSIG over three other methods tested. The authors also improved 263 of the original 720 best-known solutions. More recently, a two-stage iterated greedy algorithm (IG2S) was presented by Ruiz, Pan, and Naderi (2019). The authors obtained 497 new upper bounds and average Relative Percentage Deviations were reduced by 60% when compared to BSIG and 81% when compared to SSNR.

As mentioned above, there is no study as an exact solution method for the DPFSP problem, except for the six linear programming model proposed by Naderi and Ruiz (2010). The best performance was given by the position-based distributed permutation flowshop scheduling model and minimal sequence-based distributed permutation flowshop scheduling model among these six models presented by Naderi and Ruiz (2010). In addition to these two best models of Naderi and Ruiz (2010), in this paper; two new models are developed based on the multiple-traveling salesman problem (mTSP) formulations available in Bektas (2006). After detecting the most effective existing and new mathematical formulations, these are the permutation flowshop scheduling model and the model developed through the multiple-traveling mTSP-assignment based integer programming formulation, different benders algorithms are developed using these models. Four different Benders decomposition algorithms are developed and tested based on the permutation flowshop scheduling model. The proposed pure benders algorithm through the permutation flowshop scheduling model follows the 4 different strategies that are given below.

1. Upon having solved the master model, the entire solution obtained from the master model solution (all decision

variables taking the value of 1) is fixed in the subproblem. One large optimality cut is obtained from the subproblem solution and this optimality cut is added to the master model to be solved in the next Benders iteration.
2. After having solved the master model, the master model solution is decomposed according to the factories to which at least 1 job is assigned. Then, separate cuts are obtained for each solution and afterward, each optimality cut obtained is added to the master problem separately for all factories for the subsequent master problem solution.
3. After having solved the master model, the makespans are calculated disjointedly for the factories to which at least 1 job is assigned in the master problem solution. The subproblem is solved only for the factory yielding the longest makespan and a single optimality cut is inserted into the master problem only for the factory yielding the longest makespan.
4. After having solved the master model, the makespan is calculated separately for the factories to which at least 1 factory is assigned in the master problem solution. The subproblem is solved only for the factory yielding the longest makespan and the optimality cut obtained from the subproblem solution is inserted separately for all factories.

Also, two different pure Benders decomposition algorithms are developed and tested based on the mTSP-assignment based integer programming formulation. The mTSP-assignment based integer programming formulation is structurally only suitable for the first and third the cut adding strategies mentioned above. In addition to these newly developed eight different exact methods, a hybrid Benders decomposition algorithm is developed by hybridizing the LS3 local search algorithm with the strategy one based pure Benders algorithm that is developed over the permutation flowshop scheduling model, in order to be able to accelerate the convergence of the Benders algorithm. Twelve different exact solution methods are compared with each other by using 84 problem instances and also the automatic Benders decomposition algorithm available ready-to-use in the CPLEX software. Experimental results demonstrated that the proposed hybrid algorithm is superior to all of the existing and new methods in terms of its efficiency and effectiveness. In this paper, 4 new best solutions are also determined for the DPFSP.

The remainder of this paper is structured as follows. Two new linear programming models developed by inspiring by the mTSP and the best two linear programming models put forward by Naderi and Ruiz (2010) are given in Section 2. Six different Benders decomposition algorithms developed are given in Section 3 and also the proposed hybrid Benders decomposition algorithm in Section 4. Computational results are presented in Section 5, followed by conclusions in Section 6.

2. Linear Programming Models

In this Section, the most performant two mathematical given in Naderi and Ruiz (2010) are presented and also two new models are developed by inspiring two different mTSP formulations. Before presenting each model, we initiate by informing about the parameters and indices employed. The common parameters and indices for all six proposed models are defined in Table 1.

The objective function for all models is makespan minimization:

$$\text{Objective: Minimize } C_{\max} \tag{1}$$

2.1. Position-based Linear Programming Model

Six models were proposed in Naderi and Ruiz (2010) and the following model is reported as the model giving the 2nd best performance in Naderi and Ruiz (2010) and called the model number 3 in their paper. In this paper, this model is demonstrated as Model_1. In this model, the decision variables given below have been used:

$C_{k,i,f}\rightarrow$ Continuous variable representing the completion time of the job in position k on machine i at factory f.

$X_{j,k,f}\rightarrow$ Binary variable that takes value 1 if job j occupies position k in factory f, and 0 otherwise.

The Model_1 is a position-based distributed permutation flowshop scheduling model and its mathematical model is given in detail below.

Objective function: Minimize $C_{\max}$ (Equation 1)

Subject to:

$$\sum_{k=1}^{N}\sum_{f=1}^{F} X_{j,k,f} = 1 \quad j \in \{1,\dots,N\} \tag{2}$$

$$\sum_{j=1}^{N}\sum_{f=1}^{F} X_{j,k,f} = 1 \quad k \in \{1,\dots,N\} \tag{3}$$

$$C_{k,i,f} \geq C_{k,i-1,f} + \sum_{j=1}^{N} X_{j,k,f} \cdot P_{j,i} \quad k \in \{1,\dots,N\};\ i \in \{1,\dots,M\};\ f \in \{1,\dots,F\} \tag{4}$$

$$C_{k,i,f} \geq C_{k-1,i,f} + \sum_{j=1}^{N} X_{j,k,f} \cdot P_{j,i} \quad k \in \{2,\dots,N\};\ i \in \{1,\dots,M\};\ f \in \{1,\dots,F\} \tag{5}$$

$$C_{max} \geq C_{k,M,f} \quad k \in \{1,\dots,N\};\ f \in \{1,\dots,F\} \tag{6}$$

$$C_{k,0,f} = 0 \quad k \in \{1,\dots,N\};\ f \in \{1,\dots,F\} \tag{7}$$

$$C_{k,i,f} \geq 0 \quad k \in \{1,\dots,N\};\ i \in \{1,\dots,M\};\ f \in \{1,\dots,F\} \tag{8}$$

$$X_{j,k,f} \in \{0,1\} \quad k \in \{1,\dots,N\};\ i \in \{1,\dots,M\};\ f \in \{1,\dots,F\} \tag{9}$$

With Constraint set (2), it is enforced that every job must occupy exactly one position in the sequence. Constraint set (3) states that n positions among all nF possible must be occupied. Constraint set (4) controls that the processing of job in position k of factory f at each machine can only start when the processing of the same job on the previous machine is finished. Constraint set (5) ensures that each job can start only after the previous job assigned to the same machine at the same factory is completed. Notice that this previous job might not be exactly in the previous position but in any preceding positions. Constraint set (6) formulates the makespan. Constraint set (7) ensures that the completion time of the job in position k on machine 0 at factory f must be 0. Lastly, Constraint sets (8) and (9) define the decision variables.

Table 1. Parameters and indices used in the models.

Parameter	Description
N	Number of jobs
M	Number of machines
F	Number of factories
j, k	Index for jobs (or job positions in a sequence); $j, k \in \{1, 2,..., N\}$
i, l	Index for machines; $i, l \in \{1, 2,...,M\}$
f	Index for factories; $f \in \{1, 2,..., F\}$
$O_{j,i}$	Operation of job j at machine i
$P_{j,i}$	Processing time of $O_{j,i}$
$bigM$	A sufficiently large positive number

2.2. Minimal Sequence-based Linear Programming Model

The model given below is the best performing model among the models proposed by Naderi and Ruiz (2010) and called as the model 5 in their paper. In this paper, this model is denoted as Model_2. The said model is a minimal sequence-based distributed permutation flowshop scheduling model. Model_2 can solve the DPFSP without actually indexing the factories. Since the sequence-based variables are used, dummy jobs 0 need to be defined. Model_2 needs the following decision variables given below.

$X_{k,j} \rightarrow$ Binary variable that takes value 1 if the job j is processed immediately after the job k; and 0 otherwise.

$C_{j,i} \rightarrow$ Continuous variable for the completion time of the job j on the machine i.

Model_2 exploits the dummy job 0 in such a way that it partitions a complete sequence into F parts each of which corresponds to a factory. This is performed through F repetitions of dummy job 0 in the sequence along with the other jobs. Therefore, this model searches the space with a sequence counting $n+F$ positions. One of these repetitions takes place in the first position of the sequence. All the subsequent jobs until the second repetition of dummy job 0 are scheduled in factory 1 with their current relative order. The jobs between the second and third repetitions of dummy job 0 are those allocated to factory 2. This repeating for all the subsequent repetitions of dummy job 0, the jobs after the F-th repetition of dummy job 0 until the last job in the sequence are those assigned to factory F. For example, consider a problem with $N = 6$ and $F = 2$, one of the possible solutions is $X_{0,1} = X_{1,3} = X_{3,6} = X_{6,0} = X_{0,4} = X_{4,2} = X_{2,1} = X_{1,5} = 1$; that is, {0, 3, 6, 0, 4, 2, 1, 5}. In this example, the jobs 3 and 6 are allocated to factory 1 with this order {3, 6} whereas the other jobs are assigned to factory 2 with the permutation or sequence {4, 2, 1, 5}. Model_2 is given in detail below.

Objective function: Minimize Cmax (Equation 1)
Subject to:

$$\sum_{k=0}^{N} X_{k,j} = 1 \quad j \in \{1, \dots, N\} \mid j \neq k \tag{10}$$

$$\sum_{j=0, k \neq j}^{N} X_{k,j} \leq 1 \quad k \in \{1, \dots, N\} \mid k \neq j \tag{11}$$

$$\sum_{j=1}^{N} X_{0,j} = F \tag{12}$$

$$\sum_{k=1}^{N} X_{k,0} = F - 1 \tag{13}$$

$$X_{k,j} + X_{j,k} \leq 1 \quad k \in \{1, \dots, N-1\};\ j \in \{1, \dots, N\} \mid j \neq k, j > k \tag{14}$$

$$C_{j,i} \geq C_{j,i-1} + P_{j,i} \quad j \in \{1, \dots, N\};\ i \in \{1, \dots, M\} \tag{15}$$

$$C_{j,i} \geq C_{k,i} + P_{j,i} + bigM\left(X_{k,j} - 1\right)\ k, \in \{1, \dots, N\} \mid k \neq j;\ i \in \{1, \dots, M\} \tag{16}$$

$$C_{max} \geq C_{j,M} \quad j \in \{1, \dots, N\} \tag{17}$$

$$C_{j,i} \geq 0 \quad j \in \{1, \dots, N\};\ i \in \{1, \dots, M\} \tag{18}$$

$$X_{k,j} \in \{0, 1\} \quad k, j \in \{0, \dots, N\} \mid k \neq j \tag{19}$$

Note that $C_{k,0} = C_{0,l} = 0$. Constraint set (10) ensures that every job is required to be accurately at one position. Constraint set (11) indicates that every job has at most one subsequent job. Constraint set (12) enforces that dummy job 0 appears F times in the sequence as a predecessor where Constraint set (13) assures dummy job 0 must be a successor $F-1$ times. Constraint set (14) avoids the occurrence of cross-precedencies, meaning that a job cannot be at the same time both a predecessor and a successor of another job. Constraint set (15) forces that for every job j, $O_{j,i}$ cannot begin before $O_{j,i-1}$ completes. Similarly, Constraint set (16) specifies that if job j is scheduled immediately after the job k it's processing on each machine i cannot begin before the processing of the job k on the machine i finishes. Constraint set (17) defines the makespan. Lastly, Constraint sets (18) and (19) define the decision variables.

2.3. Linear Programming Model Based on the mTSP-assignment Based Integer Programming Formulation

This model (represented by Model_3) is developed based on the assignment-based integer programming formulation (Bektas (2006)) proposed for the mTSP solution. DPFSP is correlated to the scheduling of M jobs for F factories and mTSP is a generalization of the well-known traveling salesman problem, where more than one salesman is allowed to be used in the solution. In this sense, the scheduling of the jobs for F separate factories in the DPFSP and the usage of more than one salesman in the mTSP illustrates structural similarity. As a result, the developed new model (Model_3) enables the usage of the subtour elimination constraints proven to be effective. Model_3 uses the same decision variables ($X_{k,j}$ and $C_{j,i}$) with Model_2. In the original mTSP model, the decision variable $X_{k,j}$ is as follows and this decision variable is also pertinent to the DPFSP.

$X_{k,j}$ (in mTSP) → Binary variable that takes value 1 if arc (k, j) is used on the tour; and 0 otherwise.
$X_{k,j}$ (in Model_3) → Binary variable that take-s value 1 if the job j is processed immediately after the job k; and 0 otherwise.
Model_3 also needs the continuous variable ($C_{j,i}$) for deciding the completion time of the job j on the machine i.

The integer linear programming formulation developed by drawing inspiration from the mTSP-assignment based integer programming formulation for the DPFSP solution is as follows:

Objective function: Minimize $C_{\max}$ (Equation 1)
Subject to:

$$\sum_{j=1}^{N} X_{0,j} = F \quad (20)$$

$$\sum_{k=1}^{N} X_{k,0} = F \quad (21)$$

$$\sum_{k=0}^{N} X_{k,j} = 1 \quad j \in \{1, \dots, N\} \mid j \neq k \quad (22)$$

$$\sum_{j=0}^{N} X_{k,j} = 1 \quad k \in \{1, \dots, N\} \mid k \neq j \quad (23)$$

$$U_k - U_j + NX_{k,j} \leq N - 1 \quad k, j \in \{1, \dots, N\} \mid k \neq j \quad (24)$$

And, Constraint sets (15), (16), (17), (18) and (19)

Note that $C_{k,0} = C_{0,l} = 0$. Constraint sets (20) and (21) ensure that exactly F salesmen depart from and return back to node 0 (the depot). Constraint sets (22) and (23) are the usual assignment constraints. Constraints (24) are used for preventing the sub tours, being degenerate tours that are formed between intermediate nodes and not associated to the origin. These constraints are named as subtour elimination constraints.

2.4. Linear Programming Model Based on the mTSP-flow-based Formulation

This model (shown by Model_4), is developed by inspiring the flow-based formulation available in Bektas (2006). In Model_4, the three-index decision variable given below is used.

$X_{k,j,f}$ → Binary variable that takes value 1 if vehicle f (factory) visits the node j (job j) immediately after the node k (job k); and 0 otherwise.

Similarly, Model_4 also requires the continuous variable ($C_{j,i}$) for deciding the completion time of the job j on the machine i.
Objective function: Eq. (1)
Subject to:

$$\sum_{k=0 \mid k \neq j}^{N} \sum_{f=1}^{F} X_{k,j,f} = 1 \quad j \in \{1, \dots, N\} \quad (25)$$

$$\sum_{k=0 \mid k \neq p}^{N} X_{k,p,f} - \sum_{j=0 \mid j \neq p}^{N} X_{p,j,f} = 0 \quad p \in \{0, \dots, N\};\ f \in \{1, \dots, F\} \quad (26)$$

$$\sum_{j=1}^{N} X_{0,j,f} = 1 \quad f \in \{1, \dots, F\} \quad (27)$$

$$U_k - U_j + N \sum_{f=1}^{F} X_{k,j,f} \leq N - 1 \quad k, j \in \{1, \dots, N\} \mid k \neq j \quad (28)$$

$$C_{j,i} \geq C_{k,i} + P_{j,i} + bigM\left(X_{k,j,f} - 1\right) \quad k, j \in \{1, \dots, N\} \mid k \neq j;\ i \in \{1, \dots, M\};\ f \in \{1, \dots, F\} \quad (29)$$

$$X_{k,j,f} \in \{0, 1\} \quad k, j \in \{0, \dots, N\} \mid k \neq j;\ f \in \{1, \dots, F\} \quad (30)$$

And, Constraint sets (15), (17) and (18)

Note that $C_{k,0} = C_{0,l} = 0$. Constraints (25) state that each customer (job) be visited exactly once and (26) are the flow conservation constraints ensuring that once a salesman (factory) visits a customer (job), then he must also depart from the same customer (job). Constraints (27) ensure that each vehicle (factory) is used exactly once and (28) is the extension of the sub tour elimination constraint to a three-index model. Constraint set (29), in a manner similar to the Constraint set (16), specifies that if the job j is scheduled immediately after the job k its processing on each machine i at the factory f cannot begin before processing the job k on the machine i at the factory f finishes.

3. Pure Benders Decomposition Algorithm-based Models

Benders decomposition has been proven a powerful technique for solving specially-structured large-scale linear and mixed-integer programs since its presentation in Benders (1962) (Sherali, and Fraticelli (2002)). The decomposition of a given model into master and subproblem is allowed. No more than a subset of the variables and constraints of the original model is incorporated in the master problem. The subproblem is the original model, the master problem variables of which are fixed, whose solution yields either optimality or feasibility cut for the master problem (Costa et al., (2012)). The Benders decomposition algorithm repeats between the master and sub-problem until an optimal solution is obtained. Readers can refer to the recent survey article presented by Rahmaniani et al. (2017) on the application of the Benders decomposition algorithm to combinatorial optimization problems.

In the following two sub-sections, four different pure Benders decomposition algorithm are developed based on Model_1 and two different pure Benders decomposition algorithm by using Model_3.

3.1. Model_1 Based Pure Benders Decomposition Algorithm

In this sub-section, four diverse pure Benders decomposition algorithms are developed based upon Model_1.

3.1.1. Version 1 of the Model_1 Based Pure Benders Decomposition Algorithm

This version of the pure Benders decomposition algorithm developed based on Model_1 is denominated as PB_Model1_V1. In PB_Model1_V1, after having solved the master model, the whole solution obtained from the master model solution (all decision variables taking the value of 1) is fixed in the subproblem. One large cut is obtained from the subproblem solution and this cut is added to the master model to be solved in the next Benders iteration. The subproblem is the dual of the primal subproblem and the primal subproblem is obtained by excluding the constraints from the original model that are common to the master model.

Let $M(C,X)$ denote the formulation (1)-(9) where $X = |j,k = 1,\dots,N;\ f = 1,\dots,F\}$ and $C = \{C_{k,i,f}|\ k = 1,\dots,N;\ i = 1,\dots,M;\ f = 1,\dots,F\}$ are the vectors of the decision variables. Let's suppose that the variables X have been fixed as $X = \hat{X} = \{X \mid X \text{ satisfies } (2),(3),(9)\}$. The resulting formulation, shown by $M(C,\hat{X})$, consists of only the variables C, and the constraints of which are assigned the dual variables $\alpha = \{\alpha_{k,i,f} \geq 0 \mid k = 1,\dots,N; i = 1,\dots,M; f = 1,\dots,F\}$ corresponding to constraints (4), $\beta = \{\beta_{k,i,f} \geq 0 \mid k = 2,\dots,N; i = 1,\dots,M; f = 1,\dots,F\}$ corresponding to constraints (5), $\gamma = \{\gamma_{k,f} \geq 0 \mid k = 1,\dots,N; f = 1,\dots,F\}$ corresponding to constraints (6), and $\delta = \{\delta_{k,f} = unrestricted \mid k = 1,\dots,N; f = 1,\dots,F\}$ corresponding to constraints (7), respectively. The dual $D = (\alpha,\beta,\gamma,\delta,\hat{X})$ of $M(C,\hat{X})$ is given by the following:

$$\text{Maximize} \quad \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{f=1}^{F}\alpha_{k,i,f}\sum_{j=1}^{N}\hat{X}_{j,k,f}P_{j,i} + \sum_{k=2}^{N}\sum_{i=1}^{M}\sum_{f=1}^{F}\beta_{k,i,f}\sum_{j=1}^{N}\hat{X}_{j,k,f}P_{j,i} \tag{31}$$

Subject to:

$$\delta_{k,f} - \alpha_{k,1,f} \leq 0 \quad k \in \{1,\dots,N\}; f \in \{1,\dots,F\} \tag{32}$$

$$\sum_{k=1}^{N}\sum_{f=1}^{F}\gamma_{k,f} \leq 1 \tag{33}$$

$$\alpha_{1,i,f} - \alpha_{1,i+1,f} - \beta_{2,i,f} \leq 0 \quad i \in \{1,\dots,M-1\};\ f \in \{1,\dots,F\} \tag{34}$$

$$\alpha_{1,M,f} - \beta_{2,M,f} - \gamma_{1,f} \leq 0 \quad f \in \{1,\dots,F\} \tag{35}$$

$$\alpha_{k,i,f} - \alpha_{k,i+1,f} + \beta_{k,i,f} - \beta_{k+1,i,f} \leq 0 \quad k \in \{2,\dots,N-1\}; i \in \{1,\dots,M-1\}; f \in \{1,\dots,F\} \tag{36}$$

$$\alpha_{k,M,f} + \beta_{k,M,f} - \beta_{k+1,M,f} - \gamma_{k,f} \leq 0 \quad k \in \{2,\dots,N-1\}; f \in \{1,\dots,F\} \tag{37}$$

$$\alpha_{N,i,f} - \alpha_{N,i+1,f} + \beta_{N,i,f} \leq 0 \quad i \in \{1,\dots,M-1\}; f \in \{1,\dots,F\} \tag{38}$$

$$\alpha_{N,M,f} + \beta_{N,M,f} - \gamma_{N,f} \leq 0 \quad f \in \{1,\dots,F\} \tag{39}$$

The model (master model) consisting of the Constraint sets (2), (3), and (9) always generates a viable solution, which, in turn, means that $D = (\alpha,\beta,\gamma,\delta,\hat{X})$ is always feasible for a given $\hat{X}$, and for an optimal solution $(\alpha,\beta,\gamma,\delta)$ of the dual problem, one obtains the following Benders optimality cuts:

$$z \geq \sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{f=1}^{F}A_{j,k,f}X_{j,k,f} + \sum_{j=1}^{N}\sum_{k=2}^{N}\sum_{f=1}^{F}B_{j,k,f}X_{j,k,f}$$

where z is a lower bound on the optimal solution value of $M(C,X)$, $A_{j,k,f} = \sum_{i=1}^{M}\hat{\alpha}_{k,i,f}\,P_{j,i}$ and $B_{j,k,f} = \sum_{i=2}^{M}\hat{\beta}_{k,i,f}\,P_{j,i}$. Using this result, we are now ready to present the following reformulation of $M(C,X)$, referred to as the master problem constructed using the set P_D of extreme points of $D = (C,X)$ and shown as MP(P_D) below:

$$\text{Minimize } z \tag{40}$$

Subject to:

$$\sum_{k=1}^{N}\sum_{f=1}^{F}X_{j,k,f} = 1 \quad j \in \{1,\dots,N\} \tag{41}$$

$$\sum_{j=1}^{N}\sum_{f=1}^{F}X_{j,k,f} = 1 \quad k \in \{1,\dots,N\} \tag{42}$$

$$z \geq \sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{f=1}^{F}A_{j,k,f}X_{j,k,f} + \sum_{j=1}^{N}\sum_{k=2}^{N}\sum_{f=1}^{F}B_{j,k,f}X_{j,k,f} \quad (\alpha,\beta,\gamma,\delta) \in P_D \tag{43}$$

$$X_{j,k,f} \in \{0,1\} \quad k \in \{1,\dots,N\};\ i \in \{1,\dots,M\};\ f \in \{1,\dots,F\} \tag{44}$$

Since the MP includes a large number of optimality cuts, it can be solved by using a cutting plane algorithm in practice, normally starting with MP ($\emptyset$) with no optimality cuts (43) and generating the cuts on an as-needed basis. The algorithm usually stops after having solved a certain MP(P), where $P \subset P_D$.

3.1.2. Version 2 of the Model_1 Based Pure Benders Decomposition Algorithm

In this version (shown by PB_Model1_V2) as a more dissimilar approach, rather than producing only one cut over the solution received from the master model, the jobs found in the solution received from the master model are decomposed in accordance with the factories to which they are assigned. Subsequently, separate cuts are obtained for the factory solutions to which at least 1 job is assigned in the master model and afterward, each cut obtained is added to the master problem independently for all factories for the subsequent master problem solution. In other words, let's assume that the number of factories (F) is 2 and at least 1 job is assigned to each factory in the master problem solution. The subproblem is solved by using the cluster of jobs assigned to each factory and the produced cut is separately added to the master model for each factory. Since there is at least 1 job assigned to each factory in this example, 4 cuts in total are generated in each iteration and added to the master model. To explain with a more explanatory example, consider a problem with N = 6 and F = 2, one of the possible MP solutions is $X_{1,1,1}$ =$X_{2,3,1}$ =$X_{3,6,1}$ =$X_{4,2,2}$ =$X_{5,4,2}$ =$X_{6,5,2}$ = 1. In this example, the jobs 1, 2 and 3 are allocated to factory 1 with this order {1, 2, 3} while the other jobs are assigned to factory 2 with the permutation or sequence {4, 5, 6}. According to this multiple cut generating and adding strategy, 2 different submodels consisting of the jobs {1, 2, 3} and {4, 5, 6} are generated. Each submodel is solved and each cut obtained from the submodel is added to the master model for every 2 factories. Since at least 1 job is assigned to each factory in the example, 2 different submodels are generated and solved. Each cut is added to the master model for every 2 factories. Thus, 4 cuts in total are added to the master model for this example.

In this version, the dual problem is independent from the factory indices. The main steps of this version of the problem are given below.

Input: Problem data, allowable optimality gap $\varepsilon \geq 0$
1. Set LB = -∞, UB = ∞
2: while ($LB \leq UB$**) do**

3: Solve MP in order to obtain $\hat{X} = \{X | X \text{ satisfies } (2), (3), \text{ and } (9)\}$, and obtain solution value, *obj_master*
4: **if (***LB* < *obj_master***)**
5: *LB = obj_master;*
6: **endif**
7: **for (***f* = 1 to *F***)** //factory indices
8: *counter* = 1;
9: **for (***j*= 1 to *N***)** //job indices
10: **for (***k* = 1 to *N***)** //position indices
11: **if (**$X_{j,k,f} == 1$**)**
12: *JL* [*counter*] = *j*; //job list
13: *PL* [*counter*] = *k*; //position list
14: *counter* = *counter* + 1;
15: **endif**
16: **endfor**
17: **endfor**
18: *NJ = counter -1; //number of job in the factory f*
19: **if (***NJ* > 0**)**
20: Solve SP depending on the jobs in *JL* and obtain the solution value of SP, *obj_sub*
21: **if (***UB* > *obj_sub***)**
22: *UB = obj_sub;*
23: **endif**
25: **for (***g*=1 to *F***)** //g used for denoting the factories
26: Insert optimality cut to MP for factory *g* by using the jobs in *JL* and theirs original positions in list *PL*
27: **endfor**
24: **endif**
28: endfor
29: endwhile
30: Report the best solution found by the last MP solution

The subproblem is the dual of the model given below. Please note that the model has been transformed into the classic flow shop model. If there is at least 1 job assigned to at least one factory whatsoever in the solution obtained from the master problem, the subproblem is solved for that factory and the obtained cut is inserted again into the master problem separately by considering the original positions of the decision variables coming from the master problem solution. That is to say, the same cut is inserted into the master problem separately for each factory by considering the positions of the decision variables coming from the master problem solution.

$$\text{Minimize } C_{\max} = C_{NJ,M} \tag{45}$$

$$C_{k,i} \geq C_{k,i-1} + \sum_{j=1}^{NJ} \hat{X}_{JL[j],k} P_{JL[j],i} \quad k \in \{1,\dots,NJ\};\ i \in \{1,\dots,M\} \tag{46}$$

$$C_{k,i} \geq C_{k-1,i} + \sum_{j=1}^{NJ} \hat{X}_{JL[j],k} P_{JL[j],i} \quad k \in \{2,\dots,NJ\};\ i \in \{1,\dots,M\} \tag{47}$$

$$C_{k,0} = 0 \quad k \in \{1,\dots,NJ\} \tag{48}$$

$$C_{k,i} \geq 0 \quad k \in \{1,\dots,NJ\};\ i \in \{1,\dots,M\} \tag{49}$$

In the formulation (45)-(49), $\hat{X} = \{\widehat{X_{j,k}} | j \in JL[c], c = 1,\dots,NJ;\ k = 1,\dots,NJ\}$ comes from a factory solution of MP, and $C = \{C_{k,i} | k = 1,\dots,NJ;\ i = 1,\dots,M\}$ are the vectors of the decision variables. The resulting formulation, shown by $M(C,\hat{X})$, consists of the variables C only, and the constraints of which are assigned the dual variables $\alpha = \{\alpha_{k,i} \geq 0 \mid k = 1,\dots,NJ;\ i = 1,\dots,M\}$ corresponding to constraints (46), $\beta = \{\beta_{k,i} \geq 0 \mid k = 2,\dots,NJ;\ i = 1,\dots,M\}$ corresponding to constraints (47), and $\delta = \{\delta_k = unrestricted \mid k = 1,\dots,NJ\}$ corresponding to constraints (48), respectively. The dual $D = (\alpha, \beta, \delta, \hat{X})$ of $M(C,\hat{X})$ is given in the formulation (50)-(57).

$$\text{Maximize} \sum_{k=1}^{NJ}\sum_{i=1}^{M} \alpha_{k,i} \sum_{j=1}^{NJ} \hat{X}_{JL[j],k} P_{JL[j],i} + \sum_{k=2}^{NJ}\sum_{i=1}^{M} \beta_{k,i} \sum_{j=1}^{NJ} \hat{X}_{JL[j],k} P_{JL[j],i} \tag{50}$$

Subject to:

$$\delta_k - \alpha_{k,1} \leq 0 \quad k \in \{1,\dots,NJ\} \tag{51}$$

$$\alpha_{1,i} - \alpha_{1,i+1} - \beta_{2,i} \leq 0 \quad i \in \{1,\dots,M-1\} \tag{52}$$

$$\alpha_{1,M} - \beta_{2,M} \leq 0 \tag{53}$$

$$\alpha_{k,i} - \alpha_{k,i+1} + \beta_{k,i} - \beta_{k+1,i} \leq 0 \quad k \in \{2,\dots,NJ-1\};\ i \in \{1,\dots,M-1\} \tag{54}$$

$$\alpha_{k,M} + \beta_{k,M} - \beta_{k+1,M} \leq 0 \quad k \in \{2,\dots,NJ-1\} \tag{55}$$

$$\alpha_{NJ,i} - \alpha_{NJ,i+1} + \beta_{NJ,i} \leq 0 \quad i \in \{1,\dots,M-1\} \tag{56}$$

$$\alpha_{NJ,M} + \beta_{NJ,M} \leq 1 \tag{57}$$

If any factory in the solution obtained from the master problem comprises at least 1 job, the cuts are added to the master model for all factories as given below.

for (*g*=1 to *F***)** //g used for denoting the factories

$$z \geq \sum_{j=1}^{NJ}\sum_{k=1}^{NJ} A_{JL[j],PL[k]} X_{JL[j],PL[k],g} + \sum_{j=1}^{NJ}\sum_{k=2}^{NJ} B_{JL[j],PL[k]} X_{JL[j],PL[k],g}$$

endfor

where z is a lower bound on the optimal solution value of $M(C,X)$, $A_{JL[j],PL[k]} = \sum_{i=1}^{M} \hat{\alpha}_{k,i} P_{JL[j],i}$ and $B_{JL[j],PL[k]} = \sum_{i=2}^{M} \hat{\beta}_{k,i} P_{JL[j],i}$.

In each Benders iteration, F*F cuts (maximum) are inserted in this version (if each factory includes at least 1 job in the master problem-solution).

3.1.3. Version 3 of the Model_1 Based Pure Benders Decomposition Algorithm

In this version (represented by PB_Model1_V3), differently from PB_Model1_V2, the makespans are calculated separately for the factories to which at least 1 job is assigned in the master problem solution. The subproblem is solved only for the factory yielding the longest makespan and the cut is inserted into the MP only for the factory yielding the longest makespan. In this version, only one cut is subsequently inserted in each Benders iteration.

After having obtained a job permutation from a factory solution of the master problem solution, the makespan is calculated by using a completion time matrix as proposed by Onwubolu and Davendra (2006). For illustrating the operating principle of the completion time matrix, let's apply it to a 5-machine and 10-job problem. Processing times are given in Table 2. Suppose that the

job permutation of i^{th} factory obtained from the master problem solution is $\pi = \{3, 5, 2, 1, 4\}$.

Table 2. Processing times for a 10×5 example problem.

Machine	Job									
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
1	5	7	4	3	6	7	5	3	6	8
2	6	5	7	6	7	5	6	5	1	6
3	7	8	3	8	5	8	7	8	7	4
4	8	6	5	5	8	5	6	5	5	4
5	4	4	8	7	3	7	7	8	8	2

The completion time matrix is shown below with a makespan value of 58, which is the last entry in the matrix.

$$[C] = \begin{bmatrix} 4 & 10 & 17 & 22 & 25 \\ 11 & 18 & 23 & 29 & 35 \\ 14 & 23 & 31 & 38 & 46 \\ 19 & 31 & 37 & 46 & 51 \\ 27 & 34 & 41 & 50 & 58 \end{bmatrix}$$

3.1.4. Version 4 of the Model_1 Based Pure Benders Decomposition Algorithm

In this version (shown by PB_Model1_V4), similarly, with PB_Model1_V3, the makespan is calculated separately for the factories to which at least 1 factory is assigned in the master problem solution. The subproblem is solved only for the factory yielding the longest makespan and the cut obtained from the subproblem solution is inserted separately for all factories as is the case in PB_Model1_V2 (the same cut is applied to all factories). In this version, F cuts are subsequently inserted in each Benders iteration.

3.2. Model_3 Based Pure Benders Decomposition Algorithms

As will be seen in the Computation results section, the model called Model_3 has given the best performance among the models including the big M constraint. Therefore, 2 different Model_3 based pure Benders decomposition algorithms are developed in this Section. As also underlined often previously, Model_2 was presented by Naderi and Ruiz (2010) as the model yielding the best performance. A separate pure Benders decomposition algorithm is also developed for Model_2. The master problem for the developed Model_2 consists of the Constraint sets (10), (11), (12), (13), (14), and (19). But this generated master model may also yield unfeasible solutions and also needs the feasibility cuts accordingly. The Model_2 based pure Benders decomposition algorithm is not included in the paper since it produces very bad performance when compared to the Model_3 based Benders algorithm.

3.2.1. Version 1 of the Model_3 Based Pure Benders Decomposition Algorithm

In a manner similar to PB_Model1_V1, in this version (represented by PB_Model3_V1), the subproblem is solved by fixing all of the decision variables obtained from the master model solutions in the subproblem. In this version, one large cut indicating the entire solution is inserted into master the problem in each Benders iteration. The master model consists of the Constraint sets (19) – (24) in the Benders algorithm of this model and the subproblem is the dual of the model given below.

$$\text{Minimize } C_{max} \tag{58}$$

Subject to:

$$C_{j,i} \geq C_{j,i-1} + P_{j,i} \quad j \in \{1, \dots, N\};\ i \in \{1, \dots, M\} \tag{59}$$
$$C_{j,i} \geq C_{k,i} + P_{j,i} + bigM\left(\hat{X}_{k,j} - 1\right) \tag{60}$$
$$k \in \{0,1, \dots, N\}, j \in \{1, \dots, N\} \mid k \neq j;\ i \in \{1, \dots, M\}$$
$$C_{max} \geq C_{j,M} \quad j \in \{1, \dots, N\} \tag{61}$$
$$C_{j,0} = 0 \quad j \in \{1, \dots, N\} \tag{62}$$
$$C_{0,i} = 0 \quad i \in \{1, \dots, M\} \tag{63}$$
$$C_{j,i} \geq 0 \quad j \in \{1, \dots, N\};\ i \in \{1, \dots, M\} \tag{64}$$

In the formulation (58)-(64), $\hat{X} = \left\{\hat{X}_{k,j} \middle| k \in \{0,1, \dots, N\},\ j \in \{1, \dots, N\} | k \neq j\right\}$ comes from the MP solution, and $C = \{C_{k,i} |\ k = 1, \dots, NJ;\ i = 1, \dots, M\}$ are the vectors of the decision variables. Note that there is no need for the decision variables $\hat{X} = \left\{\hat{X}_{k,0} \middle| k \in \{0,1, \dots, N\}\right\}$ in this formulation**.** The resulting formulation, shown by $M(C, \hat{X})$, consists of the variables *C* only and the constraints of which are assigned the dual variables;
$\alpha = \left\{\alpha_{j,i} \geq 0 \ \middle|\ j \in \{1, \dots, N\};\ i \in \{1, \dots, M\}\right\}$ for constraints set (59),
$\beta = \left\{\beta_{j,k,i} \geq 0 \ \middle|\ j \in \{1, \dots, N\};\ k \in \{0,1, \dots, N\} | k \neq j;\ i \in \{1, \dots, M\}\right\}$ for constraints set (60),
$\gamma = \left\{\gamma_j \geq 0 \ \middle|\ j \in \{1, \dots, N\}\right\}$ for constraint set (61),
$\delta = \left\{\delta_j = unrestricted \ \middle|\ j = 1, \dots, N\right\}$ for constraints (62),
$\theta = \{\theta_i = unrestricted \mid i = 1, \dots, M\}$ for constraints (63), respectively.
The dual $D = \left(\alpha, \beta, \gamma, \delta, \theta, \hat{X}\right)$ of $M(C, \hat{X})$ is given in the formulation (65)-(70).

$$\begin{aligned} \text{Maximize } & \sum_{j=1}^{N} \sum_{i=1}^{M} \alpha_{j,i} P_{j,i} \\ & + \sum_{k=0}^{N} \sum_{j=1 | j \neq k}^{N} \sum_{i=1}^{M} \beta_{j,k,i} \left[P_{j,i} \right. \\ & \left. + bigM\left(\hat{X}_{k,j} - 1\right)\right] \end{aligned} \tag{65}$$

Subject to:

$$\sum_{j=1}^{N} \gamma_j \leq 1 \tag{66}$$

$$\alpha_{j,i} - \alpha_{j,i+1} - \sum_{k=1|k\neq j}^{N} \beta_{k,j,i} + \sum_{k=0|k\neq j}^{N} \beta_{j,k,i} \leq 0 \quad j \in \{1,\dots,N\};\ i \in \{1,\dots,M\} \tag{67}$$

$$\alpha_{j,M} - \gamma_j - \sum_{k=1|k\neq j}^{N} \beta_{k,j,M} + \sum_{k=0|k\neq j}^{N} \beta_{j,k,M} \leq 0 \quad j \in \{1,\dots,N\} \tag{68}$$

$$\delta_j - \alpha_{j,1} \leq 0 \quad j \in \{1,\dots,N\} \tag{69}$$

$$\theta_i - \sum_{j=1}^{N} \beta_{j,0,i} \leq 0 \quad i \in \{1,\dots,M\} \tag{70}$$

The model (master model) constantly generates a viable solution, which, in turn, means that $D = (\alpha, \beta, \gamma, \delta, \theta, \hat{X})$ is always feasible for a given $\hat{X}$, and for an optimal solution $(\alpha, \beta, \gamma, \delta, \theta)$ of the dual problem, one obtains the following Benders optimality cuts:

$$z \geq total + \sum_{k=0}^{N} \sum_{j=1|j\neq k}^{N} A_{k,j} X_{k,j}$$

where z is a lower bound on the optimal solution value of $M(C,X)$,

$$total = \sum_{j=1}^{N} \sum_{i=1}^{M} \hat{\alpha}_{j,i} P_{j,i} + \sum_{j=1}^{N} \sum_{k=0|k\neq j}^{N} \sum_{i=1}^{M} \hat{\beta}_{j,k,i} P_{j,i} - \sum_{j=1}^{N} \sum_{k=0|k\neq j}^{N} \sum_{i=1}^{M} \hat{\beta}_{j,k,i} bigM$$

$$A_{k,j} = \sum_{i=1}^{M} \hat{\beta}_{j,k,i} bigM$$

Using this result, we are now ready to present the following reformulation of $M(C,X)$, referred to as the master problem constructed by using the set P_D of extreme points of $D = (C,X)$ and shown as MP(P_D) below:

Minimize z (71)

Subject to:

$$z \geq total + \sum_{k=0}^{N} \sum_{j=1|j\neq k}^{N} A_{k,j} X_{k,j} \tag{72}$$

And, Constraint set (19) – (24)

As the MP includes a large number of optimality cuts, it can be solved by using a cutting plane algorithm in practice, normally starting with MP($\emptyset$) with no optimality cuts (72) and generating the cuts on an as-needed basis. The algorithm usually stops after having solved a certain MP(P), where $P \subset P_D$.

3.2.2. Version 2 of the Model_3 Based Pure Benders Decomposition Algorithm

In this version (represented by PB_Model3_V2), similarly, with PB_Model1_V3, the makespan is calculated separately for the factories to which at least 1 factory is assigned in the MP solution. The subproblem is run only for the factory yielding the longest (maximum) makespan and the cut for the master model is generated only for the jobs causing the maximum makespan and inserted to the master model. For example, consider a problem with N = 6 and F = 2, one of the possible solutions of MP is $X_{0,1} = X_{1,3} = X_{3,6} = X_{6,0} = X_{0,4} = X_{4,2} = X_{2,1} = X_{1,5} = X_{5,0} = 1$; that is,{0, 3, 6, 0, 4, 2, 1, 5, 0}. In this example, jobs 3 and 6 are allocated to factory 1 with this order {3,6} while the other jobs are assigned to factory 2 with the permutation or sequence {4, 2, 1, 5}. Once the makespans of the factories in the example by the makespan calculation method in PB_Model1_V3, let's assume that the 2nd factory is the factory yielding the longest makespan. In this version, only the cut consisting of the jobs {4, 2, 1, 5} at the factory is generated and inserted into the master model.

Similarly, the master model consists of the Constraint sets (19)-(24) in the Benders algorithm developed for this version. While modeling the subproblem, let's assume that a list denominated *JL* holds the jobs assigned to the factory yielding the longest (maximum) makespan in the master model solution. Let's start it with the position index. Let 0 be in the position 0 and total number of jobs except for 0 available in the *JL* list be *NJ*. For example, for the example above, *JL*= {0, 4, 2, 1, 5} and also *NJ* becomes equal to 4. The subproblem formed by the jobs in the *JL* list is the dual of the model, the primal of which is given below.

Minimize C_{max} (73)

Subject to:

$$C_{JL[j],i} \geq C_{JL[j],i-1} + P_{JL[j],i} \quad j \in \{1,\dots,NJ\};\ i \in \{1,\dots,M\} \tag{74}$$

$$C_{JL[j],i} \geq C_{JL[k],i} + P_{JL[j],i} + bigM\left(\hat{X}_{JL[k],JL[j]} - 1\right) \quad k \in \{0,1,\dots,NJ\}, j \in \{1,\dots,NJ\} \mid k \neq j;\ i \in \{1,\dots,M\} \tag{75}$$

$$C_{max} \geq C_{JL[j],M} \quad j \in \{1,\dots,NJ\} \tag{76}$$

$$C_{JL[j],0} = 0 \quad j \in \{1,\dots,NJ\} \tag{77}$$

$$C_{0,i} = 0 \quad i \in \{1,\dots,M\} \tag{78}$$

$$C_{JL[j],i} \geq 0 \quad j \in \{1,\dots,NJ\};\ i \in \{1,\dots,M\} \tag{79}$$

In the formulation (73)-(79), $\hat{X} = \{\hat{X}_{JL[k],JL[j]} | k \in \{0,1,\dots,NJ\},\ j \in \{1,\dots,NJ\} | k \neq j\}$ comes from the MP solution, and $C = \{C_{JL[k],i} |\ k = 1,\dots,NJ;\ i = 1,\dots,M\}$ are the vectors of the decision variables. Note that there is no need for the decision variables $\hat{X} = \{\hat{X}_{JL[k],0} | k \in \{0,1,\dots,NJ\}\}$ in this formulation. The resulting formulation, shown by $M(C,\hat{X})$, consists of the variables C only, and the constraints of which are assigned the dual variables; In the formulation (73)-(79), $\hat{X} = \{\hat{X}_{JL[k],JL[j]} | k \in \{0,1,\dots,NJ\},\ j \in \{1,\dots,NJ\} | k \neq j\}$ comes from the MP solution, and $C = \{C_{JL[k],i} |\ k = 1,\dots,NJ;\ i = 1,\dots,M\}$ are the vectors of the decision variables. Note that there is no need for the decision variables $\hat{X} = \{\hat{X}_{JL[k],0} | k \in \{0,1,\dots,NJ\}\}$ in this formulation. The resulting formulation, shown by $M(C,\hat{X})$, consists of the variables C only, and the constraints of which are assigned the dual variables;

$\alpha = \{\alpha_{JL[j],i} \geq 0 \mid j \in \{1,\dots,NJ\};\ i \in \{1,\dots,M\}\}$ for constraints set (74),
$\beta = \{\beta_{JL[j],JL[k],i} \geq 0 \mid j \in \{1,\dots,NJ\};\ k \in \{0,1,\dots,NJ\} | k \neq j;\ i \in \{1,\dots,M\}\}$ for constraints set (75),
$\gamma = \{\gamma_{JL[j]} \geq 0 \mid j \in \{1,\dots,NJ\}\}$ for constraint set (76),
$\delta = \{\delta_{JL[j]} = unrestricted \mid j = 1,\dots,NJ\}$ for constraints (77),
$\theta = \{\theta_i = unrestricted \mid i = 1,\dots,M\}$ for constraints (78), respectively.

The dual $D = (\alpha, \beta, \gamma, \delta, \theta, \hat{X})$ of $M(C, \hat{X})$ is given by the following:

$$\text{Maximize} \quad \sum_{j=1}^{NJ}\sum_{i=1}^{M} \alpha_{j,i} P_{JL[j],i} + \sum_{k=0}^{NJ}\sum_{j=1|j\neq k}^{NJ}\sum_{i=1}^{M} \beta_{j,k,i}\left[P_{JL[j],i} + bigM\left(\hat{X}_{JL[k],JL[j]} - 1\right)\right] \tag{80}$$

Subject to:

$$\sum_{j=1}^{NJ} \gamma_{JL[j]} \leq 1 \tag{81}$$

$$\alpha_{JL[j],i} - \alpha_{JL[j],i+1} - \sum_{k=1|k\neq j}^{NJ} \beta_{JL[k],JL[j],i} + \sum_{k=0|k\neq j}^{NJ} \beta_{JL[j],JL[k],i} \leq 0 \tag{82}$$

$j \in \{1,\dots,NJ\};\ i \in \{1,\dots,M\}$

$$\alpha_{JL[j],M} - \gamma_{JL[j]} - \sum_{k=1|k\neq j}^{N} \beta_{JL[k],JL[j],M} + \sum_{k=0|k\neq j}^{N} \beta_{JL[j],JL[k],M} \leq 0 \tag{83}$$

$j \in \{1,\dots,NJ\}$

$$\delta_{JL[j]} - \alpha_{JL[j],1} \leq 0 \quad j \in \{1,\dots,NJ\} \tag{84}$$

$$\theta_i - \sum_{j=1}^{NJ} \beta_{JL[j],0,i} \leq 0 \quad i \in \{1,\dots,M\} \tag{85}$$

The model (master model) always generates a feasible solution. This, in turn, means that $D = (\alpha, \beta, \gamma, \delta, \theta, \hat{X})$ is always feasible for a given $\hat{X}$, and for an optimal solution $(\alpha, \beta, \gamma, \delta, \theta)$ of the dual problem, one obtains the following Benders optimality cuts:

$$z \geq total + \sum_{k=0}^{N}\sum_{j=1|j\neq k}^{N} A_{k,j} X_{JL[k],JL[j]}$$

where z is a lower bound on the optimal solution value of $M(C, X)$,

$$total = \sum_{j=1}^{N}\sum_{i=1}^{M} \hat{\alpha}_{JL[j],i} P_{JL[j],i} + \sum_{j=1}^{N}\sum_{k=0|k\neq j}^{N}\sum_{i=1}^{M} \hat{\beta}_{JL[j],JL[k],i} P_{JL[j],i} - \sum_{j=1}^{N}\sum_{k=0|k\neq j}^{N}\sum_{i=1}^{M} \hat{\beta}_{JL[j],JL[k],i} bigM$$

$$A_{k,j} = \sum_{i=1}^{M} \hat{\beta}_{JL[j],JL[k],i} bigM$$

Using this result, we are now ready to present the following reformulation of $M(C, X)$, referred to as the master problem constructed by using the set P_D of extreme points of $D = (C, X)$ and shown as MP(P_D) below:

$$\text{Minimize } z \tag{86}$$

Subject to:

$$z \geq total + \sum_{k=0}^{N}\sum_{j=1|j\neq k}^{N} A_{k,j} X_{JL[k],JL[j]} \tag{87}$$

And, Constraint set (19) – (24)

Since the MP includes a large number of optimality cuts, it can be solved by using a cutting plane algorithm in practice, normally starting with MP($\emptyset$) with no optimality cuts (87) and generating the cuts on an as-needed basis. The algorithm usually stops after having solved a certain MP(P), where $P \subset P_D$.

4. Hybrid Benders Decomposition Algorithm

This section describes a hybrid algorithm that uses Benders Decomposition with a simple yet effectual enhancement mechanism entailing the generation of additional cuts by using LS3 algorithm (Ruiz, Pan, and Naderi (2019)) to help accelerate convergence. As also seen from the computational comparison section, the best performance is shown by PB_Model1_V1 (A single optimality cut inserting) among the pure Benders versions. Therefore, a hybrid Benders decomposition algorithm is developed based on PB_Model1_V1 in this section. In this hybrid Benders decomposition algorithm, one extra cut is generated and inserted into the MP (master problem) in each Benders iteration by using the local search algorithm denominated LS3 developed by Ruiz, Pan, and Naderi (2019). Different from the original LS3 algorithm, the LS3 algorithm used in this paper has taken its preliminary solution from the master problem solution. The LS3 algorithm does not oblige any algorithm parameters.

The LS3 algorithm, if summarized in a few words, starts with taking the MP solution. Then, the factory generating the C_{max} is selected. A job is arbitrarily extracted from this factory and inserted into all possible positions in all factories (including the one generating the makespan). If the best C_{max} in all these insertions is better than the starting C_{max} , the job is relocated and the search starts again from the beginning; otherwise, the job is reinserted back into its original position and the search continues. The procedure iterates until all jobs from the factory generating

the C_{max} will have been tested (randomly and without repetition). The pseudo code of LS3 is given in the following.

procedure LS3 (π= {π_1, π_2, ..., π_F }) //It starts with taking the MP solution
1: $C^*_{\max} = \max^F_{f=1}\{C_{\max}(\pi_1), \{C_{\max}(\pi_2), \dots, \{C_{\max}(\pi_F)\}$
2: $f_{\max} = \arg(C^*_{\max})$ % (factory with the largest $C_{\max}$)
3: *Cnt* = 0
4: while (*Cnt* ≤ $|\pi_{f_{\max}}|$) **do** % (all jobs in factory $f_{\max}$)
5: Randomly extract, without repetition, a job *j* from position *k* of $\pi_{f_{\max}}$
6: **for** (*f* = 1to *F*)
7: Test job *j* in all possible positions of π_f % (Taillard-BSIG accelerations)
8: $C^f_{\max}$ is the lowest $C_{\max}$ obtained
9: p^f is the position where the lowest $C_{\max}$ obtained
10: endfor
11: $f_{\min} = \arg(\min^F_{f=1}(C^f_{\max}))$
12: if ($C^f_{\max} < C^*_{\max}$ **)**
13: Place job *j* at position p^f of factory $f_{\min}$
14: $C^*_{\max} = \max^F_{f=1}\{C_{\max}(\pi_1), \{C_{\max}(\pi_2), \dots, \{C_{\max}(\pi_F)\}$
15: $f_{\max} = \arg(C^*_{\max})$ // (factory with the largest $C_{\max}$)
16: *Cnt* = 0
17: elseif
18: Return job *j* to position *k* of $f_{\max}$
19: *Cnt* = *Cnt* + 1
20: endif
21: endwhile

The hybrid Benders decomposition is an iterative algorithm generating optimality cuts (43) in each iteration based on an optimal MP solution $\hat{X}$ and uses $\hat{X}$ as an input to the LS3 to generate a neighbor solution $\hat{X}_{LS3}$, inducing an supplementary optimality cut inserted into the master problem. In the hybrid Benders algorithm, 2 cuts, one of which is generated from a previous master model problem solution and the other from the LS3 algorithm, are inserted to the master model in each Benders iteration. The pseudo-code of the proposed algorithm is given in the following.

Input: Problem data, allowable optimality gap $\varepsilon \geq 0$
1. Set *LB* = -∞, *UB* = ∞
2: while (*LB* ≤ *UB*) **do**
3: Solve MP in order to obtain $\hat{X} = \{X | X \text{ satisfies } (2), (3), \text{ and } (9)\}$, and obtain solution value, *obj_master*
4. **if** (*LB* < *obj_master*)
5: *LB* = *obj_master;*
6: **endif**
7: Solve SP depending on Constraint sets (31) – (39) with $\hat{X}$, obtain the solution value of SP, *obj_sub*
8: **if** (*UB* > *obj_sub*)
9: *UB* = *obj_sub;*
10: endif
11: Considering the MP solution ($\hat{X}$) as the initial solution of LP3, run the LS3 algorithm to get a new solution ($\hat{X}_{LS3}$)
12: Solve the SP depending on Constraint sets (31) – (39) with $\hat{X}_{LS3}$, obtain the solution value of SP, *obj_sub_LS3*
13: if (*UB* > *obj_sub_LS3*)
14: *UB* = *obj_sub_LS3;*
15: endif
16. Insert the optimality cuts into the MP for the solutions $\hat{X}$ and $\hat{X}_{LS3}$ //Inserting Constraint set (43) to MP for both solutions
17: endwhile
18: Report the best solution found by the last MP solution

5. Computational Results

The experiments are conducted in three main stages. First, the mathematical models and automated Benders decomposition versions of these models (available within the software) are compared with each other. The pure Benders algorithms proposed in the subsequent sub-section are compared with each other. Then the developed hybrid Benders decomposition algorithm was compared to the other methods. The algorithm and its variants are coded in Visual C++, using CPLEX 12.7.1 as the solver. An Intel Core i5-2450M computer with a 2.5 GHz CPU and 4 GB memory was used. The tests are accomplished on 84 problem instances of the distributed permutation flowshop scheduling problem available at http://soa.iti.es. The data used in the experiments are taken from the data file named DPFSP_Small. The problem instances in the DPFSP_Small data file has been demonstrated with 4 main indices. For instance, such as I_2_4_2_1 and I_4_16_5_1. The numbers here indicate the dataset number {1, 2, 3, 4, 5} and how many factories {2, 3, 4}, how many jobs {4, 6, 8, 10, 12, 14, 16}, and how many machines {2, 3, 4, 5} are available in the dataset, respectively. In the experiments, the datasets whose last index is 1 in the data file named DPFSP_Small are solved. Furthermore, all models are solved under a limitation of 1800 seconds. In the execution of the variants of the Benders algorithm, the presolver of the CPLEX is deactivated, whereas, in the solution of the mathematical models, this decision is left to the CPLEX solver. Deterministic mode with four threads is used in the CPLEX solver in all exact algorithms runs.

5.1. Performance Comparisons of the Mathematical Models

In this stage, the four models given in Section 2.1 are evaluated with the automatic Benders decomposition algorithm obtainable as ready-to-use within the CPLEX. The automatic Benders decomposition algorithm of the CPLEX (shown by ABD) is applied to 4 mathematical models given in Section 2.1. Among these four models, only Model_1 is able to produce solutions with the automatic Benders decomposition algorithm. The other 3 models are unable to produce solution with the automatic Benders algorithm. The summarized results are shown in Table 3 and the results are shown in Figure 1 to facilitate the reading of the data given in Table 3, as well. As seen from Table 3 and Figure 1, maximum number of instances are solved optimally by Model_1 (56 instances), Model_3 (54 instances), Model_2 (53 instances), Model_1_ABD (48 instances), and Model_4 (46 instances), respectively. In terms of average time, the Model_3 yielded the lowest time average with 56.40 (average of the times of 54 optimal solutions). 46 instances are also solved optimally by all methods. The instances solved jointly are solved by Model_3 in the shortest average period (5.91 seconds on average). Model_3 is followed by Model_2, Model_1, Model_1_ABD, and Model 4 with 6.81, 24.91, 49.69, and 109.25 seconds in average, respectively. The worst performance is given by Model_4 with 109.25 seconds on average. In terms of suboptimal solutions gaps, Model_1 yielded the lowest gap value with 9.13 on average. The best performance is given by Model_1

with a gap average of 9.48 in 24 common instances that cannot be solved optimally. Model_1 is followed by Model_1_ABD, Model_3, Model_2, and Model_4 with the gap averages of 23.45, 25.83, 27.60, and 30.60, respectively. The maximum best integer solution is yielded by Model_1 with 82 solutions. Model_1 is followed by Model_3, Model_2, Model_1_ABD, and Model_4 with 71, 67, 61, and 58 solutions, respectively. If we consider all performance values, Model_3 is given better performance compared to Model_2 and it can be evidently said that the worst performance is given by Model_4 among 4 models. It is not possible to make a definitive distinction as to whether Model_1 or Model_3 is the best. Moreover, different comparisons can be made within Model_3 by using different sub tour elimination constraints available in the literature.

5.2. Performance Evaluation of the Pure Benders Decomposition Algorithms

In this stage, 4 different pure Benders decomposition algorithms generated from Model_1 and 2 different Benders decomposition algorithms generated from Model_3 are compared. Summary results are shown in Table 4 and the results are also shown in Figure 2 to facilitate the reading of the data given in Table 4. As seen from Table 4 and Figure 2, maximum optimal solutions are found by PB_Model1_V1 with 54 solutions. PB_Model1_V1 was followed by PB_Model1_V3, PB_Model3_V1, PB_Model3_V2, PB_Model1_V2, and PB_Model1_V4 with 51, 35, 35, 30, and 30 optimal solutions, respectively. 29 common instances can be solved optimally by all 6 methods. The instances solved jointly are solved in the shortest time by PB_Model3_V1 with an average time of 9.94; moreover, 30 widespread instances cannot be solved optimally by all 6 methods. The lowest average gap value is yielded by PB_Model1_V1 with 39.06 in 30 instances that cannot be solved jointly. PB_Model1_V1 yields the highest average lower bound and lowest upper bound values among the 4 Model_1 based Benders decomposition algorithm version for the 30 instances that cannot be solved jointly. PB_Model3_V1 and PB_Model3_V2 models do not succeed in raising the lower bound that cannot be solved optimally and the gap values in the instances that cannot be solved by them optimally are 100%. On the other hand, the shortest average solution time is reached by these two models (PB_Model3_V1 and PB_Model3_V2) in the instances they are able to solve optimally. As a general interpretation, it can be said that PB_Model1_V1 is the most attractive Benders algorithm since it can solve maximum number of examples optimally and also yields the lowest average gap value in the instances that cannot be solved optimally.

5.3. Performance of the proposed hybrid Benders decomposition algorith

In this stage, the results of the proposed hybrid Benders decomposition algorithm are given. As seen from Table 5 and 6, the proposed hybrid Benders decomposition algorithm solves 75 instances optimally and in 273.68 seconds on average, except for 9 instances (data numbers 25, 52, 53, 54, 55, 56, 80, 83, and 84). Average gap values of 9 instances that cannot be solved by it optimally are calculated to be 22.72. If we include the results of the other 11 models given in Tables 3 and 4 in the comparison, it can be easily said that the proposed hybrid Benders decomposition algorithm has given the most effective performance among 12 models. We also developed and tested various versions of the Pareto cuts for the proposed hybrid Benders decomposition algorithm. Since the results were much worse in terms of performance, we did not include them in the paper. Besides, since the hybrid Benders decomposition algorithm was deficient to optimally solve problem instances larger than 16 jobs, we also completed the experiments here and we also did not add the results for the big problem instances to the paper.

Although it is not very convenient to compare any exact solution method directly with any approximation algorithm, ultimately, the results obtained by the hybrid Benders decomposition algorithm are compared with the results of a recently published state-of-the-art heuristic for solving this problem, namely the iterated greedy algorithm of Ruiz et al. (2019), in terms of the value of the solutions determined. It is seen that the results obtained for 4 problem instances are better than the best-known solutions given in Ruiz et al. (2019) available at http://soa.iti.es. The mentioned problem instances are I_2_16_5_1, I_3_16_3_1, I_3_16_5_1 and I_4_16_4_1. The previous best-known solutions for problem instances mentioned are 526, 340, 453 and 323, respectively, while the new best-found solutions are 523, 339, 451, and 319, respectively. The corresponding Gantt chart for these instances is presented in Figure 3-6.

Table 3. Summarized comparisons of the mathematical models and ABD.

General Statistics		Model_1	Model_2	Model_3	Model_4	Model_1_ABD
Optimal Solution	**Proven**	56	53	54	46	48
	Avg. Time	86.46	67.69	56.40	109.25	81.24
Common instances being optimally solved by five methods (46 instances)	**Avg. Time**	24.91	6.81	5.91	109.25	49.69
Suboptimal Solution	**Feasible**	28	31	30	38	36
	Avg. Gap%	9.13	27.99	27.36	29.91	10.57
Common instances not being optimally solved by five methods (24 instances)	**Avg. Gap%**	9.48	27.60	25.83	30.60	23.45
Number of a best integer solution		82	67	71	58	61

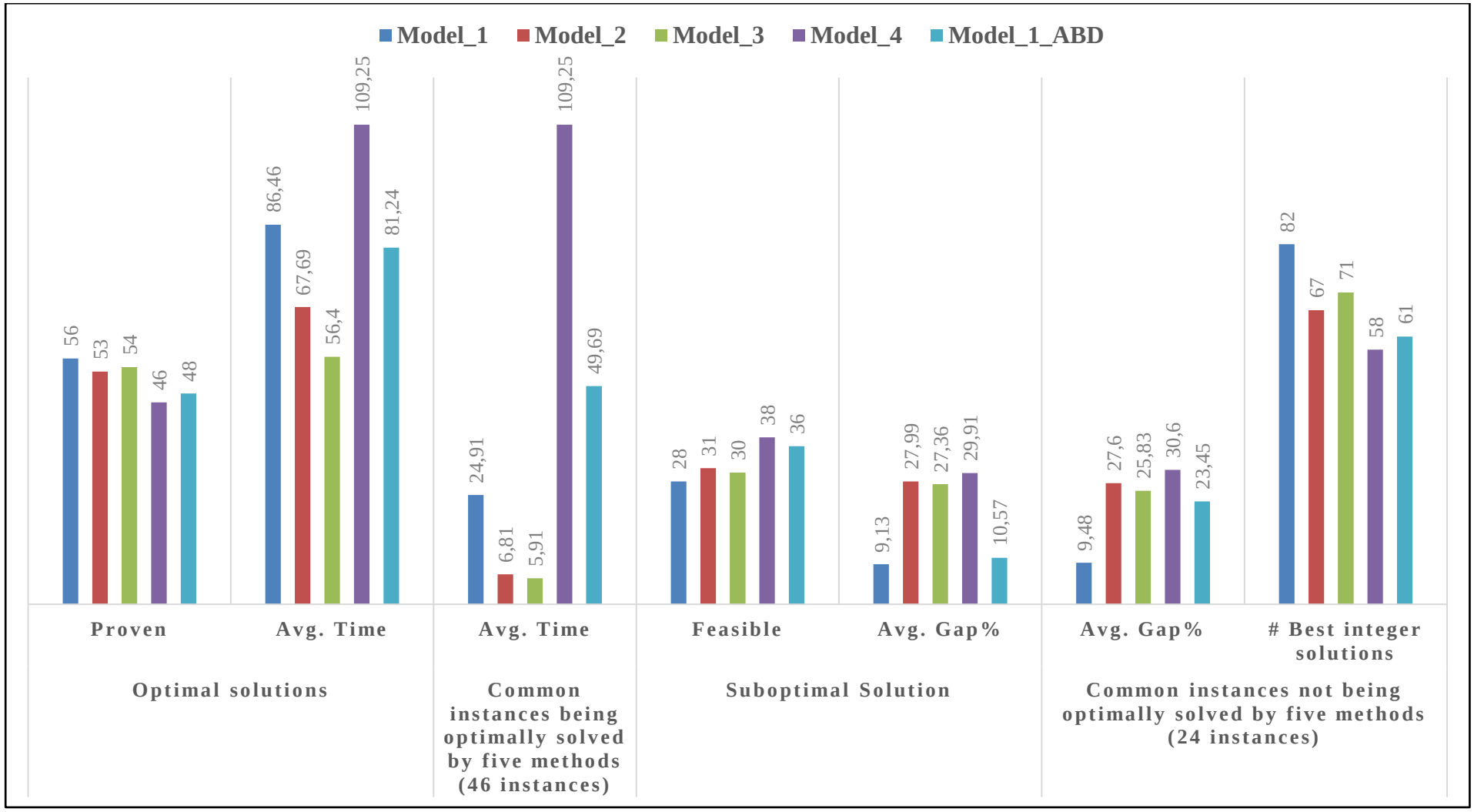

Figure 1. Summarized comparisons of the mathematical models and ABD.

Table 4. Summarized comparisons of the pure Benders decomposition models.

General Statistics		PB_Model1_V1	PB_Model1_V2	PB_Model1_V3	PB_Model1_V4	PB_Model3_V1	PB_Model3_V2
Optimal Solution	**Proven**	54	30	51	30	35	35
	Avg. Time	393.70	134.37	460.52	121.77	41.83	34.41
	Avg. Iteration	71.43	82.73	420.69	119.56	292.02	304.57
Common instances being optimally solved by six methods (29 instances)	**Avg. Time**	35.40	92.68	136.92	77.30	9.94	11.43
	Avg. Iteration	49.96	76.13	224.75	105.62	144.37	163.93
Suboptimal Solution	**Feasible**	30	54	33	54	49	49
	Avg. Gap%	43.25	25.54	40.39	28.66	100	100
	Avg. Iteration	22.18	179.68	383.75	260.62	4676.91	5229.12
	Avg. Lower Bound	258.63	292.96	247.67	278.24	0	0
	Avg. Upper Bound	449.18	407.57	419.39	406.81	404	400.97
Common instances not being optimally solved by six methods (30 instances)	**Avg. Gap%**	39.06	43.25	41.32	45.35	100	100
	Avg. Iteration	22.18	90.59	365.36	104.59	4766.05	5501.90
	Avg. Lower Bound	258.63	277.31	249.81	253.04	0	0
	Avg. Upper Bound	449.18	465.95	430.90	474.86	438.09	435.72
Number of Best Bounds	**Highest Lower Bound**	63	42	55	38	35	35
	Lowest Upper Bound	56	31	58	34	40	41

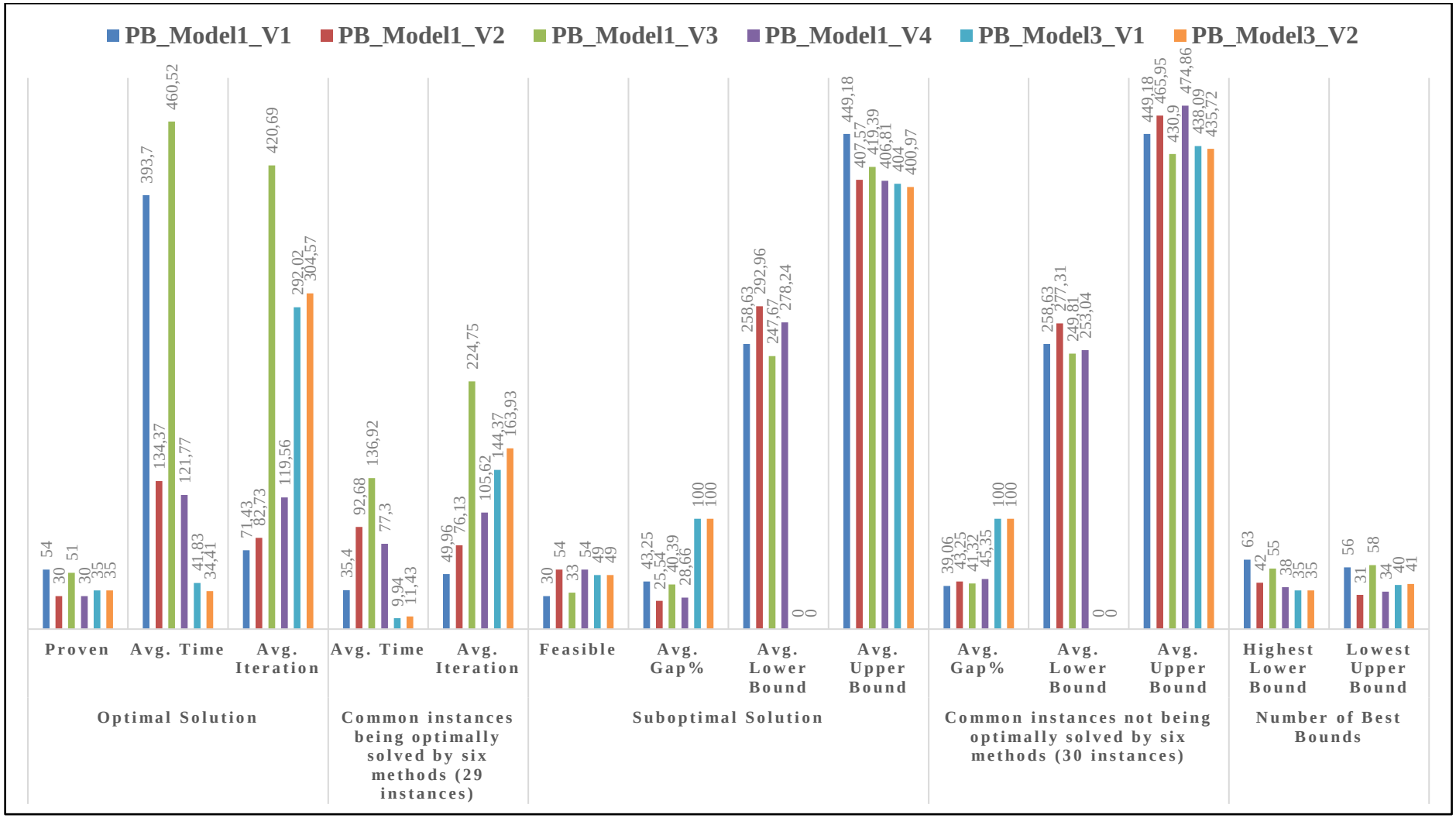

Figure 2. Summarized comparisons of the pure Benders decomposition models.

Table 5. Results of the proposed hybrid Benders decomposition algorithm.

Data No	Number of Machines	Number of Jobs	Number of Factories	Lower Bound	Upper Bound	Gap%	Number of Iterations	Cpu Time
1	2	4	2	112	112	0.00	8	1.60
2	3	4	2	219	219	0.00	9	8.38
3	4	4	2	267	267	0.00	9	1.24
4	5	4	2	337	337	0.00	12	1.62
5	2	6	2	184	184	0.00	10	2.64
6	3	6	2	274	274	0.00	18	5.44
7	4	6	2	323	323	0.00	12	2.58
8	5	6	2	386	386	0.00	26	8.90
9	2	8	2	188	188	0.00	11	6.65
10	3	8	2	341	341	0.00	21	14.29
11	4	8	2	364	364	0.00	27	11.77
12	5	8	2	468	468	0.00	42	28.69
13	2	10	2	345	345	0.00	18	117.55
14	3	10	2	360	360	0.00	35	17.56
15	4	10	2	421	421	0.00	31	5.52
16	5	10	2	452	452	0.00	64	37.75
17	2	12	2	354	354	0.00	20	34.79
18	3	12	2	431	431	0.00	26	30.73
19	4	12	2	423	423	0.00	63	104.72
20	5	12	2	538	538	0.00	87	601.19
21	2	14	2	474	474	0.00	12	38.35
22	3	14	2	514	514	0.00	25	304.37
23	4	14	2	458	458	0.00	66	500.89
24	5	14	2	536	536	0.00	71	752.70
25	2	16	2	507	569	10.89	11	1800
26	3	16	2	489	489	0.00	56	1159.95
27	4	16	2	585	585	0.00	21	812.12
28	5	16	2	523	523	0.00	45	1798.15
29	2	4	3	139	139	0.00	11	0.42
30	3	4	3	197	197	0.00	12	0.36
31	4	4	3	263	263	0.00	11	0.46
32	5	4	3	390	390	0.00	9	0.78
33	2	6	3	161	161	0.00	12	0.51
34	3	6	3	222	222	0.00	14	0.91
35	4	6	3	249	249	0.00	19	1.04
36	5	6	3	351	351	0.00	15	0.93
37	2	8	3	210	210	0.00	18	2.04
38	3	8	3	271	271	0.00	25	3.26
39	4	8	3	343	343	0.00	33	6.08
40	5	8	3	344	344	0.00	28	3.19
41	2	10	3	208	208	0.00	29	23.46
42	3	10	3	270	270	0.00	35	16.08

Table 6. Results of the proposed hybrid Benders decomposition algorithm (continued).

Data No	Number of Machines	Number of Jobs	Number of Factories	Lower Bound	Upper Bound	Gap%	Number of Iterations	Cpu Time
43	4	10	3	331	331	0.00	41	44.64
44	5	10	3	338	338	0.00	37	14.11
45	2	12	3	215	215	0.00	23	149.36
46	3	12	3	336	336	0.00	57	648.67
47	4	12	3	357	357	0.00	44	376.24
48	5	12	3	458	458	0.00	85	1568.31
49	2	14	3	225	225	0.00	16	753.86
50	3	14	3	324	324	0.00	25	1545.62
51	4	14	3	383	383	0.00	41	856.40
52	5	14	3	392	463	15.33	30	1800.02
53	2	16	3	260	392	33.67	11	1800.01
54	3	16	3	339	348	2.58	30	1800.02
55	4	16	3	338	481	29.73	23	1800.02
56	5	16	3	451	462	2.38	20	1800.03
57	2	4	4	164	164	0.00	14	0.87
58	3	4	4	229	229	0.00	12	0.73
59	4	4	4	251	251	0.00	6	0.21
60	5	4	4	248	248	0.00	15	0.56
61	2	6	4	164	164	0.00	15	1.48
62	3	6	4	227	227	0.00	14	0.67
63	4	6	4	262	262	0.00	29	3.23
64	5	6	4	309	309	0.00	18	1.44
65	2	8	4	187	187	0.00	22	4.10
66	3	8	4	213	213	0.00	39	7.50
67	4	8	4	326	326	0.00	31	5.05
68	5	8	4	359	359	0.00	22	4.43
69	2	10	4	155	155	0.00	29	10.34
70	3	10	4	219	219	0.00	33	9.70
71	4	10	4	346	346	0.00	47	15.71
72	5	10	4	327	327	0.00	85	96.19
73	2	12	4	183	183	0.00	23	51.47
74	3	12	4	237	237	0.00	69	801.38
75	4	12	4	290	290	0.00	77	364.45
76	5	12	4	411	411	0.00	110	103.43
77	2	14	4	235	235	0.00	24	962.26
78	3	14	4	303	303	0.00	28	1658.01
79	4	14	4	357	357	0.00	63	1355.33
80	5	14	4	390	399	2.26	57	1800.02
81	2	16	4	295	295	0.00	11	1558.69
82	3	16	4	294	294	0.00	23	1081.95
83	4	16	4	319	328	2.74	25	1800.02
84	5	16	4	304	457	33.48	24	1800.05

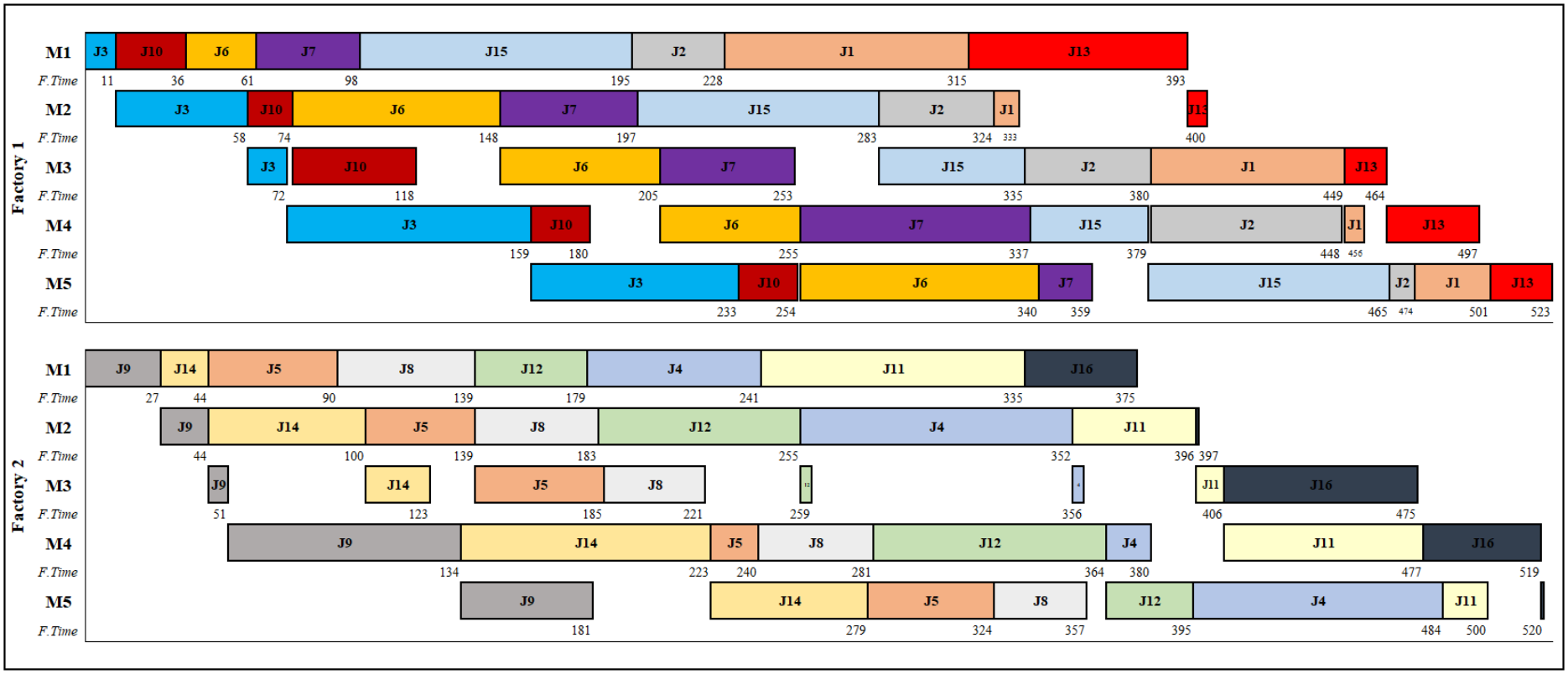

Figure 3. Gantt chart of the new best solution obtained by the Hybrid Benders Algorithm for instance I_2_16_5_1.

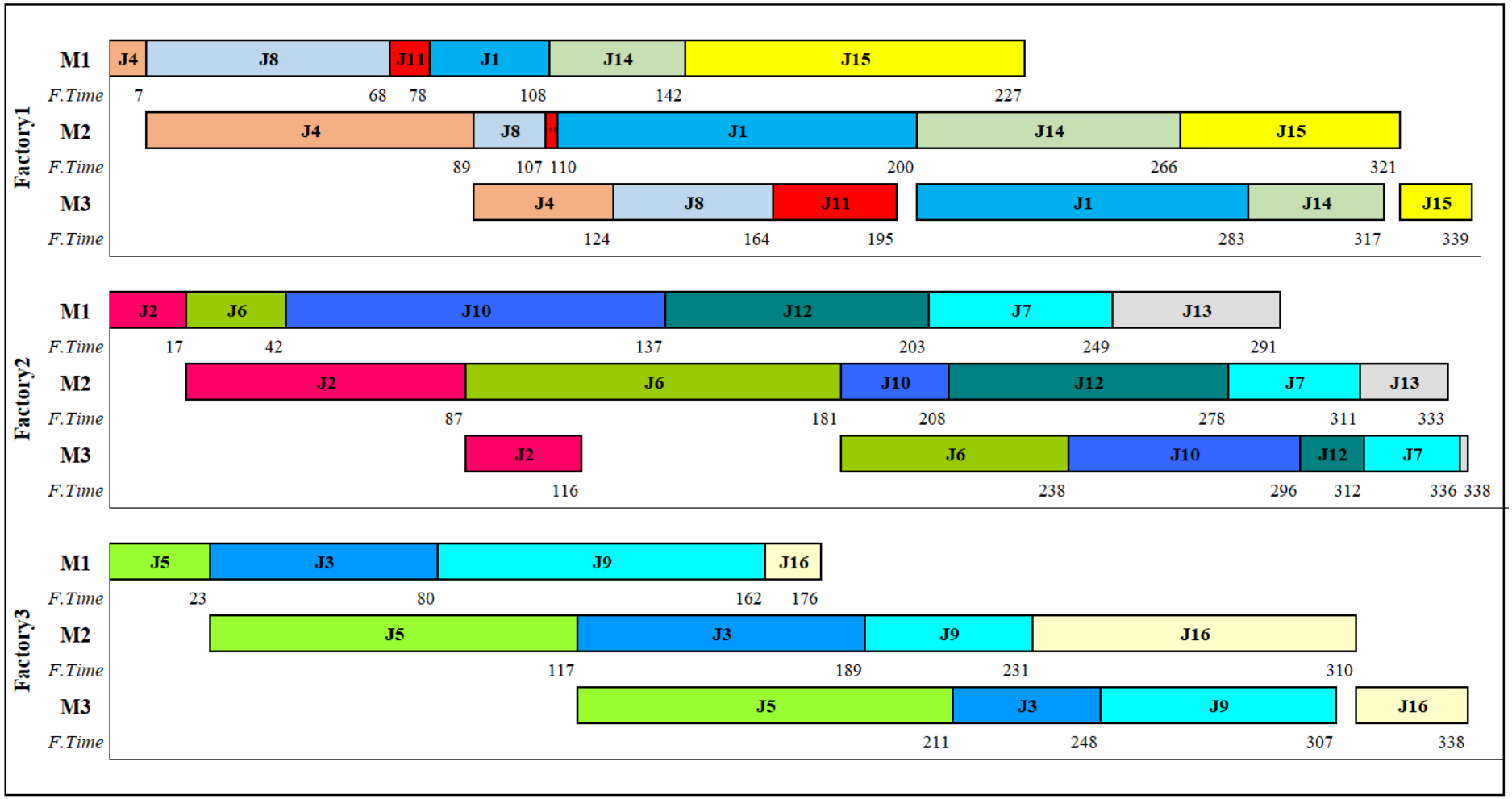

Figure 4. Gantt chart of the new best solution obtained by the Hybrid Benders Algorithm for instance I_3_16_3_1.

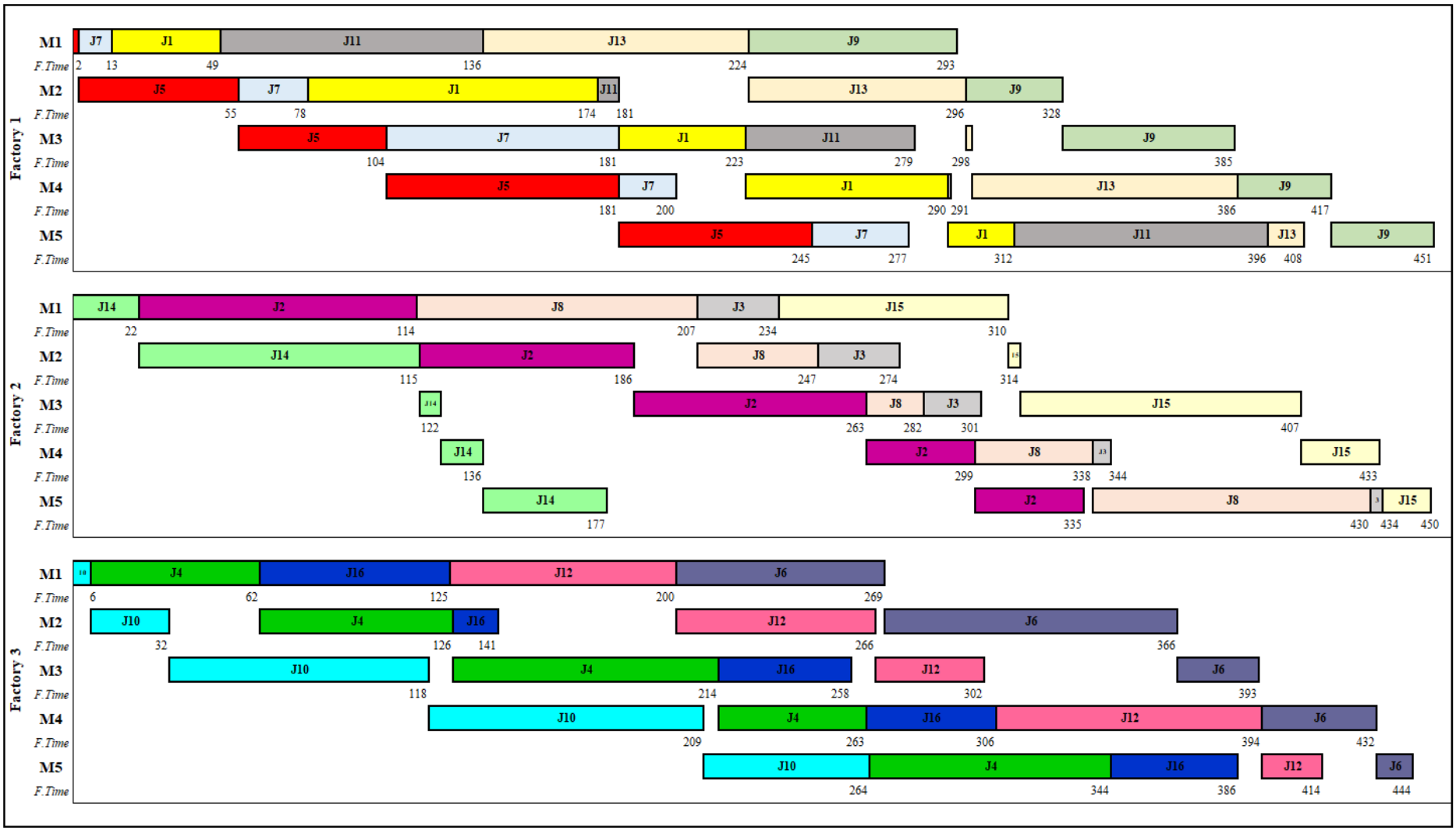

Figure 5. Gantt chart of the new best solution obtained by the Hybrid Benders Algorithm for instance I_3_16_5_1.

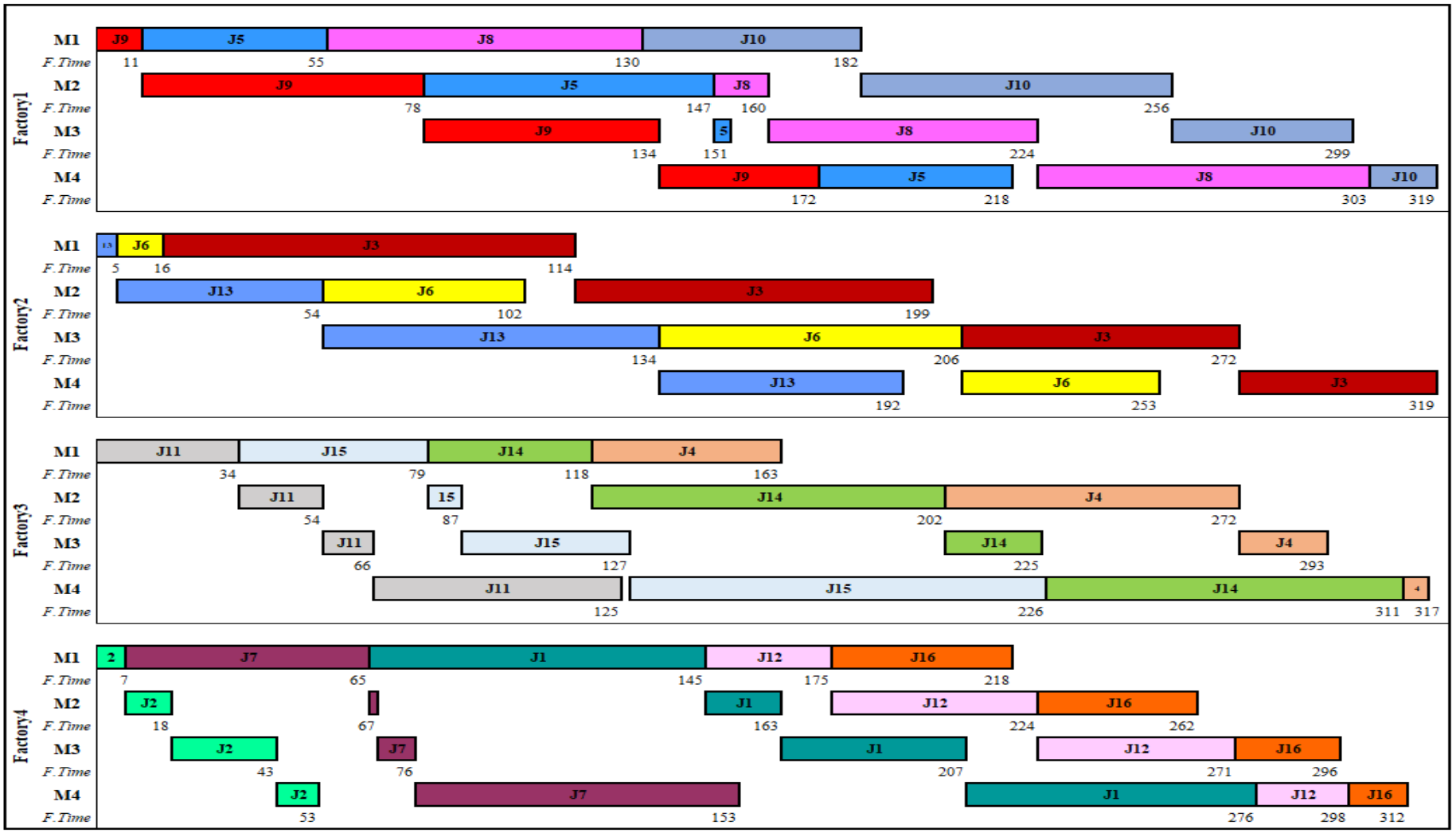

Figure 6. Gantt chart of the new best solution obtained by the Hybrid Benders Algorithm for instance I_4_16_4_1.

6. Conclusion

The distributed permutation flowshop scheduling problem (DPFSP) has in recent times occurred as a generalization of the regular flowshop scheduling problem where several factories are available and accessible for dispensation of the jobs. The DPFSP dealing with real-life applications has attracted the attention of researchers for more or less a decade. The only exact method on this problem that minimizes makespan is that of Naderi and Ruiz (2010), which presents 6 different mathematical models for the solution of the problem. The best performance was given by the position-based distributed permutation flowshop scheduling model and minimal sequence-based distributed permutation flowshop scheduling model among these models. In addition to these two best models, in this paper, 2 new models are developed by inspiring the multiple-traveling salesman problem (mTSP) formulations (Bektas (2006)). The new models mentioned are developed by inspiring mTSP-assignment based integer programming formulation and mTSP-flow-based formulation, respectively. 4 different Benders decomposition algorithms are developed based on the permutation flowshop scheduling model and 2 different Benders decomposition by using the model developed by inspiring mTSP-assignment based integer programming formulation. In addition to these newly developed 8 different exact methods, a hybrid Benders decomposition algorithm is developed by using the permutation flowshop scheduling model and LS3 local search algorithm (Ruiz, Pan, and Naderi (2019)). All of the existing and new exact methods are compared with each other and the automatic Benders decomposition algorithm characteristic available ready-to-use in the CPLEX software by using 84 problem instances. The proposed mTSP-assignment based integer programming formulation based mathematical model has given superior performance in terms of all performance criteria than the minimal sequence-based distributed permutation flowshop scheduling stated to be giving the best results by Naderi and Ruiz (2010). In addition, the hybrid Benders decomposition algorithm developed by hybridizing the permutation flowshop scheduling model and LS3 local search algorithm has outperformed compared to the other 11 models by solving 75 out of 84 problem instances optimally under a time limitation of 1800 seconds. In this paper, 4 new best solutions are also indentified for the DPFSP. The results obtained in this paper encourage the use of such a strategy in solving other variants of the DPFSP, such as non-idle and no-wait DPFSP with or without setup times.

References

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega https://doi.org/10.1016/j.omega.2004.10.004.

Benders, J.F. (1962). Partitioning procedures for solving mixed-variables programming problems, Numerische Math. 4, 238–252.

Chan, F.T.S., Chung, S.H., Chan, L.Y., Finke, G., & Tiwari, M.K. (2006). Solving distributed FMS scheduling problems subject to maintenance: genetic algorithms approach, Robotics and Comput. Integrated Manufac. https://doi.org/10.1016/j.rcim.2005.11.005.

Costa, A.M., Cordeau, J. F., Gendron, B., & Laporte, G. (2012). Accelerating Benders decomposition with heuristic master problem solutions, Pesquisa Operacional http://dx.doi.org/10.1590/S0101-74382012005000005.

Deng, J., & Wang, L. (2017). A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem, Swarm and Evolutionary Comput. https://doi.org/10.1016/j.swevo.2016.06.002.

Fernandez-Viagas, V., & Framinan, J.M. (2015). A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem, Int. J. of Prod. Res. https://doi.org/10.1080/00207543.2014.948578.

Framinan, J.M., Gupta, J.N.D., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective, J. Oper. Res. Soc. https://doi.org/10.1057/palgrave.jors.2601784.

Framinan, J.M., Leisten, R., & Ruiz, R. (2014). Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools. Springer, New York.

Fernandez-Viagas, V., Ruiz, E., & Framinan, J.M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2016.09.055.

Gao, J., & Chen, R. (2011a). A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem, Int. J. Comput. Intel. Syst. 4, 497–508.

Gao, J., & Chen, R. (2011b). An NEH-based Heuristic Algorithm for Distributed Permutation Flowshop Scheduling Problems, Sci. Res. and Essays 6, 3094–3100.

Gao, J., Chen, R., Deng, & W., Liu, Y. (2012). Solving multi-factory flowshop problems with a novel variable neighbourhood descent algorithm. J. Comput. Inf. Syst. 8, 2025–2032.

Gao, J., Chen, R., & Deng, W. (2013). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem, Int. J. Prod. Res. https://doi.org/10.1080/00207543.2011.644819.

Garey, M.R., Johnson, D.S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling, Math. Oper. Res. https://doi.org/10.1287/moor.1.2.117.

Giovanni, L.D., & Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2009.01.008.

Gupta, J.N.D., & Stafford Jr, E. F. (2006). Flowshop scheduling research after five decades, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2005.02.001.

Hejazi, S.R., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review, Int. J. Prod. Res. https://doi.org/10.1080/00207540500056417.

Jia, H.Z., Fuh, J.Y.H., Nee, A.Y.C., & Zhang, Y.F. (2007). Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems, Comput. Indust. Eng. https://doi.org/10.1016/j.cie.2007.06.024.

Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included, Naval Res. Logistics Quarterly, https://doi.org/10.1002/nav.3800010110.

Liu, H., & Gao, L. (2010). A Discrete Electromagnetism-like Mechanism Algorithm for Solving Distributed Permutation Flowshop Scheduling Problem. International Conference on Manufacturing Automation https://ieeexplore.ieee.org/document/5695172.

Lin, S.W., Ying, K.C., & Huang, C.Y. (2013). Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm, Int. J. Prod. Res. https://doi.org/10.1080/00207543.2013.790571.

McKay, K.N., Pinedo, M., & Webster, S. (2002). Practice-focused research issues for scheduling systems, Prod. Oper. Manage. https://doi.org/10.1111/j.1937-5956.2002.tb00494.x.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem, Comput. Oper. Res. https://doi.org/10.1016/j.cor.2009.06.019.

Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2014.05.024.

Nawaz, M., Enscore, E.E., & Ham, J.I. (1983). A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem, Omega https://doi.org/10.1016/0305-0483(83)90088-9.

Onwubolu, G., & Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2004.08.043

Pinedo, M. (2016). Scheduling: Theory, Algorithms and Systems. Springer, New York.

Rahmaniani, R., Crainic, T.G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2016.12.005.

Reisman, A., Kumar, A., & Motwani, J. (1997). Flowshop scheduling/sequencing research: a statistical review of the literature 1952-1994. IEEE Trans. Eng. Manage. 44, 316-329.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics, Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2004.04.017.

Ruiz, R., Pan, Q.K., & Naderi, B. (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem, Omega https://doi.org/10.1016/j.omega.2018.03.004.

Sherali, H.D., & Fraticelli, B.M.P. (1962). A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse, J. Global Optimization. 22, 319-342.

Wang, S.Y., Wang, L., Liu, M., & Xu, Y. (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem, Int. J. Prod. Eco. https://doi.org/10.1016/j.ijpe.2013.05.004.

Xu, Y., Wang, L., Wang, S., & Liu, M. (2014). An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem, Eng. Optimization https://doi.org/10.1080/0305215X.2013.827673.

Ying, K.C., Lin, S.W., Cheng, C.Y., & He, C.D. (2017). Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems, Comput. Indust. Eng. https://doi.org/10.1016/j.cie.2017.06.025.