Research Article

Hybrid Metaheuristics and Linear Programming for Finite Capacity MRP in Multi-Stage Flexible Flow Shop with Permutation and Non-permutation Scheduling Options

Watchara Songserm and Teeradej Wuttipornpun*
Department of Industrial Engineering, Faculty of Engineering, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand

Chorkaew Jaturanonda
Department of Production Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand

* Corresponding author. E-mail: teeradejw@kmutnb.ac.th DOI: 10.14416/j.ijast.2018.05.004
Received: 11 May 2017; Accepted: 13 July 2017; Published online: 22 May 2018

Abstract
This paper presents a new algorithm for Finite Capacity MRP (FCMRP) in a multi-stage flexible flow shop. The proposed algorithm consists of four conventional metaheuristics namely, Genetic Algorithm (GA), Tabu Search (TS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA) hybridized with Linear Programming (LP). The objective is to minimize the total cost, which is the sum of tardiness, earliness, and flow-time costs. There are two main steps of the proposed algorithm. Firstly, an efficient sequence of orders is generated by the proposed metaheuristics in a way that reduce the total cost. In this step, the required operations of the orders are scheduled based on two scheduling options called permutation and non-permutation. Secondly, the total cost is minimized by the LP model. The required parameters of the metaheuristics are tuned by using real data from automotive companies. The result shows that the proposed algorithm significantly outperforms the existing algorithm, and GA obtains the best total cost.

Keywords: Finite capacity MRP, Flexible flow shop, Metaheuristic, Linear programming, Permutation scheduling, Non-permutation scheduling

1 Introduction

Material Requirement Planning (MRP) is a powerful planning tool used to generate production and purchasing plans. Unfortunately, the production plan obtained from the MRP system is reported that it could not be implemented on shop floor. A reason for this is that MRP assumes infinite resource capacity or constant lead-time [1]–[3]. This problem is later called Finite Capacity MRP (FCMRP). Since the FCMRP problem for industrial scale instances is normally the NP-hard class, the metaheuristic algorithm is one of the appropriate approaches to solve the problem [4]–[6]. In the literature, there are two approaches of metaheuristic algorithms developed for the FCMRP problems summarized in Table 1. The first approach is called population search algorithm. It includes Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Cuckoo Search (CS) [7]–[18]. The second approach is called non-population search algorithm. It includes TS, Iterated Local Search (ILS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA) [19]–[34]. Based on the principles of each metaheuristic, the population search algorithm

Please cite this article as: W. Songserm, T. Wuttipornpun, and C. Jaturanonda, "Hybrid metaheuristics and linear programming for finite capacity MRP in multi-stage flexible flow shop with permutation and non-permutation scheduling options," *KMUTNB Int J Appl Sci Technol*, vol. 11, no. 3, pp. 173–183, Jul.–Sep. 2018.

Table 1: Algorithm developed for the FCMRP problems

Approach	Authors	Production Shops	Objectives
Population Search			
GA	Chen *et al*. [7]	Job shop	Tardiness, idle time and makespan
	Wang & Liu [8]	Flexible flow shop	Makespan
	Rahman, Sarker & Essam [9]	Permutation flow shop	Setup and holding cost
PSO	Kuo *et al*. [10]	Flow shop	Makespan
	Tang *et al*., 2016 [11]	Flexible flow shop	Makespan
	Eddaly, Jarboui & Siarry [12]	Blocking flow shop	Makespan
ACO	Yagmahan & Yenisey [13]	Flow shop	Makespan and flow-time
	Ahmadizar [14]	Permutation flow shop	Makespan
	Zhang & Jing [15]	Permutation flow shop	Makespan
CS	Marichelvam, Prabaharan & Yang [16]	Flexible flow shop	Makespan
	Dasgupta & Das [17]	Flow shop	Makespan and flow-time
	Komaki *et al*. [18]	Flow shop	Makespan
Non-population Search			
TS	Grabowski & Pempera [19]	Permutation flow shop	Makespan
	Chen, Pan & Wu [20]	Permutation flow shop	Makespan
	Liao & Huang [21]	Two machine flow shop	Makespan
	Bozejko, Pempera & Smutnicki [22]	Flexible flow shop	Makespan
ILS	Pan & Ruiz [23]	Flow shop	Flow-time
	Dong *et al*. [24]	Permutation flow shop	Flow-time
	Wang *et al*. [25]	Flow shop	Makespan
	Xua *et al*. [26]	Permutation flow shop	Makespan
VNS	Costa, Goldbarg & Goldbarg [27]	Flow shop	Flow-time
	Hallah [28]	Permutation flow shop	Earliness and tardiness
	Lei [29]	Flow shop	Makespan and tardiness
	Rifai, Ngugen & Dawal [30]	Permutation flow shop	Makespan, and average tardiness
SA	Low [31]	Flow shop	Flow-time
	Naderi, Moghaddam, & Khalili [32]	Flow shop	Makespan and tardiness
	Jarosław, Czeslaw & Domonik [33]	Flow shop	Makespan and tardiness
	Nikzad *et al*. [34]	Flexible flow shop	Makespan

seems to be more efficient than the other. However, there is no strong conclusion that the population search approach is always better than the non-population one. It depends on mostly about the problem characteristics.

This paper presented four metaheuristic algorithms, which are GA, TS, SA, and VNS, hybridized with an LP model to solve the industrial scale FCMRP problem. The metaheuristics are used to generate the efficient sequence of orders on machines based on permutation and non-permutation scheduling concepts, whereas the LP model is used to schedule the required operations of orders to the optimal start times. The proposed algorithm is intently developed to improve the solution obtained from the existing FCMRP algorithm developed by Wuttipornpun and Yenradee [6]. The objective is to minimize the total cost defined as the sum of tardiness, earliness, and flow-time costs. The production shop used to evaluate the performance of the proposed algorithm is a multi-stage flexible flow shop. The planning horizon is one month without overtimes and preemptive options.

The remaining of the paper is organized as follows. Section 2 deals with details of the proposed algorithm. Case studies and experiments for parameter tuning and performance evaluation are explained in Section 3. Results and discussion are provided in Section 4. Finally, the conclusion of this paper and recommendations for future research are given in Section 5.

```
overall procedure: the proposed algorithm
input: orders, bill of materials, and machines information
output: solution from the proposed algorithm
begin
//Step 1: Construct the efficient sequence of orders
        apply GA, TS, SA, and VNS to generate the sequence of orders;
        schedule the required operations of orders to less tardiness machines;
//Step 2: Optimize the start times of operations
        apply LP to the schedule from step 1;
output: solution from the proposed algorithm
end;
```

Figure 1: Pseudo code of the proposed algorithm.

```
overall procedure: GA algorithm
input: orders, bill of materials, and machines information, GA parameters
output: solution from the GA algorithm
begin
        n←0; //n: generation number
//Step 1: Construct initial population
        create Population(n) consists of Y chromosomes; //Y: population size
        while (not terminating condition of GA) do
//Step 2: Evaluate performance of each chromosome
            Each chromosome is evaluated its performance through the total cost
//Step 3: Selection
            apply elitist method to Population(n);
            select Parent(n) from Population(n) by roulette wheel method;
//Step 4: Apply genetic operators
            crossover Parent(n) to get Offspring(n);
            mutate Offspring(n) to get MOffspring(n); //MOffspring: mutated offspring
            Population(n+1)←a sequence from elitist method and MOffspring(n);
            calculate total cost of Population(n+1);
            n←n+1;
        end;
        select the sequence with minimum total cost from Population(n);
//Step 5: Apply the LP model
        apply LP to the sequence from step 4;
output: solution from the GA algorithm
end;
```

Figure 2: Pseudo code of GA.

2 Details of the Proposed Algorithm

The proposed algorithm has two main steps as shown in Figure 1. They are explained as follows.

Step 1: Construct the efficient sequence of orders

The objective of this step is to construct the efficient sequence of orders by the conventional GA, SA, VNS, and TS algorithms. The conventional procedures of each algorithm are shown in Figures 2, 3, 4, and 5 respectively. Note that details of these metaheuristics are available in Artificial Intelligent (AI) books [35], [36].

In each iteration of a given metaheuristic algorithm, the sequence of orders is evaluated to obtain the total cost calculated by Equation (1), where Q_i, t_i, e_i, and f_i are the order quantity, tardiness, earliness, and flow-time of the i^{th} order. CT_i, CE_i, and CF_i are the cost per unit of tardiness, earliness, and flow-time of the i^{th} order.

```
overall procedure : SA algorithm
input : orders, bill of materials, and machines information, SA parameters
output : solution from the SA algorithm
begin
        create S; //S: : a sequence of orders
        set T; // T: initial temperature
        while (not terminating condition of SA) do
        generate neighbourhood sequences from S;
        evaluate the neighbourhood sequences;
        select the improved sequence;
            if (improved sequence < S)
                S←improved sequence;
            else
                accept improved sequence as S with probability;
            end;
          update T;
        end;
output : solution from the SA algorithm
end;
```

Figure 3: Pseudo code of SA.

```
overall procedure: VNS algorithm
input: orders, bill of materials, and machines information, VNS parameters
output: solution from the VNS algorithm
begin
        create S; //S: a sequence of orders
        while (not terminating condition of VNS) do
          NS= 1; //NS: neighbourhood structures
          while (not terminating condition of NS) do
            S'← perturbation S;
            generate neighbourhood sequences from S';
            evaluate neighbourhood sequences;
            select the improved sequence;
            if (improved sequence < S)
              S←improved sequence;
              NS = 1;
            else
              NS = NS + 1;
            end;
          end;
        end;
output: solution from the VNS algorithm
end;
```

Figure 4: Pseudo code of VNS.

```
overall procedure: TS algorithm
input: orders, bill of materials, and machines information, TS parameters
output: solution from the TS algorithm
begin
        create S; //S: a sequence of orders
        while (not terminating condition of TS) do
            generate neighbourhood sequences from S and check with TL;
            evaluate the neighbourhood sequences;
            select the improved sequence and update TL;
            S←improved sequence;
        end;
output: solution from the TS algorithm
end;
```

Figure 5: Pseudo code of TS.

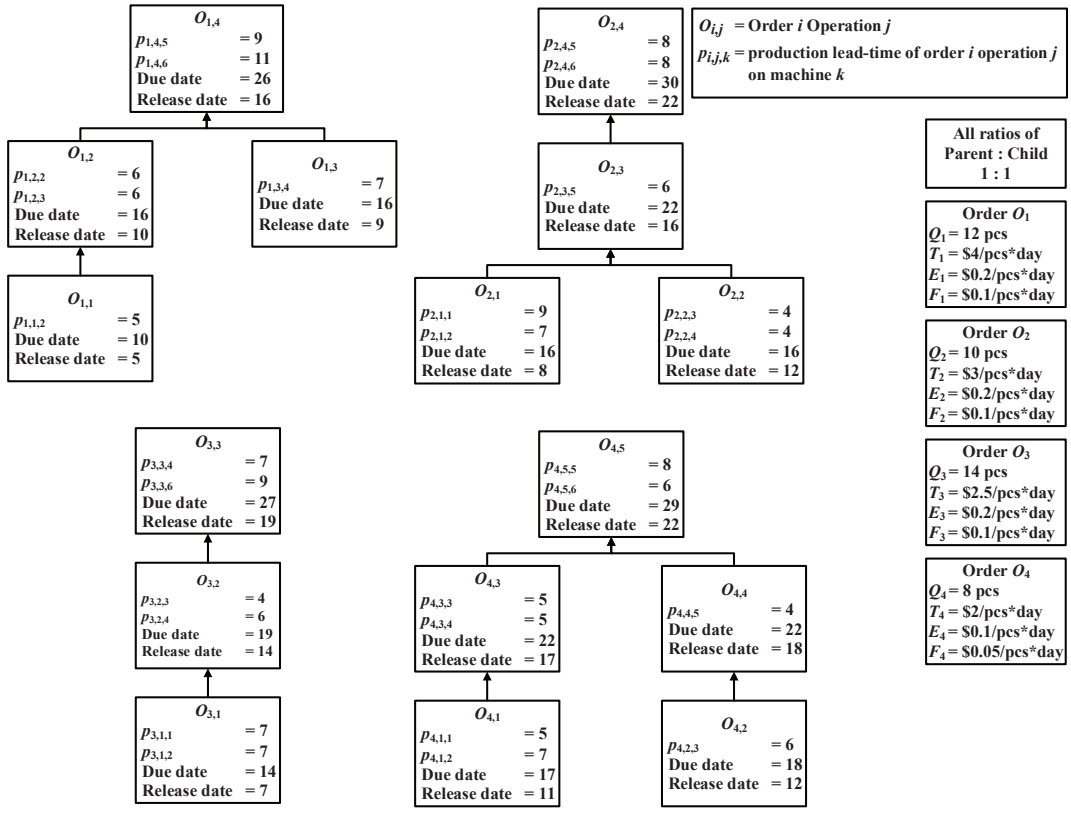

Figure 6: Details of operations after applying the VMRP explosion.

$$Total\ cost = \sum_{i=1}^{n} CT_i Q_i t_i + \sum_{i=1}^{n} CE_i Q_i e_i + \sum_{i=1}^{n} CF_i Q_i f_i \quad (1)$$

To calculate the total cost, the required operations of orders in the sequence must be scheduled to machines in order to calculate the tardiness, earliness, and flow-time. It can be done by the following steps: (1) the orders in the sequence are exploded by variable lead-time MRP (VMRP) to obtain the required operations (see Wuttipornpun and Yenradee [6] for details of VMRP), (2) these operations are scheduled to less tardiness machines by forward scheduling based on permutation and non-permutation scheduling options, and (3) the tardiness, earliness, and flow-time of each order are calculated by the start time, due time, and completion time.

Details of the required operations ($O_{i,j}$) of the four orders (O_i) after VMRP explosion shown in Figure 6 are used to demonstrate this step. Suppose that a sequence of orders is O_4 ⇨ O_3 ⇨ O_2 ⇨ O_1. When the permutation option is selected, all operations must be scheduled based on the sequence of orders in a way that the operations of the first order in the sequence are scheduled to less tardiness machines first, and the operations of the second order in the sequence are scheduled next and so on. By this way, it is obvious that all machines have the same sequence of operations that complies with the permutation sequencing concept (see second column of Table 2, M_1 to M_6). When the non-permutation option is selected, the operation that is scheduled later based on the sequence of orders may be started earlier than the operation that is scheduled before. It results in different sequence of operations of each machine as shown in the fourth column of Table 2. It can be seen that the sequence of operations on M_4 is different from other machines. The total costs of these sequences are shown in Table 2. Note that the permutation and non-permutation scheduling concepts are applied and evaluated separately.

Table 2: Total cost of a sequence of orders based on permutation and non-permutation schedules

Sequence of Orders (O_i)	Sequence of Operations ($O_{i,j}$) on Machines (Permutation)	Total Cost (Permutation)	Sequence of Operations ($O_{i,j}$) on Machines (Non-permutation)	Total Cost (Non-permutation)
$O_4 \Rightarrow O_3 \Rightarrow O_2 \Rightarrow O_1$	M_1: $O_{4,1}$ M_2: $O_{3,1} \Rightarrow O_{2,1} \Rightarrow O_{1,1} \Rightarrow O_{1,2}$ M_3: $O_{4,2} \Rightarrow O_{3,2} \Rightarrow O_{2,2}$ M_4: $O_{4,3} \Rightarrow O_{3,3} \Rightarrow O_{1,3}$ M_5: $O_{4,4} \Rightarrow O_{2,3} \Rightarrow O_{2,4}$ M_6: $O_{4,5} \Rightarrow O_{1,4}$	\$1,522.00	M_1: $O_{4,1}$ M_2: $O_{3,1} \Rightarrow O_{2,1} \Rightarrow O_{1,1} \Rightarrow O_{1,2}$ M_3: $O_{4,2} \Rightarrow O_{3,2} \Rightarrow O_{2,2}$ M_4: $O_{1,3} \Rightarrow O_{4,3} \Rightarrow O_{3,3}$ M_5: $O_{4,4} \Rightarrow O_{2,3} \Rightarrow O_{2,4}$ M_6: $O_{4,5} \Rightarrow O_{1,4}$	\$1,325.20

Step 2: Optimize the start times of the operations

From the first step, the operations are scheduled to machines by forward scheduling. As a result, the start times of some operations may not be optimal. This step tries to determine the optimal start times of all operations by the LP model in order to minimize the total cost. In the LP model, only the start times of operations are optimized, whereas the sequence of operations on machines obtained from the first step is still maintained. The LP model is formulated by using indices, parameters, and variables explained as follows.

Indices

i = index of order starting from 1 to n

j = index of operation of order i starting from 1 to m

j^* = index of the last operation of order i

k = index of machine starting from 1 to s

j' = index of child operations of order i operation j

j'' = index of the first operations of order i

q = index of operations on machine starting from 1 to t. For example, there are three operations on machine 4 (M_4), which are $O_{4,3} \Rightarrow O_{3,3} \Rightarrow O_{1,3}$ (see second column in Table 2), when $q = 2$, it refers to the second operation on this machine ($O_{3,3}$).

Sets and Parameters

$p_{i,j}$ = processing time of operation i,j (order i operation j)

$p_{i,j,k}$ = processing time of operation i,j (order i operation j) on machine k

d_i = due time of order i

CT_i = cost per unit of tardiness of order i

CE_i = cost per unit of earliness of order i

CF_i = cost per unit of flow-time of order i

Q_i = quantity of order i

Seq_k = set of sequence of operation i,j on machine k. For example, there are three operations on machine 4 (M_4), which are $O_{4,3} \Rightarrow O_{3,3} \Rightarrow O_{1,3}$ (see second column in Table 2), so Seq_4 = {(4,3), (3,3), (1,3)}.

$CH_{i,j}$ = set of child operations of operation i,j (index j). For example, $O_{1,4}$ consists of $O_{1,2}$ and $O_{1,3}$ (see Figure 6), so $CH_{1,4} = \{2, 3\}$.

FO_i = set of the first operations of order i. For example, the first operations of order 4 are $O_{4,1}$ and $O_{4,2}$ (see Figure 6), so $FO_4 = \{1, 2\}$.

Decision Variables

$x_{i,j}$ = start time of operation i,j

$x_{i,j,k}$ = start time of operation i,j on machine k

c_i = completion time of order i

$c_{i,j}$ = completion time of operation i,j

f_i = flow-time of order i

e_i = earliness of order i

t_i = tardiness of order i

Z = total cost

Objective function [Equation (2)]

$$Minimize\ Z = \sum_{i=1}^{n} CT_i Q_i t_i + \sum_{i=1}^{n} CE_i Q_i e_i + \sum_{i=1}^{n} CF_i Q_i f_i \quad (2)$$

Constraints

Constraint (3) is used to maintain the sequence of operations on each machine

$$x_{q*,k} \geq x_{q,k} + p_{q,k} \qquad \forall k,\ \forall q,\ \text{and}\ q* > q \qquad (3)$$

Constraint (4) is used to maintain the precedence relationships of operations

$$x_{i,j} \geq c_{i,j'} \qquad \forall i,\ \forall j,\ \text{and}\ j' \in CH_{i,j} \qquad (4)$$

Constraints (5) – (11) are used to calculate completion time, tardiness, earliness, and flow-time

For completion time:

$$c_i \geq x_{i,j^*} + p_{i,j^*} \qquad \forall i,\ \forall j^* \qquad (5)$$

$$c_{i,j} \geq x_{i,j} + p_{i,j} \qquad \forall i,\ \forall j \qquad (6)$$

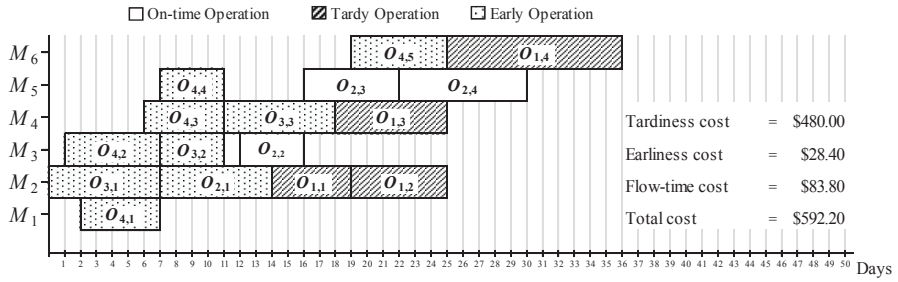

Figure 7: Total cost after applying the LP model (Permutation).

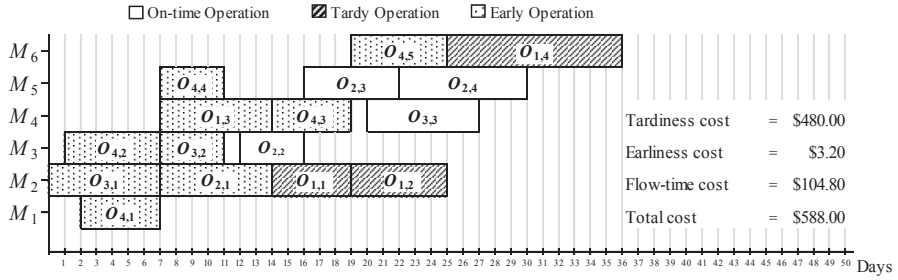

Figure 8: Total cost after applying the LP model (Non-permutation).

For tardiness:

$$t_i \geq c_i - d_i \qquad \forall i \qquad (7)$$
$$t_i \geq 0 \qquad \forall i \qquad (8)$$

For earliness:

$$e_i \geq d_i - c_i \qquad \forall i \qquad (9)$$
$$e_i \geq 0 \qquad \forall i \qquad (10)$$

For flow-time:

$$f_i \geq c_i - x_{i,j''} \qquad \forall i,\ \forall j'' \in FO_i \qquad (11)$$

Non-negativity constraint [Constraint (12)]

$$x_{i,j}, x_{i,j,k} \geq 0 \qquad \forall i, \forall j, \forall k \qquad (12)$$

Figures 7 and 8 show the results after applying the LP model. It can be seen that the total costs shown in Table 2 are dramatically reduced. This shows that our LP model is very efficient. Also, it observes that the total cost obtained from non-permutation scheduling is slightly better than that obtained from permutation scheduling. However, it can be happened in the opposite direction depending on data sets.

3 Case Studies and Experiments

This section consists of three parts. The first part deals with the details of case studies. The second part deals with the experiment for parameters tuning of GA, SA, TS, and VNS. The final part is the experiment to evaluate the performance of the proposed algorithm. They are explained as follows.

3.1 *Details of case studies*

Our case studies are derived from three automotive-part companies. The different characteristics are summarized in Table 3, whereas the common characteristics are summarized as follows.

- The production shop is a multi-stage flexible flow shop
- The planning horizon is one month
- Lot sizing rule is lot-for-lot since it results in lowest inventory
- No preemptive schedule and overtime

Table 3: Different characteristics of case studies

Characteristics	Case 1	Case 2	Case 3
Number of finished products	16	19	18
Total order quantities	1,512	1,735	1,986
Min/Max levels in bill of materials	2/5	3/5	4/6
Number of operations	520	610	690
Number of machines	18	20	20

3.2 *Experiments for parameters tuning of GA, SA, TS, and VNS*

The required parameters for GA, SA, TS, and VNS are summarized in Tables 4 and 5. These parameters are considered as independent variables for a factorial experiment. The experiment is conducted with five replicates for all case studies to obtain the response variable, which is the total cost before applying the LP model. The stopping criterion for each run is 120 minutes (set by the planner). Note that data in Tables 4 and 5 are obtained by a set of screening experiments beforehand.

Table 4: Parameters for GA

Algorithm	p_s	Crossover	p_c	Mutation	p_m
GA	5, 10	PMX, PBX	0.6, 0.8	SWAP, INSERT	0.005, 0.01

p_s : population size, p_c : crossover probability, p_m : mutation probability.

To obtain the practical solution for the planner, the best common parameter setting for all case studies is required rather than seeking the best setting for an individual case study. To obtain the best common setting for a given algorithm, the relative percentage deviation over the best total cost of each case study (*RPD*) and the average relative percentage deviation for all case studies (*ARPD*) are determined by using Equations (13) and (14), where Sol_{Alg} is the total cost obtained from each run, Sol_{Best} is the minimum total cost across all runs, v is the index of case study, and V is the number of case studies. The best common setting is a setting obtained at the minimum *ARPD*, since it guarantees that the total cost from this setting is very close to its best total cost (near best total cost).

$$RPD_v = \frac{Sol_{Alg(v)} - Sol_{Best(v)}}{Sol_{Best(v)}} \times 100\% \quad (13)$$

$$ARPD = \frac{1}{V}\sum_{v=1}^{V}(RPD)_v \quad (14)$$

3.3 *Experiments to evaluate the performance of the proposed algorithm*

To evaluate the performance, we compare the total cost obtained from the proposed algorithm with a benchmark algorithm (existing FCMRP algorithm by Wuttipornpun and Yenradee [6]). There are 8 options available in the proposed algorithm. They all use the best common settings obtained from section 3.2, and use the following notations.

GA-P: GA/Permutation
GA-NP: GA/Non-permutation
SA-P: SA/Permutation
SA-NP: SA/Non-permutation
TS-P: TS/Permutation
TS-NP: TS/Non-permutation
VNS-P: VNS/Permutation
VNS-NP: VNS/Non-permutation

All options in the proposed algorithm are set as independent variables for a one-way ANOVA experiment. The response variable is the total cost after applying the LP model. To determine the performance improvement of the proposed algorithm,

Table 5: Parameters for TS, SA, and VNS

Algorithm	No. of NBH Sequences	NBH Operators	TL Size	No. of Iterations for each Temperature	Initial Temperature	Alpha	Shaking Operators
TS	10, 15	SWAP, INSERT	30, 40, 50	-	-	-	-
SA	5, 10	SWAP, INSERT	-	1, 10	80, 100	0.8, 0.9	-
VNS	70, 90	SWAP and INSERT	-	-	-	-	SWAP, INSERT

NBH: Neighborhood, TL: tabu list, Alpha: temperature reduction.

the relative percentage improvement over the best total cost obtained from the benchmark algorithm (*RPI*) is calculated using Equation (15), where Sol_{Alg} is the total cost of each algorithm, Sol_{BM} is the best total cost from the benchmark algorithm.

$$RPI_v = \frac{\left|Sol_{Alg} - Sol_{BM}\right|}{Sol_{BM}} \times 100\% \quad (15)$$

4 Results and Discussions

The experiments mentioned in section 3 are run by using C# language on a Core i5 Processor 2.5 GHz with 4 GB of RAM desktop computer. The results are analyzed by using a statistical software called Minitab 17.

Since we evaluate the proposed algorithm using three different case studies, the best setting for each case study is then varied depending on the different characteristics. Therefore, the best common setting for all case studies is more practical for the planner. Tables 6 and 7 show the best common setting of each algorithm recommended for all case studies. It is a setting selected from the minimum value of *ARPD*. It is also observed that the permutation option and non-permutation have different parameter settings.

Table 6: Best common parameter settings for GA

Algorithms	p_s	Crossover	p_c	Mutation	p_m
GA-P	10	PBX	0.6	SWAP	0.005
GA-NP	10	PBX	0.6	INSERT	0.01

p_s : population size, p_c : crossover probability, p_m : mutation probability.

Table 8 shows the best and near best total costs obtained from all available options in the proposed algorithm. Based on *RPD*, it shows that there is only a narrow range of 0–3.85% between the near best total cost and its best total cost. From the *ARPD* value, it can be seen that VNS-P and VNS-NP obtain the minimum deviation of 0.61%, whereas the SA-NP obtains the maximum deviation of 1.58%. However, the deviations are very small. This proves that our method to select the near best total cost works very well.

Table 9 shows the improved total cost, Relative Percentage Improvement (*RPI*), and the rank after applying the LP model. The rank numbers shown in this table are obtained from Tukey's comparison test. The lower rank means better performance than the higher rank. It can be seen that the near best total cost in Table 8 is substantially reduced when applying the LP

Table 7: Best common parameter settings for TS, SA, and VNS

Algorithms	No. of NBH Sequences	NBH Operators	TL Size	No. of Iterations for each Temperature	Initial Temperature	Alpha	Shaking Operators
TS-P	10	SWAP	30	-	-	-	-
TS-NP	10	SWAP	50	-	-	-	-
SA-P	5	INSERT	-	1	100	0.9	-
SA-NP	10	INSERT	-	10	80	0.9	-
VNS-P	70	SWAP and INSERT	-	-	-	-	SWAP
VNS-NP	90	SWAP and INSERT	-	-	-	-	INSERT

NBH: Neighborhood, TL: tabu list, Alpha: temperature reduction.

Table 8: Best total cost, near best total cost, and RPD before applying the LP

Algorithms	Best Total Cost ($)/Near Best Total Cost ($)/RPD_{TC}			Min. *ARPD*
	Case 1	Case 2	Case 3	
GA-P	8,203 / 8,275 / 0.88%	13,439 / 13,634 / 1.45%	17,836 / 17,836 / 0.00%	0.78%
GA-NP	8,943 / 9,072 / 1.45%	13,758 / 13,758 / 0.00%	18,501 / 18,760 / 1.40%	0.95%
SA-P	10,082 / 10,082 / 0.00%	15,162 / 15,589 / 2.82%	19,965 / 19,965 / 0.00%	0.94%
SA-NP	8,621 / 8,953 / 3.85%	13,656 / 13,656 / 0.00%	18,572 / 18,738 / 0.89%	1.58%
TS-P	8,247 / 8,277 / 0.36%	13,470 / 13,534 / 0.47%	17,971 / 18,027 / 0.31%	0.38%
TS-NP	8,475 / 8,632 / 1.85%	14,068 / 14,138 / 0.50%	18,451/ 18,525 / 0.42%	0.92%
VNS-P	7,824 / 7,883 / 0.75%	12,688 / 12,825 / 1.08%	16,996 / 16,996 / 0.00%	0.61%
VNS-NP	8,509 / 8,509 / 0.00%	13,945 / 14,201 / 1.84%	18,223 / 18,223 / 0.00%	0.61%

Table 9: Total costs after applying the LP model

Algorithms	Total Cost ($)/*RPI*/Ranking			Average
	Case 1	Case 2	Case 3	
Existing FCMRP [6]	9,424 (Benchmark)	16,016 (Benchmark)	19,280 (Benchmark)	14,906.58 (Benchmark)
GA-P	7,300 / 22.54% / 4	11,960 / 25.33% / 4	13,707 / 28.91% / 6	10,988.99 / 25.59% / 3
GA-NP	7,016 / 25.55% / 3	10,265 / 35.91% / 1	12,814 / 33.54% / 1	10,031.70 / 31.66% / 1
SA-P	8,305 / 11.87% / 6	12,770 / 20.27% / 6	13,894 / 27.94% / 4	11,656.33 / 20.03% / 4
SA-NP	7,056 / 25.13% / 3	10,489 / 34.51% / 2	13,787 / 28.49% / 6	10,444.25 / 29.94% / 2
TS-P	7,540 / 19.99% / 5	12,480 / 22.07% / 4	14,116 / 26.79% / 5	11,378.91 / 22.95% / 3
TS-NP	7,057 / 25.12% / 3	12,155 / 24.10% / 5	13,681 / 29.04% / 3	10,964.69 / 26.44% / 3
VNS-P	6,629 / 29.66% / 1	11,476 / 28.35% / 3	13,583 / 29.55% / 3	10,562.67 / 29.18% / 2
VNS-NP	6,845 / 27.37% / 2	11,961 / 25.32% / 4	13,183 / 31.62% / 2	10,663.34 / 28.47% / 2

model. This proves that the LP model is very efficient. It allows some operations to start prior to the release times from VMRP to reduce the tardiness cost, which is the main cost in our objective function.

Based on the *RPI* value, it is obvious that all available options in the proposed algorithm significantly outperform the benchmark algorithm. The improvement gap over the benchmark algorithm is in a range of 20.03–31.66%. GA-NP offer the highest improvement of 31.66%, whereas SA-P offers the lowest improvement of 20.03%. For our problem characteristics, GA, SA, and TS with non-permutation option outperform the permutation option, while it happens in the opposite direction for VNS.

Comparing the required runtimes between the proposed algorithm and the benchmark algorithm, which are 120 and 10 minutes, respectively, the proposed algorithm requires substantially longer runtime than the benchmark algorithm. However, the total cost obtained from the proposed algorithm is dramatically lower than that obtained from the benchmark algorithm. Therefore, it is worth for the planner to wait slightly longer.

5 Conclusions

This paper presents a new algorithm for the FCMRP problem in automotive multi-stage flexible flow shop. It is a hybrid of metaheuristic algorithms (GA, SA, TS, and VNS) and linear programming model. The objective is to improve the solution obtained from the existing FCMRP algorithm done by Wuttipornpun and Yenradee [6] and also offer a practical alternative method for the planner. There are two main steps in the proposed algorithm. In the first step, the efficient sequence of orders is determined by the proposed metaheuristic algorithms. After obtaining the sequence of orders, the required operations of these orders are scheduled to the less tardiness machines by using the forward heuristic with permutation and non-permutation scheduling options. In the second step, the start times of the required operations are optimally determined using the LP model to minimize the total cost, which is the sum of tardiness, earliness, and flow-time costs.

There are 8 options available in the proposed algorithm. They are controlled by the parameters of metaheuristic algorithms. Since we evaluate the proposed algorithm by different case studies, the best parameter setting for each case study may be set differently. To offer a practical setting for the planner, the best common setting for all case studies is required rather than the best setting for an individual case. Based on the best common setting, it offers a solution which slightly deviates from its best solution in a deviation range of 0–3.85%.

For the effectiveness perspective, all options in the proposed algorithm significantly outperform the existing algorithm for all case studies. GA-NP offers the best improvement of 31.66%. Based on the different characteristics of case studies, we can conclude that the proposed algorithm can be applied to various industrial situations. Although the proposed algorithm requires a significant longer runtime than the existing algorithm, it is worth for the planner to wait since the solution quality of the proposed algorithm is much better.

There are some interesting research gaps for future investigations. The lot sizing policy is only lot-for-lot. Other lot-sizing techniques should be studied. Since it is especially designed for the flexible flow shop, the performance of the proposed algorithm in other manufacturing shops such as job shop, open shop should be investigated. The production overtime and order preemption are also of interest because they can significantly reduce the total cost. Therefore, a new algorithm should be developed to address these problems.

Acknowledgments

This research has been supported by Faculty of Engineering, King Mongkut's University of Technology North Bangkok.

References

[1] P. B. Nagendra and S. K. Das, "Finite capacity scheduling method for MRP with lot size restrictions," *International Journal of Production Research*, vol. 39, pp. 1603–1623, 2001.

[2] A. M. Örnek and O. Cengiz, "Capacitated lot sizing with alternative routings and overtime decisions," *International Journal of Production Research*, vol. 44, no. 24, pp. 5363–5389, 2006.

[3] C. Öztürk and A. M. Örnek, "A MIP based heuristic for capacitated MRP systems," *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 926–942, 2012.

[4] N. A. Bakke and R. Hellberg, "The challenges of capacity planning," *International Journal of Production Economics*, vol. 30–31, no. 1, pp. 243–264, 1993.

[5] S-H. Lee, S. Trimi, D. Choi, and J. S. Rha, "A comparative study of proprietary ERP and open source ERP modules on the value chain," *International Journal of Information and Decision Sciences*, vol. 3, no. 1, pp. 26–38, 2011.

[6] T. Wuttipornpun and P. Yenradee, "Finite capacity material requirement planning system for assembly flow shop with alternative work centres," *International Journal of Industrial & Systems Engineering*, vol. 18, no. 1, pp. 95–124, 2014.

[7] J. C. Chen, C-C. Wu, C-W. Chen, and K-H. Chen, "Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm," *Expert Systems with Applications*, vol. 39, pp. 10016–10021, 2012.

[8] S. Wang and M. Liu, "A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem," *Computers & Operations Research*, vol. 40, no. 4, pp. 1064–1075, 2013.

[9] H. F. Rahman, R. Sarker, and D. Essam, "A genetic algorithm for permutation flow shop scheduling under make to stock production system," *Computers & Industrial Engineering*, vol. 90, no. 1, pp. 12–24, 2015.

[10] I-H. Kuo, S-J. Horng, T-W. Kao, T-L. Lin, C-L. Lee, T. Terano, and Y. Pan, "An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model," *Expert Systems with Applications*, vol. 36, pp. 7027–7032, 2009.

[11] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Computers in Industry*, vol. 81, pp. 82–95, 2016.

[12] M. Eddaly, B. Jarboui, and P. Siarry, "Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem," *Journal of Computational Design and Engineering*, vol. 3 no. 4, pp. 295–311, 2016.

[13] B. Yagmahan and M. M. Yenisey, "A multi-objective ant colony system algorithm for flow shop scheduling problem," *Expert Systems with Applications*, vol. 37, pp. 1361–1368, 2010.

[14] F. Ahmadizar, "A new ant colony algorithm for makespan minimization in permutation flow shops," *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 355–361, 2012.

[15] Z. Zhang and Z. Jing, "An improved ant colony optimization algorithm for permutation flow shop scheduling to minimize makespan," in *Proceedings 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2012, pp. 605–609.

[16] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan," *Applied Soft Computing*, vol. 19, pp. 93–101, 2014.

[17] P. Dasgupta and S. Das, "A discrete inter-species cuckoo search for flowshop scheduling problems,"

Computers & Operations Research, vol. 60, pp. 111–120, 2015.

[18] G. M. Komaki, E. Teymourian, V. Kayvanfar, and Z. Booyavi, "Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 105, pp. 158–173, 2017.

[19] J. Grabowski and J. Pempera, "The permutation flow shop problem with blocking: A tabu search approach," *International Journal of Management Science*, vol. 35, no. 3, pp. 302–311, 2007.

[20] J-S. Chen, J. C-H. Pan, and C-K. Wu, "Hybrid tabu search for re-entrant permutation flow-shop scheduling problem," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1924–1930, 2008.

[21] L-M. Liao and C-J. Huang, "Tabu search heuristic for two-machine flowshop with batch processing machines," *Computers & Industrial Engineering*, vol. 60, no. 3, pp. 426–432, 2011.

[22] W. Bozejko, J. Pempera, and C. Smutnicki, "Parallel tabu search algorithm for the hybrid flow shop problem," *Computers & Industrial Engineering*, vol. 65, no. 3, pp. 466–474, 2013.

[23] Q-K. Pan and R. Ruiz, "Local search methods for the flowshop scheduling problem with flowtime minimization," *European Journal of Operational Research*, vol. 222, no. 1 pp. 31–43, 2012.

[24] X. Dong, P. Chen, H. Huang, and M. Nowak, "A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time," *Computers & Operations Research*, vol. 40, no. 2, pp. 627-632, 2013.

[25] Y. Wang, X. Dong, P. Chen, and Y. Lin, "Iterated local search algorithms for the sequence-dependent setup times flow shop scheduling problem minimizing makespan," *Foundation of Intelligent Systems*, vol. 277, pp. 329–338, 2014.

[26] J. Xua, C-C. Wub, Y. Yinc, and W-C. Lin, "An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times," *Applied Soft Computing*, vol. 52, pp. 39–47, 2017.

[27] W. E. Costa, M. C. Goldbarg, and E. G. Goldbarg, "Expert systems with applications," *Applied Soft Computing*, vol. 39, no. 9, pp. 8149–8161, 2012.

[28] R. M. Hallah, "Minimizing total earliness and tardiness on a permutation flow shop using VNS and MIP," *Computers & Industrial Engineering*, vol. 75, pp. 142–156, 2014.

[29] D. Lei, "Variable neighborhood search for two-agent flow shop scheduling problem," *Computers & Industrial Engineering*, vol. 80, pp. 125–131, 2015.

[30] A. P. Rifai, H-T. Nguyen, and S. Z. M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling" *Applied Soft Computing*, vol. 40, pp. 42–57, 2016.

[31] C. Low, "Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines," *Computers & Operations Research*, vol. 32, no. 8, pp. 2013–2025, 2005.

[32] B. Naderi, R. T. Moghaddam, and M. Khalili, "Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan," *Knowledge-Based Systems*, vol. 23, no. 2, pp. 77–85, 2010.

[33] P. Jarosław, S. Czeslaw, and Z. Domonik, "Optimizing bicriteria flow shop scheduling problem by simulated annealing algorithm," *Procedia Computer Science*, vol. 18, pp. 936–945, 2013.

[34] F. Nikzad, J. Rezaeian, I. Mahdavi, and I. Rastgar, "Scheduling of multi-component products in a two-stage flexible flow shop," *Applied Soft Computing*, vol. 32, pp. 132–143, 2015.

[35] F. W. Glover and G. A. Kochenberger, *Handbook of Metaheuristics*. New York: Springer, 2003.

[36] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*. 2nd ed., New York: Springer, 2010.