IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received February 22, 2021, accepted March 3, 2021, date of publication March 10, 2021, date of current version March 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065109

An Evolutionary Variable Neighbourhood Search for the Unrelated Parallel Machine Scheduling Problem

SALWANI ABDULLAH[1], AYAD TURKY[2], MOHD ZAKREE AHMAD NAZRI[1], AND NASSER R. SABAR[3]

[1]Centre for Artificial Intelligence Technology, Data Mining and Optimisation Research Group, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia
[2]College of Engineering and Science, Victoria University, Melbourne, VIC 3011, Australia
[3]Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3083, Australia

Corresponding author: Nasser R. Sabar (n.sabar@latrobe.edu.au)

This work was supported by the Universiti Kebangsaan Malaysia under Grant DIP-2016-024.

ABSTRACT This article addresses a challenging industrial problem known as the unrelated parallel machine scheduling problem (UPMSP) with sequence-dependent setup times. In UPMSP, we have a set of machines and a group of jobs. The goal is to find the optimal way to schedule jobs for execution by one of the several available machines. UPMSP has been classified as an NP-hard optimisation problem and, thus, cannot be solved by exact methods. Meta-heuristic algorithms are commonly used to find sub-optimal solutions. However, large-scale UPMSP instances pose a significant challenge to meta-heuristic algorithms. To effectively solve a large-scale UPMSP, this article introduces a two-stage evolutionary variable neighbourhood search (EVNS) methodology. The proposed EVNS integrates a variable neighbourhood search algorithm and an evolutionary descent framework in an adaptive manner. The proposed evolutionary framework is employed in the first stage. It uses a mix of crossover and mutation operators to generate diverse solutions. In the second stage, we propose an adaptive variable neighbourhood search to exploit the area around the solutions generated in the first stage. A dynamic strategy is developed to determine the switching time between these two stages. To guide the search towards promising areas, a diversity-based fitness function is proposed to explore different locations in the search landscape. We demonstrate the competitiveness of the proposed EVNS by presenting the computational results and comparisons on the 1640 UPMSP benchmark instances, which have been commonly used in the literature. The experiment results show that our EVNS obtains better results than the compared algorithms on several UPMSP instances.

INDEX TERMS Machine scheduling problem, genetic algorithm, local search algorithm.

I. INTRODUCTION

Increasing productivity and minimising cost are the most challenges issues faced by manufacturing companies. Effectively handling these issues can significantly improve companies business sustainability. The machine scheduling problem (MSP) is the most critical task in production planning and control, which has a direct impact on productivity and production costs. The main goal is to reduce production completion times by efficiently allocating all resources in order to improve customer satisfaction, production costs and delivery. Due to its importance, application and complexity, MSP has been extensively researched in the literature and many variants were introduced [1]–[3]. This article addresses a particular variant of MSP known as the unrelated parallel machine scheduling problem (UPMSP) with sequence-dependent setup times [4]–[6]. UPMSP schedules a number of jobs for execution on a group of machines. The objective is to minimise the schedule's completion time (aka the makespan). UPMSP has several real-world applications such as ceramics plant, mail facilities, semiconductor manufacturing, sport tournaments, block erection scheduling in a shipyard, automobile gear manufacturing process, steel making industry, chemical processes, hospital operating rooms management and human resources [7]–[12].

Due to UPMSP NP-hardness [4], exact algorithms or mathematical approaches can be computationally infeasible for large scale UPMSP instances. Meta-heuristics (or

The associate editor coordinating the review of this manuscript and approving it for publication was Pengcheng Liu.

approximate algorithms), therefore, are widely used to deal with large scale problem instances [4]. These algorithms are able achieve good results within acceptable amount of time [13]–[18]. In the literature, different hybrid meta-heuristic algorithms that combine two or more algorithms have been proposed to address UPMSP. Vallada and Ruiz [4] introduced large-scale UPMSP instances and presented a hybrid meta-heuristic algorithm to solve these instances. Their algorithms combine the genetic algorithm with the local search method. An enhanced crossover operator was developed for the UPMSP instances. Cota *et al.* [19] proposed a heuristic algorithm that uses an iterated local search method and path relinking algorithm to tackle the UPMSP instances. Santos *et al.* [20] investigated the performance of four different local search methods to solve the UPMSP. Generally speaking, only a few meta-heuristic algorithms have been applied on UPMSP benchmark instances introduced by [4]. This might be due to these instances' complexity and size, which serves as the main motivation for this work to consider this challenging UPMSP benchmark.

Despite the success of the aforementioned methods on UPMSP, no single method is able to consistently work well across all UPMSP problem instances. Furthermore, large-scale UPMSP instances present a great challenge to optimisation algorithms. Hence, this work proposes a hybrid evolutionary variable neighbourhood search (EVNS) approach to effectively solve large-scale UPMSP instances. Several new components have been introduced and combined adaptively to handle the large-scale search spaces. The proposed approach iteratively alternates between two stages to effectively explore search space and exploit the areas around the new solution. In the first stage, we propose an evolutionary descent algorithm that uses a mix of evolutionary operators (crossover and mutation) as an exploration algorithm to evolve high quality and diverse solutions for the second stage. In the second stage, an adaptive variable neighbourhood search is proposed to exploit the area around the solution generated by the first stage. A dynamic strategy is proposed to determine the switching time between these two stages. To guide EVNS towards promising regions in the search space, we devise a diversity-based fitness function to decide which area should be visited. The main contributions of this study are:

1) development of a new hybrid approach to effectively solve large-scale UPMSP benchmark instances that combine various algorithmic components in an adaptive manner.
2) development of an evolutionary descent algorithm that uses a mix of evolutionary operators to effectively explore the search space.
3) development of a new diversity-based fitness function to intelligently guide the search towards promising areas.
4) development of an adaptive variable neighbourhood search algorithm that employs different customised problem specific neighbourhood operators.
5) introducing a dynamic strategy to adaptively switch between the stages of the proposed hybrid approach.
6) extensive computational experiments were performed on UPMSP large-scale instances introduced by [4]. The results demonstrate that our algorithm obtains better results compared to the state-of-the-art approaches.

TABLE 1. The nomenclature.

Variable	Description
n	a group of jobs, J={$j_1, j_2 \ldots j_n$}
M	a group of unrelated machines, M={m_1,$m_2 \ldots m_M$}
p_{jm}	the processing time of j^{th} job
S_{ijm}	job setup time
S_{0jm}	machine setup time
x_{ijk}	a binary indicator for job sequencing
C	makespan or the completion time of the last job

The rest of the paper is organised as follows. The formulation and definition of the UPMSP are discussed in Section II. The proposed approach is explained in Section III. In Section IV, we present the experiment settings and the results. The conclusions and future work are presented in Section V.

II. PROBLEM DESCRIPTION

This section describes the formulation and definition of unrelated parallel machine scheduling problem (UPMSP) [4]. We first present the problem nomenclature.

In UPMSP n and M are denoted as follows: $J = \{j_1, j_2 \ldots j_n\}$ and $M = \{m_1, m_2 \ldots m_M\}$. Each $j \in J$ is associated with p_{jm} which depends on the allocated m. Each $j \in J$ has S_{ijm} which depends on the sequence and m where i is the sequence of m. Each $m \in M$ has S_{0jm} to execute the first job. $x_{ijk} = 1$ if j_i precedes j_k on m_i; otherwise $x_{ijk} = 0$. The UPMSP allocates all J into a set of M unrelated machines such that

- Each $j \in J$ is allocated to one machine only and executed exactly once.

$$\sum_{j \in M} \sum_{j \in \{0\} \cup \{N\} j \neq k} x_{ijk} = 1 \quad \forall k \in N \tag{1}$$

- Each $m \in M$ can process only one j at a time and cannot be interrupted until completing the current j.

$$\sum_{j \in M} \sum_{k \int N j \neq k} x_{ijk} \leqslant 1 \quad \forall k \in N \tag{2}$$

- All jobs must be properly allocated into machines in a predefined manner. That is if j is allocated to m_i, a predecessor h must exist on the same m.

$$\sum_{k \in \{0\} \cup \{N\} h \neq k h \neq j} x_{ihj} \geqslant x_{ijk} \quad \forall j, k \in N \; j \neq k \; \forall i \in M \tag{3}$$

- There are setup times (S_{ijk}) between jobs allocated to the same m in which C of j_j is greater than j_i.

$$C_{ik} + V(1 - x_{ijk}) \geqslant C_{ij} + S_{ijk} + P_{ik} \quad j \in \{0\} \cup \{N\}$$
$$\forall k \in N \quad j \neq k \quad \forall i \in M$$
$$V = constant \tag{4}$$

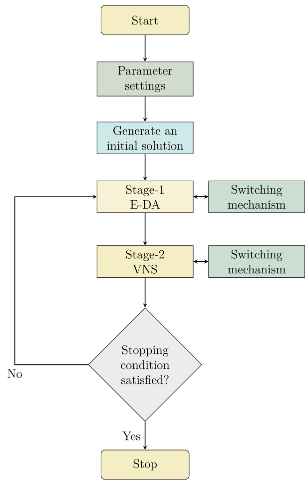

FIGURE 1. **Flowchart of the proposed approach.**

The objective of optimising the UPMSP is to minimise C of the last job (makespan) [4]:

$$C_j = Max(r_j, C_{im} + S_{ijm}) + p_{jm} \tag{5}$$

III. METHODOLOGY

Hybrid meta-heuristic approaches are known as effective approaches to deal with hard optimisation problems [21], [22]. These approaches hybridise various algorithmic components in order to obtain the best performance. This work proposes a new two-stage hybrid meta-heuristic approach to solve the UPMSP. The proposed approach hybridises various algorithmic components in an adaptive fashion to effectively and effectively deal with the search space of the UPMSP. The proposed two-stage approach works in an iterative manner where the first one focuses on exploration whereas the second stage tries to exploit the areas around the explored solution. Both stages work in a collaborative manner to effectively explore the solution spaces. We propose an evolutionary descent algorithm (denoted as E-DA) as an exploration algorithm (Stage-1) and adaptive problem-specific operators based on VNS to exploit the areas around the current solution (Stage-2). To ensure we have a proper balance between Stage-1 (exploration) and Stage-2 (exploitation), the approach adaptively alternates between them based on the search performance and status. For this purpose, we introduce an effective switching mechanism to decide which stage should be executed first. A diversity-based fitness function is proposed to direct the searching process into promising areas. Figure (1) presents an overview of our hybrid approach. It has three main components: (1) Stage-1, (2) Stage-2, and (3) switching mechanisms.

The approach first sets the parameter values and then generates an initial starting solution. Next, it enters the main loop in an iterative manner. It first executes Stage-1 (E-DA) to generate a new diverse solution. Next, it calls Stage-2 (VNS algorithm) to further improve the solution generated by Stage-1. The proposed approach checks the switching mechanism of each stage to determine when the current stage should be terminated and the search should move to the other stage. The proposed approach keeps alternating between Stage-1 and Stage-2 for a specific number of iterations (satisfying the stopping condition). In this work the search space of our approach is limited to feasible solutions only. This means any infeasible solutions generated during the search process will be immediately discarded.

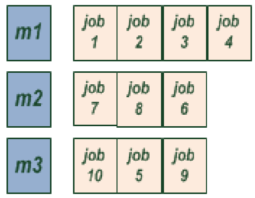

FIGURE 2. **Example of solution representation.**

In the following subsections, we present and discuss in details the main components and the steps of our approach.

A. PARAMETER SETTINGS

The first step of the proposed approach initialises the value of parameters. These are:

- The number of solutions *(S_No.)* - indicate the number of solutions that will be created by the applied operator.
- The probability of mutating a job *(Pm)* - represent the possibility of moving (or swapping) the current job to a different location.
- The maximum number of iterations *(I)* - indicate when the search process will stop.

In this article, the values of the aforementioned parameters were determined by preliminary experiments. Further detail is presented in section IV-A.

B. INITIAL SOLUTION

In our approach, each solution is represented by a group of m arrays. Each array represents the processing order of the set of j_J allocated to this particular m, as shown in Figure 2. In this figure, we have three machines (denoted as M_1, M_2 and M_3) and ten jobs (denoted as $J_1, J_2, \ldots, J_{10}$). From the figure, we can see that jobs J_1, J_2, J_3 and J_4 are assigned to M_1, jobs J_7, J_8 and J_6 are assigned to M_2 and jobs J_{10}, J_5 and J_9 are allocated to M_3. The initial solution is constructed in a random way where all jobs have the same selection probability of being allocated to machines.

C. STAGE-1

The first stage (Stage-1) of our approach is responsible for the exploration process. It searches for new areas in the solution spaces to provide a new solution for the second stage (Stage-2). To this end, we propose an evolutionary descent algorithm (denoted as E-DA) as an exploration algorithm to be used in Stage-1 of the proposed approach. The proposed E-DA uses two solutions as an input (best and current) and various evolutionary operators (crossover operators and mutation operators) to navigate different regions of the search landscape and avoid the local attraction points. In E-DA, an application of the operator generates *S_No.* solutions.

Following the (1 + $S\_No.$) rule, the best solution among the current solutions and its $S\_No.$ is selected as the starting solution for the next iteration. To avoid revisiting the same areas or preventing the search from oscillating around an already explored area, a quality-diversity-based fitness function is proposed to intelligently guide the evolutionary process into a new, yet effective, solution in the search landscape. We propose a switching mechanism to decide when E-DA should be terminated and the search should execute Stage-2. Both evolutionary operators and the quality-diversity- based fitness function ensures the search is of high quality and provides a diverse solution for the second stage of the proposed approach.

The main components of Stage-1 including the decent algorithm, the evolutionary operators for the UPMSP and the proposed quality-diversity-based fitness function are presented in the following subsections.

1) EVOLUTIONARY DESCENT ALGORITHM

The proposed evolutionary descent algorithm (E-DA) is a single-solution-based algorithm [23]. E-DA uses a set of evolutionary operators to jump out of the local optimum point and to effectively navigate the solution spaces. The main idea of E-DA is that the local optima solution obtained by the current operator is not necessarily a local optimum for a different operator and thus using a set of operators instead of one can help the search to obtain a high-quality solution. The proposed E-DA begins with a single initial solution, randomly generated or by a heuristic method, and then tries to improve it using a set of evolutionary operators applied in a pre-defined manner. E-DA first randomly makes a list of evolutionary operators based on the associated probability and then applies the first one as long as it can evolve a better solution. If the current operator cannot generate a better solution, E-DA will stop using it and will restart the search using the next operator in the list. E-DA will halt the search process if the last evolutionary operator in the list has been called and no further improvement can be achieved. In this work, the proposed E-DA uses various evolutionary operators (crossover operators and mutation operators) as search operators. The pseudocode of E-DA is presented in Algorithm 1.

Algorithm 1: Evolutionary Descent Algorithm (E-DA)

Let K be the number of the evolutionary operators (NS);
Set i ← 1;
S_0 ← Generate Initial Solution();
while $i < k$ **do**
  S_1 ← Generates a neighbourhood of S_0 using NS_i;
  if $f(S_1) < f(S_0)$ **then**
    $S_0 \leftarrow S_1$;
    $i = 1$;
  end
  else
    $i = i + 1$
  end
end
Return the best solution

2) EVOLUTIONARY OPERATORS

Several various evolutionary operators are implemented as neighbourhood operators for the evolutionary descent algorithm. The utilised operators (crossover operators and mutation operators) have different characteristics to explore the search space. Crossover operators evolve new solutions around the current parent by combining the characteristics of parent solutions. Mutation operators work as exploration procedures to introduce diversity into the search process. This work makes use of the following crossover and mutation operators:

- *Single-Point Crossover:* This operator first randomly select a single point to divide the parent. Then, the group of J on the left-side is moved from the parent to the child, and the other J are allocated in order of their appearance in the other parent. An example is depicted in Figure 3.

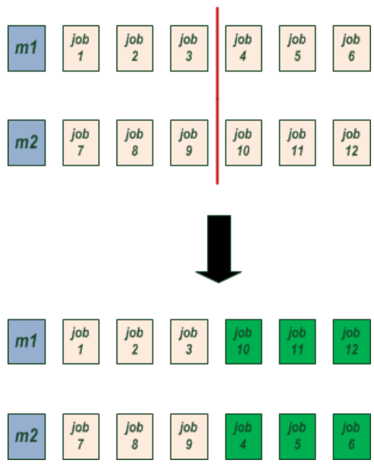

FIGURE 3. Example of a single-point crossover.

- *Two-Point Crossover:* Two-point crossover is a generalisation of the one-point crossover wherein alternating segments are swapped to get new offsprings. Figure 4 shows an example of a two-point crossover.
- *Swap Mutation:* Two different J are randomly selected from two different parents and interchanged.
- *Shift Mutation:* Select a random job and a random target position, then shift the selected job to the selected target position.

3) QUALITY-DIVERSITY-BASED FITNESS FUNCTION

In optimisation, the performance of the search algorithm is evaluated using the so-called fitness function. It compares the quality of the evolved solution against the current one and determines whether the new one should be accepted. It guides the search process in deciding which area should be visited and which area should be discarded. Most of the existing fitness functions only use solution quality as an indicator of the acceptance or rejection of the new solution. However, in many combinatorial optimisation problems such

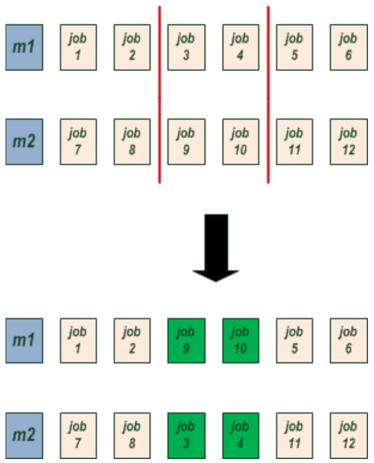

FIGURE 4. Example of two-point crossover.

as the UPMSP, two solutions might have the same quality but different structures in terms of which jobs have been allocated to which machines. In such a case, the quality indicator is not very effective to guide the search process as it might discard the new solution even if it is located in a different area in comparison to the current one. To address this issue, we propose a quality-diversity-based fitness function for the E-DA algorithm. The proposed fitness function has two parts: the quality part and the diversity part. The quality part ($f(x_{new})$) is calculated using Equation (5). The diversity part ($Dis(x, x_{new})$) uses a matching function to calculate the difference between two solutions. It counts how many jobs in both solutions have been allocated to different machines. The value returned by the count indicates if the two solutions are similar, close to each other or highly different. A high count value means the two solutions are very different while a small value indicates they are very similar. The proposed fitness function is presented in Equation (6).

$$F_t = f(x_{new}) + \frac{1}{Dis(x, x_{new})} \tag{6}$$

where $f(x)$ represents value of the new solution x_{new}. $Dis(x, x_{new})$ refers to dissimilarity count value between solution x and the new solution x_{new}. The proposed fitness function seeks to minimise the quality of the evolved solution and maximise the difference between the evolved one and the other solution. A new solution will be accepted if its quality is better than the previous one and different from the previous one.

4) SWITCHING MECHANISM

The main purpose of the proposed switching mechanism is to decide when Stage-1 should be terminated. Once stage-1 is terminated, the proposed approach calls Stage-2 to further improve the solution returned by Stage-1. In this work, we use solution diversity as an indicator of when Stage-1 should be halted. The diversity indicator starts high and gradually decreases as the approach progress. This can help the proposed approach to begin with a highly diverse solution at the early iterations and then a less diverse solution at the late iterations to allow the search to focus on examining the neighbourhood areas. The proposed switching mechanism first calls the diversity indicator to calculate how different the current solution returned by E-DA is from the initial solution using the matching function ($Dis(x_{int}, x_{E-DA})$)) which counts how many jobs in both solutions have been allocated to different machines. The switching mechanism will be triggered and Stage-1 will be terminated if Dis is greater or equal to Ind. Ind is an iteration-based indicator that takes a high value at the beginning and then gradually decreases based on the algorithm iteration as follows:

$$Ind = (Max - Min)\frac{(Max_{iter} - iter)}{Max_{iter}} + Min \tag{7}$$

where Min and Max are constants and their values were set based on the minimum and the maximum diversity between the current solution and the initial solution (see Section III-C3). Max_{iter} is the total number of iterations and $iter$ represents the current iteration number.

D. STAGE-2

This work proposes a variable neighbourhood search (VNS) algorithm as an effective exploitation method for the second stage of the proposed approach. VNS is a well-known method for solving difficult combinatorial optimisation problems [23]. VNS uses more than one neighbourhood operators to move out of the local minimal. It iteratively alternates between two procedures known as the shaking procedure and the local search (LS) procedure until reaching the pre-defined stopping condition. The shaking procedure provides a new diverse starting solution for the LS procedure to exploit its neighbourhood areas. The shaking procedure plays a crucial role on the algorithm performance as it determines in which region of the search space the newly generated solution will be located. This is because the performance of LS procedure highly affected by the solution provided by the shaking procedure. To this end, we propose an adaptive VNS that uses various operators in an adaptive manner to efficiently deal with UPMSP large-scale instances. The steps of VNS are given in Algorithm 2.

- *Shaking Procedure:* This procedure is responsible for evolving a new solution in the k^{th} neighbourhood of the current solution. It will be used as a new starting solution for the LS procedure. A new solution is created through the application of a neighbourhood operator which perturb the current one to generate a new one in the neighbourhood area. The success of VNS is highly affected by the applied shaking procedure because it guides the search of local search procedure to exploit new areas around the current solution. Thus, devising an effective and efficient shaking procedure can greatly improve the VNS performance. In UPMSP, different instances have different characteristics and size. This indicates that a specific neighbourhood operator suits certain instances only and might not work well over other instances.

Algorithm 2: Variable Neighbourhood Search (VNS)

Select the set of neighbourhood structures N_k, $k = 1, \ldots, K$ *where* $K = 5$;
Set Stopping condition (I);
$S_0 \leftarrow$ Generate Initial Solution();
$k \leftarrow$ Neighbourhood operators ();
$i \leftarrow 1$;
while *Stopping condition is not met* **do**
  $S_1 \leftarrow$ Shaking(S_0, k);
  $S_2 \leftarrow$ Local search ($S_1, k[i]$);
  if S_2 *has a lower cost value than* S_0 **then**
    $S_0 \leftarrow S_2$
    $i = 1$;
  end
  else
    $i = i + 1$
  end
end
Return the best solution

To this end, we propose an adaptive shaking procedure to evolve a new solution. It employs five different neighbourhood operators (described below) in an adaptive manner. We first allocate a ranking value for each neighbourhood operator. Then, based on allocated rank, an adaptive selection mechanism is used to create a pool of operators. The adaptive mechanism has two parts: ranking and selection. For the ranking part, the neighbourhood operators are ranked using the distance metric ($D(x, x_a)$). The D metric counts the percentage of the object in solution x modified by the current neighbourhood operator compared to the input solution x_a. A high metric value indicates that the generated solution is placed in a different area. This work uses the quality of the solution as a secondary metric whereas all solutions are first ranked based on the distance metrics and then quality values. That is we prefer a solution that has high distance metrics and good quality value. For the selection part, we use the Boltzmann selection mechanism. In this mechanism, a neighbourhood operator a at iteration i is chosen based on a probability from the Boltzmann distribution:

$$P(a_i) = \frac{\exp^{Q(a)/t}}{\sum_{i=1}^{n} Q(a)_i} \tag{8}$$

where $n = 5$ refers to the number of neighbourhood operators. t represents the temperature. If $t = 0$ the selection process will be greedy, while the high value will make it equiprobable. We set the initial t value to 50% of the value of the initial solution. This will be updated during the search using a decay rate of 0.08. Using the Boltzmann selection mechanism a neighbourhood operator will be applied to evolve a new diverse solution. In this work, the new solution will be used in the second procedure regardless of its quality.

- *Local Search (LS) Procedure:* In this procedure, the solution produced by the shaking procedure is further improved. We first randomly create a sequence of neighbourhood operator. The local search procedure applies the first neighbourhood operator in the list as long as it generates a better solution. It applies the next neighbourhood operator in the list if the current one cannot generate a better solution. If we get an improvement, LS returns to the first neighbourhood operator and updates the best known value. LS procedure stops if the last neighbourhood operator in the list has been applied without any improvement.

The utilised neighbourhood list contains a pool of problem-specific neighbourhood operators. These operators evolve neighbourhood solutions by modifying the current one, while respecting problem constraints. Various neighbourhood operators that complement each other have been used. A set of problem-specific neighbourhood operators were implemented and customised to exploit problem knowledge. These operators are:

- *Shift:* A solution is generated by randomly rescheduling a job to another position on the same machine. Figure 5 gives an example of this operator.
- *Two-Shift:* A solution is constructed via randomly rescheduling two jobs to other positions on the same machine. An example is shown in Figure 6.
- *Switch:* A solution is constructed by randomly swapping the order of two different jobs allocated to the same machine. An example is shown in Figure 7.
- *Task Move:* A solution is generated by randomly moving a job from one machine to a random target machine. An example is provided in Figure 8.
- *Swap:* A solution is generated by randomly swapping two different jobs allocated to two different machines. An example is given in Figure 9

1) SWITCHING MECHANISM

The main purpose of the proposed switching mechanism is to determine when Stage-2 should be terminated. If Stage-2 has been terminated, the proposed approach will call Stage-1 to perform the exploration step. In this work, the switching mechanism of Stage-2 will be triggered if the best found solution cannot be further improved for 10 consecutive iterations.

IV. EXPERIMENT RESULTS

In this section, the experimental setup is presented, followed by the results of applying the proposed approach to a set of UPMSP benchmarks. The results of the proposed approach are compared against algorithms from the literature and tested for statistical significance.

A. EXPERIMENT SETUP

This work uses the UPMSP benchmark instances proposed by [4] in the experimental test. This benchmark involves 1640 instances ($40 \times n \times k$). These instances are categorised based on the number of jobs $n \in \{6, 8, 10, 12, 50, 100, 150,$

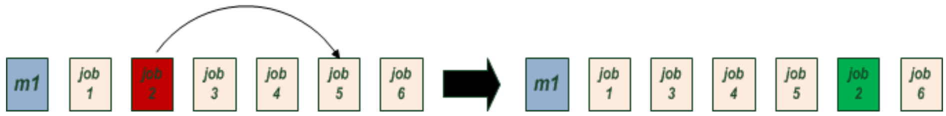

FIGURE 5. Example of a shift neighbourhood operator.

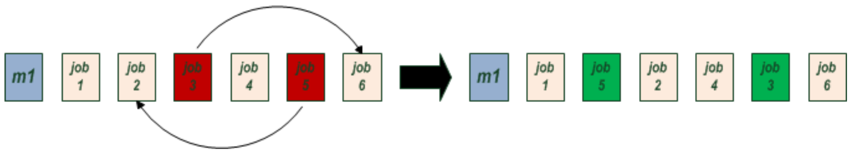

FIGURE 6. Example of a two-shift neighbourhood operator.

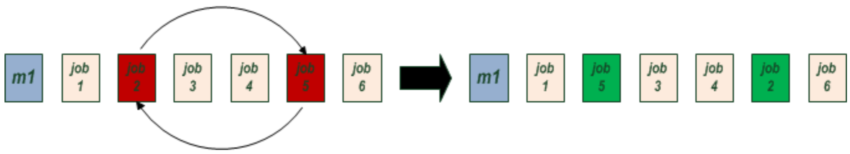

FIGURE 7. Example of a switch neighbourhood operator.

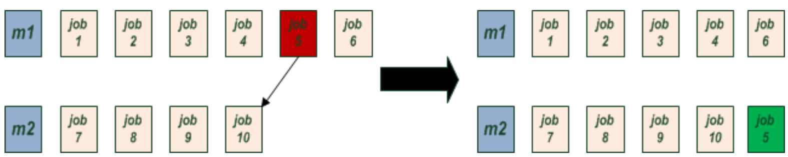

FIGURE 8. Example of a task move neighbourhood operator.

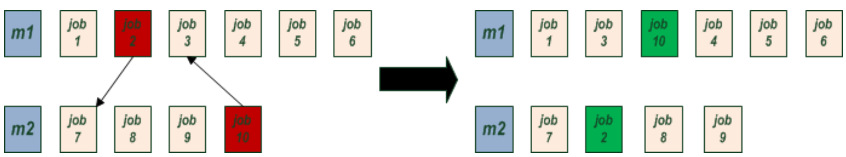

FIGURE 9. Example of a swap neighbourhood operator.

200, 250} and machines $k \in \{2, 3, 4, 5, 10, 15, 20, 25, 30\}$. In small-scale group $n \leq 12$ and $k \leq 5$ whereas in large-scale instances $n \geq 50$ and $k \geq 10$. The processing times is set in the range between 1 to 99. The setup times are set into four values uniformly distributed as follows: 1) $\{1, \ldots, 9\}$, 2) $\{1, \ldots, 49\}$, 3) $\{1, \ldots, 99\}$, and 4) $\{1, \ldots, 124\}$. The relative percentage deviation (RPD) is used to evaluate the solutions which is presented in Equation 9. For more detail, please refer to [4].

$$RPD = 100 \times \frac{M\_solution - BK\_solution}{BK\_solution} \tag{9}$$

where $M\_solution$ is the value of solution returned by the proposed approach and $BK\_solution$ is the best known solution.

B. PARAMETER SETTINGS

The proposed approach has five parameters. To set the parameters, a preliminary test is carried out to select the most appropriate value for each one. Several parameter values were combined for each one to choose the most effective values [24]. Specifically, we have randomly selected 10 different instances from each group for the parameter tuning purpose. We tuned the parameters one by one by using various values for each parameter, while keeping the other parameters fixed. Taking into consideration the computational time and solution quality, the average value of each parameter was selected. The selected values are shown in Table 2.

TABLE 2. The parameter values of the proposed approach.

#	Parameter	Selected value
1-	Probability of a crossover (*Pc*)	0.5
2-	Probability of a mutation (*Pm*)	0.2
3-	Maximum diversity between the current and initial solutions (*Max*)	70%
4-	Minimum diversity between the current and initial solutions (*Min*)	5%
5-	Number of consecutive non-improving iterations	10

For a fair comparison, we used the time limits proposed by [4] as our stopping condition as follows:

$$stopping_{cond} = n \times (\frac{m}{2}) \times t \tag{10}$$

TABLE 3. Consecutive non-improving parameter results.

Instances	5 iterations	10 iterations	15 iterations	20 iterations	25 iterations
1	3.2	1	1.1	1	0.0
2	0.1	0.1	0	0.3	0.2
3	4.2	0.0	0.1	0.0	0.0
4	1.1	0.0	0.4	1.2	0.6
5	1.8	0.0	0.3	3.4	1.4
6	2.2	1.3	0.6	4.1	1
7	2.3	0.1	0.5	0.4	0.3
8	2.5	1.4	0.3	1.2	3.4
9	0.8	1.2	0.8	2.2	1.6
10	4.11	0.0	1.4	4.7	2.4

where n represents the total number of jobs and m is the total number of machines in a given instance to be solved. t is set to 10, 30 and 50 based on the instance size. The $stopping_{cond}$ is used as a total running time in milliseconds.

For the consecutive non-improving iterations of Stag-2 parameter, we have tested different number of iterations: 5, 10,15,20 and 25. The results in term of the relative percentage deviation (RPD) are reported in Table 3. Based on the computational time and RPD values, 10 iterations perform the best overall instance.

C. NUMERICAL RESULTS

In this section, we first investigate the impact of the developed components on the performance of our approach and then compare the obtained results against the algorithms from the literature.

1) EFFECTIVENESS EVALUATION

The results of the proposed approach (denoted as EVNS) are compared with three different variants. The purpose is to examine the effectiveness of the developed components to the variable neighbourhood search method. The following variants and algorithms are used in the comparison.

- *VNS:* Variable neighbourhood search method only.
- *EVNSWDS:* Evolutionary variable neighbourhood search method without switching mechanism.
- *EVNS:* Evolutionary variable neighbourhood search method with switching mechanism.

The computational results and comparisons of the above variants are provided in Table 4. The variants are compared in terms of the average RPD obtained by each algorithm for different instance sizes. Values shown in bold font indicate the best achieved results. A close scrutiny of Table 4 demonstrates that the proposed EVNS outperforms other variants in all instances. This is because of the use of evolutionary operators and the switching mechanism improved the exploration and exploitation and helped EVNS in maintaining solution diversity during the search process.

2) COMPARISON WITH ALGORITHMS FROM THE LITERATURE

The results obtained by our proposed EVNS for all instances are presented and compared with the following algorithms that have been published in the literature:

- Genetic algorithm (GA) [4].
- Heuristic algorithm (AIRP) [19].

TABLE 4. The results of EVNS compared to VNS and EVNSWDS.

n	k	VNS	EVNSWDS	EVNS
6	2	4.06	1.98	**0.0**
	3	4.99	3.44	**0.0**
	4	6.25	3.97	**0.01**
	5	9.07	5.41	**0.0**
10	2	4.51	3.02	**0.0**
	3	2.86	2.05	**0.0**
	4	4.19	3.35	**0.13**
	5	4.57	2.01	**0.2**
50	10	5.68	4.02	**2.26**
	15	6.53	4.07	**1.57**
	20	4.09	3.11	**0.89**
	25	6.35	2.55	**1.22**
	30	4.02	2.06	**1.64**
100	10	5.19	3.87	**2.01**
	15	4.77	3.08	**2.1**
	20	7.54	4.42	**2.07**
	25	5.91	5.43	**1.81**
	30	5.13	4.84	**2.17**
200	10	4.39	3.74	**1.49**
	15	4.21	3.86	**1.73**
	20	4.29	4.02	**1.97**
	25	8.56	7.99	**2.4**
	30	6.99	5.03	**2.29**

- Simulated annealing (SA) [20].
- Iterated local search (ILS) [20].
- Late acceptance hill-climbing (LAHC) [20].
- Step counting hill-climbing (SCHC) [20].

The stopping condition of EVNS is fixed same as GA, AIRP, SA, ILS, LAHC and SCHC which is defined in Equation (6) as the maximum execution time limits. To ensure a fair comparison between EVNS and GA, AIRP, SA, ILS, LAHC and SCHC, the total time limits are scaled down using PassMark software. It should be noted that, up to our knowledge, only the above algorithms have been tested on the utilised UPMSP benchmarks. We suspect it might be due to the instance complexity, the number of instances (1,640 instances) and the instance size.

Our obtained results and the compared algorithms from the literature are provided in Table 5. The results are compared based on the average RPD obtained by each algorithm for different instance sizes, where the lower the better. In the table, the values in bold font indicate the best results obtained by one of these algorithms. In Table 5, the first column (denoted by n) represents the number of jobs, whereas the second column (denoted by k) represents the number of machines in each instance. Each value of n has been combined with different values of k indicated by one row. The different values of n show the instance difficulty, where the higher values are more difficult than the small ones. From the results in Table 5 we can make the following conclusions:

- For $n = 6$, our proposed algorithm (EVNS), is either better than others on all k values or obtained the same results as others. Specifically, EVNS is better than GA, AIRP, SA, ILS, LAHC and SCHC on 3, 4, 4, 4, 1 and

TABLE 5. Average RPDs of the compared algorithm for UPMSP instances.

n	k	GA	AIRP	SA	ILS	LAHC	SCHC	EVNS
6	2	**0.00**	0.87	0.86	2.04	**0.00**	**0.00**	**0.00**
	3	0.08	0.08	1.28	2.08	**0.00**	0.01	**0.00**
	4	0.27	3.80	0.75	2.25	0.04	0.05	**0.01**
	5	0.21	10.25	0.32	1.33	**0.00**	**0.00**	**0.00**
8	2	0.03	**0.00**	2.01	2.76	0.04	0.13	**0.00**
	3	0.24	**0.00**	1.33	4.28	0.12	0.19	**0.00**
	4	0.39	0.48	1.16	2.93	0.10	0.17	**0.08**
	5	0.20	0.38	0.12	1.64	**0.00**	0.55	**0.00**
10	2	0.17	**0.00**	1.35	3.23	0.05	0.33	**0.00**
	3	0.20	**0.00**	1.21	2.93	0.09	0.44	**0.00**
	4	0.32	**0.13**	1.18	3.83	0.30	0.96	**0.13**
	5	1.15	0.27	0.56	3.33	1.05	1.41	**0.20**
12	2	0.09	0.11	1.54	2.63	0.15	0.72	**0.08**
	3	0.08	**0.03**	1.52	3.94	0.22	0.95	0.08
	4	0.75	**0.12**	0.99	3.62	0.51	1.67	0.13
	5	1.67	0.63	1.28	5.10	1.66	2.53	**0.61**
50	10	12.42	5.97	2.38	5.37	6.43	6.87	**2.26**
	15	20.83	7.10	1.58	3.14	5.33	5.32	**1.57**
	20	25.65	9.30	0.94	1.96	3.90	4.26	**0.89**
	25	30.21	11.54	1.26	1.82	4.14	4.52	**1.22**
	30	32.83	12.80	**1.63**	1.74	3.24	3.61	1.64
100	10	13.96	8.79	2.06	3.81	3.92	5.20	**2.01**
	15	22.41	10.78	2.15	3.48	5.34	5.40	**2.10**
	20	29.63	13.42	2.15	3.51	5.33	5.58	**2.07**
	25	36.39	15.40	1.97	2.97	4.22	4.82	**1.81**
	30	41.64	19.22	**2.13**	3.52	4.20	3.87	2.17
150	10	15.81	10.03	**1.78**	3.77	3.08	4.39	**1.78**
	15	22.83	12.57	1.47	3.46	3.67	4.37	**1.41**
	20	30.62	14.78	2.33	3.68	4.09	4.42	**2.32**
	25	37.45	17.19	2.19	3.78	3.84	3.78	**2.09**
	30	43.27	20.06	**2.70**	4.49	3.80	4.15	**2.70**
200	10	16.14	10.47	1.55	3.39	2.92	3.98	**1.49**
	15	24.76	13.64	1.78	3.96	3.37	4.27	**1.73**
	20	31.89	15.89	2.03	4.18	3.49	3.92	**1.97**
	25	39.33	18.99	2.42	4.57	3.58	3.56	**2.40**
	30	44.81	21.08	2.32	5.11	3.24	3.36	**2.29**
250	10	17.14	10.52	1.57	3.44	2.67	3.64	**1.45**
	15	25.09	13.44	1.72	3.98	3.25	4.09	**1.64**
	20	33.38	16.18	2.04	4.45	3.45	3.93	**1.99**
	25	39.79	19.15	2.37	5.26	3.39	3.72	**2.33**
	30	44.95	21.29	2.10	5.74	2.81	3.32	**2.05**

2 instances, respectively. For other instances, EVNS obtained the same results.

- For $n = 8$, our results are better than other algorithms on all instances of $k = 2,3,4,5$. EVNS also achieved the same best results compared to AIRP and LAHC for $k = 2$, 3, and 5.
- For $n = 10$, EVNS is outperformed GA, SA, ILS, LAHC and SCHC on all k values. For AIRP, EVNS obtained the same results for three instances and better on one instances.
- For $n = 12$, our results are better on two instances and very competitive on two instances. If we examine individual comparison, EVNS is better than GA, SA, ILS, LAHC and SCHC.
- For $n = 50$, EVNS is achieved better results than GA, AIRP, ILS, LAHC and SCHC on all k values. Only SA is slightly better than EVNS when $k = 30$ where EVNS results is 1.64 while SA 1.63.
- For $n = 100$, our results are better than GA, AIRP, ILS, LAHC and SCHC on all k values. Only when $k = 30$, SA is slightly better than EVNS, where EVNS results is 2.17 while SA 2.13
- For $n = 150$, EVNS results are better than GA, ILS, LAHC and SCHC on all instances. EVNS is also better than SA on three instances and obtained the same results on two instances.

TABLE 6. Statistical test results of EVNS, GA, AIRP, SA, ILS, LAHC and SCHC.

n	k	GA	AIRP	SA	ILS	LAHC	SCHC
6	2	S=	S+	S+	S+	S=	S=
	3	S+	S+	S+	S+	S=	S=
	4	S+	S+	S+	S+	S+	S+
	5	S+	S+	S+	S+	S=	S=
8	2	S+	S=	S+	S+	S+	S+
	3	S+	S=	S+	S+	S+	S+
	4	S+	S+	S+	S+	S=	S+
	5	S+	S+	S+	S+	S=	S+
10	2	S+	S=	S+	S+	S+	S+
	3	S+	S=	S+	S+	S+	S+
	4	S+	S=	S+	S+	S+	S+
	5	S+	S+	S+	S+	S+	S+
12	2	S+	S+	S+	S+	S+	S+
	3	S+	S+	S+	S+	S+	S+
	4	S+	S+	S+	S+	S+	S+
	5	S+	S+	S+	S+	S+	S+
50	10	S+	S+	S+	S+	S+	S+
	15	S+	S+	S+	S+	S+	S+
	20	S+	S+	S+	S+	S+	S+
	25	S+	S+	S+	S+	S+	S+
	30	S+	S+	S-	S+	S+	S+
100	10	S+	S+	S+	S+	S+	S+
	15	S+	S+	S+	S+	S+	S+
	20	S+	S+	S+	S+	S+	S+
	25	S+	S+	S+	S+	S+	S+
	30	S+	S+	S-	S+	S+	S+
150	10	S+	S+	S=	S+	S+	S+
	15	S+	S+	S+	S+	S+	S+
	20	S+	S+	S+	S+	S+	S+
	25	S+	S+	S+	S+	S+	S+
	30	S+	S+	S=	S+	S+	S+
200	10	S+	S+	S+	S+	S+	S+
	15	S+	S+	S+	S+	S+	S+
	20	S+	S+	S+	S+	S+	S+
	25	S+	S+	S+	S+	S+	S+
	30	S+	S+	S+	S+	S+	S+
250	10	S+	S+	S+	S+	S+	S+
	15	S+	S+	S+	S+	S+	S+
	20	S+	S+	S+	S+	S+	S+
	25	S+	S+	S+	S+	S+	S+
	30	S+	S+	S+	S+	S+	S+

- For $n = 200$, EVNS outperformed GA, AIRP, SA, ILS, LAHC and SCHC on all instances.
- For $n = 250$, EVNS is better than all considered algorithms (GA, AIRP, SA, ILS, LAHC and SCHC) on all instances.

It should be noted that the executions time of all algorithms is the same as calculated by Equation (6). From the above, we can see that EVNS is very effective and efficient in solving UPMSP benchmark instances. One can also see that the performance of our algorithm is highly effective on large-scale instances, where it achieved better results when n increases. This clearly justifies the benefit of the proposed two stages in dealing with large and difficult search space by alternating between different search procedures. The good performance can also be attributed to the use of various evolutionary operators that help to navigate diverse areas in the search landscape. These results are also verified by the statistical test as shown in the next section.

D. STATISTICAL SIGNIFICANCE

The results presented in the previous subsection showed an advantage of EVNS over the remaining tested algorithms. In order to further verify these promising results, we also performed the Wilcoxon statistical test using 0.05 as a significance level. The results of the statistical test are shown in Table 6. A symbol $S+$ means EVNS is statistically better than the compared algorithms, symbol $S\ =$ means it is equal, while a symbol $S-$ means no significant difference between the algorithms. As shown in Table 6, EVNS is significantly outperforming the compared algorithms on most of the instances.

E. DISCUSSION

The numerical results presented in Table 5 demonstrated that the proposed EVNS obtained better results than other algorithms that were presented in the literature. The conducted statistical tests reported in Table 6 also supported these results. The good performance obtained by the proposed EVNS, we believe, can be attributed to the following features:

- The ability of the proposed EVNS in using a two-stage approach to strike a balance between exploration and exploration processes. By alternating, during the solving process, between the two different stages, the proposed EVNS can deal with various instances of different sizes and complexities.
- The ability of the proposed evolutionary framework to explore new regions in the search space. By utilising various crossover operators and mutation operators, the proposed EVNS can escape from the local optima points and jump into unexplored regions.
- The ability of the developed adaptive variable neighbourhood search algorithm in choosing, for each instance, different neighbourhood operators. By using, for each case, a diverse list of neighbourhood operators, the proposed EVNS can handle the changes that might happen during the solving process and effectively exploit the areas around the new solution.
- The ability of the proposed dynamic strategy to determine the switching time between the two stages. By using an active approach, the proposed EVNS can provide a new good starting solution for the second stage to exploit its neighbourhood areas instead of constructing a new one from scratch.
- The ability of the proposed fitness function to guide the solution update process that takes into account the diversity and fitness of each solution. By using diversity and fitness metric, the proposed EVNS can guide the search to avoid basin attraction points and generate a high-quality solution.

V. CONCLUSION

This article proposed a new hybrid approach for the unrelated parallel machine scheduling problem with sequence-dependent setup times. The proposed approach integrates a two-stage variable neighbourhood search method into the evolutionary framework. In the first stage, we proposed an evolutionary descent algorithm that utilises various crossover operators and mutation operators to explore the search space. In the second stage, we proposed an adaptive variable neighbourhood search to exploit the area around current solutions. A dynamic strategy to determine the switching time between these two stages and a diversity-based fitness function were proposed to guide the search towards promising areas. The competitiveness of our proposed algorithm was evaluated using 1640 benchmark instances which have been commonly used in the literature. The experiment results confirmed that the proposed approach can obtain results that are much better than those obtained by the state-of-the-art algorithms. In particular, the proposed approach outperformed the state-of-the-art algorithms on all tested instances. In future work, we intended the evaluate the effectiveness of the proposed approach on other challenging optimisation problems such as vehicle routing problem and timetabling problem.

REFERENCES

[1] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, Jun. 2008.

[2] O. Avalos-Rosales, A. M. Alvarez, and F. Angel-Bello, "A reformulation for the problem of scheduling unrelated parallel machines with sequence and machine dependent setup times," in *Proc. ICAPS*, Jun. 2013, pp. 278–282.

[3] O. Avalos-Rosales, F. Angel-Bello, and A. Alvarez, "Efficient Metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times," *Int. J. Adv. Manuf. Technol.*, vol. 76, nos. 9–12, pp. 1705–1718, Feb. 2015.

[4] E. Vallada and R. Ruiz, "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times," *Eur. J. Oper. Res.*, vol. 211, no. 3, pp. 612–622, Jun. 2011.

[5] M. Terzi, T. Arbaoui, F. Yalaoui, and K. Benatchba, "Solving the unrelated parallel machine scheduling problem with setups using late acceptance hill climbing," in *Intelligent Information and Database Systems* (Lecture Notes in Computer Science), vol. 12033, N. Nguyen, K. Jearanaitanakij, A. Selamat, B. Trawiński, and S. Chittayasothorn, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-41964-6_22.

[6] D. Li, J. Wang, R. Qiang, and R. Chiong, "A hybrid differential evolution algorithm for parallel machine scheduling of lace dyeing considering colour families, sequence-dependent setup and machine eligibility," *Int. J. Prod. Res.*, pp. 1–17, Mar. 2020.

[7] J. de Armas and M. Laguna, "Parallel machine, capacitated lot-sizing and scheduling for the pipe-insulation industry," *Int. J. Prod. Res.*, vol. 58, no. 3, pp. 800–817, Feb. 2020.

[8] F. Orts, G. Ortega, A. Puertas, I. García, and E. Garzón, "On solving the unrelated parallel machine scheduling problem: Active microrheology as a case study," *J. Supercomput.*, vol. 76, no. 11, pp. 8494–8509, 2020.

[9] H. Li, Z. Li, Y. Zhao, and X. Xu, "Scheduling customer orders on unrelated parallel machines to minimise total weighted completion time," *J. Oper. Res. Soc.*, pp. 1–11, Feb. 2020.

[10] L. Zhang, Q. Deng, G. Gong, and W. Han, "A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption," *Int. J. Prod. Res.*, vol. 58, no. 22, pp. 6826–6845, 2019.

[11] B. Vincent, C. Duhamel, L. Ren, and N. Tchernev, "A population-based metaheuristic for the capacitated lot-sizing problem with unrelated parallel machines," *Int. J. Prod. Res.*, vol. 58, no. 21, pp. 6689–6706, 2020.

[12] J. Batista Abikarram, K. McConky, and R. Proano, "Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing," *J. Cleaner Prod.*, vol. 208, pp. 232–242, Jan. 2019.

[13] M. Wang and G. Pan, "A novel imperialist competitive algorithm with multi-elite individuals guidance for multi-object unrelated parallel machine scheduling problem," *IEEE Access*, vol. 7, pp. 121223–121235, 2019.

[14] A. Arram, M. Ayob, G. Kendall, and A. Sulaiman, "Bird mating optimizer for combinatorial optimization problems," *IEEE Access*, vol. 8, pp. 96845–96858, 2020.
[15] M. S. A. Daweri, S. Abdullah, and K. A. Z. Ariffin, "A migration-based cuttlefish algorithm with short-term memory for optimization problems," *IEEE Access*, vol. 8, pp. 70270–70292, 2020.
[16] A. Arram and M. Ayob, "A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems," *Comput. Ind. Eng.*, vol. 133, pp. 267–274, Jul. 2019.
[17] S. Abdullah and N. S. Jaddi, "Dual kidney-inspired algorithm for water quality prediction and cancer detection," *IEEE Access*, vol. 8, pp. 109807–109820, 2020.
[18] A. A. Hassan, S. Abdullah, K. Z. Zamli, and R. Razali, "Combinatorial test suites generation strategy utilizing the whale optimization algorithm," *IEEE Access*, vol. 8, pp. 192288–192303, 2020.
[19] L. P. Cota, M. N. Haddad, M. J. F. Souza, and V. N. Coelho, "Airp: A heuristic algorithm for solving the unrelated parallel machine scheduling problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1855–1862.
[20] H. G. Santos, T. A. M. Toffolo, C. L. T. F. Silva, and G. V. Berghe, "Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem," *Int. Trans. Oper. Res.*, vol. 26, no. 2, pp. 707–724, Mar. 2019.
[21] X. Gu, Y. Li, and J. Jia, "Feature selection for transient stability assessment based on kernelized fuzzy rough sets and memetic algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 64, pp. 664–670, Jan. 2015.
[22] 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," *Int. J. Prod. Res.*, vol. 52, no. 19, pp. 5841–5856, Oct. 2014.
[23] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997.
[24] G. Kendall, R. Bai, J. Blazewicz, P. De Causmaecker, M. Gendreau, R. John, J. Li, B. McCollum, E. Pesch, R. Qu, and N. Sabar, "Good laboratory practice for optimization research," *J. Oper. Res. Soc.*, vol. 67, no. 4, pp. 676–689, 2016.

SALWANI ABDULLAH received the B.Sc. degree in computer science from Universiti Teknologi Malaysia, the master's degree specializing in computer science from Universiti Kebangsaan Malaysia (UKM), and the Ph.D. degree in computer science from the University of Nottingham, U.K. She is currently a Professor of computational optimization with the Faculty of Information Science and Technology, UKM. Her research interests include artificial intelligence and operation research, particularly computational optimization algorithms (heuristic and meta-heuristic, evolutionary algorithms, and local search) that involve different real world applications for single and multi-objective continuous and combinatorial optimization problems, such as timetabling, scheduling, routing, nurse rostering, dynamic optimization, data mining problems (feature selection, clustering, classification, and time-series prediction), intrusion detection, and search-based software testing.

AYAD TURKY received the Ph.D. degree in computer science and IT from the School of Science, RMIT University, Melbourne, VIC, Australia. He has published more than 31 papers in international journals and peer-reviewed conferences. His current research interests include the design and development of hyper-heuristic frameworks, deep learning, machine learning, evolutionary computation, and hybrid algorithms, with a specific interest in big data optimization problems, cloud computing, dynamic optimization, and data-mining problems.

MOHD ZAKREE AHMAD NAZRI received the bachelor's degree in system science and management from Universiti Kebangsaan Malaysia (UKM) and the master's and Ph.D. degrees in computer science from Universiti Teknologi Malaysia (UTM). He has been an Associate Professor with the Faculty of Information Science and Technology, UKM, since 1999. He is currently the Principal Researcher of the Centre for Artificial Intelligence Technology (CAIT), UKM. He has published more than 70 papers in international journals and peer-reviewed international conferences. His main research interests include decision support systems, data analytics, machine learning, and optimization. He has served on a programme committee for various international conferences and a reviewer for journals.

NASSER R. SABAR is currently a Lecturer with the Department of Computer Science and Information Technology, La Trobe University, Australia. He has published more than 75 papers in international journals and peer-reviewed conferences. His current research interests include hyper-heuristic frameworks, evolutionary computation, and hybrid algorithms with a specific interest in big data optimization problems, cloud computing, dynamic optimization, and scheduling problems.