Metaheuristics for Warehouse Storage Location Assignment Problems

Warisa Wisittipanich[1*] and Chompoonoot Kasemset[2]

[1]Excellence Centre in Logistics and Supply Chain Management, Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai 50200, Thailand
[2]Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai 50200, Thailand

**Corresponding author. E-mail: warisa@eng.cmu.ac.th*

ABSTRACT

This study addressed warehouse storage location assignment problems (SLAP) to minimize total travelling distances in an order-picking process. The problem was formulated and presented as a mixed integer programming model. The LINGO optimization solver was then used to find solutions for a set of generated problems. The results showed that the LINGO optimization solver easily attained optimal solutions for small-sized problems; however, as the problem size increased, computational time increased rapidly. Eventually, when the problem size became very large, LINGO was unable to find solutions. Due to the competence limitations of the exact solution method, this research presented two effective metaheuristic approaches – Differential Evolution (DE) and Global Local and Near-Neighbor Particle Swarm Optimization (GLNPSO) – to solve SLAP. To illustrate the performance of the algorithms, the numerical results were evaluated and compared with a set of generated problems. The experimental results showed that, for small-sized problems, DE and GLNPSO found optimal solutions equal to those obtained from LINGO, with fast computing times. For medium-sized problems, DE and GLNPSO were not significantly different in terms of solution quality, but DE found solutions approximately twice as fast as GLNPSO. For large-sized problems, DE was significantly superior to GLNPSO in terms of both solution quality and computational times. The average DE solutions, obtained from five independent runs, were equal to or better than those obtained from GLNPSO in all large-sized instances. In addition, DE showed faster convergence behavior than GLNPSO, since it yielded better solutions while using a fewer number of function evaluations.

Keywords: Metaheuristics, Optimization, Warehouse, Storage location assignment

INTRODUCTION

In today's competitive environment, companies are challenge to enhance both customer satisfaction and profitability, by strengthening supply chain management. According to the principles of supply chain management, companies must aim to distribute products to customers within short response times, while holding inventory at a minimum throughout the chain. To cope with these challenges, well-organized warehouse management is important to overall supply chain efficiency.

A warehouse is a key module in logistics, and plays a vital role in controlling and reducing logistics costs. More importantly, warehouse management is the art of storing and moving inventory throughout the warehouse, so that the flow of products is optimized. Typically, operations in warehouse management consist of five basic functions: receiving, sorting, storing, order picking and delivering. Among these operations, product storing and order picking are among the most resource-consuming activities (Van Den Berg and Zijm, 1999). Storing is expensive due to inventory holding costs, and order picking is labor intensive (Van Belle et al., 2012). However, effective warehouse management can reduce storage costs by reducing inventory levels, while maintaining defined service levels. In addition, efficient allocation of storage locations can reduce storage times and order picking activities, lessen total travelling distances, and decrease congestion in product flow. In addition, it helps harmonize the different activities in a warehouse.

This study addressed storage location assignment problems (SLAP) in warehouse management in order to minimize the total travelling distances in an order-picking process. A storage location assignment problem (SLAP) is a problem of assigning goods to storage locations that aims to satisfy one or more objectives – i.e., space utilization, total transfer times, and total transportation distances. The storage management system can be categorized into three main policies: dedicated storage policy, random storage policy, and class-based storage policy. In dedicated storage policy, each stock-keeping unit (SKU) is assigned to a certain designed location. In random storage policy, any SKU can be stored at any storage location. In class-based storage policy, a group of storage locations is allocated to a class of SKUs and random storage is allowed within the group of storage locations.

Some storage location assignment problems can be easily solved to optimality by traditional exact methods. However, in real-world practice, when the size of a warehouse is very large and numerous product types are stored, the problem tends to get complicated. To deal with high-computational complexity problems, most recent research has shifted toward developing an alternative approximation approach, called metaheuristic algorithms. Although metaheuristic algorithms do not guarantee finding the optimal solution, they do provide high-quality or near-optimal solutions within acceptable computing times.

Van Den Berg and Zijm (1999) proposed a hierarchical decision problem for storage systems in the stage of designing, planning, and controlling. Roodbergen and De Koster (2001) presented heuristic methods for solving order picking and

routing problems in warehouses where two or more aisles exist and random storage was used. Peterson and Aase (2004) applied simulation techniques to compare the principles for picking, storage, and routing in a manual order picking system. Hsu et al. (2005) presented a batching approach based on metaheuristics, called a genetic algorithm (GA), to directly minimize total travel distance. Ho and Tseng (2006) solved the optimal batch-picking problem by a genetic algorithm in order to minimize total distance travelled. Chen and He (2008) presented a mathematical model for SLAP, with strategies for automatic storage systems, and implemented Particle Swarm Optimization (PSO) based on the Pareto concept to overcome big-sized warehouse assignment problems. Önüt et al. (2008) proposed a novel PSO to design a multiple-level warehouse shelf configuration with minimum annual carrying cost. Muppani and Adil (2008a) studied the integer programing of a storage system with class-based storage policy and developed a simulated annealing (SA) algorithm to form the classes and solve storage assignment problems. Then, Muppani and Adil (2008b) proposed a non-linear integer programming method for class-based storage systems that considered area decrease, handling costs, and storage area cost; they used a brand and bound (B&B) algorithm for solving the developed non-linear model. Sooksaksun et al. (2012) proposed a PSO algorithm for class-based storage policy to determine the aisle layout and dimension, while simultaneously assigning shelf spaces for storing the items based on item classes in order to minimize total distance in the warehouse. Ene and Öztürk (2012) developed a faster genetic algorithm to form optimal batches and optimal routes for an order-picking system in the automotive industry. The experimental result showed that the proposed algorithm provided a quick response in production orders in real-time application. Xie et al. (2014) proposed a method based on a genetic programming (GP) algorithm to solve real-world SLAP. The result showed that the proposed GP could obtain near-optimal solutions on the training data within a short computing time.

This research extended the work of Kasemset and Meesuk (2014), who proposed a mathematical model for assigning storage locations for a particular warehouse layout that considered three-axis travel distance. The storage management used in this study was based on a dedicated storage policy in which each stock-keeping unit (SKU) is assigned to a certain designed location. Since SLAP is one kind of assignment problem, and considered NP-hard, the computational effort, in general, required to find an optimal solution grows exponentially with the problem size. Therefore, metaheuristics are preferred to find a near optimal solution for larger, more practical problems. Particle Swarm Optimization (PSO) had been employed in some warehouse management research, especially in warehouse design problems, but limited research has applied PSO to SLAP compared to other heuristic approaches. Another recent metaheuristics approach, the Differential Evolution (DE) algorithm, has been successfully applied in several areas due to its effectiveness and simplicity. However, to the best of our knowledge, no DE algorithm has been developed to solve the storage location assignment problem. Thus, this research focused on implementing two effective metaheuristic approaches – Particle Swarm Optimization (PSO) and Differential Evolution (DE)

– to solve SLAP in order to minimize total travelling distance in an order-picking process.

MATERIALS AND METHODS

Mathematical model

The problem of storage location assignment is to allocate each product unit to an appropriate storage location in order to satisfy certain objectives subject to the problem constraints. In this research, the objective was to determine the storage location for each product unit in order to minimize total traveling distance along three-axis traveling distance: two horizontal axes and one vertical axis.

As mentioned earlier, the warehouse layout used in this study was based on the work of Kasemset and Meesuk (2014). The assumptions of the problem are as follows: (1) warehouse layout is assumed to be symmetric, (2) the input/output (I/O) point is located at one corner of the warehouse, (3) the number of storage blocks is limited, and (4) one storage block can be assigned to one product unit only. An example of warehouse layout with four columns, two racks, three levels, and two rows, viewed from the top and the side of the storage rack, is illustrated in Figure 1.

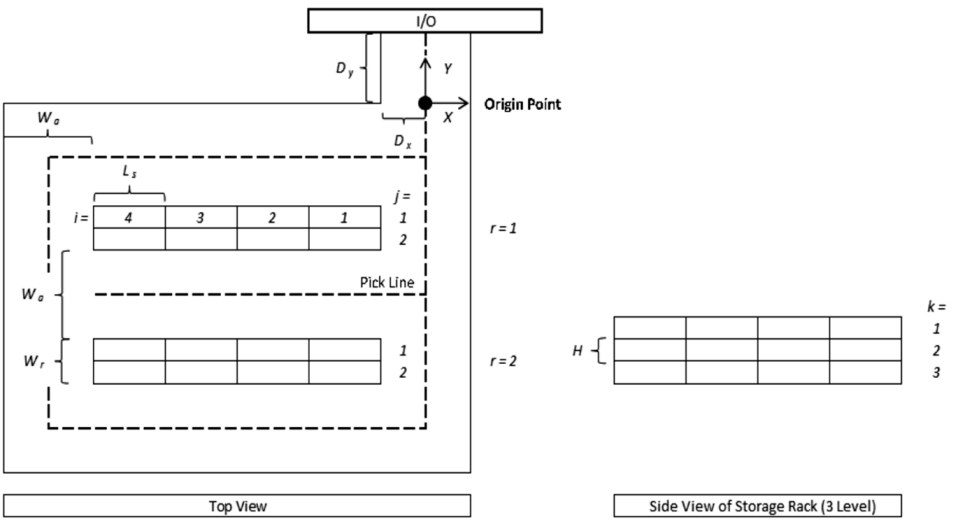

Figure 1. A warehouse layout with four columns, two racks, three levels, and two rows.

The notations and variables used in the model are as follows:

D_x : Distance from *I/O point to origin point along X-axis*
D_y : Distance from *I/O point to origin point along Y-axis*
W_a : Width of aisle
W_r : Width of storage *row* (*equal to two times storage block width*)
D_{ijkr} : *Total traveling distance of three-axis*
X : *Total traveling distance along X-axis*
Y : *Total traveling distance along Y-axis*
Z : *Total traveling distance along Z-axis*
L_s : *Length of storage block*
H : *Height of storage block*
T_p : Number of picking operations for product p (p = 1,2,3,…, n)
S_p : Number of storage blocks required for product p
I : Order of storage block position i (i = 1,2,3,…, m)
J : *Order of storage rack* j (j = 1,2,3,…, q)
K : *Order of level* k (k = 1,2,3,…, k)
R : *Order of row* r (r = 1,2,3,…, r)

Decision Variable:

$$X_{pijkr} = \begin{cases} 1, \text{ if product } p \text{ is assigned to storage position } i, \text{ rack } j, \text{ level } k, \\ \text{row r } 0, \text{ Otherwise} \end{cases}$$

The mathematical model of the problem is formulated as follows:
Objective function:
Minimization of total traveling distance:

$$f = \sum_{p=1}^{n}\sum_{i=1}^{m}\sum_{j=1}^{q}\sum_{k=1}^{k}\sum_{r=1}^{r}\left(\frac{Tp}{Sp}\right) X_{pijkr} D_{ijkr} \tag{1}$$

Subjected to constraints:

$$D_{ijkr} = X + Y + Z \tag{2}$$
$$X = D_x + (I - 0.5)L_s \tag{3}$$
$$Y = D_y + (0.5J\ W_a) + [(J\text{-}1)(W_r + 0.5W_a)] + [(W_r + W_a)(R\text{-}1)] \tag{4}$$
$$Z = H(K\text{-}1) \tag{5}$$
$$\sum_{p=1}^{n} X_{pijkr} \leq 1 \quad , \quad \forall\, i,j,k,r \tag{6}$$
$$\sum_{i=1}^{m}\sum_{j=1}^{q}\sum_{k=1}^{k}\sum_{r=1}^{r} X_{pijkr} = S_p \quad , \quad \forall\, p \tag{7}$$

Equation (1) expresses the objective function of the model – minimization of the total traveling distance in three-axis traveling distance. It is important to note that traveling distances are measured along the aisle centerline and the

centerline of storage blocks. The horizontal traveling distance is measured along the picker movement on the aisle floor, and the vertical distance is calculated by considering the height of the rack shelf. Equations (2), (3), (4), and (5) explain the calculation of three-axis total traveling distance. Constraint (6) guarantees that no more than one product is assigned to one storage unit and constraint (7) assures that all units of a particular product type are equal to the required storage blocks for that product.

Metaheuristics

As mentioned earlier, this research was motivated by the limitations of applying DE and PSO to solve storage location assignment problems (SLAP). Therefore, this study focused on applying two effective metaheuristics – Differential Evolution (DE) and a modified version of PSO, referred to as Global Local and Near-Neighbor Particle Swarm Optimization (GLNPSO) – to allocate products to the best storage locations in order to minimize the total travelling distance in an order-picking process. The search procedures of each algorithm are explained in the following sections.

Differential evolution (DE)

In 1995, Storn and Price proposed Differential Evolution (DE) as an evolutionary algorithm (EA) for global optimization over a continuous search space. DE has been applied recently to solve many combinatorial *NP*-hard problems, due to its advantage of relatively few control variables, but performing well in search ability and convergence.

As a population-based search method, DE begins with a randomly generated initial population of size *N*. Each population is represented as a *D*-dimensional vector, and each variable's value in the *D*-dimensional space is represented as a real number. The main idea that differentiates DE from other EAs is its mechanism for generating a new solution by particular mutation and crossover operations. At the initialization stage ($g = 0$), the j^{th} value of the i^{th} vector is generated as equation (8).

$$x_{j,i,0} = u_j.(b_{j,U} - b_{j,L}) + b_{j,L} \tag{8}$$

The lower bound, b_L, and upper bound, b_U, for the value in each dimension j^{th} (j=1,2,…,D) must be specified. A uniform random number, u_j, is in the range [0, 1]. DE performs the mutation operation by combining randomly selected vectors to produce a mutant vector. For each target vector, $X_{i,g}$ at generation g, the mutant vector, $V_{i,g}$, is generated as equation (9).

$$V_{i,g} = X_{r1,g} + F(X_{r2,g} - X_{r3,g}) \tag{9}$$

X_{r1},X_{r2}, and X_{r3} are vectors randomly selected from the current population. They are mutually exclusive and different from the target vector, $X_{i,g}$. F is a scale factor that controls the scale of the difference vector between X_{r2} and X_{r3}, added

to the base vector, X_{r1}. DE uses a crossover operator on X_{ig} and $V_{i,g}$ to produce the trial vector $Z_{i,g}$. In this study, the binomial crossover is employed and the trial vector is generated by equation (10).

$$z_{j,i,g} = \begin{cases} v_{j,i,g}\,, + \text{ if } u_j \leq C_r \text{ or } j = j_u \\ x_{j,i,g}\,, \text{ otherwish} \end{cases} \quad (10)$$

Where C_r is the crossover probability in the range [0, 1], and j_u is a random chosen index ($j_u \in \{1, 2, ...D\}$). C_r value controls the probability of selecting the value in each dimension from a mutant vector over its corresponding target vector. Then, the selection or replacement of an individual occurs only if the trial vector outperforms its corresponding vector. As a result, all individuals in the next generation are as good as or better than their counterparts in the current generation. The evolution of DE population continues through repeated cycles of three main operations – mutation, crossover, and selection – until stopping criteria are met.

The pseudo codes of the DE algorithm used in this study are shown in Algorithm DE.

Algorithm DE
1. Initialize Population, Pop
2. Evaluate objective function value of each vector, i, in Pop
3. While (Iteration <= MaxIteration)
For vector i =1 to Pop
Perform mutation operation
Perform binomial crossover operation
Perform selection operation
end For
Update best solution in Pop
End While

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), originally proposed by Kennedy and Eberhart in 1995, is a population-based random search approach that simulates the behavior of bird flocking or fish schooling. In the original model, each particle in a swarm learn and adapt its search behavior based on its own experience (cognitive term) and global experience (social term) during the search. The new solution of particle *i* is generated by the updated velocity (ω_{id}) and position (q_{id}). The standard PSO is formulated as equation (11) and (12).

$$\omega_{id}\,(t+1) = \omega_{id}\,(t) + c_p u\left(\psi^p_{id} - q_{id}\,(t)\right) + c_g u\left(\psi^g_{id} - q_{id}\,(t)\right) \quad (11)$$
$$q_{id}\,(t+1) = q_{id}\,(t) + \omega_{id}\,(t) \quad (12)$$

Where q_{id} : current position of dth dimension of ith particle
ω_{id} : velocity of dth dimension of ith particle
ψ^p_{id} : personal best position of dth dimension of ith particle
ψ^g_{id} : global best position of dth dimension of ith particle

c_p : weight of personal best position term
c_g : weight of global best position term
u : uniform random number in range [0,1]

One major drawback of the standard PSO is that the swarm tends to be frequently trapped in local optima and can no longer move. Thus, several variants of PSO have been proposed to remedy this common problem of premature convergence. This research implemented a modified version of PSO, referred to as GLNPSO (Global Local and Near-Neighbor Particle Swarm Optimization), to solve this problem. In 2005, Pongchairerks and Kachitvichyanukul introduced GLNPSO with multiple social learning terms. Instead of using only the global best solution as a social reference, GLNPSO incorporates the local best and near-neighbor best as additional social learning references. As a consequence, the diversification of particles is increased, and the chance of a solution getting trapped in local optima is reduced. In GLNPSO, position and velocity in the d^{th} dimension of particle i are updated by the following equations.

$$\omega_{id}(t+1) = w\omega_{id}(t) + c_p u\left(\psi^p_{id} - q_{id}(t)\right) + c_g u\left(\psi^g_{id} - q_{id}(t)\right) + c_l u\left(\psi^l_{id} - q_{id}(t)\right) + c_n u\left(\psi^u_{id} - q_{id}(t)\right) \quad (13)$$

$$q_{id}(t+1) = q_{id}(t) + \omega_{id}(t) \quad (14)$$

In this formula, w, introduced by Shi and Eberhart (1998) to improve the search ability, represents inertia weight. The value of w linearly decreases so that the swarm can search the whole space aggressively at the early stage and gradually reduces the search space at the later stage. ψ^p_{id}, ψ^q_d, ψ^l_{id}, ψ^n_{id}, represent the d^{th} dimension of the personal best, the global best, the local best, and the near neighbor best, respectively, and $q_{id}(t)$ is the current position of particle i at t^{th} generation. The local best position of particle i, $\psi^l_i = \{\psi^l_{i0}, \psi^l_{i1},\ldots,\psi^l_{iD}\}$ can be determined as the best particle among K neighboring particles. It is equivalent to dividing the whole swarm into multiple sub-swarms with population of size K, the local best particle is the best particle among the K neighbors, i.e., the particle gives the best fitness value in the sub-swarms.

The near neighbor best position of particle i, $\psi^n_i = \{\psi^n_{i0}, \psi^n_{i1},\ldots,\psi^n_{iD}\}$ represents the interaction between particles to achieve a better value of the objective function. The neighborhood is determined by the direct linear distance from the best particle. Each element of ψ^n_i is determined by equation (15).

$$\text{FDR} = \frac{Fitness(\Theta_i) - Fitness(\Psi_i)}{q_{id} - \psi_{id}} \quad (15)$$

The equations for updating position and inertia weight, w, in GLNPSO are the same as standard PSO formulas. Other parameters and working procedures are also similar to those in standard PSO.

The pseudo codes of the GLNPSO algorithm used in this study are shown in *Algorithm GLNPSO*.

Algorithm GLNPSO
1. Initialize Population, Pop
2. Evaluate objective function value of each particle, i, in Pop
3. Update best solution in Pop
4. While (Iteration <= MaxIteration)
For particle i =1 to Pop
Update velocity
Update position
Evaluate objective function value of particle i
end For
Update best solution in Pop
End While

Solution representation

Since PSO and DE were originally developed for continuous optimization domain, in order to apply PSO and DE for combinatorial optimization problems, each individual particle or vector must be transformed into a practical solution. Typically, solutions in the population-based random search method are represented as a string of several dimensions. In this research, the number of dimensions is set to be equal to the total number of storage locations or storage blocks. Each dimension index stands for the index of the storage location. Consider an example of a warehouse with two columns (i = 1, 2, 3), two racks (j = 1, 2), two levels (k = 1, 2), and two rows (r = 1, 2). The warehouse stores four product types: *A*, *B, C,* and *D*. The information for each product type is shown in Table 1. T_p is the number of picking operations for a product, S_p is the number of storage blocks required for a particular product type, and, thus, T_p/S_p determines the movement ratio of a product.

Table 1. Data information for product type *A, B, C,* and *D*.

	Tp	***Sp***	***Tp/Sp***
Product Type *A*	12	4	3.00
Product Type *B*	20	5	4.00
Product Type *C*	18	3	6.00
Product Type *D*	12	6	2.00

In this example, the number of dimensions is set to be equal to the number of storage units, which is 24. Figure 2 illustrates an encoding scheme of a random key representation, where each value in a dimension is initially generated with a uniform random number between 0 and 1.

Dimension d	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*
	0.52	0.83	0.69	0.63	0.49	0.58	0.82	0.71	0.97	0.79	0.78	0.27	0.21	0.70	0.80	0.39	0.93	0.81	0.84	0.66	0.96	0.46	0.90	0.34

Figure 2. An encoding scheme for SLAP.

Next, this research applied the permutation of *n*-repetition of *n* jobs (Bierwirth, 1995) with a sorting list rule to assign each product unit to a storage location. In Table 1, product type *C* has the maximum value of movement ratio (*Tp/Sp*), followed by product type *B*, product type *A*, and product type *D*, respectively. To decode these dimension values to a solution, all product units of type *C* are first allocated to the dimension with sorted values, then product units of type *B, A,* and *D* are allocated correspondingly. Since the number of storage blocks is 24, and the number of total product units is 18, six storage blocks are not assigned to store any product. These procedures result in a completed storage location assignment as shown in Figure 3.

Dimension d	*13*	*12*	*24*	*16*	*22*	*5*	*1*	*6*	*4*	*20*	*3*	*14*	*8*	*11*	*10*	*15*	*18*	*7*	*2*	*19*	*23*	*17*	*21*	*9*
	0.21	0.27	0.34	0.39	0.46	0.49	0.52	0.58	0.63	0.66	0.69	0.70	0.71	0.78	0.79	0.80	0.81	0.82	0.83	0.84	0.90	0.93	0.96	0.97
Product Type	*C*	*C*	*C*	*B*	*B*	*B*	*B*	*B*	*A*	*A*	*A*	*A*	*D*	*D*	*D*	*D*	*D*	*D*	-	-	-	-	-	-

Dimension d	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*
	0.52	0.83	0.69	0.63	0.49	0.58	0.82	0.71	0.97	0.79	0.78	0.27	0.21	0.70	0.80	0.39	0.93	0.81	0.84	0.66	0.96	0.46	0.90	0.34
Product Type	*B*	-	*A*	*A*	*B*	*B*	*D*	*D*	-	*D*	*D*	*C*	*C*	*A*	*D*	*B*	-	*D*	-	*A*	-	*B*	-	*C*
Storage location	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*

Figure 3. Decoding procedures for SLAP.

Parameter setting

The performance of metaheuristics is sensitive to not only their algorithms and solution representation, but also parameter setting. For a fair comparison between the performances of DE and GLNPSO, the numbers of function evaluations are set as a constant 50,000 to provide sufficient exploration in the search spaces. Corresponding to a fixed number of function evaluations, the population size and number of iterations are set as 50 and 100, respectively, for both GLNPSO and DE. The search procedures are terminated when the maximum number of iterations is met. Based on some preliminary experiments, other parameters in DE and GLNPSO are set as in Table 2 and 3, respectively.

Table 2. Parameter setting in DE algorithm.

DE parameters	Value
Scale factor, F	Uniformly randomize between 1 and 1.5
Crossover rate, C_r	0.5
Crossover type	Binomial crossover

Table 3. Parameter setting in GLNPSO.

GLNPSO parameters	Value
Inertia weight, w	Linearly decrease from 0.9 to 0.4
Constant acceleration	c_p, c_g, c_l, c_n = 1

For parameter setting in DE, some preliminary experiments determined that the value of the scale factor (F) as a uniformly randomized value between 1.5 and 2 provided generally good solution quality. In addition, the use of a binomial crossover yielded better results than an exponential crossover, and the crossover rate (C_r) was set as a constant value of 0.5. Thus, each dimension value in trial vectors was derived from a target vector and a mutant vector at equal probability. This crossover setting boosted the exploration of the search and decreased the chance of a solution getting trapped in a local optima.

In GLNPSO, an inertial weight (w) was set linearly decrease from 0.9 to 0.4, so that the swarm can search the whole space aggressively at the early stage and gradually reduce the search space at the later stage. The values of c_p, c_g, c_l, and c_n were all set to 1, to let the PSO population utilize knowledge from different learning terms equally.

Tested problems

Each tested problem was generated randomly based on a real-case warehouse. The problem was characterized by its problem size: (number of product types) x (total number of products) x (total number of storage locations). In this study, ten tested problems were generated and used in the numerical experiments.

Performance evaluations

The PSO and DE algorithms were implemented in C# programming language under the Microsoft Visual Studio version 12.0 development environment. The numerical experiments ran on the platform of Intel ® Core™ i7 CPU 1.9GHz with 4 GB RAM.

As mentioned earlier, the performances of DE and PSO were evaluated on a set of generated problems. This study focuses on the comparison of two meta-heuristics for SLAP with respect to one single objective – minimization of total travelling distances. The comparison is based on the use of the same number of function evaluations in both proposed DE and PSO. For each instance, the best, average, and standard deviations are determined from five independent runs. The effectiveness and performances of algorithms are evaluated in terms of solution quality, computational time, and convergence behavior.

RESULTS

Tables 4 shows the best solutions of total traveling distances derived from DE, GLNPSO, and optimal solutions obtained from the LINGO optimization program. The computational times were also reported.

Table 4. Experimental results of total traveling distances on a set of generated problems.

Instance	Problem size	LINGO		DE		GLNPSO	
		Optimal solution	Time (sec)	Best solution	Time (sec)	Best solution	Time (sec)
WH1	3x38x48	588.46	4	588.46	2.7	588.46	6.8
WH2	60x259x300	47273.04	8	47273.04	4.4	47273.04	9.7
WH3	135x840x1000	112705.19	240	112705.19	6.9	112705.19	16.0
WH4	270x1675x2000	204283.57	1200	204283.57	15.5	204283.57	35.7
WH5	500x2825x3360	290062.88	3600	290062.88	45.2	290062.88	70.4
WH6	600x2534x3360	N/A	-	373436.52	51.36	368487.50	87.50
WH7	500x2825x4992	N/A	-	271333.12	64.7	269784.52	300.9
WH8	800x3993x4992	N/A	-	520550.52	74.59	521045.50	378.75
WH9	500x2825x6300	N/A	-	259022.57	83.5	260944.12	590.6
WH10	1000x5040x6300	N/A	-	734698.08	102.14	734782.20	661.35

According to the results in Table 4, all solution methods – LINGO optimization program, DE, and GLNPSO – achieved the optimal solutions in small-sized problems with fast computing times. However, when the problem size increased, the computational times of LINGO increased rapidly, and when the problem size became very large, LINGO could not find optimal solutions within acceptable times. On the other hand, both GLNPSO and DE algorithms were able to provide solutions with relatively faster computing times.

The statistical comparison between DE and GLNPSO are provided in Table 5, with the best, average, and standard deviations of total traveling distances from five independent runs for each instance. A two-sample t-test was performed to statistically compare the results of two algorithms at the 95% confidence level. The average convergence graphs were also observed using the average fitness value versus number of iterations, as shown in Figure 4.

Table 5. Statistical comparison between GLNPSO and DE.

Instance	Problem size	DE			GLNPSO			*p*-Value
		Best	Avg.	SD	Best	Avg.	SD	
WH1	3x38x48	588.46	588.46	0	588.46	588.46	0	1
WH2	60x259x300	47273.04	47273.04	0	47273.04	47273.04	0	1
WH3	135x840x1000	112705.19	112705.19	0	112705.19	112705.19	0	1
WH4	270x1675x2000	204283.57	204283.57	0	204283.57	204283.57	0	1
WH5	500x2825x3360	290062.88	290703.96	822.42	290062.88	290944.90	909.29	0.706
WH6	600x2534x3360	373436.52	376414.85	2412.58	368487.50	379121.50	5958.29	0.438
WH7	500x2825x4992	271333.12	275007.92	2811.85	269784.52	277158.80	3859.55	0.398
WH8	800x3993x4992	520550.52	523911.60	2581.06	521045.50	531725.70	5830.73	0.058
WH9	500x2825x6300	259022.57	262702.86	1868.43	260944.12	268947.70	4232.90	0.043
WH10	1000x5040x6300	734698.08	740051.96	3207.73	734782.20	757305.80	11512.3	0.045

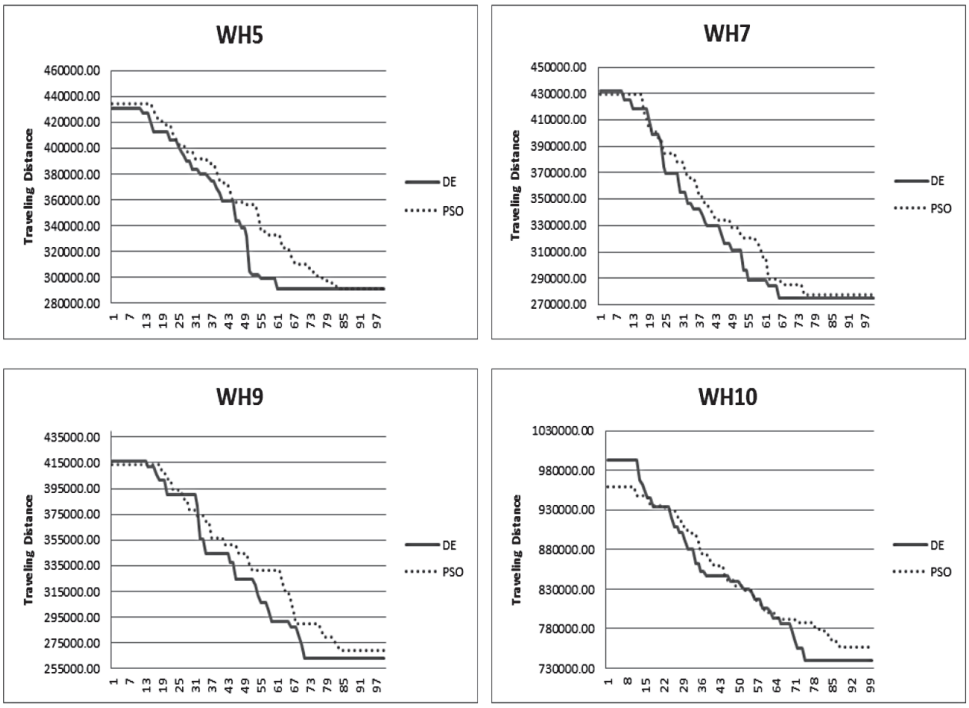

Figure 4. Comparison of convergence behavior between GLNPSO and DE.

According to the results from Table 5 and Figure 4, although the best solutions obtained from DE were inferior to those obtained from GLNPSO in two instances (WH6 and WH7), all of the average solutions obtained from the DE algorithm were equal to or better than those obtained from PSO, with relatively faster computing times. The statistical t-test indicated that the performances of DE and GLNPSO were not significantly different in small-sized to medium-sized problems. However, for large-sized problems, the DE algorithm significantly outperformed GLNPSO, since it generated relatively small values of best and average traveling distances compared to those obtained from GLNPSO, and the value of p was less than 0.05. In addition, the average convergence graphs clearly demonstrated the faster convergence behavior of DE compared to PSO.

DISCUSSION

This study found that DE and GLNPSO are effective solution techniques and can be used as alternative approaches for solving storage location assignment problems (SLAP). Both algorithms were able to find optimal solutions equal to those obtained by a LINGO optimization solver in small-sized problems. Although solutions from LINGO were guaranteed to be optimal, when the problem size increased, computing time increased rapidly; eventually, as the problem size became very large, a solution could not be found. However, for large-sized problems, DE and GLNPSO were capable of finding solutions with relatively faster computing times.

For a particular comparison between DE and GLNPSO, the numerical results demonstrated that, in general, DE outperformed GLNPSO in terms of solution quality, computational time, and convergence behavior. For small to medium-size problems, the performances of DE and GLNPSO were not significantly different in terms of solution quality, but DE found solutions approximately twice as fast as GLNPSO. The differentiation between DE and GLNPSO performances was clearly seen in large-sized problems. Under the same experimental conditions – encoding and decoding scheme, population size, and number of iterations – the DE algorithm was superior to GLNPSO in terms of solution quality and computing times. The average solutions, obtained from five independent runs, derived from the DE algorithm were equal to or better than those obtained from GLNPSO in all instances with relatively faster computing time. This demonstrated that the DE algorithm is more robust than PSO. In addition, DE showed faster convergence behavior than GLNPSO, since it yielded better solutions, while using a fewer number of function evaluations. The superiority of DE to PSO was also demonstrated in other research fields, such as numerical benchmark problems (see Wisittipanich and Kachitvichyanukul (2014)), production scheduling problems (see Wisittipanich and Kachitvichyanukul (2011, 2013, 2015), Chakaravarthy et al. (2013)), and vehicle routing problems (see Kunnapapdeelert and Kachitvichyanukul (2013), Liao et al. (2012)).

CONCLUSION

In conclusion, this research proposed DE and GLNPSO algorithms with a particular solution representation that were experimentally shown to be efficient for solving SLAP. In general, DE outperformed GLNPSO in terms of solution quality, computational time, and convergence behavior for all problem sizes. However, this study still has some limitations. The warehouse layout used in this study was based on a particular warehouse with a single I/O point, which may not represent other warehouse layout designs. Thus, the mathematical model needs to be modified for different warehouse configurations. Moreover, the total traveling distances used in this study were calculated from rectilinear distances along three-axes, and the velocity of the vehicle was assumed to be constant. Nevertheless, in some real-world situations, the velocities of the vehicle moving in horizontal and vertical axes might be unequal, and as a consequence, affect the total operating times in an order-picking process. Further studies in SLAP should consider these aforementioned issues to effectively deal with more real-world conditions in an order-picking process. Furthermore, it is recommended to further investigate other mixes of various strategies or other solution representations to increase the performance and robustness of the algorithms.

ACKNOWLEDGEMENTS

The authors would like to gratefully acknowledge the Excellence Centre in Logistics and Supply Chain Management (E-LSCM), Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, for supporting this research.

REFERENCES

Bierwirth, C. In: E. Pesch, and S. Vo (Eds.). 1995. A generalized permutation approach to job shop scheduling with genetic algorithms. OR-Spektrum. Special issue: Applied Local Search 17: 87-92. doi:10.1007/BF01719250

Chakaravarthy, G., S. Marimuthu, and A.N. Sait. 2013. Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling m-machine flow shops with lot streaming. Journal of Intelligent Manufacturing 24(1):175-191. doi:10.1007/s10845-011-0552-2

Chen, Y., and F. He. 2008. Research on particle swarm optimization in location assignment optimization. In Proceedings of the 7th World Congress on Intelligent Control and Automation. doi:10.1109/WCICA.2008.4594428

Ene, S., and N. Öztürk. 2012. Storage location assignment and order picking optimization in the automative industry. International Journal of Advanced Manufacturing Technology (60): 787-797. doi:10.1007/s00170-011-3593-y

Ho, Y.C., and Y.Y. Tseng. 2006. A study of order-batching methods of order-picking in a distribution centre with two cross-aisles. International Journal of Production Research 44(17): 3391-3417. doi:10.1080/00207540600558015

Hsu, C.M., K.Y. Chen, and M.Y. Chen. 2005. Batching orders in warehouse by minimizing travel distance with genetic algorithms. Computers in Industry 56: 169-178. doi:10.1016/j.compind.2004.06.001

Kasemset, C., and P. Meesuk. In: P. Golinska (ed).2014. Storage location assignment considering three-axis traveling distance: A mathematical model. Logistics Operations, Supply Chain Management and Sustainability, Eco-Production. Springer-Verlag, Cham, Heidelberg, New York, Dordrecht, London: 499-506. doi:10.1007/978-3-319-07287-6_35

Kennedy, J., and R.C. Eberhart. 1995. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ: 1942-1948.

Kunnapapdeelert, S., and V. Kachitvichyanukul. 2013. Differential evolution algorithm for generalized multi-depot vehicle routing problem with pickup and delivery requests. In: Proceedings of the Institute of Industrial and Engineer Asian Conference 2013, 749-756. 10.1007/978-981-4451-98-7_90. doi:10.1007/978-981-4451-98-7_90

Liao, T.W., P.J. Egbelu, and P.C. Chang. 2012. Two hybrid differential evolution algorithms for optimal inbound and outbound truck sequencing in cross docking operations. Applied Soft Computing 12: 3683-3697. doi:10.1016/j.asoc.2012.05.023

Muppani, V.R., and G.K. Adil. 2008a. Efficient formation of storage classes for warehouse storage location assignment. A simulated annealing approach. Omega 36: 609-618. doi:10.1016/j.omega.2007.01.006

Muppani, V.R., and G.K. Adil. 2008b. A branch and bound algorithm for class-based storage- location assignment. European Journal of Operational Research 189: 492-507. doi:10.1016/j.ejor.2007.05.050

Önüt, S., U.R. Tuzkaya, and B. Doğaç. 2008. A particle swarm optimization algorithm for the multi-level layout design problem. Computers and Industrial Engineering 54(4): 783-799. doi:10.1016/j.cie.2007.10.012

Peterson, C.G., and G. Aase. 2004. A comparison of picking, storage, and routing policies in manual order picking. International Journal of Production Economics 92: 11-19. doi:10.1016/j.ijpe.2003.09.006

Pongchairerks, P. and V. Kachitvichyanukul. 2005. Non-homogenous particle swarm optimization with multiple social structures. In: Proceedings of International Conference on Simulation and Modeling, Asian Institute of Technology, Bangkok, Thailand.

Roodbergen, K.J., and R. De Koster. 2001. Routing methods for warehouses with multiple cross aisle. International Journal of Production Research, 39: 1865-1883. doi:10.1080/00207540110028128

Shi, Y., and R. Eberhart.1998. A modified particle swarm optimizer. In Proceeding of the Evolutionary Computation World Congress on Computational Intelligence, Washington DC: 69-73.

Sooksaksun N., V. Kachitvichyanukul, and D. Gong. 2012. A class-based storage warehouse design using a particle swarm optimization algorithm. International Journal of Operational Research 13: 219-237. doi:10.1504/IJOR.2012.045188

Storn, R., and K. Price. 1995. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science. Berkeley, CA.

Van Den Berg, J.P., and W. Zijm. 1999. Models for warehouse management: Classification and examples. International Journal of Production Economics 59: 519-528. doi:10.1016/S0925-5273(98)00114-5

Van Belle, J., P. Valckenaers, and D. Cattrysse. 2012. Cross-docking: State of the art. Omega 40(6): 827-846. doi:10.1016/j.omega.2012.01.005

Wisittipanich W., and V. Kachitvichyanukul. 2011. Two enhanced differential evolution algorithms for job shop scheduling problems. International Journal of Production Research, DOI: 10.1080/00207543.2011.588972, August 2011. doi:10.1080/00207543.2011.588972

Wisittipanich, W., and V. Kachitvichyanukul. 2013. A Pareto-Based Differential Evolution Algorithm for Multi-Objective Job Shop Scheduling Problems. In: Proceedings of Institute of Industrial Engineers Asian Conference 2013, 1117-1125. DOI: 10.1007/978-981-4451-98-7_133. doi:10.1007/978-981-4451-98-7_133

Wisittipanich W., and V. Kachitvichyanukul. 2014a. Mutation strategies toward Pareto front for multi-objective differential evolution algorithm. International Journal of Operational Research 19(3): 315-337. doi:10.1504/IJOR.2014.059507

Wisittipanich W., and V. Kachitvichyanukul. 2015. Multiobjective Differential Evolution Algorithm for Flexible Job Shop Scheduling Problems. International Journal of Logistics and SCM Systems 8(1). March 2015: 1-10.

Xie, J., Y. Mei, A.T. Ernst, X. Li, and A. Song. 2014. A Genetic programming-based hyper-heuristic approach for storage location assignment problem. IEEE Congress on Evolutionary Computation (CEC), July 6-11, 2014, Beijing, China. doi:10.1109/CEC.2014.6900604

none