P-ISSN 2586-9000
E-ISSN 2586-9027
**Homepage** : *https://tci-thaijo.org/index.php/SciTechAsia* **Science & Technology Asia**
Vol.27 No.3 July - September 2022 Page: [204-215]

*Original research article*

# A New Discretization Technique for Enhancing Discrete Particle Swarm Optimization's Performance

Choosak Pornsing[1,*], Noppakun Sangkhiew[1], Pheera Sakonwittayanon[1], Peerapop Jomtong[2], Shunichi Ohmori[3]

[1]*Department of Industrial Engineering and Management, Faculty of Engineering and Industrial Technology, Silpakorn University, Nakhon Pathom 73000, Thailand*
[2]*Department of Biomedical Engineering, Faculty of Health Sciences, Christian University, Nakhon Pathom, 73000, Thailand*
[3]*Graduate School of Creative Science and Engineering, Waseda University, Tokyo 169-8050, Japan*

Received 5 March 2021; Received in revised form 13 September 2021
Accepted 21 September 2021; Available online 29 September 2022

## ABSTRACT

In this paper, a new technique of converting continuous values to discrete values for Particle Swarm Optimization (PSO) is proposed. PSO is a powerful metaheuristic search technique that has been deployed to solve many complicated and large optimization problems. However, it has a drawback on deteriorated performance when dealing with a discrete search space. The proposed discretization technique borrowed the idea of a traditional gambling game, Sic Bo game. The new technique allowed us to cultivate the information about the global best of the swarm and the personal best of the particles that the existing discretization techniques do not. The proposed technique was tested on uncapacitated fixed-charge location problems and compared to other three existing techniques: Sigmoid function, Hyperboric Tangent function, and Khanesar et al.'s technique. Furthermore, ten benchmark problems were drawn from the OR-library. The experimental results showed that the new technique outperformed the competitive techniques on medium to large size benchmark problems. It yielded better solutions which varied from 0.17% to 4.19%. The comparable technique in this study was Khanesar et al.'s technique. It yielded 0.13% difference from our technique on large size benchmark problem. Nevertheless, by performing statistical testing, our proposed technique yielded the better solutions significantly.

**Keywords:** Discretization; Facility location problems; Metaheuristics; Particle swarm optimization

***Corresponding author:** pornsing_c@su.ac.th **doi:** 10.14456/scitechasia.2022.56

## 1. Introduction

Optimization is concerned with finding the best (or optimal) solution to a problem [1]. The problem means a system that there exist an enormous variety of activities in the everyday world, it varies from actual systems such as chemical processing plants, manufacturing systems, logistics and distribution systems, and virus outbreak models. There are numerous decision variables required to be set which maximize the gain and minimize the loss. The decision variables are assumed to be dependent upon several factors and some of these factors are often under the control or partial control.

Generally, the process of optimization of a system can be divided into six steps: (i) mathematical analysis; (ii) construction of a mathematical model; (iii) validation of the model; (iv) manipulation of the model to produce the optimal or a satisfactory solution; (v) implementation of the solution; and (vi) introduction of a strategy which monitors the performance of the system. Theoretically, there are many mathematical models and optimization techniques are being exploited to solve practical problems. One of the most well-known optimization tools is linear programming. Linear programming refers to a mathematical model that is specified by a linear, multivariable function that is to be optimized (maximized or minimized) subject to several linear equations/non-equation constraints [2, 3]. The most popular algorithm used to solve linear programming problems is the simplex method, invented by G. B. Dantzig in 1963. The simplex method has been modified into an efficient algorithm to solve large linear programming problems by computer. Several practical problems can be modeled as linear programming and solved by the simplex method such as resource allocation problems in business and government planning, network analysis for logistics planning, production planning problems in industry, management of transportation and distribution problems, and so on. Hence, linear programming is one of the successes of modern optimization techniques.

However, there are numerous optimization problems whose variables do not fall into continuous values. *Integer programming*, thus, is proposed. It is concerned with the solution of optimization problems in which at least some of the variables must assume only integer or binary values. The subtopic of this is often called *integer linear programming*. Many problems of a combinatorial nature can be formulated in terms of integer programming. Practical examples include facility location, activity scheduling, assembly line balancing, matching problems, inventory control, and machine replacement problems

There are several exact algorithms for the combinational problems [4], e.g., the branch and bound method, the cutting plane method, the branch and cut method, the column generation, the Lagrangian relaxation, the Dantzig-Wolfe decomposition, and the Bender's decomposition. Even with progress, however, the methods are not applicable to a large-sized problems, which can often arise in practical applications. For example, the exact method for facility layout problem can only handle the problem with 12 departments with state-of-art algorithms, whereas most industrial applications include 30-50 departments [5]. This will increase the need for the study of efficient metaheuristics

This kind of problem is called the combinatorial optimization problem. Combinatorial optimization is the process of searching for maxima or minima of an objective function whose domain is a discrete and finite set of objects but large configuration space. Theoretically, those problems are $\mathcal{NP}$*-hard* problems for which the optimal solution could not be found in reasonable computational time.

Many metaheuristics have been proposed to solve such problems efficiently: genetic algorithm (GA), simulated annealing (SA), and gravitational search algorithm

(GSA) are some examples. Among these techniques, swarm intelligence is one of the powerful approaches. It is an adaptive computing technique that gains knowledge from the collective behavior of a decentralized system composed of simple agents interacting locally with each other and the environment [6-8].

One of the prominent techniques is particle swarm optimization (PSO). It was originally designed and introduced by Kennedy and Eberhart in 1995 [9] and has been applied to many practical optimization problems through its 25 years because of its rapid convergence towards an optimum property [10]. However, as mentioned before, many optimization problems fall into a combinatorial optimization problem with at least one of its decision variables being a discrete number while; basically, the particle swarm optimization is a continuous search algorithm. Thus, it needs a mechanism to deal with such characteristics.

To the best of our knowledge, there are some techniques in machine learning adopted to convert continuous values to discrete values for PSO, such as Sigmoid function and Hyperbolic tangent function. There are some shortcomings of these techniques when applied to solve discrete decision variables in PSO. For example, in the continuous PSO, a large value for maximum velocity of the particle encourages exploration. However, in the discrete PSO, a small value for maximum velocity still promotes exploration, regardless of a good solution being found so far. One more example of the existing techniques' deficiency is that the next position of a particle is quite independent from the current position. It ignores the information about the current position and leaps to a new position solely updated using the new velocity. In the continuous PSO, the update rule uses current position of swam and the velocity vector just determines the movement of the particle in the search space.

To this point, this paper presents a new technique of discretization for particle swarm optimization (PSO). The new technique is a probabilistic-based technique the same as the existing ones. However, it still uses the information of the current position and the size of velocity affects the swarm's exploration. The new technique is also designed to be easy to encode and decode; furthermore, it consumes less computational resources.

The rest of this manuscript is organized as follows. Section 2 briefly explains the classical PSO, its mathematical formulation, and the procedure of this algorithm. This section also includes the related works about the discretization techniques in the literature. Section 3 proposes the new technique on the discretization procedure for PSO. It also shows the computational experiments with benchmark techniques on uncapacitated facility location problems which are a classical problem in logistics and supply chain management. The results and discussion are drawn in section 4. A conclusion summarizing the contributions of this study is in section 5.

## 2. Particle Swarm Optimization

PSO is a multi-agent search algorithm based on the flocks of social animals, such as the flocks of birds or the schools of fish. The agents are called 'particles' and the pack of particles is called 'swarm'.

PSO is an iterative algorithm. The particles determine their fitness values concerning the objective function, at their current positions. Later, each particle calculates its travel direction in the search space by exploiting the information about its current fitness, its best fitness ever visited (individual perspective) and best fitness locations ever visited by a particle in the swarm (social perspective), with random perturbations. Finally, the next iteration is executed after all particle positions have been updated.

## 2.1 Classical particle swarm optimization

The outline of PSO algorithm is described as follows. The particle has four pieces of information; the velocity of $i$-th particle after $k$-th iteration in $n$-th dimensional space $X_i = (X_{i1}, X_{i2}, ..., X_{in})$ where $i = 1,...,m,$ and $m$ is the size of the swam, $k = 1,...,K$ and $K$ is the number of iterations; the velocity of $i$-th particle $V_i = (V_{i1}, V_{i2}, ..., V_{in})$; the best position of $i$-th particle visited so far, called $pbest$; the best position of the entire group visited so far, called $gbest$ [11].

The new positions and velocities of the particles are determined by using equations (2.1) and (2.2).

$$X_i^{k+1} = X_i + V_i^{k+1}, \tag{2.1}$$

$$V_i^{k+1} = \omega V_i^k + \phi_1\beta_1\left(pbest_t - X_i^k\right) + \phi_2\beta_2\left(gbest_t - X_i^k\right), \tag{2.2}$$

Where $\omega$ is the inertia weight factor, $\phi_1$ and $\phi_2$ are cognitive and social weighted factors respectively, and $\beta_1$ and $\beta_2$ are random variables uniformly distributed within $[0,1]$. The movement of a particle in the swarm is shown in Fig. 1 and the pseudocode of the classical PSO is described in Table 1.

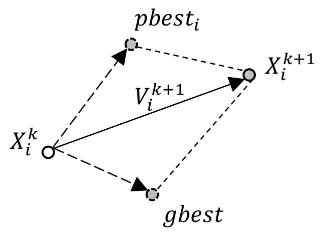

**Fig. 1.** Movement of the particle.

The advantages of PSO are its quality of being easy to encode and decode, being fast to compute, and being simple to parallel compute computing. However, as mentioned before, PSO was invented to search in a continuous space. It needs a mechanism to convert continuous values to discrete values, called the discretization techniques [12].

**Table 1.** Pseudocode of the classical PSO.

| Step | Description |
|---|---|
| 1 | Initialization<br>(a) Set $k = 0$.<br>(b) Randomly initialize the position of the $m$ particles.<br>(c) Randomly initialize the velocity of the $m$ particles.<br>For $i = 1,...,m$ do $pbest_i^k = X_i$.<br>Set $gbest^k = \arg\min_{i\in\{1,...,m\}} f(X_i)$. |
| 2 | Terminate Check. If the termination criteria hold stop. Then, the outcome of the algorithm will be $gbest$. |
| 3 | For $i = 1,...,m$ do<br>Updating particle swarm.<br>(a) Update the velocity $V_i^k$ using Eq. (2.1).<br>(b) Update the position $X_i^k$ using Eq. (2.2).<br>Updating $pbest_i$ and $gbest$.<br>(c) If $f(X_i) \le f(pbest_i)$ then $pbest = X_i$.<br>(d) If $f(pbest_i) \le f(gbest)$ then<br>$gbest = pbest_i$.<br>End For. |
| 4 | Set $k = k + 1$. |
| 5 | Go to step 2. |

Source: [8].

## 2.2 Discretization techniques

The most popular technique for converting continuous values to discrete values in metaheuristics and machine learning is the sigmoid function. Originally, it was invented to deal with binary conversion. The following equations are used.

$$\theta(z) = \frac{1}{1+e^{-z}}, \tag{2.3}$$

$$x = \begin{cases} 1, & \text{if } r < \theta(z), \\ 0, & \text{otherwise,} \end{cases} \tag{2.4}$$

where $z$ is a real number and $r$ is the uniform random number in the range [0, 1]. The equation yields the binary number $x$, 0 or 1. Fig. 2 shows the curve of the sigmoid function.

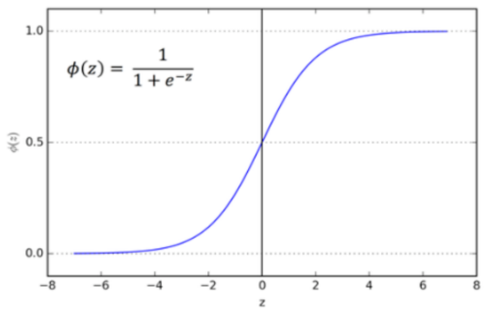

**Fig. 2.** The curve of the sigmoid function.

The sigmoid function can formulate integer numbers between 0 and $m-1$ by using the following functions.

$$\theta(z)=\frac{m}{1+e^{-z}}, \tag{2.5}$$

$$x=round\left(\tilde{x}+(m-1)\times\alpha\times\beta\right), \tag{2.6}$$

$$x=\begin{cases} m-1, & \text{if } x>m-1, \\ 0, & \text{if } x<0. \end{cases} \tag{2.7}$$

Eq. (2.5) transforms the continuous number to the continuous number between 0 and $m$. Eq. (2.6) generates a discrete number between 0 and $m-1$ using a normal distribution with $\mu=\theta(z),\sigma$ as parameters and $\beta$ is a random number, [0, 1]. Then, the number is rounded to the closet integer number by using the round function. Eq. (2.7) fixes the outlier integer numbers to the endpoints.

This is a simple technique among many techniques used in metaheuristics. However, to the best of our knowledge, there has been no study to reveal its efficiency.

The hyperbolic tangent function is one of the well-known mechanisms used in the artificial neural network as a transfer function. It is the ratio between hyperbolic sine function and hyperbolic cosine function.

$$\tanh(x)=\frac{\sinh(x)}{\cosh(x)}=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \tag{2.8}$$

where $\tanh(x)$ is hyperbolic tangent function and $x$ is the value between $[-1,1]$. The curve of it is shown in Fig. 3.

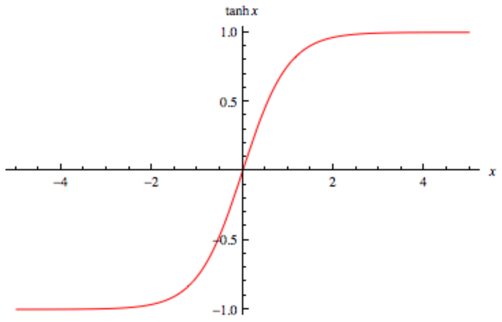

**Fig. 3.** The curve of the hyperbolic tangent function.

The procedure of transferring from $\tanh(x)$ to desired integer number can make use of Eqs. (2.6) - (2.7).

All the above functions are called the simple transform functions that make use of basic mathematics. Some techniques have been particularly invented for PSO. Khanesar et al. [13] proposed a new technique to update the velocity of a particle. The new technique was based on the sigmoid function. The authors described two problems when using this function to normalize the continuous values. First, in the real-number version of PSO a high value of particle velocity promotes an exploration; whilst, the low value of particle velocity encourages exploitation in the binary PSO. The conversion tends to maintain the particle velocity at 0. There is a troublesome consequence with selecting a proper inertia weight $\omega$. For binary PSO, if the value of $\omega>1$, it keeps out of convergence. Conversely, if the value of $-1<\omega<1$, the particle velocity stays at 0 over time. Second, updating particle velocity by using Eq. (2.4) does not need the information of the current values of that vector. It is independent of the current situation. This is much different from the continuous PSO version. The new position is calculated from the current position and the velocity vector. The authors advised a new procedure to update binary numbers. In their report, there are two velocity vectors for a particle, $V_{i,0}^{k}$ , and $V_{i,1}^{k}$. $V_{i,0}^{k}$ is the probability of the bits of the particle to change to zero while $V_{i,1}^{k}$ is the probability that bits of particle change to one. $V_{i,0}^{k}$, and $V_{i,1}^{k}$ change based on the information of the current situation. The logical order is shown below.

If $pbest_i = 1$ then $d_{i,1}^{1} = r_1\delta_1$ and $d_{i,0}^{1} = -r_1\delta_1$,
If $pbest_i = 0$ then $d_{i,0}^{1} = r_1\delta_1$ and $d_{i,1}^{1} = -r_1\delta_1$,
If $gbest_i = 1$ then $d_{i,1}^{2} = r_2\delta_2$ and $d_{i,0}^{2} = -r_2\delta_2$,
If $gbest_i = 0$ then $d_{i,0}^{2} = r_2\delta_2$ and $d_{i,1}^{2} = -r_2\delta_2$,

Where $d_{i,0}^{1}, d_{i,1}^{1}, d_{i,0}^{2}, d_{i,1}^{2}$ are temporary values, $r_1$ and $r_2$ are random numbers between (0, 1) that are revised each iteration, $\delta_1$ and $\delta_2$ are user-predefined numbers. Then, $V_{i,0}^{k}$ and $V_{i,1}^{k}$ are updated as follows:

$$V_{i,0}^{k} = \omega V_{i,0}^{k} + d_{i,0}^{1} + d_{i,0}^{2}, \tag{2.9}$$

$$V_{i,1}^{k} = \omega V_{i,1}^{k} + d_{i,1}^{1} + d_{i,1}^{2}. \tag{2.10}$$

This idea exploits the information of $gbest$ and $pbest_i$. If the $i$-th bit in the global best is zero or $i$-th bit in the personal best vector is zero, $V_{i,0}^{k+1}$ is increased and vice versa. The velocity changes by using Eqs. (2.3) - (2.4), and then update the particle position as follows:

$$X_i^{k+1} = \begin{cases} 1, & \text{if } r < sig\left(V_i^{k+1}\right), \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

Tsia & Wu [14] introduced 'shift operator' from computer programming languages to cope with feeder reconfiguration in power distribution systems. SO(B$_i$, D$_{L,R}$, S$_c$) means an act of changing the position in the sequence of switch states system. B$_i$ is the index of $i$-th the bit. D$_{L,R}$ is the direction of shifting left or right. S$_c$ is the number of shifting steps. A set of SO operators is called shift operator set. SOS = {SO$_1$, SO$_2$, …, SO$_n$} where n is the number of shift operators. Suppose there are two sequence of switch states, SSS$_1$ = [1 0 1 1 0] and SSS$_2$ = [0 0 1 0 0]. The determination of shift operator set SSS$_2$ Θ SSS$_1$ = {SO$_1$, SO$_2$}. The symbol Θ means getting the shift operator from SSS$_1$ to SSS$_2$. Likewise, $X_i$ Θ $gbest$ means getting the shift operator from $gbest$ to $X_i$. It encourages using social information as the concept of original PSO. New particle velocity and particle position calculations were updated as follows:

$$X_i^{k+1} = X_i \oplus V_i^{k+1}, \tag{2.12}$$

$$\begin{aligned} V_i^{k+1} &= \left(\omega \otimes V_i^k\right) \oplus \\ &\left(\beta_1 \langle\times\rangle \left(pbest_i \Theta X_i^k\right)\right) \oplus \\ &\left(\beta_2 \langle\times\rangle \left(gbest \Theta X_i^k\right)\right), \end{aligned} \tag{2.13}$$

where ⊕ is the combining of two-shift operator sets, ⊗ is the number of steps in shifting, and $\langle\times\rangle$ is the selection of the number of shift operators. This technique is also a simple one. However, it is not easy to code the operator set and it also consumes substantial computational resources.

There are some effective techniques in the literature. Pannambalam et al. [15] introduced a specific discrete PSO for flowshop scheduling problems. The encoding formulation allows the exchange of jobs between vectors in the given order. Tasgetiren & Liang [16] combined the genetic operator with the procedure of PSO. This allowed them to code binary values to solve lot-sizing problems. The main mechanism was not much different from the sigmoid function. However, its performance compared to Wagner and Whitin algorithm was noted. Wang et al. [17] presented a general binary PSO, called a novel probability binary PSO. The mechanism of converting continuous values to binary values was defined as an equation below.

$$L(z) = \frac{(z - R_{\min})}{(R_{\max} - R_{\min})}, \tag{2.12}$$

$$pz = \begin{cases} 1, & \text{if } r < L(z), \\ 0, & \text{otherwise}, \end{cases} \tag{2.13}$$

where $L(z)$ is a linear function, and $[R_{\max}, R_{\min}]$ is a predefined range for gaining the probability value with $L(z)$ function. The drawback of this idea is not different from other simple transfer functions. That is, they lost the information from the swarm. Accordingly, its logic deviates from the swarm behavior concept. A reader can find a more thorough review on discretization techniques in [18-20].

## 3. Research Method

In this section, the proposed discretization technique is explained. Its numerical illustration is also included. The experimental design is detailed.

### 3.1 Proposed technique

Our new technique borrows the concept of 'Sic Bo Game' method that is a traditional gambling game of China. There are 3 dice, each die has 6 faces that score from 1 to 6. Accordingly, the total score can be varied from 3 to 18. The probability of scores 3-10 and 11-18 are 108/216 that can be adopted for the binary discretization technique of PSO. In this new technique, the first die represents the current position of a particle, the second die represents the best position ever visited by the particle, and the third die represents the best position ever visited by the swarm. The mechanism of the discretization technique is calculated as below.

#### *3.1.1 Particle velocity*

Let $V_i^k = [v_i^k(1), v_i^k(2), \dots, v_i^k(j)]$ be the velocity vector of particle $i$ that has $j$ dimensions, and $a_i^k(x_i^k(j))$ be an acceleration of particle $i$ at dimension $j$. Likewise, $a_i^k(pbest_i^k(j))$ is an acceleration of personal best positon of particle $i$ at dimension $j$, and $a_i^k(gbest_i^k(j))$ is an acceleration of global best position of the swarm at dimension $j$. Each acceleration value is computed as follows:

$$\text{if } x_i^k(j) = 1, a_i^{k+1}(x_i^k(j)) = \begin{cases} 3, r \le 0.1 \\ 4, r \le 0.4 \\ 5, r \le 0.7 \\ 6, r \le 1.0 \end{cases},$$

$$\text{if } x_i^k(j) = 0, a_i^{k+1}(x_i^k(j)) = \begin{cases} 4, r \le 0.1 \\ 3, r \le 0.4 \\ 2, r \le 0.7 \\ 1, r \le 1.0 \end{cases},$$

if $pbest_i^k(j) = 1$,

$$a_i^{k+1}(pbest_i^k(j)) = \begin{cases} 3, r \le 0.1 \\ 4, r \le 0.4 \\ 5, r \le 0.7 \\ 6, r \le 1.0 \end{cases},$$

if $pbest_i^k(j) = 0$,

$$a_i^{k+1}(pbest_i^k(j)) = \begin{cases} 4, r \le 0.1 \\ 3, r \le 0.4 \\ 2, r \le 0.7 \\ 1, r \le 1.0 \end{cases},$$

if $gbest_i^k(j) = 1$,

$$a_i^{k+1}(gbest_i^k(j)) = \begin{cases} 3, r \le 0.1 \\ 4, r \le 0.4 \\ 5, r \le 0.7 \\ 6, r \le 1.0 \end{cases},$$

if $gbest_i^k(j)=0$,

$$a_i^{k+1}\left(gbest_i^k(j)\right)=\begin{cases}4, r\le 0.1\\3, r\le 0.4\\2, r\le 0.7\\1, r\le 1.0\end{cases}.$$

The particle velocity is updated by using equation (3.1).

$$\begin{aligned}v_i^{k+1}(j) &= a_i^{k+1}\left(x_i^k(j)\right)+\\&\quad a_i^{k+1}\left(pbest_i^k(j)\right)+\\&\quad a_i^{k+1}\left(gbest_i^k(j)\right).\end{aligned} \tag{3.1}$$

In this technique, it is not needed to add the inertia weight.

### *3.1.2 Particle position*

Then, the particle position is updated as follows:

$$x_i^{k+1}(j)=\begin{cases}1 & \text{if } 11<v_i^{k+1}(j)\le 18,\\0 & \text{if } v_i^{k+1}(j)\le 10,\\\text{No change} & \text{if } 10<v_i^{k+1}(j)\le 11.\end{cases} \tag{3.2}$$

The pseudocode of the proposed PSO is shown in Table 2.

**Table 2.** Pseudocode of the proposed PSO.

```
Step  Description
1     Initialization
        (a) Set $k=0$. .
        (b) Randomly initialize the position of the $m$ particles in binary numbers.
        (c) Randomly initialize the velocity of the $m$ particles in binary numbers.
        For $i=1,\dots,m$ do $pbest_i^k = X_i$.
        Set $gbest^k = \arg\min_{i\in\{1,\dots,m\}} f(X_i)$.
2     Terminate Check. If the termination criteria hold stop. Then, the outcome of the algorithm will be $gbest$.
3     For $i=1,\dots,m$ do
        Calculating acceleration values.
          (a) Calculate $a_i^{k+1}\left(x_i^k(j)\right)$.
          (b) Calculate $a_i^{k+1}\left(pbest_i^k(j)\right)$.
          (c) Calculate $a_i^{k+1}\left(gbest_i^k(j)\right)$. .
        Updating particle swarm.
          (a) Update the velocity $V_i^k$ using Eq. (3.1).
          (b) Update the position $X_i^k$ using Eq. (3.2).
        Updating $pbest_i$ and $gbest$.
          (c) If $f(X_i)\le f(pbest_i)$ then $pbest = X_i$.
          (d) If $f(pbest_i)\le f(gbest)$ then
                $gbest = pbest_i$.
      End For.
4     Set $k=k+1$.
5     Go to step 2.
```

In this technique, the information about the best position so far of personal experience and the best position so far of the swarm is used to calculate the direction that the particle is going to. This concept exploits the original idea of swarm behavior.

## 3.2 Computational experiments

The uncapacitated fixed-charge location problems were used to evaluate the new technique's performance. The problem is a kind of mixed-integer linear programming and also falls into an $\mathcal{NP}$*-hard* problem. Its mathematical model, encoding scheme, decoding scheme, and experimental design are described below.

### *3.2.1 Uncapacitated fixed-charge location problem*

An uncapacitated fixed-charge location problem is a classical problem in logistics and supply chain management. The purpose of the problem is to select facility locations to minimize the total cost of setting facilities and transporting products from facilities to customer points [21]. Let $S$ be the set of customers and $T$ the set of potential facility locations. There are $h_s$ annual demand of customer $s, f_t$ fixed cost to operate the facility at location $t$, and $c_{st}$ cost of delivery one unit of product from facility $t$ to customer $s$. The mathematical model is shown as follows:

minimize $\sum_{t\in T} f_s x_t + \sum_{s\in S}\sum_{t\in T} h_s c_{st} y_{st}$ (3.3)

subject to

$$\sum_{t\in T} y_{st} = 1, \forall s \in S, \quad (3.4)$$

$$y_{st} \le x_t, \forall s \in S, \forall t \in T, \quad (3.5)$$

$$x \in \{0,1\}, \forall t \in T, \quad (3.6)$$

$$y_{st} \ge 0, \forall s \in S, \forall t \in T. \quad (3.7)$$

The objective function, Eq. (3.3), is the total cost of facility setting and transportation costs. The requirement of each customer must be fully fulfilled by facilities, as in Eq. (3.4). Equation (3.5) forces a closed facility does not to serve customers. Equations (3.6) and (3.7) are restricted constraints.

### *3.2.2 Encoding and decoding schemes*

A key point factor of applying a metaheuristic to real-world problems is the technique of encoding and decoding. In this study, we made use of a straightforward but highly efficient technique. The vector of particle position has elements equals to the potential facility sites.

Accordingly, element 1 represents the potential sites 1, element 2 represents the potential sites 2, and so on. If element 1 is labeled 1, it means open site 1. Conversely, if element 1 is labeled 0, it means site 2 is closed, as shown in Fig. 4.

Please note that the allocation scheme is based on the nearest opened facility. Since the capacity is unlimited, the basic rule is simple to deploy.

| Element $j$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Vector $i$ | 0 | 1 | 1 | 0 | 1 | 0 |

**Fig. 4.** Encoding and decoding schemes.

### *3.2.3 Experimental design*

The algorithm was coded in Python and ran on a laptop with Intel Core i7 2.6 GHz processor and RAM 4 GB DDR4. The benchmark problems were drawn from the OR-Library [22]. There were 10 selected problems vary from small to large problems, as shown in Table 3.

Please note that the number of decision variables is the number of $x_t$ and $y_{st}$ in the mathematical model. However, since we deploy a simple rule on the customer allocation part, the nearest facility allocation rule, we may ignore these decision variables.

For all experiments, the population size was 40 particles and the number of iterations was 1,000. Additionally, each experiment was executed with 40 independent runs to extinguish random discrepancy. Other basic parameters, if needed, were set as $\omega = 0.7, \phi_1 = \phi_2 = 0.2$. These settings were designed by conducting a design of experiment at our preliminary study phase.

**Table 3.** Benchmark problems.

| Problem | $T$ | $S$ | Decision variables |
|---|---|---|---|
| cap41 | 16 | 50 | 816 |
| cap51 | 16 | 50 | 816 |
| cap63 | 16 | 50 | 816 |
| cap74 | 16 | 50 | 816 |
| cap83 | 25 | 50 | 1,275 |
| cap84 | 25 | 50 | 1,275 |
| cap91 | 25 | 50 | 1,275 |
| cap122 | 50 | 50 | 2,550 |
| cap131 | 50 | 50 | 2,550 |
| cap134 | 50 | 50 | 2,550 |

Remark: $T$ = set of potential sites, $S$ = set of customers.

We selected the competitive binary converse techniques: the sigmoid function, the hyperbolic tangent function, and a modern technique of Khanesar et al. [13].

## 4. Results and Discussion

The test results in terms of the mean final best value and its standard deviation are concluded. Furthermore, we also showed the dispersion index as the coefficient of variation of each set of experiments. In our study, the problems cap41 to cap 74 fell into small-size problems, cap83 to cap91 fell into medium- size problems, and cap122 to cap134 fell into large-size problems.

As mentioned, the dispersion index is a focus of our attention. It helps us indicate the performance of the technique in terms of its consistency. We modified the variance to mean ratio ($vmr$) as follows.

$$vmr = \frac{\sigma^2}{\bar{x}} \times 100. \tag{3.8}$$

Equation (3.8) is the modified variance to mean ratio by multiplying 100 to extend its value.

**Table 4.** Experiment results.

| Problem | Sigmoid | | | Hyperbolic Tangent | | | Khanesar et al. [11] | | | Proposed Technique | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\bar{x}$ | $\sigma$ | $vmr$ | $\bar{x}$ | $\sigma$ | $vmr$ | $\bar{x}$ | $\sigma$ | $vmr$ | $\bar{x}$ | $\sigma$ | $vmr$ |
| cap41 | **914165.3** | 0.5 | 0.00 | **914165.3** | 0.5 | 0.00 | **914165.3** | 0.5 | 0.00 | **914165.3** | 0.5 | 0.00 |
| cap51 | **943521.8** | 1.2 | 0.00 | **943521.8** | 1.0 | 0.00 | 943537.8 | 10.3 | 0.01 | 943545.8 | 25.7 | 0.07 |
| cap63 | **893631.5** | 0.8 | 0.00 | 893639.5 | 2.3 | 0.00 | 893641.5 | 15.4 | 0.03 | 893821.5 | 20.8 | 0.05 |
| cap74 | 789351.8 | 1.7 | 0.00 | 789271.6 | 0.9 | 0.00 | **789007.8** | 45.2 | 0.26 | 789351.8 | 37.3 | 0.18 |
| cap83 | 983212.3 | 25.3 | 0.07 | 973613.5 | 38.4 | 0.15 | 945321.2 | 73.5 | 0.57 | **943712.6** | 69.5 | 0.51 |
| cap84 | 893798.4 | 16.7 | 0.03 | 893732.1 | 21.5 | 0.05 | 887493.4 | 68.6 | 0.53 | **883524.7** | 89.3 | 0.90 |
| cap91 | 932741.2 | 18.5 | 0.04 | 933674.5 | 20.7 | 0.05 | 925862.5 | 80.5 | 0.70 | **921733.6** | 100.2 | 1.09 |
| cap122 | 832567.7 | 12.4 | 0.02 | 834267.3 | 13.6 | 0.02 | **829425.7** | 73.5 | 0.65 | 830475.1 | 82.4 | 0.82 |
| cap131 | 993451.3 | 18.5 | 0.03 | 989754.8 | 22.4 | 0.05 | 984251.9 | 49.8 | 0.25 | **980374.3** | 72.7 | 0.54 |
| cap134 | 987278.8 | 15.3 | 0.02 | 985217.3 | 32.4 | 0.10 | 965381.4 | 68.9 | 0.49 | **941362.8** | 78.1 | 0.65 |

Remark: Bold values indicate better results than other techniques.

From Table 4, the bold values show the better results among the competitive technique of each problem set. It can be seen that for the small-size problems, the sigmoid function dominated other techniques. It yielded the best solution for cap41, cap51, and cap63 while other techniques yielded the best solution on some problems. However, on problem cap74, Khanesar et al.'s technique yielded the best solution even though this problem fell into the small-size problem. Notice that its variance to mean ratio $(vmr)$ was 0.26 which was the highest one on cap74.

In the category of medium-size problems, our proposed technique outperformed all other techniques in this study. The proposed technique yielded the minimum fitness values and better than the competitive techniques varied from 0.17% to 4.19%. Nevertheless, a drawback of our technique was the consistency. The values of variance to mean ratio were much higher than other techniques. It varied from 0.51 to 1.09 while Khanesar et al.'s technique varied from 0.53 to 0.57. The other two traditional techniques, the sigmoid function, and the hyperbolic tangent function, were lower than that.

The proposed technique also worked better than others in the large-size problems. It yielded the best solution on cap131 and cap134. Even though Khanesar et al.'s technique dominated on cap122, the result from our technique was not much different, just 0.13%.

This means the proposed technique cultivates the benefit of the swarm behavior that each other particle communicates their best positions. They could keep their exploration during the search course while the two traditional techniques omitted this information. The technique of Khanesar et al. was the good one. It could balance between exploration and exploitation. The $vrm$ values told us that this technique worked well by yielding consistent results.

On the other hand, our proposed technique showed the ability of exploration that is suitable for large and complicated problems. The proposed technique kept searching for the best solution probabilistically. The drawback was it deteriorated the consistency. Accordingly, a practical analyst should be aware of this when using the proposed technique in an algorithm.

We conducted further statistical testing to investigate an interesting perspective. We needed to know the difference between Khanesar et al' s technique and the proposed technique on the medium-size and large-size problems. To do so, we used paired t-test to test on their mean of cap83 to cap134. The null hypothesis was that the results from the two techniques were not different and the alternative hypothesis was that the results from the two techniques were different.

$$H_0 : \mu_D = 0$$
$$H_1 : \mu_D \neq 0$$

The paired t-test result is shown in Fig. 5.

```
Paired T for Khanesar one-Proposed one

                 N    Mean  StDev  SE Mean
Khanesar one     6  922956  56694    23145
Proposed one     6  916864  52826    21566
Difference       6    6092   9008     3678


95% CI for mean difference:(-3361, 15546)
T-Test of mean difference = 0 (vs not = 0):
T-Value = 1.66  P-Value = 0.159
```

**Fig. 5.** Paried t-test result.

By setting $\alpha = 0.05$, the conclusion can be drawn that since $t_0 = 1.66$ and $p$-value = 0.159, the discretization techniques yield different results. Specifically, the data indicate that the Khanesar et al.' s technique yields, on average, higher fitness values than does the proposed technique.

## 5. Conclusion

A new technique of discretization for metaheuristics was proposed. The proposed technique does not diminish the main swarm behavior, the information of the particle position, and their best so far positions. The new technique showed that its search exploration was good and dominated other competitive techniques in this study; specifically, the two traditional techniques, the sigmoid function, and the hyperbolic tangent function. The new technique was comparable to Khanesar et al.' s technique. From statistical tests, our technique worked better on both medium- size and large- size problems.

In conclusion, the main contribution of our study is showing that a discretization technique affects the performance of continuous search algorithms. Designing and selecting a good discretization technique can yield a good optimization solution, reasonably.

To this point, we are interested to investigate the performance of our proposed technique combined with PSO on other combinatorial optimization problems such as assignment problems, vehicle routing problems, production scheduling problems, etc. The applications of the proposed technique on other continuous search algorithms such as gravitational search algorithms, harmony search algorithms, and artificial neural networks are the subject of a further study.

## Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions in the earlier version.

This research project was granted by Silpakorn University Research, Innovation and Creative Fund.

## References

[1] Foulds LR. Optimization techniques: an introduction. Springer Science & Business Media; 2012 Dec 6.

[2] Bertsimas D, Tsitsiklis JN. Introduction to linear optimization. Belmont, MA: Athena Scientific; 1997 Jan.

[3] Winston WL. Introduction to Mathematical Programming: Applications and Algorithms, Duxbury,(2002).

[4] Conforti M, Cornuéjols G, Zambelli G. Integer programming. Berlin: Springer; 2014 Nov 15.

[5] Chae J, Regan AC. Layout design problems with heterogeneous area constraints. Computers & Industrial Engineering. 2016 Dec 1;102:198-207.

[6] Kennedy J, Eberhart RC. A discrete binary version of the particle swarm algorithm. In1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation 1997;5:4104-8.

[7] Kennedy J, Eberhart R. Particle swarm optimization. InProceedings of ICNN'95-international conference on neural networks 1995;4:1942-8.

[8] Du KL, Swamy MN. Particle swarm optimization. InSearch and optimization by metaheuristics 2016. p.153-73.

[9] Osadciw, L. & Veeramachaneni, K. Particle swarms for continuous, binary, and discrete search space. In: Particle swarm optimization. 1st ed. In- Tech, 2009, p.451-60.

[10] Strasser S, Goodman R, Sheppard J, Butcher S. A new discrete particle swarm optimization algorithm. InProceedings of the Genetic and Evolutionary Computation Conference 2016. p.53-60.

[11] Choi H, Ohmori S, Yoshimoto K, Ohtake H. Improvement of particle swarm optimization: Application of the mutation concept for the escape from local minima. In2010 8th International Conference on Supply Chain Management and Information 2010. p.1-5.

[12] Pornsing C, Sodhi MS, Lamond BF. Novel self- adaptive particle swarm optimization methods. Soft Computing. 2016 Sep;20(9):3579-93.

[13] Khanesar MA, Teshnehlab M, Shoorehdeli MA. A novel binary particle swarm optimization. In2007 Mediterranean conference on control & automation 2007. p.1-6.

[14] Tsai MS, Wu WC. A novel binary coding particle swarm optimization for feeder reconfiguration. chapter; 2009 Jan 1.

[15] Ponnambalam SG, Jawahar N, Chandrasekaran S. Discrete particle swarm optimization algorithm for flowshop scheduling. Particle Swarm Optimization. 2009 Jan 1:397.

[16] Taşgetiren MF, Liang YC. A binary particle swarm optimization algorithm for lot sizing problem. Journal of Economic and Social Research. 2003 Jul 1;5(2):1-20.

[17] Wang L, Wang X, Fu J, Zhen L. A novel probability binary particle swarm optimization algorithm and its application. Journal of software. 2008 Dec;3(9):28-35.

[18] Tran B, Xue B, Zhang M. A new representation in PSO for discretization-based feature selection. IEEE Transactions on Cybernetics. 2017 Jun 23;48(6):1733-46.

[ 19] Garcia S, Luengo J, Sáez JA, Lopez V, Herrera F. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. IEEE Transactions on Knowledge and Data Engineering. 2012 Feb 16;25(4):734-50.

[20] Shang L, Zhou Z, Liu X. Particle swarm optimization- based feature selection in sentiment classification. Soft Computing. 2016 Oct;20(10): 3821-34.

[21] Snyder LV, Shen ZJ. Fundamentals of supply chain theory. Hoboken: Wiley; 2011 Aug 23.

[22] Beasley, J., 2021. OR-LIBRARY. [online] People. brunel. ac. uk. Available at: <http://people.brunel.ac.uk/~mastjjb/jeb/info.html> [Accessed 15 December 2020].