Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 1398448, 17 pages
https://doi.org/10.1155/2022/1398448

Research Article

Selective Strategy Differential Evolution for Stochastic Internal Task Scheduling Problem in Cross-Docking Terminals

Dollaya Buakum[1] **and Warisa Wisittipanich**[2,3]

[1]*Department of Industrial Engineering and Manufacturing, Faculty of Engineering, Prince of Songkla University, 15 Karnjanavanich Road, Hat Yai, Songkhla 90110, Thailand*
[2]*Advanced Manufacturing and Management Technology Research Center (AM2Tech), Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, 239 Huay Keaw Road, Suthep, Muang, Chiang Mai 50200, Thailand*
[3]*Supply Chain and Engineering Management Research Unit, Chiang Mai University, Chiang Mai 50200, Thailand*

Correspondence should be addressed to Warisa Wisittipanich; warisa.o@gmail.com

Received 2 June 2022; Revised 1 October 2022; Accepted 19 October 2022; Published 4 November 2022

Academic Editor: Miaolei Zhou

This study proposed an algorithm called selective strategy differential evolution (SSDE) to handle the complexity of the stochastic internal task scheduling problem in cross-docking terminals. The aims of this study are to assign workers and transfer equipment to internal operations and sequence those operations under randomness and uncertainty with the purpose to minimise total tardiness. The main feature of SSDE is its ability to adapt itself in order to execute the best search strategy. The proposed algorithm was tested on 16 instances using generated data based on real-case scenarios of a pharmaceutical distribution centre. The results showed the significant performance of SSDE to other existing algorithms in terms of solution quality and computational time. The key success factors of SSDE are the use of various search strategies in a single run and the application of suitable termination conditions.

1. Introduction

Warehouse management problem has driven researchers to seek the least-cost operation which optimize ordering and holding costs and maximize space utilization by leveraging existing technologies. A cross-docking strategy is one of the logistics practice of unloading goods from incoming trucks and loading them directly into outbound trucks with little to or storage in between. Since cross-docking does not involve storing goods in the warehouse, costs associated with handling and storage are reduced and deliveries are faster. In addition, cross-docking enables greater throughput without the need for opening up a new warehouse. Undeniably, insufficient operational management in cross-docking may obstruct its successful implementation. Recently, studies concerning operation problems in cross-docking have been addressing the real-world problem or specific conditions. Rijal et al. [1] studied truck scheduling and dock door assignment at unit-load cross-dock terminals where dock doors can operate in a mixed service mode. Zheng et al. [2] addressed the cold-chain cross-docking truck scheduling problem. Shahmardan and Sajadieh [3] investigated a truck scheduling problem at a cross-docking centre where inbound trucks were also used as outbound and they can be partially unloaded. Correa Issi, Linfati, and Escobar [4] proposed a mathematical model for truck scheduling in cross-docking in a mixed service mode dock area of a multinational food company in Chile. Chargui et al. [5] proposed a mathematical model to optimize the scheduling, storage, assignment, and sequencing of trucks at receiving and shipping docks for a problem inspired from a multiple door cross-dock facility of an industrial partner with multiple temporary storage zones. However, there exist some limitations to the aforementioned references. One of the common suggestions from those articles is to incorporate stochastic considerations into the problem to provide more practical and novel directions for future studies. In addition, adding a resource-constrained in the model was suggested by Shahmardan and Sajadieh (2020) as a direction for future research.

Task environments in a real-world practice are subject to several uncertainties and randomness. Therefore, a practical model for scheduling problems should be established to address uncertainty issues. Picking orders is typically discussed under the assumptions that processing times and due dates are known in advance and machines are continuously available. However, in practice, some of these assumptions in man-to-goods picking systems are unrealistic. In fact, workers may have different speeds owing to fatigue or other reasons. In addition, workers moving between picking locations can disrupt and lock each other. The phenomenon of pickers locking results in the time loss arising from waiting for the ability to continue the picking process [6].

The stochastic internal task scheduling problem is a more practical model to deal with uncertainties and randomness in a real-world environment. This consideration could fill the current research gap in the cross-docking platform. To date, there are few studies on the stochastic internal task scheduling problem in a cross-dock terminal. Buakum and Wisittipanich [7] proposed a mathematical model for stochastic internal task scheduling to minimise the total tardiness of customer orders. The goal of their study was to simultaneously assign internal cross-docking workers and transportation equipment to obtain the optimal internal task schedule in a single unloading activity. Figure 1 represents a single unloading activity in a cross-docking terminal. In the cross-docking terminal, the cross-docking operations can be divided into two phases: incoming and outgoing phases. In the incoming phase, the containers reach the cross-dock with different products. Each incoming container is processed by a breakdown operation such as scanning or sorting processes. In the outgoing phase, customers' orders are built up. Each customer order is processed by a build-up order operation, and some orders may require value-adding services, such as repacking or labelling.

In 2020, Buakum and Wisittipanich formulated a mathematical model of stochastic internal task scheduling problems in a cross-docking terminal using chance-constrained programming. In their study, an exact method using an optimisation solver was employed to obtain the optimal solution. The preliminary results showed that the problem was NP-hardness (nondeterministic polynomial-time hardness). The model consumed very high amount of computational time when the problem size slightly increased, and it failed to solve a large-scale case of the real-world problem. As a result, the exact method was not robust for dealing with uncertainty of processing time and due date.

Thus, this study focuses on metaheuristic applications to obtain a near-optimal solution within an acceptable computing time. This application aims to apply the stochastic model proposed by Buakum and Wisittipanich [7] in order to schedule large-scale internal tasks for timely medicine delivery to prevent negative effects on patients. Taking inspiration from the aforementioned problem statements, differential evolution (DE), a type of stochastic search and optimisation method, is used in this study. Starting from a population of randomly initialised solutions, the original DE algorithm employs simple mutation and crossover operators to generate new candidate solutions. Therefore, the performance of the DE algorithm is sensitive to the mutation and crossover schemes. Recently, various DE strategies with new mutation and crossover schemes have been introduced to enhance DE searching ability. However, a large amount of time is required to determine the appropriate DE evaluation strategies for specific problems under consideration.

To overcome this dilemma, this study proposes a novel self-adaptive and strategy selection-based DE technique called selective strategy differential evolution (SSDE). The main contributions of this study are listed as follows:

(i) The robust methods of the SSDE algorithm is proposed for dealing with uncertainty of processing time and due date; thus, the SSDE can properly handle large-scale internal task scheduling problems in a cross-docking terminal.

(ii) An intelligible modification procedure that transforms classical DE to the SSDE is presented in this work. In addition, the application of suitable termination conditions is implemented in SSDE.

(iii) The development of encoding and decoding procedures is proposed for finding the optimal order-picking operation in a single unloading manner under stochastic environments.

The remainder of this paper is organised as follows: Section 2 reviews related studies on the internal task scheduling problem in a cross-docking terminal. This section also reviews the modification of the classical DE algorithm and DE applications in cross-docking. Section 3 presents the mathematical model of the problem. Section 4 describes three classical metaheuristics used for experimental comparison in this study. Section 5 introduces the SSDE algorithm, and the solution representation of the encoding and decoding schemes is described in Section 6. A computational experiment using scenarios from a real-case study is reported in Section 7. Finally, the conclusion is provided in Section 8.

2. Related Studies

The internal task scheduling problem in cross-dock terminals was first studied by Li et al. [8]. The problem was considered to be a parallel machine scheduling problem. The mathematical model was formulated with the objective of just-in-time shipments, and the resource constraint in the model was limited to the number of workers. Then, two heuristic algorithms such as squeaky wheel optimisation embedded in a genetic algorithm (SWOGA) and linear programming within a genetic algorithm (LPGA) were implemented to solve the problem and compared with solutions obtained from a CPLEX solver. The comparison results showed that the CPLEX solver has limited memory for solving the problem, whereas metaheuristic methods can provide near-optimal solutions for all test problems. Later,

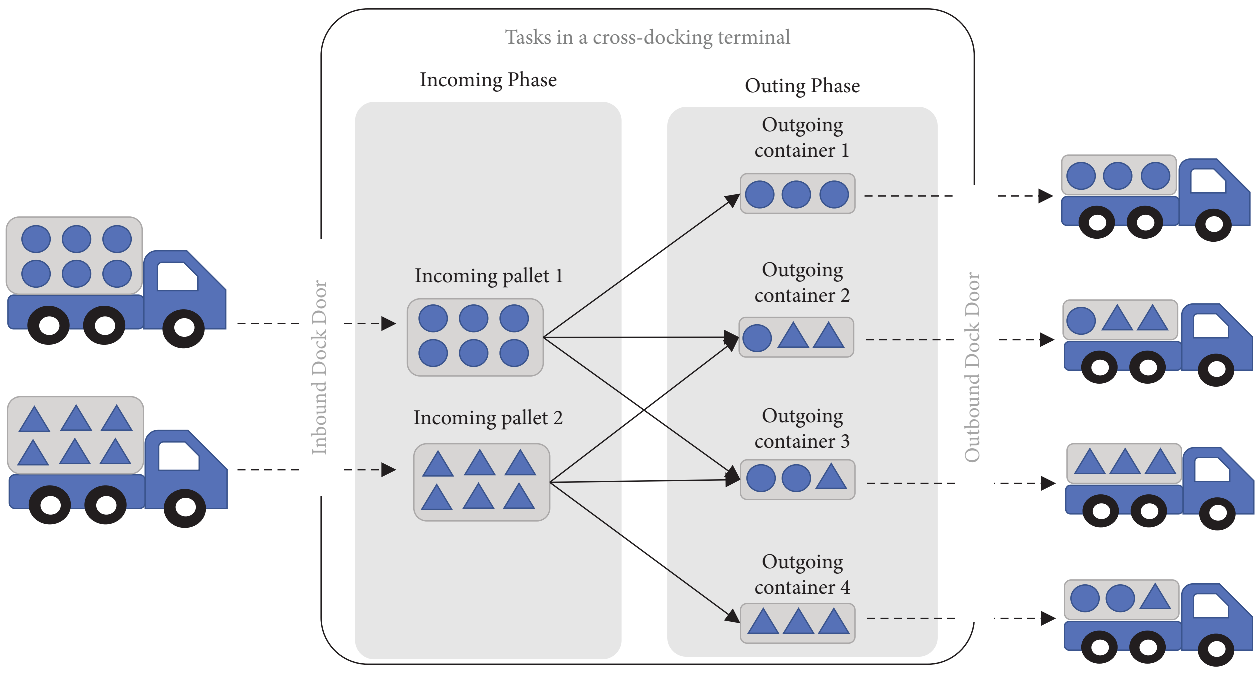

Figure 1: Tasks for a single unloading activity in a cross-docking terminal.

Alvarez-Perez et al. [11] proposed another metaheuristic, the reactive greedy randomised adaptive search procedure and tabu search (RGTS), to solve the same problem as that of Li et al. (2004). The results showed that the RGTS obtains better objective values and less computational time than the SWOGA and LPGA in some instances.

Workers and transferring tools are both important resources for internal tasks in cross-docking terminals in real-world problems. Recently, the developed mathematical model of internal task scheduling in cross-dock terminals was proposed to simultaneously assign workers and transfer equipment for operations to minimise the makespan [12]. The model was formulated in a deterministic environment; however, tasks in the real-world were operated under a stochastic environment. Therefore, Buakum and Wisittipanich [7] transformed the deterministic model into a stochastic model by using chance-constrained programming to minimise the total tardiness, in which the processing times and due dates of operations were considered as stochastic parameters. An exact method using the LINGO optimisation solver was used to find the optimal solution for the generated data. The results showed that LINGO required high computational time and was terminated with memory insufficiency in a large-scale case of the real-world problem. Several research studies have considered the internal task scheduling problem as a parallel machine scheduling (Li et al. [8] [7, 11, 12]). Afshar-Bakeshloo et al. [9] considered the internal task scheduling problem as single machine scheduling while Hamdi and Tekaya [10] considered the internal task scheduling as flow shop scheduling in deterministic environment.

A classical DE was first suggested by Storn and Price [13], which proved its high performance in solving scheduling problems [14–17]. In addition, better DE performance was observed when compared to other evolutionary algorithms (EAs), such as PSO on a suite of 34 widely used benchmark problems [18]. Nevertheless, low convergence speed is a problem of a classical DE, which requires high computational efforts. To overcome this restriction, several variations of the DE algorithm have been suggested to improve the performance of DE by modifying the evolution process during the search or integrating some additional processes into the DE to increase the convergence rate, such as local search [19], external archive [20], or antiaging mechanisms [21]. Pant et al. [22] presented an extensive survey of existence of DE. A number of research articles have been shown through variants of differential evolution like initialization techniques, modifications in mutation schemes, modified crossover schemes, modifications done in selection schemes, changes in parameters, and hybrid variants of different evolution.

Consequently, in terms of adaptive DE variants, different adaptation mechanisms have been introduced to dynamically update the DE algorithm to reproduce new generation of vectors without any prior knowledge of the relationship between the algorithm and the characteristics of optimisation problems. The adaptation mechanisms in the DE algorithm can be categorized into three classes as follows.

2.1. Parameter Adaptation. The performance of DE highly depends on the control parameters in the mutation operator, scaling factor F, the crossover operator, and crossover rate CR. Therefore, some studies have focused on the modification of control parameters in mutation and crossover operations [20, 23–28]. For example, Huynh et al. [28] proposed Q-learning model; the parameter controller adaptively adjusts the algorithm parameters at runtime using information from previous iterations.

2.2. Strategy Adaptation. In 2005, Price et al. [29] proposed other variants of DE derived from the different strategies of mutation and crossover schemes. However, a significant amount of time was required to find the best strategies for each specific problem. In order to automatically select the most suitable strategy while solving a problem without any prior knowledge, some approaches were introduced to apply multistrategies in a single run [30–33]. In the strategy adaptation class, parameter adaptive ability and/or other additional abilities could be implemented along with strategies adaptive ability [34–47]. For example, Do et al. [46] proposed a modified DE by adjusting scale factor F and crossover rate *c* as well as the mutation and selection phases of the original DE are also replaced by the best individual-based mutation and elitist selection techniques. In addition to adaptive mutation or crossover phases, the adaptive initialization phase was considered in this class. For example, Tang and Lee [47] proposed a straightforward and effective scheme for adaptive initialization and coupled with a linear reduction to the DE population size [26], called L-SHADE, for solving constrained optimisation problems.

2.3. Algorithm Adaptation. Since several DE variants have been proposed in recent years, some studies proposed approaches to employ multiple state-of-the-art DE variants for solving particular problems [48, 49].

According to the aforementioned references, Buakum and Wisittipanich [33] proposed the modified DE algorithm in the strategy adaptation class which contained both different mutation and crossover schemes in the strategy pool called self-learning differential evolution (SLDE) algorithm. The adaptive ability of SLDE was made by the ranking and updating probability of each constituent strategy in each generation. Thus, both the mutation and crossover operators were adaptable, and the strategy assignment of the proposed algorithm was based on one active strategy. However, SLDE was designed in deterministic aspect only.

To present a more practical problem and provide novel directions for addressing this problem, this study consider the stochastic internal task scheduling problem and proposed the modified DE algorithm which extends the SLDE in the stochastic aspect. Then, five metaheuristic algorithms, GA, PSO, DE, RAM-EPSDE, and SSDE are applied in the comparison experiments.

3. Mathematical Model of the Problem

The problem statement and a mathematical model were taken from Buakum and Wisittipanich [13]. The problem was formulated to minimise the total tardiness of the order-picking operation according to prescriptions in a single unloading activity. In a stochastic environment, processing times and due dates were set as random parameters, and their distributions were assumed to be known in advance. Based on the limited number of workers and transfer equipment, the goal of the formulation was to obtain an optimal schedule to simultaneously assign workers and transfer equipment for handling medicine pallets and prescriptions. The due date of each prescription was set based on the delivery date and patient location. The basic assumptions of the mathematical model were shown as follows:

(i) All workers are identical and 100% reliable.
(ii) No preemptions occur in scheduling.
(iii) The optimal schedule is based on a single unloading activity. Thus, the picking order of each prescription proceeds once the medicine pallets are completely unloaded.

In the stochastic model formulation, the processing time for breaking down the medicine pallets and picking order of the prescription was assumed to be random variables subjected to a normal distribution, while the due date of the picking order was assumed to be a random variable subject to a uniform distribution.

3.1. Indices

i,i': incoming medicine pallet ($i,i' = 1,\ldots,n$)

j,j': outgoing medicine container ($j, j' = 1,\ldots,o$)

k: worker ($k = 1,\ldots,m$)

l: transfer equipment; TF ($l = 1,\ldots,q$)

3.2. Decision Variables

x_{ik} is 1 if worker k is ass igne d for incom ing pallet i; otherwise, 0

y_{jk} is 1 if worker k is assigne d for outgoing container j; otherwise, 0

$I_{ii'k}$ is 1 if incom ing pallet i prece de s i' by the same worker k; otherwise, 0

$J_{jj'k}$ is 1 if outgoing container j prece de s j' by the same worker k; otherwise, 0

$a_{ii'l}$ is 1 if incoming pa llet i prece de s i' by the same TF l; otherwise, 0

$b_{jj'l}$ is 1 if outgoing container j prece de s j' by the s ame TF l; otherwise, 0

U_{ikl} is 1 if worker k an d tool l are assigne d for incom ing pallet i; otherwise, 0

V_{jkl} is 1 if outgoing container j is processe d by worker k an d TF l; otherwise, 0

α_{ikl} is the start time of medicine incoming pallet i break down by worker k TF l

β_{jkl} is the start time of outgoing medicine container j preparation by worker k TF l

c_{ikl} is the completion time of incoming pallet i break down by team k TF l

z_{jkl} is the completion time of outgoing medicine container j by worker k TF l

t_j is the tardiness of outgoing medicine container j

3.3. Parameters

n is the number of incoming medicine pallets

o is the number of outgoing medicine containers

m is the number of workers

q is the number of transfer equipment

r_i is the ready time of incoming medicine pallet i break down

$p\mathrm{MU}_i$ is the mean processing time required to break down incoming medicine pallet i

$p\mathrm{STD}_i$ is the standard deviation of processing time required to break down incoming medicine pallet i

$po\mathrm{MU}_j$ is the mean processing time for preparing outgoing medicine container j

$po\mathrm{STD}_j$ is the standard deviation of processing time for preparing outgoing medicine container j

dU_j is the upper point of due date of outgoing medicine container j

dL_j is the lower point of due date of outgoing medicine container j

S_{ij} is 1 if outgoing medicine container j is pulled from medicine pallet i; otherwise, 0

h_{il} is 1 if incoming medicine pallet i can be transferred by TF l; otherwise, 0

f_{jl} is 1 if outgoing medicine container j can be transferred by TF l; otherwise, 0

G is a large number

ρ is the confidence level for satisfaction with the constraint set, $1 \geq \rho \geq 0$

3.4. Objective Function

$$\text{Minimise} \quad \sum_{j=1}^{n} t_j, \forall j \, . \tag{1}$$

Equation (1) is the objective function for minimising the total tardiness.

3.5. Constraints. Equations (2) and (3) ensure that each incoming medicine pallet and outgoing medicine container must be processed by one worker, respectively.

$$\sum_{k=1}^{m} x_{ik} = 1, \forall i, \tag{2}$$

$$\sum_{k=1}^{m} y_{jk} = 1, \forall j. \tag{3}$$

Equations (4)–(7) guarantee the precedence relationship when incoming medicine pallets or outgoing medicine containers are processed with the same worker [11]:

$$x_{ik} + x_{i'k} - \left(I_{ii'k} + I_{i'ik}\right) \leq 1, \forall ii'k, i \neq i', \tag{4}$$

$$2\left(I_{ii'k} + I_{i'ik}\right) - x_{ik} - x_{i'k} \leq 0, \forall ii'k, i \neq i', \tag{5}$$

$$y_{jk} + y_{j'k} - \left(J_{jj'k} + j_{j'jk}\right) \leq 1, \forall jj'k, j \neq j', \tag{6}$$

$$2\left(J_{jj'k} + J_{j'jk}\right) - y_{jk} - y_{j'k} \leq 0, \forall jj'k, j \neq j'. \tag{7}$$

Equations (8) and (9) guarantee the relationships when incoming medicine pallets or outgoing medicine containers are transferred by the same transfer equipment, respectively.

$$U_{ikl} + U_{i'kl} - \left(a_{ii'l} + a_{i'il}\right) \leq 1, \forall ii'l, i \neq i', \tag{8}$$

$$V_{jkl} + V_{j'kl} - \left(b_{jj'l} + b_{j'jl}\right) \leq 1, \forall jj'l, j \neq j'. \tag{9}$$

Equations (10)–(13) ensure that each incoming medicine pallet or outgoing medicine container must be transferred by only one transfer equipment that is eligible for use.

$$\sum_{k=1}^{m} \sum_{l=1}^{q} U_{ikl} = 1, \forall i, \tag{10}$$

$$\sum_{k=1}^{m} \sum_{l=1}^{q} V_{jkl} = 1, \forall j, \tag{11}$$

$$\sum_{k=1}^{m} \sum_{l=1}^{q} U_{ikl} \times h_{il} \times x_{ik} = 1, \forall i, \tag{12}$$

$$\sum_{k=1}^{m} \sum_{l=1}^{q} V_{jkl} \times f_{jl} \times y_{ik} = 1, \forall j. \tag{13}$$

Equations (14)–(17) state that the probability of the constraint sets must be satisfied by at least ρ to ensure sufficient time between breaking down an incoming medicine pallet and to prepare an outgoing medicine container for each worker or piece of transferring equipment.

$$\text{Prob}\left\{c_{ikl} \leq \left(c_{i'kl} - p_{i'}\right) + G\left(1 - I_{ii'k}\right)\right\} \geq \rho, \forall l, \forall ii'k, i \neq i', \tag{14}$$

$$\text{Prob}\left\{c_{ikl} \leq \left(c_{i'kl} - p_{i'}\right) + G\left(1 - a_{ii'l}\right)\right\} \geq \rho, \forall k, \forall ii'l, i \neq i', \tag{15}$$

$$\text{Prob}\left\{z_{jkl} \leq \left(z_{j'kl} - po_{j'}\right) + G\left(1 - J_{jj'k}\right)\right\} \geq \rho, \forall ljj'k, j \neq j', \tag{16}$$

$$\text{Prob}\left\{z_{jkl} \leq \left(z_{j'kl} - po_{j'}\right) + G\left(1 - b_{jj'l}\right)\right\} \geq \rho, \forall kjj'l, j \neq j'. \tag{17}$$

Equations (18) and (19) enforce the start time of breaking down an incoming medicine pallet and start time of preparing an outgoing medicine container, respectively.

$$\alpha_{ikl} \geq r_i, \forall ikl, \tag{18}$$

$$\beta_{jkl} \geq c_{ikl}, \forall ijkl. \tag{19}$$

Equations (20) and (21) state that the probability of the constraint sets must be satisfied by at least ρ to ensure the ready times of braking down an incoming medicine pallet and preparing an outgoing medicine container, respectively.

$$\mathrm{Prob}\{c_{ikl} - r_i \geq p_i\} \geq \rho, \forall ikl, \tag{20}$$

$$\mathrm{Prob}\left\{z_{jkl} - c_{ikl} \geq po_j\right\} \geq \rho, \forall jk;\ i = 1\text{st}, .., \text{last predecessor of } j. \tag{21}$$

Equations (22) and (23) state that the probability of the constraint sets must be satisfied by at least ρ to determine the completion time of each incoming medicine pallet and outgoing medicine container, respectively.

$$\mathrm{Prob}\{\alpha_{ikl} + p_i \geq c_i\} \geq \rho, \forall ikl, \tag{22}$$

$$\mathrm{Prob}\left\{\beta_{jkl} + po_j \geq z_j\right\} \geq \rho, \forall ikl. \tag{23}$$

Equation (24) ensures that the probability of tardiness for outgoing medicine container j must be satisfied by at least ρ.

$$\mathrm{Prob}\left\{t_j \geq \max\left(0, c_{jkv} - d_j\right)\right\} \geq \rho, \forall j, k, v. \tag{24}$$

Equation (25) specifies that all decision variables are binary.

$$y_{ik}, Y_{jk}, I_{ii'k}, J_{jj'k}, a_{ii'l}, A_{jj'l}, U_{ikl}, V_{jkl} \in \{0, 1\}. \tag{25}$$

4. Classical Metaheuristic Approach

4.1. GA. The genetic algorithm (GA) is an adaptive search technique used to solve optimisation problems. Although the GA was developed much earlier than 1975, the basic principles of GA were first emphasized by Holland in 1975 [50]. The GA evolution procedure is based on the laws of natural selection and genetics. In the GA, the solutions are represented as chromosomes. The chromosomes are evaluated for fitness values and ranked from best to worst based on their fitness values.

During a genetic operation, chromosomes are selected from the population and recombined to produce offspring that comprise the population of the next generation. This process is accomplished through repeated applications of three genetic operators: selection, crossover, and mutation.

4.2. PSO. Particle swarm optimisation (PSO) is a population-based random search method that imitates the physical movements of individuals in a swarm as a searching mechanism. The PSO algorithm was originally proposed by Eberhart and Kennedy in 1995 [51]. Similar to the GA, the population in PSO is initialised with random solutions. In PSO, a solution is represented as a particle, and the population of solutions is called a swarm of particles. Each particle has two main attributes: position and velocity. The difference between PSO and GA lies in evolutionary procedures. The key concept of PSO is that each particle learns from the cognitive knowledge of its experiences (personal best, *pbest*) and the social knowledge of the swarm (global best, *gbest*) to guide itself to a better position. A particle moves to a new position using an updated velocity. Once a new position is reached, the best position of each particle and the best position of the swarm are updated as needed. The velocity of each particle is then adjusted based on the particle experiences. This process is repeated until a stopping criterion is satisfied.

4.3. DE. Differential evolution (DE), introduced by Storn and Price in 1995 [13], is an evolutionary algorithm designed to deal with continuous optimisation problems. Similar to GA and PSO, DE is a population-based random search algorithm. In DE, the initial D-dimensional vector population of size N is randomly generated and should cover the entire search space. Typically, the DE population evolves through repeated cycles of three main DE operators: mutation, crossover, and selection. However, DE has a different mechanism for generating new solutions explained as follows.

4.3.1. Mutation. A mutant vector is generated by combining three vectors randomly selected from the population, excluding the target vector, X_G. Equation (26) shows the combination process of three randomly selected vectors to form the mutant vector V_G as follows:

$$V_G = X_{1,G} + F\left(X_{2,G} - X_{3,G}\right), \tag{26}$$

where $X_{1,G}$, $X_{2,G}$, and $X_{3,G}$ are three randomly selected vectors from the population in generation G and F is a scale factor which is the main parameter of the DE algorithm.

4.3.2. Crossover. To increase the diversity of the perturbed parameter vectors after mutation, a trial vector, U_G, is generated by a crossover operation. The crossover probability $(0 \leq C_r \leq 1)$ must be specified for the crossover operator to control the probability of selecting the value in each dimension from a mutant vector. In classic DE, the uniform crossover is employed, and the trial vector is generated using the following equation:

$$U_{j,G} = \begin{cases} V_{j,G}, & \text{if } (randb(j) \leq C_r, \\ X_{j,G}, & \text{Otherwise,} \end{cases} \tag{27}$$

where *randb(j)* is the j^{th} evaluation of a uniform random number generator with the outcome $\in [0, 1]$.

4.3.3. Selection. To determine a survival vector for the next generation, a comparison of fitness values between a trial vector, U_G, and a target vector, X_G, is performed in the selection operator. The simple criterion is to maintain the vector with a better fitness value. If U_G yields a better fitness value than X_G, X_{G+1} is set to U_G; otherwise, the old value X_G is retained.

4.4. RAM-EPSDE. Ranked-Based Mutation Adaptation (RAM), proposed by Leon and Xiong in 2018, is a selection method of different mutation strategies of DE. In RAM, the mutation strategy was selected according to probabilities of each population subgroups. RAM was used to enhance the classical DE algorithm by integrating with other adaptive DE algorithms.

Ensemble of Parameters and Mutation Strategies DE (EPSDE) algorithm, proposed by [31], was another adaptive parameter and searching strategies DE. EPSDE randomly selected different mutation and crossover strategies and parameter values in its pool listed as follows:

(1) Mutation strategies are DE/best/2/bin, DE/rand/1, and DE/current-to-rand/1/bin

(2) Crossover strategies are binomial crossover and exponential crossover

(3) Parameter values are the population size ($NP = 50$), $F \in [0.5, 0.9]$, and $\mathrm{CR} \in [0.1, 0.5, 0.9]$

Later, Leon et al. [19] introduced an integration of RAM into EPSDE and named the algorithm as RAM-EPSDE with an aim to improve the search efficiency.

5. SSDE

5.1. SSDE Framework. Selective strategy DE (SSDE) is proposed in this study to enhance the DE performance. The modifications to the classical DE algorithm are listed in Table 1. In contrast to the classical DE, SSDE uses six different strategies obtaining from the combination of mutation and crossover schemes proposed by Price et al. [29] in order to maintain both the exploitation and exploration abilities of DE [29]. In the classic DE, a new population is generated using one strategy, while a new population of SSDE can be generated using various strategies. The ability to adjust itself to use different potential strategies towards a better solution is the key to enhancing the search speed of the SSDE.

In addition, to improve the computational time cost, other termination conditions are added to SSDE as follows:

(i) Maximum iterations: similar to classical DE and other evolutionary algorithms, SSDE algorithm is terminated when the maximum number of iterations is reached

(ii) No further improvement: different from the termination condition of classical DE, the algorithm is terminated when an improvement probability does not increase for a specific number of n iterations

(iii) The optimal solution is obtained: when the objective reaches the target value, for example, zero tardiness, the algorithm is terminated

5.2. SSDE Procedure. The SSDE procedure is divided into two main stages: learning and running. In the learning stage, each strategy is used to perform the evaluation process for n iterations. Then, the ability to achieve a better fitness value for each strategy is recorded according to the improvement probability. Next, the improvement probabilities of all strategies are sorted in ascending order to rank for qualities. In the running stage, the improvement probability is updated for every m iterations. First, the evolutionary process is performed by employing the first-rank strategy. Subsequently, if the improvement probability of the candidate strategy is lower than that of the others after the update, SSDE switches its strategy to the highest improvement probability strategy. This process is repeated until the algorithm terminates. The pseudocode of the SSDE algorithm with the proposed modification is shown in Figure 2.

6. Solution Representation

Since evolutionary algorithms were developed for continuous optimisation, each individual in the population must be transformed into a practical solution. In this study, a solution representation with encoding and decoding was developed to obtain the schedule of internal tasks in a cross-dock terminal with the minimum total tardiness.

In the encoding procedure, the number of dimensions was set to be equal to the total number of operations in a single unloading activity. Then, a uniform random number was generated for each dimension in the interval [0, 1]. The pseudocode of the encoding procedure is shown in Figure 3.

The decoding procedure transforms random numbers of each dimension into a practical solution. The decoding procedure consists of the following steps:

Step 1: defining the condition for assigning workers to operations according to the range and the boundary value, which are calculated from the dimension value and number of workers.

Step 2: allocating the transferring equipment to operations in a balanced manner. The assignment in this step is based on the dimension value and its eligibility for handling that operation.

Step 3: applying a sorting list rule to generate a sequence of operations.

Step 4: generating random values of processing times and due dates according to the stochastic environment of the problem. Then, the start time, end time, the tardiness of each operation, and the total tardiness were calculated according to the random processing times and due dates. The pseudocode of the decoding pro-

Table 1: Modifications in DE algorithm.

No.	Modification	Classical DE	SSDE
1	Mutation operator	1 scheme $V_G = X_{1,G} + F(X_{1,G} - X_{2,G})$	3 schemes $V_G = X_{1,G} + F(X_{1,G} - X_{2,G})$ $V_G = X_{b,G} + F(X_{2,G} - X_{2,G})$ $V_G = X_{b,G} + F(X_{1,G} - X_{2,G}) + F(X_{3,G} - X_{4,G})$
2	Crossover operator	1 scheme *Binomial crossover*	2 schemes *Binomial crossover* *Exponential crossover*
3	Generating new population	Using a DE strategy in a single run	Using various DE strategies in a single run *There are 6 strategies from the combination of mutation and crossover schemes*
4	Termination condition	1 condition *Max iterations*	3 condition *Max iterations* *No more improvement* *Objective reach to the target*

```
Algorithm: SSDE
1.  Initialize individuals randomly.
2.  Evaluate objective value of each target vector.
3.  Operation in Learning phase
        For strategy $i = 0$; $i <$ number of strategy; $i$++
            {
            For iteration $i = 0$; $i < n$ iteration; $i$++
                {
                Perform mutation and crossover to obtain trial vector;
                Selection between target vector and its trial vector;
                Update the global best vector;
                Update improvement count
                    {
                    IF the best fitness value $i$ <= the best fitness value $i$-1
                        {
                        Improvement count +=1;
                        {
                    }
                }
            Calculate Improvement probability = Improvement count/$n$ iteration;
            }
        Rank strategy by sorting the improvement probability from min to max;
4.  Operation in Running phase
        For iteration $i = n$; $i <$ max iteration; $i$++
            {
            For strategy $i = 0$; $i <$ number of strategy; $i$++
                {
                For iteration $i = 0$; $i < m$ iteration; $i$++
                    {
                    Perform mutation and crossover to obtain trial vector;
                    Selection between target vector and its trial vector;
                    Update the global best vector;
                    Evaluate objective value of the global best vector
                        {
                        IF the best fitness value = 0
                            {
                            End;
                            {
                        }
                    Update improvement count
                        {
                            IF the best fitness value $i$ <= the best fitness value $i$-1
                                {
                                Improvement count +=1;
                                {
                        }
                    }
                  Calculate Improvement probability = Improvement count/$m$ iteration;
                  IF Improvement probability strategy $i$ <= Improvement probability strategy $i$+1
                    {
                    End;
                    {
                }
            }
        End;
```

Figure 2: The pseudocode of the SSDE algorithm.

```
Encoding
1.  Specify the number of individual dimensions
        For Dimension i=0; i < number of operations; i++
            {
            Dimension += number of operations;
            {
2.  Generate random number for each dimension
        For Dimension i=0; i < number of dimension; i++
            {
            Dimension value i = Continuous uniform random number ∈ [0,1];
            {
```

Figure 3: The pseudocode of the encoding procedure.

cedure is shown in Figure 4.

7. Computational Experiments

The computational experiments were divided into two parts. SSDE performance is first tested using benchmark functions. Then, SSDE is used for solving the larger-scale problem of internal tasks scheduling in a cross-docking terminal. The performance of SSDE was evaluated and compared with other metaheuristics, GA, PSO, DE, and RAM-EPSDE. The analysis of numerical results in terms of solution quality and convergence behaviour was also investigated. It is noted that the computational experiments were run on an Intel® Pentium IntelCoreTM i5 8th Gen CPU (1.60 GHz) with 8 GB RAM.

7.1. Benchmark Test Functions. In this section, the performance of the SSDE is verified using five benchmark functions. These functions are bound-constrained high-dimensional single-objective optimisation (Dim = 100/1000/2000) of CEC 2020 [52]. The unimodal functions $F1$ and $F2$ have only one global optimal value, so they can be used to test the local search ability of the algorithm, while the multimodal functions $F3$–$F5$ with more than two optimal values are used to evaluate the global exploration ability. The benchmark functions used in this study are listed in Table 2.

The numeric experiment results obtained from SSDE were compared with those from other algorithms, including MFO [53], SOGWO [54], PFA [55], EO [56], and MMPA [57]. The parameter setting of the comparison algorithms is shown in Table 3.

These parameters were selected according to the suggestions presented by the developers of the algorithms in the aforementioned references. The population size of each algorithm was set to 30, and the number of iterations was set to 500. Table 4 shows the comparison results of SSDE and other existing algorithms obtained from 30 independent runs.

According to the results in Table 4, although the SSDE is not able to obtain better solutions than SOGWO and MMPA, it performs better than PSO, MFO, and PFA in both unimodal and multimodal functions. This can be due to the fact that SSDE was particularly developed for solving a combinatorial problem, not a continuous optimisation problem. In the next section, experiments are conducted to demonstrate the performance of SSDE in solving internal tasks scheduling problems in cross-docking terminals.

7.2. Parameter Setting and Instances of Internal Task Scheduling Problem. To determine the best set of key parameter values of each algorithm, the Taguchi method was used to reduce the computational load from a full factorial experiment. For each algorithm, a three-factor with four-level experiment was performed on three instances of the scenario of the real-world cross-docking problem. It is noted that, in this study, the parameters of the RAM-EPSDE were set according to the original EPSDE in [31]. In addition, the number of function evaluations (number of iterations × population size) was set to a fixed value of 50,000 for all algorithms. The best parameter values for each algorithm derived from the parameter settings are listed in Table 5.

The computational experiment was performed on 15 instances. Instances 1 to 7 were small-size problems with generated data. Instances 8 to 15 were large-size problems in which the data were derived from the real-scenario of a medicine distribution centre. The problem size is identified by four generated parameters which are number of workers (m), number of transfer equipment (q), number of incoming medicine pallets (n), and number of outgoing medicine containers (o). In addition, to make the problem more practical, in some instances, some customer orders are restricted for the use of particular transferring equipment.

7.3. Experimental Results on the Instances of Internal Task Scheduling Problem. In this section, the performance of the proposed SSDE was evaluated using the real-case scenarios of a pharmaceutical distribution centre. The experimental results obtained from the SSDE, in terms of solution quality and computational time, were compared to solutions obtained from the optimisation solver LINGO and other metaheuristics. It is noted that the comparison of all metaheuristics was executed under the same conditions of solution representation and total number of function evaluations. Table 6 shows the comparison results of total tardiness among different metaheuristics. The best, average, and standard deviation of total tardiness obtained from those algorithms are also reported.

It can be seen from Table 6 that, for small-size problems, the total tardiness obtained from the metaheuristic method was lower than the exact method. This may be because of a

```
Decoding
1.  Assign working team to operations
        Range = (Maximum Dimension value – Minimum Dimension value)/Number of working team;
        Number of boundary = Number of working team + 1;
        For Boundary i = 0;
            {
            Boundary value i = Minimum Dimension value;
            {
        For Boundary i = 1; i < number of boundary; i++
            {
            Boundary value i = Boundary value i − 1 + Range;
            }
        For Operation i = 0; i < number of operations; i++
            {
            For Working team i = 0; i < number of working team; i++
                {
                IF Dimension value >= Boundary value i && Dimension value<Boundary value i + 1
                    {
                    Assign working team i;
                    Break;
                    }
                }
            }
2.  Assign transferring equipment to operations
        For Operation i = 0; i < number of operations; i++
            {
            For TF i = 0; i < number of TF; i++
                {
                IF Operation i % number of TF == i && dimension value>=low &&dimension value<up
                    {
                    IF eligibility of TF i =1
                        {
                        Assign transferring equipment i;
                        Break;
                        }
                    }
                 }
            }
3.  Determine sequence of operations
        Sort dimension value of break down operation from min to max;
        Sort dimension value of picking order operation from min to max;
        Add sequence to each operation;
```

Figure 4: Pseudocode of the decoding procedure.

Table 2: The benchmark functions.

ID	Function	Range	$f_{\min}$
$F1$	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	[−10, 10]	0
$F2$	$f(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$	[−100, 100]	0
$F3$	$(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[−5.12, 5.12]	0
$F4$	$f(x) = -20\exp\left(-20\sqrt{1/n\sum_{i=1}^{n} x_i^2} - \exp\left(1/n\sum_{i=1}^{n}\cos(2\pi x_i)\right)\right.$	[−32, 32]	0
$F5$	$f(x) = 1/4000\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1$	[−600, 600]	0

Table 3: Parameter setting for testing the benchmark functions.

Algorithm	Parameters
MFO	$a = -1$ (linearly decreased over iterations)
SOGWO	$a = 2$ (linearly decreased over iterations)
PFA	~
EO	$a1 = 2$, $a2 = 1$, $GP = 0.5$, $t = 1$ (nonlinearly decreased over iterations)
MMPA	$FADs = 0.2$
SSDE	$F = 0.2$, $Cr = 0.9$

Table 4: Comparison results among different algorithms for unimodal and multimodal functions with 100 D/1000 D/2000 D.

ID	Dimensions	PSO		MFO		PFA		SOGWO		MMPA		SSDE		
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Best
	100	$3.87E+01$	$9.22E+00$	$2.42E+02$	$4.33E+01$	$2.51E-17$	$7.89E-02$	$4.16E-08$	$1.45E-08$	$0.00E+00$	$0.00E+00$	$6.70E+01$	$4.39E+00$	$5.17E+01$
*F*1	1000	$1.39E+03$	$5.48E+01$	Infeasible		Infeasible		$6.62E-01$	$3.00E-01$	$0.00E+00$	$0.00E+00$	$1.28E+03$	$6.01E+01$	$1.13E+03$
	2000	$1.52E+42$	$7.71E+42$	Infeasible		Infeasible		$9.74E+00$	$3.79E+00$	$3.47E-51$	$1.32E-50$	$2.59E+04$	$1.58E+03$	$2.26E+04$
	100	$2.02E+01$	$5.55E+00$	$5.83E+04$	$1.31E+04$	$4.36E+00$	$1.63E+00$	$1.01E+01$	$1.10E+00$	$4.41E+00$	$4.17E-01$	$9.81E+01$	$6.40E+00$	$6.76E+01$
*F*2	1000	$4.10E+04$	$1.86E+03$	$1.30E+04$	$4.94E+04$	$2.29E+05$	$2.74E+04$	$2.03E+02$	$2.03E+02$	$1.94E+02$	$3.20E+00$	$3.63E+04$	$3.08E+03$	$3.10E+04$
	2000	$1.88E+05$	$6.32E+03$	$5.94E+06$	$6.94E+04$	$1.02E+06$	$9.48E+04$	$8.23E+04$	$8.00E+04$	$4.26E+02$	$2.52E+00$	$7.56E+05$	$7.87E+04$	$5.86E+05$
	100	$6.06E+02$	$7.36E+01$	$8.59E+02$	$7.15E+01$	$4.87E+02$	$6.42E+01$	$7.63E+00$	$5.54E+00$	$0.00E+00$	$0.00E+00$	$2.62E+02$	$2.94E+01$	$1.98E+01$
*F*3	1000	$1.47E+04$	$6.48E+02$	$1.55E+04$	$2.09E+02$	$1.03E+04$	$4.34E+02$	$1.99E+02$	$4.99E+01$	$0.00E+00$	$0.00E+00$	$7.31E+03$	$4.01E+02$	$6.26E+03$
	2000	$3.15E+04$	$1.03E+03$	$3.31E+04$	$2.74E+02$	$2.35E+04$	$8.57E+02$	$5.59E+02$	$1.07E+02$	$0.00E+00$	$0.00E+00$	$1.92E+04$	$6.38E+02$	$1.72E+04$
	100	$3.73E+00$	$2.96E-01$	$1.99E+01$	$9.58E-02$	$6.18E+00$	$6.46E+00$	$1.44E-07$	$5.68E-08$	$8.88E-16$	$0.00E+00$	$5.41E+00$	$7.62E-01$	$3.76E+00$
*F*4	1000	$1.59E+01$	$2.90E-01$	$2.04E+01$	$2.03E-01$	$1.99E+01$	$5.90E-01$	$1.92E-02$	$2.83E-03$	$8.88E-16$	$0.00E+00$	$8.23E+00$	$5.05E-01$	$6.86E+00$
	2000	$1.76E+01$	$1.13E-01$	$2.04E+01$	$2.67E-01$	$2.04E+01$	$2.99E-01$	$1.18E-01$	$1.50E-02$	$8.88E-16$	$0.00E+00$	$9.86E+00$	$4.37E-01$	$8.75E+00$
	100	$3.93E-01$	$9.30E-02$	$5.30E+02$	$1.62E+02$	$6.93E-01$	$2.20E-01$	$8.69E-03$	$1.25E-02$	$0.00E+00$	$0.00E+00$	$6.16E+00$	$2.23E+00$	$1.99E+00$
*F*5	1000	$2.80E+02$	$1.71E+01$	$2.45E+04$	$5.24E+02$	$2.16E+03$	$2.18E+02$	$1.55E-01$	$1.16E-01$	$0.00E+00$	$0.00E+00$	$2.84E+02$	$3.26E+01$	$2.13E+02$
	2000	$7.55E+02$	$3.21E+01$	$5.36E+04$	$6.40E+02$	$9.22E+03$	$8.38E+02$	$1.14E+00$	$2.22E-01$	$0.00E+00$	$0.00E+00$	$7.62E+02$	$1.42E+02$	$4.57E+02$

Table 5: Best parameter values for each algorithm.

Algorithm	Parameters and their best level		
	Function evaluation	Mutation rate	Crossover rate, C_r
GA	100×500	0.10	0.3
	Function evaluation	Inertia weight	Acceleration (C_p, C_g)
PSO	200×250	0.6–1.0	0.7, 0.7
	Function evaluation	Scale factor, F	Crossover rate, C_r
DE	500×100	1.0	0.6
RAM-ESPDE	1000×50	[0.5, 0.9]	[0.1, 0.5, 0.9]
SSDE	100×500	2.0	0.9

stochastic setting in the mathematical model of the problem. In addition, there was no difference in the recording results including the best, mean, and standard deviation among metaheuristic methods. However, for large-sized problems, LINGO could not find solution since it ran out of memory for generating the model. Among metaheuristics, SSDE generally provided superior solution quality than other algorithms since most of the best, average, and standard deviation values of total tardiness are lower. In addition, SSDE clearly outperformed GA, PSO, classic DE, and RAM-EPSDE by providing solutions with zero tardiness values and zero standard deviations for most instances. It is noteworthy that the quality of the solutions obtained by the metaheuristic methods deteriorated as the problem size increased. This may suggest that newly generated solutions could not escape from the local optimal when the problem became more complex. Therefore, a high diversification for generating new populations leads to a better solution. Consequently, GA provided the highest total tardiness, whereas PSO, classic DE, and RAM-EPSDE yielded lower total tardiness. In conclusion, the numerical results showed that SSDE yielded the best total tardiness and computational time.

Moreover, a one-sided t-test was performed to compare SSDE performance with those of the classical DE algorithm and RAM-EPSDE in terms of the average total tardiness for large-size instances. The t-value according to equation (28) was calculated to confirm the significant difference of compared algorithms.

$$t = \frac{\left(\overline{x}_{\text{Compared algo}} - \overline{x}_{\text{SSDE}}\right)}{\sqrt{s^2_{\text{Compared algo}}/n_{\text{Compared algo}} + s^2_{\text{SSDE}}/n_{\text{SSDE}}}}. \tag{28}$$

In Equation (28), $\overline{x}$ and s are the mean and standard deviations of total tardiness obtained from 30 replicated runs (n), respectively. It is noteworthy that α is an uncertainty level, set at 5% for this experiment, and n is the degree of freedom ($n = 30 - 1 = 29$). Hence, the critical value of $t_{0.05,\,29}$ is equal to 1.699. If t-value is higher than the critical value, the average total tardiness obtained from SSDE is significantly lower. Table 7 shows the results of a one-sided t-test of average total tardiness. It can be seen that the t-values of average total tardiness were higher than the critical value for all large-size instances. Thus, it was clear that the total tardiness obtained from the SSDE was significantly lower than that of the classical DE and RAM-EPSDE with 95% confidence level.

In addition, the ability to escape from the local optimal and attain a better solution can be observed by behaviour of convergence. In this study, the comparison of convergence behaviour among different algorithms was investigated using the graph of average fitness values versus the number of function evaluations as shown in Figure 5.

According to Figure 5, GA showed the lowest convergence speed, while PSO and DE yielded better speeds. This may be because the task scheduling problem requires diversification to generate new populations through an evolutionary process and handle a large number of prescriptions. The convergence speeds of RAM-EPSDE were better than the classical DE due to the use of multisearching strategies. Although integrating RAM with EPSDE leads to an improvement of the original DE, crossover strategies and parameter settings are not considered in the adaptation. In contrast, SLDE adapts both of its mutation and crossover strategies. In addition, the parameter values of SLDE were derived from the parameter testing using the real-case study. Consequently, SLDE showed its robustness compared to RAM-EPSDE in internal task scheduling problems. In addition, SSDE showed its outstanding competence in escaping from the local optimal and reaching a better solution. The superior performance of the SSDE is attributed to the application of various modified DE strategies and self-adaptive methods through the evolutionary processes of SSDE.

Table 6: Comparison of total tardiness and computational time among different algorithms.

Ins	Problem size	Lingo		GA				PSO				DE				RAM-EPSDE				SSDE			
	(m, q, n, o)	Tardiness	Run time (s)	Tardiness			Run time (s)	Tardiness			Run time (s)	Tardiness			Run time (s)	Tardiness			Run time (s)	Tardiness			Run time (s)
				Best	Avg	SD	Avg	Best	Avg	SD	Avg	Best	Avg	SD	Avg	Best	Avg	SD	Avg	Best	Avg	SD	Avg
1	2, 2, 3, 3	0.97	35.2	1.0	1.4	0.4	0.1	1.0	1.4	0.3	0.2	1.1	1.6	0.5	0.1	1.0	1.3	0.3	0.2	1.0	1.5	0.4	0.2
2	2, 2, 4, 3	9.15	137.4	6.9	8.5	0.8	0.2	6.3	8.2	1.0	0.2	6.2	8.5	1.3	0.2	9.2	10.0	1.0	0.3	6.8	8.9	1.0	0.3
3	2, 2, 4, 4	12.12	513.5	12.6	14.0	0.7	0.2	12.9	13.9	0.6	0.2	12.5	14.0	0.5	0.2	13.2	14.0	0.4	0.3	12.7	14.0	0.5	0.3
4	3, 3, 4, 5	30.31	1169.2	9.5	10.5	0.6	0.2	9.0	10.7	0.7	0.3	9.2	10.4	0.6	0.2	9.4	10.3	0.6	0.3	9.3	10.4	0.6	0.3
5	3, 3, 4, 6	29.39	1452.1	8.4	9.5	0.5	0.3	8.5	9.5	0.5	0.3	8.8	9.5	0.4	0.3	9.1	9.7	0.5	0.3	8.5	9.6	0.5	0.3
6	3, 3, 4, 7	72.63	15579.7	18.8	19.8	0.5	0.3	18.5	19.6	0.7	0.4	18.4	19.7	0.5	0.3	21.3	21.9	0.4	0.3	19.4	21.4	1.4	0.3
7	3, 3, 4, 8	84.41	197696.9	28.4	29.9	0.9	0.3	28.4	30.0	0.9	0.4	28.6	30.0	0.7	0.3	28.4	29.5	0.8	0.4	28.6	30.7	1.2	0.4
8	5,5,40,239	n/a	n/a	2359.4	2675.5	150.8	51.5	23.9	343.5	222.4	55.7	0.0	21.6	29.4	50.9	0.0	33.6	33.3	51.3	0.0	0.0	0.0	11.7
9	5,5,40,247	n/a	n/a	2779.6	3315.4	189.1	55.2	6.6	340.2	271.4	56.1	0.0	32.7	31.6	53.9	1.9	289.6	144.8	54.6	0.0	1.2	3.8	24.1
10	5,5,40,250	n/a	n/a	1724.2	2113.6	219.9	58.0	4.7	518.4	410.5	57.3	0.0	4.6	6.8	55.1	0.0	8.3	10.8	55.9	0.0	0.0	0.0	9.7
11	5,5,40,256	n/a	n/a	3327.2	3831.0	210.6	59.9	104.5	757.9	399.2	59.1	2.6	64.9	52.8	57.6	23.9	307.3	156.0	58.9	0.0	0.0	0.0	14.8
12	5,5,40,264	n/a	n/a	2615.2	3010.5	217.4	61.2	62.1	464.9	292.9	64.6	0.0	27.1	29.2	60.3	0.0	129.3	163.8	60.8	0.0	0.0	0.0	13.1
13	8,8,60,328	n/a	n/a	1788.8	2183.7	182.5	89.8	12.9	426.6	370.3	91.6	22.1	606.8	405.3	92.3	0.0	178.1	172.2	61.3	0.0	0.0	0.0	22.3
14	8,8,60,370	n/a	n/a	5263.1	6144.6	443.3	108.9	1781.7	3441.9	924.6	104.9	1978.3	3072.9	706.9	110.2	1019.2	2641.6	858.3	109.4	0.0	1.5	6.4	42.2
15	10,10,80,518	n/a	n/a	12247.5	14265.1	919.1	197.3	7828.5	11064.3	1635.4	202.8	9015.9	10928.3	1004.5	198.8	6465.9	9917.7	1356.9	198.6	0.0	71.3	167.2	139.6

Notes: (1) n/a was due to LINGO running out of working memory to generate the model. (2) All customer orders in instances 1, 4, 8, 9, 11, 12, and 14 could use any transferring equipment. (3) Some customer orders in instances 2, 3, 5, 6, 7, 10, 13, and 15 could not use some of transferring equipment.

Table 7: The t-values of average total tardiness from the one-sided t-test.

Ins	Problem size (m, q, n, o)	t-value of avg total $\mu_{\text{DE}} - \mu_{\text{SSDE}}$	t-value of avg total $\mu_{\text{RAM-EPSDE}} - \mu_{\text{SSDE}}$
8	5-5-40-239	4.02	5.53
9	5-5-40-247	5.42	10.91
10	5-5-40-250	3.71	4.21
11	5-5-40-256	6.73	10.79
12	5-5-40-264	5.08	4.32
13	8-8-60-328	8.20	5.66
14	8-8-60-370	23.80	16.85
15	10-10-80-518	58.40	39.45

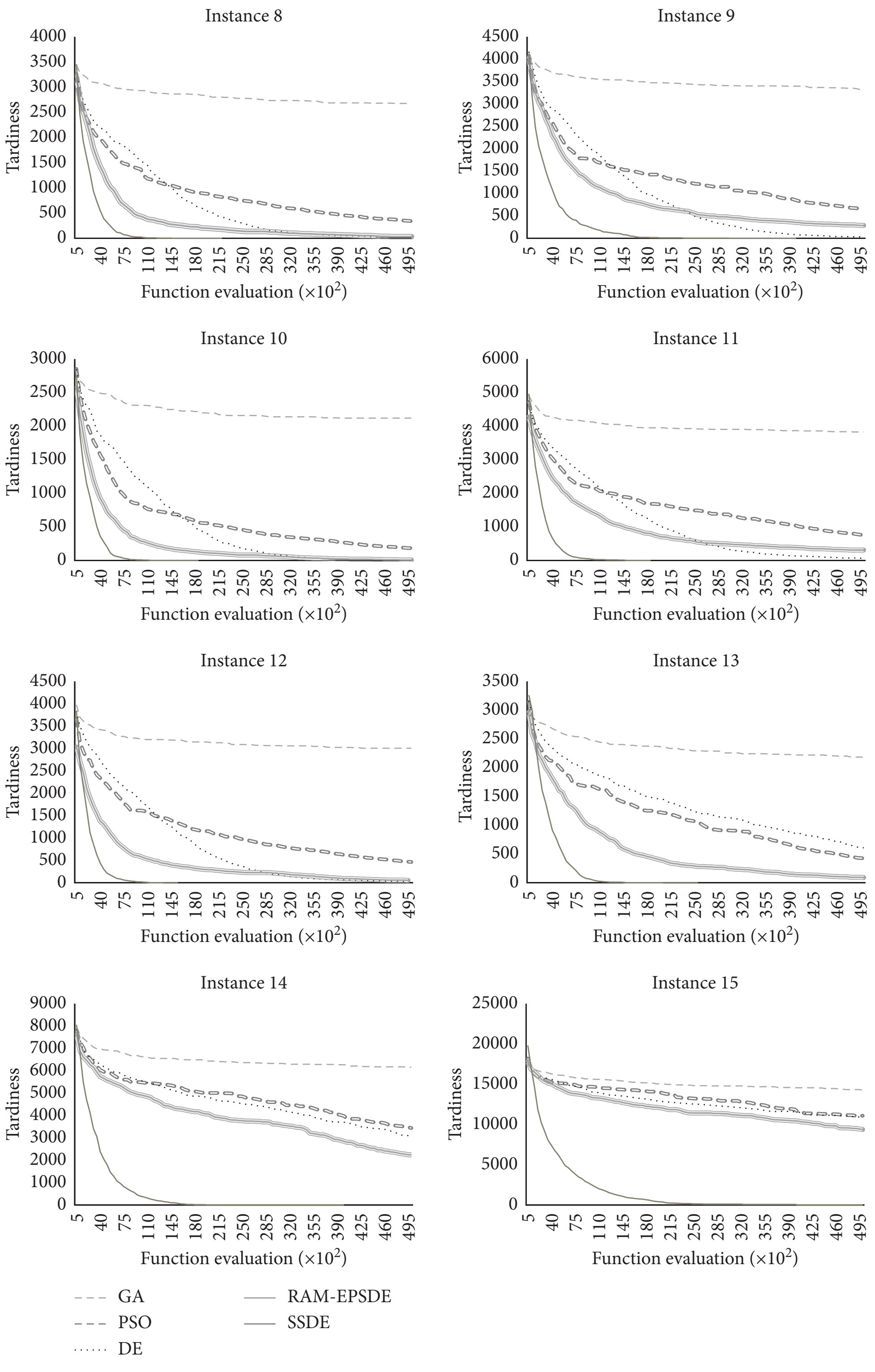

Figure 5: Comparison of convergence behaviour among algorithms.

8. Conclusion

A novel SSDE algorithm was proposed to handle the complexity of stochastic internal task scheduling problems in cross-dock terminals. The aim of this study was to assign workers and transfer equipment to operations and to sequence those operations to minimise total tardiness. The ability to adapt itself to execute the best strategy was the key to the success of SSDE. In SSDE, fitness values were considered to prioritise the capability of each strategy in the learning stage. Subsequently, the improvement probability was used in the strategy selection during the running stage.

In this study, a solution representation with encoding and decoding procedures was also developed to transform a random number into a scheduling solution. Then, an exact method using the LINGO optimisation solver and metaheuristics was applied to solve 15 instance problems using generated data based on a real-case scenario of a pharmaceutical distribution centre. The performance of the SSDE was compared with the results obtained from LINGO and other classical metaheuristics in terms of solution quality and computational time. Based on the experimental results, there was no difference in the solution quality when solving small problems. However, LINGO took a long time to solve small problems and suffered from insufficient working memory when attempting to solve large problems. In contrast, all metaheuristic algorithms required less computing time to provide solutions for all instances. Based on numerical results and convergence behaviour, SSDE significantly outperformed classical DE and other algorithms in terms of both solution quality and computational time. In addition, SSDE presented faster convergence than the other algorithms. Hence, SSDE is an alternative approach for solving stochastic internal task scheduling problems in cross-dock terminals.

Data Availability

Access to data is restricted due to third party rights.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was partially supported by Chiang Mai University (CMU), Thailand, and Prince of Songkla University (PSU), Thailand.

References

[1] A. Rijal, M. Bijvank, and R. de Koster, "Integrated scheduling and assignment of trucks at unit-load cross-dock terminals with mixed service mode dock doors," *European Journal of Operational Research*, vol. 278, no. 3, pp. 752–771, 2019.

[2] Z. Feifeng, Y. Pang, Y. Xu, and M. Liu, "Heuristic algorithms for truck scheduling of cross-docking operations in cold-chain logistics," *International Journal of Production Research*, vol. 59, no. 3, pp. 1–22, 2020.

[3] S. Amin and M. S. Sajadieh, "Truck scheduling in a multi-door cross-docking center with partial unloading–Reinforcement learning-based simulated annealing approaches," *Computers & Industrial Engineering*, vol. 139, p. 106134, 2020.

[4] C. Issi, Gustavo, R. Linfati, John, and W. Escobar, "Mathematical optimization model for truck scheduling in a distribution center with a mixed service-mode dock area," *Journal of Advanced Transportation*, vol. 2020, p. 13, Article ID 8813372, 2020.

[5] T. Chargui, A. Bekrar, M. Reghioui, and D. Trentesaux, "Scheduling trucks and storage operations in a multiple-door cross-docking terminal considering multiple storage zones," *International Journal of Production Research*, vol. 60, no. 4, pp. 1153–1177, 2020.

[6] M. Kłodawski and J. Żak, "Order picking area layout and its impact on the efficiency of order picking process," *Journal of Traffic and Logistics Engineering*, vol. 1, no. 1, pp. 41–45, 2013.

[7] D. Buakum and W. Wisittipanich, "Stochastic internal task scheduling in cross docking using chance-constrained programming," *International Journal of Management Science and Engineering Management*, vol. 15, no. 4, pp. 258–264, 2020.

[8] L. Yanzhi, A. Lim, and B. Rodrigues, "Crossdocking–JIT scheduling with time windows," *Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1342–51, 2004.

[9] M. Afshar-Bakeshloo, F. Jolai, M. Mazinani, and R. Tavakkoli-Moghaddam, "A satisfactory multi-agent single-machine considering a cross-docking terminal," *International Journal of System of Systems Engineering*, vol. 9, no. 4, pp. 307–330, 2019.

[10] H. Imen and M. Fadhel Tekaya, "A genetic algorithm to minimize the makespan in a two-machine cross-docking flow shop problem," *Journal of the Operations Research Society of China*, vol. 8, pp. 1–20, 2019.

[11] G. A. Álvarez-Pérez, J. L. González-Velarde, and J. W. Fowler, "Crossdocking-just in time scheduling: an alternative solution approach," *Journal of the Operational Research Society*, vol. 60, no. 4, pp. 554–564, 2009.

[12] D. Buakum and W. Wisittipanich, "A mathematical model for internal task scheduling in cross docking," in *Proceedings of the 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Macao, China, December 2019.

[13] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[14] G. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, no. 2, pp. 674–692, 2006.

[15] N. Damak, B. Jarboui, P. Siarry, and T. Loukil, "Differential evolution for solving multi-mode resource-constrained project scheduling problems," *Computers & Operations Research*, vol. 36, no. 9, pp. 2653–2659, 2009.

[16] W. Wisittipanich and V. Kachitvichyanukul, "Two enhanced differential evolution algorithms for job shop scheduling problems," *International Journal of Production Research*, vol. 50, no. 10, pp. 2757–2773, 2012.

[17] W. Wisittipanich and P. Hengmeechai, "A multi-objective differential evolution for Just-In-Time door assignment and truck scheduling in multi-door Cross docking problems," *Industrial Engineering and Management Systems*, vol. 14, no. 3, pp. 299–311, 2015.

[18] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and

evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, Portland, OR, USA, June 2004.

[19] M. Leon, N. Xiong, D. Molina, and F. Herrera, "A novel memetic framework for enhancing differential evolution algorithms via combination with alopex local search," *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 795–808, 2019.

[20] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[21] X. Gou, T. Huang, S. Yang, M. Su, and F. Zeng, "Optimized differential evolution algorithm for software testing," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 215–226, 2018.

[22] M. Pant, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, "Differential evolution: a review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, Article ID 103479, 2020.

[23] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[24] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[25] J. Brest and M. Sepesy Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[26] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proceedings of the 2013 IEEE congress on evolutionary computation*, Cancun, Mexico, June 2013.

[27] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proceedings of the 2014 IEEE congress on evolutionary computation (CEC)*, Beijing, China, June 2014.

[28] T. N. Huynh, D. T. Do, and J. Lee, "Q-Learning-based parameter control in differential evolution for structural optimization," *Applied Soft Computing*, vol. 107, Article ID 107464, 2021.

[29] K. Price, R. Storn, and J. Lampinen, "Differential evolution-a practical approach to global optimization," *Natural Computing*, vol. 141, 2005.

[30] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics: A Publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 41, no. 2, pp. 397–413, 2011.

[31] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," in *Proceedings of the International conference on swarm, evolutionary, and memetic computing*, Chennai, India, December 2010.

[32] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[33] D. Buakum and W. Wisittipanich, "Self-learning differential evolution algorithm for scheduling of internal tasks in cross-docking," *Soft Computing*, vol. 26, no. 21, pp. 11809–11826, 2022.

[34] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, September 2005.

[35] J. Brest and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Computing*, vol. 15, no. 11, pp. 2157–2174, 2011.

[36] Q.-K. Pan, P. N. Suganthan, L. Wang, L. Gao, and R. Mallipeddi, "A differential evolution algorithm with self-adapting strategy and control parameters," *Computers & Operations Research*, vol. 38, no. 1, pp. 394–408, 2011.

[37] W. Gong, A. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: an empirical study," *Information Sciences*, vol. 181, no. 24, pp. 5364–5386, 2011.

[38] Q. Fan and X. Yan, "Differential evolution algorithm with self-adaptive strategy and control parameters for P-xylene oxidation process optimization," *Soft Computing*, vol. 19, no. 5, pp. 1363–1391, 2015.

[39] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 219–232, 2016.

[40] Z. Zhao, J. Yang, Z. Hu, and H. Che, "A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric latin hypercube design for unconstrained optimization problems," *European Journal of Operational Research*, vol. 250, no. 1, pp. 30–45, 2016.

[41] Q. Fan and Y. Zhang, "Self-adaptive differential evolution algorithm with crossover strategies adaptation and its application in parameter estimation," *Chemometrics and Intelligent Laboratory Systems*, vol. 151, pp. 164–171, 2016.

[42] G. Wu, R. Mallipeddi, P. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Information Sciences*, vol. 329, pp. 329–345, 2016.

[43] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-based adaptive operator selection mechanism for differential evolution," *Information Sciences*, vol. 418-419, pp. 383–404, 2017.

[44] T. M. Moussa and A. A. Awotunde, "Self-adaptive differential evolution with a novel adaptation technique and its application to optimize ES-SAGD recovery process," *Computers & Chemical Engineering*, vol. 118, pp. 64–76, 2018.

[45] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, UK, July 2020.

[46] D. T. T. Do, S. Lee, and J. Lee, "A modified differential evolution algorithm for tensegrity structures," *Composite Structures*, vol. 158, pp. 11–19, 2016.

[47] H. Tang and J. Lee, “Adaptive initialization LSHADE algorithm enhanced with gradient-based repair for real-world constrained optimization,” *Knowledge-Based Systems*, vol. 246, Article ID 108696, 2022.
[48] Q. Fan, X. Yan, and Y. Zhang, “Auto-selection mechanism of differential evolution algorithm variants and its application,” *European Journal of Operational Research*, vol. 270, no. 2, pp. 636–653, 2018.
[49] Q. Fan, Y. Jin, W. Wang, and X. Yan, “A performance-driven multi-algorithm selection strategy for energy consumption optimization of sea-rail intermodal transportation,” *Swarm and Evolutionary Computation*, vol. 44, pp. 1–17, 2019.
[50] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT press, Cambridge, MA, USA, 1992.
[51] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS’95*, Nagoya, Japan, October 1995.
[52] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, and P. N. Suganthan, “Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization,” Technical Report D, Nanyang Technological University, Singapore, 2020.
[53] S. Mirjalili, “Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm,” *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.
[54] S. Dhargupta, M. Ghosh, S. Mirjalili, and R. Sarkar, “Selective opposition based grey wolf optimization,” *Expert Systems with Applications*, vol. 151, Article ID 113389, 2020.
[55] H. Yapici and N. Cetinkaya, “A new meta-heuristic optimizer: pathfinder algorithm,” *Applied Soft Computing*, vol. 78, pp. 545–568, 2019.
[56] A. Faramarzi, M. Heidarinejad, B. Stephens, and S. Mirjalili, “Equilibrium optimizer: a novel optimization algorithm,” *Knowledge-Based Systems*, vol. 191, Article ID 105190, 2020.
[57] Q. Fan, H. Haisong, C. Qipeng, Y. Liguo, Y. Kai, and H. Dong, “A modified self-adaptive marine predators algorithm: framework and engineering applications,” *Engineering with Computers*, vol. 38, pp. 1–26, 2021.