Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management
Kuala Lumpur, Malaysia, March 8-10, 2016

Particle Swarm Optimization for Just-In-Time Trucks Scheduling in Cross Docking Terminals

Warisa Wisittipanich
Department of Industrial Engineering
Faculty of Engineering, Chiang Mai University
Chiang Mai, Thailand
warisa.o@gmail.com

Piya Hengmeechai
Department of Industrial Engineering
Faculty of Engineering, Chiang Mai University
Chiang Mai, Thailand
piyahengmeechai@gmail.com

***Abstract*—In this paper, we present an application of Particle Swarm Optimization (PSO) for solving truck scheduling problem in a cross docking system in the content of just-in-time concept The objective is to find the schedule of inbound and outbound trucks that minimize the total earliness and the total tardiness simultaneously. The mathematical model is first presented as a mixed integer programming (MIP) model and LINGO optimization solver is then used to find the optimal solution. Due to the limitation of LINGO to obtain only one single solution related to one objective at a time, it requires additional runs to get a solution in the other objective aspect. Moreover, when the problem size becomes very large, LINGO cannot find solutions in an acceptable time. Consequently, we apply a multi-objective particle swarm optimization (MOPSO) to find a set of truck schedules with minimum total earliness and total tardiness. The performances of MOPSO are evaluated using 20 generated instances and compared with those obtained from multi-objective Differential Evolution (MODE). The experimental results demonstrate that both MOPSO and MODE are capable of finding a set of diverse and high quality non-dominated solutions with reasonable computing time.**

Keywords—cross docking; just-in-time; scheduling; metaheuristics; multi-objective

I. INTRODUCTION

The main constitutive of distribution network system is a warehouse or distribution center. As the competitive market environment has aimed to rapidly supply products to customers, the design of distribution network system become very crucial. Generally, the operations of a distribution center include five basic functions which are receiving, sorting, storing, retrieving, and shipping. One type of a distribution center that eliminates the most two expensive functions, storing and retrieving, is called cross docking. Cross docking is a logistics strategy developed under the circumstances of customers ordering small volume of products at the same time and demanding a more accurate and timely delivery. At cross docking terminals, when inbound trucks arrive, products are immediately picked out, routed, and loaded to outbound truck for on time delivery to customers. As a consequence, a little or no inventory remains at the center. Nevertheless, if the shipments need temporarily storage, it must be for a short period of time.

Nowadays, the concept Just-in-Time (JIT) becomes more and more important in several applications. Since the focus is on the transshipment of products, an efficient cross docking system requires the harmonized flow of inbound and outbound vehicles to minimize total waiting times of trucks. This on-time delivery becomes very important especially in a distribution network consisting of several cross docking centers and several customers. One approach to harmonize the flow of transshipment is to determine the arrival time and departure time of all trucks as close to their predetermined delivery times as much as possible. Therefore, an efficient truck scheduling is required to synchronize the distributions throughout the network in order to guarantee that all distributions can be made as promised.

Truck scheduling problem is one of critical decisions at the cross docking terminals. Simultaneously scheduling of inbound and outbound trucks enables harmonization of incoming flows and outbound flows of the products so that the Just-in-Time (JIT) supply of products can be attained. The truck scheduling problems in a cross docking terminal with multiple inbound and outbound doors problem are proven to be NP-hard problem which can take extremely long computing time to find the optimal solution by the traditional exact methods. Therefore, for practical purpose, heuristics and metaheuristics algorithms such as Tabu Search (TS), Simulated Annealing (SA) Algorithm, Genetic Algorithm (GA) are more preferable as they do provide high-quality or near-optimal solutions within acceptable computing times.

Yu and Egbelu [1] considered the truck scheduling problem with a single receiving door and a single shipping door. They presented the problem as a mixed integer programing (MIP) model and proposed the heuristics to minimize the makespan. The experimental results showed that solutions from the heuristics are very close to optimal solutions. Vahdani and Zandieh [2] further extended the work of [1] by using initial solutions obtained from method in [1] and applying five metaheuristics; a genetic algorithm (GA), a tabu search (TS), a simulated annealing (SA), an electromagnetism-like algorithm, and a variable neighborhood search (VSN) to improve solution quality in the evolution process. Li et al. [3] considered a multiple door cross docking system in which all doors are used either as inbound or outbound doors. The problem was formulated as a parallel machine scheduling problem and assumed that there is no difference between inbound and outbound operations. Alpan et al. [4] solved the multiple inbound and outbound doors cross docking configuration by a bounded dynamic programming. Liao et. Al [5] studied the dock assignment and sequencing of inbound trucks with an objective to minimize total weighted tardiness under the fixed outbound truck scheduling. The problem was solved by six metaheuristics: SA, TS, ant colony optimization (ACO), differential evolution (DE), and two hybrid differential evolution algorithms. The experimental results showed that ACO provided the best solutions compared to other methods. Lee et al. [6] proposed GA using simple chromosomes with dispatching rule (GA_DR) to solve the scheduling problem of inbound trucks and outbound trucks at multiple dock doors. The objective was to maximize the number of products transferred and shipped within a given working horizontal time. Van Belle et al. [7] studied truck scheduling problem at the multiple dock doors system. The objective is to minimize the total travel time and the total tardiness. They showed that the mixed integer programing can be used to solve the optimal solution but the computing time is expensive. Thus, TS approach was applied to solve the problem and results showed that TS was able to find the good solution in the short period of time.

As mentioned, on-time delivery in a distribution network becomes very crucial especially when several distribution centers and customers are involved. A truck scheduling plan is needed to be harmonized, so that the distribution throughout the network can be optimized. Nevertheless, there are very few researcher works on JIT truck scheduling problem. Li et al. [8] considered the cross dock operations with the JIT approach in which the problem was modelled as a machine scheduling problem and the GA with two local search approaches were presented. The numerical experiments showed that the first algorithms (IPGA) yielded higher quality solutions but slower than the second algorithms (SWOGA). Another study concerned with JIT scheduling in cross dock terminals is the work of [9]. Three metaheuristics; a genetic algorithm (GA), particle swarm optimization (PSO) and differential evolution (DE) were used to solve a single inbound and outbound door in a cross dock in order to find a schedule of trucks with minimization of earliness and tardiness.

Due to its successfully implementations on many application areas ([10], [11]) , this study firstly applies MOPSO to solve the problem of truck scheduling in multi-door cross docking terminals in the content of JIT philosophy. The objective is to find the schedule of trucks that minimize total earliness and total tardiness. The performances of MOPSO are particularly compared with those obtained from MODE which showed its effectiveness in solving large-scale JIT truck scheduling in the previous research [12]**.** The paper is organized as follows. Section II provides the problem description and the mathematical model used in this study. Section III describes the MOPSO algorithm and its application. Section VI shows the comparison of experimental results between MOPSO and MODE , and conclusions is given in section V.

II. MATHEMATICAL MODEL

The mathematical model used in this study is referred to the previous work of [12] in which the mixed integer programing model was first presented to find the best solution for door assigning and sequencing in multi-door cross docking problem. In this model, cross docking operations include loading, unloading and transferring operations only. Other operations such as sorting, labelling, packing, and unpacking are not considered. All inbound and outbound trucks have to be completely loaded or unloaded before it leaves the door, and the truck changeover time is fixed for all trucks. The packaging size is predetermined as the same; thus the time for load and unload one product unit is fixed as constant. The arriving products are transferred to any shipping dock by product needed of each outbound truck, and the capacity of the temporary storage is assumed to be infinite. According to the model in [12], the parameters used in the model are listed as the following:

I : number of inbound trucks (i =1,2,…, I)
O : number of outbound trucks (j =1,2,…, O)
P : number of product types (k =1,2,…, P)
R : number of doors at receiving door (m=1,2,…,R)
S : number of doors at shipping door (n =1,2,…, S)
r_{ik} : number of units of product type k that is initially loaded in inbound truck i
s_{jk} : number of units of product type k that is initially needed for outbound truck j
T_{mn} : transfer time per units of product type k from receiving door m to shipping door n
di_i : departure time of inbound truck i

do_j : departure time of outbound truck j
L : loading or unloading time per product unit
D : truck changeover time
M : a positive big number

The decision variables are as follows:

e_i : time at which inbound truck i enters the receiving door
F_i : time at which inbound truck i leaves the receiving door
d_j : time at which outbound truck j enters the shipping door
L_j : time at which outbound truck j leaves the shipping door
b_{ijk} : number of units of product type k transferred from inbound truck i to outbound truck j

$$v_{ij} = \begin{cases} 1, \text{if any products transfer from inbound truck } i \text{ to outbound truck } j \\ 0, \text{otherwise} \end{cases}$$

$$x_{im} = \begin{cases} 1, \text{if inbound trucks } i \text{ is assigned to receiving door } m \\ 0, \text{otherwise} \end{cases}$$

$$p_{ii'} = \begin{cases} 1, \text{if inbound trucks } i \text{ and } i' \text{are assigned to the same door and truck } i \text{ is a predecessor of truck } i' \\ 0, \text{otherwise} \end{cases}$$

$$y_{jn} = \begin{cases} 1, \text{if outbound trucks } j \text{ is assigned to shipping door } n \\ 0, \text{otherwise} \end{cases}$$

$$q_{jj'} = \begin{cases} 1, \text{if outbound trucks } j \text{ and } j' \text{are assigned to the same door and truck } j \text{ is a predecessor of truck } j' \\ 0, \text{otherwise} \end{cases}$$

$$z_{ijmn} = \begin{cases} 1, \text{if inbound trucks } i \text{ is assigned to receiving door } m, \text{outbound trucks } j \text{ is assigned to shipping door } n \text{ and } v_{ij} = 1 \\ 0, \text{otherwise} \end{cases}$$

The truck door assigning and sequencing problem can be formulated as a mixed integer programming model as follows.

Minimize the Total Earliness:

$$\sum_{i=1}^{I} \max(0, di_i - F_i) + \sum_{j=1}^{O} \max(0, do_j - L_j) \quad (1)$$

Minimize the Total Tardiness:

$$\sum_{i=1}^{I} \max(0, F_i - di_i) + \sum_{j=1}^{O} \max(0, L_j - do_j) \quad (2)$$

Subjected to constraints:

$$\sum_{j=1}^{O} b_{ijk} = r_{ik} \ (\forall i = 1 \dots I, \forall k = 1 \dots P) \quad (3)$$

$$\sum_{i=1}^{I} b_{ijk} = s_{jk} \ (\forall j = 1 \dots O, \forall k = 1 \dots P) \quad (4)$$

$$\sum_{k=1}^{P} b_{ijk} \leq M v_{ij} \ (\forall i = 1 \dots I, \forall j = 1 \dots O) \quad (5)$$

$$\sum_{m=1}^{R} x_{im} = 1 \ (\forall i = 1 \dots I) \quad (6)$$

$$\sum_{n=1}^{S} y_{jn} = 1 \ (\forall j = 1 \dots O) \quad (7)$$

$$\sum_{m=1}^{R} \sum_{n=1}^{S} z_{ijmn} = v_{ij} \ \ (\forall i = 1 \dots I, \forall j = 1 \dots O) \quad (8)$$

$$z_{ijmn} \leq x_{im} \ (\forall i = 1 \dots I, \forall j = 1 \dots, \forall m = 1 \dots R, \forall n = 1 \dots S) \quad (9)$$

$$z_{ijmn} \leq y_{jn} \ (\forall i = 1 \dots I, \forall j = 1 \dots O, \forall m = 1 \dots R, \forall n = 1 \dots S) \quad (10)$$

$$x_{im} + x_{jm} - 1 \leq p_{ii'} + p_{i'i} \ (\forall i, i' = 1 \dots I, i \neq i', \ \forall m = 1 \dots R) \quad (11)$$

$$p_{ii'} + p_{i'i} \leq 1 \ (\forall i, i' = 1 \dots I) \quad (12)$$

$$y_{in} + y_{jn} - 1 \leq q_{jj'} + q_{j'j} \ (\forall j, j' = 1 \dots O, j \neq j', \quad \forall n = 1 \dots S) \quad (13)$$

$$q_{jj'} + q_{j'j} \leq 1 \ (\forall j, j' = 1 \dots O) \quad (14)$$

$$e_j \geq F_i + D - M(1 - p_{ii'}) \ (\forall i, i' = 1 \dots I) \quad (15)$$

$$F_i \geq e_i + L \sum_{k=1}^{P} r_{ik} \ (\forall i = 1 \dots I) \quad (16)$$

$$d_j \geq L_i + D - M\left(1 - q_{jj'}\right) (\forall j, j' = 1 \dots O) \quad (17)$$

$$L_j \geq d_j + L \sum_{k}^{P} s_{jk} \ (\forall j = 1 \dots O) \quad (18)$$

$$L_j \geq F_i + \sum_{m=1}^{R} \sum_{n=1}^{S} T_{mn} z_{ijmn} + L \sum_{k}^{P} s_{jk} \quad - M(1 - v_{ij}) \ (\forall i = 1 \dots I, \forall j = 1 \dots O) \quad (19)$$

$$all \ variables \ \geq 0 \quad (20)$$

The objective of the model is to minimize the total earliness and the total tardiness as shown in equation (1) and (2), respectively. Equation (3) guarantees that the total number of product type k transferring from inbound truck i to all outbound trucks is equal to the number of product type k unloaded from inbound truck i. Similarly, Equation (4) ensures that the total number of product type k transferring from all inbound trucks to outbound truck j is equal to the number of product type k needed for outbound truck j. Equation (5) enforces the exact relationship between the variable b_{ijk} and v_{ij}. Equations (6) and (7) ensure that every inbound truck is assigned to a receiving door and every outbound truck is assigned to a shipping door. Equations (8)-(10) enforce the correct relationship between the variables x_{im},y_{jn},z_{ijmn}. Equations (11) and (12) enforce the correct relationship between the variable x_{im} and p_{ij}. Equations (13) and (14) enforce the correct relationship between the variable y_{jn} and q_{ij}. Equation (15) sets the entering time of each inbound truck equal to its predecessor plus truck changeover time. Equation (16) sets the leaving time of each inbound truck equal to its entering time plus the time required to unloading all products. Equation (17) states that the leaving time of each outbound truck must be greater than its predecessor leaving time plus the truck changeover time. Equation (18) guarantees that the leaving time of each outbound truck must be greater than its entering time plus the time for loading all products. Equation (19) ensures that the leaving time of each outbound truck must be greater than the leaving time of inbound truck plus the time to transfer and unload all products. Equation (20) states that all variable are non-negative variables.

III. PARTICLE SWARM OPTIMIZATION AND ITS APPLICATION TO JUST-IN-TIME TRUCK SCHEDULING PROBLEM

This study aims to find the truck schedule with minimum total earliness and total tardiness simultaneously. Due to the fact that truck scheduling problem is known as NP-hard, it requires heuristics or metaheuristics to find solution in the practical application. The tradition metaheuristic algorithms was developed to optimal a single objective, and the evolutionary processes are mainly based on the best solution in that objective. Therefore, to deal with multi-objectives perspective, the traditional algorithms need to be modified. This study firstly attempts to apply the Multi-objective Particle Swarm Optimization (MOPSO), proposed by [10], to find a set of non-dominated solution in the JIT truck scheduling in a multi-door cross docking system. MOPSO has been successfully applied in many application areas ([10], [11]). The brief description of MOPSO and its applications to the problem are given in the following sections.

A. Multi-objective Paticle Swarm Optimization

Multi-objective Particle Swarm Optimization (MOPSO) was proposed in [10] to solve multi-objective problems based on Pareto concept. The framework of MOPSO is illustrated in Fig.1. In MOPSO, the knowledge of a swarm is stored in an

external archive, called Elite group, as a set of non-dominated solutions. For each movement, each particle position is updated, and the objective functions corresponding to the position of a particle are re-evaluated. Then, the dominated sorting procedure proposed in NSGA-II [13] is applied to identify the group of new non-dominated solutions. This sorting procedure applies to the group of new solutions and current solutions in the external archive and store only non-dominated solutions into an archive for the Elite group. Then, Elite group screens its solutions to eliminate inferior solutions. In addition, the number of solutions in the Elite group is usually limited to reduce the computational time for sorting and updating procedures. When the number of non-dominated solutions exceeds the limit, the particles located in the crowded areas will be selectively removed. As a result, the Elite group still contains the best non-dominated solutions found so far in the searching process of the swarm.

Different from single-objective optimization aspect, multi-objective optimization is more complicated because of the existence of multiple global non-dominated solutions. Thus, another important decision is how to select the candidates among the Elite group to guide a particle. It is important to mention that the movement of particles is a very important to enhance the quality of non-dominated solutions on the Pareto front. In MO problems, the existence of multiple candidates in the archive provides a large number of options for the movement of particles. Several potential movement strategies were proposed in MOPSO framework to utilize the non-dominated solutions in Elite group as the movement guidance.

This study adopts the MOPSO algorithm with movement strategy 1 (ms1) and the movement strategy 3 (ms3) to solve JIT truck scheduling problems. The purpose of MOPSO-ms1 is to search for new non-dominated solutions located in the less crowded area on the Pareto front in order to increase the distribution of solutions. Therefore, the particles in a swarm are guided to the less crowded areas of non-dominated front. MOPSO-ms3 aims to search solutions in the unexplored area particularly the gaps between two non-dominated solutions on the Pareto front. Thus, MOPSO-ms3 can remedy the common problem of premature convergence at some segments on the Pareto front. For more details, please see [10].

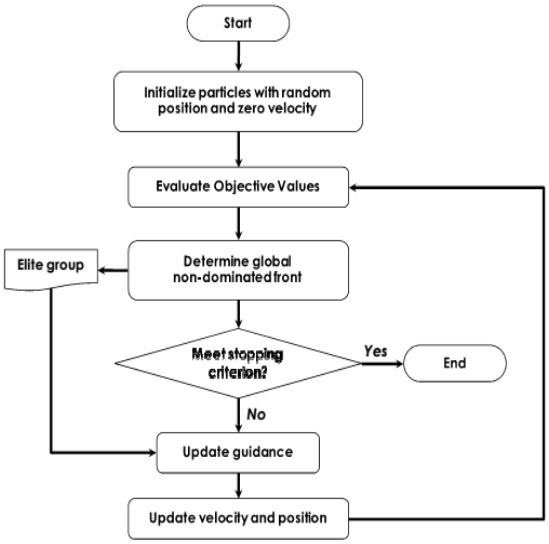

Fig. 1. MOPSO Framework [10]

B. Application of MOPSO in JIT truck scheduling problem

Since PSO is originally designed for solving problems in continuous domain, thus, in order to apply PSO to scheduling problems, a solution must be transformed. This study adopts the solution representation proposed in the previous work of [12] to schedule all trucks base on JIT concept. The procedures of solution mapping used in this study are illustrated in the example of five inbound trucks, five outbound trucks, three inbound doors, and four outbound doors. To encode a particle to a solution, each solution is represented using a particle with dimensions equal to twice as much as the summation of the number of inbound trucks (I) and outbound trucks (O). In this example, the total number of trucks is 10. Therefore, the dimensions of each particle are set to be 20. Each value in a dimension is initially generated with a uniform random number between 0 and 1. These dimensions are divided into four parts sequentially; inbound trucks, outbound trucks, inbound doors, and outbound doors. An example of a particle solution is shown in Fig. 3 (a).

After a particle is generated, a sorting list rule is applied to decode an individual particle into a sequence of trucks and the assignment of trucks to doors. As shown in Fig. 3 (b), a sequence of trucks is determined according to the order of sorted values in a vector dimension, and the door assignment is given using a sorting list rule with a repetitive-run number of dock

doors. Therefore, each truck is assigned to its door correspondingly. The advantage of this decoding scheme is that it always provides a feasible solution.

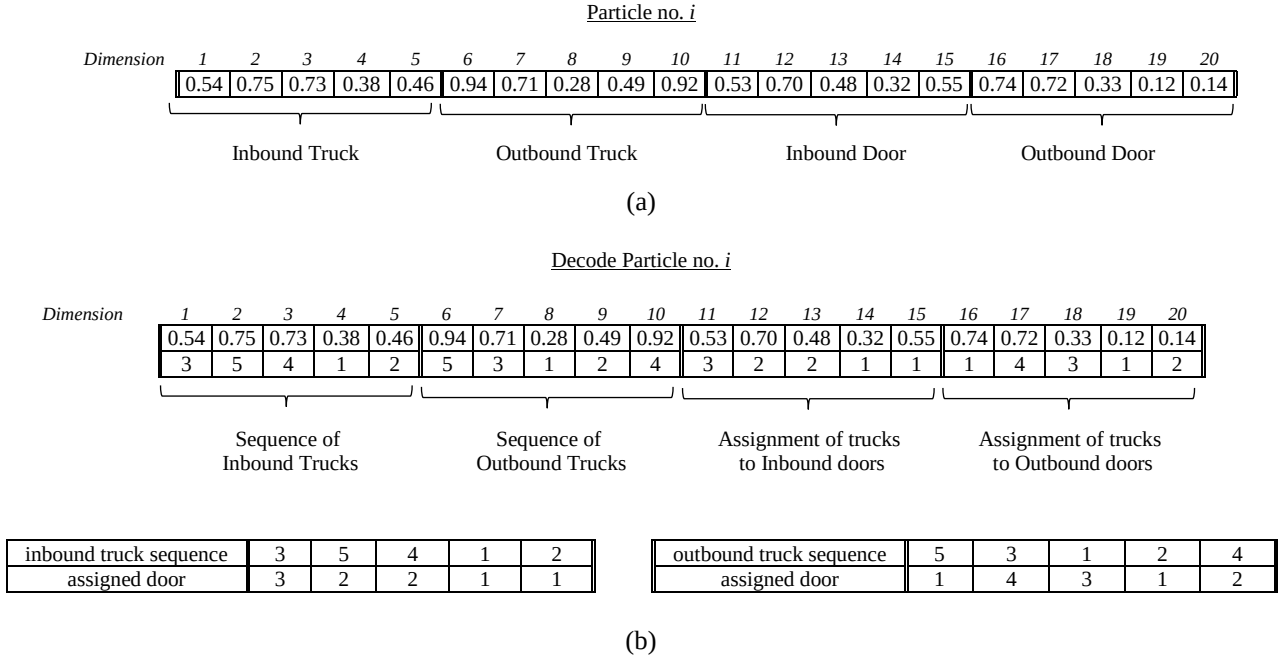

Fig. 3. Example of solution representation of cross dock terminals with I = O = 5 and R = 3 S = 4

According to [12], the decoding rules are divided into two stages. The purpose of the first stage is to find the schedule of inbound trucks, and the purpose of the second stage is to determine the allocation of products to each outbound truck and schedule the outbound trucks. In the first stage, two decoding strategy, named as ITSH (Inbound_Truck_Shift) and ITDD (Inbound_Truck_DueDate), are adopted. The concept of ITSH is to determine the arrival time of each inbound truck randomly by assigning the arrival time of a truck between its ready time and its maximum value of the possible arrival time. Thus, it is expected that a truck arrives and departs not too early or too late. On the other hand, ITDD algorithm aims to determine the departure time of inbound trucks as close to their due dates as possible. Thus the total tardiness of inbound trucks could be minimized. In ITDD, the time at the due date of an inbound truck is first checked whether it is available. If the time at the due date is available, the arrival time and departure time of a truck are then determined. However, if the time at the due date is occupied, the truck is scheduled at the best position next to the previous scheduled truck.

In the second stage, the focus is on product allocation and a schedule of outbound trucks. Two decoding strategies, named OTSH (Outbound_Truck_Shift) and OTDD (Outbound_Truck_DueDate) are used. In both OTSH and OTDD strategies, the ready time of each product to be unloaded from incoming trucks is first determined, and the product is then allocated and loaded, according to the FCFS rule, to an outbound truck. The arrival time of an outbound truck occurs only when the last product required for that truck is ready to be loaded. Similar to ITSH, OTSH determines the arrival time of each outbound truck, derived from a sequence, randomly between the time at which the last required product is ready and its maximum value of the possible arrival time. So, it provides possibilities for a truck not to arrive beforehand or behind its due date. The purpose of OTDD is to schedule the departure time of outbound trucks to their due dates as close as possible, so that the total tardiness of outbound trucks could be minimized. When the time of the last product required for an outbound truck in a sequence is determined, OTDD firstly observes if the time at due date of the truck is occupied. If the time at the due date is available, the arrival time and departure time of an outbound truck are then determined. However, if the time at the due date is occupied, the outbound truck is scheduled at the best position next to the previous scheduled truck. For more details of these decoding strategies, please see [12].

Due to the fact that each decoding strategy has its own advantages, a combination of these pairs can be used to provide diverse and good solution quality. There are four possible pairs of decoding strategies which are ITDD-OTDD, ITSH-OTSH, ITDD-OTSH, and ITSH-OTDD. While ITDD-OTDD and ITDD-OTSH have the potential to search for solutions with minimized total earliness (the left side of a Pareto front); ITSH-OTSH and ITSH-OTDD tends to perform well in searching solutions with the compromised schedules and minimized total tardiness (the center and the right side of a Pareto front).

IV. COMPUTATIONAL RESULTS

A. Experimental Results

Based on some preliminary experiments, the setting of MOPSO parameters that are suitable for the problem in this study are summarized in TABLE I. The performances of MOPSO-ms1 and MOPSO-ms3 are evaluated on 20 generated instances and the results are compared to those obtained from MODE-ms1 which showed outstanding results among other MODE strategies in the previous research [12]. For a fair comparison, the population size and number of iterations of MOPSO are set as 200 and 500 respectively equal to those in MODE. The inertia weight is linearly increased from 0.5 to 0.9 to extensively explore the search space at the beginning of the search. After the good solutions are found, the focus is then shifted to intensively search at particular areas. The acceleration constant of the personal best (*Cp*), the global best (*Cg*), the local best (*Cl*) and the near-neighbor best (*Cn*) is set as 0.1, 0.1, 0.7, and 0.1 respectively. Due to the distinct advantage of each pair of decoding strategies, a combination of these pairs (ITDD-OTDD: ITSH-OTSH: ITDD-OTSH: ITSH-OTDD) in the experiment is set as same as that in MODE which is 15:15:40:30. The comparison results between MOPSO and MODE are shown in TABLE II. It is noted that DE stands for MODE-ms1, and Pms1 and Pms3 stand for MOPSO-ms1 and MOPSO-ms3 respectively.

TABLE I. PARAMETER SETTING OF MOPSO

Parameter	***Value***
Number of iteration	500
Number of particle	200
Inertia weight	Linearly increase from 0.5 to 0.9
Cp, Cg, Cl, Cn	0.1, 0.1, 0.7, 0.1
Combination of decoding strategies	15:15:40:15

As shown in TABLE II, MODE-ms1, MOPSO-ms1, and MOPSO-ms3 shows good performances in generating a set of diverse non-dominated solutions. In most small-size and medium-size problems, all algorithms are able to find schedules with minimum total earliness and total tardiness as equal to those obtained from LINGO optimization solver. For small and medium problem sizes, MODE-ms1 clearly shows its superiority to MOPSO-ms1 and MOPSO-ms3 by generating solutions with lower value of total earliness and total tardiness. However, when the problem becomes very large, MOPSO-ms1 and MOPSO-ms3 are able to generate several solutions that dominate those obtained by MODE-ms1. In addition, both MOPSO-ms1 and MOPSO-ms3 are able to find several solutions with relatively small total earliness and total tardiness compared to MODE-ms1.

B. Performance Measurement

The performance evaluation of multi-objective optimization problem is different from the single objective optimization problem since, in MO problems, a set of solutions make it difficult for comparison. This study uses *C* metric to compare the set of non-dominated solutions obtained from the MODE-ms1, MOPSO-ms1, and MOPSO-ms3. *C* (*A*,*B*) measures the fractions of members of *B* that are dominated by members of *A* as shown in equation (21).

$$C\ (A,B) = \frac{|\{b \in B: \exists a \in A, a \succ b\}|}{|B|} \tag{21}$$

Where $|B|$ is the number of solutions in *B*. Therefore *C* (*A*,*B*) = 1 means that each solutions in B is dominated by some solutions in A. On the other hand, *C* (*A*,*B*) = 0 states that all solutions in B are non-dominated by any solution in *A*. The lower the ratio *C* (*A*,*B*) is, the better the solution set is. TABLE III shows the comparison results between MODE-ms1, MOPSO-ms1, and MOPSO-ms3.

TABLE II. COMPARISON OF NUMERICAL RESULTS BETWEEN MOPSO AND MODE ALGORITHM

Instance	Problem Size					LINGO		MODE-ms1				MOPSO-ms1				MOPSO-ms3			
	I	*O*	*P*	*R*	*S*	*Total Earliness*	*Total Tardiness*												
1	3	4	5	1	2	0	1.5	(0, 25.3)	(14, 12.3)	(20, 7.3)	(27, 2.5)	(0, 25.3)	(15, 11.3)	(20, 7.3)	(27, 2.5)	(0, 25.3)	(14, 12.3)	(18, 8.3)	(26, 3.3)
								(7, 19.3)	(16, 10.3)	(23, 5.3)	(28, 2.3)	(9, 17.3)	(16, 10.3)	(24, 4.3)	(28, 2.3)	(6, 20.3)	(15, 11.3)	(20, 7.3)	(27, 2.5)
								(9, 17.3)	(17, 9.3)	(24, 4.3)	(29, 1.5)	(10, 16.3)	(17, 9.3)	(26, 3.3)	(29, 1.5)	(9, 18.3)	(16, 10.3)	(23, 5.3)	(28, 2.3)
								(10, 16.3)	(18, 8.3)	(26, 3.3)		(14, 12.3)	(18, 8.3)			(10, 16.3)	(17, 9.3)	(24, 4.3)	(29, 1.5)
2	4	4	5	1	1	0	17	(0, 110)	(31, 58)	(42, 45)	(58, 26.5)	(0, 110)	(36, 52)	(46, 35.5)	(60, 24.5)	(0, 110)	(28, 62)	(42, 45)	(58, 26.5)
								(19, 81)	(35, 53)	(44, 36.5)	(60, 24.5)	(17, 78)	(37, 50)	(48, 33.5)	(98, 22)	(17, 86)	(31, 58)	(44, 36.5)	(60, 24.5)
								(20, 74)	(36, 52)	(48, 33.5)	(95, 23)	(24, 69)	(39, 49)	(51, 32.5)	(101, 21)	(18, 84)	(36, 52)	(48, 33.5)	(95, 23)
								(25, 66)	(37, 50)	(50, 32.5)	(98, 22)	(28, 62)	(40, 46)	(55, 29.5)	(103, 19)	(20, 74)	(37, 50)	(50, 32.5)	(98, 22)
								(28, 62)	(38, 49)	(54, 29.5)	(101, 21)	(30, 60)	(42, 45)	(56, 27.5)		(25, 66)	(39, 49)	(54, 29.5)	(100, 20)
								(29, 61)	(40, 46)	(56, 27.5)	(103, 19)	(31, 58)	(44, 36.5)	(58, 26.5)		(26, 65)	(40, 46)	(56, 27.5)	(103, 19)
3	4	4	3	1	2	0	8	(0, 90)	(31, 53)	(45, 31)	(62, 18)	(0, 90)	(38, 49)	(46, 36)	(67, 15)	(0, 90)	(27, 63)	(46, 32)	(62, 18)
								(3, 87)	(32, 50)	(52, 25)	(67, 15)	(3, 80)	(40, 46)	(47, 35)	(70, 14)	(3, 80)	(32, 50)	(47, 29)	(63, 17)
								(4, 79)	(35, 45)	(54, 24)	(72, 12)	(24, 68)	(41, 43)	(50, 28)	(71, 13)	(5, 78)	(40, 42)	(48, 26)	(69, 13)
								(8, 76)	(37, 40)	(56, 22)	(74, 10)	(34, 54)	(42, 39)	(51, 25)	(78, 10)	(7, 76)	(42, 38)	(55, 21)	(78, 10)
								(27, 63)	(40, 37)	(61, 19)	(108, 9)	(35, 53)	(43, 37)	(64, 18)	(111, 9)	(8, 75)	(45, 37)	(61, 19)	(129, 9)
4	4	5	5	1	1	0	1	(0, 55)	(31, 9)	(42, 4)	(64, 2)	(0, 55)	(35, 8)	(46, 4)	(63, 2)	(0, 55)	(2, 20)	(42, 4)	(74, 1)
								(2, 20)	(34, 6)	(56, 3)	(73, 1)	(2, 20)	(42, 6)	(52, 3)	(72, 1)		(30, 8)	(58, 2)	
5	4	6	5	2	1	0	0	(0, 60)	(5, 7)	(29, 4)	(34, 1)	(0, 60)	(5, 14)	(8, 7)	(48, 1)	(0, 60)	(5, 10)	(8, 5)	(33, 1)
								(3, 11)	(8, 5)	(33, 2)	(36, 0)	(2, 19)	(6, 12)	(9, 5)	(51, 0)	(3, 17)	(6, 8)	(30, 3)	(40, 0)
								(4, 9)				(3, 15)	(7, 8)	(38, 2)		(4, 12)			
6	5	4	3	2	1	0	0	(0, 8)	(27, 3)	(36, 0)		(0, 8)	(28, 5)	(32, 3)	(39, 1)	(0, 9)	(25, 6)	(31, 3)	(40, 0)
								(0.5, 7.5)	(33, 1)			(2.5, 7.5)	(29, 4)	(35, 2)	(42, 0)	(0.5, 7.5)	(28, 4)	(33, 2)	
7	5	5	6	2	2	0	1.5	(0, 50.5)	(34, 23.5)	(44, 13)	(61, 5.5)	(0, 50.5)	(35, 27)	(46, 13.5)	(54, 9.5)	(0, 50.5)	(37, 17.5)	(52, 8.5)	(67, 4.5)
								(25, 33)	(36, 19.5)	(48, 12.5)	(69, 4.5)	(21, 46)	(37, 19.5)	(48, 13)	(61, 5.5)	(23, 48)	(38.5, 13.5)	(59, 7.5)	(78, 3.5)
								(29, 26.5)	(38, 14.5)	(52, 10.5)	(77, 4)	(23, 41)	(40, 17.5)	(52, 12.5)	(84, 5)	(24, 33.5)	(44, 12.5)	(62, 6.5)	(90, 3)
								(32, 25)	(43, 14)	(54, 6.5)	(84, 2.5)	(30, 29.5)	(42, 14.5)	(53, 11)	(89, 2.5)	(26.5, 31.5)	(46, 11.5)	(66, 5.5)	(94, 2.5)
								(32.5, 24.5)				(34, 27.5)	(45, 14)			(27, 24)	(51, 9.5)		
8	5	5	6	3	2	0	0	(0, 27)	(14, 9)	(18, 5)	(23, 1)	(0, 27)	(15, 15)	(19, 8)	(29, 1)	(0, 27)	(13, 16)	(18, 5)	(25, 2)
								(10, 20)	(15, 6)	(20, 3)	(55, 0)	(13, 26)	(16, 13)	(21, 6)	(48, 0)	(10, 26)	(14, 11)	(20, 4)	(26, 1)
								(12, 13)				(14, 22)	(18, 11)	(24, 3)		(11, 21)	(15, 7)	(21, 3)	(55, 0)
9	5	6	4	2	2	0	0	(0, 49)	(27, 26)	(36, 13)	(55, 6)	(0, 49)	(33, 30)	(47, 14)	(60, 5)	(0, 49)	(30, 22)	(44,13)	(61, 4)
								(1, 47)	(30, 23)	(42, 10)	(56, 2)	(1, 47)	(34, 27)	(52, 13)	(67, 4)	(1, 47)	(31, 21)	(45, 11)	(68, 2)
								(10, 38)	(31, 18)	(46, 8)	(60, 1)	(31, 32)	(40, 18)	(54, 10)	(71, 2)	(27, 35)	(37, 20)	(51, 8)	(73, 1)
								(25, 29)	(32, 17)	(51, 7)	(82, 0)		(44, 15)	(55, 9)	(77, 0)	(29, 24)	(39, 14)	(52, 7)	(75, 0)
10	6	7	5	2	3	0	9	(0, 117)	(54, 52)	(95, 28)	(123, 16)	(0, 117)	(65, 42)	(102, 23)	(133, 15)	(0, 117)	(55, 44)	(96, 27)	(133, 14)
								(30, 84)	(60, 46)	(99, 24)	(131, 14)	(31, 112)	(79, 36)	(107, 21)	(134, 14)	(30, 112)	(69, 42)	(103, 24)	(139, 13)

Instance	Problem Size					LINGO		MODE-ms1				MOPSO-ms1				MOPSO-ms3			
	I	***O***	***P***	***R***	***S***	***Total Earliness***	***Total Tardiness***												
								(33, 79)	(65, 40)	(103, 23)	(140, 12)	(32, 66)	(88, 32)	(110, 18)	(146, 12)	(35, 111)	(71, 36)	(109, 22)	(144, 12)
								(40, 70)	(75, 36)	(104, 22)	(267, 11)	(56, 60)	(96, 28)	(122, 16)	(273, 11)	(43, 103)	(80, 34)	(114, 18)	(255, 11)
								(42, 65)	(77, 33)	(108, 20)	(287, 10)	(60, 44)				(46, 65)	(87, 32)	(117, 17)	(256, 10)
								(50, 64)	(87, 32)	(110, 18)	(292, 9)					(49, 56)	(94, 31)	(125, 16)	(278, 9)
								(53, 56)	(94, 29)	(117, 17)							(95, 29)		
11	7	6	4	4	2	0	0	(0, 69)	(36, 43)	(69, 31)	(94, 5)	(0, 69)	(40, 43)	(89, 13)	(101, 4)	(0, 69)	(36, 43)	(82, 31)	(95, 8)
								(15, 62)	(55, 41)	(87, 17)	(101, 4)	(15, 62)	(86, 16)	(98, 7)	(103, 2)	(15, 63)	(59, 35)	(90, 14)	(98, 3)
								(18, 59)	(61, 32)	(92, 11)	(106, 0)	(18, 59)	(94, 10)	(99, 6)	(110, 0)	(18, 60)	(65, 32)	(94, 13)	(101, 0)
12	8	8	6	3	3	N/A	N/A	(0, 142)	(58, 63)	(91, 25)	(130, 8)	(0, 142)	(63, 111)	(99, 34)	(141, 8)	(0, 142)	(67, 70)	((95, 26)	(137, 7)
								(7, 137)	(63, 55)	(97, 23)	(136, 4)	(7, 137)	(73, 75)	(102, 24)	(143, 5)	(8, 137)	(71, 50)	(96, 22)	(142, 6)
								(8, 129)	(71, 44)	(109, 21)	(142, 3)	(11, 129)	(79, 61)	(110, 23)	(146, 4)	(11, 129)	(72, 45)	(114, 20)	(145, 5)
								(10, 128)	(84, 40)	(111, 16)	(146, 2)	(12, 128)	(83, 60)	(114, 18)	(150, 3)	(16, 127)	(78, 41)	(115, 19)	(146, 3)
								(55, 114)	(86, 33)	(122, 12)	(149, 1)	(19, 127)	(84, 54)	(120, 15)	(156, 1)	(43, 98)	(87, 39)	(117, 9)	(151, 1)
								(56, 108)	(90, 28)	(128, 11)	(206, 0)	(62, 112)	(86, 39)	(128, 10)	(181, 0)	(45, 90)	91, 35)	(131, 8)	(188, 0)
13	9	10	10	2	4	N/A	N/A	(0, 157)	(62, 123)	(126, 53)	(163, 17)	(0, 156)	(39, 136)	(100, 75)	(148, 29)	(0, 149)	(58, 127)	(131, 44)	(168, 25)
								(4, 154)	(66, 121)	(128, 44)	(179, 16)	(3, 155)	(44, 131)	(110, 70)	(155, 28)	(28, 144)	(61, 122)	(132, 43)	(170, 23)
								(6, 152)	(80, 117)	(130, 42)	(192, 12)	(11, 153)	(49, 128)	(111, 68)	(166, 26)	(31, 142)	(68, 121)	(137, 41)	(172, 21)
								(26, 149)	(90, 85)	(137, 41)	(196, 10)	(26, 152)	(58, 126)	(113, 61)	(167, 17)	(38, 140)	(87, 85)	(148, 37)	(187, 18)
								(27, 144)	(103, 77)	(145, 40)	(255, 5)	(32, 151)	(60, 122)	(118, 56)	(248, 11)	(39, 137)	(93, 82)	(149, 35)	(188, 14)
								(31, 133)	(108, 65)	(146, 29)	(263, 4)	(33, 146)	(85, 118)	(127, 53)	(263, 9)	(41, 134)	(98, 81)	(151, 34)	(237, 10)
								(41, 132)	(119, 60)	(156, 26)	(278, 2)	(36, 145)	(86, 106)	(131, 52)	(270, 3)	(43, 133)	(101, 71)	(154, 32)	(267, 5)
								(44, 131)	(121, 59)	(159, 25)	(285, 1)	(37, 142)	(88, 99)	(132, 50)	(297, 0)	(47, 132)	(115, 49)	(159, 29)	(287, 3)
								(50, 126)	(125, 57)	(161, 18)		(38, 141)	(93, 87)	(138, 37)		(50, 129)	(122, 47)	(163, 26)	(308, 2)
14	10	11	6	3	4	N/A	N/A	(0, 170)	(81, 94)	(110, 57)	(139, 27)	(0, 170)	(98, 81)	(122, 48)	(171, 24)	(0, 173)	(69, 115)	(105, 58)	(150, 27)
								(1, 168)	(85, 89)	(111, 52)	(151, 26)	(11, 161)	(99, 70)	(126, 38)	(180, 21)	(1, 171)	(70, 111)	(108, 55)	(151, 26)
								(9, 165)	(86, 85)	(115, 48)	(158, 24)	(14, 158)	(104, 61)	(141, 37)	(187, 19)	(8, 169)	(77, 104)	(113, 53)	(159, 21)
								(11, 160)	(88, 73)	(118, 45)	(161, 21)	(63, 128)	(108, 60)	(149, 35)	(192, 18)	(9, 163)	(86, 89)	(118, 36)	(172, 20)
								(12, 159)	(95, 70)	(119, 41)	(168, 18)	(65, 119)	(109, 57)	(150, 33)	(202, 15)	(11, 159)	(88, 85)	(129, 34)	(177, 19)
								(15, 157)	(101, 66)	(131, 39)	(170, 16)	(83, 98)	(113, 55)	(151, 32)	(218, 14)	(12, 158)	(89, 79)	(133, 33)	(181, 18)
								(19, 156)	(102, 62)	(133, 38)	(172, 15)	(88, 91)	(118, 51)	(156, 31)	(305, 12)	(13, 156)	(90, 72)	(140, 32)	(198, 16)
								(59, 113)	(103, 58)	(136, 36)	(186, 13)	(89, 83)	(119, 50)	(158, 27)		(17, 155)	(97, 63)	(143, 30)	(220, 14)
								(63, 96)								(62, 124)	(103, 61)	(146, 28)	(230, 12)
15	12	13	8	3	3	N/A	N/A	(0, 371)	(238, 138)	(332, 62)	(389, 20)	(0, 378)	(214, 255)	(301, 70)	(395, 34)	(0, 375)	(229, 224)	(278, 139)	(328, 87)
								(26, 365)	(250, 136)	(336, 56)	(406, 18)	(24, 377)	(219, 207)	(339, 56)	(399, 25)	(45, 373)	(233, 205)	(300, 132)	(330, 82)
								(53, 285)	(259, 132)	(337, 55)	(410, 17)	(31, 368)	(227, 206)	(349, 42)	(414, 24)	(53, 363)	(251, 195)	(303, 130)	(335, 65)
								(199, 259)	(261, 109)	(349, 37)	(422, 11)	(87, 357)	(229, 184)	(375, 38)	(434, 14)	(76, 303)	(252, 177)	(305, 109)	(342, 21)
								(219, 242)	(293, 103)	(368, 35)	(428, 6)	(99, 356)	(265, 139)	(376, 37)	(445, 9)	(213, 270)	(255, 156)	(319, 91)	(557, 17)
								(222, 179)	(297, 80)	(374, 34)	(685, 5)	(130, 334)	(278, 117)	(389, 35)	(461, 8)				

Instance	Problem Size					LINGO		MODE-ms1				MOPSO-ms1				MOPSO-ms3			
	I	***O***	***P***	***R***	***S***	***Total Earliness***	***Total Tardiness***												
								(223, 157)	(327, 72)	(386, 28)		(212, 311)							
16	15	10	10	6	4	N/A	N/A	(0, 415)	(134, 291)	(471.5, 83.5)	(608, 19)	(0, 385)	(463, 221)	(496, 49.5)	(548.5, 17)	(0, 360)	(30, 306)	(490, 132.5)	(508.5, 53.5)
								(26, 404.5)	(478.5, 83)	(514.5, 45.5)	(612, 17)	(6, 367)	(473, 147.5)	(507.5, 46)	(559, 16.5)	(5, 355)	(475, 293)	(491, 107.5)	(511.5, 38.5)
								(26.5, 395.5)	(479, 81)	(519, 27.5)	(549, 23)	(10, 352)	(474, 139)	(515.5, 35)	(580, 15)	(6, 354)	(476, 274.5)	(492.5, 99.5)	(512, 35)
								(32, 362.5)	(485, 73.5)	(553.5, 22)	(649, 14)	(59, 345)	(481, 138.5)	(518, 29)	(611.5, 14)	(7, 352)	(478, 220.5)	(493.5, 96)	(534.5, 22.5)
								(35, 305)	(490.5, 64)	(585.5, 21)	(678.5, 10.5)	(61, 326)	(482.5, 107)	(567, 15.5)	(622.5, 12)	(8, 344)	(484, 166)	(496, 94)	(551, 18.5)
								(113, 295)				(117, 305)	(483, 88)	(523, 22)	(638, 10.5)	(14, 334)	(487, 155)	(497, 77.5)	(582.5, 14.5)
												(137, 298)	(488.5, 62)	(544.5, 18)	(718.5, 10)	(24, 307)	(488, 151.5)	(504, 67)	
17	20	25	15	4	5	N/A	N/A	(0, 1821)	(34, 1518)	(119, 1213)	(733, 775)	(0, 1513)	(85, 1306)	(696, 714)	(1008, 457)	(0, 1575)	(633, 789)	(720, 613)	(929, 413)
								(9, 1765)	(54, 1426)	(153, 1185)	(848, 719)	(6, 1472)	(107, 1261)	(750, 671)	(1123, 450)	(5, 1262)	(640, 782)	(728, 604)	(1107, 409)
								(16, 1707)	(75, 1320)	(672, 977)	(879, 564)	(33, 1418)	(553, 1074)	(751, 658)	(1146, 439)	(516, 974)	(649, 751)	(730, 583)	(1141, 403)
								(19, 1598)	(101, 1305)	(688, 965)	(1127, 515)	(37, 1401)	(602, 898)	(775, 595)	(1153, 434)	(538, 913)	(681, 677)	(738, 570)	(1149, 386)
								(21, 1556)	(114, 1265)	(716, 926)	(1274, 483)	(42, 1360)	(651, 880)	(780, 573)	(1159, 380)	(561, 898)	(698, 638)	(758, 560)	(1195, 335)
								(26, 1552)	(116, 1240)	(731, 827)	(1372, 443)	(76, 1319)	(671, 737)	(866, 514)	(1231, 355)	(630, 871)	(716, 630)	(846, 433)	(1270, 325)
18	25	25	10	7	7	N/A	N/A	(0, 700)	(54, 506)	(141, 383)	(531, 208)	(0, 1017)	(40, 696)	(113, 547)	(436, 306)	(0, 598)	(53, 478)	(359, 303)	(452, 177)
								(1, 692)	(57, 497)	(152, 381)	(536, 205)	(6, 1014)	(43, 690)	(138, 537)	(478, 304)	(5, 572)	(60, 455)	(374, 278)	(455, 176)
								(2, 663)	(61, 483)	(392, 332)	(544, 197)	(8, 923)	(47, 656)	(141, 527)	(481, 260)	(8, 553)	(62, 417)	(396, 268)	(501, 171)
								(10, 639)	(66, 449)	(424, 321)	(572, 196)	(11, 884)	(54, 652)	(365, 523)	(510, 246)	(17, 527)	(149, 404)	(402, 267)	(503, 108)
								(19, 599)	(83, 440)	(430, 319)	(620, 156)	(16, 811)	(64, 644)	(369, 495)	(535, 241)	(19, 508)	(343, 403)	(410, 232)	(605, 117)
								(22, 592)	(100, 423)	(478, 237)	(662, 146)	(20, 797)	(71, 584)	(378, 450)	(557, 225)	(34, 503)	(351, 365)	(430, 198)	(648, 65)
								(45, 531)	(130, 398)	(500, 221)	(681, 137)	(30, 721)	(103, 567)	(391, 403)	(592, 194)				
								(48, 528)	(136, 388)	(525, 209)	(701, 115)	(36, 720)	(104, 558)	(400, 360)	(653, 175)				
19	32	30	10	6	4	N/A	N/A	(342, 9029)	(613, 3263)	(1883, 2177)	(3840, 987)	(144, 6011)	(1401, 2075)	(2877, 1140)	(4552, 680)	(14, 4461)	(247, 1763)	(2725, 1133)	(4160, 520)
								(356, 8838)	(792, 2921)	(2619, 1765)	(4473, 818)	(150, 4226)	(1517, 1866)	(2962, 1037)	(4566, 643)	(24, 3981)	(256, 1757)	(2834, 810)	(4446, 450)
								(385, 4685)	(875, 2912)	(2637, 1679)	(4790, 796)	(212, 3189)	(2698, 1667)	(3223, 1028)	(4574, 423)	(26, 2975)	(822, 1690)	(3420, 750)	(4924, 403)
								(466, 4511)	(1001, 2698)	(2706, 1555)	(5027, 653)	(383, 2665)	(2731, 1516)	(3776, 1020)	(5579, 393)	(63, 2907)	(839, 1589)	(3835, 746)	(5288, 259)
								(492, 4197)	(1185, 2697)	(2857, 1523)	(5182, 619)	(546, 2525)	(2788, 1283)	(3802, 820)	(5656, 376)	(105, 2391)	(1022, 1376)	(4105, 703)	
								(510, 3745)	(1309, 2673)	(3038, 1341)	(5445, 551)	(730, 2486)	(2874, 1262)	(3803, 686)	(3778, 910)	(172, 1943)	(1568, 1240)		
								(545, 3552)	(1343, 2498)	(3052, 1259)	(5939, 503)	(851, 2158)							
20	40	35	15	4	6	N/A	N/A	(342, 9029)	(613, 3263)	(1883, 2177)	(3840, 987)	(151, 4155)	(384, 3223)	(3617, 1012)	(5278, 537)	(43, 5673)	(246, 3083)	(3754, 844)	(4932, 453)
								(356, 8838)	(636, 3050)	(2619, 1765)	(4174, 843)	(220, 4002)	(385, 3193)	(3679, 990)	(5283, 513)	(78, 5113)	(270, 3014)	(3782, 769)	(5009, 451)
								(385, 4685)	(792, 2921)	(2637, 1679)	(4790, 796)	(251, 3984)	(426, 3098)	(3695, 905)	(5321, 396)	(84, 4959)	(304, 2933)	(4121, 715)	(5173, 449)
								(466, 4511)	(875, 2912)	(2706, 1555)	(5027, 653)	(258, 3901)	(475, 2930)	(3817, 744)	(5803, 368)	(88, 3976)	(349, 2814)	(4133, 695)	(5300, 372)
								(489, 4373)	(1001, 2698)	(2857, 1523)	(5182, 619)	(267, 3731)	(630, 2822)	(4046, 732)	(5881, 356)	(100, 3512)	(726, 2607)	(4172, 682)	(5658, 336)
								(492, 4197)	(1185, 2697)	(2892, 1366)	(5445, 551)	(319, 3610)	(736, 2756)	(4166, 669)	(5986, 345)	(120, 3449)	(3359, 2356)	(4435, 641)	(5719, 329)
								(510, 3745)	(1309, 2673)	(3038, 1341)	(5939, 503)	(330, 3510)	(760, 2736)	(4814, 620)	(6060, 293)	(175, 3412)	(3382, 1263)	(4563, 575)	(5879, 319)
								(545, 3552)	(1343, 2498)	(3052, 1259)		(342, 3503)	(3499, 1136)	(4856, 587)	(6110, 287)	(204, 3270)	(3521, 1194)	(4716, 564)	(6002, 232)
								(555, 3487)	(1401, 2451)	(3214, 1066)		(361, 3333)	(3614, 1075)	(4979, 546)	(6227, 244)	(222, 3233)	(3697, 866)	(4766, 523)	(6071, 213)

TABLE III. THE C METRIC COMPARISON BETWEEN MODE AND MOPSO

Instance	*C (DE, Pms1)*	*C (Pms1, DE)*	*C (DE, Pms3)*	*C (Pms3, DE)*	*C(Pms1, Pms3)*	*C(Pms3, Pms1)*
1	0.000	0.000	0.056	0.000	0.056	0.000
2	0.160	0.037	0.037	0.037	0.074	0.160
3	0.480	0.087	0.280	0.130	0.120	0.440
4	0.333	0.333	0.143	0.222	0.143	0.333
5	0.818	0.000	0.556	0.222	0.111	0.727
6	0.875	0.000	0.714	0.000	0.143	0.875
7	0.778	0.000	0.278	0.353	0.278	0.722
8	0.818	0.111	0.750	0.000	0.083	0.818
9	0.722	0.059	0.636	0.118	0.000	0.722
10	0.529	0.185	0.560	0.222	0.280	0.471
11	0.385	0.083	0.500	0.167	0.333	0.538
12	0.815	0.080	0.741	0.240	0.333	0.593
13	0.629	0.200	0.750	0.314	0.444	0.486
14	0.839	0.091	0.558	0.455	0.047	0.935
15	0.920	0.074	0.950	0.148	0.600	0.520
16	0.321	0.571	0.483	0.476	0.621	0.214
17	0.000	0.800	0.000	0.880	0.034	0.875
18	1.000	0.000	0.000	0.898	0.000	1.000
19	0.000	1.000	0.000	1.000	0.095	0.923
20	0.000	1.000	0.000	1.000	0.047	0.865

According to results from TABLE III, it is confirmed that MODE-ms1 shows its outstanding results over MOPSO-ms1 and MOPSO-ms3 especially for the small and medium problem size. The MODE-ms1 mostly obtains the *C* metric values (*C* (*Pms1, DE*) and *C* (*Pms3, DE*)) smaller than *C* metric values of MOPSO-ms1 and MOPSO-ms3 (*C* (*DE, Pms1*) and *C (DE, Pms3*)). This shows that most solutions generated by MOPSO-ms1 and MOPSO-ms3 are dominated by those obtained by MODE-ms1. However, in the large size problem, MOPSO-ms1 and MOPSO-ms3 demonstrate their superiority to MODE-ms1 since MODE-ms1 obtain *C* metric value of 0 or close to 0 whereas MOPSO-ms1 and MOPSO-ms3 generally obtain the *C* metric values of 1 or close to 1. This demonstrates that the majority of solutions generated from MODE-ms1 are completely dominated by those from MOPSO-ms1 and MOPSO-ms3, and there are no or only few solutions, found by MOPSO-ms1 and MOPSO-ms3, are dominated by those from MODE-ms1. In addition, it is clearly seen that MOPSO-ms3 generally outperforms MOPSO-ms1 in most instances.

V. CONCLUSION

This study presents the first implementation of MOPSO algorithm for solving JIT truck scheduling problem in multi-door cross docking terminal with an objective to simultaneously minimize total earliness and total tardiness of trucks. The performances of MOPSO are evaluated on a set of generated problems. The experimental results from MOPSO are particular compared those obtained from MODE-ms1 in the previous research work. Under the same number of function evaluations and solution representation, the numerical results show that both MODE and MOPSO algorithms are competitive approaches and capable of finding a set of diverse and high quality non-dominated solutions on Pareto front. All algorithms are able to find schedules with minimum total earliness and total tardiness as equal to those obtained from LINGO optimization solver. Generally, in most small to medium size problems, MODE-ms1 clearly shows its superiority to MOPSO-ms1 and MOPSO-ms3 by generating solutions with lower value of total earliness and total tardiness. However, when the problem becomes very

large, MOPSO-ms1 and MOPSO-ms3 are able to generate several dominated with relatively small total earliness and total tardiness compared to MODE-ms1.

REFERENCES

[1] W. Yu and P. J. Egbelu, "Scheduling of inbound and outbound trucks in cross docking systems with temporary storage," Eur. J. Oper. Res., 2008, vol. 184, pp. 377-396.

[2] B.Vahdani and M. Zandieh, "Scheduling trucks in cross-docking systems: robust meta-heuristics," Comput. Ind. Eng., 2010, vol. 58, pp. 12-24.

[3] Z. P. Li, M. Y. H. Low, M. Shakeri, and Y.G. Lim, "Cross docking planning and scheduling : problems and algorithms," in SIMTech Technical Reports, 2009, vol. 10, pp. 159-167.

[4] G. Alpan, R. Larbi, and B. Penz, "A bounded dynamic programming approach to schedule operations in a cross docking platform," Comput. Ind. Eng., 2010, vol. 60, pp. 385-396.

[5] T. W. Liao, P. J. Egbelu, and P. C. Chang, "Simultaneous dock assignment and sequencing of inbound trucks under a fixed outbound truck schedule in multi-door cross docking operations," Int. J. Prod. Econ., 2013, vol. 141, pp. 212-229.

[6] K. Lee, B. S. Kim, and C. M. Joo, "Genetic algorithms for door-assigning and sequencing of trucks at distribution centers for the improvement of operational performance," Expert. Syst. Appl., 2012, vol. 39, pp. 12975-12983.

[7] J. Van Belle, P. Valckenaers, G. Vanden Berghe, and D. Cattrysse, "A tabu search approach to the truck scheduling problem with multiple docks and time windows," Comput. Ind. Eng., 2013, vol. 66, pp. 818-826.

[8] Y. Li, A. Lim, and B. Rodrigues, "Crossdocking: JIT Scheduling with Time Windows," J. Oper. Res. Soc., 2014, vol. 55, pp. 1342-1351.

[9] A. R. B. Arabani, S. M. T. F. Ghomi, and M. Zandieh, "A multi-criteria cross-docking scheduling with just-in-time approach," Int. J. Adv. Manuf. Tech., 2010, vol. 49, pp. 741-756.

[10] S. Nguyen and V. Kachivichyanukul, "Movement strategies for multi-objective particle swarm optimization," IJAMC., 2010, vol. 1, no.3, pp. 59-79.

[11] Warisa Wisittipanich and Voratas Kachitvichyanukul, "An efficient PSO algorithm for finding pareto-frontier in multi-objective job shop scheduling problems", IEMS., 2013, Vol.12, No. 2, 151-160.

[12] W. Wisittipanich and P. Hengmeechai, "A multi-objective differential dvolution for just-in-time door assignment and truck scheduling in multi-door cross docking problems," IEMS., 2015, vol. 14, no.3, pp. 299-311.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evol. Comput., 2002, vol. 6, pp. 182-197.

BIOGRAPHY

Warisa Wisittipanich is currently a lecturer in the department of Industrial Engineering, Faculty of Engineering, Chiang Mai Univerity, Chiang Mai, Thailand. She received D. Eng in Industrial and Management Engineering, Asian Institute of Technology, Thailand. Her research interests include operations research, applied operations research, evolutionary algorithm, production scheduling, and lean manufacturing.

Piya Hengmeechai is a graduate student with a master degree in logistic and supply chain management, department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand. His research interested areas include operations research, applied operations research, evolutionary algorithms, and simulation.