Differential Evolution Algorithm for Storage Location Assignment Problem

Warisa Wisittipanich* and Pongsakorn Meesuk

Industrial Engineering Department, Faculty of Engineering,
Chiang Mai University 239 Huay Kaew Road, Muang District, Chiang Mai, Thailand, 50200
warisa.o@gmail.com, enjoyneering@hotmail.com

Abstract. This study addresses warehouse storage location assignment problems (SLAP) where the traveling distance in an order-picking process is considered with three-axis traveling distance; two-horizontal and one-vertical distance. A mathematical model of the problem is first presented, then LINGO is used to find optimal solutions for a set of generated problems. However, as the problem size increases, computing time increases rapidly, and eventually the solution could not be found when the problem size is very large. Thus, this study presents an application of Differential Evolution (DE) algorithm to solve SLAP. The performance of proposed DE is evaluated on a set of generated problems, and the experimental results shows that the algorithm is able to provide good solutions especially for the large-size problems with relatively shorter computing time.

1 Introduction

Warehouse management is an art of storage and movement of inventory throughout the warehouse. Typical operations in warehouse management are comprised of receiving, storing, picking and delivering. Among these operations, product storage and retrieval are considered as one of the most critical resource-consuming activities [1]. Storage location assignment problem (SLAP) in warehouse management is a problem of assigning goods to storage locations that aims to satisfy one or more objectives i.e. space utilization, total transfer time, total transportation distance.

The storage management system can be classified into three main policies; dedicates storage policy, random storage policy, and class based storage policy. Brynzer and Johansson [2] stated the different between dedicated storage policy in which each stock-keeping unit (SKU) has a set of certain designed location, random storage policy in which any SKU can occupy any storage location, and class-based storage policy in which a group of storage location is allocated to a class of SKUs and random storage is allowed within the group of storage locations.

Many researchers have proposed several solution techniques to solve SLAP. Heragu et al. [3] considered a warehouse with five functional areas and proposed a

* Corresponding author.

M. Gen et al. (eds.), *Industrial Engineering, Management Science and Applications 2015,*
Lecture Notes in Electrical Engineering 349, DOI: 10.1007/978-3-662-47200-2_29

heuristic algorithm to determine SKUs' s allocation to different storage areas and the size of each functional area in order to minimize the total cost of material management. Chen and He [4] presented warehouse assignment strategies for storage systems with automation and developed a mathematical model for warehouse assignment optimization. Then, they applied particle swarm optimization with Pareto concept to overcome big-size problems. Muppani and Adil [5] studied the integer programing of a storage system with classed-based storage policy, and developed the simulated annealing (SA) algorithm to solve storage assignment and forming the classes. Next, they [6] proposed a non-linear integer programing for class-based storage system considering area decrease, handling costs and storage area cost, and used the brand and bound (B&B) algorithm for solving the developed nonlinear model. Hsu, Chen and Chen [7] presented a batching approach based on genetic algorithm (GA), which directly minimize the total travel distance. Roodbergen and De Koster [8] proposed heuristic methods for solving order picking routing problem in warehouses where two or more aisles exist and random storage is used.

This paper presents an implementation of a novel evolutionary algorithm called differential evolution (DE) to solve large-scale storage location assignment problems (SLAP) considering three-axis traveling distance. The storage management used in this study is based on a dedicated storage policy. The remainder of this paper is organized as follows. The problem description and model specification of SLAP are provided in section 2. Section 3 describes the proposed DE algorithm and its application to the problem. Experimental results are reported in section 4. Finally, conclusion and further research are provided in section 5.

2 Problem Description

The classical SLAP is to assign each product to a storage location in order to obtain certain objectives subjected to the constraints. In this paper, the objective of the problem is to minimize the total traveling distance along three-axis traveling distance; two horizontal axis and one vertical axis.

In this study, the warehouse layout used is based on the work in [9]. Warehouse layout is assumed to be symmetric. Therefore, the width of all storage locations are the same. The input/output (I/O) point is located at one corner of the warehouse. The number of storage blocks is limited and one storage block can be assigned for one product only. Fig.1 illustrates an example of a warehouse layout with four columns, two racks, three levels, and two rows, viewed from the top and the side of storage rack.

The notation and variable used in this model are listed 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 of storage block width)
L_s : Length of storage block

H : Height of storage block
T_p : Number of picking product p (p = 1,2,3,…, n)
S_p : Number of storage block 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)

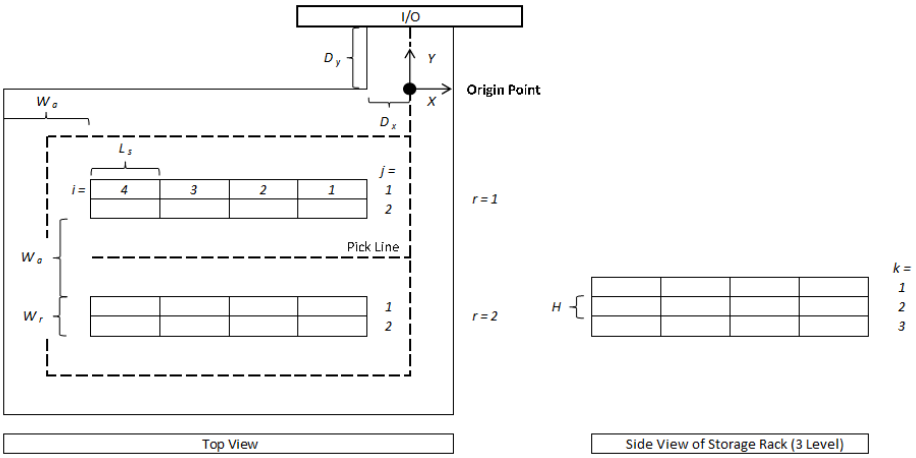

Fig. 1. A warehouse layout with four columns, two racks, three levels, and two rows

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:
Minimization of Total traveling distance:

$$f_2 = \sum_{p=1}^{n} \sum_{i=1}^{m} \sum_{j=1}^{q} \sum_{k=1}^{k} \sum_{r=1}^{r} \left(\frac{Tp}{Sp}\right) X_{pijkr} D_{ijkr} \tag{1}$$

Subjected to constraints:

$$D_{ijkr} = TD_x + TD_y + TD_z \tag{2}$$
$$TD_x = D_x + (I - 0.5)L_s \tag{3}$$
$$TD_y = D_y + (0.5J\ W_a) + [(J\text{-}1)(W_r + 0.5W_a)] + [(W_r + W_a)(R\text{-}1)] \tag{4}$$
$$TD_z = H(K\text{-}1) \tag{5}$$

$$\sum_{p=1}^{n} X_{pijkr} \leq 1 \qquad \forall\, i,j,k,r \tag{6}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{q}\sum_{k=1}^{k}\sum_{r=1}^{r} X_{pijkr} = S_p \qquad \forall\, p \tag{7}$$

The objective function of the model is expressed in equation (1); minimization of the total traveling distance. It is noted that traveling distances are measured along the aisle centerline and centerline of storage block, and horizontal traveling distance considered by the picker moved along the aisle floor and vertical distance considered by height of rack shelf. Equation (2) to (5) illustrates the calculation of three-axis traveling distance. Equation (6) ensure that no more than one product is assign to one storage unit. Equation (7) is to guarantee that each product stored in the storage units as equal to the number of storage needed.

3 Adaptation of Differential Evolution Algorithm

Differential Evolution (DE), proposed by Storn and Price in 1995 [10], is one of the latest Evolutionary Algorithms (EAs) for global optimization over continuous search space. Due to its advantage of relatively few control variables but performing well in search ability and convergence, DE has been recently applied to solve many combinatorial *NP*-hard problems.

3.1 Differential Evolution Algorithm

As a population-based search method, DE begin with randomly generate 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 the real number. The main idea which makes DE different from other EAs is its mechanism for generating a new solution by mutation and crossover operation. At 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 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 equation (9).

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

It is noted that X_{r1}, X_{r2}, and X_{r3} are vectors randomly chosen from the current population. They are mutually exclusive and different from the target vector, $X_{i,g}$. F is a scale factor which controls the scale of the difference vector between X_{r2}, and X_{r3} , added to the base vector, X_{r1} . DE applies crossover operator on $X_{i,g}$ and $V_{i,g}$ to generate the trial vector $Z_{i,g}$. In the classic DE, the binomial crossover is employed and the trial vector is generated by equation (10).

$$z_{j,i,g} = \begin{cases} v_{j,i,g} , & \text{if } u_j \leq C_r \text{ or } j = j_u \\ x_{j,i,g} , & \text{otherwise} \end{cases} \tag{10}$$

Where C_r is crossover probability in the range [0, 1], and j_u is a random chosen index ($j_u \in \{1, 2, \ldots D\}$). 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 criterion are met.

3.2 Solution Mapping to SLAP

In this study, a solution of the problem were represented using a DE vector with dimensions equal to the total number of storage blocks. Consider an example of a warehouse with three columns (i = 1, 2, 3), two racks (j = 1, 2), two levels (k = 1, 2), and one rows (r = 1). The warehouse store two product types; A and B. The value of *Tp* and *Sp* value each product type is shown in Table 1.

Table 1. Data for product type A and product type B

	Tp	*Sp*	*Tp/Sp*
Product Type A	15	4	3.75
Product Type B	20	6	3.33

The number of vector dimension is equal the number of storage units which is 12. Fig.2. illustrates a random key representation encoding scheme where each value in a vector dimension is initially generated with a uniform random number in range [0, 1].

Dimension d	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
	0.03	0.55	0.62	0.48	0.86	0.25	0.19	0.97	0.46	0.81	0.35	0.23

Fig. 2. Random key representation encoding scheme

Next, this study adopts the permutation of n-repetition of n jobs [11] with a sorting list rule to determine the assignment of each product unit to a storage location. According to the data in Table 1, product type A has a higher value of movement ration (*Tp*/*Sp*) than product type B. Therefore product type A are first allocated to the dimension with sorted values until the last unit of product type A has been assigned, and the product B unit are allocated next. Since the number of storage blocks is 12 and the number of total product units is 10, there are two storage blocks that are not assigned to store any product. The advantage of this approach is that it always provides a feasible storage allocation. This procedure results in completed storage location assignment as shown in Fig. 3.

Dimension d	*1*	*7*	*12*	*6*	*11*	*9*	*4*	*2*	*3*	*10*	*5*	*8*
	0.03	0.19	0.23	0.25	0.35	0.46	0.48	0.55	0.62	0.81	0.86	0.97
Product Type	*A*	*A*	*A*	*A*	*B*	*B*	*B*	*B*	*B*	*B*	-	-

Dimension d	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
	0.03	0.55	0.62	0.48	0.86	0.25	0.19	0.97	0.46	0.81	0.35	0.23
Product Type	*A*	*B*	*B*	*B*	-	*A*	*A*	-	*B*	*B*	*B*	*A*
Storage location	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*

Fig. 3. Decoding scheme for SLAP

4 Computational Experiments

4.1 Parameter Setting

In this study, the DE population size (*N*) and number of iterations are set as 10 and 20 respectively to provide an adequate number of function evaluations used in the search procedures. After some preliminary experiments, the value of *F* is set to be uniformly randomized between 1 and 1.5 to retain population diversity throughout the search process. Based on some preliminary experiments, the use of binomial crossover yields better results than exponential crossover in SLAP. Thus, binomial crossover operation is used in this experiment with constant crossover rate (*Cr*) at 0.5.

4.2 Experimental Results

The performance of proposed DE algorithm is evaluated using seven generated data sets. Each instance is characterized by problem size: (number of product types) x (total number of products) x (total number of slots). Tables 2 shows the comparison of the traveling distance obtained by the proposed DE and those obtained by LINGO optimization software. It is noted the computational time of the proposed DE is also reported and the best result of the proposed DE for each instance is obtained from 5 independent runs.

Table 2. Comparison of total traveling distances on a set of generate instances

Instance	Problem Size	LINGO	Time (sec.)	Proposed DE	Time (sec.)
WH1	3x38x48	588.46	4	588.46	2.7
WH2	60x259x300	47273.04	8	47273.04	4.4
WH3	135x840x1000	112705.19	240	112705.19	6.9
WH4	270x1675x2000	204283.57	1200	204283.57	15.5
WH5	500x2825x3360	290062.88	3600	290062.88	45.2
WH6	500x2825x4992	-	-	271333.12	64.7
WH7	500x2825x6300	-	-	259022.57	83.5

According to the results from Table 2, it can be easily seen that the proposed DE is an effective solution technique to SLAP in this study. DE is able to find optimal solutions equal to those obtained by LINGO in small-size problem. Although solutions from LINGO is guaranteed to be optimal, when the problem size increases, computing time increases rapidly, and eventually the solution could not be found when the problem size is very large. On the other hand, for large-size problems, DE is able to obtain solution with relatively faster computing time.

5 Conclusion

This paper presents an implementation of differential evolution (DE) algorithm for solving storage location assignment problem (SLAP). The proposed DE employs the classic DE mutation scheme with binomial crossover operation. The random key representation and permutation of n-job repetition is applied to assign products to storage locations. The performance of proposed method is evaluated on a set of generated instances and compared with results from LINGO optimization program.

The experimental results indicates that DE can be used as efficient alternative approach for solving SLAP as it yields competitive solutions in term of quality and computing time especially for the large size problems. The ongoing researches are under investigation to improve DE performance and apply DE to deal with other aspects of combinatorial optimization problems.

References

1. Van Den Berg, J., Zijm, W.: Models for warehouse management: Classification and examples. Int. J. Prod. Econ. 59(1), 519–528 (1999)
2. Brynzer, H., Johansson, M.I.: Storage location assignment: using the product structure to reduce order picking times. Int. J. Oper. Res. 47, 595–603 (1996)
3. Heragu, S.S., Du, L., Mantel, R.J., Schuur, P.C.: Mathematical model for warehouse design and product allocation. Int. J. Pro. Res. 43(2), 327–338 (2005)
4. Chen, Y., He, F.: Research on particle swarm optimization in location assignment optimization. In: Proceedings of the 7th World Congress on Intelligent Control and Automation (2008)

5. Muppani, V.R., Adil, G.: Efficient formation of storage classes for warehouse storage location assignment: A simulated annealing approach. Int. J. Manag. Sci. 36, 609–618 (2008)
6. Muppani, V.R., Adil, G.K.: A branch and bound algorithm for class-based storage-location assignment. Eur. J. Oper. Res. 189, 492–507 (2008)
7. Hsu, C.M., Chen, K.Y., Chen, M.Y.: Batching orders in warehouse by minimizing travel distance with genetic algorithms. Comput. Ind. 56, 169–178 (2005)
8. Roodbergen, K.J., De Koster, R.: Routing methods for warehouses with multiple cross aisle. Int. J. Prod. Res. 39(9), 1865–1883 (2001)
9. Kasemset, C., Meesuk, P.: Storage location assignment considering three-axis traveling distance: A mathematical model. In: Golinska, P. (ed.) Logistics Operations, Supply Chain Management and Sustainability, EcoProduction, pp. 499–506. Springer, Heidelberg (2014)
10. Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science, Berkeley, CA (1995)
11. Bierwirth, C.: A generalized permutation approach to job shop scheduling with genetic algorithms. In: Pesch, E., Vo, S. (eds.) OR-Spektrum. Special issue: Applied Local Search, vol. 17(213), pp. 87–92 (1995)