Applying Particle Swarm Optimization to Schedule Order Picking Routes in a Distribution Center

LING-FENG HSIEH[*], CHAO-JUNG HUANG AND CHIEN-LIN HUANG

Department of Technology Management, Chung Hua University, Taiwan

ABSTRACT

The performance of a distribution center is typically judged on throughput-based criteria. Order picking consumes 30% to 40% of operation time in a typical distribution center. To effectively execute an order picking operation in a distribution center depends upon coordinating the formulation of the storage strategy, order processing, and planning the order picking route. However, it is usually too complicated to employ traditional optimization methods, such as linear programming, to solve this kind of problem. As a result, we applied the Particle Swarm Optimization (PSO) Algorithm to schedule order picking routes. PSO is one of the latest swarm intelligence algorithms; consequently, when compared to previous sophisticated algorithms such as genetic algorithms and simulated annealing, the study of its properties and applications is still in its infancy. This research considers the convergence rate, the convergent reliability (i.e., solution precision), and the performance test function (i.e., fitness function) in scheduling order picking routes. We apply genetic algorithms to determine the initial solution in order to locate the optimal solution faster by PSO. This paper also compares the effects of different parameters on particle swarm optimization. In order to verify the result, we also made a comparison with the Ant system in finding the optimal solution in order route planning. Overall, the research result will enhance the system of order picking in distribution centers and improve the efficiency of order picking operations.

Key words: picking routing, particle swarm optimization.

1. INTRODUCTION

The order picking operation is an important but multifarious process in the internal processes of a distribution center. In the past customer orders were less in quantity and categories in comparison with the orders now, where they are still small in quantity, but with many more categories. How to fulfill orders within the set time frame has a direct impact on the operational cost and service quality of a distribution center. Additionally, looking from a labor demand perspective, a distribution center is a highly labor-intensive organization where more than 50% of the workers are involved in order picking and related work (Dong & Chen, 1995). The current issue is that when there is a vast variety of products, adjusting the internal process will inevitably improve company cost.

Classifying storage, order batching and route planning of order picking will inevitably reduce cost, improve production efficiency, and increase space utilization. Numerous studies have focused on the efficiency of the order picking operations in a distribution center. By storing these frequently ordered items together, the company can reduce the picking time and shorten the route distance. This paper attempts to apply PSO to route planning. The main characteristics of PSO, speed, position and fitness functions, are compared with previous studies of the Ant system to find the optimal solution in route planning. The overall research

[*] Corresponding author. E-mail: lfhsieh@chu.edu.tw

result will enhance the best route planning of order picking systems in distribution centers in order to improve the efficiency of the order picking system.

The purpose of this research includes:

(1) Outlining the main characteristics of PSO, that is, its speed, position and fitness functions, and applying them to elucidate the optimal solution in route planning.
(2) Using genetic algorithms to define an initial solution and then determining a faster way to find the optimal solution by applying PSO.
(3) Observing the significant PSO parameters applied to the picking route and picking time to find the setting of the optimal parameter in the picking system.
(4) Verifying the efficiency of PSO when applied to the problems of distribution center order picking systems and comparing its result with those of the Ant system (Huang, 2004) and other heuristic algorithms.

2. LITERATURE REVIEW

This paper focuses on the discussion and method of applying particle swarm optimization to plan picking routing in warehouse systems, with specific focus on "route planning of the order picking system" and "particle swarm optimization."

2.1 Route Planning of Order Picking System

The purpose of route planning is to minimize unnecessary picking distances to achieve the shortest picking route in an efficient manner. Roodbergen & Koster (2001a) evaluated different picking strategies with different numbers of aisles under a cross aisle warehousing system. Methods of calculating the shortest route include: S-shape Heuristic, Largest Gap Heuristic, Aisle-by-Aisle Heuristic, Optimal Algorithm, Combined Heuristic and Combine$^+$ Heuristic. The main principle with the Combined Heuristic is to allow the picking personnel to start from the furthest sub-aisle and move to the next area once all items are picked. Thus, the storage position of each item and the direction of the picking route are considered to provide an improved strategy, but a gap still exists with a realistic optimum route. Some designers will specify some primary characteristics of the system, such as the physical configuration (racks, aisles, picking vehicles, etc.) and the order picking policy. Stern (1986), Vickson and Fujimoto (1996) have addressed storage schemes for non-anticipatory server location for carousel storage rack. Bozer and White (1996) extended their work to design and evaluate the performance of end-of-aisle order-picking systems with multiple pick positions per aisle and multiple aisles per picker. Roodbergen and Koster (2001b) considered a parallel aisle warehouse, where order pickers can change aisles at the ends of every aisle and also change at a cross aisle halfway along the aisles. Later Kevin, Russell and Joseph (2006), suggested that when the system is busier and pick density is high, congestion is less of a problem and workers are more productive.

Hsieh and Lin (2005) developed a heuristic method for the picking-route strategy. The cross strategy considers improvements of picking methods within an area and between areas. There are two main improvements: firstly, start from the left sub-aisle of the picking route and move to the furthest area away from the loading zone. Secondly, when the picking process starts from the area closest to the loading zone, the picking route will start from right to left. This modified strategy is known as the Cross$^+$ Strategy. This has improved the Cross$^+$ Strategy proposed by Roodbergen and Koster (2001a) in that it effectively improves the picking route and is more suitable for a cross aisle designed warehouse. An Ant system developed by Huang (2004) redesigns the storage areas and compares with Cross+ Strategy to achieve a systematic analysis. It takes into consideration the size, storage position and the variety of items on the order-form to minimize the picking distance and ultimately achieve efficiency of the picking operation.

2.2 Particle Swarm Optimization

However, as a newly developed optimal method, the Particle Swarm Optimization algorithm is still in its developmental infancy and further studies are necessary. Particle Swarm Optimization (PSO) by Kennedy and Eberhart (1995) has several different versions: Kennedy and Eberhart (1995, 1997), Parsopoulos and Vrahatis (2002), Salman, Ahmad and Al-Madani (2002), Shigenori, Takamu, Toshiku and Yoshikazu (2003), Yoshida, Kawata, Fukuyama and Nakanishi (1999) and research papers which investigate convergence with an optimal parameter setting by Clerc and Kennedy (2002) and Trelea (2003). From Fan's (2002) research paper, we find that the modified PSO algorithm has a higher convergence rate and convergent reliability, and it is closer to the global optimum than the original PSO algorithm under four performance test functions. PSO is a population based stochastic optimization technique inspired by the social behavior of bird flocking or fish schooling. If one sees a desirable path to go (e.g., for food, protection, etc.), the rest of the swarm will be able to follow quickly even if they are on the opposite side of the swarm.

Bird flocking optimizes a certain objective function. Each agent knows its best value so far, called "pbest," which contains the information of position and velocities. This information is the analogy of personal experience of each agent. Moreover, each agent knows the best value so far, in the group "gbest" among pbests. This information is the analogy of knowledge, concerning how the other neighboring agents have performed. Each agent tries to modify its position by considering the current position, the current velocities, the individual intelligence (pbest), and the group intelligence (gbest).

PSO is initialized with a group of uniformly distributed random particles. Each particle searches for optima by updating generations. From the left side of Figure 1, setting T as a target and A~E as searching particles, any particle in the population that is closest to T is called gbest. Assume that after three iterations (see right side of Figure 1), the best fitness value is at E_4. This value is called gbest. This fitness value is also stored. From Figure 1, particle A is positioned at A_4, its recorded best fitness value is at A_3, and a particle takes part of the population as its topological neighbors. This paper applies these characteristics of PSO to the order picking system to improve efficiency.

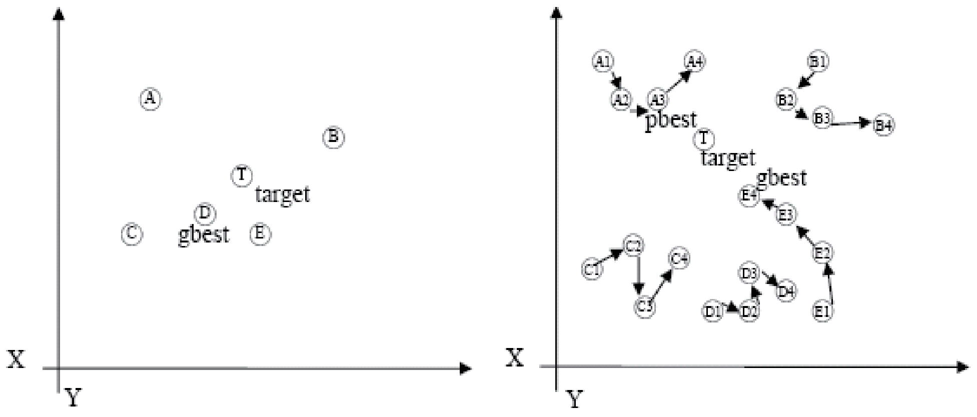


Figure 1. Scattered particle. PSO is initialized with a group of uniformly distributed random particles. Each particle searches for optima by updating generations.

PSO searches optimal solutions by individual and group experiences; however, the solution of the optimization problem may not come from previous solutions. Certain parameters need to be adjusted and random variables will be put in to distort the optimum solution (Eberhart & Shi, 2001). An advantage of PSO is that these particles remember the best position that they have seen. Members of a swarm communicate better positions to each other and based on this they can adjust their own position and velocity. Schutte and Groenwold (2005) proposed that each agent's search velocity changes as a random function of the distance between a point and a local best, and the distance between the point and the global best. PSO simulates the behavior of birds flocking. Let's suppose the following scenario: a group of birds are randomly searching for food in an area. There is only one piece of food in the area being searched. Not all the birds know where the food is. But all birds know how far to go to search for the food. So what is the best strategy to find the food? The effective one is to follow the bird which is nearest to the food (Hu, 2002).

Each individual in the PSO algorithm is called a "particle." Each particle is subject to a movement in a multidimensional space which it remembers. Particles have memory, and thus would retain part of their previous state. While there are no restrictions for particles to know the positions of other particles in the multidimensional spaces, they can still remember the best positions they have ever had. Each particle's movement is the composite of an initial random velocity, two randomly weighted influences: individuality (the tendency to return to the particle's best previous position), and sociality (the tendency to move towards the neighborhood's best previous position). All of the particles have fitness values which are evaluated by the fitness function to be optimized.

Each particle has two main characteristics: velocity and position. After finding the two best values, the particle updates its velocity and position with the following equations (1) and (2). Particles have the following parameters (Kennedy & Eberhart, 1995):

(a) i = 1, 2, …, PS represents particles within the population size which have different serial numbers.
(b) d = 1, 2, …, D represents the dimensions which each particle is composed of.
(c) The position of the ith particle is: $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$.
(d) The best position of every particle is recorded as: $P_i = (P_{i1}, P_{i2}, \ldots, P_{iD})$.
(e) The optimum position of a particle within the population is: $P_g = (P_{g1}, P_{g2}, \ldots, P_{gD})$.
(f) The velocity of the particle is: $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

From the above assumptions, the algorithms for the particles are:

$$V_i^{l+1} = WV_i^l + c_1 \cdot rand()(P_i - X_i^l) + c_2 \cdot Rand()(P_g - X_i^l); \tag{1}$$

$$X_i^{l+1} = X_i^l + V_i^{l+1} \tag{2}$$

where V_i^l in Equation (l) is the velocity of the ith particle; V_i^{l+1} is the velocity from the current position to the next position (l+1); X_i^l is the position of every particle; X_i^{l+1} is the next position of the particle; P_i represents the optimum position recorded by the ith particle; P_g is the optimum position resolved by a population of particles; Rand () and rand () are random numbers between [0, 1]; c_1 and c_2 are learning factors which control the acceleration of particle velocity; and W is the inertial constant that allows the user to control the parameters. The particles influenced by a small W value initiate new searches within the current space; however, when the particles are influenced by a large W value, they will be directed to new searches in new spaces.

PSO has been applied in areas such as solving permutation problems, neural networks and operations research. Salerno (1997) applied PSO to neural networks, and Eberhart and Shi (2001) addressed using inertia weight to develop an improved method of controlling PSO. Wang, Huang, Zhou and Pang (2003) applied PSO to the traveling salesman problem. Sousa, Silva and Neves (2004) proposed using PSO in problems of information classification in data mining and compared the results of different particle speeds in both the updated mode and genetic algorithms. Tsai and Tien (2005) applied PSO to roundness measurement in machine vision. Tsou, Fang, Pei and Kao (2005) applied PSO to non-linear inventory planning problems. This paper attempts to use PSO to solve route planning problems of an order-picking system in a distribution center.

3. MODEL CONSTRUCTING AND RESEARCH METHOD

3.1 Model Constructing

This research probes into the storage layout of the picking area in the distribution center. Traditionally, the storage layout did not have a cross aisle; in order to pick an item from the first sub-aisle, it was necessary to walk from the first storage space to the last storage space or turn back to the first storage space and then proceed to the second sub-aisle. Consequently, lots of unnecessary walking was required. In a practical sense, this paper focuses on the planning of the storage system in the picking area of a distribution center, by adding cross aisles as shown in Figure 2. This will eliminate the unnecessary walking distance. Each storage space is numbered according to its I/O distance (See Figure 2).

3.2 PSO Solving Process

An important part of the PSO algorithm is particle velocity and particle position. In each iteration solution, each particle is updated by following two best values. The first one is the best solution it has achieved so far, which is pbest. Another value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the population, called gbest. How does the particle find these two best values? These particles are flying through hyperspace and remember the best position that they have seen. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. Section 2.2 explains how particles adjust their velocity and position through Equations (1) and (2). The PSO adaptability function evaluates the optimum particle position. This paper aims to obtain the shortest possible picking route.

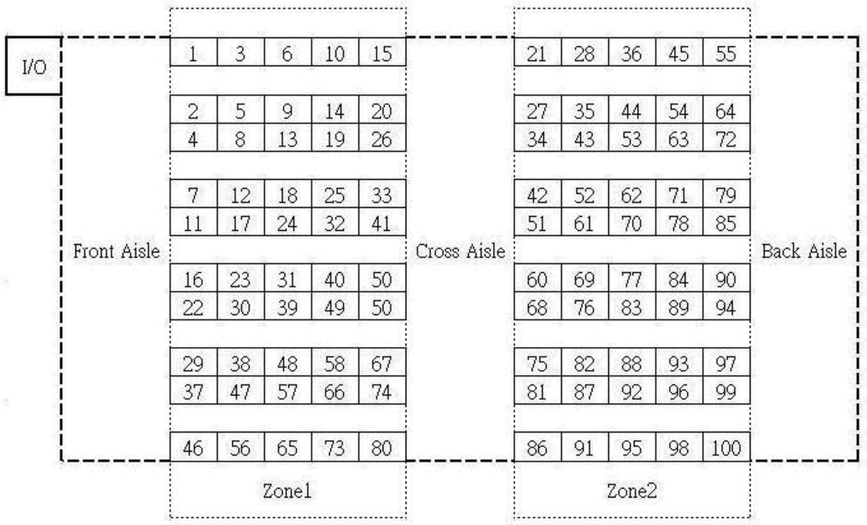


Figure 2. Storage layout of the distribution center.

3.2.1 PSO parameter setting

This paper simulates various parameters setting under different conditions and, with reference to studies by Hu (2002), we attempt to define the best parameter combination for the picking system.

(1) Number of Particles

Typical range is 20-40; however, 10 particles will obtain a better result. With larger problems, there is also an option of selecting 100 or 200 particles.

(2) Maximum Velocity

The maximum velocity sets the moving distance of the particles. If velocity of V_{id} is between (-10, 10), then the maximum velocity is at 20.

(3) Learning Factors

The learning factors of c_1 and c_2 usually have a value of 2. Different problems will have different values, and this ranges from 0 to 4.

(4) Stop Condition

The condition includes the maximum number of iterations the PSO executes and the minimum error requirement. This stop condition depends on the problem to be optimized.

(5) Inertia Weight

Inertia weight "*W*" was not mentioned by Eberhart & Kennedy in 1995; it was introduced by Shi & Eberhart in 1998. Shi & Eberhart found that a PSO with an inertial weight in the range of 0.8 to 1.2 has, on average, a better performance; that is, it has a larger chance of finding the global optimum within a reasonable number of iterations.

3.2.2 Applying PSO in route planning of the picking system

In order to fully adapt the PSO algorithm in our simulation we have modified the PSO process and present it as flow chart in Figure 3.

Step 1: Generate a random order for picking X_i^l and the initial velocity V_i^l and set $l = 0$. We will call this step iteration 0.

Step 2: Calculate particle adaptability according to the picking route and obtain the best fitness value P_i and global best value P_g.

Step 3: Update the best fitness value P_i and global best value P_g.

Step 4: Adjust the picking order of X_i^l and velocity V_i^l according to the best fitness value P_i and global best value P_g.

Step 5: If the stop condition satisfied, then stop; otherwise, return to Step 2.

4. SIMULATION MODEL AND TESTING RESULT

The structure of this simulation testing applies the Visual C^{++} language to the picking system in a distribution center. The current distribution center layout has 100 storage locations and 100 product items. The computer randomly generates 100 orders. Our results are verified by comparing Huang's (2004) results obtained by using an Ant colony optimization algorithm, and also by comparison with other search algorithms.

The simulated picking environment is rectangular in shape (see Figure 2). There are five main aisles, 10 storage spaces on each side of the aisle, resulting in a total of 100 storage spaces. Each storage space measures 1m wide and 1m deep, and the main aisle and cross aisle are both 2.5m in width. Picking starts from point

I and finishes at point O. By using a right angle calculation, the distance between storage *i* and *j* can be shown as:

$$\mathrm{Dist}(i, j) = \left|x_i - x_j\right| + \left|y_i - y_j\right| \cdot$$

According to PSO, the behavior of each individual is affected by either the best local or the best global individual to help it fly through a hyperspace. Moreover, an individual can learn from its past experiences to adjust its flying speed and direction. Therefore, by observing the behavior of the flock and memorizing their flying histories, all the individuals in the swarm can quickly converge to near-optimal geographical positions with well-preserved population density distribution (Lu, 2002). So, in this paper, we will consider different population size settings in PSO and also in other algorithms.

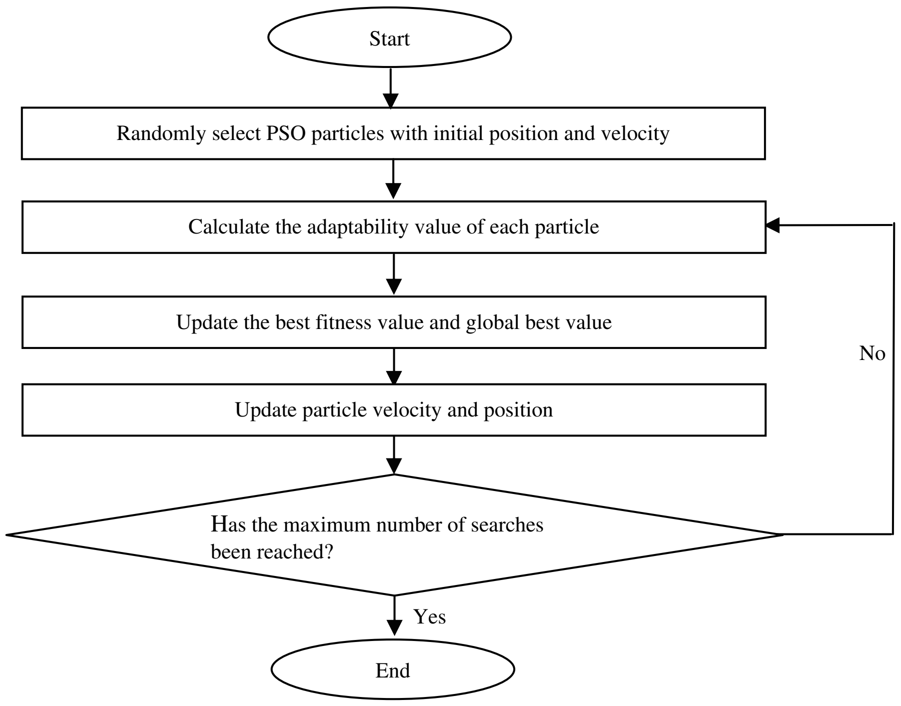

Start
Randomly select PSO particles with initial position and velocity
Calculate the adaptability value of each particle
Update the best fitness value and global best value
Update particle velocity and position
Has the maximum number of searches been reached?
No
Yes
End

Figure 3. PSO flow chart.

4.1 PSO Result Analysis and Comparison with other Algorithms

In this section, we compare the PSO with other search algorithms such as the genetic algorithm (GA), and also verify the Ant system (AS) of (Huang, 2004). We also propose using a GA to determine the initial solution, in order to let PSO reach an optimal solution faster. We consider the performance, total picking distance, total picking time, and CPU run time. The computer simulation structure, based on Figure 2, was used in all the search algorithms. The same stopping condition as in the PSO was also used in all the search algorithms. The following sections will show the parameters settings for all the search algorithms.

4.1.1 PSO

The learning factors $c_1 = c_2 = 2$ and the inertia weight at 0.8 are explained in Section 3.2.1. The results shown in Table 1 are the average of 20 results and also of a comparison of different population sizes. Our objective function is the best at a population size of 30 and the worst at 50. Thus, selecting an appropriate population size avoids over learning and will achieve an optimal function.

Table 1. *PSO Algorithm (100 orders)*

Population size		5	10	15	20	25	30	35	40	45	50
PSO	Total Picking Distance (m)	21687	22101	22865	24210	21104	20378	21548	24522	25895	26854
	Total Picking Time (s)	22414	22352	24100	25218	21198	21546	21996	25120	26455	27574
	CPU run time (s)	3	4	3	4	4	5	4	6	7	9

4.1.2 GA

This section is divided into two parts. Firstly we use a GA to find the initial solution and then apply PSO to determine the optimal solution. We label this optimal solution "GA-PSO." Secondly we apply a GA to find the optimal solution and then make comparisons with other search algorithms.

To begin the search cycle, a fitness function of the population members must be defined. Secondly, several parameters includes the population size, frequency of mutation, and maximum generations must be set. An initial population is then generated and the fitness of each individual is computed. If the fitness of any individual satisfies the stopping criteria, the search is terminated and the solution returned. Otherwise, a new generation is created through reproduction, crossover, mutation, and possibly other operations (Hashim & Tokhi, 2004).

The procedure for applying GA to find an initial solution and the optimal solution is as follows. First, create a group of many random tours in what is called a population. This algorithm uses a greedy initial population that gives preference to linking cities that are close to each other. Second, pick 2 of the better (shorter) tour parents in the population and combine them to make 2 new child tours. Hopefully, these child tours will be better than either parent. Third, a small percentage of the time is needed to ensure the child tours are mutated. This is done to prevent all tours in the population from looking identical. Fourth, the new child tours are inserted into the population replacing two of the longer tours. The size of the population remains the same. Finally, new child tours are repeatedly created until the desired goal is reached.(Michael LaLena, 2006)

The parameters for a GA are shown in the following statement. We set the population size to be the same as PSO. The fitness function is also set the same as PSO, which is the shortest distance. The mutation percentage is set as 3%. The results are shown in Table 2.

Table 2. *GA-PSO and GA Algorithm (100 orders)*

Population size		5	10	15	20	25	30	35	40	45	50
GA-PSO	Total Picking Distance (m)	20312	20318	20314	20342	20368	20342	20392	20338	20298	20227
	Total Picking Time (s)	21014	21348	20398	20378	22734	21438	21548	21545	21548	21001
	CPU run time (s)	3	3	2	3	3	4	3	2	3	2
GA	Total Picking Distance(m)	25796	25782	25125	25421	25942	25125	24953	25002	24951	24912
	Total Picking time(s)	26121	26529	26489	26148	26521	26415	26489	26102	25943	25928
	CPU run time (s)	6	5	7	5	6	5	5	5	6	5

4.1.3 Ant

In this paper, the probability function for the *k*-th ant traveling from location *i* to location *j* is similar to that in the previous literature on the ant system. In formula (3) below, τ_{ij} represents the pheromone released by ants when traveling from storage location *i* to storage location *j*; η_{ij} represents the reciprocal of the distance between storage location *i* and storage location *j*, $\eta_{ij} = 1/d_{ij}$, generally an indicator of an ant's sense of sight, in which the closer the city is, the easier it would be seen by the ants; d_{ij} represents the distance between storage location *i* to storage location *j*; the weights α and β are used for setting the degree of significance for τ_{ij} and η_{ij} as the extent *k* traveled by the ants must be feasible, which is represented by ***allowed***$_k$. The significance of the settings for each parameter value in the ant system theory is well known in the literature, especially the index value α and value β which controls the degree of significance between τ_{ij} and η_{ij}. If not set appropriately, it might result in phenomena such as stagnation or absence of a solution. Based on the suggestions from Dorigo, Maniezzo and Colorni (1991), this paper has set the two parameters values of α and β at the combinatorial optimum, which are (0.5, 5).

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \times \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in allowed_k} \left[\tau_{ik}(t)\right]^{\alpha} \times \left[\eta_{ik}\right]^{\beta}} & if\ j \in allowed_k; \\ 0 & otherwise. \end{cases} \tag{3}$$

The results shown in Table 3 are the average of 20 runs and also of a comparison of different population sizes. Thus, selecting an appropriate population size avoids over learning and will achieve the optimum function. In AS, the optimal solution of total picking distance performance is similar. With a population size of twenty being the best and fifty being the worst.

4.1.4 Comparison of all Results

Nowadays PSO and ACO have developed to be real competitors for other comparatively older and well-established techniques for population-based evolutionary computation, notably Genetic Algorithms, Evolutionary Programming, Genetic Programming, etc., and other prominent non-population based metaheurisitc techniques, e.g., simulated annealing and tabu search. Among PSO and ACO, PSO has particularly gained prominence due to its relative ease of operation and capability to arrive quickly at an optimal or near-optimal solution. However it was pointed out that although PSO can show significant performance in the initial iterations, the algorithm might encounter problems in reaching optimum solutions efficiently due to several function approximation problems (Angeline, 1998).

From the numeric results, we can see that a meta-heuristic or artificial intelligence algorithm (e.g., GA) has a higher convergence rate (i.e., less CPU time required), but it seems to fall easily into a local optimum. On the other hand, the original PSO algorithm also falls into the trap of an intervening valley before finally moving to a position that is close to the global optimum; but this consumes more computational time.

In this section we will analyze all the output data which are shown in the previous sections. We transfer the numerics to figures, in order to see clearly the difference between different algorithms. So, we can see that the shortest path and shortest time is GA-PSO, which is shown in Figures 4 and 5. The best CPU run time is achieved by applying GA-PSO and AS.

Table 3. *PSO Algorithm (100 orders)*

	Population size	5	10	15	20	25	30	35	40	45	50
AS	Total Picking Distance(m)	27745	25421	27854	24675	25465	26421	27542	28012	28421	28422
	Total Picking time(s)	28501	26424	28465	25542	26897	27001	28301	28945	29100	29124
	CPU run time (s)	0	1	1	1	2	1	2	4	4	5

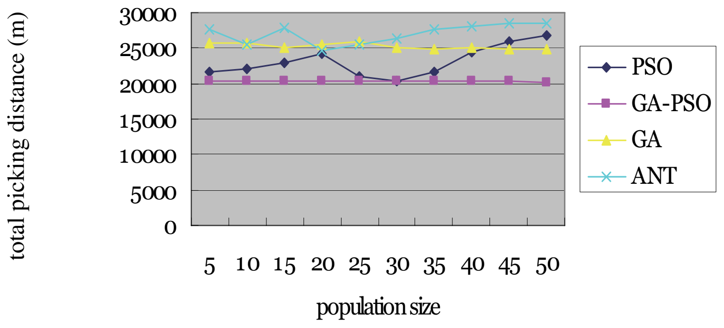


Figure 4. Relationship of total picking distance and population size.

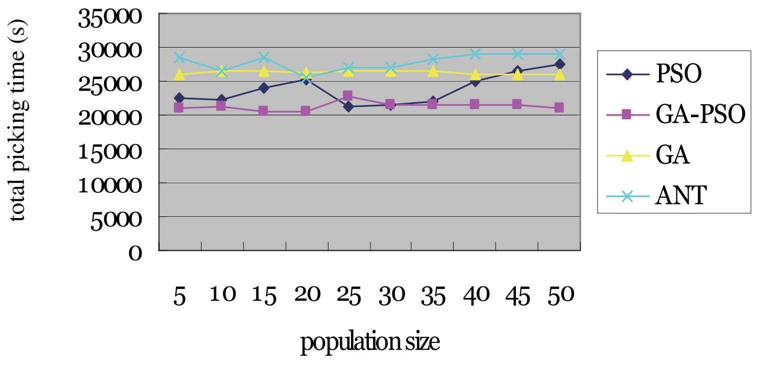


Figure 5. Relationship of total picking time and population size.

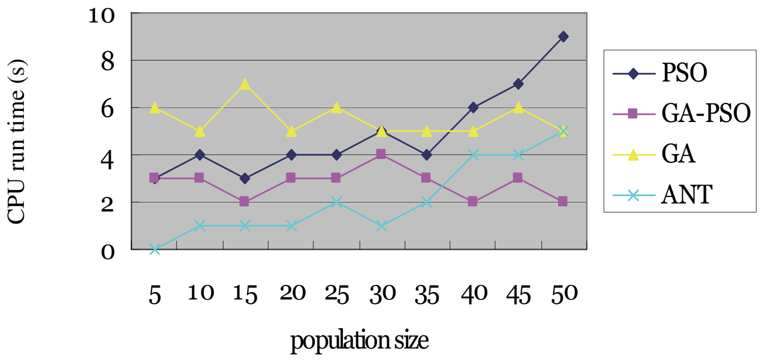


Figure 6. Relationship of CPU run time and population size.

4.2 PSO Parameter Analysis

The most important modified parameter in PSO is its learning factors c_1, c_2 and inertia weight W. We analyze the parameters from Table 1.

(1) Learning Factors (c_1, c_2)

Firstly, the learning factor parameters of c_1 and c_2 from Table 1 both have a value of 2. This is the value widely used in research studies. However, when faced with different problems, different setting methods will be adopted. This paper attempts to analyze the parameters c_1 and c_2 by making them equal, unequal, making c_1 larger than c_2, or vice versa. Using a sample of thirty particles (see Table 1), with terminal number at 100, the W value at 0.8 and parameter set randomly, the results are presented in Table 4.

c_1 is the learning experience of a single particle, c_2 is a shared learning experience for the whole class of particles. Table 4 is the average of twenty results. Comparisons are made with different c_1 and c_2 parameters. When both are set at the value of 2, the total picking distance and the total picking time are the shortest. When $c_1 = 4$, $c_2 = 0$ and when $c_1 = 2$, $c_2 = 0$, individual particles were able to search the optimum picking route. When $c_1 = 0$, $c_2 = 4$ and 2, independent learning experience is not required, as the whole class of particles is able to find the optimum picking route.

Table 4. *Analysis Parameters of c1 and c2*

c_1 and c_2 parameter setting	c_1=4, c_2=0	c_1=0, c_2=4	c_1=2, c_2=0	c_1=0, c_2=2	c_1=1, c_2=1	c_1=1, c_2=2	c_1=1, c_2=3	c_1=2, c_2=1	c_1=2, c_2=2	c_1=3, c_2=1
Total Picking Distance(m)	26894	27542	26542	27245	24321	23421	22124	21421	20378	20875
Total Picking Time(s)	28542	28221	27542	27986	25698	24512	23218	22684	21546	20968
CPU run time(s)	2	3	2	3	4	5	5	5	5	5

(2) Inertia weight (*W*)

In Shi and Eberhart (1998) the *W* value is to be applied cautiously. An overly small *W* value will only achieve a partial solution, but optimal timing can be missed with an overly large *W* value. Therefore, a value between 0.8 and 1.2 is suggested to achieve the best solution. However, different problems will warrant a different setting method. The *W* value in Table 1 is set at 0.8, which is within the suggested range of Shi & Eberhart. This paper will attempt to modify the *W* value to see if the results are affected by such modifications. With a sample of 30 particles (see Table 1), terminating at 100 times with the values of c_1 and c_2 at 2 (see Table 4), parameters are set randomly. The results are presented in Table 5.

From Table 5, we can see that when the *W* value is at 0.8, the best results for the total picking distance, total picking time and CPU run time is achieved. When the *W* value is at 1.0 the longest picking distance is 21,401m, which differs from the optimum distance by only 1,023m. Thus, we can conclude that the setting of the *W* value has no significant effect on the problem of the picking operation.

Table 5. *Analysis Parameters of W*

W	0.8	0.9	1.0	1.1	1.2
Total Picking Distance(m)	20378	20452	21401	20965	20384
Total Picking Time(s)	21546	21982	21584	22101	21551
CPU run time (s)	5	5	5	6	5

(3) Comparison of Maximum Recurring Number

The value of the maximum recurring number will affect the final results, as a smaller value will cause the particles to scatter around and not be able to leave the local optima. By using a sample of thirty particles (see Table 1), setting the values of c_1 and c_2 at 2 (see Table 4), W at 0.8 (see Table 5) and randomly setting the parameters, the results are shown in Table 6 from which we can see that when the recurring number is at 300, the average picking time and picking distance is at its optimum. Thus, we can derive the conclusion that the larger the recurring number is, the better the solution we could get.

On the other hand, about the CPU run time, the larger the recurring number, the more time will be required to compute the result, as the CPU run time and the recurring numbers have a linear relationship. Even though large recurring numbers reduce the average picking distance and time, it is worth noting the learning process. Results may decline due to excessive learning. Thus selecting a suitable recurring number will avoid undue learning time.

Table 6. *Maximum Recurring Number Analysis*

Maximum Recurring Number	50	100	150	200	250	300
Total Picking Distance (m)	20875	20378	20324	20101	20098	20085
Total Picking Time(s)	22149	21546	21422	21982	21489	21321
CPU run time (s)	4	5	6	6	7	9

5. CONCLUSIONS

The purpose of this paper is to analyze and discuss the efficiency of order picking operations by applying Particle Swarm Optimization (PSO) to route planning. Empirical testing results demonstrate that PSO has improved the average order-picking time and distance. Thus, our conclusions for route planning are as follows:

(1) None of the published literature has applied PSO in the route planning of the order-picking operations. This paper has successfully applied PSO to find the optimal solution in route planning and it is hoped to bring forth further in-depth studies by other researchers.
(2) In comparison with the Ant system in finding the optimal solution in route planning, our empirical results indicate that by applying PSO we have achieved optimal quality within the required time frame in an efficient manner.
(3) Compared with GA, PSO has the flexibility to control the balance between the global and local exploration of the search space. This property enhances the search capabilities of the PSO technique and gives an improved performance, compared to that of a GA.
(4) In this paper, we proposed using a GA to find the initial solution, and then apply this to PSO to find the optimal solution. The results shown in Figure 4 and 5 emphasize that the proposed approach finally leads to an optimal solution.
(5) Results can vary depending on the setting of different PSO parameters.

In a practical sense, this paper focuses on the planning of the storage system in the picking area of a distribution center. The addition of cross aisles between storage spaces will improve the overall efficiency of the distribution center and thus we have provided a good reference for warehouse layout design.

In this paper, we proposed using a GA to define the initial solution, and verified that it really can help the PSO to find an optimal solution faster. Further research can be directed not only to compare other search algorithms, such as simulated annealing (SA), but also to combine with other search algorithms. It may also be profitable to try using neural networks to determine the fitness weight of PSO. Fuzzy numeric data, or dynamic data can also be considered in the solution algorithm. The authors wish to pursue this development as an immediate direction for future research.

APPENDIX: LIST OF SYMBOLS

Symbol	Description
gbest	Global best solution
pbest	Individual best solution
i	Representing particles within the population size that have different serial numbers
d	Representing the dimensions which each particle is composed of
X_i	The position of i^{th} particle
P_i	Best position of every particle
P_g	Optimum position of a particle within the population
V_i	Velocity of the particle
V_i^l	The velocity of the i^{th} particle
V_i^{l+1}	The velocity from the current position to the next position ($l + 1$)
X_i^l	The position of every particle
X_i^{l+1}	The next position of the particle
Rand()	Random number between [0, 1]
rand()	Random number between [0, 1]
c_1	Learning factors which control the acceleration of particle velocity
c_2	Learning factors which control the acceleration of particle velocity
W	The inertial constant that allows user to control the parameters
$p_{ij}^k(t)$	The probability function for the k-th ant traveling from location i to location j
d_{ij}	Representing the distance between storage location I to storage location j
τ_{ij}	Representing the pheromone released by ants when traveling from storage location i to storage location j
η_{ij}	representing the reciprocal of the distance between storage location i and storage location j
$allowed_k$	The weights α and β are used for setting the degree of significance for τ_{ij} and η_{ij} as the extent k traveled by the ants must be feasible
α	The weights used for setting the degree of significance for τ_{ij} and η_{ij} as the extent k traveled by the ants must be feasible
β	The weights used for setting the degree of significance for τ_{ij} and η_{ij} as the extent k traveled by the ants must be feasible

ACKNOWLEDGEMENT

The authors would like to thank The National Science Council, Republic of China, for financially supporting this research under the grant of NSC-95-2221-E-216-026-.

REFERENCES

Angeline, P. J. (1998). Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. *Proceedings of Evolutionary Programming VII, Lecture Notes in Computer Science*, Springer Berlin, 601-610.

Bozer, Y. A., & White, J. A. (1996). A generalized design performance analysis model for end-of-aisle order-picking systems. *IIE Transactions*, *28*, 271-280.

Clerc, M., & Kennedy, J. (2002). The particle swarm explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*, 58-73.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Positive feedback as a search strategy* (Tech. Rep. No. 91016). Italy: Dipartimento di Elettronica e Informatica, Politecnico di Milano.

Dong, F. C., & Chen, M. T. (1995). Picking Operation in Distribution Center, *Ministry of Economic affairs*, *Commercial Automation Series*.

Eberhart, R. C., & Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *Proceedings of Congress on Evolutionary Computation*, *1*, 27-30.

Fan, H. (2002). A modification to particle swarm optimization algorithm. *Engineering Computations*, *19*(8), 970-989.

Hashim, S.Z. & Tokhi M.O. (2004). Genetic Modelling and Simulation of Flexible Structure. *Studies in Informatics and Control,* 13(4).

Hu, X. H. (2002). *PSO Tutorial*. Retrieved April 2, 2006 from the World Wide Web: http://www.swarmintelligence.org/tutorials.php

Hsieh, L. F., & Lin, C. C. (2005). Compatibility study between storage assignment and order picking strategy in distribution centers. Submitted to *Production Planning and Control.*

Huang, C. J. (2004). *Integrated Design of the Zoning Storage, Order Batching, Route Planning and Performance Evaluation in Distribution Center*. Unpublished Master's Thesis, Graduate Institute of Department of Technology Management, Chung Hua University, Hsinchu, Taiwan.

Kevin, G., Russell, M., & Joseph, S. (2006). The effects of pick density on order picking areas with narrow aisles. *IIE Transactions*, *38*(10), 859-868.

Kennedy, J., & Eberhart, R. C. (1995). Particle Swarm Optimization. *Proceedings of IEEE Int'l. Conf. on Neural Networks*, *IV*, 1942-1948.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Proceedings of IEEE Int'l. Conf. on Systems, Man, and Cybernetics*, 4104-4108.

Lu, H. (2002). Dynamic population strategy assisted particle swarm optimization in multiobjective evolutionary algorithm design. *School of Electrical and Computer Engineering Oklahoma State University,* Stillwater, Oklahoma, USA.

Michael LaLena, Traveling Salesman Problem Using Genetic Algorithms. Retrieved April 2, 2006 from the world wide web: http://www.lalena.com/AI/Tsp/

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, *1*, 235-306.

Roodbergen, K. J., & Koster, R. D. (2001a). Routing method for warehouse with multiple aisles. *International Journal of Production Research*, *39*(9), 1865-1883.

Roodbergen, K. J., & Koster, D. R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, *133*, 32-43.

Salerno, J. (1997). Using the particle swarm optimization technique to train a recurrent neural model. *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*, 45-49.

Salman, A., Ahmad, I., & Al-Madani, S. (2002). Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, *26*, 363-371.

Schutte, J., & Groenwold, A. (2005). A study of global optimization using particle swarms. *Journal of Global Optimization*, *31*(1), 93-108.

Shi, Y., & Eberhart R. (1998). A modified particle swarm optimizer. *Proceedings of IEEE International Conference on Evolutionary Computation,* Anchorage, Alaska, USA.

Shigenori, N., Takamu, G., Toshiku, Y., & Yoshikazu, F. (2003). A hybrid particle swarm optimization for distribution state estimation. *IEEE Transactions on Power Systems*, *18*, 60-68.

Sousa, T., Silva, A., & Neves, A. (2004). Particle swarm based data mining algorithms for classification tasks. *Parallel Computing*, *30*(5-6), 767-783.

Stern, H. I. (1986). Parts location and optimal picking rules for a carousel conveyor automatic storage and retrieval system. In J. A. White (ed.), *Proceedings of 7th International Conference Automation Warehousing* (pp. 185-193). New York, USA: Springer.

Trelea, L. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, *85*(6), 317-325.

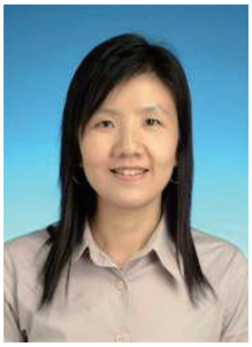
Tsai, T. H., & Tien, F. C. (2005). Applied particle swarm optimization algorithm to roundness measurement, *Proceedings of Chinese Institute of Industrial Engineers Annual Conference, Taiwan, 2005.*

Tsou, C. S., Fang, H. H., Pei, W., & Kao, C. H. (2005). Applying particle swarm optimization to multiple objectives inventory planning problem, *Proceedings of Chinese Institute of Industrial Engineers Annual Conference, Taiwan, 2005.*

Vickson, R., & Fujimoto, A. (1996). Optimal storage locations in a carousel storage and retrieval system. *Location Science*, *4*, 237-245.

Wang, K. P., Huang, L., Zhou, C. G., & Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *Proceedings of International Conference on Machine Learning and Cybernetics*, Xian, China, 1583-1585.

Yoshida, H., Kawata, K., Fukuyama, Y., & Nakanishi, Y. (1999). A particle swarm optimization for reactive power and voltage control considering voltage stability. *Proceedings of International Conference on Intelligent System Application to Power Systems*, Rio de Janeiro, Brazil, 117-121.



Ling-Feng Hsieh, Professor, Department of Technology Management, Chung Hua University, Taiwan. Dr. Hsieh received her Doctoral degree from the department of industrial engineering and management at National Chiao Tung University, her Master's degree from the Department of Industrial Engineering and Engineering Management from National Tsing Hua University, and a Bachelor's degree from the Department of Applied Mathematics from Fu Jen Catholic University. From August 2006, she has been the Vice Provost of Academic Affairs of Chung Hua University in Taiwan.

Professor Hsieh's major research interests include performance evaluation and management, production management, and supply chain management.

Chao-Jung Huang received her Master's degree from the Department of Technology Management at Chung Hua University, and a Bachelor's degree from the Department of Industrial Management at Chung Hua University.

Chien-Lin Huang is a graduate school student in the Department of Technology Management at Chung Hua University. He received a Bachelor's degree from the Department of Industrial Management at Chung Hua University.