IJE TRANSACTIONS A: Basics Vol. 37 No. 07, (July 2024) 1443-1465

International Journal of Engineering

Journal Homepage: www.ije.ir

Energy-Conscious Common Operation Scheduling in an Identical Parallel Machine Environment

H. Ataei, F. Ahmadizar*, J. Arkat

Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran

PAPER INFO

Paper history:
Received 14 October 2023
Received .30 November 2023
Accepted 12 December 2023

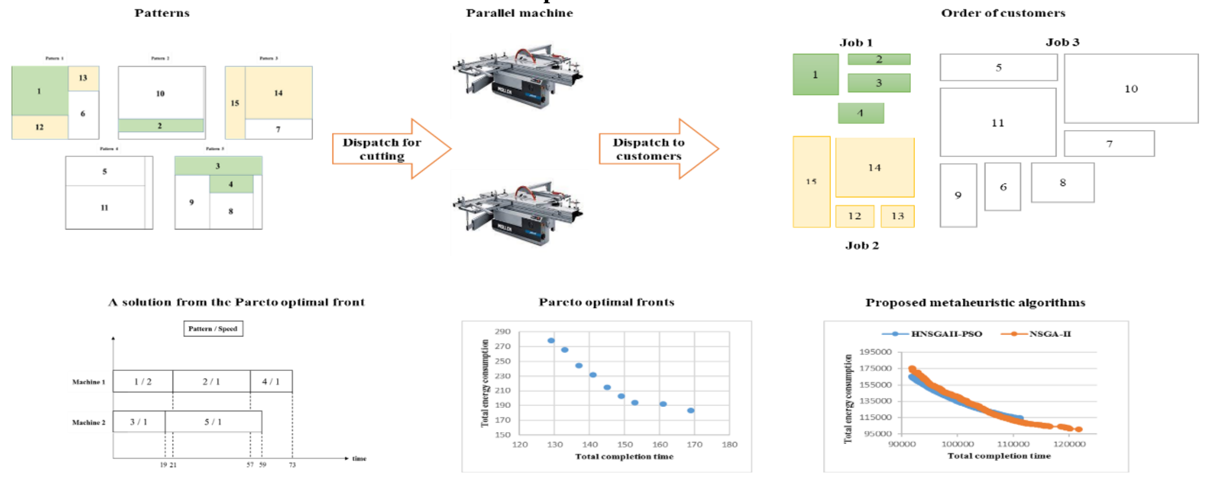
Keywords:
Bi-objective Mixed Integer Linear Programming
Identical Parallel Machine Scheduling
Common Operation
Total Energy Consumption
Total Completion Time

ABSTRACT

The relentless growth of global energy consumption poses a multitude of complex challenges, including the depletion of finite energy resources and the exacerbation of greenhouse gas emissions, which contribute to climate change. In the face of these pressing environmental concerns, the manufacturing sector, a significant energy consumer, is under immense pressure to adopt sustainable practices. The critical intersection of energy consumption management and production operation scheduling emerges as a pivotal domain for addressing these challenges. The scheduling of common operations, exemplified by the cutting stock problem in industries like furniture and apparel, represents a prevalent challenge in production environments. For the first time, this paper pioneers an investigation into an identical parallel machine scheduling problem, taking into account common operations to minimize total energy consumption and total completion time concurrently. For this purpose, two bi-objective mixed integer linear programming models are presented, and an augmented ε – constraint method is used to obtain the Pareto optimal front for small-scale instances. Considering the NP-hardness of this problem, a non-dominated sorting genetic algorithm (NSGA-II) and a hybrid non-dominated sorting genetic algorithm with particle swarm optimization (HNSGAII-PSO) are developed to solve medium- and large-scale instances to achieve good approximate Pareto fronts. The performance of the proposed algorithms is assessed by conducting computational experiments on test problems. The results demonstrate that the proposed HNSGAII-PSO performs better than the suggested NSGA-II in solving the test problems.

***doi**: 10.5829/ije.2024.37.07a.20*

Graphical Abstract



*Corresponding Author Institutional Email: *f.ahmadizar@uok.ac.ir* (F. Ahmadizar)

1. INTRODUCTION

Energy plays a critical role in human society, and as nonrenewable energy resources continue to deplete, the need for effective and efficient use of energy becomes increasingly vital. The management of energy consumption across diverse processes and sectors has garnered significant attention among researchers owing to its demonstrated ability to mitigate energy expenditures, alleviate environmental repercussions, and enhance energy security (1-4). The production and consumption of energy are major sources of greenhouse gas emissions. The manufacturing sector, which accounts for roughly half of the world's total energy consumption, is a major contributor to greenhouse gas emissions (5-7). It is essential for manufacturing industries to prioritize improving energy efficiency and minimizing greenhouse gas emissions. To achieve this, production managers can choose from a variety of energy-saving strategies. Implementing energy-efficient machines is one potential strategy, although this approach may incur greater upfront costs, which can strain a company's finances. To reduce the impact of this issue, production managers may need to implement scheduling strategies that strike a balance between energy conservation and cost reduction (8). Scheduling problems have been classified into well-known categories based on the machine environment. The most common types of scheduling problems are single machine, parallel machine, flow shop, and job shop. Among these, parallel machine scheduling is particularly significant in scheduling problems because it is a generalization of single machine scheduling and a specific mode of flexible flow shop scheduling (9). Over the past few years, scheduling problems that involve simultaneously obtaining objectives for scheduling and energy have been the subject of extensive research. The related literature has introduced and accepted various electricity consumption strategies, such as power-down and speed-scaling. Furthermore, different policies for determining electricity consumption costs, including fixed, time-of-use (TOU), and tiered pricing, have also been formulated and employed (10-13). Parallel machine scheduling problems that consider energy consumption account for a significant portion of the studies mentioned.

Li et al. (14) investigated the unrelated parallel machine scheduling problem to minimize energy costs and tardiness. In this problem, energy consumption has been considered for various machine modes, including setup, idleness, and processing. For this problem, a mathematical model has been developed so that two objectives are regarded as one single objective. To solve the problem, ten heuristic methods based on the priority rules, combination rules, and energy consumption have been proposed. Che et al. (15) investigated an energy-efficient unrelated parallel machine scheduling problem to minimize total electricity consumption costs. In this problem, the cost of electricity consumption is calculated using the TOU tariffs, and the makespan is limited. These researchers proposed a two-stage algorithm to solve the problem. Wang et al. (16) studied the parallel machine scheduling problem with a bounded power demand peak. In this problem, the jobs can be processed at different speeds and therefore have various processing times and power demands. The goal of this problem is to minimize the makespan, and to solve it, the researchers proposed a genetic algorithm based on a two-stage heuristic method.

Zeng et al. (17) conducted a comprehensive investigation of a bi-objective optimization problem pertaining to uniform parallel machine scheduling. The aim of their study was to minimize both the number of machines employed and the total electricity cost within the framework of TOU tariffs. To accomplish this, the researchers devised an iterative search framework, facilitating the attainment of the Pareto front. Wu & Che (18) investigated an energy-efficient bi-objective unrelated parallel machine scheduling problem with the goal of minimizing both total energy consumption and makespan. They suggested a memetic differential evolution (MDE) algorithm to solve the problem, and they developed a local search approach to improve the proposed algorithm. Cota et al. (19) conducted an investigation into the unrelated parallel machine scheduling problem, incorporating sequence-dependent setup times to minimize both total electrical energy consumption and makespan. Furthermore, to find solutions close to the Pareto optimal front, they developed and tested a novel heuristic algorithm called Smart Pool. Safarzadeh & Niaki (20) investigated bi-objective green scheduling in uniform parallel machine environments. In this problem, various green cost rates for every machine were considered to model the impact of production resources on sustainability, such as energy consumption and carbon emissions. The researchers modeled the problem with the objective of minimizing the total green costs and makespan, and they used the ε-constraint method to identify Pareto optimal solutions. Wang et al. (21) investigated the identical parallel machine scheduling problem with the goal of minimizing both the makespan and total energy consumption. In this problem, the cost of electricity consumption is calculated using the TOU tariffs. The researchers used the augmented ε-constraint method to solve the problem in small-scale instances, and they developed a constructive heuristic (CH) method with a local search strategy based on the problem's characteristics to solve the problem in larger-scale instances. Anghinolfi et al. (22) developed an ad hoc heuristic method to solve the problem proposed by Wang et al. (21). This method is divided into two parts. The first section of the method is an improved and modified version of the constructive heuristic (CH) algorithm proposed by Wang et al. (21). The second section introduces a new local search method for enhancing the efficacy of Pareto solutions. They conducted computational tests to assess the suggested method's efficiency and effectiveness. Zhang et al. (23) delved into a two-stage parallel machine scheduling problem, aiming to minimize total electricity costs under TOU tariffs. Their study considered a two-stage parallel machine system comprising identical parallel speed-scaling machines at stage 1 and unrelated parallel machines at stage 2. The researchers modeled the investigated problem as mixed-integer linear programming and developed a Tabu Search-Greedy Insertion Hybrid (TS-GIH) algorithm to solve it. Keshavarz et al. (24) studied the unrelated parallel machine scheduling problem with sequence-dependent setup times to minimize the energy consumption costs and makespan. They used the ε-constraint method to solve the problem in small instances, and they developed the multiple objective simulated annealing and multiple objective particle swarm optimization algorithms to solve the problem in medium and large instances. Zhou & Gu (25) examined the unrelated parallel machine scheduling problem by taking into account multiple resource constraints to minimize total energy consumption and total completion time. They developed a multi-objective artificial immune algorithm to solve the problem. Módos et al. (26) examined the problem of parallel dedicated machines while keeping energy consumption constraints in mind. In this problem, the peak energy consumption at specific time intervals should not exceed a specified limit. The researchers studied four different variants of the problem and designed a heuristic algorithm for the general problem. Rego et al. (27) proposed a novel bi-objective unrelated parallel machine scheduling problem that considers TOU tariffs and sequence-dependent set-up times. To tackle the problem, the researchers suggested a bi-objective mixed-integer linear programming formulation to minimize the total energy consumption and makespan. To solve small and large instances of the problem, they used the weighted sum method and the non-dominated sorting genetic algorithm (NSGA-II), respectively. Asadpour et al. (28) studied the identical parallel machine scheduling problem with the job-splitting property to minimize the total number of tardy jobs and total energy consumption. In this problem, the jobs can be further subdivided. An augmented ε-constraint method was used to solve small-scale problems, and a simulated annealing (SA) algorithm was designed to solve medium- and large-scale problems. Heydar et al. (28) proposed an approximate dynamic programming (ADP) approach to address an energy-efficient unrelated parallel machine scheduling problem characterized by random job arrivals. The objective of their work is to minimize a weighted combination of makespan and total energy costs. The energy costs encompass the energy consumption incurred during machine switching, job processing, and idle periods. At each stage of the ADP, a binary program was formulated to optimize the scheduling problem. The energy efficiency strategy considered in their study is TOU electricity tariffs. Gaggero et al. (29) investigated the problem of bi-objective scheduling on parallel identical machines with TOU costs (BPMSTP). They introduced a new mathematical formulation for the BPMSTP, which allowed them to develop a more efficient exact algorithm for finding the optimal Pareto front. Additionally, they proposed an alternative heuristic approach called the Enhanced Heuristic Scheduler (EHS), which proved to outperform existing heuristics in experiments. Not only did EHS demonstrate superior performance, but it also enhanced the computational efficiency of the exact approach. Sanati et al. (28) conducted a comprehensive investigation into an unrelated parallel machine scheduling problem considering sequence-dependent setup times under TOU electricity tariffs. Their study meticulously examined setup times in two distinct modes: disjointed from and jointed to processing time. For each of these problem variations, two mixed-integer linear programming models were meticulously formulated. The presented models for the problem with setup time disjointed from processing time demonstrated the capability of solving instances involving up to 16 machines and 45 jobs. In contrast, this capability was extended to 20 machines and 40 jobs for the processing time jointed to the setup time problem. Furthermore, to address large-size instances effectively, a fix and relax heuristic algorithm was proposed. This algorithm exhibited the ability to solve instances of up to 20 machines and 100 jobs for each of the two considered problems.

In studies on parallel machine scheduling, after a job is assigned to a machine, that machine processes the job, and the job is ready to be delivered. In the real world, however, there are problems in which each job consists of several sub-jobs, and the job is ready to be delivered when all of its sub-units have been completed after processing one or more activities. Furthermore, processing an activity may have an impact on the completion of multiple jobs. These problems are introduced as "common operation scheduling" (COS) problems. Common operation scheduling problems are used to find the optimal arrangement of operations required by a set of jobs under the assumption that when an activity is completed, it is completed for all jobs that require it (30). These types of problems have different applications, including movie shooting (31), progressive network recovery (32), and pattern sequencing in cutting stock problems (33).

The cutting stock problem includes cutting a set of available pieces in stock (objects) to produce a specific set of smaller pieces (items) that optimizes an objective function such as minimizing total waste, maximizing profit, or minimizing production costs, in order to meet customer demand for different items. The cutting stock problem occurs in numerous industrial processes where the objects can be sheets of wood, sheets of metal, steel bars, paper or aluminum rolls, printed circuit boards, sheets of glass, etc. In these industries, using appropriate cutting programs is frequently associated with reducing production costs and increasing productivity. A cutting program is a solution to the cutting stock problem, which is proposed by a set of cutting patterns and the frequency of their use. A cutting pattern specifies a subset of items to be obtained from cutting an object. If the items in the cutting problem are two-dimensional or multi-dimensional, then the cutting patterns also determine the arrangement of the items on each of the objects (34). For example, in a furniture factory, sheets of wood must be cut by one or more machines according to predetermined cutting patterns to produce smaller pieces. Each customer order (job) consists of several small pieces that may be placed in one or more different patterns. Therefore, an order is ready to be delivered to the customer when all of the patterns required by that order have been processed by cutting machines.

Several papers in the related literature have attempted to categorize cutting stock problems, which can be used as references to study and gain more information about these categories (35-37). Different criteria can be used to categorize cutting stock problems. The most common criterion is cutting dimensions, used to describe cutting patterns according to the type of problem. Based on this criterion, cutting stock problems are divided into one-dimensional, two-dimensional, three-dimensional, or multi-dimensional problems. According to the literature review, most cutting stock problems are focused on one- and two-dimensional cutting, which can be used in problems such as cutting paper, cables, pipes, sheet wood, etc. (37). In most related studies, the problem not only involves designing cutting patterns and selecting a number of them to produce items smaller than objects (customer orders), but also determining the sequence of patterns to achieve objectives related to completion times, due dates, production costs, etc. Arbib & Marinelli (38) studied the one-dimensional cutting stock problem by considering due dates to minimize the weighted tardiness of the jobs and raw material costs. In this problem, each job has a due date and consists of several pieces of the same size. Researchers developed and tested implicit enumeration, upper bounds, and heuristic methods to solve the problem. Cui et al. (39) studied the one-dimensional cutting stock problem, considering the setup costs. In their study, they developed an integer linear programming model to minimize the sum of setup and material costs over a given pattern set. To solve the problem, they introduced a heuristic algorithm based on sequential grouping to generate patterns. Wuttke & Heese (40) investigated the two-dimensional cutting stock problem with sequence-dependent setup times and permissible tolerances in the textile industry. For the problem under study, they provided a mixed-integer program, and to solve it, they used a sequential heuristic with a feedback loop based on Gilmore and Gomory's approach.

While mentioning that the optimal cutting patterns for the production of items smaller than objects are designed by commercial software considering the minimum cutting waste, some papers have considered the pre-designed patterns as the input of the problem and have set the sequence of patterns to achieve the objectives regarding completion time and due date (30, 33). Arbib et al. (30) studied the pattern sequencing problem, which is introduced as a common operation scheduling problem, with the goal of minimizing the weighted number of tardy jobs in the single machine environment. They reformulated the problem as a stable set problem on a special graph and analyzed the graph structure. In this problem, the processing times of the patterns on the machine are considered the same and equal to one unit. Arbib et al. (33) investigated the problem of common operation scheduling in single machine and parallel machine environments. In this study, in a single machine environment, the processing times of the patterns on the machine are variable, and in a parallel machine environment, the processing times of the patterns on each of the machines are considered the same and equal to one unit. The researchers formulated the problem as a set-covering problem and solved it using the branch-and-cut algorithm.

Table 1 provides a concise summary of the literature review conducted on energy-efficient parallel machine scheduling. The reviewed articles have been categorized based on the energy consumption strategy employed. To facilitate a comparison of the problem addressed in this article with those investigated in the literature, four criteria have been employed: problem properties, objective function, solution algorithm, and energy consumption strategy. These criteria and their corresponding details are also presented in Table 1. In the realm of energy-efficient parallel machine scheduling problems, an underlying assumption is that the processing of each operation by a machine exclusively affects the completion of a single job. However, this assumption may not always hold true for real-world problems. In certain scenarios, several jobs to be completed may require the common operation to be performed on a shared resource simultaneously, introducing a new dimension of complexity to the scheduling problem that has not been discussed in the literature.

This study focuses on common operation scheduling in an environment of identical parallel machines while considering energy consumption. To this end, two mixed-integer linear programming models, a position-based model and a sequence-based model, have been

TABLE 1. Summary of the Literature Review on Energy-Efficient Parallel Machine Scheduling

Author	Other properties of the problem	Objective/ Solution method	Strategy of energy consumption
[14]	unrelated parallel machine, energy consumption has been considered for various machine modes	tardiness and energy consumption cost/ heuristic algorithms	fixed
[20]	uniform parallel machine	total green costs and makespan/ the ε-constraint method	
[25]	unrelated parallel machine, multiple resource constraint	total energy consumption and total completion time/ multi-objective artificial immune algorithm	
[28]	identical parallel machine, job-splitting property	total number of tardy jobs and total energy consumption/ augmented ε-constraint, simulated annealing algorithm	
[16]	bounded power demand peak	makespan/ genetic algorithm	speed-scaling
[18]	unrelated parallel machine	total energy consumption and makespan/ memetic differential evolution	
[19]	unrelated parallel machine, sequence-dependent setup times	total energy consumption and makespan/ heuristic algorithm	
[15]	unrelated parallel machine, makespan is limited	energy consumption cost/ heuristic algorithm	TOU
[17]	uniform parallel machine	number of machines employed and the total electricity cost/ heuristic algorithm	
[21]	identical parallel machine	makespan and total energy consumption/ augmented ε-constraint method, constructive heuristic, NSGA-II	
[23]	two-stage parallel machine	total energy consumption/ Tabu Search-Greedy Insertion Hybrid algorithm	
[24]	unrelated parallel machine, sequence-dependent setup times	energy consumption costs and makespan/ the ε-constraint method, multiple objective simulated annealing algorithm and multiple objective particle swarm optimization algorithms	
[26]	parallel dedicated machines, peak energy consumption at specific time intervals should not exceed a specified limit	makespan/ heuristic algorithm	
[27]	unrelated parallel machine, sequence-dependent set-up times	total energy consumption and makespan/ weighted sum method, NSGA-II	
[29]	unrelated parallel machine, random job arrivals	weighted combination of makespan and total energy costs/ approximate dynamic programming	
[30]	identical parallel machine	total energy consumption and makespan/ Enhanced Heuristic Scheduler	
[31]	unrelated parallel machine, sequence-dependent setup times, bounded makespan	total electricity cost/fix and relax heuristic algorithm	
Current paper	identical parallel machine, common operation scheduling	total energy consumption and total completion time/ augmented ε-constraint method, HNSGAII-PSO, NSGA-II	Speed-scaling

proposed for the problem under study in order to simultaneously minimize the total energy consumption and the total completion time. To solve small-scale instances, the augmented ε-constraint method (AUGMECON) is used to obtain the Pareto optimal front. Since the problem is NP-hard, a non-dominated sorting genetic algorithm (NSGA-II) and a hybrid non-dominated sorting genetic algorithm with particle swarm optimization (HNSGAII-PSO) have been developed to solve medium- and large-scale instances. In the proposed NSGA-II algorithm, each chromosome represents a solution from the problem-solving space, and the quality of each chromosome can be evaluated by calculating the objective functions of total completion time and total energy consumption. In the proposed HNSGAII-PSO algorithm, each chromosome represents a region of the problem-solving space where all solutions in this region have the same total energy consumption. To evaluate the quality of each chromosome, the best solution for the region covered by that chromosome is considered. In the investigated problem, the best solution in the region covered by each chromosome is the one that minimizes the total completion time. Therefore, to determine the best solution in each region, the PSO algorithm is executed once for each chromosome. The total completion time for each chromosome of the HNSGAII-PSO algorithm is determined based on the output of the PSO algorithm. According to the mentioned explain, the contributions of the paper can be summarized as follows:

- Energy consumption is considered in the common operation scheduling problem.
- Two bi-objective mixed integer linear programming models, namely the position-based and sequence-based models, are proposed to investigate the trade-off between total completion time and total energy consumption.
- The NSGA-II and HNSGAII-PSO algorithms are developed to solve large-scale instances.
- The performance of the methods is evaluated using computational experiments.

The remaining sections of this paper are as follows: After the problem is described in section 2, mathematical models are proposed. In section 3, solution methods will be presented. Section 4 discusses the results of the computations. In section 5, the conclusion and future research are mentioned.

2. PROBLEM DESCRIPTION AND MATHEMATICAL MODELING

The common operation scheduling problem is one of the scheduling problems with numerous applications in the actual world. One of the applications of common operation scheduling in manufacturing environments is the cutting stock problem, which is posed in industries such as the furniture industry (cutting wooden panels) and the apparel industry (cutting fabric). This paper investigates common operation scheduling in an identical parallel machine environment, considering energy consumption. In the problem under study, each job includes several small pieces (items), and all the pieces of different jobs are placed on a number of cutting patterns to produce the required small pieces according to these patterns and by cutting larger pieces (objects). Therefore, each job is completed when all the small pieces related to it have been produced by cutting the required patterns. The relationship between jobs and cutting patterns is shown by the job-pattern matrix. In fact, this matrix indicates which patterns must be processed to complete each job. The optimal cutting patterns are predetermined using relevant software, taking the dimensions of small pieces into account, and will be available at time 0. Each pattern contains one or more pieces and contributes to the completion of one or more jobs. Energy consumption is investigated by considering the speed-scaling strategy. In accordance with this strategy, each machine possesses varying speed levels. Consequently, when the machine operates at a higher speed, the processing time is reduced while the consumption of electrical energy increases. In this problem, the assignment of cutting patterns to machines, the sequence of patterns on each machine, and the appropriate speed of the machine to process each of the assigned patterns are determined in order to simultaneously minimize the total completion time and the total energy consumption. The underlying assumptions are listed below:

- Cutting patterns have already been designed and determined and will be available at time zero.
- Larger pieces (objects) will be available at time 0 to produce smaller parts (items), and their number is equal to the number of cutting patterns.
- Each machine processes a maximum of one pattern at a time.
- Each pattern can be processed by only one machine at a time.
- The processing time of each pattern at the varying speed levels of each machine is specified and definite.
- The amount of energy consumed by each machine at various speed levels for processing each pattern is specified and definite.
- There is no precedence relationship between different patterns.
- There is no preemption in the processing of patterns.

In the following, the notations and parameters are presented.

Sets and Indices	
I	Set of jobs $\{i \epsilon I\}$.
J	Set of cutting patterns $\{j, k \epsilon J\}$.
$V = J \cup \{0\}$	Set of cutting patterns that includes the fictitious pattern 0 $\{j, k \epsilon V\}$.
M	Set of machines $\{m \epsilon M\}$.
Q	Set of positions on each machine $\{q \epsilon Q\}$.
S	Set of speed levels of each machine $\{s \epsilon S\}$.
N_i	Subset of cutting patterns set that must be processed to complete job i ($\bigcup_i N_i = J$).

Parameters	
w_j	The required workload of pattern j.
v_s	Machine processing speed at speed level s (the workload processed per unit of time by the machine at speed level s).
p_{js}	The time required to process pattern j at speed level s of the machine ($p_{js} = \frac{w_j}{v_s}$).
π_j	Machine energy consumption rate for processing pattern j per unit of time.
π_{js}	Machine energy consumption rate at speed level s for processing pattern j per unit of time ($\pi_{js} = \pi_j v_s^{\alpha}$, $\alpha > 1$).
e_{js}	The energy required to process pattern j at speed level s of the machine ($e_{js} = \pi_{js} p_{js}$).
B	A large number.

The **sequence-based** mathematical model is presented here.

Decision varibales	
x_{mkjs}	Equal to 1 if pattern j is processed immediately after pattern k on machine m with speed level s,and 0 otherwise.
r_j	Completion time of pattern j.
c_i	Completion time of job i.

Sequence-based model		
$\min Z_1 = \sum_{i\epsilon I} c_i$		(1)
$min\, Z_2 = \sum_{m\epsilon M} \sum_{k\epsilon V:j\neq k} \sum_{j\epsilon J} \sum_{s\epsilon S} e_{js} x_{ml}$		(2)
$\sum_{m\epsilon M} \sum_{k\epsilon V:j\neq k} \sum_{s\epsilon S} x_{mkjs} = 1$	$\forall\, j\epsilon J$	(3)
$\sum_{m\epsilon M} \sum_{j\epsilon J:j\neq k} \sum_{s\epsilon S} x_{mkjs} \leq 1$	$\forall\, k\epsilon J$	(4)
$\sum_{j\epsilon J} \sum_{s\epsilon S} x_{m0js} \leq 1$	$\forall\, m\epsilon M$	(5)
$\sum_{j\epsilon V:k\neq j} \sum_{s\epsilon S} x_{mkjs} - \sum_{h\epsilon V:h\neq j} \sum_{s\epsilon S} x_{mhks} = 0$	$\forall\, k\epsilon J\,, m\epsilon M$	(6)
$r_j - r_k + B(1 - x_{mkjs}) \geq p_{js}$	$\forall\, k\epsilon V : j \neq k, j\epsilon J\,, m\epsilon M\,, s\epsilon S$	(7)
$c_0 = 0$		(8)
$c_i = \max_{j\epsilon N_i} r_j$	$\forall\, i\epsilon I$	(9)
$x_{mkjs} = \{0\,,1\}$	$\forall\, j,k\ \epsilon V\,, m\epsilon M\,, s\epsilon S$	(10)
$c_i\,, r_j \geq 0$	$\forall\, j\epsilon V\,, i\epsilon I$	(11)

The objective functions 1 and 2 represent the minimization of total completion time and total energy consumption for the sequence-based model, respectively. Constraint 3 guarantees that each pattern is assigned to only one machine at a specified speed level and that only one other pattern is processed before it. Constraint 4 indicates that after each pattern, a maximum of one other pattern can be processed. Constraint 5 ensures that in each machine, after fictitious pattern 0, a maximum of one other pattern can be processed. Constraint 6 ensures the right order for allocating patterns in each machine: if pattern k is processed before pattern j, then another pattern must be processed before pattern k. Constraint 7 computes the completion time of each pattern. If $x_{mkjs} = 1$, then the completion time of pattern j is obtained from the sum of the completion time of pattern k and the processing time of pattern j with the speed level s of the machine, and if $x_{mkjs} = 0$, then the large number B ensures the relation. Equation 8 shows that the completion time of the fictitious pattern 0 is zero. Each job's completion time is calculated by constraint 9. The completion time of job i is equal to the maximum completion time of patterns that include the pieces of job i. Constraints 10 and 11 show the range of decision variables.

Considering Equation 9, the proposed model is mixed integer nonlinear programming. The proposed sequence-based model is converted into a mixed integer linear programming model by substituting Equation 12 for Equation 9.

$$c_i \geq r_j \qquad \forall\, i\epsilon I\ , j\epsilon N_i \tag{12}$$

The **position-based** mathematical model is presented here.

Decision varibales	
x_{jmqs}	Equal to 1 if pattern j is processed in position q of machine m with speed level s,and 0 otherwise.
h_{qm}	Start time of position q of machine m.
f_{qm}	Finish time of position q of machine m.
r_{jm}	Completion time of pattern j on machine m.
c_i	Completion time of job i.

Position-based model		
$min\, Z_1 = \sum_{i\epsilon I} c_i$		(13)
$min\, Z_2 = \sum_{j\epsilon J} \sum_{m\epsilon M} \sum_{q\epsilon Q} \sum_{s\epsilon S} e_{js} x_{jmqs}$		(14)
$\sum_{m\epsilon M} \sum_{q\epsilon Q} \sum_{s\epsilon S} x_{jmqs} = 1$	$\forall\, j\epsilon J$	(15)
$\sum_{j\epsilon J} \sum_{s\epsilon S} x_{jmqs} \leq 1$	$\forall\, m\epsilon M\,, q\epsilon Q$	(16)
$f_{qm} = h_{qm} + \sum_{j\epsilon J} \sum_{s\epsilon S} p_{js} x_{jmqs}$	$\forall\, m\epsilon M\,, q\epsilon Q$	(17)
$f_{qm} \leq h_{(q+1)m}$	$\forall\, m\epsilon M\,, q\epsilon Q$	(18)
$\sum_{j\epsilon J} \sum_{s\epsilon S} x_{jmqs} \leq \sum_{j\epsilon J} \sum_{s\epsilon S} x_{jm(q-1)s}$	$\forall\, m\epsilon M\,, q\epsilon Q : q > 1 M\,, s\epsilon S$	(19)
$f_{qm} \leq r_{jm} + B(1 - \sum_{s\epsilon S} x_{jmqs})$	$\forall\, j\epsilon J\,, m\epsilon M\,, q\epsilon Q$	(20)
$c_i = \max_{j\epsilon N_i,\ m\epsilon M} r_{jm}$	$\forall\, i\epsilon I$	(21)
$x_{jmqs} = \{0\,,1\}$	$\forall\, j\epsilon J\,, m\epsilon M\,, q\epsilon Q, s\epsilon S$	(22)
$c_i\,, r_{jm}, f_{qm}, h_{qm} \geq 0$	$\forall\, j\epsilon J\,, m\epsilon M\,, q\epsilon Q, i\epsilon I$	(23)

The position-based model's objectives are to minimize total completion time and total energy consumption, which are demonstrated by Equations 13 and 14, respectively. Equation 15 guarantees that each pattern is processed only at one position on one machine and at a specified speed level. Constraint 16 indicates that at each position of each machine, a maximum of one pattern is processed at a specified speed level. Constraints 17 and 18 compute the start time and end time of each position of each machine. Constraint 19 ensures that the positions of each machine are assigned to patterns in numerical order from first to last. Each pattern's completion time is calculated by constraint 20. The completion time of each job is calculated using Equation 21. The completion time of job i is equal to the maximum completion time of patterns that include the pieces of job i. Constraints 22 and 23 show the range of decision variables.

Considering Equation 21, the proposed model is mixed-integer nonlinear programming. In order to linearize the mathematical model, Equation 24 is used instead of Equation 21. Due to the changes made, the proposed position-based model is a mixed integer linear programming model.

$$c_i \geq r_{jm} \qquad \forall\, i\epsilon I\ \ , j\epsilon N_i\ \ , m\epsilon M \qquad (24)$$

Based on the triple notation provided by Wuttke and Heese (41), the problem studied in this paper is denoted as $Pm|cos|TEC, \sum C_i$, where Pm indicates identical parallel machines, cos indicates common operation, and TEC and $\sum C_i$ represent the total energy consumption and the total completion time, respectively. In this problem, if the machines have only one level in terms of processing speed and the objective function of the total energy consumption is not considered, and if each pattern is only effective in completing one job and the completion of each job only requires the processing of one pattern, then the problem becomes an identical parallel machine problem in order to minimize the total completion time and is shown as $Pm||\sum C_i$. According to the previous studies, the problem $Pm||\sum C_i$ is the NP-hard (42). Therefore, it can be concluded that the problem proposed in this article is at least NP-hard.

3. NUMERICAL EXAMPLE

To further illustrate the problem described, consider a small numerical example involving three jobs and two identical machines. In this example, *job 1* includes pieces {1,2,3,4}, *job 2* includes pieces {5,6,7,8,9,10,11}, and *job 3* includes pieces {12,13,14,15}, and each machine has two levels of slow speed (level 1) and fast speed (level 2). Figure 1 shows the placement of small pieces (items) on objects using five cutting patterns. Table 2 lists the processing time and energy required to process patterns at different speed levels for each machine. Figure 2 shows the job-pattern matrix, which is formed based on the placement of small pieces of each job and their relationship with cutting patterns. In other words, this matrix determines the patterns associated with each job.

Figure 3 shows a solution from the Pareto optimal front for this example. In this solution, the first machine

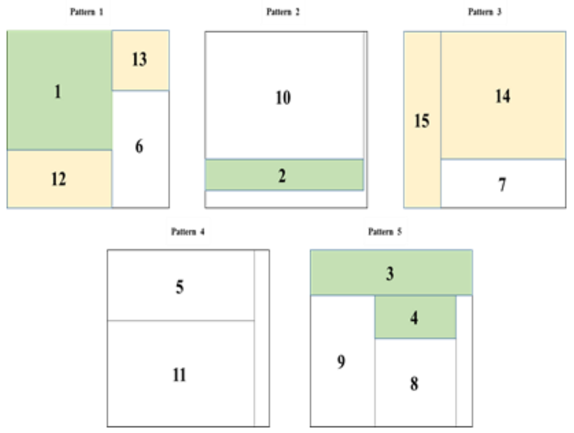
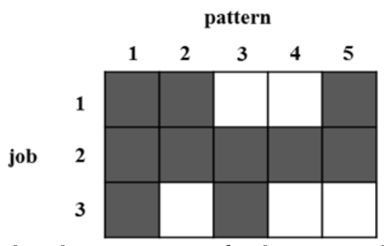


Figure 1. Cutting patterns for the numerical example

TABLE 2. Energy consumption and processing time to process patterns for the numerical example

			Pattern				
			1	2	3	4	5
Machine	Level 1 (slow)	time	27	36	19	16	40
		energy	18	61	20	16	68
	Level 2 (fast)	time	21	28	15	12	32
		energy	29	95	32	25	97



		pattern				
		1	2	3	4	5
job	1	■	■	□	□	■
	2	■	■	■	■	■
	3	■	□	■	□	□

Figure 2. The job-pattern matrix for the numerical example

processes pattern 1 with speed level 2, then it processes, respectively, patterns 2 and 4 with speed level 1. Patterns 3 and 5 are assigned to the second machine and processed there at speed level 1 of the machine, respectively. In this solution, based on the completion times of the patterns and the job-pattern matrix, job 1 is completed at time 59, job 2 at time 73, and job 3 at time 21. Therefore, in this solution, the total completion time is 153 time units, and the energy consumption is 194 units. In Figure 4, the Pareto optimal front is provided for this example. The analysis demonstrates that the optimal speed of the machines depends on the objective function. If the objective function seeks to minimize the total completion time, then processing all patterns at speed level 2 yields the best result. However, this decision increases the objective function of total energy consumption. In

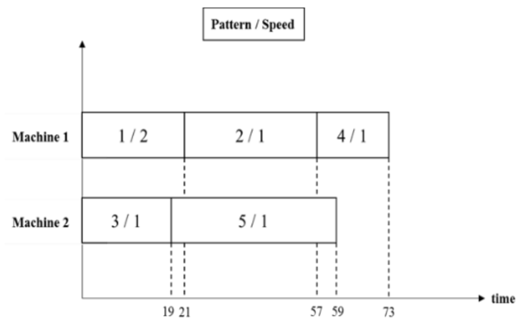


Figure 3. A solution from the Pareto optimal front for the numerical example

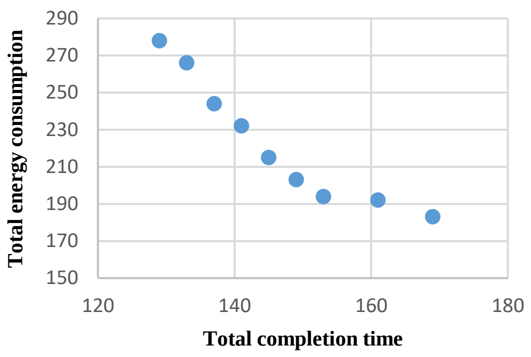


Figure 4. The Pareto optimal front for the numerical example

contrast, when all patterns are processed at speed level 1, the objective function of total energy consumption reaches its minimal value. Thus, decision-makers must consider the trade-off between the objectives and identify the best options for achieving the desired balance between the goals.

4. SOLUTION APPROACHES

In solving bi- and multi-objective problems, the primary goal is to identify a set of Pareto optimal solutions, considering trade-offs between objectives. Due to the NP-hard nature of the problem under investigation, a non-dominated sorting genetic algorithm (NSGA-II) and a hybrid non-dominated sorting genetic algorithm with particle swarm optimization (HNSGAII-PSO) are developed to tackle instances with medium and large scales and to obtain approximate Pareto fronts

4. 1. NSGA-II Algorithm

The NSGA-II algorithm (43) is a popular evolutionary algorithm that is widely used in solving multi-objective optimization problems (44, 45). This algorithm can provide a suitable set of Pareto solutions for solving multi-objective problems by using the elitism mechanism and taking the crowding distance of the solutions into account.

In the proposed NSGA-II algorithm, each chromosome represents a solution from the problem-solving space. In this algorithm, by decoding each chromosome, the cutting patterns assigned to each machine, the sequence of patterns on each machine, and the speed level of the machine for processing each pattern are determined, and using this information, the objective functions of total completion time and total energy consumption can be calculated for each chromosome. After calculating the value of the objective functions and determining the fitness for all chromosomes in the population, the sorting of chromosomes is done. This process begins with performing paired comparisons and calculating the number of times each solution is dominated in order to form Pareto fronts and determine the rank of each solution. Then the members placed in each Pareto front are sorted based on the crowding distance metric. Next, offspring are produced by selecting parents and using crossover and mutation operators. The offspring obtained from the mutation and crossover operators are added to the population, creating a larger population called R_t. After calculating the objective functions for the generated offspring, rank and crowding distance are determined for each member of the R_t population, and based on them, the R_t population is sorted. The sorting is done as follows: first, the members are sorted by rank and in ascending order so that the solutions with lower ranks are placed at the beginning of the list. Then, among the members with the same rank, another sorting is done based on the crowding distance and in a descending manner, so that the solution with the greatest crowding distance occupies a higher position among the members of the same rank. Finally, the elitist strategy is used to form the new generation, in which members with a higher position are selected from the R_t population in a number equal to the size of the population. This process continues until the termination condition of the algorithm is established. The proposed NSGA-II algorithm stops when a certain number of iterations is reached, and all the solutions in the first Pareto front are presented as the output of the algorithm.

4. 1. 1. Solution Representation

In order for the solution algorithm to establish a logical relationship between the problem space and the search space, the solution properties should be represented by a string of symbols. Each chromosome of the NSGA-II algorithm is represented by a two-row matrix, where the length of each row is n (n: number of cutting patterns). The first row of the matrix contains the permutation of numbers 1 to n, so that the members of the set $\{1,2,3,\dots,n\}$ represent the cutting pattern number. The second row of the matrix shows the speed level of the machine for processing each of the cutting patterns, where each gene is coded randomly with values from the set $\{1,2,3,\dots,s\}$. The cutting patterns assigned to each machine and the sequence of patterns on each machine are determined by decoding each chromosome. Extracting the required information is as follows: the pattern placed in the first gene of the first row of the matrix is assigned to the first machine and processed at the speed determined in the first gene of the second row of the matrix. This process continues until every machine is assigned a pattern. Then, the next pattern is assigned to the first machine that finishes processing the previously assigned pattern and is processed at the determined speed. This process continues until all patterns are assigned. For example, consider a problem with two machines and five cutting patterns. Each machine has two levels of slow speed (level 1) and fast speed (level 2) to process the assigned patterns. It is assumed that each pattern needs 2 and 1 units of time for processing at the machine speed levels of 1 and 2, respectively. Figure 5 shows a chromosome for the mentioned example. After decoding this solution, it becomes clear that pattern 3 is assigned to the first machine and is processed with a speed level of 2 in 1 time unit. Pattern 1 is assigned to the second machine and processed at speed level 1 in 2 time units. Since processing pattern 3 is finished faster by machine 1, pattern 2 is assigned to the first machine and is processed at speed level 1 in 2 time units. Next, patterns 5 and 4 are assigned to machines 2 and 1, and are processed at speed levels 1 and 2, respectively. In the proposed NSGA-II algorithm, members of the initial population are generated randomly and according to the presented representation for each solution.

4. 1. 2. Selection In the proposed NSGA-II algorithm, parent selection is done using the standard binary tournament selection strategy.

4. 1. 3. Crossover In the proposed NSGA-II algorithm, the double-point crossover operator is used. In view of the permutation of numbers 1 to n in the first row of the chromosome, the permutation of numbers may not be established in the first row of each of the generated offspring. Therefore, using the partially mapped crossover (PMX) approach, the required columns are moved, and the necessary corrections are made to establish the permutation of the numbers in the first row of the generated offspring. Figure 6 depicts the double-point crossover.

3	1	2	5	4
2	1	1	1	2

Figure 5. An example of solution representation in the NSGA-II algorithm

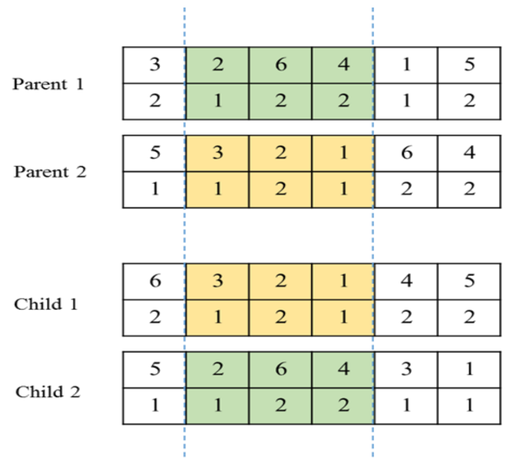


Figure 6. The crossover operator in the NSGA-II algorithm

4. 1. 4. Mutation In the proposed NSGA-II algorithm, Gaussian mutation and swap mutation operators are used.

The Gaussian mutation operator is applied to the second row of the candidate chromosome. The Gaussian mutation operates as follows: A random number between 0 and 1 is generated for each gene from the second row of the candidate chromosome. If this number is less than the gene's mutation rate, the mutation operator with $N(0,1)$ distribution is applied to that gene. Thus, the value of that gene may change. According to the chromosome's representation, the value of each gene in the second row of the chromosome can be one of the numbers in the set $S=\{1,2,3,\dots,s\}$. If, after applying the mutation operator, the value of the gene is greater than the largest value of the set S or less than the smallest value of that set, then we consider the value of the mentioned gene to be equal to the largest and smallest members of the set S, respectively. In Figure 7, the Gaussian mutation operator has been applied to three genes from the parent chromosome, and in the generated offspring, only the value of one of the selected genes has changed.

The swap mutation is executed as follows: Two columns are randomly chosen from the candidate chromosome, and they are then exchanged with one another Figure 8. Based on the results obtained from the numerical experiments, the Gaussian mutation and swap mutation operators are applied to the candidate chromosomes with probabilities of 0.8 and 0.2, respectively.

4. 2. HNSGAII-PSO Algorithm The proposed hybrid algorithm (HNSGAII-PSO) is a combination of the NSGA-II algorithm and the PSO algorithm. In the HNSGAII-PSO algorithm, each chromosome determines the speed level of the machine for processing each pattern, and this information can be used to calculate the

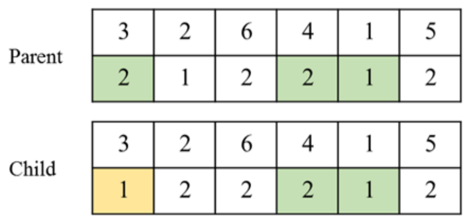


Figure 7. The Gaussian mutation operator in the NSGA-II algorithm

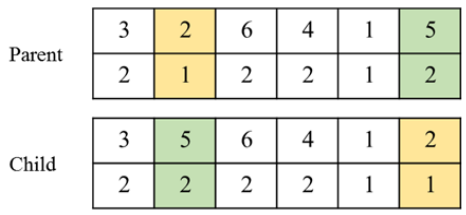


Figure 8. The swap mutation operator in the NSGA-II algorithm

objective function of the total energy consumption for that chromosome. In other words, each chromosome represents a region of the problem-solving space, so that all solutions in this region have the same total energy consumption. To evaluate the quality of each chromosome, the best solution from the region covered by that chromosome is considered. In the investigated problem, the best solution in the region covered by each chromosome is the one that minimizes the total completion time. Using the PSO algorithm, a global search is done to find the best solution in this region. Therefore, in each iteration of the hybrid algorithm, the PSO algorithm is executed once for each chromosome in the population. Each particle of the PSO algorithm specifies the assignment of patterns to machines as well as the sequence of patterns on each machine, and using this information and taking into account the speed level of the machine to process each pattern, the total completion time can be calculated for each particle of the PSO algorithm. Based on the output of the PSO algorithm, the assignment of patterns to machines, the sequence of patterns on each machine, and the objective function of the total completion time for each chromosome of the HNSGAII-PSO algorithm are determined. The main structure and the process of obtaining the approximate Pareto front in the HNSGAII-PSO algorithm are similar to the NSGA-II algorithm.

4. 2. 1. PSO Algorithm The PSO algorithm is a population-based algorithm (46) and is widely used in solving scheduling problems (47, 48). Within this algorithm, each solution is conceptualized as a particle, each possessing its own position and velocity. The position of a particle facilitates the identification of a feasible solution to the problem under investigation. The fitness value of each particle determines the quality of the corresponding solution. During each iteration of the algorithm, each particle endeavors to locate a new position based on its previous experiences and the position of the particle exhibiting the most favorable fitness value, thereby striving to enhance its own fitness value (49). In the HNSGAII-PSO algorithm proposed in this paper, for each chromosome, the PSO algorithm is executed once. To update the speed and position of each particle in the PSO algorithm, relations (25) and (26) are used, respectively.

$$v_j^{t+1} = \omega v_j^t + r_1 \times c_1 \times \left(pbest_j - x_j^t\right) + r_2 \times c_2 \times \left(gbest - x_j^t\right) \tag{25}$$

$$x_j^{t+1} = x_j^t + v_j^{t+1} \tag{26}$$

where:

v_j	Velocity of particle j.
x_j	Position of particle j.
$pbest_j$	The best position ever visited by particle j.
$gbest$	The best position ever visited by swarm.
ω	Inertia coefficient that controls the velocity of the particle.
c_1	The learning coefficient considers the current particle's attraction to its previous best position.
c_2	The learning coefficient considers the current particle's attraction to the previous best position of the swarm
r_1, r_2	Random numbers are generated from the uniform interval [0,1].

4. 2. 2. Solution Representation In the proposed HNSGAII-PSO algorithm, each chromosome contains a vector of length n, where n is the number of cutting patterns. The counter of each chromosome gene indicates the cutting pattern number, and the genes of each chromosome are randomly coded with values from the set {1, 2, 3, ..., s} that determine the speed level of the machine for processing the relevant cutting patterns, which can be used to calculate the objective function of the total energy consumption for each chromosome. In other words, each chromosome represents a region of the problem-solving space where all solutions in this region have the same total energy consumption. To evaluate the quality of each chromosome, the best solution from the region covered by that chromosome is considered. In the investigated problem, the best solution in the region covered by each chromosome is the one that minimizes the total completion time. The best solution in this region is determined using the PSO algorithm. Therefore, in each iteration of the hybrid algorithm, the PSO algorithm is executed once for each chromosome in the population.

Each particle in the PSO algorithm is represented as a matrix with one row and n columns, where n is the number of cutting patterns. A random real number between 0 and 1 is placed in each column of this matrix. For each particle, in order to determine the cutting patterns assigned to each machine and the sequence of patterns on each machine, first, through an appropriate mechanism, the real numbers must be converted into integers that represent the cutting patterns. To do this, we used the sorting-based method (50). Therefore, we sort the real numbers of the initial matrix in ascending order to form a new matrix. Then, in the new matrix, any real number is replaced by the counter of the corresponding column in the initial matrix. Accordingly, the new matrix contains a permutation of integers from 1 to n, and each number represents a cutting pattern. Now, by decoding the created matrix, the patterns assigned to each machine and the sequence of patterns on each machine can be determined. Using this information and taking into account the properties of the relevant chromosome (the speed level of the machines), the function of the total completion time for each particle of the PSO algorithm can be calculated. The output of the PSO algorithm determines the assignment of patterns to machines, the sequence of patterns on each machine, and the objective function of the total completion time for each chromosome of the HNSGAII-PSO algorithm.

Each particle of the PSO algorithm is decoded as follows: the pattern placed in the first column of the matrix is assigned to the first machine and processed at the corresponding speed. The pattern placed in the second column is assigned to the second machine and processed at the corresponding speed, and this process continues until each machine is assigned a pattern. Then, the next pattern is assigned to the first machine that finishes processing the previously assigned pattern, and this process continues until all patterns are assigned.

For example, consider a problem with two machines and five cutting patterns. Each machine has two levels of slow speed (level 1) and fast speed (level 2) to process the patterns assigned to it, so that the faster the machine processes, the more energy it consumes. It is assumed that each pattern needs 2 units and 1 unit of time to be processed at machine speed levels 1 and 2, respectively. Figure 9 is a chromosome for the mentioned example, which shows a region of the solution space. For all solutions in this region, cutting patterns numbers 1, 2, 3, 4, and 5 are processed at machine speed levels 2, 1, 1, 2, and 1, respectively. Figure 10 is a particle of the PSO algorithm and its decoding, which shows a solution of the region covered by Figure 9's chromosome. After decoding this particle, it becomes clear that pattern 3 is assigned to the first machine and is processed with a speed level 1 in 2 time units; pattern 1 is assigned to the second machine and processed at a speed level 2 in 1 time unit. Since the processing of pattern 1 is completed faster by machine 2, pattern 2 is assigned to the second machine and is processed at speed level 1 in 2 time units. Next, patterns 5 and 4 are assigned to machines 1 and 2, and they are processed at speed levels 1 and 2, respectively.

In the proposed HNSGAII-PSO algorithm, each chromosome from the initial population is randomly generated according to the representation provided for each solution. The PSO algorithm is executed once for every member of the initial population. The initial population members of the PSO algorithm are also randomly generated.

4. 2. 3. Selection

Parent selection in the proposed HNSGAII-PSO algorithm is done using the standard binary tournament selection strategy.

4. 2. 4. Crossover

The double-point crossover operator is employed in the proposed HNSGAII-PSO algorithm. Figure 11 shows the double-point crossover.

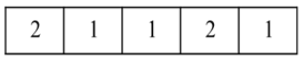


Figure 9. An example of solution representation in the HNSGAII-PSO algorithm

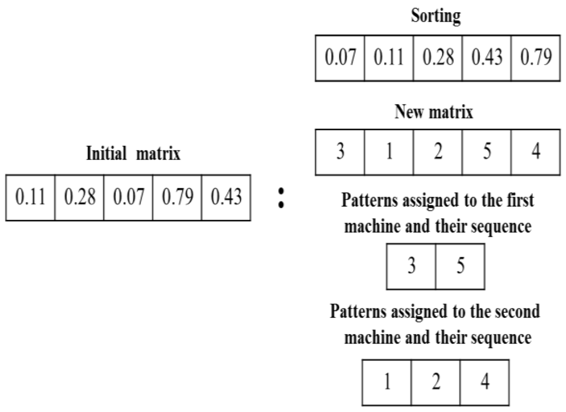

Sorting
0.07 0.11 0.28 0.43 0.79
New matrix
3 1 2 5 4
Initial matrix
0.11 0.28 0.07 0.79 0.43
Patterns assigned to the first machine and their sequence
3 5
Patterns assigned to the second machine and their sequence
1 2 4

Figure 10. A solution of the region covered by the chromosome in Figure 9 and its decoding

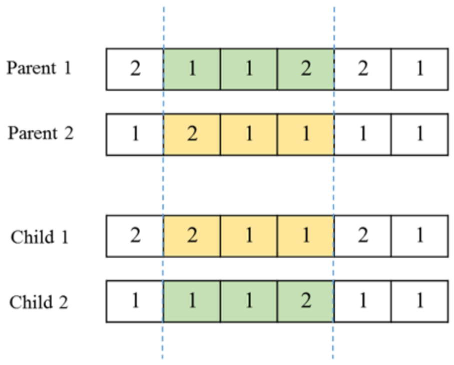


Figure 11. The crossover operator in the HNSGAII-PSO algorithm

4. 2. 5. Mutation In the proposed hybrid algorithm, the Gaussian mutation operator is used. In this algorithm, according to the representation defined for the chromosome, all chromosomes in the population are candidates for the generation of offspring through the mutation operator. Next, for each gene on the candidate chromosome, a random number between 0 and 1 is generated. If this number is less than the mutation rate of the gene, the mutation operator with the standard normal distribution $N(0,1)$ is applied to that gene. Accordingly, the value of that gene may change in the generated offspring. Figure 12 shows how to produce offspring from the parent using the Gaussian mutation operator. In this figure, the mutation operator has been applied to three genes from the parent chromosome, and in the offspring generated, only the value of one of the selected genes has changed**.**

5. COMPUTATIONAL EXPERIMENT

In this section, in order to evaluate the performance of the mathematical models and solution methods presented, numerical experiments are performed using randomly generated test instances. The process of generating the necessary data for the instances is explained in detail. Given the bi-objective nature of the problem, performance measures used to evaluate the suggested solution methods are introduced. Moreover, the parameters of the proposed solution algorithms are tuned using the Taguchi method. Finally, the performance of the proposed mathematical models and solution methods is evaluated by solving small and large-scale instances. In this paper, to optimally solve small-scale instance problems using the AUGMECON method, GAMS 28.2.0 software is used, and the HNSGAII-PSO and NSGA-II algorithms are coded in the Visual C# environment. All experiments were performed on a personal computer with 4 GB of RAM and an Intel Core i5-2410M 2.30 GHz CPU.

5. 1. Data Generation The size of instance problems is defined as $[I, J, M, \delta]$, where I represents the number of jobs, J denotes the number of cutting patterns, M shows the number of machines, and δ represents the density of the job-pattern matrix. In this paper, random test instances are generated as follows (18): To determine the required workload of each cutting pattern (w_j), a number is generated randomly from a uniform distribution in the interval [5, 50]. The energy consumption rate of the machine for processing each cutting pattern per unit of time (π_j) is randomly selected from a uniform distribution in the interval [4, 18]. Therefore, the energy consumption rate of the machine at the speed level s to process pattern j per unit of time is determined from Equation 27. In this paper, $\alpha = 3$ and for machine processing speed levels, $S = 4$ and $v_s \epsilon \{0.75, 1, 1.25, 1.5\}$ are considered. The time and energy required to process each cutting pattern at varying levels of machine speed are calculated using relations 28 and 29, respectively.

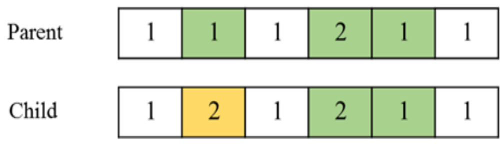


Figure 12. The mutation operator in the HNSGAII-PSO algorithm

$$\pi_{js} = \pi_j v_s^{\alpha} \quad , \quad \alpha > 1 \tag{27}$$

$$p_{js} = \frac{w_j}{v_s} \tag{28}$$

$$e_{js} = \pi_{js} p_{js} \tag{29}$$

To randomly generate the job-pattern matrix, first, the density δ of the matrix must be determined. Equation 30 calculates the value of δ for the job-pattern matrix. In this equation, $|N_i|$ is the number of cutting patterns that include small pieces (items) of job i. i and j also represent the number of jobs and the number of cutting patterns, respectively. In determining the minimum value of δ for test instances, a noteworthy point to consider is that each cutting pattern contributes to completing at least one job, and the pieces related to each job are placed in at least one cutting pattern. Therefore, the minimum possible value for δ is equal to max $\left(\frac{1}{i}, \frac{1}{j}\right)$, where i is the number of jobs and j is the number of cutting patterns, and this relation should be considered to determine the value of δ in small-scale instances. In medium- and large-scale instances, we consider three values of 20%, 30%, and 40% for δ (33). After determining the value of δ, we generate the job-pattern matrix at random so that each cutting pattern contributes to the completion of at least one job and each job requires at least one cutting pattern to complete itself.

$$\delta = \frac{\sum_i |N_i|}{i \times j} \tag{30}$$

5. 2. Performance Measures The Pareto solution set is the output of solving multi-objective problems, whose quality and diversity are evaluated. In this paper, to evaluate the results obtained from the solution algorithms, four performance measures are used as follows:

Number of Pareto solutions (NPS): This metric specifies the number of non-dominated solutions obtained from the solution algorithm. Based on this metric, the greater the number of these solutions, the more efficient the algorithm (21, 51).

Diversity of distribution (D_1): This performance measure shows that the existing solutions in the Pareto front are uniformly placed next to each other. This metric is calculated using Equation 31. In this equation, $|N|$ expresses the number of non-dominated solutions, d_i is the Euclidean distance between consecutive solutions, and $\bar{d}$ is the average of d_i. For a solution algorithm, the lower the value of the D_1 metric, the more efficient that algorithm is (21, 51).

$$D_1 = \sum_{i=1}^{|N|-1} \frac{|d_i - \bar{d}|}{|N|-1} \tag{31}$$

Spacing (D_2): This measure is an extension of the previous metric and is obtained from Equation 32. In this equation, $|N|$ shows the number of non-dominated solutions. The values of d_i and $\bar{d}$ are obtained via Equations 33 and 34, respectively. For a solution algorithm, the lower the value of the D_2 metric, the better the performance of that algorithm (22, 52).

$$D_2 = \left(\frac{1}{|N|}\sum_{i=1}^{|N|}\left(d_i - \bar{d}\right)^2\right)^{1/2} \tag{32}$$

$$d_i = \min_{k\epsilon N, k\neq i} \sum_{m=1}^{M}\left|f_m^i - f_m^k\right| \tag{33}$$

$$\bar{d} = \sum_{i=1}^{|N|-1} \frac{d_i}{|N|-1} \tag{34}$$

Mean ideal distance (MID): This performance measure calculates the average Euclidean distance of the ideal solution from the Pareto front obtained by the solution algorithm. In this paper, the best possible value for each of the objective functions obtained by different algorithms is considered the ideal solution. Equation 35 is used to calculate MID. In this equation, $|N|$ is the number of non-dominated solutions, and C_i represents the Euclidean distance of each member of the Pareto front from the ideal point, which is calculated through Equation 36. An algorithm with a lower MID metric has better performance (51).

$$MID = \frac{1}{|N|}\sum_{i=1}^{|N|} C_i \tag{35}$$

$$C_i = \sqrt{(f_{1i} - f_1^*)^2 + \cdots + (f_{mi} - f_m^*)^2} \tag{36}$$

5. 3. Parameter Tuning

Metaheuristic algorithms possess inherent parameters, and assigning appropriate values to these parameters can substantially enhance the quality of the obtained results. This paper employs the Taguchi method to optimize the parameters of the developed algorithms. The Taguchi method utilizes orthogonal arrays, which are standardized arrays that enable the execution of a limited number of experiments while retaining comprehensive information on all factors influencing the performance of the algorithms (53). The HNSGAII-PSO factors are: number of population (N), number of generations (G), crossover rate (P_c), mutation rate of the gene (P_{mg}), number of population for the PSO algorithm (N_{PSO}), number of generations for the PSO algorithm (G_{PSO}), inertia coefficient (ω), and learning coefficients (c_1 , c_2) and The NSGA-II factors are: number of population (N), number of generations (G), crossover rate (P_c), mutation rate (P_m) and mutation rate of the gene (P_{mg}). Tables 3 and 4 show the considered levels for the factors of the HNSGAII-PSO and NSGA-II algorithms for small-scale problems and medium- and large-scale problems, respectively.

TABLE 3. Algorithm parameters and their levels for small-scale problems

Algorithm	Parameters	Symbol	Level
HNSGAII-PSO	N	A	100 – 150 – 200
	G	B	50 – 80 – 100
	P_c	C	0.6 – 0.7 – 0.8
	P_{mg}	D	0.2 – 0.3 – 0.4
	N_{PSO}	E	2 – 3 – 4
	G_{PSO}	F	1 – 2 – 3
	ω	G	0.5 – 0.75 – 1
	c_1	H	1 – 1.5 – 2
	c_2	J	1 – 1.5 – 2
NSGA-II	N	A	100 – 150 – 200
	G	B	50 – 80 – 100
	P_c	C	0.5 – 0.6 – 0.7
	P_m	D	0.5 – 0.6 – 0.7
	P_{mg}	E	0.5 – 0.6 – 0.7

TABLE 4. Algorithm parameters and their levels for medium- and large-scale problems

Algorithm	Parameters	Symbol	Level
HNSGAII-PSO	N	A	300 – 350 – 400
	G	B	100 – 150 – 200
	P_c	C	0.6 – 0.7 – 0.8
	P_{mg}	D	0.2 – 0.3 – 0.4
	N_{PSO}	E	2 – 3 – 4
	G_{PSO}	F	1 – 2 – 3
	ω	G	0.5 – 0.75 – 1
	c_1	H	1 – 1.5 – 2
	c_2	J	1 – 1.5 – 2
NSGA-II	N	A	400 – 600 – 800
	G	B	300 – 400 – 500
	P_c	C	0.5 – 0.6 – 0.7
	P_m	D	0.5 – 0.6 – 0.7
	P_{mg}	E	0.5 – 0.6 – 0.7

Considering the number of parameters and determined levels, the right orthogonal array for both algorithms is L_{27}, which includes 27 experiments. Then, for each algorithm, all experiments are conducted on an instance problem considering various combinations of parameter levels, and the obtained results are recorded for performance measures. It's important to note that each experiment was repeated five times, and the average of the results was considered. Since the Taguchi method only accepts one value as a response for each experiment, the results obtained for the performance measures are unscaled using the relative percentage deviation (RPD) method. then for each experiment, the weighted average of the relevant unscaled performance measures is calculated as a combined function (CF) using equation 37 and considered as the result of that experiment (51).

$$CF = \frac{NPS + D_1 + D_2 + 2MID}{5} \tag{37}$$

Figures 13, 14, 15, and 16 show the results obtained from the Taguchi method for small-scale problems and medium- and large-scale problems, respectively. Based on these results, the parameter values of each of the NSGA-II and HNSGAII-PSO algorithms are presented in Tables 5 and 6, respectively.

5. 4. Evaluation of Solution Algorithms for Small-Scale Instances

In this sub-section, the performance of the AUGMECON method, the NSGA-II algorithm, and the HNSGAII-PSO algorithm are compared for solving small-scale instances. The results of solving 30 instances using the three methods mentioned are shown in Table 7. In this table, column 1 shows the instance number, and column 2 shows the instance size. The results of solving the instances with the AUGMECON method are presented for the position-based model and the sequence-based model in columns 3 and 4, respectively. The AUGMECON method for both presented models has the ability to obtain the optimal Pareto front for 26 instances; however, for 4 instances, this method is not able to solve the sequence-based model

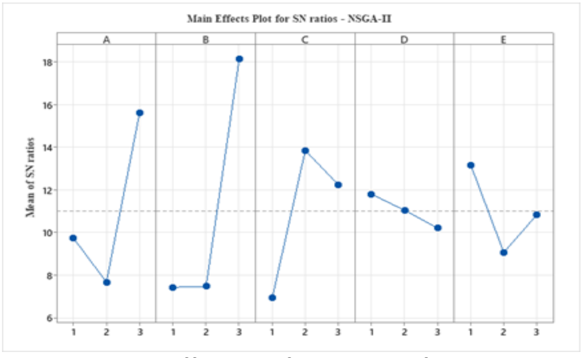


Figure 13. Main effects plot for S/N ratios for the NSGA-II algorithms: small-scale problems

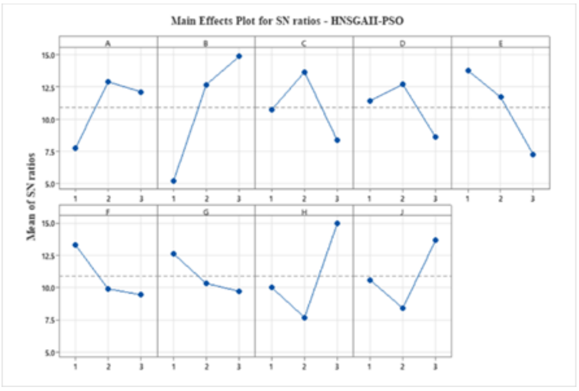


Figure 14. Main effects plot for S/N ratios for the HNSGAII-PSO algorithms: small-scale problems

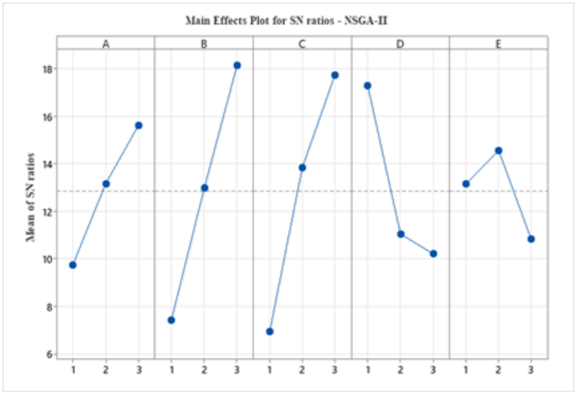


Figure 15. Main effects plot for S/N ratios for the NSGA-II algorithms: medium- and large-scale problems

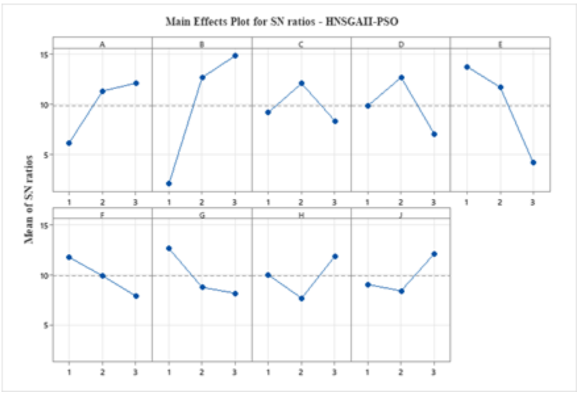


Figure 16. Main effects plot for S/N ratios for the HNSGAII-PSO algorithms: medium- and large-scale problems

TABLE 5. Parameter tuning results for the NSGA-II algorithm

	Parameters				
	N	G	P_c	P_m	P_{mg}
Small-scale	200	100	0.6	0.5	0.5
Medium and large-scale	800	500	0.7	0.5	0.6

TABLE 6. Parameter tuning results for the HNSGAII-PSO algorithm

	Parameters								
	N	G	P_c	P_{mg}	N_{PSO}	G_{PSO}	ω	c_1	c_2
Small-scale	150	100	0.7	0.3	2	1	0.5	2	2
Medium and large-scale	400	200	0.7	0.3	2	1	0.5	2	2

TABLE 7. Results of solving small-scale instances using the AUGMECON method, HNSGAII-PSO algorithm, and NSGA-II algorithm

No.	$[I,J,M,\delta]$	AUGMECON				HNSGAII-PSO					NSGA-II				
				$t(s)$											
		NPS	MID	position-based model	sequence-based model	NPS	D_1	D_2	MID	$t(s)$	NPS	D_1	D_2	MID	$t(s)$
1	2-4-2-50%	21	572.48	4.08	6.68	21	43.88	17.59	572.48	**0.73**	21	43.88	17.59	572.48	**0.73**
2	2-4-2-75%	24	947.25	5.54	6.78	24	65.61	60.42	947.25	**0.87**	24	65.61	60.42	947.25	**0.87**
3	2-5-2-50%	36	1581.3	62.36	183.26	36	69.01	48.09	1581.3	**0.93**	36	69.01	48.09	1581.3	**0.93**
4	2-5-2-60%	45	1635	101.35	290.27	45	53.94	52.92	1635	**1.23**	45	53.94	52.92	1635	**1.23**
5	3-4-2-40%	21	822.58	20.22	23.77	21	47.88	61.18	822.58	**1.59**	21	47.88	61.18	822.58	**1.59**
6	3-4-2-60%	25	786.76	18.65	21.55	25	38.19	44.54	786.76	**1.6**	25	38.19	44.54	786.76	**1.6**
7	3-4-3-50%	30	970.58	24.29	18.38	30	43.41	51.27	970.58	**1.68**	30	43.41	51.27	970.58	**1.68**
8	3-4-3-75%	23	959.61	13.58	11.54	23	47.31	58.02	959.61	**1.69**	23	47.31	58.02	959.61	**1.69**
9	3-5-2-40%	38	722.14	252.99	437.14	38	25.16	22.23	722.14	**1.72**	38	25.16	22.23	722.14	**1.72**
10	3-5-2-60%	38	614.85	93.64	179.86	38	35.73	41.67	614.85	**1.76**	38	35.73	41.67	614.85	**1.76**
11	3-5-3-40%	35	688.7	190.09	286.29	35	35.29	23.92	688.7	**1.76**	35	35.29	23.92	688.7	**1.76**
12	3-5-3-60%	40	732.17	146.49	211.24	40	26.01	19.71	732.17	**1.78**	40	26.01	19.71	732.17	**1.78**
13	4-4-2-25%	30	540.82	26.53	24.68	30	41.38	59.46	540.82	**1.61**	30	41.38	59.46	540.82	**1.61**
14	4-4-2-50%	39	756.27	37.95	39.74	39	22.89	24.71	756.27	**1.65**	39	22.89	24.71	756.27	**1.65**
15	4-4-2-75%	31	665.67	33.60	32.19	31	31.70	35.71	665.71	1.67	31	31.70	35.71	665.71	1.67
16	4-4-3-30%	34	915.67	25.56	28.75	34	26.58	30.79	915.67	**1.73**	34	26.58	30.79	915.67	**1.73**
17	4-4-3-50%	29	947.81	55.81	61.64	29	38.59	22.35	947.81	**1.64**	29	38.59	22.35	947.81	**1.64**
18	4-5-2-30%	48	868.82	118.48	124.83	48	37.85	31.46	868.82	**1.78**	48	37.85	31.46	868.82	**1.78**
19	4-5-2-40%	67	1011.1	468.33	789.60	64	18.97	16.04	1046	1.76	64	18.97	16.04	1046	1.76
20	4-5-3-30%	53	719.08	1743.38	1812.21	53	17.08	10.82	719.08	**1.77**	53	17.08	10.82	719.08	**1.77**
21	4-5-3-40%	66	1024.8	1927.79	1967.18	66	20.17	19.01	1024.8	**1.78**	66	20.17	19.01	1024.8	**1.78**
22	4-6-2-25%	55	614.16	5293.29	11035.73	55	12.04	14.58	614.16	**1.79**	55	12.04	14.58	614.16	**1.79**
23	4-6-2-50%	58	734.96	2083.45	29303.34	63	13.54	13.98	766.23	1.79	63	13.54	13.98	766.23	1.79
24	4-6-2-75%	65	698.25	2359.10	33179.49	65	12.17	14.71	699.64	1.81	65	12.17	14.71	699.64	1.81
25	4-6-3-30%	31	891.77	8876.68	11499.63	30	46.11	56.1	893.38	1.77	30	46.11	56.1	893.38	1.77
26	4-6-3-40%	53	763.95	13521.75	28961.49	57	35.82	29.54	792.65	1.78	57	35.82	29.54	792.65	1.78
27	4-6-3-50%	44	911.54	19492.31	-	42	43.13	45.92	923.75	1.8	42	43.13	45.92	923.75	1.8
28	5-7-2-20%	67	1174.23	12548.52	-	73	39.46	42.96	1227.62	1.78	73	39.46	42.96	1227.62	1.78
29	5-7-2-30%	61	938.47	21649.39	-	59	41.44	39.12	932.54	1.79	59	41.44	39.12	932.54	1.79
30	5-7-2-40%	71	1480.2	34394.65	-	64	33.01	27.84	1518.6	1.82	64	33.01	27.84	1518.6	1.82

The running time of the algorithm that obtained the optimal Pareto front is shown with a bold value and find the optimal Pareto front in 36000 seconds. Based on the presented results, the position-based model has better performance in terms of optimal solution time using the AUGMECON method compared to the sequence-based model. To validate the approximate Pareto solutions obtained by the NSGA-II and HNSGAII-PSO algorithms, a comparison with the AUGMECON method was conducted. The proposed NSGA-II and HNSGAII-PSO algorithms were employed to solve small-scale instances. The results of solving small-scale instances using the HNSGAII-PSO and NSGA-II algorithms are presented in columns 5 and 6 of Table 7, respectively. An examination of Table 7 revealed that the NSGA-II and HNSGAII-PSO algorithms successfully identified the optimal Pareto front in 19 instances, demonstrating superior computational efficiency compared to the AUGMECON method. In the remaining instances, the solutions obtained by both algorithms exhibited close proximity to the optimal Pareto solutions identified by the AUGMECON method. Consequently, based on these findings, it is evident that the NSGA-II and HNSGAII-PSO algorithms constitute are valid and effective approaches for solving instances in a reasonable time.

To comprehensively evaluate Pareto dominance and compare the solutions obtained by the AUGMECON method and the proposed algorithms, the MID metric was calculated for all three methods, and the results are presented in Figure 17. The analysis of small-scale instances revealed that the solutions generated by the proposed algorithms are comparable to the optimal Pareto fronts when compared to the AUGMECON method. Notably, the NSGA-II and HNSGAII-PSO algorithms provide acceptable results in significantly less time than the AUGMECON method. Figure 18 demonstrates the CPU time for solving small-scale instances by all three methods. As observed, with increasing instance sizes, the CPU time for the AUGMECON method escalates dramatically. Therefore, based on the comprehensive comparison of Pareto solutions, it can be concluded that NSGA-II and HNSGAII-PSO are viable and efficient algorithms for solving small-scale instances within a reasonable computational time frame.

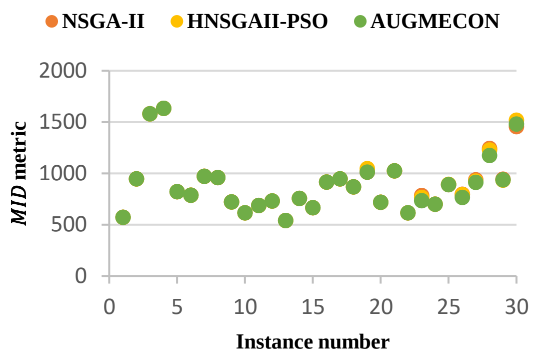


Figure 17. MID metric obtained through the NSGA-II, HNSGAII-PSO and AUGMECON methods for small-scale instances

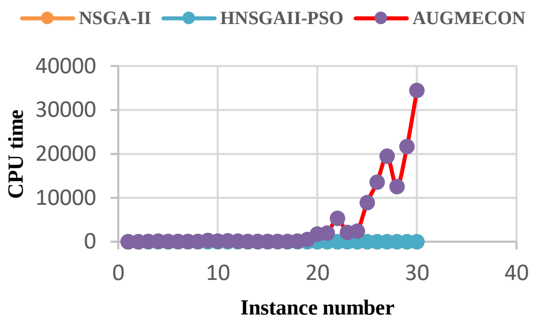


Figure 18. CPU time for solving small-scale instances by the NSGA-II, HNSGAII-PSO and AUGMECON methods

5. 5. Evaluation of Solution Algorithms for Medium- and Large -Scale Instances

Based on the results presented in Table 7, by increasing the scale of the problem, the computation time increases. For example, in instance #30 with size [5,7,2,40%], the AUGMECON method needs 34395 seconds to obtain the optimal Pareto front. Therefore, HNSGAII-PSO and NSGA-II algorithms are used to solve medium- and large-scale instances, and their obtained results are compared. To ensure a fair comparison, we have set the running times of both algorithms to be equal. This means that the time considered when comparing two algorithms is equal to the execution time of the algorithm that finishes faster in each instance. Each of the two algorithms is executed five times for each instance, and the average values derived for the performance measures are recorded. The outcomes of applying the NSGA-II and HNSGAII-PSO algorithms to solve 30 medium- and large-scale instances are presented in Table 8. Figure 19 shows the Pareto fronts generated by both algorithms for problem instance #55. Based on the obtained results, the HNSGAII-PSO algorithm performs better in the NPS, D_1, and D_2 criteria, which means that this algorithm can generate a greater number of non-dominated solutions with more diversity. The evaluation of the MID criterion demonstrates that the HNSGAII-PSO algorithm performs better, indicating that it can produce non-dominated solutions with better convergence**.**

In order to further analyze the performance of the HNSGAII-PSO and NSGA-II algorithms in solving instance problems, paired samples t-test is performed for every performance measure. Statistical results for different measures are stated in Table 9. The mean and

TABLE 8. Results of solving medium- and large-scale instances using the HNSGAII-PSO algorithm and the NSGA-II algorithm

No.	$[\boldsymbol{I},\boldsymbol{J},\boldsymbol{M},\boldsymbol{\delta}]$	**Time (s)**	$\boldsymbol{NPS}$		$\boldsymbol{D_2}$		$\boldsymbol{D_1}$		$\boldsymbol{MID}$	
			HNSGAII-PSO	**NSGA-II**	**HNSGAII-PSO**	**NSGA-II**	**HNSGAII-PSO**	**NSGA-II**	**HNSGAII-PSO**	**NSGA-II**
31	15-30-5-20%	25	59.60	47.03	59.60	47.03	59.60	47.03	4878.05	4801.21
32	30-30-5-30%	33	48.81	32.22	48.81	32.22	48.81	32.22	5065.85	5046.76
33	50-30-5-40%	42	22.38	42.54	22.38	42.54	22.38	42.54	7909.23	7894.54
34	30-50-7-20%	47	84.16	63.73	84.16	63.73	84.16	63.73	8521.25	8492.57
35	50-50-7-30%	56	28.01	52.37	28.01	52.37	28.01	52.37	9757.75	9737.86
36	80-50-7-40%	78	31.19	58.38	31.19	58.38	31.19	58.38	11026.79	11284.65
37	50-80-10-20%	81	38.53	77.22	38.53	77.22	38.53	77.22	12357.81	13701.54
38	80-80-10-30%	111	33.78	63.41	33.78	63.41	33.78	63.41	12549.56	12783.00
39	100-80-10-40%	128	40.47	73.42	40.47	73.42	40.47	73.42	13850.82	15555.38
40	80-100-12-20%	131	39.75	95.05	39.75	95.05	39.75	95.05	15473.07	16454.84
41	100-100-12-30%	152	41.16	71.94	41.16	71.94	41.16	71.94	13842.15	14764.67
42	150-100-12-40%	204	39.80	90.48	39.80	90.48	39.80	90.48	15940.74	16934.13
43	100-150-15-20%	223	54.25	103.11	54.25	103.11	54.25	103.11	17261.59	19042.25
44	150-150-15-30%	278	53.49	118.75	53.49	118.75	53.49	118.75	19587.96	21591.61
45	200-150-15-40%	336	58.67	111.19	58.67	111.19	58.67	111.19	20524.40	21737.06
46	150-200-18-20%	347	63.72	123.72	63.72	123.72	63.72	123.72	19943.05	23042.00
47	200-200-18-30%	419	66.89	153.43	66.89	153.43	66.89	153.43	23681.41	28178.78
48	250-200-18-40%	494	65.76	158.35	65.76	158.35	65.76	158.35	24610.11	28376.25
49	200-250-20-20%	504	86.13	145.93	86.13	145.93	86.13	145.93	24744.62	28052.78
50	250-250-20-30%	599	79.92	166.71	79.92	166.71	79.92	166.71	24539.97	29819.36
51	300-250-20-40%	698	78.02	159.17	78.02	159.17	78.02	159.17	25810.26	30246.06
52	250-300-25-20%	718	70.41	148.80	70.41	148.80	70.41	148.80	21604.30	28446.83
53	300-300-25-30%	825	95.05	177.20	95.05	177.20	95.05	177.20	26398.03	32207.54
54	400-300-25-40%	1050	107.09	212.11	107.09	212.11	107.09	212.11	29752.38	35398.91
55	300-400-30-20%	1100	98.86	195.93	98.86	195.93	98.86	195.93	27888.72	31671.08
56	400-400-30-30%	1290	124.71	220.91	124.71	220.91	124.71	220.91	32467.22	37604.07
57	500-400-30-40%	1690	119.76	232.55	119.76	232.55	119.76	232.55	33281.22	41192.72
58	400-500-35-20%	1760	136.67	198.92	136.67	198.92	136.67	198.92	33257.67	40642.41
59	450-500-35-30%	1984	138.73	245.32	138.73	245.32	138.73	245.32	35466.96	41484.99
60	500-500-35-40%	2145	139.89	321.04	139.89	321.04	139.89	321.04	38320.90	47772.89
Average value			**350.53**	302.67	**90.59**	156.25	**71.52**	132.03	**20343.80**	23465.29

The algorithm with better performance in each measure is shown with a bold value

SD columns show the mean value and standard deviation of the performance measures, respectively. Based on the results presented in Table 9, the p-value for all measures is smaller than 0.05. Considering the common significance level of 0.05, the results of the paired samples t-test showed that for all metrics, there is a significant difference between the performance of the HNSGAII-PSO algorithm and that of the NSGA-II algorithm.

To meticulously evaluate the performance of the algorithms, comprehensive statistical analyses were conducted for each performance measure. The results obtained for the performance metrics in Table 8 were transformed using the relative percentage deviation

TABLE 9. Paired sample t-test results of the HNSGAII-PSO and NSGA-II algorithms

Metric	HNSGAII-PSO		NSGA-II		p-value	Significance
	Mean	SD	Mean	SD		
NPS	350.53	41.24	302.67	32.46	0.0021	Yes
D_1	90.59	46.94	156.25	89.16	0.0003	Yes
D_2	71.52	35.02	132.03	71.47	0.0002	Yes
MID	20343.8	9360.77	23465.29	11957.24	0.0008	Yes

(RPD) method to facilitate an unbiased comparison. Interval plots with a 95% confidence level were constructed for each performance measure to assess the algorithm's accuracy and robustness (53). A smaller interval indicates superior accuracy, while lower interval values compared to other algorithms indicate greater robustness. These findings are illustrated in Figures 20 to 23.

An analysis of the NPS metric (Figure 20) revealed that the HNSGAII-PSO algorithm exhibited superior robustness compared to the NSGA-II algorithm, despite their equivalent accuracy. Regarding the D_1 indicator (Figure 21), the HNSGAII-PSO algorithm demonstrated exceptional performance, achieving both the highest level of robustness and accuracy. For both the D_2 and MID metrics, Figures 22 and 23, respectively, illustrate the HNSGAII-PSO algorithm's superior accuracy and robustness compared to the NSGA-II algorithm.

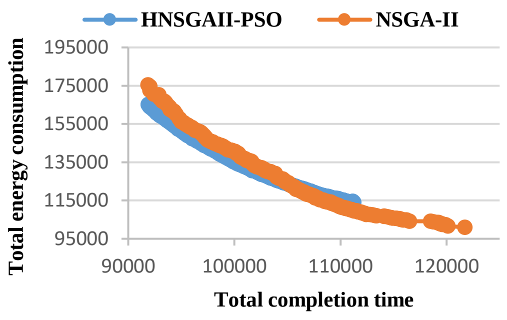


Figure 19. Pareto fronts obtained by various algorithms for problem instance #55

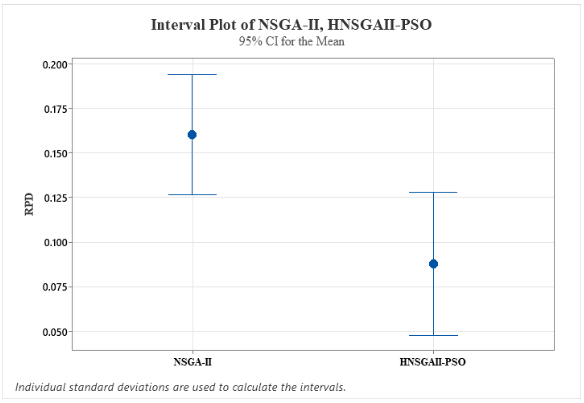


Figure 20. Interval plot based on a 95% confidence level for the comparison of the NSGA-II and HNSGAII-PSO algorithms using the NPS metric

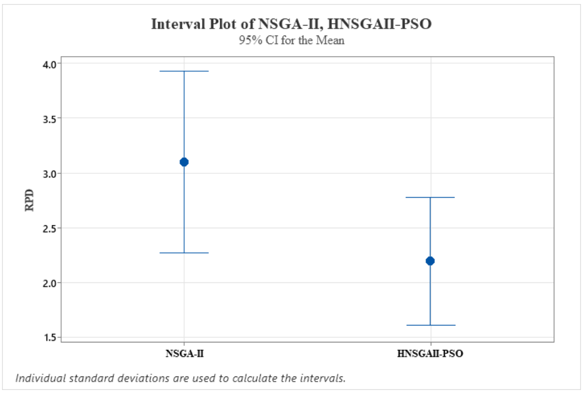


Figure 21. Interval plot based on a 95% confidence level for the comparison of the NSGA-II and HNSGAII-PSO algorithms using the D_1 metric

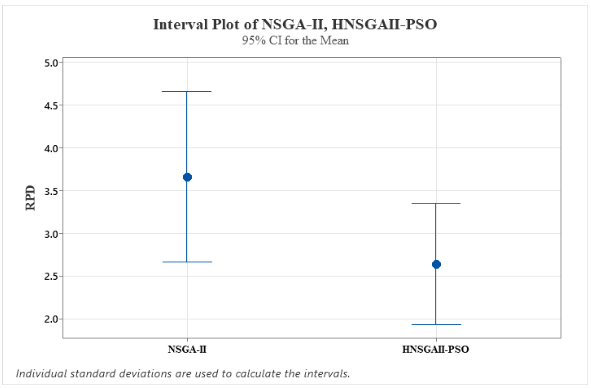


Figure 22. Interval plot based on a 95% confidence level for the comparison of the NSGA-II and HNSGAII-PSO algorithms using the D_2 metric

5. 6. Sensitivity Analysis In this subsection, the effect of varying the parameters p_{js}, e_{js}, and m on the

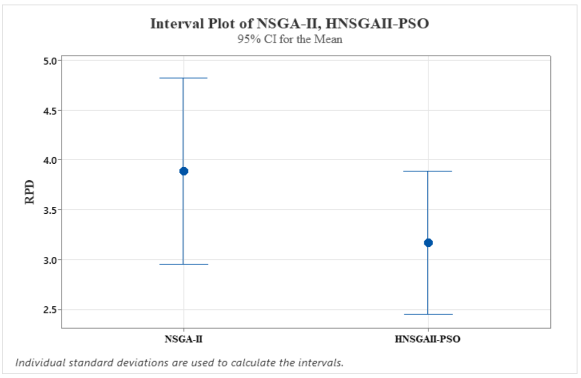


Figure 23. Interval plot based on a 95% confidence level for the comparison of the NSGA-II and HNSGAII-PSO algorithms using the MID metric

objective functions is analyzed. Problem instance #7 is considered for the sensitivity analysis. Figures 24, 25, and 26 depict the Pareto fronts obtained by solving novel problems based on varying the parameters p_{js}, e_{js}, and m, respectively. Figure 24 shows that by increasing the p_{js} parameter, the processing time of each pattern increased, and when the job-pattern matrix is considered, the objective function of the total completion time also increased. This could potentially lead to customer dissatisfaction. However, the utilization of advanced machines and trained personnel can lower the p_{js} parameter. To avoid an increase in the p_{js} parameter, decision-makers must allocate sufficient funds for the purchase and installation of new machines and provide regular training programs for employees to improve their skills. Figure 25 illustrates how varying the e_{js} parameter influences the total energy consumption objective function. The Pareto front shifts towards higher values of this objective function as the e_{js} parameter increases. Utilizing high-tech machinery and performing timely maintenance and repairs can be effective in preventing the increase in the e_{js} parameter and its associated expenses. Therefore, decision-makers should conduct the necessary comparisons between various alternatives to implement the optimal strategy for maximizing overall profit. Figure 26 demonstrates that decreasing the m parameter increases both the total completion time and total energy consumption. In fact, as the number of machines is reduced, more patterns are assigned to the remaining machines, resulting in an increase in the time required to complete all jobs. To tackle this issue and reduce customer dissatisfaction, it is necessary to select the machine's fast levels for pattern processing in order to reduce their completion time, resulting in an increase in energy consumption. This analysis enables decision-makers to strike a balance between the required budget for purchasing and deploying new machines and the costs resulting from an increase in total completion time and total energy consumption.

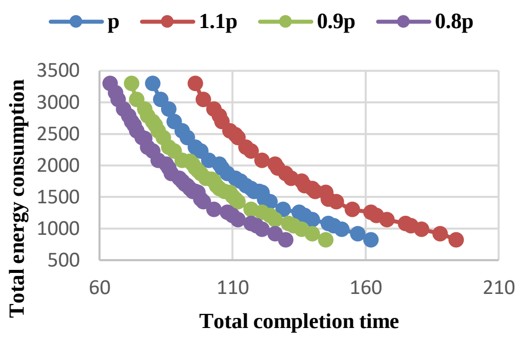


Figure 24. Sensitivity of the Pareto frontier to processing time

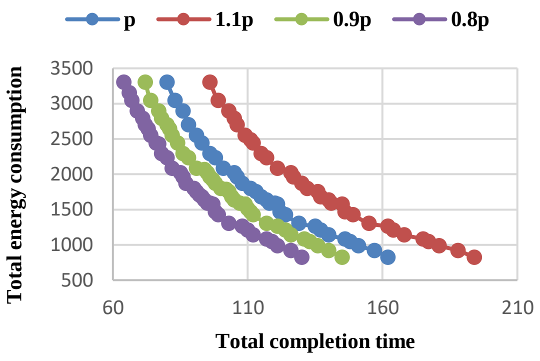


Figure 25. Sensitivity of the Pareto frontier to energy consumption

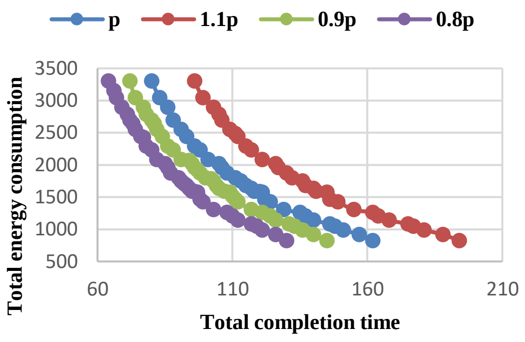


Figure 26. Sensitivity of the Pareto frontier to the number of machines

6. CONCLUSION AND FUTURE RESEARCH

In the realm of manufacturing, the cutting stock problem, frequently encountered in industries such as furniture and apparel, exemplifies the application of common operation scheduling. Conversely, effective management and energy consumption reduction have emerged as pressing concerns within the industry. Addressing these issues is paramount, considering the manufacturing sector's substantial energy consumption. In this paper, the common operation scheduling in an environment of identical parallel machines was studied, considering the energy consumption. In the investigated problem, each job includes several pieces, and all the pieces of different jobs are placed on cutting patterns. Each cutting pattern can contribute to the completion of one or more jobs. Each job is completed when all pieces of that job have been produced by processing related patterns. Each machine in this problem possesses varying speed levels. Consequently, when the machine operates at a higher speed, the processing time is reduced while the consumption of electrical energy increases.

In the investigated problem, to simultaneously minimize the total completion time and the total electrical energy consumption, two position-based and sequence-based mixed integer linear programming models were presented, and to solve small-scale instances, the AUGMECON method was used to obtain the Pareto optimal front. To solve medium- and large-scale instances, the HNSGAII-PSO and NSGA-II algorithms were developed to achieve good approximate Pareto fronts. In the NSGA-II algorithm, each chromosome represents a solution from the problem-solving space, and the quality of each chromosome can be evaluated by decoding it. In the HNSGAII-PSO algorithm, each chromosome represents a region of the problem-solving space where all solutions in this region have the same total energy consumption. To assess the quality of each chromosome, the top solution within the chromosome's covered region is taken into consideration. This entails running the PSO algorithm once for every chromosome to determine the best solution in each region. The performance of the presented algorithms was evaluated by solving test instances of different sizes. The results of numerical experiments show that both presented algorithms perform well in solving small-scale instances and can obtain the optimal Pareto front in much less time than the AUGMECON method. Based on the results of solving medium- and large-scale instances, the HNSGAII-PSO algorithm has better performance compared to the NSGA-II algorithm and can obtain more diverse non-dominant solutions with better convergence. At last, the problem's sensitivity to the parameters of processing time, energy consumption, and the number of machines was analyzed, and the impact of varying each of these parameters on the two objective functions was demonstrated. The results indicate that decision-makers should compare the budgets, costs, and revenues associated with different changes to make well-informed decisions that maximize overall profit.

Considering sequence-dependent setup times for processing cutting patterns, studying the problem in an environment of unrelated parallel machines, and using the TOU tariffs or the tiered price to calculate the cost of power consumption are all attractive fields for future research.

7. REFERENCES

1. Babaee Tirkolaee E, Goli A, Mardani A. A novel two-echelon hierarchical location-allocation-routing optimization for green energy-efficient logistics systems. Annals of Operations Research. 2021. 10.1007/s10479-021-04363-y
2. Zhang X, Zhou H, Fu C, Mi M, Zhan C, Pham DT, et al. Application and planning of an energy-oriented stochastic disassembly line balancing problem. Environmental Science and Pollution Research. 2023:1-15. 10.1007/s11356-023-27288-4
3. Zandvakili A, Mansouri N, Javidi M. Energy-aware task scheduling in cloud compting based on discrete pathfinder algorithm. International Journal of Engineering, Transactions C: Aspects. 2021;34(9):2124-36. 10.5829/IJE.2021.34.09C.10
4. Tian G, Zhang C, Fathollahi-Fard AM, Li Z, Zhang C, Jiang Z. An enhanced social engineering optimizer for solving an energy-efficient disassembly line balancing problem based on bucket brigades and cloud theory. IEEE Transactions on Industrial Informatics. 2022. 10.1109/TII.2022.3193866
5. Fang K, Uhan N, Zhao F, Sutherland JW. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. Journal of Manufacturing Systems. 2011;30(4):234-40. 10.1016/j.jmsy.2011.08.004
6. Lu C, Gao L, Li X, Pan Q, Wang Q. Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. Journal of cleaner production. 2017;144:228-38. 10.1016/j.jclepro.2017.01.011
7. Jovane F, Yoshikawa H, Alting L, Boer CR, Westkamper E, Williams D, et al. The incoming global technological and industrial revolution towards competitive sustainable manufacturing. CIRP annals. 2008;57(2):641-59. 10.1016/j.cirp.2008.09.010
8. Mori M, Fujishima M, Inamasu Y, Oda Y. A study on energy efficiency improvement for machine tools. CIRP annals. 2011;60(1):145-8. 10.1016/j.cirp.2011.03.099
9. Pinedo ML. Scheduling: Springer; 2012.
10. Ding J, Schulz S, Shen L, Buscher U, Lü Z. Energy aware scheduling in flexible flow shops with hybrid particle swarm optimization. Computers & Operations Research. 2021;125:105088. 10.1016/j.cor.2020.105088
11. Zhang M, Yan J, Zhang Y, Yan S. Optimization for energy-efficient flexible flow shop scheduling under time of use electricity tariffs. Procedia CIRP. 2019;80:251-6. 10.1016/j.procir.2019.01.062
12. Guo J, Lei D, Li M. Two-phase imperialist competitive algorithm for energy-efficient flexible job shop scheduling. Journal of Intelligent & Fuzzy Systems. 2021;40(6):12125-37. 10.3233/JIFS-210198
13. Fathollahi-Fard AM, Woodward L, Akhrif O. Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept. Journal of Industrial Information Integration. 2021;24:100233. 10.1016/j.jii.2021.100233
14. Li Z, Yang H, Zhang S, Liu G. Unrelated parallel machine scheduling problem with energy and tardiness cost. The International Journal of Advanced Manufacturing Technology. 2016;84:213-26. 10.1007/s00170-015-7657-2
15. Che A, Zhang S, Wu X. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. Journal

of cleaner production. 2017;156:688-97. 10.1016/j.jclepro.2017.04.018

16. Wang Y-C, Wang M-J, Lin S-C. Selection of cutting conditions for power constrained parallel machine scheduling. Robotics and Computer-Integrated Manufacturing. 2017;43:105-10. 10.1016/j.rcim.2015.10.010

17. Zeng Y, Che A, Wu X. Bi-objective scheduling on uniform parallel machines considering electricity cost. Engineering Optimization. 2018;50(1):19-36. 10.1080/0305215X.2017.1296437

18. Wu X, Che A. A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. Omega. 2019;82:155-65. 10.1016/j.omega.2018.01.001

19. Cota LP, Coelho VN, Guimarães FG, Souza MJ. Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. International Transactions in Operational Research. 2021;28(2):996-1017. 10.1111/itor.12566

20. Safarzadeh H, Niaki STA. Bi-objective green scheduling in uniform parallel machine environments. Journal of cleaner production. 2019;217:559-72. 10.1016/j.jclepro.2019.01.166

21. Wang S, Wang X, Yu J, Ma S, Liu M. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. Journal of cleaner production. 2018;193:424-40. 10.1016/j.jclepro.2018.05.056

22. Anghinolfi D, Paolucci M, Ronco R. A bi-objective heuristic approach for green identical parallel machine scheduling. European Journal of Operational Research. 2021;289(2):416-34. 10.1016/j.ejor.2020.07.020

23. Zhang H, Wu Y, Pan R, Xu G. Two-stage parallel speed-scaling machine scheduling under time-of-use tariffs. Journal of Intelligent Manufacturing. 2021;32:91-112. 10.1007/s10845-020-01561-6

24. Keshavarz T, Karimi E, Shakhsi-Niaei M. Unrelated parallel machines scheduling with sequence-dependent setup times to minimize makespan and tariff charged energy consumption. Advances in Industrial Engineering. 2021;55(1):91-113. 10.22059/jieng.2021.326682.1788

25. Zhou B-H, Gu J. Energy-awareness scheduling of unrelated parallel machine scheduling problems with multiple resource constraints. International Journal of Operational Research. 2021;41(2):196-217. 10.1504/IJOR.2021.115623

26. Módos I, Šucha P, Hanzálek Z. On parallel dedicated machines scheduling under energy consumption limit. Computers & Industrial Engineering. 2021;159:107209. 10.1016/j.cie.2021.107209

27. Rego MF, Pinto JCE, Cota LP, Souza MJ. A mathematical formulation and an NSGA-II algorithm for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling. PeerJ Computer Science. 2022;8:e844. 10.7717/peerj-cs.844

28. Asadpour M, Hodaei Z, Azami M, Kehtari E, Vesal N. A green model for identical parallel machines scheduling problem considering tardy jobs and job splitting property. Sustainable Operations and Computers. 2022;3:149-55.

29. Gaggero M, Paolucci M, Ronco R. Exact and Heuristic Solution Approaches for Energy-Efficient Identical Parallel Machine Scheduling with Time-of-Use Costs. European Journal of Operational Research. 2023.

30. Arbib C, Servilio M, Felici G, Servilio M. Sorting common operations to minimize the number of tardy jobs. Networks. 2014;64(4):306-20. 10.1002/net.21576

31. Cheng T, Diamond J, Lin BM. Optimal scheduling in film production to minimize talent hold cost. Journal of Optimization Theory and Applications. 1993;79(3):479-92. 10.1007/BF00940554

32. Wang J, Qiao C, Yu H, editors. On progressive network recovery after a major disruption. 2011 Proceedings IEEE INFOCOM; 2011: IEEE. 10.1109/INFCOM.2011.5934996

33. Arbib C, Felici G, Servilio M. Common operation scheduling with general processing times: A branch-and-cut algorithm to minimize the weighted number of tardy jobs. Omega. 2019;84:18-30. 10.1016/j.omega.2018.04.002

34. Cherri AC, Arenales MN, Yanasse HH, Poldi KC, Vianna ACG. The one-dimensional cutting stock problem with usable leftovers–A survey. European Journal of Operational Research. 2014;236(2):395-402. 10.1016/j.ejor.2013.11.026

35. Hinxman A. The trim-loss and assortment problems: A survey. European Journal of Operational Research. 1980;5(1):8-18. 10.1016/0377-2217(80)90068-5

36. Dyckhoff H. A typology of cutting and packing problems. European journal of operational research. 1990;44(2):145-59. 10.1016/0377-2217(90)90350-K

37. Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. European journal of operational research. 2007;183(3):1109-30. 10.1016/j.ejor.2005.12.047

38. Arbib C, Marinelli F. On cutting stock with due dates. Omega. 2014;46:11-20. 10.1016/j.omega.2014.01.004

39. Cui Y, Zhong C, Yao Y. Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. European Journal of Operational Research. 2015;243(2):540-6. 10.1016/j.ejor.2014.12.015

40. Wuttke DA, Heese HS. Two-dimensional cutting stock problem with sequence dependent setup times. European Journal of Operational Research. 2018;265(1):303-15. 10.1016/j.ejor.2017.07.036

41. Graham RL, Lawler EL, Lenstra JK, Kan AR. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of discrete mathematics. 5: Elsevier; 1979. p. 287-326.

42. Garey MR, Johnson DS. ``strong"np-completeness results: Motivation, examples, and implications. Journal of the ACM (JACM). 1978;25(3):499-508.

43. Deb K, Agrawal S, Pratap A, Meyarivan T, editors. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. International conference on parallel problem solving from nature; 2000: Springer.

44. Duan J, Wang J. Energy-efficient scheduling for a flexible job shop with machine breakdowns considering machine idle time arrangement and machine speed level selection. Computers & Industrial Engineering. 2021;161:107677.

45. Xue L, Wang X. A multi-objective discrete differential evolution algorithm for energy-efficient two-stage flow shop scheduling under time-of-use electricity tariffs. Applied Soft Computing. 2023;133:109946.

46. Eberhart R, Kennedy J, editors. Particle swarm optimization. Proceedings of the IEEE international conference on neural networks; 1995: Citeseer.

47. Hulett M, Damodaran P, Amouie M. Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization. Computers & Industrial Engineering. 2017;113:425-36.

48. Marichelvam M, Geetha M, Tosun Ö. An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors–A case study. Computers & Operations Research. 2020;114:104812. 10.1016/j.cor.2019.104812

49. Damodaran P, Diyadawagamage DA, Ghrayeb O, Vélez-Gallego MC. A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines. The International Journal of Advanced Manufacturing Technology. 2012;58(9):1131-40.

50. Lian Z. A united search particle swarm optimization algorithm for multiobjective scheduling problem. Applied Mathematical Modelling. 2010;34(11):3518-26. 10.1016/j.apm.2010.03.001

51. Afzalirad M, Rezaeian J. A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches. Applied Soft Computing. 2017;50:109-23. 10.1016/j.asoc.2016.10.039

52. Zandi A, Ramezanian R, Monplaisir L. Green parallel machines scheduling problem: A bi-objective model and a heuristic algorithm to obtain Pareto frontier. Journal of the Operational Research Society. 2020;71(6):967-78. 10.1080/01605682.2019.1595190

53. Taguchi G. Introduction to quality engineering, Asian productivity organization. Dearborn, Michigan: American Supplier Institute Inc. 1986.

Persian Abstract

چکیده

رشد بی‌امان مصرف انرژی در جهان، چالش‌های پیچیده زیادی از جمله کاهش منابع تجدیدناپذیر انرژی و تشدید انتشار گازهای گلخانه‌ای را به همراه دارد که به تغییرات آب و هوایی کمک می‌کند. در مواجهه با این نگرانی‌های شدید زیست‌محیطی، فشار زیادی به بخش تولید به عنوان مصرف‌کننده مهم انرژی برای اتخاذ شیوه‌های پایدار وارد می‌شود. بنابراین همزمان در نظر گرفتن مدیریت مصرف انرژی و زمان‌بندی عملیات تولید به عنوان یک حوزه محوری برای پرداختن به این چالش‌ها از اهمیت بالایی برخوردار است. زمان‌بندی عملیات مشترک که نمونه آن مسأله برش موجودی در صنایعی مانند مبلمان و پوشاک است، یک چالش رایج در محیط‌های تولید است.در این مقاله، برای اولین بار مسأله زمان‌بندی در محیط ماشین‌های موازی یکسان با در نظر گرفتن عملیات مشترک به منظور کمینه نمودن همزمان مجموع زمان‌های تکمیل و مجموع انرژی مصرفی مورد مطالعه قرار می‌گیرد. بدین منظور برای مسأله مورد بررسی، دو مدل برنامه‌ریزی خطی عدد صحیح آمیخته دو هدفه ارائه می‌گردد و برای حل مسائل با ابعاد کوچک از روش محدودیت اپسیلون تکامل‌یافته (AUGMECON) به منظور دستیابی به مجموعه نقاط پارتو بهینه استفاده می‌شود. با توجه به پیچیدگی محاسباتی مسأله، الگوریتم ژنتیک مرتب‌سازی نامغلوب (NSGA-II) و الگوریتم ژنتیک مرتب‌سازی نامغلوب ترکیب شده با بهینه‌سازی ازدحام ذرات (HNSGAII-PSO) برای حل مسائل با ابعاد متوسط و بزرگ و دستیابی به جبهه های پارتو تقریبی مناسب، توسعه داده می‌شوند. کارایی و عملکرد الگوریتم‌های پیشنهادی با انجام آزمایش‌های محاسباتی بر روی مسائل نمونه، مورد ارزیابی قرار می‌گیرد. نتایج به دست آمده نشان می‌دهند که در حل مسائل نمونه، الگوریتم HNSGAII-PSO در مقایسه با الگوریتم NSGA-II عملکرد بهتری دارد.