Engineering Journal issn: 0125-8281

Article

A Multi-Objective Variable Neighborhood Search Algorithm for Precast Production Scheduling

Lehuang Zong[a] **and Wanatchapong Kongkaew**[b,*]

Department of Industrial Engineering, Faculty of Engineering, Prince of Songkla University, Songkhla 90110, Thailand
E-mail: [a]2795243384@qq.com, [b]wanatchapong.k@psu.ac.th (Corresponding author)

Abstract. In real life, precast production schedulers face the challenges of creating a reasonable schedule to satisfy multiple conflicting objectives. Practical constraints and objectives encountered in the precast production scheduling problem (PPSP) were addressed, with the goal to minimize makespan and total earliness and tardiness penalties. A multi-objective variable neighborhood search (MOVNS) algorithm was proposed and the performance was tested on 11 problem instances. Ten of these were generated using precast concrete production information taken from the literature. One real industrial problem from a precast concrete company was considered as a case study. Extensive experiments were conducted, and the spread and distance metrics were used to evaluate the quality of the non-dominated solutions set. Statistical analysis demonstrated that the result was statistically convincing. Computational results showed that the proposed MOVNS algorithm was significantly better when compared to the other nine algorithms. Therefore, the proposed MOVNS algorithm was a very competitive method for the considered PPSP.

Keywords: Precast production scheduling, multi-objective, metaheuristic, variable neighborhood search, spread and distance.

ENGINEERING JOURNAL Volume 24 Issue 6
Received 19 November 2019
Accepted 12 October 2020
Published 30 November 2020
Online at https://engj.org/
DOI:10.4186/ej.2020.24.6.139

DOI:10.4186/ej.2020.24.6.139

1. Introduction

Construction industries face problems of delay, cost overruns, low quality, poor safety records and environmental impacts because of risk, uncertainties, labor complexity and recent dynamic change [1-2]. To overcome these problems, the industrialized precast construction technique has developed rapidly since its inception in the 1950's. Precast construction is different from traditional construction methods and requires components such as beams, columns and girders to be prefabricated in factories before transportation and installation on-site according to the erection schedule. Production scheduling has a dramatic impact on the success of precast fabrication because it involves making accurate decisions on when the many precast components (PCs) need to be produced to meet their due dates of delivery. Unfortunately, in practice, precast production schedules are arranged by experience-based estimation as a subjective approach that frequently results in inefficient precast manufacturing processes [3-6]. Therefore, to eliminate the unexpected consequences of manually arranged production schedules, modeling and computational techniques are now used to consider more realistic constraints prevailing in the precast industry, such as off-normal working time and non-preemptible fabrication processes [7-8], limited workers and cranes [9], buffer size between production stations [3-4, 10], mold availability [11], different concrete formulas [12] and multiple production lines [13].

Most previous investigations used the precast production scheduling problem (PPSP) model based on the traditional flow shop scheduling problem (FSSP) model that was defined as a non-deterministic polynomial-time hard (NP-hard) problem [6-7, 11] and solved by genetic algorithm (GA) based optimization methods [4, 6-11, 13-14]. Other heuristics such as Palmer's heuristic [15], Gupta's heuristic [16], the Campbell Dudek Smith (CDS) heuristic [17], rapid access (RA) heuristic [18], earliest due date (EDD) rule, as soon as possible rule, shortest processing time (SPT) rule, and least slack time rule have also been developed to provide performance verification of the algorithms to solve the PPSP [4, 7, 8, 13, 19]. Results indicated that variants of GA metaheuristics provided better solutions than the heuristics and dispatching rules for solving the PPSP. However, previous studies ignored other competitive metaheuristic algorithms such as multi-objective variable neighborhood search (MOVNS) and non-dominated sorting genetic algorithm II (NSGA-II), while GA based metaheuristics were not compared with any of these algorithms.

The variable neighborhood search (VNS) algorithm [20-21] is a local search-based metaheuristic that was first proposed in 1997. By employing systematic changes of neighborhood, VNS is able to explore increasingly distant neighborhoods of current incumbent solutions to obtain promising results. Due to its simplicity and powerful search ability, VNS approaches have been successfully applied to solve various scheduling problems in both single and multiple objectives, including the flow shop scheduling problem [22-26], the job shop scheduling problem [27-30], and the single machine scheduling problem [31]. Similarly, the NSGA-II, first introduced by Deb et al. [32] is one of the most proficient evolutionary algorithms used for solving multi-objective optimization problems. The NSGA-II has been successfully applied to solve flow shop and job shop scheduling problems [23-24, 33-34]. Although the VNS and NSGA-II have been successfully applied to solve many scheduling problems, an extensive literature review revealed little evidence, if any, as to whether VNS and NSGA-II have been applied to solve the PPSP. Therefore, the objective of this study was to extend the application of VNS and NSGA-II algorithms to solve the PPSP.

Ko and Wang [4] applied the multi-objective genetic local search (MOGLS) algorithm of [35] to solve the PPSP. The MOGLS algorithm successfully searched for optimum production schedules and outperformed seven methods, including the Palmer, Gupta, CDS and RA heuristics, the EDD rule, the vector evaluated genetic algorithm (VEGA) and the constant weight genetic algorithm (CWGA). Therefore, here, the MOGLS was considered as a comparative algorithm to solve the PPSP due to its competitive performance.

The main contributions of this study are summarized as follows. Firstly, MOVNS and NSGA-II were implemented to solve the multi-objective PPSP based on the mathematical model proposed in [7]. The application of MOVNS and NSGA-II also fills the research gap since these algorithms have never been applied before to solve the PPSP. Secondly, this study extended the PPSP size up to 100 jobs, while previous studies only considered the PPSP for 6 jobs [7], 10 jobs [3-4, 6, 10-11, 13-14], 30 jobs [4], 36 jobs [14] and 44 jobs [7]. A real-world PPSP was also provided as a case study. Lastly, distance metrics were applied to measure all tested algorithms to solve the PPSP. Distance metrics consider both the diversity and convergence of population solutions, while the spread and spacing metrics used in [4] only considered diversity. Moreover, to demonstrate the competitive performance over algorithms proposed in previous studies to solve the FSSP, some recent metaheuristic algorithms such as cuckoo search algorithm, bat algorithm, and firefly algorithm were used to compare the search capability.

The remainder of this paper is organized as follows: Section 2 discusses the mathematical model of the PPSP. Section 3 describes the proposed MOVNS and NSGA-II algorithms for solving the PPSP. Section 4 demonstrates the experiments and computational results of the proposed approaches, while conclusions and recommendations are presented in Section 5.

2. Precast Production Scheduling Problem

Warszawski and Ishai [36] divided precast production systems into two basic types, namely the stationary production system and the traveling production system. In the stationary production system, all basic production operations are performed at fixed locations and a comprehensive workforce is involved. In the traveling

system, molds are moved among different workstations and diverse operations are processed by different workforces using specialized tools and methods. Precast production processes using specialized methods can be broken down into six tasks, i.e., (1) mold assembly, (2) reinforcement setting, (3) concrete pouring, (4) concrete curing, (5) demolding, and (6) product finishing. With the traditional FSSP, each job consists of operations and each operation is executed on a specific machine. This cannot be applied to the precast production scheduling problem (PPSP) directly because some practical constraints encountered in the industry are disregarded in FSSP. A PPSP model based on FSSP was proposed by Chan and Hu [7] with the following assumptions.

Firstly, there is no distinction between normal working time and off-normal working time in the traditional FSSP; however, interruptions inevitably happen in precast plants when workers punch out after working time, which is normally 8 hours a day. Furthermore, the labor force needs to be paid to work overtime within the limited hours (assuming no more than 4 hours) if necessary. Secondly, all operations in the traditional FSSP are uninterruptible. This means that an operation once started cannot be interrupted until completion. Operations in precast plants can be divided into interruptible operations (mold assembly, reinforcement setting, demolding and product finishing) and uninterruptible operations (concrete pouring and concrete curing). Interruptible operations can be interrupted and continued to execute the unfinished part on the next day if they cannot be completed within normal working time; this causes inevitable interruption time (off-normal working time). A description of notations used in the mathematical formulation is listed in Table 1.

Table 1. Mathematical notations and definitions.

Notation	Definition
n	total number of precast components (PCs)
m	total number of workstations
i	serial number of the PC (called "job" in traditional FSSP model).
j	serial number of the workstation (called "machine" in traditional FSSP model).
t_{ij}	processing time of the ith PC on the jth workstation
c_{ij}	completion time of the ith PC on the jth workstation
d_i	due date of the ith PC
D	working days
ε_i	unit earliness penalty for the ith PC
τ_i	unit tardiness penalty for the ith PC
T	accumulated completion time
T_W	normal working time of a workday (8 hours)
T_N	off-normal working time
T_A	allowable overtime (assumed to be limited to 4 hours in one workday)

In practice, production schedules are constructed to minimize production duration and cost. Minimization of makespan, which is defined as the total time needed to complete all jobs, is a commonly-used objective to estimate the performance of PPSP models [3-4, 6-10, 13-14]. Earliness and tardiness are related to job due dates. Certain penalty costs are incurred when a job is completed either before or after its due date. Minimizing the total penalty costs of earliness and tardiness is, therefore, important to meet the just-in-time production control policy [37]. In this research, minimizing makespan and minimizing the total penalty costs of earliness and tardiness were computed by Eqs. (1) and (2) and applied as multi-objective functions in the PPSP model.

To handle the constraint on interruptible operations, Eq. (3) was utilized to calculate the completion time of interruptible processes. An uninterruptible operation such as concrete pouring would have to be postponed to the next working day if it could not be finished within the working hours or allowable overtime. The curing process is also an uninterruptible operation. Curing of the concrete occurs after pouring and no labor is required. A fast cure generally takes a few hours, while steam curing can be completed within 12-16 hours. The completion time of concrete pouring and curing was computed by Eqs. (4)-(6), respectively. The accumulated completion time and the working days were calculated from Eqs. (7) and (8), respectively. More details of the PPSP model are available in [4, 7-8, 11, 14].

Objectives:

$$f_1(x) = c_{nm} \tag{1}$$

$$f_2(x) = \left[\sum_{i=1}^{n} \varepsilon_i \times \max(0, d_i - c_i)\right] + \left[\sum_{i=1}^{n} \tau_i \times \max(0, c_i - d_i)\right] \tag{2}$$

Constraints:

$$c_{ij} = \begin{cases} T & \text{if} \quad T < 24D + T_W \\ T + T_N & \text{if} \quad T \geq 24D + T_W \end{cases}, \; i = 1,2,...,n; \; j = 1,2,5,6 \tag{3}$$

$$c_{ij} = \begin{cases} T & \text{if} \quad T < 24D + T_W + T_A \\ 24(D+1) + t_{ij} & \text{if} \quad T \geq 24D + T_W + T_A \end{cases}, \; i = 1,2,...,n; \; j = 3 \tag{4}$$

$$T^* = c_{i3} + t_{i4}, \quad i = 1,2,...,n \tag{5}$$

$$c_{ij} = \begin{cases} T^* & \text{if} \quad T^* > 24(D+1) \quad \text{or} \quad T^* < 24D + T_W \\ 24(D+1) & \text{if} \quad 24D + T_W \leq T^* \leq 24(D+1) \end{cases}, \; i = 1,2,...,n; \; j = 4 \tag{6}$$

$$T = \max\left(c_{(i-1)j}, c_{i(j-1)}\right) + t_{ij}, \; i = 2,3,...,n; \; j = 2,3,4,5,6 \tag{7}$$

$$D = \text{integer}(T/24) \tag{8}$$

3. Proposed Multi-Objective Algorithms for the PPSP

In this paper, two metaheuristic algorithms as the multi-objective variable neighborhood search (MOVNS) and a non-dominated sorting genetic algorithm II (NSGA-II) were first implemented to find the set of Pareto optimal solutions for the PPSP. Figure 1 illustrates a flowchart of these algorithms with details described in the following subsections.

3.1. Proposed MOVNS for the PPSP

Variable neighborhood search (VNS) is a local search-based metaheuristic that was originally proposed by Mladenović and Hansen [20]. It is based on the principle of systematic changes of neighborhood in both the descent phase to find a local optimum and the perturbation phase to escape from the corresponding local minimum valley. Thus, VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent solutions to obtain promising neighboring solutions. The MOVNS algorithm used here was developed based on Geiger's algorithm [38] and the flowchart of MOVNS is depicted in Fig. 1(a).

The main steps of the developed MOVNS algorithm outline are as follows.

Step 1 (Encoding): This study encodes production scheduling by precast components (PCs) sequencing. Then, a production schedule of PC numbers can be represented by a chromosome (solution) in the population of MOVNS. The encoding schema of the proposed MOVNS algorithm is shown in Fig. 2.

Step 2 (Initialization): Choose a stopping criterion, define the set of neighborhood structure (N_k, $k = 1 : k_{\max}$), and randomly generate the initial population of N_{pop} chromosomes to represent as precast production schedules. Each chromosome represents a solution, which is the schedule of PCs sequence in PPSP.

Step 3 (Evaluation): Evaluate objective values, i.e., makespan and the total earliness/tardiness penalty of each chromosome in the current population *pop* using Eqs. (1) and (2), respectively.

Step 4 (Update the Pareto front): The tentative set D where all non-dominated solutions are stored separately from the current population is updated according to the concept of domination. A solution p is said to dominate solution q if and only if $f_i(p) \leq f_i(q), \forall i \in \{1,2,...,q\}$ and $f_i(p) < f_i(q), \exists i \in \{1,2,...,q\}$.

Step 5 (Selection): Randomly select an *unvisited* base solution from D, and randomly select a neighborhood structure N_k from the following two common neighborhood structures.

(a) *Insertion neighborhood* (N_1): Randomly select two positions r_1 and r_2 (where $r_1 < r_2$) in the solution representation. Then remove the PC at position r_2 and insert it before r_1 in the scheduling string, as shown in Fig. 3(a).

(b) *Swap neighborhood* (N_2): Randomly select two positions r_1 and r_2 in the solution representation and then swap the two PCs at the r_1 and r_2 in the scheduling string, as shown in Fig. 3(b).

Step 6 (Mark): The selected base solution is marked as *visited* to avoid selection in the next iterations. If all solutions in the tentative set D have been marked as *visited*, then all the marks will be removed.

Step 7 (Shaking): Randomly generate a solution $\mathbf{x}'$ from the N_k neighborhood of current solution $\mathbf{x}$.

Step 8 (Local search): Apply a complete local search in the N_k neighborhood of $\mathbf{x}'$, denote the obtained local optimum with $\mathbf{x}''$.

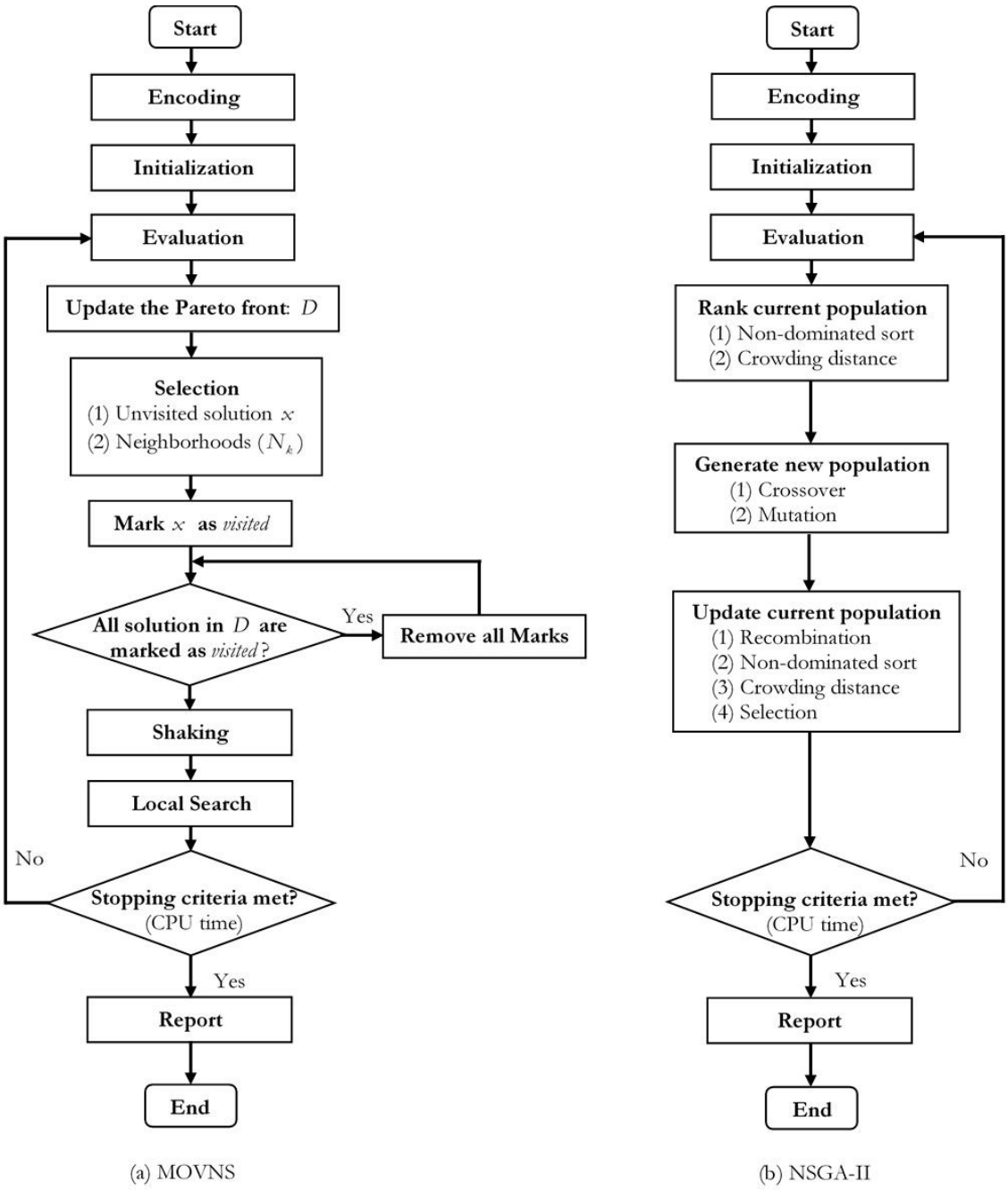


Fig. 1. Flowchart of the two proposed metaheuristics for the PPSP.

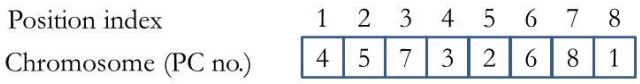


Fig. 2. Chromosome representation.

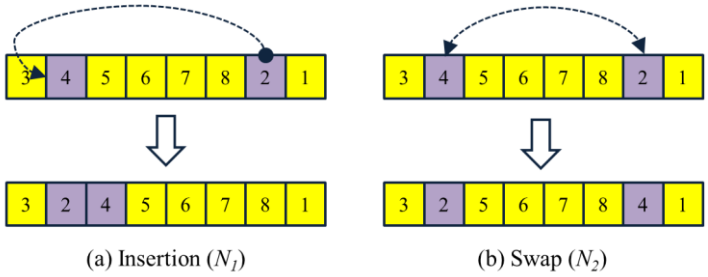


Fig. 3. Neighborhood structures.

Step 9 (Termination): If the algorithm reaches maximum CPU time, end the algorithm. Otherwise, iteratively execute Steps 3-8 (update the Pareto front using generated solution $\mathbf{x}''$).

Step 10 (Report): The sequences of PCs are represented by Pareto optimal solutions, i.e., solutions in the tentative set D of final iteration are reported as optimum schedules for precast concrete production.

3.2. Proposed NSGA-II for the PPSP

As noted earlier, the application and popularity of NSGA-II were the reasons for the choice of the algorithm in this study. In the algorithm, parent population is ranked to create Pareto fronts using the fast non-domination sorting and crowding distance procedures. Then, the algorithm applies binary tournament selection, crossover and mutation operators to generate an offspring population as the next generation. Finally, the best individuals in terms of non-dominance and diversity are selected as the solutions. The main components of the algorithm are summarized in Fig. 1(b).

The detailed steps of the NSGA-II algorithm are outlined as follows.

Step 1 (Encoding): Encode the solution of NSGA-II by PCs sequencing with the method shown in Fig. 2.

Step 2 (Initialization): Randomly generate initial population of N_{pop} chromosomes to represent as precast production schedules.

Step 3 (Evaluation): Evaluate objective values of each chromosome in the current population pop using Eqs. (1) and (2), respectively.

Step 4 (Rank current population): The current generation population is ranked by the following steps.

(4.1) *Non-dominated sort*: each chromosome of pop is assigned a rank using the fast non-domination sorting procedure described below.

(4.1.1) Initialize front counter: $r = 0$.

(4.1.2) Increase: $r = r + 1$.

(4.1.3) Find non-dominated solutions from pop according to the concept of domination.

(4.1.4) Assign rank r to these non-dominated solutions.

(4.1.5) Remove these non-dominated solutions from pop.

(4.1.6) Repeat Steps (4.1.2)-(4.1.5) until pop is empty.

(4.2) *Crowding distance*: the crowding distance value for each chromosome is calculated as follows.

(4.2.1) Initialize the distance of all Z individuals to be zeros: $d_i = 0$ for $i = 1, 2, ..., Z$.

(4.2.2) For the objective function f_x (f_x is makespan or penalty cost), sort the set in ascending order.

(4.2.3) Let d_1 and d_Z be infinite distance:

(4.2.4) For $j = 2, 3, ..., Z-1$, let

$$d_j = d_j + \left(\frac{f_k^{(j+1)} - f_k^{(j-1)}}{f_k^{\max} - f_k^{\min}} \right).$$

(4.3) *Crowded-comparison-operator*: Once the chromosomes are assigned rank by the fast non-domination and assigned crowding distance, the crowded-comparison-operator ($\prec_n$) is employed in the selection process at various stages of the algorithm. Assume that every chromosome i has non-domination rank (i_{rank}) and crowding distance (d_i), the partial order $\prec_n$ is defined by $i \prec_n j$ if ($i_{rank} < j_{rank}$) or (($i_{rank} = j_{rank}$) and $d_i > d_j$); then the chromosome i is selected using a binary tournament selection operator.

Step 5 (Generate new population):

(5.1) *Crossover*: Generate an offspring population $pop_{crossover}$ by the following steps.

(5.1.1) Select a pair of parent solutions from the population using binary tournament selection operator based on non-domination rank and crowding distance.

(5.1.2) Implement a two-point cut crossover operator, as shown in Fig. 4(a), to the selected pair of parents to generate two new child solutions.

(5.1.3) Evaluate objective values for the two new child solutions according to Eqs. (1) and (2).

(5.1.4) Repeat Steps (5.1.1)-(5.1.3) until $N_{crossover}$ child solutions are generated through the crossover operator among $N_{crossover}$ selected parent solutions.

(5.2) *Mutation*: Generate an offspring population $pop_{mutation}$ by the following steps.

(5.2.1) Select a solution from the population using binary tournament selection with crowded-comparison operator.

(5.2.2) Implement mutation operator, as shown in Fig. 4(b), to the selected solution to create a new solution.

(5.2.3) Evaluate objective values for the new child solution according to Eqs. (1) and (2).

(5.2.4) Repeat Steps (5.2.1)-(5.2.3) until $N_{mutation}$ parent solutions are selected to mutate $N_{mutation}$ child solutions.

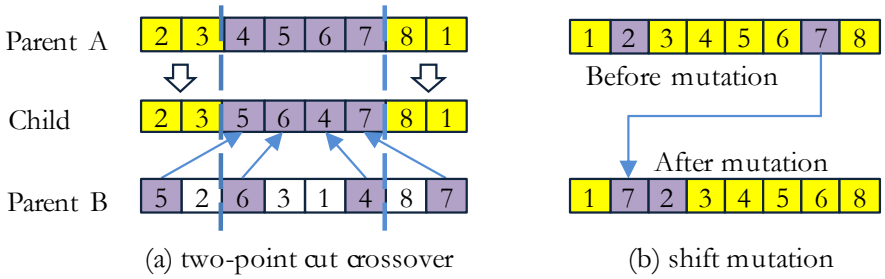


Fig. 4. Genetic operators' schemas.

Step 6 (Update current population): The current population pop is updated for a further run of the algorithm by the following steps.

(6.1) *Recombination*: The current population pop is merged with its offspring populations $pop_{crossover}$ and $pop_{mutation}$: $pop = [pop, pop_{crossover}, pop_{mutation}]$.

(6.2) *Non-dominated sort*: Each individual in the recombination population pop is assigned a rank based on non-domination criteria.

(6.3) *Crowding distance*: Calculate the crowding distance value for each individual in the recombination population pop.

(6.4) *Selection*: Once the recombination population pop is sorted, based on descending crowding distance and ascending non-domination rank, individuals of the new generation need to be selected from the current population. The new generation is filled by each subsequent front until the population size exceeds the current population size and deletes the extra individuals. If the population exceeds N by adding all the individuals with rank r, then individuals with rank r are selected based on their crowding distance in descending order until the population size is N_{pop}: $pop = pop(1 : N_{pop})$.

Step 7 (Termination test): If the algorithm reaches maximum generations, terminate the algorithm and return the Pareto optimal solutions in the current population; Otherwise, return to Step 3 to start the loop for the next iteration.

Step 8 (Report): Similar to the last step of MOVNS in solving PPSP, pop solutions of final iteration are reported as optimum schedules for precast production.

4. Computational Experiments

This paper analyzed the efficiency of the two proposed metaheuristics: multi-objective variable neighborhood search (MOVNS) algorithm and non-dominated sorting genetic algorithm II (NSGA-II) for solving the PPSP. The performance of these two algorithms has been tested by several problem instances with results compared against the multi-objective precast production scheduling model (MOPPSM) algorithm [4], and some recent algorithms successfully solved the FSSP. A statistical analysis was also carried out to demonstrate and validate the results.

4.1. Problem Instance Sets

To verify the capability of the proposed algorithms, two types of problem instance sets were used. In the first problem instance set, a case study of a real industrial scheduling problem in a precast production company was considered. In the second set, precast production data were collected from the literature and used to develop ten generated problem instances with different numbers of production jobs.

4.1.1. Generated Problem Instances

To test the performance of the proposed algorithms, experiments were conducted on generated problem instances. To develop the generated problem instances, data were taken from three data sets in the literature. The data are illustrated in Table 2 as 26 types of precast components (PCs). The first 6 types of PC (out of 26 types) were taken from [8], the next 10 types of PC (PC type nos. 7-16) were taken from [11] and the remaining 10 types of PC (PC type nos. 17-26) were taken from [14]. Based on the collected data set, ten problem instances with different numbers of jobs (up to 100 jobs) were developed, as shown in Table 3. In each problem instance, the integer number was arbitrarily generated and represented as the number of jobs to be produced for each type of PC in the precast production process. For example, PPSP03 with size of 30 jobs (i.e., 30 customer orders) was generated arbitrarily to each type of PC from PC types nos. 1-26 with numbers of 0, 0, 0, 0, 0, 0, 4, 2, 2, 5, 3, 3, 2, 2, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0 and 0, respectively. This sequence of numbers indicates that all the operations of precast concrete production were performed only for the PC types nos. 7-16 with the number of jobs (or the number of times to be produced): 4, 2, 2, 5, 3, 3, 2, 2, 5, 2. Each job in the PC type was assigned a serial number of the PC as to be PC no. Thus, the PC type no. 7 was produced four times in the production plan and each was assigned a serial number (i.e., PC nos. 1-4). For the PC types nos. 8 and 9, each was produced twice in the production plan and each was assigned a serial number: PC nos. 5-6 for PC type no. 8 and PC nos. 7-8 for PC type no. 9. The remaining 7 PC types were assigned a serial number (PC nos. 9-30) following the manner as described above.

4.1.2. Case Study

In this paper, data were collected from a leading precast concrete production company in Trang Province, Thailand. The company produces a variety of precast concrete components. From among them, two precast concrete products as concrete plank slabs and concrete piles were considered. The case study contained a variety of customer orders on the two precast products. There were a total of 24 customer orders, and each customer order was considered as a different job (i.e, different PC no.). The jobs were processed in six operations: mold assembly, reinforcement setting, concrete pouring, concrete curing, demolding, and product finishing. Processing times for the different jobs at different operations are given in Table 4. Production planning scheduling at the company was performed manually by the production supervisor.

4.2. Performance Measures

Various performance metrics have been used for numerical comparisons of the non-domination fronts on the different multi-objective algorithms. In this paper, the following two metrics were chosen.

4.2.1. Distance Metric

The distance metric is utilized to measure the performance of a non-domination solution set S_k, $k = 1,2,...,A$, relative to the reference set S^* (the best known Pareto front) and A is the total number of tested algorithms. In this paper, the distance metrics were calculated as follows.

Minimum distance ($D_{\min}$):

$$D_{\min} = \min_{y \in S^*} \left\{ \min \left\{ d_{xy}, x \in S_k \right\} \right\} \tag{9}$$

Average distance ($D1_R$):

$$D1_R = \frac{1}{\left| S^* \right|} \sum_{y \in S^*} \min \left\{ d_{xy}, x \in S_k \right\} \tag{10}$$

Maximum distance ($D_{\max}$):

$$D_{\max} = \max_{y \in S^*} \left\{ \min \left\{ d_{xy}, x \in S_k \right\} \right\} \tag{11}$$

where d_{xy} represents the Euclidean distance between a solution x and a reference solution y, computed by

$$d_{xy} = \sqrt{\left(f_1^*(y) - f_1^*(x) \right)^2 + \left(f_2^*(y) - f_2^*(x) \right)^2} \tag{12}$$

where $f_i^*(\square)$ is the ith objective that is normalized according to the reference set S^*. The normalization is accomplished by the following equation.

$$f_i^*(x) = 100 \times \left(\frac{f_i(x) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \right) \tag{13}$$

where $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of the ith objective in the reference set S^*, respectively. The reference set S^* consists of the non-dominated solutions of all the algorithms considered, i.e., $\bigcup_{k=1}^{A} S_k$. Details of the distance metric on $D1_R$ and $D_{\max}$ can be found in [39-41], while $D_{\min}$ measures the performance in an opposite manner to $D_{\max}$.

More specifically, by using the above distance metrics, both the convergence and diversity of a Pareto front set can be considered. An algorithm having smaller values is better. The distance metric is widely applied to measure multi-objective scheduling problems [23-24, 31, 41-44]. Unlike the generation distance (GD) metric in [41, 45-46], the measure of $D1_R$ in Eq. (10) is not the average distance from each solution in the solution set S_k to its nearest reference solution set S^*. However, using the reference set S^*, the metric $D1_R$ can evaluate the distribution of S_k as well as the proximity of S_k in the metric GD.

4.2.2. Spread Metric

The spread (SP) metric is employed to estimate the extent of the final non-dominated solution set S_k, $k = 1,2,...,A$, obtained by each algorithm. The spread metric $SP(S_k)$ can be calculated as follows [4, 45-46]:

$$SP(S_k) = \sqrt{\sum_{i=1}^{N} \left(\max_{j=1}^{|S_k|} f_i^*(x_j) - \min_{j=1}^{|S_k|} f_i^*(x_j) \right)^2} \tag{14}$$

where $x_j \in S_k$, for $j = 1,2,...,|S_k|$. The metric $SP(S_k)$ in Eq. (14) is calculated as the length of the diagonal line of the minimum N-dimension hyper-rectangle in the solution space. Details of the spread metric can be found in [47-48].

4.3. Parameter Settings and Experimental Setup

This section demonstrated an experimental study on the performance of the proposed approaches. Experiments were conducted on the set of problem instances described in Section 4.1. To compare the performance of MOVNS and NSGA-II, eight recent metaheuristic algorithms were assessed as follows.

1. Multi-objective precast production scheduling model (MOPPSM) algorithm [4].
2. The standard bat algorithm for multiple objectives (MOBA) [49].
3. The bat algorithm in [50], renamed BA-HFS.
4. The standard firefly algorithm for multiple objectives (MOFA) [51].

Table 2. Production data of precast components (PCs) taken from the literature.

PC type no.	Processing times (h)						Due date (h)	Penalties	
	Mold assembly	Reinforcement setting	Concrete casting	Concrete curing	Mold stripping	Product finishing		Earliness (ε_i)	Tardiness (τ_i)
1	1	0.8	1.2	12	1.5	0.5	28	2	10
2	1.7	2	2	12	1.5	2.5	28	2	10
3	0.4	0.5	0.6	12	0.5	0	28	1	10
4	0.3	0.4	0.5	12	0.4	1	28	1	10
5	1.5	1.8	1.2	12	1.5	1.5	52	2	10
6	1.5	1.6	1.5	12	1.8	0.8	32	2	10
7	2	1.6	2.4	12	2.5	1	112	2	10
8	3.4	4	4	12	2.4	5	112	2	10
9	0.8	1	1.2	12	0.8	0	112	1	10
10	0.6	0.8	1	12	0.6	2	112	1	10
11	3	3.6	2.4	12	2.4	3	208	2	10
12	3	3.2	3	12	3	1.6	128	2	10
13	1.3	0.9	2.4	12	1.9	1.8	144	2	10
14	1.7	1.4	1.1	12	0.9	0.7	144	2	20
15	2.2	1.8	1.2	12	2.3	0.7	144	1	20
16	1.6	3.2	2.3	12	2.1	2.7	240	1	20
17	1.5	2	0.5	8	1	0.5	164	2	10
18	1	2	0.4	8	1	0.5	140	2	10
19	1	1.5	0.5	8	0.5	0.5	164	2	10
20	0.5	1	0.3	8	0.3	0.5	160	2	10
21	1	0.8	1	8	1.5	0.5	160	2	10
22	0.5	2	0.4	8	0.5	0.5	164	2	10
23	1.5	2	0.5	8	1	0.4	140	2	10
24	0.5	2	0.3	8	0.6	0.3	164	2	10
25	1.5	1.8	1.2	8	1.5	1.5	140	2	10
26	0.4	0.5	0.6	8	0.5	0.5	164	2	10

DOI:10.4186/ej.2020.24.6.pp

Table 3. Problem instances and numbers of customer orders for each PC type.

Problem instance	Size of instance	PC type no.																									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
PPSP01	10 jobs	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
PPSP02	20 jobs	1	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
PPSP03	30 jobs	0	0	0	0	0	0	4	2	2	5	3	3	2	2	5	2	0	0	0	0	0	0	0	0	0	0
PPSP04	40 jobs	2	1	2	1	2	1	1	2	1	2	1	2	1	1	2	1	2	1	2	2	1	2	2	2	2	1
PPSP05	50 jobs	2	2	2	2	2	2	2	2	2	2	3	2	2	2	2	2	2	1	2	2	2	2	2	1	1	2
PPSP06	60 jobs	2	3	4	1	5	1	4	1	2	5	2	1	2	2	2	1	3	4	1	2	1	2	1	4	2	2
PPSP07	70 jobs	1	3	5	2	3	4	1	2	3	1	2	3	1	3	6	2	5	2	3	1	4	1	4	2	5	1
PPSP08	80 jobs	1	2	1	5	4	2	6	2	3	2	4	3	5	4	4	2	4	3	1	5	3	4	1	4	3	2
PPSP09	90 jobs	6	3	4	5	7	5	6	4	5	10	7	5	4	5	12	2	0	0	0	0	0	0	0	0	0	0
PPSP10	100 jobs	1	5	2	5	7	5	9	4	5	3	7	6	5	5	2	4	2	5	3	2	1	5	1	3	2	1

Table 4. Precast production data of a real industrial scheduling problem: *Case Study*.

PC no.	Name	Processing times (h)						Due date (h)	Penalties	
		Mold assembly	Reinforcement setting	Concrete casting	Concrete curing	Mold stripping	Product finishing		Earliness (ε_i)	Tardiness (τ_i)
1	Plank Slabs	0.5	0.5	1	22	2	0.8	96	0.4	4.2
2	Plank Slabs	0.5	0.5	1	22	2	0.8	96	0.4	4.2
3	Plank Slabs	0.5	0.5	1	22	2	0.8	192	0.4	4.2
4	Plank Slabs	0.5	0.5	1	22	2	0.8	192	0.4	4.2
5	Plank Slabs	0.5	0.5	1	22	2	0.8	120	0.4	4.2
6	Piles	0.5	1.25	1	27	2	0.5	48	1.7	4.2
7	Plank Slabs	0.5	0.5	1	22	2	0.8	120	0.4	4.2
8	Plank Slabs	0.5	0.5	1	22	2	0.8	144	0.4	4.2
9	Piles	0.5	1.25	1	27	2	0.5	240	2.1	4.2
10	Piles	0.5	1.25	1	27	2	0.5	168	1.1	4.2
11	Piles	0.5	1.25	1	27	2	0.5	336	1.1	4.2
12	Piles	0.5	1.25	1	27	2	0.5	216	1.1	4.2
13	Piles	0.5	1.25	1	27	2	0.5	216	2.1	4.2
14	Piles	0.5	1.25	1	27	2	0.5	144	2.1	4.2
15	Piles	0.5	1.25	1	27	2	0.5	192	2.1	4.2
16	Piles	0.5	1.25	1	27	2	0.5	216	2.1	4.2
17	Piles	0.5	1.25	1	27	2	0.5	216	2.1	4.2
18	Piles	0.5	3	1.5	27	2	0.5	288	3.8	10.5
19	Piles	0.5	3	1.5	27	2	0.5	336	3.8	10.5
20	Piles	0.5	3	1.5	27	2	0.5	336	3.8	10.5
21	Plank Slabs	0.5	0.5	1	22	2	0.8	48	0.4	4.2
22	Piles	0.5	3	1.5	27	2	0.5	360	5.3	10.5
23	Piles	0.5	3	1.5	27	2	0.5	360	8.7	10.5
24	Plank Slabs	0.5	0.5	1	22	2	0.8	168	0.4	4.2

5. Two variants of the discrete firefly algorithm, renamed DFA-1 [52] and DFA-2 [53].
6. The standard cuckoo search algorithm for multiple objectives (MOCS) [54].
7. Hybrid CS (HCS) [55].

All algorithms were implemented in MATLAB software and executed on a personal computer with an Intel Core i5 3.3GHz Processor and 4GB of memory. Table 5 lists the main parameter settings involved in each algorithm. To test the impact of the parameters on the performance, three parameter values of MOVNS and 27 different parameter combinations for NSGA-II and MOPPSM were obtained. Algorithms with each combination were run for all problem instances, based on extensive experiments. For the MOPPSM, values of N_{elite} and k_t were set to 4 and 2, respectively for all instances according to [4]. The best results of the algorithms measured using the above metrics were achieved over all problem instances with parameter settings: $N_{pop} = 150$ for the MOVNS, (N_{pop}, pc, pm) = (150, 0.9, 0.3) for the NSGA-II, and (N_{pop}, pc, pm) = (100, 0.7, 0.5) for the MOPPSM.

Parameter settings for MOBA, BA-HFS, MOFA, DFA-2, MOCS, and HCS employed the default values taken from the related literature, while parameter settings for DFA-1 were obtained by testing different types of parameters based on the suggested ranges in [52] over the PPSP instances. To fairly compare all the algorithms, this paper used the maximum CPU times (in seconds, s) as the termination criterion of the algorithm. Maximum CPU time is a widely used criterion for performance comparison of different metaheuristics [31, 56-58]. The maximum CPU times for solving each problem instance are shown in Table 6.

4.4. Computational Results and Comparison

Based on the parameter settings and the termination criterion described in subsection 4.3, each problem instance was conducted for 20 trials (runs) for each algorithm. In this paper, the total number of tested algorithms (A) was equal to 10. Computational results of the ten algorithms are illustrated in Tables 7-10, in which S^* is the reference set that consists of all non-dominated

DOI:10.4186/ej.2020.24.6.pp

Table 5. Parameter settings of all algorithms.

Parameter setting	MOVNS	NSGA-II	MOPPSM [4]	MOBA [49]	BA-HFS [50]	MOFA [51]	DFA-1 [52]	DFA-2 [53]	MOCS [54]	HCS [55]
Population size (N_{pop})	(50, 100, 150)	(50, 100, 150)	(50, 100, 150)	50	100	50	50	10	50	20
Crossover rate (pc)	-	(05, 0.7, 0.9)	(05, 0.7, 0.9)	-	-	-	-	-	-	-
Mutation rate (pm)	-	(0.1, 0.3, 0.5)	(0.1, 0.3, 0.5)	-	-	-	-	-	-	-
Elite number (N_{elite})	-	-	4	-	-	-	-	-	-	-
Termination value of local search (k_t)	-	-	2	-	-	-	-	-	-	-
Loudness (A_i)	-	-	-	0.9	0.9	-	-	-	-	-
Pulse rate (r_i)	-	-	-	0.9	0.9	-	-	-	-	-
Randomization parameter (α)	-	-	-	-	-	0.25	1	0.25	-	-
Light absorption (γ)	-	-	-	-	-	1	0.5	1	-	-
Attractiveness of firefly (β_0)	-	-	-	-	-	1	1	0.2	-	-
Discovery probability (p_a)	-	-	-	-	-	-	-	-	0.5	0.25
Lévy exponent (λ)	-	-	-	-	-	-	-	-	1.5	1.5

Table 6. Maximum CPU times for terminating the tested algorithms in each instance.

Problem instance	PPSP01	PPSP02	PPSP03	PPSP04	PPSP05	PPSP06	PPSP07	PPSP08	PPSP09	PPSP10	Case Study
Size of instance (jobs)	10	20	30	40	50	60	70	80	90	100	24
CPU time (s)	15	70	120	300	600	600	600	600	600	600	70

solutions of $\bigcup_{k=1}^{10} S_k$. S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10} were represented the non-dominated set of MOVNS, NSGA-II, MOPPSM, MOBA, BA-HFS, MOFA, DFA-1, DFA-2, MOCS and HCS, respectively. Then, values of the indicators SP, $D1_R$, $D_{\min}$ and $D_{\max}$ were calculated as the corresponding metric values of S_k. Graphical results of all metrics are illustrated in Fig. 5.

For performance comparison, the paired-sample t-test was conducted to compare MOVNS with the other nine algorithms based on four of the aforementioned indicators. Table 11 shows the p-value results of the above tests for the null hypothesis. There was no difference in the mean value of each indicator for a paired algorithm. The term "Paired-variable (A, B)" means that a paired t-test was conducted on the paired algorithm of (A, B) to judge whether algorithm B gave a better sample mean than algorithm A. Using a significance level of 0.05, the result was considered statistically significant if the corresponding p-value was less than 0.05, indicating that A performed better than B in the statistical sense.

4.4.1 Comparisons of the Proposed Algorithms

Results in Tables 7-11 indicated that the MOVNS was superior to NSGA-II. For the metric SP, the algorithm accepts only dominating solutions (or solutions with larger improvement on objectives) with a small search area in replacing if the value of SP becomes smaller. Conversely, a larger value of the spread can be expected if solutions are accepted from a larger area. As shown in Table 7, results showed that MOVNS produced more or a similar extent of non-dominated solutions than NSGA-II for almost all problem instances, except for the PPSP04. As shown in Table 8, $D1_R$ of MOVNS was less than $D1_R$ of NSGA-II for 7 problem instances, while $D1_R$ of both algorithms was equal to 0 for 2 problem instances. The zero value of $D1_R$ indicated that all members of S^* were generated by MOVNS and NSGA-II. Because most of the solutions in S^* were provided by MOVNS, the $D1_R$ value of NSGA-II was greater than that of MOVNS.

As shown in Tables 9 and 10, MOVNS gave better results on $D_{\min}$ than NSGA-II for all problem instances, while $D_{\max}$ of NSGA-II was greater than $D_{\max}$ of MOVNS for almost all problem instances, except for PPSP06. Moreover, results in Table 11 also showed that the MOVNS performance was statistically superior to NSGA-II. Thus, it can be concluded that the performance of MOVNS was better than the performance of NSGA-II, and MOVNS was reasonable and effective to solve the PPSP.

4.4.2 Comparisons among Eight Algorithms

From Tables 7-11, it can be concluded that the MOVNS outperformed MOPPSM, MOBA, BA-HFS, MOFA, DFA-1, DFA-2, MOCS, and HCS. As shown in Table 10, MOVNS provided all members of the set S^* on 3 problem instances, MOPPSM obtained the whole set S^* of two problem instances and MOCS obtained the whole set S^* for only one problem instance, while MOBA, BA-HFS, MOFA, DFA-1, and DFA-2 did not generate any members of S^* on 11 problem instances. MOVNS obtained more members of S^* than MOPPSM, MOBA, BA-HFS, MOFA, DFA-1, DFA-2, MOCS, and HCS on 4, 11, 11, 8, 9, 11, 9 and 9 problem instances, respectively. In Tables 8-10, the average values of metrics solutions of MOBA, BA-HFS, DFA-2, MOCS, and HCS were located far away from the members of the set S^*. MOPPSM performed better than NSGA-II, MOFA and DFA-1 on $D1_R$ and $D_{\max}$; however, $D_{\min}$ of MOPPSM was greater than that of NSGA-II. Statistical analysis results in Table 11 showed that MOVNS gave significantly better performance than MOPPSM on two metrics: SP and $D_{\min}$. Moreover, MOVNS was superior to MOFA, DFA-1, MOCS and HCS on four metrics, while it was inferior to MOBA, BA-HFS, and DFA-2 on only the metric SP. This result further justified the advantages of MOVNS according to the analysis of performance comparisons.

The performance comparison of all algorithms showed that MOVNS outperformed NSGA-II, MOPPSM, MOBA, BA-HFS, MOFA, DFA-1, DFA-2, MOCS, and HCS. The good performances of MOVNS mainly resulted from two reasons. The first related to its simplicity of the basic scheme and the non-dominated set updating process. The proposed MOVNS produced more non-dominated solutions. The second was that the proposed MOVNS implemented more than one neighborhood structure randomly to explore and exploit the solution space, while other comparing algorithms employed only one neighborhood structure to improve the convergence speed for the Pareto solution. Therefore, the advantages of MOVNS result from its simplicity in structure, strong local search ability and variable neighborhood mechanism. Thus, the MOVNS was shown to be a very competitive method for the considered PPSP.

5. Conclusions

Precast production scheduling plays an important role for decision-making in the precast fabrication industry. In this paper, the multi-objective precast production scheduling problem (PPSP) model with two conflicting objectives of optimizing makespan and total penalty costs of earliness and tardiness was investigated. A MOVNS was presented and tested on ten generated problem instances and a real case study of an industrial scheduling problem in a precast concrete company. The MOVNS metaheuristic was compared with the other algorithms, namely NSGA-II, MOPPSM, MOBA, BA-HFS, MOFA, DFA-1, DFA-2, MOCS, and HCS. Computational results showed that the MOVNS provided better solutions than the other algorithms in almost all problem instances. Thus, the MOVNS was determined as a promising approach to solve the PPSP. The MOVNS metaheuristic can provide

DOI:10.4186/ej.2020.24.6.pp

Table 7. Results of ten algorithms on SP.

Problem instance	MOVNS	NSGA-II	MOPPSM	MOBA	BA-HFS	MOFA	DFA-1	DFA-2	MOCS	HCS
PPSP01	138.37	138.37	138.37	104.72	136.05	138.76	138.91	140.69	138.37	138.37
PPSP02	100.98	67.64	72.44	68.37	119.75	102.48	57.18	60.80	70.63	73.73
PPSP03	285.16	97.50	97.80	104.13	546.81	120.58	69.47	532.29	98.05	96.47
PPSP04	154.08	155.29	154.65	93.76	103.49	131.54	112.15	101.29	142.20	24.95
PPSP05	94.97	44.33	18.84	101.99	103.82	131.54	105.98	139.12	122.83	95.49
PPSP06	161.41	92.15	100.15	550.45	43.56	131.54	154.19	89.07	99.43	91.36
PPSP07	141.42	100.69	100.11	131.29	79.93	101.57	150.65	109.63	98.64	57.71
PPSP08	161.41	135.09	130.33	131.29	91.07	54.48	101.84	85.39	14.58	82.01
PPSP09	304.14	93.86	125.48	211.91	455.34	168.90	102.97	139.57	96.85	93.64
PPSP10	216.71	75.10	143.20	578.98	92.39	96.77	98.77	49.07	40.27	102.35
Case Study	104.93	104.93	104.93	96.93	111.61	41.99	41.99	118.27	41.99	104.93
Average	169.42	100.45	107.85	197.62	171.26	110.92	103.10	142.29	87.62	87.36

Table 8. Results of ten algorithms on $D1_R$.

Problem instance	MOVNS	NSGA-II	MOPPSM	MOBA	BA-HFS	MOFA	DFA-1	DFA-2	MOCS	HCS
PPSP01	0.00	0.00	0.00	7.94	1.91	0.10	0.06	1.15	0.00	0.00
PPSP02	0.00	1.22	0.53	16.22	15.63	0.55	0.64	8.34	1.41	0.12
PPSP03	3.78	7.68	0.33	12.62	31.74	9.37	13.21	23.89	23.17	13.10
PPSP04	1.87	13.15	4.77	15.35	8.04	1.25	9.36	5.22	7.85	24.54
PPSP05	0.07	37.83	30.98	15.84	11.55	6.36	8.86	8.84	3.68	16.83
PPSP06	9.01	2.94	1.55	48.69	10.45	6.47	7.73	2.53	5.47	15.90
PPSP07	0.48	11.37	1.15	13.75	18.27	10.22	6.79	18.76	10.02	19.15
PPSP08	9.84	9.49	5.87	12.20	12.80	9.63	8.95	10.85	24.81	22.40
PPSP09	4.27	12.43	1.85	22.07	19.80	9.83	7.75	25.28	18.30	11.92
PPSP10	2.13	10.33	1.95	22.30	13.12	15.18	9.90	39.69	31.64	14.70
Case Study	0.00	0.00	0.00	16.99	8.27	0.76	0.76	5.11	0.76	0.00
Average	2.86	9.68	4.45	18.54	13.78	6.34	6.73	13.61	11.56	12.61

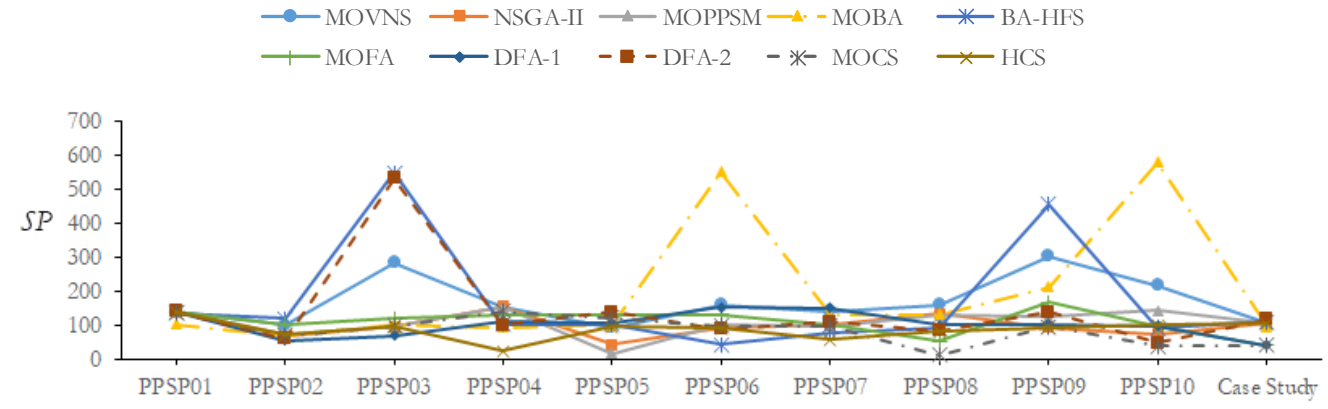
Table 9. Results of ten algorithms on D_{min}.

Problem instance	MOVNS	NSGA-II	MOPPSM	MOBA	BA-HFS	MOFA	DFA-1	DFA-2	MOCS	HCS
PPSP01	0.00	0.00	0.00	1.09	0.52	0.00	0.00	0.52	0.00	0.00
PPSP02	0.00	0.03	0.02	1.61	9.85	0.33	0.52	3.90	0.11	0.07
PPSP03	0.00	0.06	0.01	10.79	9.98	0.51	1.37	3.72	0.02	0.48
PPSP04	0.00	0.03	0.00	2.51	1.67	0.10	0.09	0.23	0.00	0.25
PPSP05	0.07	0.30	0.10	0.43	5.73	0.31	0.09	2.88	0.45	0.11
PPSP06	0.00	0.86	0.04	11.56	1.41	0.25	0.01	1.42	0.07	0.19
PPSP07	0.00	0.00	0.00	1.57	5.23	0.00	0.01	0.00	0.30	0.43
PPSP08	0.00	0.05	0.81	2.41	10.22	0.39	1.32	0.48	2.05	1.02
PPSP09	0.00	1.05	1.60	1.49	1.41	1.76	0.74	0.04	1.60	0.75
PPSP10	0.00	0.74	1.03	6.87	0.98	1.19	0.97	2.89	1.26	2.19
Case Study	0.00	0.00	0.00	9.37	8.27	0.76	0.76	0.99	0.76	0.00
Average	0.01	0.28	0.33	4.52	5.02	0.51	0.53	1.55	0.60	0.50

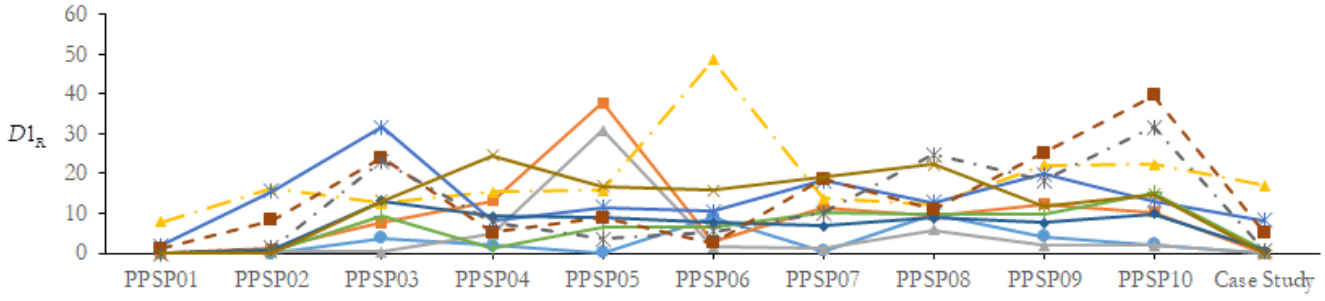
Table 10. Results of ten algorithms on D_{max}.

Problem instance	MOVNS	NSGA-II	MOPPSM	MOBA	BA-HFS	MOFA	DFA-1	DFA-2	MOCS	HCS
PPSP01	0.00	0.00	0.00	53.33	2.86	0.35	0.77	2.91	0.00	0.00
PPSP02	0.00	27.79	29.63	70.37	26.60	1.91	1.93	25.37	29.63	0.16
PPSP03	26.20	90.13	27.29	13.14	73.51	36.34	86.70	87.65	88.91	92.05
PPSP04	8.30	31.93	16.43	37.30	12.17	7.79	52.06	26.76	24.41	94.71
PPSP05	0.07	75.78	82.42	58.58	50.85	44.92	52.98	25.60	20.75	89.63
PPSP06	12.67	8.68	76.53	96.54	80.28	21.99	22.04	4.85	9.46	91.93
PPSP07	18.44	88.19	10.28	22.98	31.82	72.17	22.04	55.26	94.40	96.12
PPSP08	14.69	29.34	22.41	39.62	16.64	64.25	52.98	81.37	85.76	96.65
PPSP09	31.49	87.03	17.15	57.83	62.80	33.80	52.98	96.02	68.39	88.72
PPSP10	2.53	65.43	8.69	55.29	56.32	48.13	52.98	80.34	88.18	65.12
Case Study	0.00	0.00	0.00	39.23	8.27	0.76	0.76	25.73	0.76	0.00
Average	10.40	45.85	26.44	49.47	38.37	30.22	36.20	46.53	46.42	65.01

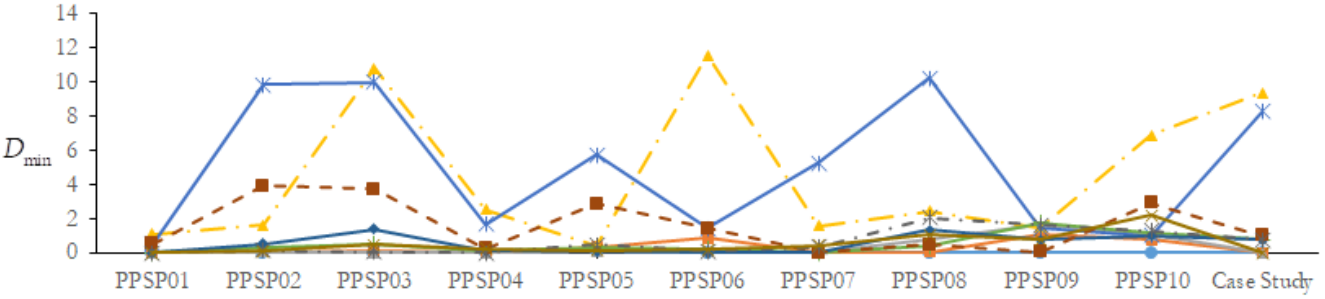
DOI:10.4186/ej.2020.24.6.pp



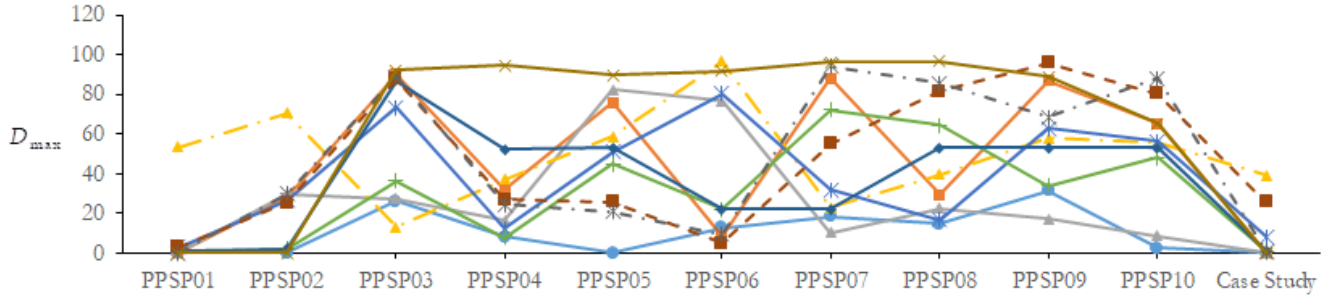
(a) Values of SP



(b) Values of $D1_R$



(c) Values of $D_{\min}$



(d) Values of $D_{\max}$

Fig. 5. Performance comparison of the ten algorithms using the SP, $D1_R$, $D_{\min}$ and $D_{\max}$ metrics.

Table 11. Paired-sample t-test for ten algorithms.

Paired-samples t-test	p-value ($D1_R$)	p-value (D_{min})	p-value (D_{max})	p-value (SP)
Paired-variable (MOVNS, NSGA-II)	0.040	0.022	0.002	0.007
Paired-variable (MOVNS, MOPPSM)	0.307	0.043	0.056	0.006
Paired-variable (MOVNS, MOBA)	0.000	0.003	0.001	0.693
Paired-variable (MOVNS, BA-HFS)	0.001	0.001	0.001	0.521
Paired-variable (MOVNS, MOFA)	0.020	0.006	0.009	0.007
Paired-variable (MOVNS, DFA-1)	0.006	0.004	0.002	0.01
Paired-variable (MOVNS, DFA-2)	0.008	0.003	0.001	0.221
Paired-variable (MOVNS, MOCS)	0.008	0.011	0.002	0.004
Paired-variable (MOVNS, HCS)	0.001	0.016	0.000	0.002

production schedules to assist managers in decision-making on precast concrete production.

In the future, extensions of MOVNS will be used to solve the PPSP by considering more complex manufacturing disturbance situations such as rush order arrival, due date change, uncertain processing time, and buffer size. Some novel methods to enhance performance including global search and intensification strategies on neighborhood search will also be considered.

Acknowledgement

This work was supported by the Prince of Songkla University, the Higher Education Research Promotion and the Thailand Education Hub for the Southern Region of ASEAN Countries (TEH-AC) Project Office of the Higher Education Commission.

References

[1] A. Konczak and J. Paslawski, "Decision support in production planning of precast concrete slabs based on simulation and learning from examples," *Procedia Engineering*, vol. 122, pp. 81–87, 2015.

[2] M. K. F. Othman, W. M. N. W. Muhammad, N. A. Hadi, and M. A. Azman, "The significance of coordination for industrialised building system (IBS) precast concrete in construction industry," in *Proceedings of the International Symposium on Civil and Environmental Engineering*, Wuhan, China, 2017, pp. 1–8.

[3] C. H. Ko, "Impact of the buffer size on precast fabrication," in *Proceedings of the 24th Annual Conference of the International Group for Lean Construction*, Boston, USA, 2016, pp. 83–92.

[4] C. H. Ko and S. Wang, "Precast production scheduling using multi-objective genetic algorithms," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8293–8302, 2011.

[5] B. D. A. Prata, A. R. Pitombeira-Neto, and C. J. D. M. Sales, "An integer linear programming model for the multi-period production planning of precast concrete beams," *Journal of Construction Engineering and Management*, vol. 141, no. 10, pp. 1–4, 2015.

[6] Z. Wang, H. Hu, and J. Gong, "Framework for modelling operational uncertainty to optimize offsite production scheduling of precast components," *Automation in Construction*, vol. 86, pp. 69–80, 2018.

[7] W. T. Chan and H. Hu, "An application of genetic algorithms to precast production scheduling," *Computers and Structures*, vol. 79, no. 17, pp. 1605–1616, 2001.

[8] W. T. Chan and H. Hu, "Production scheduling for precast plants using a flow shop sequencing model," *Journal of Computing in Civil Engineering*, vol. 16, no. 3, pp. 165–174, 2002.

[9] S. S. Leu and S. T. Hwang, "GA-based resource-constrained flow-shop scheduling model for mixed precast production," *Automation in Construction*, vol. 11, no. 4, pp. 439–452, 2002.

[10] C. H. Ko and S. Wang, "GA-based decision support systems for precast production planning," *Automation in Construction*, vol. 19, pp. 907–916, 2010.

[11] V. Benjaoran, N. N. Dawood, and B. Hobbs, "Flow shop scheduling model for bespoke precast concrete production planning," *Construction Management and Economics*, vol. 23, no. 1, pp. 93–105, 2005.

[12] W. Tharmmaphornphilas and N. Sareinpithak, "Formula selection and scheduling for precast concrete production," *International Journal of Production Research*, vol. 51, no. 17, pp. 5195–5209, 2013.

[13] Z. Yang, Z. Ma, and S. Wu, "Optimized flow shop scheduling of multiple production lines for precast production," *Automation in Construction*, vol. 72, no. 3, pp. 321–329, 2016.

[14] Z. Wang and H. Hu, "Improved precast production–scheduling model considering the whole supply chain," *Journal of Computing in Civil Engineering*, vol. 31, no. 4, pp. 1–12, 2017.

[15] D. S. Palmer, "Sequencing jobs through a multi-stage process in the minimum total time: A quick method of Obtaining a Near Optimum," *Journal of the Operational Research Society*, vol. 16, no. 1, pp. 101–107, 1965.

[16] J. N. Gupta, "A functional heuristic algorithm for the flow shop scheduling problem," *Journal of the Operational Research Society*, vol. 22, no. 1, pp. 39–47, 1971.

[17] H. G. Campbell, R. A. Dudek, and M. L. Smith, "A heuristic algorithm for the n job, m machine

sequencing problem," *Management Science*, vol. 16, no. 10, pp. B630-B637, 1970.
[18] D. G. Dannenbring, "An evaluation of flow shop sequencing heuristics," *Management Science*, vol. 23, no. 11, pp. 1174–1182, 1977.
[19] W. T. Chan and H. Hu, "Constraint programming approach to precast production scheduling," *Journal of Construction Engineering and Management*, vol. 128, no. 6, pp. 513–521, 2002.
[20] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and Operations Research,* vol. 24, no. 11, pp. 1097–1100, 1997.
[21] P. Hansen and N. Mladenović, "Variable neighbourhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.
[22] D. Lei and X. P. Guo, "Variable neighbourhood search for minimizing total tardiness on flow shop with batch processing machines," *International Journal of Production Research*, vol. 49, no. 2, pp. 519–529, 2011.
[23] D. Lei, "Variable neighbourhood search for two-agent flow shop scheduling," *Computers and Industrial Engineering*, vol. 80, pp. 125–131, 2015.
[24] D. Lei and Y. Zheng, "Hybrid flow shop scheduling with assembly operations and key objectives: A novel neighbourhood search," *Applied Soft Computing*, vol. 61, pp. 122–128, 2017.
[25] H. Eskandari and A. Hosseinzadeh, "A variable neighbourhood search for hybrid flow-shop scheduling problem with rework and set-up times," *Journal of the Operational Research Society*, vol. 65, no. 8, pp. 221–1231, 2014.
[26] J. Li, Q. Pan, and F. Wang, "A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem," *Applied Soft Computing*, vol. 24, pp. 63–77, 2014.
[27] M. A. Adibi, M. Zandieh, and M. Amiri, "Multi-objective scheduling of dynamic job shop using variable neighbourhood search," *Expert Systems with Applications*, vol. 37, no. 9, pp. 282–287, 2010.
[28] M. Amiri, M. Zandieh, M. Yazdani, and A. Bagheri, "A variable neighbourhood search algorithm for the flexible job-shop scheduling problem," *International Journal of Production Research*, vol. 48, no. 19, pp. 5671–5689, 2010.
[29] M. Zandieh and M. A. Adibi, "Dynamic job shop scheduling using variable neighbourhood search," *International Journal of Production Research*, vol. 48, no. 8, pp. 2449–2458, 2010.
[30] D. Lei and X. P. Guo, "Variable neighbourhood search for dual-resource constrained flexible job shop scheduling," *International Journal of Production Research*, vol. 52, no. 9, pp. 2519–2529, 2014.
[31] J. E. C. Arroyo, R. Ottoni, and A. P. Oliveira, "Multi-objective variable neighbourhood search algorithms for a single machine scheduling problem with distinct due windows," *Electronic Notes in Theoretical Computer Science*, vol. 281, pp. 5–19, 2011.
[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
[33] Y. Y. Han, D. W. Gong, X. Y. Sun, and Q. K. Pan, "An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem," *International Journal of Production Research*, vol. 52, no. 8, pp. 2211–2231, 2014.
[34] J. L. Andrade-Pineda, D. Canca. P. L. Gonzalez-R, and M. Calle, "Scheduling a dual-resource flexible job shop with makespan and due date-related criteria," *Annals of Operations Research*, vol. 291, pp. 5–35, 2020.
[35] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm and its applications to flow shop scheduling," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 3, pp. 392–403, 1998.
[36] A. Warszawski and E. Ishai, "Long range planning of prefabrication industry in a national economy (summary)," *Building and Environment*, vol. 17, no. 1, pp. 47–54, 1982.
[37] J. Behnamian and S. M. T. Fatemi Ghomi, "Multi-objective fuzzy multiprocessor flowshop scheduling," *Applied Soft Computing*, vol. 21, pp. 139–148, 2014.
[38] M. J. Geiger, "Randomized variable neighbourhood search for multi objective optimization," in *Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, Nottingham, UK, 2004, pp. 34–42.
[39] E. L. Ulungu, J. Teghem, and Ch. Ost, "Efficiency of interactive multi-objective simulated annealing through a case study," *Journal of the Operational Research Society*, vol. 49, no. 10, pp. 1044–1050, 1998.
[40] P. Czyzak and A. Jaszkiewicz, "Pareto-simulated annealin – a metaheuristic technique for multi-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 1, pp. 34–47, 1998.
[41] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.
[42] V. E. Armentano and J. E. C. Arroyo, "An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem," *Journal of Heuristics*, vol. 10, no. 5, pp. 463–481, 2004.
[43] J. E. C. Arroyo and V. E. Armentano, "A partial enumeration heuristic for multi-objective flow shop scheduling problems," *Journal of the Operational Research Society*, vol. 55, no. 9, pp. 1000–1007, 2004.
[44] J. M. Framinan and R. Leisten, "A multi-objective iterated greedy search for flow shop scheduling with makespan and flowtime criteria," *OR Spectrum*, vol. 30, no. 4, pp. 787–804, 2008.
[45] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. New York: John Wiley & Sons Ltd., 2010.
[46] P. Chutima and T. Kirdphoksap, "Solving many-objective car sequencing problems on two-sided

assembly lines using an adaptive differential evolutionary algorithm," *Engineering Journal*, vol. 23, no. 4, pp. 121–156, 2019.
[47] E. Zitzler, "Evolutionary algorithm for multi-objective optimization: Methods and applications," Ph.D., Technical Sciences, Swiss Federal Institute of Technology (ETH), Switzerland, 1999.
[48] S. Kaige, T. Murata, and H. Ishibuchi, "Performance evaluation of memetic EMO algorithms using dominance relation-based replacement rules on MOO test problem," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Washington, USA, 2003, pp. 14–19.
[49] X. S. Yang, "Bat algorithm for multi-objective optimization," *International Journal of Bio-Inspired Computation*, vol. 3, no.5, pp. 267–274, 2011.
[50] M. K. Marichelvam, T. Prabaharan, X. S. Yang, and M. Geetha, "Solving hybrid flow shop scheduling problems using bat algorithm," *International Journal of Logistics Economics and Globalisation*, vol. 5, no. 1, pp. 15–29, 2013.
[51] X. S. Yang, "Multiobjective firely algorithm for continuous optimization," *Engineering with Computers*, vol. 29, no. 2, pp. 175–184, 2013.
[52] M. K. Marichelvam, T. Prabaharan, and X. S. Yang. "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, 301–305, 2014.
[53] J. Schmid, L. Kieser, T. Hanne, and R. Dornberger, "Optimizing different parameters of a discrete firefly algorithm for solving the permutation flow shop problem," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, Honolulu, USA, 2017, pp. 1–6.
[54] X. S. Yang and S. Deb, "Multi-objective cuckoo search for design optimization," *Computers and Operations Research*, vol. 40, no. 6, pp. 1616–1624, 2013.
[55] X. Li and M. Yin, "A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 16, pp. 4732–4754, 2013.
[56] A. Duarte, J. J. Pantrigo, E. G. Pardo, and N. Mladenovic, "Multi-objective variable neighbourhood search: An application to combinatorial optimization problems," *Journal of Global Optimization*, vol. 63, no. 3, pp. 515–536, 2015.
[57] S. Selvi and D. Manimegalai, "Multi-objective variable neighbourhood search algorithm for scheduling independent jobs on computational grid," *Egyptian Informatics Journal*, vol. 16, no. 2, pp. 199–212, 2015.
[58] G. Palubeckis, "A variable neighbourhood search and simulated annealing hybrid for the profile minimization problem," *Computers and Operations Research*, vol. 87, pp. 83–97, 2017.

Wanatchapong Kongkaew was born in Trang province, Thailand in 1982. He received the B.Eng. degree in industrial engineering from Kasetsart University, Chonburi, Thailand, in 2004, the M.Eng. degree in industrial and systems engineering from Prince of Songkla University, Songkhla, Thailand, in 2007, and the D.Eng. degree in industrial engineering from Kasetsart University, Bangkok, Thailand, in 2013.

From 2013 to 2017, he was a Lecturer with the Industrial Engineering Department, Prince of Songkla University, Thailand. Since 2018, he has been an Assistant Professor with the Industrial Engineering Department, Prince of Songkla University, Thailand. His research interests include operations research and optimization, evolutionary computation and metaheuristics, production planning and control, discrete-event simulation, and logistics and supply chain engineering.

Dr. Wanatchapong was recieved a regulated engineering profession license at professional engineer level from Council of engineers of Thailand.

Lehuang Zong was born in Jiangxi provice, China in 1991. He received the B.Eng. degree in industrial engineering from Jiangxi University of Science and Technology, Ganzhou, China, in 2014. He is a student in Master of Engineering program in industrial and systems engineering at Prince of Songkla University, Songkhla, Thailand.

His research interests include optimization, evolutionary computation and metaheuristics, production planning and scheduling, and data analytics.