Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

Adaptive Large Neighborhood Search Heuristic for Mixed Blocking Flowshop Scheduling Problem

Damla Kizilay, Zeynel Abidin Çil*

Izmir Democracy University, Department of Industrial Engineering, Izmir, Turkey
*damla.kizilay@idu.edu.tr, *zabidin.cil@idu.edu.tr*
Received date: 07.04.2021, Accepted date: 06.05.2021

Abstract

Traditional permutation flowshop scheduling problem (PFSP), which has unlimited buffer space, has been interested over the fifty years by several authors to account for many industrial applications. However, some industries, such as the aerospace industry and other sectors processing industrial waste, have different blocking conditions due to the limited or lack of buffer area between their machines. In this study, a mixture of different blocking types is considered to solve PFSP with the total flow time criterion regarding several blocking types. A constraint programming model is proposed to solve the PFSP with mixed blocking constraints (MBFSP). Due to the problem's NP-hard nature of the problem, an adaptive large neighborhood search heuristic is proposed to solve the large size instances. The results of the proposed algorithm are very competitive.

Keywords: Adaptive large neighborhood search, constraint programming, flowshop scheduling

Karışık Blokalmalı Akış Tipi Çizelgeleme Problemi İçin Uyarlanabilir Büyük Komşuluk Arama Sezgiseli

Öz

Sınırsız tampon alanına sahip geleneksel permütasyon akış tipi çizelgeleme problemi (PFSP), pek çok endüstriyel uygulaması olduğundan dolayı yaklaşık elli yıldır yazarlar tarafından araştırılan bir konudur. Bununla birlikte, havacılık endüstrisi ve endüstriyel atıkları işleyen bazı endüstriler, makineleri arasında sınırlı veya eksik tampon alanı olması nedeniyle farklı bloklama koşullarına sahiptir. Bu çalışmada, farklı bloklama türlerine sahip olan bir PFSP'yi toplam akış süresini dikkate alarak çözecektir. PFSP'yi karışık bloklama kısıtları (MBFSP) ile çözmek için bir kısıt programlama modeli önerilmiştir. Problemin NP-zor doğası nedeniyle, büyük boyutlu problemleri çözmek için uyarlanabilir bir büyük komşuluk arama sezgisel yöntemi önerilmektedir. Önerilen algoritmanın sonuçları çok rekabetçidir.

Anahtar Kelimeler: Akış tipi çizelgeleme, kısıt programlama, uyarlanabilir büyük komşuluk araması

INTRODUCTION

Traditional permutation flowshop scheduling problem (PFSP) was first proposed by Johnson (1954) and then has been interested in several authors (Blazewicz et al., 2007; Pan & Ruiz, 2012; Ruiz-Torres et al., 2011; Vallada & Ruiz, 2010). In PFSP, a set of jobs is processed through a set of machines following the same order and have unlimited buffer space. Then, a classical blocking constraint was added to PFSP by T. Sawik (1995) and T. J. Sawik (1993). Then, the problem turns to be a blocking permutation flowshop scheduling problem (BFSP) (Jozef Grabowski & Pempera, 2000; N. G. Hall & Sriskandarajah, 1996; Kizilay et al., 2018; Tasgetiren et al., 2015). In BFSP, a buffer area between the machines does not exist; therefore, a job must stay on the machine until the upstream machines become available to process it. Thus, the currently used machine is blocked by the awaiting task and cannot process the next job until the current job leaves the machine. This classical blocking variant is denoted as RSb (release when start blocking). BFSP is proven to be NP-Hard if the number of machines is greater than two (N. G. Hall & Sriskandarajah, 1996). Therefore, several heuristics and metaheuristic algorithms are

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

employed to solve BFSP, such as a genetic algorithm (GA) (Caraffa et al., 2001), tabu search (TS) algorithm (Józef Grabowski & Pempera, 2007), iterated greedy (IG) algorithm (Tasgetiren et al., 2017), discrete differential evolution (DDE) algorithm (Zhang et al., 2018) hybrid DDE algorithm (HDDE) (Qian et al., 2009), particle swarm optimization (PSO). Besides those heuristic algorithms, many constructive heuristics and local search procedures are also proposed for BFSP. Those algorithms include profile fitting (PF) heuristic (Mccormick et al., 1989), improvement heuristic (N. Hall & Sriskandarajah, 2000), and Nawaz, Enscore, Ham (NEH) (Nawaz et al., 1983) -based heuristics (Nawaz et al., 1983; Newton et al., 2019; Riahi et al., 2019; D P Ronconi & Armentano, 2001).

A new blocking constraint denoted as RCb, which means release when completing blocking, was introduced by Martinez et al. (2006). This type of blocking constraint states that a job blocks a machine until its operation on the next machine completes and the job leaves the next machine. In other words, a job occupies two consecutive machines at the same time. This kind of blocking constraint occurs while manufacturing the metallic parts for the aerospace industry and processing the industrial waste. Another new blocking constraint, RCb* (release when complete blocking), which is a variant of RCb, was proposed by Wajdi Trabelsi et al. (2010). It is similar to the RCb constraint, but this time a job blocks a machine until its operation on the next machine completes, regardless of releasing the next machine.

In this study, regarding the aforementioned blocking types, a mixture of them is considered to solve PFSP with the total flowtime criterion. The considered problem is denoted as $F_m/\ mixed\ blocking/\sum F$ according to Graham et al. (1979). The PFSP with mixed blocking constraints (MBFSP) was first considered by W Trabelsi et al. (2011), and its linear mathematical model with complexity analysis is provided. The model is tested on instances that have up to 12 jobs and 100 machines, and the authors conclude that heuristics or metaheuristics should be developed for larger-size problems. Then, the same authors employed the NEH heuristic and proposed Trabelsi, Sauvey, and Sauer (TSS) heuristic as well as GA metaheuristic to solve MBFSP with up to 12 jobs and 100 machines (Wajdi Trabelsi et al., 2012). A constructive heuristic to minimize the makespan in MBFSP was proposed by Khorramizadeh & Riahi (2015), and the algorithm was tested on the well-known instances of Taillard's (Taillard, 1993) that includes large-size problems up to 200 jobs and 20 machines. In both studies, the authors used the same repetitive sequence of mixed blocking types, which is (RCb, RSb, RCb*, Wb), for the successive machines. However, in 2017, the authors in Riahi et al. (2017) stated the following drawback of this blocking sequence: The RSb constraint applied to a machine, which is placed immediately after another machine running under the RCb constraint, cannot make any difference in the makespan. Therefore, the authors used a new blocking sequence generated with equal probability for each machine. In the generation procedure, they paid attention not to have RSb type of machine immediately after RCb type of machine, and they stated that the last machine always has Wb type. The authors applied a scatter search for MBFSP to minimize the makespan. Besides Taillard's instances, they used VRF large benchmark instances proposed by Vallada et al. (2015). A local search based on the characteristics of different types of blockings was proposed to solve MBFSP with the makespan criterion (Riahi et al., 2019). Most recently, a multi-temperature simulated annealing algorithm is proposed to solve the MBFSP with the makespan criterion (Lin et al., 2021). For the total completion time criterion, MBFSP is considered by Cheng et al. (2020). The authors provide results for Taillard's instances.

Several heuristic algorithms were applied to solve the MBFSP, such as GA, SA, NEH-based heuristics, and constructive heuristics. However, adaptive neighborhood-based search procedures have not been applied to MBFSP before. Neighborhood search-based methods have great potential as competitive approaches for real-life scale problems. Therefore, motivated by the challenge of computational time and the problem size, an adaptive large neighborhood search (ALNS) heuristic is proposed to solve the MBFSP by minimizing the total flowtime criterion.

The main contributions in this paper are listed as follows.

- A constraint programming (CP) model to solve the MBFSP problem, to the best of the authors' knowledge, for the first time in literature.
- ALNS heuristic with four different remove and two different insert strategies as well as two different

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

swap operations are employed. Based on an adaptive structure, the best strategy for each instance is determined by the algorithm to find the best solution.

- To the best of the authors' knowledge, solutions for the small-size VRF benchmark instance sets (Vallada et al., 2015) are proposed for the first time in literature.

The remainder of the paper is organized as follows. The problem definition and the formulation of the CP model for the MBFSP are presented in Section 2. The proposed adaptive large neighborhood search heuristic is described in Section 3. Section 4 presents the results of the computational experiments for all presented models and algorithms. Finally, Section 5 provides conclusions and future work.

PROBLEM DEFINITION AND FORMULATION

Formulation of MBFSP has a set of jobs that are processed through a set of machines following the same order. Both machines and jobs are available at time zero, and each machine can handle one job at a time. Similarly, each job can be processed by at most one machine at a time. In MBFSP, there are several blocking types, and each machine can have different blocking constraints. Those types can be classified under four definitions: Without blocking (Wb), release when starting blocking (RSb), and two types of release when completing blocking (RCb, RCb*).

In the formulation of MBFSP, different types of blocking constraints of the machines are regarded to minimize the total flow time of the jobs. In the light of this objective, a constraint programming (CP) model is developed. The sets and the parameters used by the model are presented in Table 1.

The model has a set of jobs and machines, as well as processing times of each job on each machine. Those sets and parameters are standard in all flowshop problems. Different than the other problems, MBFSP has a blocking type for each machine represented by b_{lk} parameter. In b_{lk}, the index l takes values like 0, 1, 2, and 3, and those integer numbers represent Wb, RSb, RCb*, and RCb constraints, respectively. For example, $b_{2,1} = 1$ means that machine 1 applies a blocking type of RCb*. The following sub-section explains the CP model.

Constraint Programming Model

The CP, which is a declarative method, depends on the constraints. It uses constraints to infer new constraints in order to solve a problem. Rather than the solution methodology, the problem to be solved is more important. It has several advantages, such as having a compact model compared to MILP and adaptable to meet new requirements compared to typical procedural programs. Also, using the logical rules and the constraints, it becomes easier to prove the correctness of the models. The expanded search space and the advanced relaxation techniques of the CP approaches make it promising, but the applications of the CP are limited. Although, recent works about scheduling in real scenarios are still concentrating on MILP as a tool of choice (Fuchigami & Rangel, 2018), even if CP seems better suited for scheduling problems, especially for large-scale instances (Ku & Beck, 2016).

Table 1 *Problem Notation*

	Sets
M	Set of machines $\{1, 2, \dots, m\}$
N	Set of jobs $\{1, 2, \dots, n\}$
	Parameters
p_{ik}	Processing time of job i on machine k
b_{lk}	$\begin{cases} 1, & \text{if blocking type } l \text{ is applied at machine } k \\ 0, & \text{otherwise} \end{cases}$

Table 2 CP Decision Variables

x_{ik}^{CP}	Interval variable for job i, processed by machine k with duration p_{ik}
z_i	Interval variable for job i
seq_k	Sequence variable for machine k, which is defined over a set of interval variables x_{ik}^{CP}

CP model is developed for MBFSP for the first time in literature. The decision variables include interval and sequence variables and are defined in Table 2. The CP model has two sets of interval variables, and both have a start, a duration, and an end. To define the domain of each job and to represent the process of each job on each machine, two different interval variables are introduced. Since each machine processes all the jobs, an interval variable of a job on any machine is not an optional variable. In other words, the interval variable always presents in the solution. Also, sequence variables are defined for each machine to represent the set of interval variables.

Therefore, the jobs are assigned to the machines respecting their processing times in the definition

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

step. The objective function and the constraints of the CP model are given and explained as follows.

Objective Function

$$Minimize\ \sum_{i\in N} end\left(x_{im}^{CP}\right) \quad (1)$$

Constraints

$$endBeforeStart\left(x_{i,k-1}^{CP}, x_{ik}^{CP}\right) \quad \forall i \in N, k \in M | k > 1 \quad (2)$$

$$\left(endOfPre\left(seq_k, x_{i,k}^{CP}\right)b_{1k}\right) + \left(startOfPre\left(seq_{k+1}, x_{i,k+1}^{CP}\right)b_{2k}\right) + \left(endOfPre\left(seq_{k+1}, x_{i,k+1}^{CP}\right)b_{3k}\right) + \left(startOfPre\left(seq_{k+2}, x_{i,k+2}^{CP}\right)b_{4k}\right) \leq startOf\left(x_{i,k}^{CP}\right) \quad \forall i \in N, k \in M | k < m-1 \quad (3)$$

$$\left(endOfPre\left(seq_{m-1}, x_{i,m-1}^{CP}\right)b_{1,m-1}\right) + \left(startOfPre\left(seq_m, x_{im}^{CP}\right)b_{2,m-1}\right) + \left(endOfPre\left(seq_m, x_{im}^{CP}\right)b_{3,m-1}\right) \leq startOf\left(x_{i,m-1}^{CP}\right) \quad \forall i \in N \quad (4)$$

$$noOverlap(s_k) \quad \forall k \in M \quad (5)$$

$$SameSequence(s_1, s_k) \quad \forall k \in M | k > 1 \quad (6)$$

In the objective function (1), the CP model minimizes total completion time by calculating the total end time of the jobs on the last machine m. Constraint set (2) ensures that each job is processed through the machines respectively. First, their process should be completed on the first machine, and then the job can proceed to the next machine. Constraint sets (3) indicate the RSb, RCb*, and RCb type of blocking constraints on the corresponding machines. The first part of the constraint set is employed when the machine does not have any blocking constraints. Therefore, when the operation of the previous job is completed, the next job can start its operation. If the machine works as RSb, then the second part of Constraint set (3) ensures that a job cannot start its process unless the preceding job starts its operation on the successive machine. The third part of the same constraint provides that a job can start its operation on the machine after its preceding job completes its operation on the succeeding machine. This situation is valid for the machines that work with RCb* type. The last part of the constraint set is similar to the previous part, but it ensures that the preceding job leaves the succeeding machine. Therefore, in order to provide RCb type of constraint, the start of a job interval on machine k is related to the start of the preceding job interval on machine $k+2$. Constraint set (4) is warren for the last machine to ensure the different blocking constraints on the last machine. Constraint set (5) states that the job interval variables on each machine should not be overlapped. In other words, each machine can process one job at a time. Finally, Constraint set (6) provides that the processing order of the jobs on each machine must be the same.

ADAPTIVE HEURISTIC APPROACH

The ALNS heuristic is proposed as an extension of the Large Neighborhood Search (LNS) heuristic, which is first suggested in Shaw (1998). The ALNS heuristic uses the same logic with LNS but employs the remove and insertion methods adaptively. The removal and insertion methods are chosen according to the algorithm's performance during the same search. This study employs the ALNS heuristic, developed by Pisinger & Ropke (2007) for the vehicle routing problem. The ALNS heuristic is modified and applied to the scheduling problem in the thesis of Kizilay (2018).

Initialization

ALNS heuristic is a population-based algorithm and includes several neighboring strategies in an adaptive manner. In this study, the initial sequence of the jobs is generated randomly for each individual in the population, except the four of them. Those four sequences are generated according to the following rules. For all the jobs:

- R1: Increasing order of total process times on all the machines.
- R2: Decreasing order of total process times on all the machines.
- R3: Increasing order of total process times on the machines with RCb or RCb* blocking types plus the process times on the succeeding machine.
- R4: Decreasing order of total process times on the machines with RCb or RCb* blocking types plus the process times on the succeeding machine.

The first rule is based on the shortest processing time (SPT), while the second rule is based on the longest processing time (LPT). The problem-related properties inspire the remaining rules. Once the initial sequence is generated, the NEH heuristic is applied to the individual as a constructive heuristic to start with a good solution, then ALNS is applied to this individual. In our problem, NEH is applied to an individual after generating the initial sequence.

During ALNS, several removals, insertion, and swap methods are employed in the same search. ALNS is applied many times to an individual to determine the weights of the removal, insert and swap

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

move types and thus provides an adaptive search scheme. If the new solution obtained by ALNS is better than the current solution, it replaces the current solution, and the weights of the remove-insert-swap methods are updated accordingly. Otherwise, an SA-based acceptance criterion is employed to accept the new solution to escape local minima. The following procedure presents the general solution scheme.

General Solution Procedure

for $i=1$ to NP (number of population)
 π_i: Sequence created respecting $R_i|i=\{1,2,3,4\}$
 π_i: Random sequence of population $i|i>4$
 $\pi_i = NEH(\pi_i)$
 while not Termination Criterion
  $\pi_i = ALNS(\pi_i)$
  if $(f(\pi_i) < f(\pi_{best}))$ update π_{best}
 end while
end for

Neighborhood Strategies

The neighborhood strategies include three different removal, four different insertions, and two different swap strategies. ALNS chooses these strategies in an iteration respecting their weights.

First, the algorithm selects a swap or removal method applying a roulette wheel selection. If ALNS chooses one of the swap moves concerning their weights, applies it to the current individual, and obtains the new solution. On the other hand, if ALNS selects a removal strategy, it disrupts the current solution by removing some jobs. Hence, ALNS has to choose one of the four insertion strategies to construct a new solution. The insertion strategy selection depends on the applied removal method and a roulette wheel selection considering insertion weights.

Random removal

The random removal algorithm randomly selects q jobs from the current sequence π and removes these jobs. The predetermined removal size of the ALNS specifies the number of selected jobs (q).

Random block removal

Random block removal selects q consecutive jobs (a block of jobs) randomly and removes the chosen block from the current sequence π. This procedure is inspired by the block insertion heuristic (BIH) in Tasgetiren et al. (2016).

Blocking-idle time removal

The blocking-idle time removal algorithm chooses q jobs that cause the most considerable blocking or idle time on the machines and removes them from the current sequence π.

One of the removal strategies is applied, and the removed jobs are collected in a sequence $\pi_{removed}$, and the remaining partial sequence is represented by π_{partial}. After removing jobs from the current sequence, The ALNS applies one of the insertion strategies to insert the jobs included in $\pi_{removed}$ into the partial sequence π_{partial}. If ALNS uses a random block removal, then it should apply random block insertion or best block insertion regarding the weights. If ALNS employs other removals, then it should apply random or best insertion.

Random insertion

Random insertion algorithm randomly selects q positions in partial sequence π_{partial} and inserts the removed jobs collected in $\pi_{removed}$ to the selected positions, one by one.

Best insertion

Best insertion chooses the jobs one by one from the set of $\pi_{removed}$. It searches all possible positions of partial sequence π_{partial} to insert the chosen job into the best position in terms of the objective function. The algorithm applies the same procedure until all the jobs are selected from the removed sequence and inserted into the partial sequence.

Random block insertion

Random block insertion randomly chooses a location in partial sequence π_{partial} and inserts removed a block of jobs to the chosen location.

Best Block Insertion

The best block insertion takes all removed jobs from $\pi_{removed}$ as a block and searches all possible positions at partial sequence π_{partial} to insert the block.

Swap move

In swap move, two jobs are selected randomly from sequence π, and their positions are exchanged.

Iterative Swap Move

Iterative swap move randomly chooses and exchanges two jobs in the sequence π, q times, which is equal to the remove size.

Acceptance Criterion and the Adaptive Weight Adjustment Procedure

In the acceptance criterion step, if a new sequence has a smaller objective function value than the current sequence, the new is accepted as the current sequence. If the new sequence is worse than the current sequence, then a simple SA type of acceptance criterion is used with constant

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

temperature (Osman & Potts, 1989) to determine if the new worse sequence is accepted or not. The following algorithm shows the general procedure of the ALNS heuristic with the SA acceptance criterion.

ALNS Procedure

θ_j: how many times the strategy j is selected: j={r,s,i}
w_j: weight of the strategy j in the roulette wheel.
while not Termination Criterion
 Choose a removal r or swap s strategy by roulette wheel selection
 If removal strategy r
  π_{partial} = Apply chosen removal strategy r to current sequence π
  $\theta_r = \theta_r + 1$
  Choose an insertion strategy i by roulette wheel selection w.r.t. applied removal (block or not)
  π_{new} = Apply the insertion strategy i to partial sequence π_{partial}
  $\theta_i = \theta_i + 1$
 Else if swap strategy s
  π_{new} = Apply chosen swap strategy s to current sequence π
  $\theta_s = \theta_s + 1$
 end if
 If $(f(\pi_{new}) < f(\pi))$
  $\pi = \pi_{new}$
  If $(f(\pi_{new}) < f(\pi_{best}))$
   $\pi_{best} = \pi_{new}$
   $\varepsilon_j = \varepsilon_j + \sigma_1$
  Else
   $\varepsilon_j = \varepsilon_j + \sigma_2$
 end if
 end if
 $\rho = e^{-(f(\pi_{new})-f(\pi))/T}$
 Generate a random number $a \in [0.1]$
 If $(a < \rho)$
  $\pi = \pi_{new}$
  $\varepsilon_j = \varepsilon_j + \sigma_3$
 end if
 Update Procedure
 $w_j = w_j\,(1 - rate) + (rate)\varepsilon_j/\theta_j$
end while

In the procedure, π, π_{new}, and π_{best} represent the current, new, and the best schedule at each iteration, where $f(\pi)$, $f(\pi_{new})$, and $f(\pi_{best})$ denotes their total flow time values, respectively. The algorithm always accepts the new solution as the incumbent if $f(\pi_{new}) < f(\pi)$, and always accepts the new solution as the global best if $f(\pi_{new}) < f(\pi_{best})$. It also accepts the new solution as the incumbent solution with probability $e^{-(f(\pi_{new})-f(\pi))/T}$ to provide diversification by giving a chance to the worse schedules.

The ALNS heuristic updates the weights of the removal, insertion, and swap methods in each iteration by considering the weights of the previous iteration and the current score information, as shown in the following equation: $w_j = w_j(1-rate) + \varepsilon_j/\theta_j(rate)$. In this equation, w_j represents the weight of applied strategy j at each iteration, and $rate$ represents the roulette wheel rate of using the previous weight and the current score information, where $0 \leq rate \leq 1$. The parameter ε_j represents the score of strategy j. The score is increased by σ_1 if the used strategy finds the new global best solution, by σ_2 if it finds the new current best solution, or by σ_3 if the solution is worse than the current but accepted. The parameter θ_j represents the number of times that strategy j is used. Parameter θ_j gives a chance to the other algorithms which are not used before.

RESULTS AND DISCUSSION

The performance of the ALNS algorithm is compared with the mixed-integer linear programming (MILP) and CP models. The MILP model for MBFSP is presented in Riahi et al. (2019). We used a similar model but eliminated a decision variable representing the start time of the jobs on the machines to simplify the model. Therefore, our proposed MILP model employs the same calculations by only considering the completion time of the jobs on the machines.

The MILP and CP models for the MBFSP were coded in OPL and run on the IBM ILOG CPLEX 12.10 software suite, while the ALNS algorithm was coded in C++ and run on the Eclipse IDE 4.15.0. All results were obtained on an Intel Core i7 with 8 GB RAM. The benchmark suite of the VRF data set (Vallada et al., 2015) is used to measure the performance of the MILP, CP models, and ALNS heuristic. Blocking sequences of the machines are generated by Riahi et al. (2017) for only the VRF large instance sets. The authors give each blocking type an equal probability for each machine but avoid scenarios where one machine running under RSb follows another machine running under RCb. According to the authors, this avoided scenario cannot make any difference in the makespan (Riahi et

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

al., 2017). When RCb is applied to a machine k, the RSb condition is always satisfied by machine $k+1$, automatically. Therefore, this scenario also does not make any difference for the total flowtime criterion. Moreover, because the last machine does not have any following machine, the last machine always operates under Wb constraint. Respecting all those rules, we generated the blocking types of each machine for the VRF-small instances.s

The VRF instances have 240 small instances with variable $n\,x\,m$ combinations. There are $n \in \{20,30,40,50,60\}$ jobs each with $m \in \{5,10,15,20\}$ machines. Each $n\,x\,m$ combination has ten different instances. MILP and CP models are employed for only small-sized instances due to the computational complexity of the problem. They were given 3600 seconds time limit.

The VRF large instances also have 240 test cases with variable $n\,x\,m$ combinations. There are $n \in \{100,200,300,400,500,600,700,800\}$ jobs each with $m \in \{20,40,60\}$ machines. Each $n\,x\,m$ combination has ten different instances. In order to obtain the solutions for the large data sets, the developed ALNS heuristic is employed, and its run time is limited to $\tau \times n \times m$ milliseconds for each instance. In order to see the effects of the time limit (τ) and the remove size (q), we obtained all the results for four different variants of ALNS:

- $ALNS_1$: ALNS with constant remove size, where $q = 3$, and $\tau = 30$
- $ALNS_2$: ALNS with variable remove size, where $q = n * 0.10$, and $\tau = 30$
- $ALNS_3$: ALNS with constant remove size, where $q = 3$, and $\tau = 60$
- $ALNS_4$: ALNS with variable remove size, where $q = n * 0.10$, and $\tau = 60$

All the ALNS algorithms applied five replications for each instance. The rest of the parameters for the algorithm is given in Table 3.

This study presents all the results of the models and the algorithms by calculating their relative percentage deviations (RPD): $RPD = 100 * (F_{some} - F_{best})/F_{best}$, where F_{some} is the obtained objective value of the models or the algorithms and F_{best} is the minimum objective among all the models and the algorithms.

The computational results section is divided into two parts. In the first part, the proposed two versions of the ALNS algorithms: $ALNS_1$ and $ALNS_2$, the CP approach, and the MILP, are compared for the small-size problems. Second, the results of four versions of the ALNS method are discussed in terms of solution quality for large-size problems.

Table 4 demonstrates the summary result of all the obtained solutions of the CP, MILP models, and ALNS algorithms. MILP can reach an optimal solution faster than the CP model for small-size problems. The MILP model provides 82 optimal solutions, while the CP, $ALNS_1$, and $ALNS_2$ ensure 14, 72, and 71 optimal solutions, respectively, out of 240 solutions. However, the ALNS algorithms outperform the CP and MILP models in terms of the number of best solutions, average relative percentage deviation (ARPD), average objective value, and computational time. The heuristic algorithms find nearly twice the number of best results than the MILP and the CP model for the small-size problems.

Furthermore, the ARPDs of the heuristic algorithms are approximately 0.1% for the problem. The detail of the RPD result of each technique is given in Figure 1. The RPD values of the CP and MILP model become higher when the problem size increases, but the proposed heuristic algorithms are not affected. In other words, it is clearly seen that the heuristic algorithms show respectable performance for small-size problems.

Table 5 summarizes the results of ALNS algorithms for the large-size problems. The four versions of the ALNS algorithms are reported in this table. According to Table 5, ALNS-4 dominates other versions in terms of # of best, ARPD, Max RPD, and average objective values for all problems. Furthermore, it is clearly seen that constant removal size $(q = 3)$ indicates better performance. Also, increasing the time limit helps to find a better solution.

Table 3 Parameters of the ALNS algorithms

P_{size}=20	The population size of the algorithms
rate=0.2	Roulette wheel rate
σ_1=10	Global best solution score
σ_2=7	Current best solution score
σ_3=3	SA type of acceptance score
T=100	Temperature value of SA

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

Table 4. *Comparisons of the algorithms for the small-size problem*

Methods	*# of Optimal*	*# of Best*	*ARPD*	*Avg. Obj.*	*Avg. CPU (s)*	*Max. CPU (s)*
CP	14	48	1.4	92979.4	3515	3600
MILP	82	95	0.7	92290.6	2516	3600
$ALNS_1$	71	162	0.1	91244.1	$\frac{30nm}{1000}$	$\frac{30nm}{1000}$
$ALNS_2$	70	153	0.1	91240.5	$\frac{30nm}{1000}$	$\frac{30nm}{1000}$

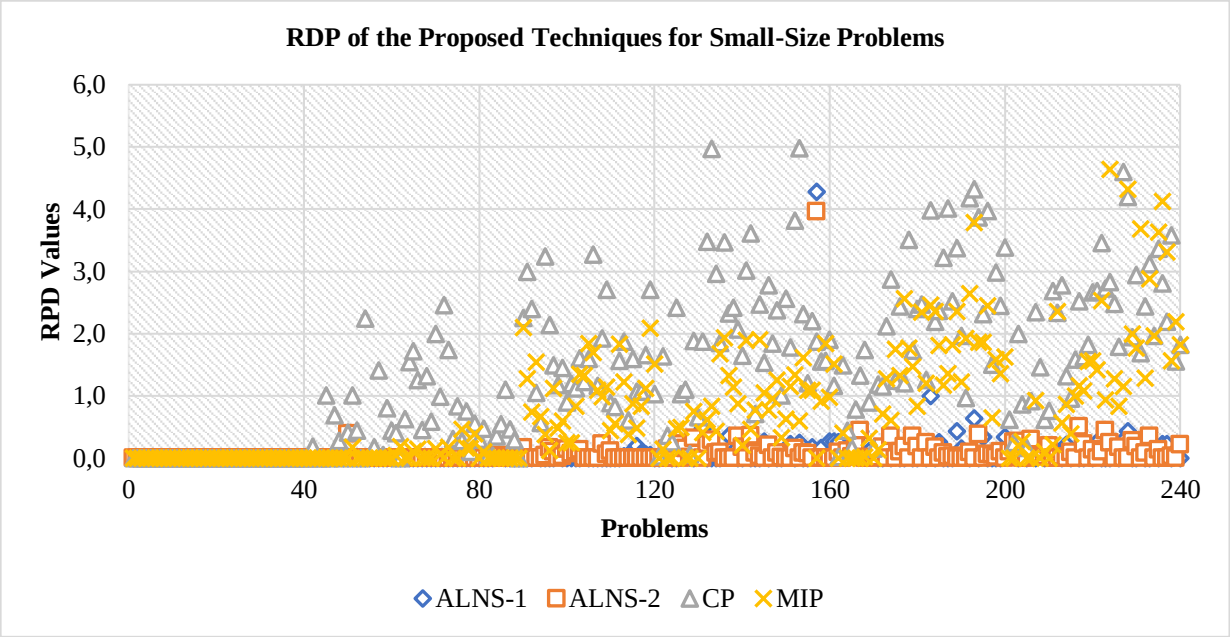

Figure 1 RPD values of each technique for the small-size problems

The detailed RPD values of each version are shown in Figure 2. It is illustrated that $ALNS_4$ provides an obviously small RPD value for each problem. Significantly, the difference of RPD value between the $ALNS_4$ and others is larger when the problem sizes are relatively small. Overall, the proposed ALNS algorithm can provide an effective solution for the considered problem.

Table 5 Comparisons of the algorithms for the large-size problems

	# of Best	*ARPD*	*Max RPD*	*Avg. Obj.*
$ALNS_1$	29	0.24	1.14	15777993.2
$ALNS_2$	47	0.15	0.75	15775594.1
$ALNS_3$	36	0.19	0.94	15776272.3
$ALNS_4$	128	0.05	0.37	15768970.5

Int. J. Pure Appl. Sci. 7(1):152-162 (2021)

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

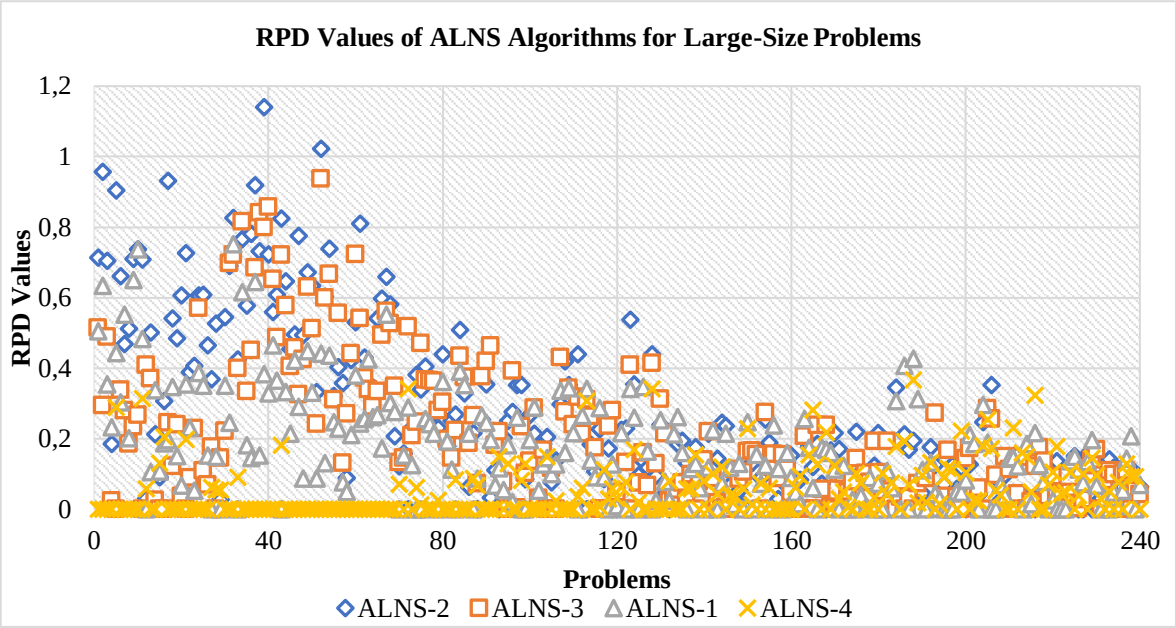

Figure 2 RPD values of each technique for the large-size problems

CONCLUSION

Contrary to traditional permutation flowshop scheduling problems, different buffer spaces and blocking conditions may be available in a real industrial case like the aerospace industry and other sectors processing industrial waste. This situation emerges a new problem type called mixed-blocking permutation flowshop. Four different blocking types, which are the classical blocking variant (RSb), two types of releasing when completing blocking on the next machine (RCb*, RCb), and no blocking (Wb), are considered in this study. A novel CP model and ALNS algorithms are proposed to solve the small and large-size problems. The performance of the models and the algorithms are tested on the well-known VRF data sets. Blocking constraints of the machines are produced for the small VRF instances respecting the same generation procedure with the literature. According to computational results, the proposed techniques ensure an effective solution to the problem.

As far as the managerial impacts of this study are concerned, MBPF problem is relatively new problem in the literature. However, many applications of this problem are avalibale in the real industry. The study provides managers an efficient total completion time value to ensure customer satisfaction and machine utilization. For companies, better solutions to the scheduling problems mean shorter completion times for their customers and high utilization for the machines or a combination thereof. Both effects can save a lot of money, which is ultimately the driving motivation of any company. Because of that, companies invest a substantial part of their budget in modern software systems to find such solutions.

Two limitations of the study are listed below. (i) The proposed heuristic algorithm is not compared with different algoirthm beucase the aim of the study is to show effective strategies to solve the problem. Therefore, it may be a limitation because of the lacking in innovation. (ii) The proposed model ant algorithm cannot works under uncertainty.

Future research should be extended with sequence-depended setup time and integrated with supply chain methodology.

CONFLICT OF INTEREST

The Author report no conflict of interest relevant to this article

RESEARCH AND PUBLICATION ETHICS STATEMENT

The author declares that this study complies with research and publication ethics.

REFERENCES

Blazewicz, J., H. Ecker, K., Pesch, E., Schmidt, G., &

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

Węglarz, J. (2007). Handbook on scheduling. From theory to applications. *International Handbook on Information Systems*. https://doi.org/10.1007/978-3-540-32220-7

Caraffa, V., Ianes, S., P. Bagchi, T., & Sriskandarajah, C. (2001). Minimizing makespan in a blocking flowshop using genetic algorithms. *International Journal of Production Economics*, *70*(2), 101–115. https://doi.org/10.1016/S0925-5273(99)00104-8

Cheng, C.-Y., Lin, S.-W., Pourhejazy, P., Ying, K.-C., & Zheng, J.-W. (2020). Minimizing Total Completion Time in Mixed-Blocking Permutation Flowshops. *IEEE Access*, *8*, 142065–142075. https://doi.org/10.1109/ACCESS.2020.3014106

Fuchigami, H., & Rangel, S. (2018). A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science*, *25*. https://doi.org/10.1016/j.jocs.2017.06.004

Grabowski, Jozef, & Pempera, J. (2000). Sequencing of jobs in some production system. *European Journal of Operational Research*, *125*(3), 535–550. https://doi.org/10.1016/S0377-2217(99)00224-6

Grabowski, Józef, & Pempera, J. (2007). The permutation flow shop problem with blocking. A tabu search approach. *Omega*, *35*(3), 302–311. https://doi.org/10.1016/J.OMEGA.2005.07.004

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics*, *5*, 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Hall, N. G., & Sriskandarajah, C. (1996). A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Oper. Res.*, *44*(3), 510–525. https://doi.org/10.1287/opre.44.3.510

Hall, N., & Sriskandarajah, C. (2000). Minimizing Cycle Time in a Blocking Flowshop. *Operations Research*, *48*, 177–180. https://doi.org/10.1287/opre.48.1.177.12451

Johnson, S. M. (1954). Optimal Two and Three Stage Production Schedules With Set-Up Time Included. *Naval Research Logistics Quarterly*, *1*, 61–68. https://doi.org/10.1002/nav.3800010110

Khorramizadeh, M., & Riahi, V. (2015). A Bee Colony Optimization Approach for Mixed Blocking Constraints Flow Shop Scheduling Problems. *Mathematical Problems in Engineering*, *2015*, 612604. https://doi.org/10.1155/2015/612604

Kizilay, D. (2018). *Integrating the Optimization of Quay and Yard Operations in Container Terminals*. Yasar University.

Kizilay, D., Eliiyi, D. T., & Van Hentenryck, P. (2018). Constraint and Mathematical Programming Models for Integrated Port Container Terminal Operations. In W.-J. van Hoeve (Ed.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (pp. 344–360). Springer International Publishing.

Ku, W.-Y., & Beck, J. C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, *73*, 165–173. https://doi.org/https://doi.org/10.1016/j.cor.2016.04.006

Lin, S. W., Cheng, C. Y., Pourhejazy, P., & Ying, K. C. (2021). Multi-temperature simulated annealing for optimizing mixed-blocking permutation flowshop scheduling problems. *Expert Systems with Applications*, *165*, 113837. https://doi.org/10.1016/j.eswa.2020.113837

Martinez, S., Dauzère-Pérès, S., Guéret, C., Mati, Y., & Sauer, N. (2006). Complexity of flowshop scheduling problems with a new blocking constraint. *European Journal of Operational Research*, *169*(3), 855–864. https://doi.org/https://doi.org/10.1016/j.ejor.2004.08.046

Mccormick, S., Pinedo, M., J. Shenker, S., & Wolf, B. (1989). Sequencing in an Assembly Line With Blocking to Minimize Cycle Time. *Operations Research*, *37*, 925–935. https://doi.org/10.1287/opre.37.6.925

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Newton, M. A. H., Riahi, V., Su, K., & Sattar, A. (2019). Scheduling blocking flowshops with setup times via constraint guided and accelerated local search. *Computers & Operations Research*, *109*, 64–76. https://doi.org/10.1016/J.COR.2019.04.024

Osman, I., & Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, *17*(6), 551–557. https://doi.org/10.1016/0305-0483(89)90059-5

Pan, Q.-K., & Ruiz, R. (2012). An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. *Omega*, *40*(2), 166–180. https://doi.org/10.1016/J.OMEGA.2011.05.002

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403–2435. https://doi.org/10.1016/J.COR.2005.09.012

Qian, B., Wang, L., Huang, D., Wang, W., & Wang, X. (2009). An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. *Computers & Operations Research*, *36*(1), 209–233. https://doi.org/10.1016/J.COR.2007.08.007

Riahi, V., Khorramizadeh, M., Hakim Newton, M. A., & Sattar, A. (2017). Scatter search for mixed blocking flowshop scheduling. *Expert Systems with*

IJPAS
ijpas@munzur.edu.tr
ISSN: 2149-0910

Research article/Araştırma makalesi
DOI: 10.29132/ijpas.911146

Applications, *79*, 20–32. https://doi.org/https://doi.org/10.1016/j.eswa.2017.02.027

Riahi, V., Newton, M. A. H., Su, K., & Sattar, A. (2019). Constraint guided accelerated search for mixed blocking permutation flowshop scheduling. *Computers & Operations Research*, *102*, 102–120. https://doi.org/10.1016/J.COR.2018.10.003

Ronconi, D P, & Armentano, V. A. (2001). Lower bounding schemes for flowshops with blocking in-process. *Journal of the Operational Research Society*, *52*(11), 1289–1297. https://doi.org/10.1057/palgrave.jors.2601220

Ronconi, Débora P. (2004). A note on constructive heuristics for the flowshop problem with blocking. *International Journal of Production Economics*, *87*(1), 39–48. https://doi.org/10.1016/S0925-5273(03)00065-3

Ruiz-Torres, A. J., Ho, J. C., & Ablanedo-Rosas, J. H. (2011). Makespan and workstation utilization minimization in a flowshop with operations flexibility. *Omega*, *39*(3), 273–282. https://doi.org/10.1016/J.OMEGA.2010.07.004

Sawik, T. (1995). Scheduling flexible flow lines with no in-process buffers. *International Journal of Production Research - INT J PROD RES*, *33*, 1357–1367. https://doi.org/10.1080/00207549508930214

Sawik, T. J. (1993). A scheduling algorithm for flexible flow lines with limited intermediate buffers. *Applied Stochastic Models and Data Analysis*, *9*, 127–138.

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In M. Maher & J.-F. Puget (Eds.), *Principles and Practice of Constraint Programming --- CP98* (pp. 417–431). Springer Berlin Heidelberg.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278–285. https://doi.org/10.1016/0377-2217(93)90182-M

Tasgetiren, M. F., Kizilay, D., Pan, Q.-K., & Suganthan, P. N. (2017). Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. *Computers and Operations Research*, *77*. https://doi.org/10.1016/j.cor.2016.07.002

Tasgetiren, M. F., Pan, Q.-K., Kizilay, D., & Gao, K. (2016). A Variable Block Insertion Heuristic for the Blocking Flowshop Scheduling Problem with Total Flowtime Criterion. *Algorithms*, *9*(4). https://doi.org/10.3390/a9040071

Tasgetiren, M. F., Pan, Q.-K., Kizilay, D., & Suer, G. (2015). A populated local search with differential evolution for blocking flowshop scheduling problem. *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings*. https://doi.org/10.1109/CEC.2015.7257235

Trabelsi, W, Sauvey, C., & Sauer, N. (2011). Complexity and Mathematical Model for Flowshop Problem Subject to Different Types of Blocking Constraint. *IFAC Proceedings Volumes*, *44*(1), 8183–8188. https://doi.org/https://doi.org/10.3182/20110828-6-IT-1002.01887

Trabelsi, Wajdi, Sauvey, C., & Sauer, N. (2010). *Heuristic methods for problems with blocking constraints solving jobshop scheduling*.

Trabelsi, Wajdi, Sauvey, C., & Sauer, N. (2012). Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems. *Computers & Operations Research*, *39*(11), 2520–2527. https://doi.org/https://doi.org/10.1016/j.cor.2011.12.022

Vallada, E., & Ruiz, R. (2010). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, *38*(1–2), 57–67. https://doi.org/10.1016/J.OMEGA.2009.04.002

Vallada, E., Ruiz, R., & Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, *240*(3), 666–677. https://doi.org/https://doi.org/10.1016/j.ejor.2014.07.033

Zhang, G., Xing, K., & Cao, F. (2018). Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. *Engineering Applications of Artificial Intelligence*, *76*, 96–107. https://doi.org/10.1016/J.ENGAPPAI.2018.09.005