Iraqi Journal for Computer Science and Mathematics

Journal Homepage: http://journal.esj.edu.iq/index.php/IJCM
e-ISSN: 2788-7421 p-ISSN: 2958-0544

Review on unrelated parallel machine scheduling problem with additional resources

Munther H. Abed*, Mohd Nizam Mohmad Kahar

Faculty of Computing, University Malaysia Pahang, Pahang, Malaysia

*Corresponding Author: Munther H. Abed and Mohd Nizam Mohmad Kahar

DOI: https://doi.org/10.52866/ijcsm.2023.02.02.020
Received March 2023; Accepted April 2023; Available online May 2023

ABSTRACT: This study deals with an unrelated parallel machine scheduling problem with additional resources (UPMR). That is one of the important sub-problems in the scheduling. UPMR consists of scheduling a set of jobs on unrelated machines. In addition to that, a number of one or more additional resources are needed. UPMR is very important and its importance comes from the wealth of applications; they are applicable to engineering and scientific situations and manufacturing systems such as industrial robots, nurses, machine operators, bus drivers, tools, assembly plant machines, fixtures, pallets, electricity, mechanics, dies, automated guided vehicles, fuel, and more. The importance also comes from the concern about the limitation of resources that are dedicated for the production process. Therefore, researchers and decision makers are still working on UPMR problem to get an optimum schedule for all instances which have not been obtained to this day. The optimum schedule is able to increase the profits and decrease the costs whilst satisfying the customers' needs. This research aims to review and discuss studies related to unrelated parallel machines and additional resources. Overall, the review demonstrates the criticality of resolving the UPMR problem. Metaheuristic techniques exhibit significant effectiveness in generating results and surpassing other algorithms. Nevertheless, continued improvement is essential to satisfy the evolving requirements of UPMR, which are subject to operational changes based on customer demand.

Keywords: Makespan, Resource constraints, Scheduling problems, Unrelated parallel machine

1. INTRODUCTION

In today's competitive world, where jobs are processed largely by machines, the requirement for smart organization becomes an essential need.

If we take into account the fact that the resources, we rely on are scarce, the need has become more urgent. Moreover, since one of the aims of the commercial sector aims is to turn a profit, then this profit could be increased if the manufacturing facilities were operating at their peak efficiency This problem arises in a variety of industry environments, including auto factories and food processing, among others. It is reasonable to expect that a variety of jobs will be processed by a variety of parallel machines, under a variety of specific circumstances, and with a variety of specific objectives. Considered scheduling jobs on parallel machines as a process involves two steps: the first is which jobs are to allocate to which machines and the second is the order of the jobs thar are to allocated to each of these machines.

The regular scheduling problem of the unrelated parallel machine (UPM) can be described as the processing some jobs on some parallel machines. The widely known objective is to minimize the maximum completion time of job. According to [1-5], UPM problem is "A set of n jobs has to be processed on exactly one machine out of a set of m machines. In this variant of the problem, processing times of the jobs differ according to the machine the job is assigned to".

UPM that considers limited resources is named as UPMR. The importance of the UPMR comes from the wealth of applications; they are applicable to engineering or scientific situations where it is needed to present the best sequence in a convincing period of time within limited number of resources. This paper shows some of the latest studies and related literature in the context of unrelated parallel machine scheduling problems with additional resources (UPMR) to ensure our research direction and many techniques have been applied for solving this problem.

*Corresponding author: munt1979@yahoo.com

In today's highly competitive world, the use of machines for job processing has become essential in various industries. As resources become scarce, optimizing scheduling is crucial to boosting profits in fields like auto factories and food processing. Scheduling jobs on parallel machines involves two primary steps: allocating jobs to machines and determining the order of jobs on each machine. The unrelated parallel machine (UPM) scheduling problem aims to minimize the job completion time by scheduling some jobs on parallel machines. However, real-world scenarios often require additional resources, leading to the UPMR problem. This problem is significant in engineering and scientific contexts where efficient resource allocation is crucial. This paper reviews recent studies and related literature on the UPMR scheduling problem, providing insights into the techniques used to solve the problem and identifying potential areas for future research. We survey four major categories of papers, including machine environment, supplemental resources, objective functions, and approaches to problem-solving.

2. DEFINITIONS AND ASSUMPTIONS

In the era of automation and intelligence systems, the production process plays a crucial part in increasing the profits, decreasing the costs whilst satisfying the users. This has attracted the attention of many researches into developing an optimum scheduling for the production process. Production scheduling involves assigning jobs to machines to optimize performance metrics. One important sub-problem in scheduling is the unrelated parallel machine scheduling (UPM) problem, which involves scheduling jobs on machines to minimize the time needed to complete all jobs (*makespan*). However, to execute a job on a machine, a set of resources is required. When additional resources are taken into account, the problem is referred to as UPMR. In an UPM problem, the order in which jobs are scheduled on each machine does not affect the *makespan*. In contrast, in UPMR, the sequence of job assignments to machines affects the *makespan*, as different sequences imply different execution times due to limited resources. Several assumptions for the UPMR problem have been proposed in recent studies (e.g., [6-10]) which includes:

- Only one machine processes one job.
- One job at a time can be processed by each machine.
- Any job processing, that was already initiated, must be fulfilled and finished uninterruptedly (non-preemptively).
- Every job requires a number of additional resources during their entire process not exceeding the R_{max} at any time.
- The precedence constraints are sometimes possible between two jobs.

3. A CLASSIFICATION OF SCHEDULING PROBLEMS

A classification scheme for parallel machine scheduling problem introduced by [11-13]. This scheme could employ a three-field problem classification $\alpha \mid \beta \mid \gamma$.

- Machine environment characteristics (α)
- Job characteristics and constraints (β)
- Global optimality criterion (Objective functions γ)

3.1 MACHINE ENVIRONMENT CHARACTERISTICS

The characteristics of machine environments can be categorized into two groups: machine variety and machine environment. Concerning the variety of machines, the regular parallel machine scheduling problem addresses independent tasks that must be processed by a set of parallel machines non-preemptively to achieve a specific performance measure [14]. The majority of studies focus on more than three machines, while some suggest that the number of machines may vary as part of the input [9]. In terms of the machine environment, the parallel machine scheduling problem (PMSP) can be classified into three types that are based on the machines' nature: identical, uniform, and unrelated [15-17]. However, previous studies employed a classification scheme that distinguished five sub-cases of parallel machines based on their environment [11, 12]. These sub-cases are: single machine ($\emptyset$), parallel dedicated machines (PD), identical parallel machines (P), uniform parallel machines (Q), and unrelated parallel machines (R).

Single machine refers to a production system that has only one machine for processing jobs or tasks. The machine is capable of performing a specific operation or a sequence of operations that are required to complete a job. Parallel dedicated machines, is a type of manufacturing system that consists of several dedicated machines that are used to produce a specific product or set of products. The machines are dedicated to the production of the product(s) and are arranged in parallel, where they operate simultaneously to produce the required output. Each machine in the parallel dedicated system performs a specific task in the production process. For example, in a parallel dedicated system for automobile manufacturing, there may be dedicated machines for welding, painting, and assembly.

Identical parallel machines, is a type of production system in which a set of machines is available to process a set of jobs, and all the machines are identical in terms of their processing capabilities, capacities, and speed [18]. In other words, each machine in this production system is interchangeable with any other machine, and they can all perform the same tasks with the same level of efficiency. Uniform parallel machines, is a type of production system in which a set of machines is available to process a set of jobs, and all the machines have uniform or identical processing capabilities, but may have different processing speeds or capacities [19]. In other words, the machines in the production system have the same processing capabilities and, thus, they can perform the same tasks, but they may process those tasks at different rates.

Unrelated parallel machines, on the other hand, is a type of production system in which a set of machines is available to process a set of jobs, and each machine has a unique processing capability or specialization [8]. In other words, the machines in the system are not identical and have different processing capabilities or functions [20]. The unrelated parallel machine is further divided into three sub-cases: regular UPM, with sequence-dependent setup time UPMSP, and with resources UPMR.

- **Regular unrelated parallel machine**
  In regular unrelated parallel machine problems, the concept of unrelated parallel machines refers to a situation where a set of jobs must be processed on a specific machine from a group of parallel machines that are not related to each other. Each job must have a processing time assigned to it, and the processing should begin at time zero.

- **Unrelated parallel machine with sequence dependent setup time**
  In unrelated parallel machines with sequence-dependent setup times, an additional constraint is introduced, which is the time required to prepare the machine for a specific job. The setup times are dependent on both the job sequence and the machine used, and each machine has its own matrix of setup times. This creates a more complex scheduling problem where the optimal distribution of jobs among machines must be determined, and the best order for each machine must be established to ensure that the resource restrictions are met at all times. One example of this type of scheduling problem is $R_M\,|S_{jkm}|C_{max}$ [3, 21].

- **Unrelated parallel machine with resources**
  In the case of unrelated parallel machines with resources (UPMR), a renewable resource is required for each job within its processing time set, and only a limited amount of the resource is available at any given time. The scheduling problem involves assigning the jobs to the machines, determining the start and completion times, and ensuring that the machines are never stopped between the completion of one job and the start of another. This version of the problem is more complex due to the additional constraint of resource availability and may result in idle times if the machines are unable to process the next job due to a lack of resources [9].

Considering the importance of resource constraints in industrial settings, the UPMR problem is a crucial factor that must be taken into account when determining the optimal scheduling solution.

3.2 JOB CHARACTERISTICS AND CONSTRAINTS

A great reference of the characteristics and constraints of a job, including processing times, preemptions, and resource constraints, can be referred to [11, 12, 22-24]. It should be noted that the resource consumption and processing times of a job can vary depending on the machine that is allocated to process the job. This is particularly important in real manufacturing environments where different machines may need to be used simultaneously. In such cases, the processing time or resource consumption of a job on an old machine may differ significantly from that on a new machine, as noted in references [9, 10]. Thus, it is essential to carefully consider the characteristics of each machine and allocate jobs accordingly to optimize scheduling and resource utilization.

3.2.1 Processing times

The processing time of a job in a parallel application is significantly impacted by the machines to which it is assigned. As such, the allocation of jobs to machines is a critical factor that can greatly affect system performance and efficiency. To describe the processing times of jobs, the parameter $\beta 5 \in [\emptyset, p_j = p]$ is used. Two possible values for $\beta 5$ are:

- $\beta 5 = \emptyset$: This indicates that the processing times of jobs are arbitrary, and the parameter p_j may or may not be used to represent these times.
- $\beta 5 = (p_j = p)$: This means that all jobs have the same processing time, which is equal to p units.

3.2.2 Possibility of preemptions

The processing of jobs can be interrupted or uninterrupted based on the allowance of preemptions. When preemptions are not allowed, the processing of any job must continue uninterruptedly until it is finished. However, in the case of preemptions, any operation's processing can be stopped and resumed later, even on a different machine. To determine whether preemption is possible or not, parameter $\beta 1 \in [\emptyset, pmtn]$ is used, where:

- $\beta 1 = \emptyset$: no preemption is allowed.
- $\beta 1 = pmtn$: preemption is allowed.

3.2.3 Machine eligibility restrictions (Job-machine assignment)

The assignment of the job to machine could be done in two ways: assigning jobs to unspecified or specified machines means a specific issue of the unrelated parallel machine environment. $\beta 7 \in [\emptyset, M_j]$: means the eligibility restrictions of machines:

- $\beta 7 = \emptyset$: all machines are eligible for all jobs (unspecified).
- $\beta 7 = M_j$: the processing of job j can only be carried out by a specific machine subset M_j of M machines (specified).

3.2.4 Additional Resource constraints

In some studies of parallel machine scheduling, machines are considered as a resource. However, in real-world manufacturing settings, jobs often require additional resources like automated guided vehicles, machine operators, tools, pallets, dies, and industrial robots for their handling and processing [12, 25, 26]. These resources have various characteristics that must be taken into account while solving the UPMR problem (refer to Fig. 1).

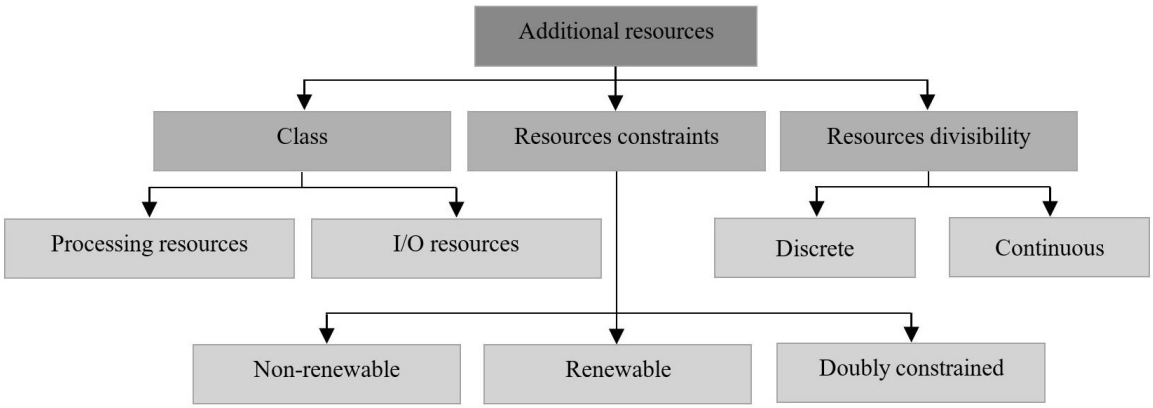

FIGURE 1. - Classification of additional resources

a) To characterize additional resources, Parameter $\beta 2$ is used, which belongs to the range $[\emptyset, res\lambda\sigma\delta]$ [22, 23]. The different values of $\beta 2$ are defined as follows:

- $\beta 2 = \emptyset$, then no additional resources are present.
- $\beta 2 = res\lambda\sigma\delta$, then specified resource constraints exist. λ, σ, and δ are described below:
  - If λ is a positive integer, it represents a constant variety of resource types that are equivalent to λ. If λ = ., it represents a part of the input, and its value is arbitrary.
  - If σ is a positive integer, then all resource sizes, or resource limits, are constant and equal to σ. If σ = ., then all resource sizes are characterized as arbitrary.
  - If δ is a positive integer, then all resource requirements have a constant upper bound that is equal to δ. If δ = ., then such bounds are not specified.

b) The additional resources are categorized into three classes based on their resource constraints [23, 25]:

- If a resource is used for a job, after being released from this job it may be used again for other job. This type of resources is called *renewable* [9, 10].

- If a resource has already been used for one job, then it cannot be used for another job. This type of resources is named *nonrenewable* [27].
- If both *renewable resources* and *nonrenewable resources* are used in the same time, this type is called *doubly constrained* [28].

c) In the resource divisibility, additional resources can be categorized into two classes, as per references [23, 25]. These classes are:

- Discrete resources, which refer to the distribution of jobs in separate amounts from a specific, finite variety of possible allocations. These allocations may consist of just one element [20].
- Continuous resources, which may be allocated to jobs in random, a priori unknown amounts from predetermined intervals. A significant amount of research on continuous resources is available [29].

d) The additional resources can be classified into two categories from the perspective of job processing [7]:

- Processing resources: These are resources that are needed for processing a given job set on a specific machine.
- Input-output resources: These are resources that are required either before or after the processing of a job [7, 23, 30, 31].

e) The allocation of resources to machines must be carefully considered, and there are two classes of allocation methods as described in [17]:

- Static allocation: the resource allocation remains fixed throughout the schedule, and the additional resource cannot be switched to other machines.
- Dynamic allocation: the assignment and non-assignment of resources are based on the type of job, and the additional resource can be switched among machines during the schedule.

3.3 GLOBAL OPTIMALITY CRITERION

There exist several types of objective functions used in the parallel machine scheduling problem. In constructing optimality criteria (refer to Table 1) [32], the following elementary functions are considered:

Table 1. - The objective function for PMS

Measure	Symbol	Formula
Flow time	F_j	$F_j = C_j - r_j$
Lateness	L_j	$L_j = C_j - d_j$
Tardiness	T_j	$max\ [C_j - d_j, 0]$
Earliness	E_j	$max\ [d_j - C_j, 0]$
Total completion time	C_j	$\sum_{j=1}^{J} C_j$
Total weighted completion time	$w_j C_j$	$\sum_{j=1}^{J} w_j C_j$
Total flow time	F_j	$\sum_{j=1}^{J} F_j$
Total weighted flow time	$w_j F_j$	$\sum_{j=1}^{J} w_j F_j$
Total tardiness	T_j	$\sum_{j=1}^{J} T_j$
Total weighted tardiness	$w_j T_j$	$\sum_{j=1}^{J} w_j T_j$
Number of tardy jobs	U_j	$\sum_{j=1}^{J} U_j$
Weighted number of tardy jobs	$w_j U_j$	$\sum_{j=1}^{J} w_j U_j$
Maximum lateness	L_{max}	$max_j L_j$
Maximum tardiness	T_{max}	$max_j T_j$
Makespan	C_{max}	$max_j C_j$

This review paper focuses on a particular type of objective function utilized in parallel machine scheduling studies, which aims to minimize the *makespan* criterion. The *makespan* is responsible for balancing the machines based on their loads, leading to optimal machine utilization [6]. Furthermore, the *makespan* can be defined as the time when the last job is completed [33, 34], and a minimum *makespan* usually indicates efficient machine handling [32].

4. RELATED WORK

This section presents a review of the approaches used to solve the dynamic unrelated parallel machine scheduling problem, both specified and unspecified versions. According to the literature, these approaches can be classified into four categories: exact methods, ρ-approximation algorithms, problem-based heuristics, and metaheuristics. Scholars have explored and studied *specified dynamic (sd)* versions for UPMR problem. For example, [35] proposed a heuristic approach with secondary resource constraints to minimize *makespan* for *specified dynamic* UPMR. The results obtained by the computational processes indicate that the presented heuristic performs better than the simulated annealing. [36] presented an effective heuristic approach based on threshold-accepting method, tabu lists, and improvement steps with assistant equipment constraints as secondary resource constraints, for the specified dynamic unrelated parallel machine to minimize total tardiness. Computational experiences point out the capability of the suggested heuristic to gain the best solutions for the problems of small size, and considerably outperforms an ATCS procedure and a SA for the large-size instances.

Constraint programming (CP), integer programming (IP), and integrated IP/CP models were proposed by [37] to minimize C_{max} for specified dynamic UPMR, and the combined IP/CP model obtained the best results in most instances. [38] have developed approaches of IP/IP and IP/CP models to minimize the *makespan* for specified dynamic unrelated parallel machine. They reported that better results are offered by IP/IP than IP when the constraints of resources are loose. Additionally, IP/CP model outperforms CP model when the constraints of the resource were tight.

Much literature on *unspecified dynamic (ud)* UPMR has also been presented in earlier studies. For example, [39] proposed 2-approximation, 3/2-approximation and 4-approximation approach so that to the C_{max} and w_jC_j are minimized for unspecified dynamic UPMR. The 4-approximation method are rendered better than the other two methods. [40] made development on 3.75-approximation approach through the use of the rounding procedure which is utilized to solve *ud*-UPMR by the means of minimizing the *makespan*. As it can be seen, the results, obtained through the application of the model in question, outperform 4-approximation and 6.83-approximation.

A Lagrangian-based CP method has been suggested by the way of keeping the resource constraints relaxed [41]. The results of this method have been compared to the results of IP and CP approaches for the sake of discovering the fact that the method of the proposed Lagrangian-based CP presents results that are really considered effective. An integer programming (IP) model has been proposed by [42] to solve *dynamic* UPMR problems which is defined as a relaxed IP based CP method used for the purpose solving the large-sized instances for *dynamic* UPMR and IP/CP model. To go further. This model, additionally, performs better than the IP model and obtains near-best solutions for problems characterized as being of large sizes. [20] study the scheduling problem of the unspecified dynamic UPMR. The aim of this study is oriented to scheduling jobs in parallel machines as a step to minimize the *makespan*. Two approaches have been presented by those two scholars: one is an integer linear programming ILP program and the other is a two-phase approach based on solutions, named the fixing algorithm. The fixing algorithm outperforms the ILP program.

L. Fanjul Peyro et al. investigated unspecified dynamic UPMR goal to minimize the *makespan* [9]. The resources were very limited and unchanged in terms of their availability within the production aspect. The number of resources is based on the job in addition to the machine. They formulate this problem via two mixed integer linear programming MILPs. Of these approaches is one that was based on a model previously reported by [42] and referred to by UPMR-S. The second approach takes into account the aspect of resemblance to the strip packing problem (referred to by UPMR-P). Additionally, three techniques, matheuristic in nature, were also reported by those scholars. These included "Job-machine reduction JMR, Machine-assignment fixing MAF and Greedy-based fixing GBF" which have been used in each one of the aforementioned methods (UPMR-P and UPMR-S) and resulted JMR-P, MAF-P, GBF-P, JMR-S, MAF-S, and GBF-S. The JMR-P approach, thus, performs better than all other approaches in most instances.

CP model has been presented to solve the unspecified dynamic unrelated parallel machine for the aim of minimizing the C_{max} [17]. The CP model, as shown by the results, outperforms the heuristic and exact methods in the earlier studies (UPMR-P, UPMR-S, MAF-P and MAF-S) for all instances used in the research. [43] has also suggested two approaches for the sake of solving the unrelated parallel machine scheduling problem with a renewable resource constraint to minimize the *makespan*. As such, the methods, in question, are MILP for two machines and MILP/CP model for more than two machines. If the solution of the MILP/CP model is not characterized as being the optimal, then, the solution of the problem is carried out by the use of a CP model. The MILP/CP outperforms CP model for a problem of a large size.

Multi-pass heuristics and local search methods (NEH_{st}, NEH_{res} and SWA) have been proposed by [10] for the sake of minimizing the *makespan* for unspecified dynamic UPMR. As for small instances, the best results were obtained by NEH_{res}; in medium and large instances, the optimal results went to the M4 and M5 respectively. [44] suggested four approaches (*RLS*, *SS*, *ESS* and *EIG*) to minimize C_{max} for unspecified dynamic unrelated parallel machines with additional resources. The results of the Enriched iterated greedy perform better than the methods that are related to instances of small, medium and large sizes.

Recently two works are applied to solve *ud*-UPMR: guided genetic algorithm (GGA) [45] and hybrid guided genetic algorithm (GGA-GD, GGA-TS and GGA-VNS) [46]. Table 2 illustrates the strengths and limitations for the methods that are applied on the *unspecified dynamic* UPMR.

Table 2. - Strengths and limitations of the methods that applied on the *ud*-UPMR

Technique	Strengths	Limitations
2-approximation algorithm [39]	- It is often faster and easier to implement than exact optimization algorithms, which can be computationally expensive and time-consuming. - It is often simple to understand and interpret, making them useful for applications where complex algorithms may not be necessary.	- It does not provide the optimal solution to an optimization problem, which can be a disadvantage for applications where exact solutions are necessary. - It may not scale well to very large optimization problems, as the size of the problem can increase the computational complexity of the algorithm. - The quality of the approximation provided by a 2-approximation algorithm may depend on the specific instance of the optimization problem, and some instances may require a larger approximation factor than 2.
3/2-approximation algorithm [39]	- It guarantees that the solution it finds is no worse than 3/2 times the optimal solution, which can provide confidence in the quality of the solution. - It can often be implemented with relatively low computational overhead, making it a practical choice for many optimization problems.	- It is not guaranteed to find the optimal solution, and in some cases, it may produce solutions that are significantly worse than the optimal solution. - It can be sensitive to the specific problem instance and input data, and may not perform well for all problems. - The solutions produced by the 3/2-approximation algorithm may be difficult to interpret or explain, particularly if the algorithm involves complex optimization techniques or heuristics.
4-approximation algorithm [39]	- It is often much faster than exact algorithms, as they do not need to explore the entire solution space. - It is often robust to changes in the problem formulation, input data, or algorithm parameters.	- It may produce suboptimal solutions, meaning that the solution may not be as good as the optimal solution. - It may not be appropriate for some problems where a higher degree of accuracy is required. - The quality of the solution produced by a 4-approximation algorithm depends on the problem structure, and in some cases, the solution produced may be significantly worse than the optimal solution.
6.83-approximation algorithm [40]	- The algorithm provides a solution that is guaranteed to be no worse than 6.83 times the optimal solution, which is a strong performance guarantee. - It is relatively simple to implement and understand, which makes it accessible to a wide range of users. - It can handle large-scale instances of the Euclidean TSP problem, making it suitable for real-world applications.	- It may not always provide a solution that is close to optimal, particularly for difficult instances of the Euclidean TSP problem. - It is designed specifically for the Euclidean TSP problem and may not be applicable to other types of optimization problems. - The performance of the algorithm can be sensitive to the distribution of the input data, and it may not perform well for certain types of distributions.
3.75-approximation algorithm [40]	- The algorithm guarantees that the size of the independent set it produces is within a factor of 3.75 of the optimal solution, providing a good trade-off between solution quality and computational efficiency. - It is relatively simple to implement and requires only basic graph operations, making it a practical choice for many applications. - It has a polynomial time complexity, making it computationally efficient for large-scale graphs.	- It may not always produce a good approximation for certain types of graphs, particularly those with complex structures or low-density graphs. - It may require careful tuning of its parameters, such as the threshold used to identify candidate independent set vertices, to achieve good performance on a particular graph. - It may not scale well to very large graphs or those with high degrees of connectivity, as it relies on a greedy approach to select independent set vertices.

table 2 continued

Technique	Strengths	Limitations
Integer Programming IP [41, 42]	- IP provides exact solutions to optimization problems, making it suitable for applications where accuracy is crucial. - IP can handle various types of constraints such as inequality, equality, and logical constraints. This makes it a versatile tool for solving optimization problems. - IP can solve complex problems faster than other optimization techniques.	- IP can become computationally intractable for larger problems, meaning the amount of time required to solve the problem grows exponentially with problem size. - The restriction that the decision variables must be integers can make it difficult to obtain feasible solutions for some problems. - Developing an IP model can be a time-consuming and challenging process, as it requires a good understanding of the problem and its constraints.
Lagrangian-based CP approach [41]	- LCP approach can quickly find good quality solutions to optimization problems, even in cases where traditional CP or mathematical programming methods fail. - It is able to handle large-scale optimization problems with thousands of variables and constraints. - It is a flexible approach that can handle a wide range of problems, including mixed-integer and nonlinear optimization problems.	- LCP can be computationally expensive, especially when solving large-scale problems, and may require high-performance computing resources. - It may not always converge to an optimal solution, especially when the relaxation parameters are not properly tuned. - The Lagrangian function in LCP can become quite complex, making it difficult to interpret the results and identify the sources of any errors or inaccuracies. - LCP often requires the tuning of parameters, such as the penalty function coefficients and Lagrange multipliers, which can be a time-consuming process.
IP/CP approach [42]	- IP/CP can handle a wide range of constraints, including both linear and nonlinear functions. - It can be very efficient in solving problems, particularly for small to medium-sized problems. - It can provide optimal solutions to complex problems that other methods may not be able to solve accurately	- IP/CP is based on linear programming, and it may not be suitable for non-linear problems or problems with continuous variables. - IP/CP problems can be computationally complex, particularly for large-scale problems, leading to slow or inefficient solutions. - Although IP/CP provides an exact solution, it may not always be practical and an approximate solution may be more efficient. - Sometimes, IP/CP may fail to find a feasible solution, resulting in an infeasible problem.
ILP program [20]	- ILP program can be applied to a wide range of problems, including scheduling, network optimization, and supply chain management, among others - It can handle problems with complex constraints and objectives, making them a flexible and powerful tool for optimization	- ILP relies on linear equations, which may not accurately represent some real-world problems. Thus, there may be situations where the optimal solution obtained may not be the best solution in practice. - Solving ILP problems can be computationally intensive, especially for large-scale problems. The computational complexity of ILP increases exponentially with the number of decision variables and constraints
Fixing algorithm [20]	- Fixing algorithm is relatively easy to implement and do not require extensive mathematical modeling or programming knowledge. - It can be adapted to handle a wide range of optimization problems and can be easily modified to accommodate different constraints and objectives.	- Fixing algorithm does not guarantee optimal solutions to the optimization problem. They may only provide good-quality solutions that are close to the optimal solution. - It may not be suitable for some types of optimization problems, especially those with complex constraints or objectives. - It may not be able to solve very large optimization problems due to computational limitations.

table 2 continued

Technique	Strengths	Limitations
MILP model [9, 43]	- MILP is an efficient algorithm for solving optimization problems that have linear constraints and integer variables - It can be applied to many types of problems, including resource allocation, scheduling, logistics, and financial planning	- MILP can become computationally expensive as the problem size increases, making it difficult to solve large-scale problems. - It is not suitable for non-linear optimization problems, which may require different techniques such as non-linear programming or heuristics. - It produces discrete solutions, which may not be suitable for problems that require continuous values.
Matheuristic strategies [9]	- Matheuristic strategies are highly flexible and can be adapted to different types of optimization problems. - They can often achieve high levels of accuracy in finding solutions, especially when compared to pure heuristic approaches. - They can solve complex optimization problems that are often intractable for mathematical programming or heuristic methods alone.	- Matheuristic strategies can be complex and difficult to implement, requiring specialized knowledge and expertise. - They often require careful parameter tuning to ensure optimal performance. This can be a time-consuming process. - They often lack theoretical guarantees of finding optimal solutions or bounds on their performance. - They may not always be applicable to all optimization problems. They may require a significant amount of problem-specific knowledge to be effective.
Constraint programming CP [17, 41, 43]	- CP can solve large-scale optimization problems with many constraints efficiently. - It can handle non-linear functions in the objective function and constraints, making it suitable for more complex problems. - It can handle complex constraints such as global constraints, symmetry-breaking constraints, and soft constraints.	- CP solutions can be less accurate than IP solutions, particularly in problems with many constraints. - It can require exploring a large search space to find optimal solutions, which can make it slower than other optimization techniques for some problems - Developing a CP model can be more challenging than developing an IP model, particularly for problems with complex constraints.
MILP/CP [43]	- MILP/CP combines the advantages of both MILP and CP, which can lead to better performance and higher-quality solutions. - It is well-suited for problems that have a mix of linear and non-linear constraints. - It can handle complex real-world problems that involve both discrete and continuous decision variables.	- MILP/CP is more complex than MILP, requiring expertise in both MILP and CP techniques. - The additional overhead of the CP solver can make MILP/CP slower than MILP for small-scale problems. - MILP/CP problems can still be computationally challenging, especially when the problem size increases.
Multi-pass heuristics algorithms [10]	- Multi-pass heuristics can improve the quality of the solution by refining the initial solution obtained from the first pass. - They can handle large-scale optimization problems with thousands of variables and constraints. - They can be used to solve a wide variety of optimization problems, including those with complex constraints and objectives.	- Multi-pass heuristics can be computationally expensive, particularly if each pass requires a significant amount of computation. - They may get trapped in a local optimum if the initial solution obtained from the first pass is not diverse enough. - They require careful tuning of parameters such as the number of iterations and the stopping criteria. - They may not always converge to the optimal solution and may require additional optimization techniques to improve convergence.

table 2 continued

Technique	Strengths	Limitations
NEH_{st} [10]	- NEH_{st} is a relatively simple algorithm that is easy to implement and understand. - It can produce high-quality solutions for permutation flow shop scheduling problems, often outperforming other heuristic algorithms.	- NEH_{st} is only applicable to permutation flow shop scheduling problems, and may not be useful for other types of scheduling problems. - NEH_{st} time complexity can be high, particularly for large-scale scheduling problems. - It may not always produce optimal solutions, particularly if the input data or problem parameters are uncertain or variable. - It is sometimes trapped in local optima and may not explore a diverse range of solutions.
NEH_{res} [10]	- NEH_{res} can be used with a variety of objective functions, including *makespan*, total flow time, and total tardiness. - It can be used to solve large-scale scheduling problems with hundreds or even thousands of jobs and machines. - It can be applied to a wide variety of scheduling problems, including those with multiple objectives and constraints.	- NEH_{res} performance can be sensitive to the order in which the jobs are processed, which can make it difficult to achieve optimal results. - It only explores a limited search space, which can prevent it from finding globally optimal solutions in certain cases. - It is a rigid algorithm that does not allow for easy modification or customization, which can limit its usefulness in certain applications.
SWA [10]	- SWA can often converge to a high-quality solution relatively quickly. - It is a relatively simple heuristic algorithm that is easy to implement and understand.	- SWA performance can be sensitive to the order in which the jobs are processed, which can be a disadvantage if the input data is poorly structured. - It is only applicable to scheduling problems, and may not be useful for other problems. - It occasionally sticks in local optima and might not explore a wide variety of solutions.
Enriched scatter search ESS [44]	- ESS can find high-quality solutions to complex optimization problems - It can be used to solve a wide range of optimization problems, including both continuous and discrete problems. - It can converge to the global optimum of the problem.	- ESS can be computationally expensive and time-consuming, especially for large-scale optimization problems. - The performance of this algorithm can be sensitive to the choice of the parameters - The convergence rate of the algorithm can be slow, especially for problems with complex solution spaces.
Enriched iterated greedy EIG [44]	- EIG is often able to find high-quality solutions quickly, even for large and complex problems - It can be applied to a wide range of optimization problems, including both continuous and discrete problems - It can be scaled up to large problem sizes with a large number of variables	- EIG can become trapped in local optima due to its lack of diversity in its search - The performance of the algorithm can be sensitive to the choice of the parameters - There is no guarantee that the algorithm will find the optimal solution to a given problem
GGA [45]	- GGA is capable of searching for the global optimum of a function in a large solution space, making it useful for problems with multiple local optima. - It can handle a wide range of optimization problems with different types of constraints and objective functions. - It can be easily parallelized, which can help reduce the optimization time for large-scale problems. - It does not require any knowledge of the gradients of the objective function, making it suitable for non-differentiable or complex gradients problems.	- GGA requires careful parameter tuning to achieve good performance, such as population size, mutation rate, and crossover rate. - It may converge to a suboptimal solution if the population diversity is not maintained or the genetic operators are not properly implemented.

table 2 continued

Technique	Strengths	Limitations
GGA-GD [46]	- GGA-GD provides a good balance between exploration and exploitation, allowing the algorithm to search a wide range of potential solutions. - The Great Deluge algorithm can help the algorithm to converge more quickly towards the optimal solution, particularly in cases where the search space is narrow or the optimization landscape is complex. - It is often robust to changes in the problem formulation, input data, or algorithm parameters. - It can be designed to handle large-scale optimization problems with thousands or even millions of variables	- GGA-GD often requires careful tuning of algorithm parameters to achieve good performance, which can be time-consuming and difficult. - It requires a more complex implementation than either GA or GDA alone, which can be challenging for some users
GGA-TS [46]	- GGA-TS can effectively explore a large search space and avoid local optima, resulting in better global optimization performance. - It can be customized to fit a wide range of optimization problems, and it can incorporate different variations of the genetic and tabu search algorithms. - It can efficiently explore the solution space and quickly converge to a good solution, especially for complex optimization problems. - It can handle noisy or uncertain input data and can adapt to changes in the problem formulation or search space.	- GGA-TS requires careful tuning of various parameters, such as the population size, crossover rate, and tabu search parameters, to achieve good performance. - It may not provide an easy-to-interpret solution, as the optimal solution may be a combination of the genetic and tabu search components. - It may struggle to converge to an optimal solution, particularly if the optimization landscape is complex or poorly understood.
GGA-VNS [46]	- GGA-VNS can maintain diversity by using the genetic algorithm to explore a wide range of solutions and the variable neighborhood search algorithm to refine promising solutions. - It is flexible and can be easily adapted to different optimization problems by modifying the genetic algorithm or variable neighborhood search algorithm parameters. - It can find good quality solutions quickly by using the genetic algorithm to explore the search space and the variable neighborhood search algorithm to refine the solutions.	- GGA-VNS may not always converge to an optimal solution, particularly if the optimization landscape is complex or poorly understood. - It can be complex and difficult to implement, particularly if the optimization problem involves complex constraints or objectives. - It requires careful tuning of parameters to achieve good performance, which can be time-consuming and difficult.

In summary, careful consideration of the advantages and disadvantages of different approaches is essential before selecting an optimization technique. Exact methods such as Integer Programming, Lagrangian-based CP approach, IP/CP approach, ILP program, fixing algorithm, MILP model, Constraint programming, and MILP/CP provide a flexible framework for modeling various types of optimization problems. However, they may be computationally complex and sensitive to problem formulation. On the other hand, heuristic algorithms like NEHst are simple and effective for permutation flow shop scheduling problems but may not be suitable for other types of scheduling problems and may be sensitive to input data. Multi-pass heuristics, such as M1, M2, M3, M4, and M5, require careful design and parameter tuning for good performance and may be computationally expensive, produce complex solutions that are difficult to interpret, and get trapped in local optima. Approximation algorithms like the 2-approximation algorithm, 3/2-approximation algorithm, 4-approximation algorithm, 6.83-approximation algorithm, and 3.75-approximation algorithm are simple, efficient, and robust. However, they may not produce the optimal solution, lack flexibility, and their performance may be sensitive to input characteristics. Metaheuristic algorithms, such as Enriched scatter search, enriched iterated greedy, guided genetic algorithm, and hybrid guided genetic algorithm, have their

advantages and disadvantages. It is important to carefully evaluate the problem and the available algorithms to determine the best approach for finding the optimal solution. Matheuristic strategies such as JMR-P, MAF-P, GBF-P, JMR-S, MAF-S, and GBF-S are powerful tools that can solve complex optimization problems, but they require careful consideration and expertise to use effectively.

5. CONCLUSION

Viewed as a decision-making form, scheduling is seen as a crucial dimension in most forms of manufacturing, production and information processing situations and contexts. In addition, scheduling is also vital in situations involving transportation, distribution and service-related industries. Over the past 70 years, researchers in management, industrial engineering, operations research, and computer science have examined scheduling extensively. Today, there is an amazing body of information in this area. The investigation of UPMR is rendered an important area of research. After presenting the key definitions of concepts, assumptions, and categorizations in this article, the relevant studies were reviewed and organized into an effective framework according to the machine environments, objective functions, additional resource characteristics, complexity, and solution approaches. Next, for the PMS problem, a few model extensions have been revealed and explained. It has been shown the majority of the problems, in question, have been faced in actual-world problems and call for the use of industrial-sized data in order for them to be solved, but nearly no one study in the survey addresses these problems. This is an important point to note in addition to the vital features of the articles revealed above.

Funding

None

ACKNOWLEDGEMENT

We are grateful to Universiti Malaysia Pahang (UMP) for supporting the research project through University Postgraduate Research Grant Scheme (PGRS1903187).

CONFLICT OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] L. Fanjul Peyro and R. Ruiz, "Iterated greedy local search methods for unrelated parallel machine scheduling," *European Journal of Operational Research*, vol. 207, no. 1, pp. 55-69, 2010, doi: 10.1016/j.ejor.2010.03.030.

[2] L. Fanjul-Peyro and R. Ruiz, "Size-reduction heuristics for the unrelated parallel machines scheduling problem," *Computers & Operations Research*, vol. 38, no. 1, pp. 301-309, 2011, doi: 10.1016/j.cor.2010.05.005.

[3] E. Vallada and R. Ruiz, "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times," *European Journal of Operational Research*, vol. 211, no. 3, pp. 612-622, 2011, doi: 10.1016/j.ejor.2011.01.011.

[4] F. J. Rodriguez, M. Lozano, C. Blum, and C. GarcíA-MartíNez, "An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem," *Computers & Operations Research*, vol. 40, no. 7, pp. 1829-1841, 2013, doi: 10.1016/j.cor.2013.01.018.

[5] J. E. C. Arroyo and J. Y.-T. Leung, "Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times," *Computers & Operations Research*, vol. 78, pp. 117-128, 2017, doi: 10.1016/j.cor.2016.08.015.

[6] M. Pinedo, "Scheduling: theory, algorithms and applications," *Scheduling: Theory, algorithms and applications, Prentice-Hall, Englewood Cliffs, NJ*, 1995, doi: 10.1007/978-3-642-46773-8_5.

[7] J. Blazewicz, N. Brauner, and G. Finke, "Scheduling with Discrete Resource Constraints," in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, E. J. Y-T. Leung, Ed., ed. USA: CRC Press, 2004.

[8] A. Grigoriev, M. Sviridenko, and M. Uetz, "Unrelated parallel machine scheduling with resource dependent processing times," *Lecture notes in computer science*, pp. 182-195, 2005, doi: 10.1007/11496915_14.

[9] L. Fanjul Peyro, F. Perea, and R. Ruiz, "Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources," *European Journal of Operational Research*, vol. 260, no. 2, pp. 482-493, 2017, doi: 10.1016/j.ejor.2017.01.002.

[10] F. Villa, E. Vallada, and L. Fanjul Peyro, "Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource," *Expert Systems with Applications*, vol. 93, pp. 28-38, 2018, doi: 10.1016/j.eswa.2017.09.054.

[11] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," in *Annals of discrete mathematics*, vol. 5: Elsevier, 1979, pp. 287-326, doi: 10.1016/S0167-5060(08)70356-X.

[12] J. Blazewicz, J. K. Lenstra, and A. R. Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete applied mathematics*, vol. 5, no. 1, pp. 11-24, 1983, doi: 10.1016/0166-218X(83)90012-4.

[13] E. B. Edis, "Resource constrained parallel machine scheduling problems with machine eligibility restrictions: Mathematical and constraint programming based approaches," DEÜ Fen Bilimleri Enstitüsü, 2009.

[14] E. Mokotoff, "An exact algorithm for the identical parallel machine scheduling problem," *European Journal of Operational Research*, vol. 152, no. 3, pp. 758-769, 2004, doi: 10.1016/S0377-2217(02)00726-9.

[15] A. H. Salem, *Unrelated parallel machine scheduling with sequence-dependent setup times and machine eligibility restrictions for minimizing the makespan*. University of Central Florida, 1999.

[16] M. Pinedo, "Scheduling: theory, algorithms, and systems Springer Science & Business Media," ed: LLC, 2012.

[17] T. Arbaoui and F. Yalaoui, "Solving the Unrelated Parallel Machine Scheduling Problem with Additional Resources Using Constraint Programming," in *Asian Conference on Intelligent Information and Database Systems*, 2018, pp. 716-725: Springer, doi: 10.1007/978-3-319-75420-8_67.

[18] A. Mensendiek, J. N. Gupta, and J. Herrmann, "Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness," *European Journal of Operational Research*, vol. 243, no. 2, pp. 514-522, 2015, doi: 10.1016/j.ejor.2014.12.002.

[19] W.-C. Yeh, M.-C. Chuang, and W.-C. Lee, "Uniform parallel machine scheduling with resource consumption constraint," *Applied Mathematical Modelling*, vol. 39, no. 8, pp. 2131-2138, 2015, doi: 10.1016/j.apm.2014.10.012.

[20] L. Fanjul-Peyro, F. Perea, and R. Ruiz, "Algorithms for the unspecified unrelated parallel machine scheduling problem with additional resources," in *Industrial Engineering and Systems Management (IESM), 2015 International Conference on*, 2015, pp. 69-73: IEEE, doi: 10.1109/IESM.2015.7380139.

[21] D. Yilmaz Eroglu, H. C. Ozmutlu, and S. Ozmutlu, "Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times," *International Journal of Production Research*, vol. 52, no. 19, pp. 5841-5856, 2014, doi: 10.1080/00207543.2014.920966.

[22] J. Blazewicz, Cellary, W., Slowinski, R., & Weglarz, J., *Scheduling under resource constraints: Deterministic models*. JC BaltzerAG, Scientific Publishing Company., 1986.

[23] J. Blazewicz, k. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, "Scheduling under Resource Constraints," *Handbook on scheduling: from theory to applications. New York: Springer: Verlag Berlin Heidelberg*, pp. 425-475, 2007, doi: 10.1007/978-3-540-32220-7_12.

[24] E. B. Edis, C. Oguz, and I. Ozkarahan, "Parallel machine scheduling with additional resources: Notation, classification, models and solution methods," *European Journal of Operational Research*, vol. 230, no. 3, pp. 449-463, 2013, doi: 10.1016/j.ejor.2013.02.042.

[25] R. Słowiński, "Two approaches to problems of resource allocation among project activities—a comparative study," *Journal of the Operational Research Society*, vol. 31, no. 8, pp. 711-723, 1980, doi: 10.1057/jors.1980.134.

[26] J. A. Ventura and D. Kim, "Parallel machine scheduling about an unrestricted due date and additional resource constraints," *Iie Transactions*, vol. 32, no. 2, pp. 147-153, 2000, doi: 10.1023/A:1007662314880.

[27] D. Shabtay and M. Kaspi, "Parallel machine scheduling with a convex resource consumption function," *European Journal of Operational Research*, vol. 173, no. 1, pp. 92-107, 2006, doi: 10.1016/j.ejor.2004.12.008.

[28] L. Özdamar and G. Ulusoy, "A local constraint based analysis approach to project scheduling under general resource constraints," *European Journal of Operational Research*, vol. 79, no. 2, pp. 287-298, 1994, doi: 10.1016/0377-2217(94)90359-X.

[29] J. Y. Leung, *Handbook of scheduling: algorithms, models, and performance analysis*. CRC press, 2004.

[30] C. A. Glass, Y. M. Shafransky, and V. A. Strusevich, "Scheduling for parallel dedicated machines with a single server," *Naval Research Logistics (NRL)*, vol. 47, no. 4, pp. 304-328, 2000, doi: 10.1002/(SICI)1520-6750(200006)47:4<304::AID-NAV3>3.0.CO;2-1.

[31] N. G. Hall, C. N. Potts, and C. Sriskandarajah, "Parallel machine scheduling with a common server," *Discrete Applied Mathematics*, vol. 102, no. 3, pp. 223-243, 2000, doi: 10.1016/S0166-218X(99)00206-1.

[32] M. Pinedo, "Scheduling: theory, and systems (3rd ed.)," ed: Springer, Science+Business Media, USA, 2008.

[33] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules: J.F. Muth, G.L. Thompson (eds.)," *Industrial scheduling, Prentice Hall, Englewood Cliffs, New Jersey*, pp. 225-251, 1963,

[34] S. French, "Sequencing and scheduling," *An Introduction to the Mathematics of the Job-Shop: Ellis Horwood Chichester*, 1982, Ellis Horwood Chichester,

[35] J.-F. Chen, "Unrelated parallel machine scheduling with secondary resource constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 285-292, 2005, doi: 10.1007/s00170-003-1622-1.

[36] J.-F. Chen and T.-H. Wu, "Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints," *Omega*, vol. 34, no. 1, pp. 81-89, 2006, doi: 10.1016/j.omega.2004.07.023.

[37] E. B. Edis and I. Ozkarahan, "A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions," *Engineering Optimization*, vol. 43, no. 2, pp. 135-157, 2011, doi: 10.1080/03052151003759117.

[38] E. B. Edis and I. Ozkarahan, "Solution approaches for a real-life resource-constrained parallel machine scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 58, no. 9-12, pp. 1141-1153, 2012, doi: 10.1007/s00170-011-3454-8.

[39] V. A. Kumar and M. V. Marathe, "Approximation algorithms for scheduling on multiple machines," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, 2005, pp. 254-263: IEEE, doi: 10.1109/SFCS.2005.21.

[40] A. Grigoriev, M. Sviridenko, and M. Uetz, "LP rounding and an almost harmonic algorithm for scheduling with resource dependent processing times," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*: Springer, 2006, pp. 140-151, doi: 10.1007/11830924_15.

[41] E. B. Edis and C. Oguz, "Parallel Machine Scheduling with Additional Resources: A Lagrangian-Based Constraint Programming Approach," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 92-98, 2011, doi: 10.1007/978-3-642-21311-3_10.

[42] E. B. Edis and C. Oguz, "Parallel machine scheduling with flexible resources," *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 433-447, 2012, doi: 10.1016/j.cie.2012.03.018.

[43] K. Fleszar and K. S. Hindi, "Algorithms for the unrelated parallel machine scheduling problem with a resource constraint," *European Journal of Operational Research*, vol. 271, no. 3, pp. 839-848, 2018, doi: 10.1016/j.ejor.2018.05.056.

[44] E. Vallada, F. Villa, and L. Fanjul-Peyro, "Enriched metaheuristics for the resource constrained unrelated parallel machine scheduling problem," *Computers & Operations Research*, vol. 111, pp. 415-424, 2019, doi: 10.1016/j.cor.2019.07.016.

[45] M. H. Abed and M. N. M. Kahar, "Guided genetic algorithm for solving unrelated parallel machine scheduling problem with additional resources," *Indones. J. Electr. Eng. Comput. Sci*, vol. 26, no. 2, pp. 1036-1049, 2022, doi: 10.11591/ijeecs.v26.i2.

[46] M. H. Abed and M. N. M. Kahar, "Hybridizing genetic algorithm and single-based metaheuristics to solve unrelated parallel machine scheduling problem with scarce resources," *Indones. Int J Artif Intell*, vol. 12, no. 1, pp. 315-327, 2023, doi: 10.11591/ijai.v12.i1.