Soft Comput (2018) 22:783–795
https://doi.org/10.1007/s00500-016-2383-8

METHODOLOGIES AND APPLICATION

CS-PSO: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems

Xiaolong Xu[1] · **Hanzhong Rong**[1] · **Marcello Trovati**[2] · **Mark Liptrott**[2] · **Nik Bessis**[2]

Published online: 3 October 2016

Abstract Combinatorial optimization problems are typically NP-hard, due to their intrinsic complexity. In this paper, we propose a novel chaotic particle swarm optimization algorithm (CS-PSO), which combines the chaos search method with the particle swarm optimization algorithm (PSO) for solving combinatorial optimization problems. In particular, in the initialization phase, the priori knowledge of the combination optimization problem is used to optimize the initial particles. According to the properties of the combination optimization problem, suitable classification algorithms are implemented to group similar items into categories, thus reducing the number of combinations. This enables a more efficient enumeration of all combination schemes and optimize the overall approach. On the other hand, in the chaos perturbing phase, a brand-new set of rules is presented to perturb the velocities and positions of particles to satisfy the ideal global search capability and adaptability, effectively avoiding the premature convergence problem found frequently in traditional PSO algorithm. In the above two stages, we control the number of selected items in each category to ensure the diversity of the final combination scheme. The fitness function of CS-PSO introduces the concept of the personalized constraints and general constrains to get a personalized interface, which is used to solve a personalized combination optimization problem. As part of our evaluation, we define a personalized dietary recommendation system, called *Friend*, where CS-PSO is applied to address a healthy diet combination optimization problem. Based on *Friend*, we implemented a series of experiments to test the performance of CS-PSO. The experimental results show that, compared with the typical HLR-PSO, CS-PSO can recommend dietary schemes more efficiently, while obtaining the global optimum with fewer iterations, and have the better global ergodicity.

Keywords Combinatorial optimization · Particle swarm optimization · Chaos search · Personalization recommendation

Communicated by V. Loia.

✉ Marcello Trovati
marcello.trovati@edgehill.ac.uk

Xiaolong Xu
xuxl@njupt.edu.cn

Hanzhong Rong
1014041127@njupt.edu.cn

Mark Liptrott
mark.liptrott@edgehill.ac.uk

Nik Bessis
nik.bessis@edgehill.ac.uk

[1] College of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

[2] Department of Computing, Edge Hill University, Ormskirk, Lancashire L39 4QP, UK

1 Introduction

Particle swarm optimization algorithm (PSO) is a heuristic optimization technology, presented by Kennedy and Eberhart (1995), which mimics the swarm behavior of bird flocks in performing their tasks, and to discover an optimal solution based on an objective function (Kennedy and Eberhart 1995; Eberhart and Kennedy 1995; Chang et al. 2014). With fewer parameters, PSO algorithm can achieve a faster convergence, while being simpler and easier to implement (Xu et al. 2012). PSO has already been applied to many fields, such as electric power systems, job scheduling of workshops, wireless sensor networks, route planning, and robotics (Lei

2014; Kumari and Jha 2014; Yao et al. 2012; Liao et al. 2012; Lee and Kim 2013). However, the performance of PSO still has space for improvement. For example, due to the fast convergence of PSO, it is easy to fall into local optima in solving multimodal optimization problems, potentially leading to the premature convergence of particle swarms. In the initialization and updating phase, the stochastic strategy of PSO generates a group of particles and finds the optimal solution through multiple iterations. During the iterations, the positions and velocities of particles are randomly updated, resulting in a low computational efficiency. There are mainly two ways to improve performance of the PSO: the first adjusts the parameters and procedure of PSO, such as dynamically adjusting the search step length and optimizing the update strategy of the particles (Chi et al. 2011; Zhao et al. 2014). Another approach would be the combination with other intelligent optimization algorithms, such as the genetic algorithm (GA), and the simulated annealing algorithm (Sharma and Singhal 2015; Nancharaiah and Mohan 2013). Most related research Guo et al. (2014), Zhu et al. (2014), Sorkunlu et al. (2013), Shi and Eberhart (1998), Elbedwehy et al. (2012) about the improvement in PSO now mainly focuses on the continuous optimization problems, while the combinatorial optimization problems (e.g., the combination of integer programming and the 0/1 knapsack) do not attract enough attentions, and the current research results are usually suitable to certain scenarios, which are not pervasive.

In order to solve combinatorial optimization problems more efficiently, we propose a novel chaotic particle swarm optimization algorithm (CS-PSO). The main contributions of this paper are as follows. First of all, the chaos initialization and the chaos perturbing of the chaos search method are introduced into PSO in place of the random initialization and the random perturbing. The ergodicity, regularity, and randomness of the chaos search method can contribute to address the PSO issues, including the local optimum and the poor search efficiency. In the initialization phase, the priori knowledge of the combination optimization problem is used to optimize the initial particles. Furthermore, the quality of the particles and the search efficiency of the algorithm are improved. In the chaos perturbing phase, a brand-new set of perturbing rules is presented to perturb the velocities and positions of particles sufficiently to realize the ideal global search capability and adaptability, effectively solving the premature problem of particles. Subsequently, we designed the fitness function of CS-PSO, which utilizes the concept of the personalized constraints and general constrains to produce a personalized interface, which is used to solve a personalized combination optimization problem. Finally, we built a personalized dietary recommendation system, *Friend*, which is based on CS-PSO to address a healthy diet combination optimization problem. *Friend* is able to recommend more reasonable dietary schemes, which proves that CS-PSO has an enhanced performance compared to other improved PSO algorithms, such as the typical PSO for generating healthy lifestyle recommendations (HLR-PSO) (Pop et al. 2013).

The rest of the paper is organized as follows: Sect. 2 presents the related works, Sect. 3 discusses CS-PSO in detail, and Sect. 4 describes the prototype personalized dietary recommendation system called *Friend* applied with CS-PSO. Experiments and performance analysis are presented in Sect. 5, and finally, Sect. 6 concludes the paper by summarizing the main contributions of this paper and commenting on future directions of our work.

2 Related work

PSO is a bio-inspired optimization meta-heuristic, which is inspired by the foraging behavior exhibited by birds, which is based on the assumption that a flock of birds are randomly distributed in an area with only one piece of food, as shown in Fig. 1a. The dot on the tree represents the available food, and its position is unknown to each bird, although they know their distance from it. Furthermore, the nearest bird to the food can notify other birds to fly to it. The food is assumed to be the optimal value, as shown in Fig. 1b, where each bird is seen as a particle, and the distance between a bird and the food is a value of the objective function. Therefore, the birds flock foraging process can be defined as a function optimization process. In Fig. 1b, X_i is the closest particle to the goal, and it is set as the current global optimal particle. Its distance from goal is $Nbest_i$, which is the global optimal value (Kennedy and Eberhart 1995; Eberhart and Kennedy 1995). The main idea of PSO (Pop et al. 2013) is that in a set of particles, each particle is defined with a position and a velocity, searching for the global optimum of an NP-hard problem. The particles iteratively update their positions according to their individual local optimal position and the global optimal position visited so far. The new position of a particle (e.g., particle i) is defined as:

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{1}$$

where t is the current (temporal) status, $t+1$ is the status post-updating, $X_i(t)$ is the current position of the particle, and $V_i(t+1)$ is the new velocity of the particle. Note that time difference $\Delta t = (t+1) - t$ is indeed a time unit.

The velocity of particle i is defined as:

$$V_i(t+1) = wV_i(t) + c_1 r_1 \left(X_i^{\mathrm{p}} - X_i(t)\right) + c_2 r_2 (X^{\mathrm{g}} - X_i(t)), \tag{2}$$

where $V_i(t)$ is the current velocity of the particle, X_i^{p} is the best position so far visited by the particle (i.e., the local best

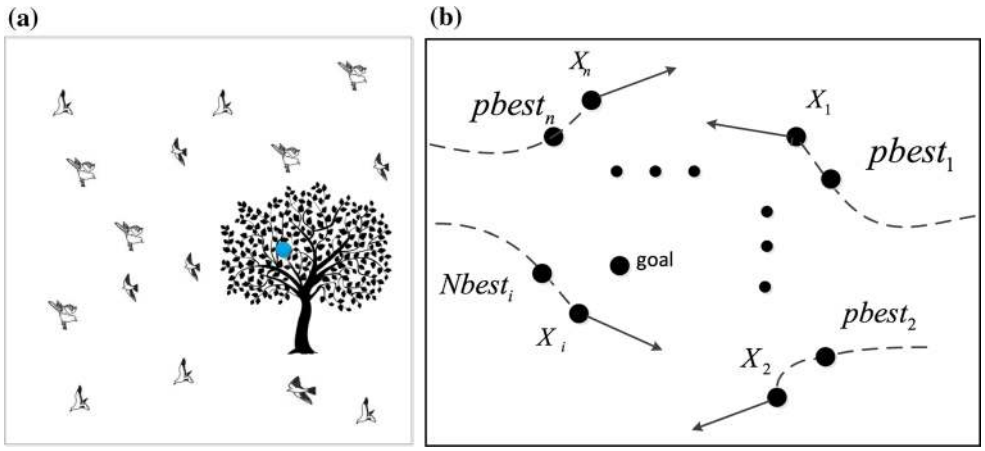


Fig. 1 Simulation of bird flocks foraging

position), X^g is the global best position so far visited by a particle at the swarm level, and w, c_1, and c_2 are constants that weight the importance of each component of the velocity. Finally, r_1 and r_2 are the random values within [0, 1].

There is much research focusing on the improvement in the performance of the original PSO. In Wen et al. (2013), the authors propose a new modified particle swarm optimization algorithm based on sub-particle circular orbit and zero-value inertial weight (MDPSO). MDPSO utilizes the trigonometric function based on nonlinear dynamic learning factors and on a prediction method of population premature convergence, which can achieve a better balance between the local exploring ability and the global converging ability of particles (Wen et al. 2013). However, MDPSO is mainly suitable for solving the composition optimization problem of Web service, and it is not, therefore, universal. In Gao et al. (2005), the authors propose a general particle swarm optimization model (GPSO), which can be naturally extended to solve discrete and combinatorial optimization problems. GPSO uses the genetic updating operator, further improving the quality of solution and the stability of convergence, and significantly saving the computational cost (Gao et al. 2005). However, the genetic updating operator brings randomness into GPSO, which cannot guarantee the diversity of the final solution. In Guo et al. (2011), the authors propose a hybrid particle swarm optimization algorithm with the Fiduccia-Mattheyses algorithm (FM), inspired by GA, utilizing the regeneration mechanism of particle's position of discrete particle swarm optimization (DPSO). In particular, it is based on genetic operations to update the position of the particle defined as two-point crossover and random two-point exchange mutation operators to avoid generating infeasible solutions. To improve the ability of local exploration, FM is applied to update its position. A mutation strategy is also built into the proposed algorithm to achieve better diversity and break away from local optima (Guo et al. 2011). However, similar to Wen et al. (2013), the algorithm is not universal and cannot solve the multi-objective optimization problems. In Ibrahim et al. (2012), the authors propose a novel multistate particle swarm optimization algorithm (MSPSO) to solve discrete combinatorial optimization problems, which is different from the binary particle swarm optimization algorithm (BinPSO). In MSPSO, each dimension variable of each particle can attain various states, and it has been applied to two benchmark instances of the traveling salesman problem (TSP). The experimental results show that MSPSO outperforms BinPSO in solving the discrete combinatorial optimization problem (Ibrahim et al. 2012). However, MSPSO utilizes the concept of multistate, leading to the exponentially growing requirements of storage space and computation time. Therefore, the efficiency is affected when MSPSO is applied to solving high-dimensional combinatorial optimization problems. In Gao and Xiei (2004), the authors attempt to apply chaos search method to PSO, while using its ergodicity, regularity, and randomness to search the current global best particle in the chaotic way, replacing a stochastic selected individual from the current "population." The performance of PSO is improved with the chaos search method, which motivates our work. The evolution process is quickened, and the abilities to seek the global optimum, the convergence speed, and accuracy are all improved (Gao and Xiei 2004). In Wang and Wu (2011) and Yang et al. (2015), improved PSO algorithms with the chaos search method are presented and applied to the optimization of logistics distribution route and vehicle routing problem with specific time windows, respectively. However, these results all simply adopt the chaos search method, not further improving the mechanism of chaos initialization and chaos perturbing or providing the personalized interface. Therefore, the diversity of the final solution cannot be guaranteed, and the search efficiency is still unsatisfactory (Wang and Wu 2011; Yang et al. 2015). In Sfrent and Florin Pop (2015), the authors introduce a simulation infrastructure

for building/analyzing different types of scenarios, which allows the extraction of scheduling metrics for three different algorithms, namely the asymptotically optimal one, FCFS and a traditional GA-based algorithm. These are combined them into a single hybrid algorithm, addressing asymptotic scheduling for a variety of tasks related to big data processing platforms. A distributed and efficient method for optimizing task assignment is introduced in Iordache et al. (2006), which utilizes a combination of genetic algorithms and lookup services. In Bessis et al. (2012), an algorithm based on a variety of e-infrastructure nodes exchanging simple messages with linking nodes is discussed, with the aim to improve the energy efficiency of the network performance.

3 Chaotic particle swarm optimization algorithm

3.1 Basic idea

The current PSO algorithms designed for solving combinatorial optimization problem generally exhibit the following issues:

- Most PSO algorithms are only suitable for one particular scenario, and they are not universal.
- Most PSO algorithms are not based on multi-objective, or do not provide a personalized interface. So, they cannot effectively solve discrete, multi-objective, and personalized combinatorial optimization problems.
- With the increasing of the particle dimension, the requirements of storage space and computation time will grow exponentially, which will lower the efficiency when solving the high-dimensional combinatorial optimization problem.

The CS-PSO proposed here adopts the chaos search method (Lorenz 2005). The chaos initialization and the perturbation of the chaos search method are used instead of the random initialization and the random perturbing. In the initialization phase, CS-PSO optimizes the initial particles according to the characteristics of combination optimization problems. Via item classification, similar items are grouped into the same category, thus reducing the number of combinations. Therefore, it is possible to enumerate all combination schemes and improve the search efficiency. In the chaos perturbing phase, a new set of perturbing rules is designed to perturb velocities and positions of particles sufficiently, so that CS-PSO has good global search capability and adaptability, and the premature convergence problem of particles is also effectively solved. In the above two phases, CS-PSO controls the number of selected items in each category to ensure the diversity of the final combination scheme. The fitness function of CS-PSO utilizes the concept of the personalized constraints and general constrains to get a personalized interface, which can be used to solve the corresponding personalized combinatorial optimization problem.

3.2 Chaos search method

Definition 1 (*Chaos search*) Chaos search is the random movement with pseudorandomness, ergodicity, and regularity, which is determined by a deterministic equation (Lorenz 2005).

Through the chaos iteration, a set of random sequences with the ergodicity and the pseudorandomness are generated. Usually, the logistic mapping equation (Dong et al. 2013) is used to generate pseudorandom sequences:

$$Z : \alpha_{n+1} = \mu\alpha_n(1-\alpha_n), \quad n = 0, 1, 2, \ldots \tag{3}$$

where Z is a chaotic variable, corresponding to α_n , and μ is the control parameter. If $\mu = 4$, the logistic map will show entirely chaotic dynamics, and the trajectory of chaotic variable are dense over the whole search space. We assume that the initial value of Z, namely α_0, is not equal to $0, 1.25, 0.5, 0.75, 1$, otherwise it would be eventually periodic.

3.3 Model of combinatorial optimization problem

Definition 2 (*Combinatorial optimization*) Combinatorial optimization refers to the process of optimizing an object via the combination of a finite set of components.

A typical case of combinatorial optimization is the "quality–cost" model of manufactured products, where a specific product consists of m components, and each of them can be chosen from a variety of options. The parameters of each optional component include a weight representing its quality and the index of cost, with the constraint that the total expenditure of the product does not exceed the available budget. There are a variety of examples where combinatorial optimization plays a crucial role. These include, for example, the assembling of the different parts of a car, such as an engine, chassis, tires, transmission, and electrical equipment, while optimizing quality versus cost. The "quality–cost" model of combinatorial optimization problem is defined as

$$\begin{aligned} &\max \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{i,j}x_{i,j} \\ &\text{such that} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{i,j}x_{i,j} \le O \\ &\sum_{j=1}^{n_i} x_{i,j}, \quad \forall\, i \in \{1, 2, \ldots, m\} \end{aligned} \tag{4}$$

where

- i is the index of category,
- j is the index of item,
- m is the total number of categories,
- n_i is the total number of items in the i-th category,
- $w_{i,j}$ is the weight of the j-th item in the i-th category,
- $c_{i,j}$ is the cost of the j-th item in i-th category.
- O is the object cost, which means the manufacturing cost shall not exceed the object cost and the quality of the product shall be the optimal; $x_{i,j} \in \{0, 1\}, \forall i$ is the mapping value of the item.

Note that if the j-th item in the i-th category is selected, then $x_{i,j} = 1$, otherwise $x_{i,j} = 0$. $\sum_{j=1}^{n_i} x_{i,j} = 1$ implies that only one item is selected from each category.

3.4 Chaos initialization

Chaos initialization refers to the process of a chaotic variable of the logistic map, which randomly identifies a value as its initial value of particle.

The parameters of chaos initialization are as follows:

1. All items are divided into m categories, which are defined as vectors, B_i, for $i = 0, 1, \ldots, m-1$, for category i.
2. The total number of items in category i is defined as N_i, for $i = 0, 1, \ldots, m-1$, which implies that $B_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,N_i-1})$.
3. According to the above points, the position of particle i can be obtained, which is defined as a vector $X_i = (B_0, B_1, \ldots, B_{m-1})$. The dimension of particle i is $\sum_{i=0}^{m-1} N_i$.

Suppose that only one item is selected from each category. Therefore, m random values are sequentially generated within the interval [0, 1], and each of them is mapped onto an item of each category. Via these m random values, the position of the first particle can be obtained. Take category B_0 as an example, so that the chaos initialization process is as follows:

1. Suppose there are N_0 items in B_0, the chaos search space [0, 1] is divided into N_0 subspaces.
2. The random function is used to generate a random number between 0 and 1, described as $k_{0,0}$, which is assigned to the chaotic variable as the initial value of the B_0 category.
3. The parameter $k_{0,0}$ is subsequently assessed to identify which subspace it belongs to. Supposing that $k_{0,0}$ belongs to the μ-th subspace, $x_{0,\mu} = 1$, and others variables of B_0 are all initialized to 0. It means that the μ-th item is selected in $B_0 = (0, 0, \ldots, 1, \ldots, 0)$.

By repeating the above procedure m times, m random values $k_{0,0}, k_{0,1}, \ldots, k_{0,m-1}$ are generated sequentially, and each random value is mapped to a corresponding item of each category. The initializations of other categories $B_1, \ldots, B_{m-1}$ can be completed in the same way. $B_0, B_1, \ldots, B_{m-1}$ are combined together to get vector X_i. Supposing that there are n particles, the $n \times m$ initialization chaotic variables matrix K can be defined as:

$$K = \begin{pmatrix} k_{0,0} & \ldots & k_{0,m-1} \\ \vdots & \vdots & \vdots \\ k_{n-1,0} & \ldots & k_{n-1,m-1} \end{pmatrix}$$

In the initialization phase, the velocity V_i and the local best position P_i of particle i are all equal to X_i, that is:

$$X_i = V_i = P_i \quad (i = 0, 1, \ldots, n-1) \tag{5}$$

3.5 Chaos perturbing

Definition 3 (*Chaos perturbation*) In the updating process of particles, their velocities and positions will be perturbed sufficiently and the search space will be traversed as sufficient as possible.

Definition 4 (*Fitness value*) Fitness value is a value obtained through a fitness function, which is a quantitative indicator and used to evaluate the advantage and disadvantage of individual.

Take particle i as an example. The parameters of chaos perturbing are as follows:

1. The local best fitness value is defined as $f_i (i = 0, 1, \ldots, n-1)$.
2. The global best fitness value is defined as F.
3. The local best position is defined as $P_i (i = 0, 1, \ldots, n-1)$.
4. The global best position is defined as G.

Subsequently, the positions of n particles are obtained, and their fitness value is initialized to 0. In particular, a higher value of the fitness value will have a positive impact on the position of particles. During the updating process of f_i and $P_i (i = 0, 1, \ldots, n-1)$, F and G can be obtained.

We define two velocity vectors V_i^{p} and V_i^{g}, where $V_i^{\mathrm{p}} = X_i^{\mathrm{p}} - X_i$, $V_i^{\mathrm{g}} = X_i^{\mathrm{g}} - X_i$. Consequently, the new velocity of the particle i is updated with

$$V_i(t+1) = wV_i(t) + c_1 r_1 V_i^{\mathrm{p}} + c_2 r_2 V_i^{\mathrm{g}} \tag{6}$$

The subtraction between two positions, for an example between X_i^{p} and X_i, is defined as

$$X_i^{\mathrm{p}} - X_i = (v_{i,0}^{\mathrm{p}}, v_{i,1}^{\mathrm{p}}, \ldots, v_{i,j}^{\mathrm{p}}, \ldots) \tag{7}$$
$$v_{i,j}^{\mathrm{p}} = \begin{cases} \texttt{Rand(1)} & \text{if } x_{i,j}^{\mathrm{p}} = x_{i,j}; \\ 0 & \text{otherwise.} \end{cases}$$

where Rand(1) is used to randomly generate either 0 or 1. According to the chaos initialization, we know that just one item or a few items are selected in every category. In fact, most of the variables are equal to 0. As a consequence, the updating rule of velocity is redefined as

$$v_{i,j}^{\mathrm{p}} = \begin{cases} v_{i,j}(t) & \text{if } v_{i,j}(t) = v_{i,j}^{\mathrm{p}} = v_{i,j}^{\mathrm{g}} \\ -1 & \text{otherwise.} \end{cases}$$

The addition between a position $X_i(t)$ and a velocity $V_i(t+1)$ is also redefined as:

$$\begin{aligned} X_i(t+1) &= X_i(t) + V_i(t+1) \\ &= (x_{i,0}(t+1), x_{i,1}(t+1), \ldots, x_{i,j}(t+1)) \end{aligned} \tag{8}$$
$$v_{i,j}^{\mathrm{p}} = \begin{cases} x_{i,j} & \text{if } v_{i,j}(t) = 1 \\ C(x_{i,j}) & \text{if } v_{i,j}(t+1) = 0 \\ J(x_{i,j}^{\mathrm{p}}) & \text{if } v_{i,j}(t+1) = -1 \quad \text{and } v_{i,j}(t) \neq v_{i,j}^{\mathrm{g}} = v_{i,j}^{\mathrm{p}} \\ J(x_{i,j}^{\mathrm{g}}) & \text{if } v_{i,j}(t+1) = -1 \quad \text{and } v_{i,j}(t) = v_{i,j}^{\mathrm{p}} \neq v_{i,j}^{\mathrm{g}} \\ J(x_{i,j}^{\mathrm{p}}) & \text{if } v_{i,j}(t+1) = -1 \quad \text{and } v_{i,j}(t) = v_{i,j}^{\mathrm{p}} \neq v_{i,j}^{\mathrm{g}} \\ C(x_{i,j}) & \text{if } v_{i,j}(t+1) = -1 \quad \text{and } v_{i,j}(t) \neq v_{i,j}^{\mathrm{p}} \neq v_{i,j}^{\mathrm{g}} \end{cases}$$

where

- i is the number of particle,
- j is the index of position $X_i(t)$,
- $x_{i,j}$ is the variable with index j from the i-th current position,
- $x_{i,j}^{\mathrm{p}}$ is the variable with j from the i-th local best position,
- $x_{i,j}^{\mathrm{g}}$ is the variable with j from the global best position,
- $C(x_{i,j})$ is a perturbing function of $x_{i,j}$ with j from the position of particle i, and finally,
- $J(x_{i,j}^{\mathrm{p}})$ and $J(x_{i,j}^{\mathrm{g}})$ are simple assessments based on the process initialization.

The detailed perturbing process of the function is defined as follows:

1. First, according to the parameter j, we can determine the associated item of the corresponding category $x_{i,j}$. In particular, the assertion

   "If $j \geq \sum_{i=0}^{h-1} N_i \quad \text{and} \quad j < \sum_{i=0}^{h} N_i$"

   implies that $x_{i,j}$ is associated with the chaotic variable $k_{i,h}$.
2. The logistic map is used to iterate $k_{i,h}$ once and generate a new chaotic variable $k_{i,h}$.
3. Subsequently, the subspace $k_{i,h}$ is assessed to understand which subspace it belongs to. Suppose that $k_{i,h}$ belongs to the p-th subspace, $x_{i,p} = 1$ and the others variables of the category B_h are equal to 0. If $j - \sum_{i=0}^{s} N_i = p(s \leq m)$, then $x_{i,j}(t+1) = 1$. Otherwise, $x_{i,j}(t+1) = 0$, and $B_h = (0, 0, \ldots, 1, \ldots, 0)$.

In order to ensure the suitability of the final solution, the following rules are assumed (without loss of generality, consider the variable $J(x_{i,j}^{\mathrm{p}})$):

1. If $x_{i,j}^{\mathrm{p}} = 1$, then $x_{i,j}(t+1) = 1$ and other variables of the corresponding category are assigned to 0.
2. If $x_{i,j}^{\mathrm{p}} = 0$, and $x_{i,j} = 0$, then the only action carried out is to assign 0 to $x_{i,j}(t+1)$.
3. If $x_{i,j}^{\mathrm{p}} = 0$, and $x_{i,j} = 1$, then $x_{i,j}(t+1) = C(x_{i,j})$.

3.6 Design of the fitness function

The fitness function is used to evaluate the performance of a combination scheme under certain constraints. Therefore, the properties of the fitness function will directly affect the combinatorial optimization results. More specifically, most combinatorial optimization problems are multi-constraints based.

In this article, we use both personalized constraints and general constraints, where the former are used to design the fitness function, and the latter are used as its constraints. We also combine the satisfaction of personalized constraints into the score model, and the average of scores is identified with the fitness value. The bigger the fitness value is, the higher the degree of satisfaction of personalized constraint will be, and the better the position of particle is considered to be. Suppose a combinatorial optimization problem with constraints A, B, and C, and both A and B are the personalized constraints, the C is a piece of general constraint. The fitness value is calculated as

$$F = \frac{S(A) + S(B)}{2}, \tag{9}$$

if constraint C is satisfied, where $S(A)$ is the score of A, and $S(B)$ is B. The algorithm can be applied to different scenarios, with different constraints and fitness functions.

3.7 Pseudocodes of CS-PSO

Algorithm 1 shows the pseudocodes of CS-PSO. In particular, N is the number of particle, M is the total number of categories, $N[]$ is the number of items of each category, K is

the matrix of chaotic variable, S is the personalized constraints, and C is the general constraints.

Algorithm 1 The CS-PSO Algorithm

```
Input: N; M; N[]; K; S; C
Output: G
Best_F = 100
for i = 0 to N − 1 do
  //chaos initialization
  X[i] = V[i] = P[i] = Initialize(K[i])
  //calculate the fitness value
  F[i] = ComputerFitness(X[i], S, C])
end for
//obtain the index of the global best particle
index = Get_Global_Best(F[i])
F^g = F[index]
G = X[index]
while F^g! = Best_F) and (Iterations! = MaxCount) do
  for i = 0 to N − 1 do
    //update speed of each particle
    V[i] = UpdateSpeed(X[i], V[i], P[i], G)
    //update position of each particle
    X[i] = UpdatePos(X[i], P[i], G, V[i], K[i])
    F[i] = ComputerFitness(X[i], S, C])
  end for
  index = Get_Global_Best(F[i])
  F^g = F[index]
  G = X[index]
end while
return G
```

4 A CS-PSO application: a healthy diet scheme

As a case study, we will consider a healthy diet scheme, which includes a balance of nutrients and an appropriate variety of different types of food. Clearly, this can be viewed as a typical combinatorial optimization problem. The main nutrients include water, protein, carbohydrate, lipids, dietary fiber, vitamins, and minerals. The main categories of food include staple food, vegetables, fruits, eggs, seafood, milk. In order to ensure the diversity of diet and satisfy users, the healthy diet scheme would better recommend a food item of each category that users prefer.

CS-PSO can be utilized in this context, and it involves the following elements:

1. For m types of food items category, every category is defined as a vector B_i, for $i = 0, \ldots, m-1$.
2. The total number of food items in each category is defined as N_i, for $i = 0, \ldots, m-1$.
3. The vector of the diet particle is defined as $X_i = (B_0, \ldots, B_{m-1})$ and $B_i = (x_{i,0}, \ldots, x_{i,N_i-1})$, and the dimension of a particle is $\sum_{i=0}^{m-1} N_i$

The chaos initialization and the chaos perturbing are the same as the above, while the fitness function needs to be redesigned and further developed. We consider three constraints, namely calories, nutrients, and costs. User's preferences are assumed to be a general constraint, which has to be satisfied prior to the initialization of the process. On the other hand, the constraints of calories, nutrients, costs are used as the personalized constraints. The fitness function is defined as:

$$F = \frac{(S_c + S_n + S_p)}{3} \tag{10}$$

$$S_c \text{ or } S_n = \begin{cases} 100(R/S) & \text{if } R < S \\ 100(1 - (R-S)/S) & \text{if } R \geq S. \end{cases}$$

$$S_p = \begin{cases} 100 & \text{if } R < S \\ 100(1 - (R-S)/S) & \text{if } R \geq S. \end{cases}$$

where S_c, S_n and S_p are the scores of the above four constraints, R is the recommended value corresponding to each constraint, and S is the standard value corresponding to each constraint. Figure 2a shows the score model of the constraint of calorie intake. If the calorie of recommended food is equal to S, then the score value is 100, otherwise the score value is less than 100. The bigger the distance from the S, the less the score value will be. Figure 2b shows that the score model of the constraint of nutrient intake is similar with the former. Figure 2c shows the score model of the constraint of cost. If the cost of recommended food is less than or equal to S, then the score is 100, otherwise the score is less than 100. The workflow of the diet recommendation system with CS-PSO includes the following steps, as shown in Fig. 3:

Step 1 The chaos initialization generates n diet particles. Based on the user's preferences, a food item of each category is initialized as the position of diet particle.

Step 2 According to the user's basic background, including height, weight, gender, age, and activity level, the amount of required calories and nutrients are calculated. And the constraint of cost can be provided by user. These values are used as the standard values corresponding to each constraint.

Step 3 The fitness values of all diet particles are calculated and assessed. According to the analysis of the above three constraints, calculate the score of each constraint, and then the fitness value equal with the average of three scores. The greater the fitness value, the better the diet particle. Therefore, the global optimal particle position and the corresponding fitness value can be obtained.

Step 4 The fitness value of the global best particle is assessed to evaluate whether it is optimal. If so, then end the process. Otherwise, assess whether it reaches the maximum number of iterations. If it does, then go to the end of the process. Otherwise, go to Step 5.

Step 5 The chaos perturbing component is used to update diet particles, and then, go to Step 3.

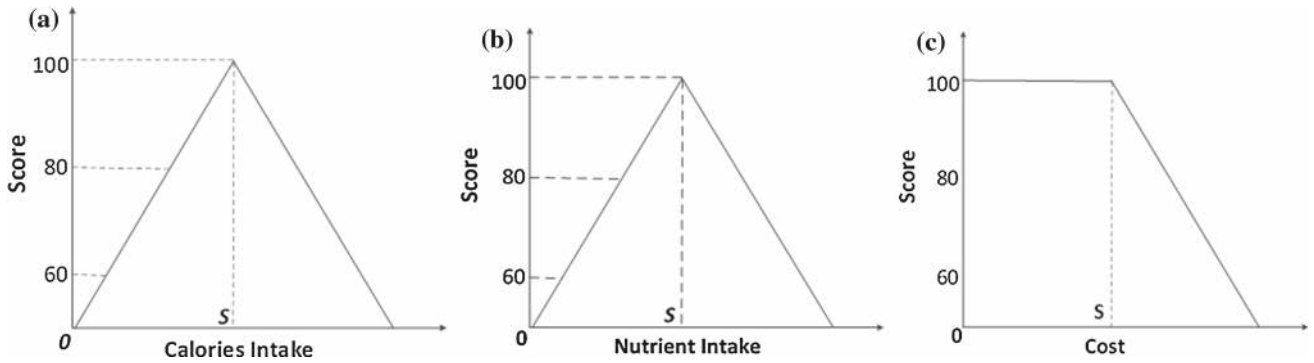


Fig. 2 Score model of personalized constraints

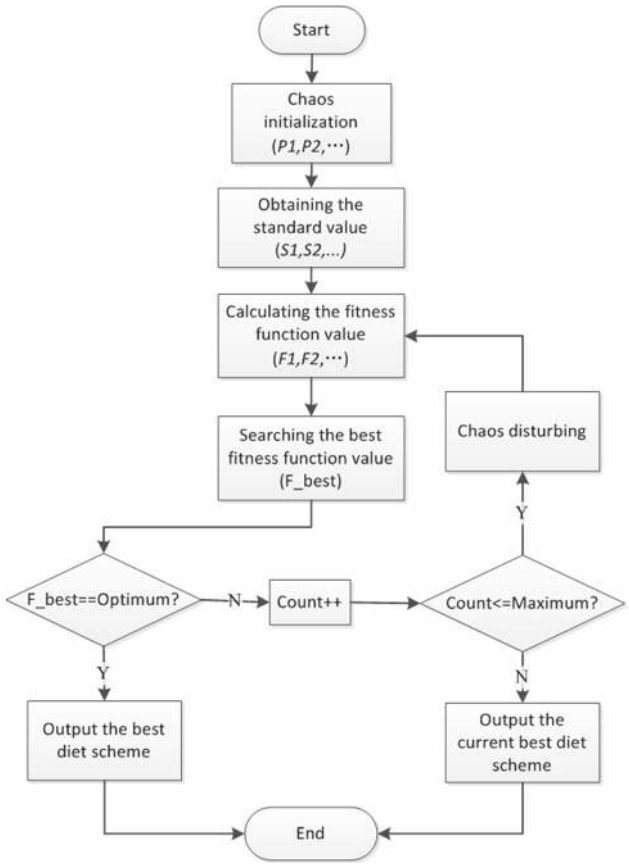


Fig. 3 Workflow of the diet recommendation system with CS-PSO

4.1 Prototype of the system

In this section, we will introduce a personalized dietary recommendation system, called *Friend*, where CS-PSO is used to address the healthy diet combination optimization problem. The system provides the interface for users to input their personal physiological data, which is used to calculate their body mass indexes (BMI), their personal standard values of calories, and standard values of nutrients. The calculation of BMI is:

Table 1 BMI for Asian adults

Figure	Standard	Related disease risk
Thinness	<18.5	Risk of developing problems such as nutritional deficiency and osteoporosis
Regular	18.5–22.9	Low risk (healthy range)
Overweight	≥23	Moderate risk of developing heart
Obesity	23–24.9	Disease, high blood pressure, stroke
Obesity—class I	25–29.9	Diabetes
Obesity—class II	≥30	High risk of developing heart disease
Obesity—class III	≥40	High blood pressure, stroke, diabetes

$$\mathrm{BMI} = \frac{w}{h^2} \tag{11}$$

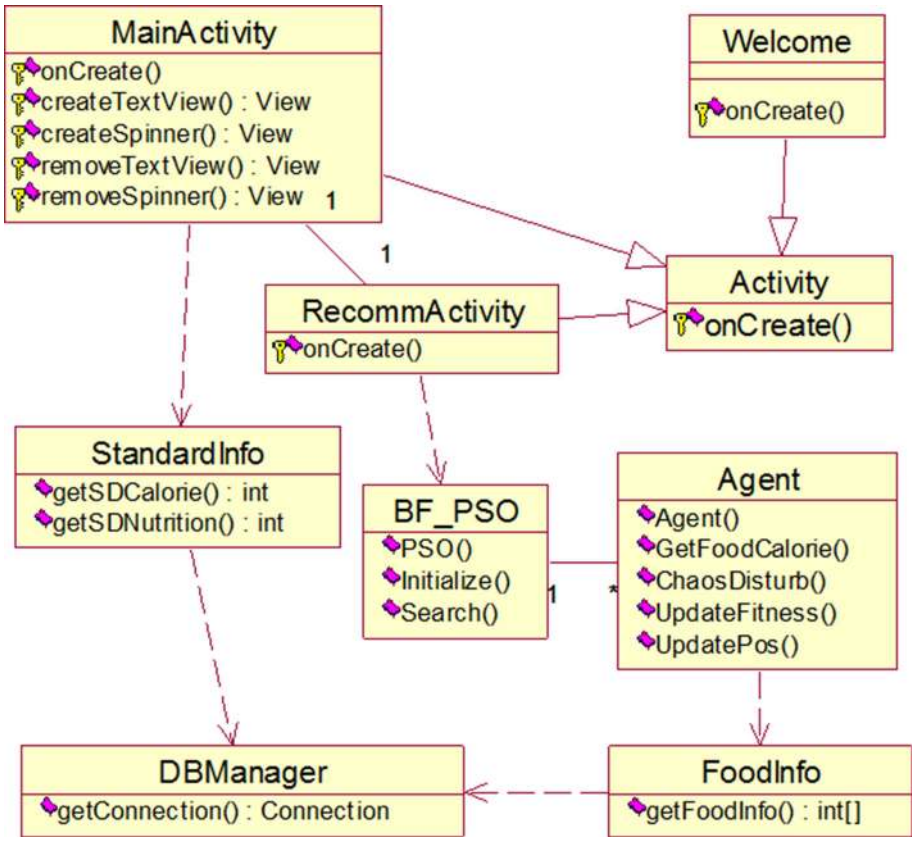
where w is the weight of a person and h is the height of a person. Table 1 shows the BMI for Asian adults

As shown in Fig. 4, *Friend* is composed of the following classes:

- `MainActivity` is the main interface of *Friend* for users to input their personal physiological data,
- `StandardInfo` and `DBManager` select the appropriate personalized standard values of calories and nutrients from the database,
- `RecommActivity` is an activity, which receives the personalized data from the interface of MainActivity, and the recommended diet scheme will show in this activity,
- `BF_PSO` is a class, which is mainly used for the initialization of the diet particles,
- `Agent` is a class, which is mainly used for updating of the diet particles,
- Finally, `FoodInfo` and `DBManager` are responsible for selecting the recommended diet from the database.

CS-PSO is achieved via `RecommActivity`, `BF_PSO`, `Agent`, and `FoodInfo`, which interact with each other to provide the scheme of diet recommendation.

Fig. 4 Classes of *Friend*



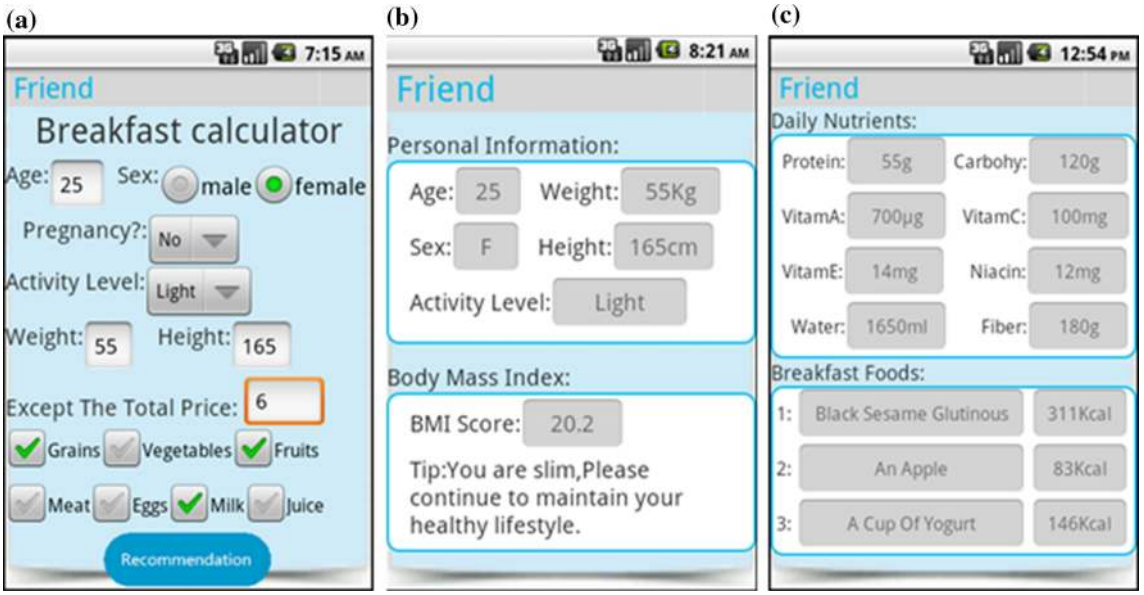


Fig. 5 User interfaces of *Friend*

Figure 5 shows the user interfaces of *Friend*. Consider breakfast for example, as shown in Fig. 5a. *Friend* requires users to input their personalized information, including age, gender, activity level, weight, height, and budget on food and food preference. After providing the above information, users need to click the recommendation button, and it will generate the scheme of diet recommendation as shown in Fig. 5b, c.

Table 2 Schemes of diet recommendation with HLR-PSO

Scheme	Food items	Amount of food	Calories (Kcal)
1	Watermelon	250.0 g	35
	Yoghurt (brand A)	100.0 g	63
	Pure milk (brand A)	460 ml	258
	Yoghurt (brand B)	200.0 g	184
2	Sweet potato	300.0 g	267
	Cucumber	130.0 g	18
	Orange	182.0 g	61
	Grapefruit	139.0 g	44
	Strawberry	500.0 g	150
3	Dumpling	100 g	250
	Watermelon	250.0 g	35
	Orange	200.0 g	70
	Grape	500.0 g	185

Table 3 Schemes of diet recommendation with CS-PSO

Scheme	Food items	Amount of food	Calories (Kcal)
1	Noodle	100 g	284
	Peach	200.0 g	83
	Milk (brand B)	250.0 ml	173
2	Dumpling	100 g	253
	Cherry	500.0 g	200
	Yoghurt (brand B)	100.0 ml	87
3	Chinese style baked roll	80 g	234
	Grape	500.0 g	185
	Milk (brand C)	200.0 ml	173

Table 4 Iteration times of HLR-PSO

HLR-PSO	Times	Times	Times	Times	Times	Average
1–5	34	17	12	13	26	15.1
6–10	16	12	23	9	10	
11–15	22	12	10	16	10	
16–20	11	10	18	9	12	

5 Experiments and performance analysis

HLR-PSO is a typical PSO for generating healthy lifestyle recommendations and has good performance. Therefore, we applied HLR-PSO and then CS-PSO to *Friend* to compare their performances in the following three aspects: the diversity of the recommended food items, the times of iteration for finding the global best value, and the ergodicity of algorithm.

5.1 Diversity

Tables 2 and 3 show the schemes of diet recommendation with HLR-PSO and CS-PSO, respectively.

Table 5 Iteration times of CS-PSO

HLR-PSO	Times	Times	Times	Times	Times	Average
1–5	3	4	6	4	4	4.1
6–10	4	5	6	2	4	
11–15	3	2	4	5	4	
16–20	4	4	4	5	5	

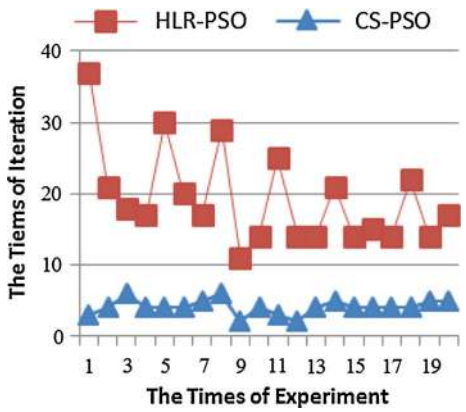


Fig. 6 Comparison of iteration times

Table 6 Index of each category with HLR-PSO

HLR-PSO	1	2	3	4	5	6	7	8	9	10
Staple	31	30	31	30	30	30	30	30	21	15
Fruits	20	30	20	30	30	30	30	30	28	28
Milk	30	25	30	25	25	25	25	25	30	29
	11	12	13	14	15	16	17	18	19	20
Milk	29	30	25	29	30	30	30	25	30	29
Staple	32	31	30	32	31	22	22	30	31	32
Fruits	28	20	30	28	20	28	28	30	20	28

Table 7 Index of each category with CS-PSO

CS-PSO	1	2	3	4	5	6	7	8	9	10
Staple	4	2	23	1	2	7	4	14	14	30
Fruits	26	29	15	29	29	4	26	19	21	30
Milk	6	9	25	29	9	27	10	24	26	25
	11	12	13	14	15	16	17	18	19	20
Staple	16	25	31	5	32	24	7	4	2	24
Fruits	7	28	20	1	28	12	4	26	7	12
Milk	27	19	30	16	29	30	27	6	30	30

As shown in Tables 2 and 3, the schemes of diet recommendation with CS-PSO are more reasonable than the schemes of diet recommendation with HLR-PSO. Scheme 1, recommended with HLR-PSO, includes three types of dairy

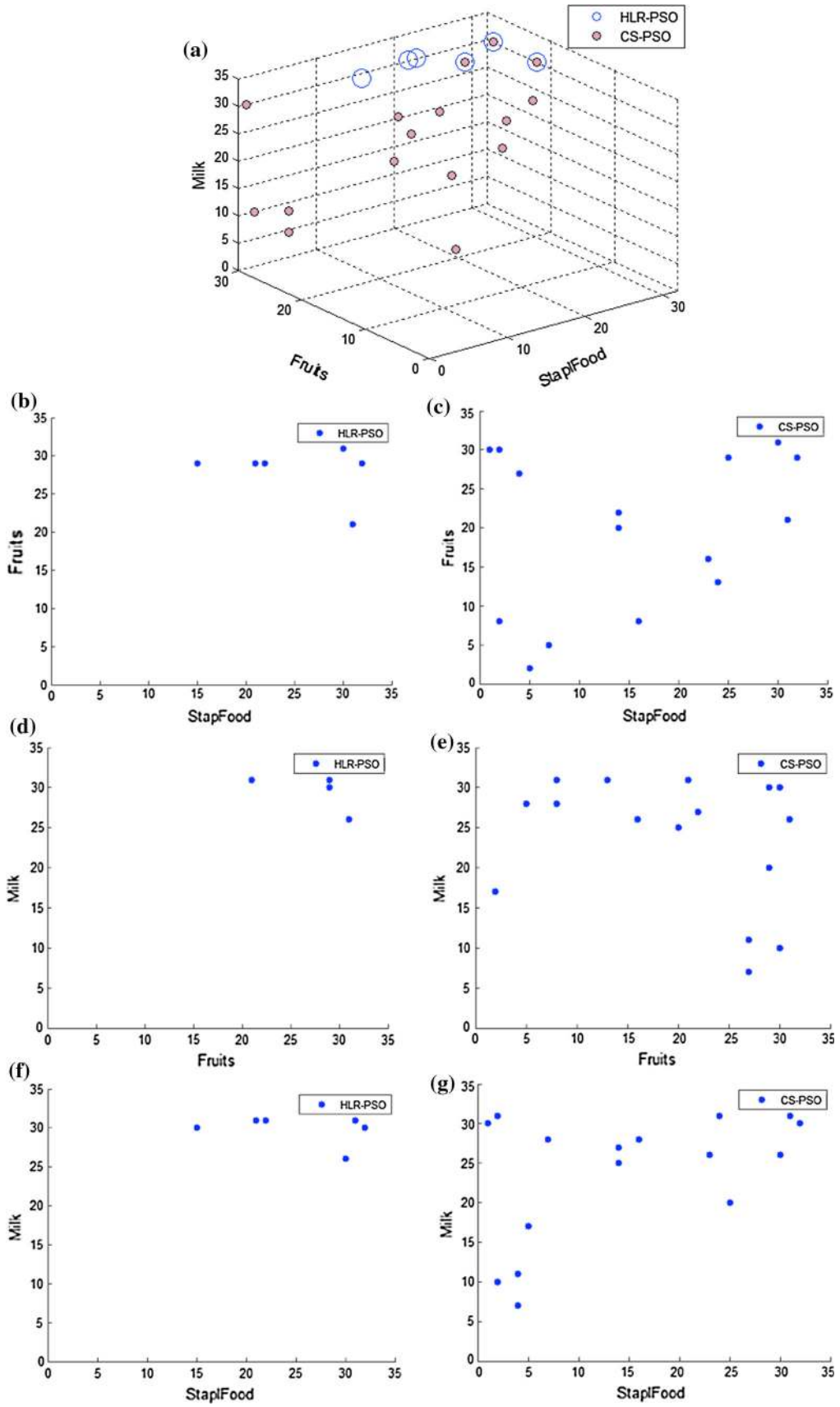


Fig. 7 Mapping graphs

products, and both scheme 2 and scheme 3 include three types of fruits, respectively, which are all not appropriate according to the standards of a healthy diet. However, scheme 2 includes three types of food, such as fruit and milk, and both scheme 1 and scheme 3 include three types of food, including cereal, fruit, and milk, respectively. As a consequence, CS-PSO can ensure the diversity of food, while HLR-PSO cannot. The reason is that CS-PSO adopts the prior knowledge of breakfast and food preferences of users.

5.2 Iteration times

Using HLR-PSO and CS-PSO run 20 times, respectively, we record the times of iteration when find the global best value. The results are shown in Tables 4 and 5.

The comparison between two algorithms about the times of iteration is shown in Fig. 6. In Fig. 6, the times of iteration of CS-PSO are far less than HLR-PSO.

5.3 Ergodicity

We chose three categories of food, including staple food, fruits, and milk. The index of staple food is between 0 and 32, fruits are between 0 and 30, and milk is between 0 and 30. HLR-PSO and CS-PSO run 20 times, respectively, and we recorded the index of each category. The experimental results are shown in Tables 6 and 7.

As shown in Fig. 7, the schemes traversed by HLR-PSO algorithm fall into six categories: schemes 15, 28, 29 and 21, 28, 30 appear once; schemes 22, 28, 30 appear twice; schemes 30, 30, 25 appear 8 times; schemes 31, 20, 30 appear 5 times; schemes 32, 28, 29 appear 3 times, and these schemes mainly concentrate on the latter three types. With the analysis of Fig. 7 and Table 7, the schemes traversed by CS-PSO algorithm fall into 16 types: the schemes of 2, 29, 9 and 4, 26, 6 and 7, 4, 27 and 24, 12, 30 appear twice; other schemes all appear only once. The high frequency schemes in Table 6 appear only once in Table 7, in the 10th, 13th, and 15th values, respectively. We can conclude that the traversal results by HLR-PSO are relatively more concentrated, and the traversal results are relatively fewer than CS-PSO. To sum up, CS-PSO has the better ergodicity than HLR-PSO.

6 Conclusion

Combinatorial optimization problem is a type of NP-hard problem. The traditional combinatorial optimization algorithms cannot guarantee the diversity of the final scheme, solve the multi-objective optimization problems effectively, or satisfy the search efficiency, etc. In order to successfully address such problems and further improve the performance of PSO, we have introduced a novel approach in solving combinatorial optimization problems, namely CS-PSO. Furthermore, we have discussed its use as part of the diet recommendation system *Friend*. The experimental results show that CS-PSO has the better diversity, ergodicity, and efficiency than HLR-PSO. In addition, CS-PSO can not only be used in diet recommendation, but also be used in product design, exercise programming, travel planning, etc. However, CS-PSO only considers the combination of the overall scheme, without considering the logical structure of combination.

In future research, we are aiming to integrate the automated construction mechanism of logical structure with combinatorial optimization problems. In particular, this approach will further enhance the performance and accuracy of the method discussed in this article, which is already supported by initial evaluations.

Acknowledgements This work was supported jointly sponsored by the National Natural Science Foundation of China under Grant 61472192 and 61472193.

Compliance with ethical standards

Conflict of interest The authors declare that none of them has any conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Bessis N, Sotiriadis S, Pop F, Cristea V (2012) Optimizing the energy efficiency of message exchanging for service distribution in interoperable infrastructures. In: 4th international conference on intelligent networking and collaborative systems (INCoS), pp 105–112

Chang YC, Hsieh CH, Xu YX et al (2014) Introducing the concept of velocity into bare bones particle swarm optimization. Presented at the 2014 international conference of information science, electronics and electrical engineering, Sapporo, Japan, 26–28 April

Chen SY, Ren L, Xin FQ (2012) Reactive power optimization based on particle swarm optimization and simulated annealing cooperative algorithm. In: Proceedings of the 31st conference of Chinese control conference, Hefei, China, 25–27 July

Chi YH, Sun FC, Wang WJ et al (2011) An improved particle swarm optimization algorithm with search space zoomed factor and attractor. Chin J Comput 34(1):115–130

Dong N, Li HJ, Liu XD (2013) Chaotic particle swarm optimization algorithm parametric identification of Bouc–Wen hysteresis model for piezoelectric ceramic actuator. In: Proceedings of the 25th Chinese control and decision conference, Guiyang, China, 25–27 May

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of ISOMMHS, Nagoya, Japan, pp 39–43

Elbedwehy MN, Zawbaa HM, Ghali H et al (2012) Detection of heart disease using binary particle swarm optimization. In: Proceedings of the 2012 conference on computer science and information systems, Wroclaw, Poland, 9–12 Sept

Gao Y, Xiei SL (2004) Chaos particle swarm optimization algorithm. J Comput Sci 31(8):13–15

Gao HB, Zhou C, Gao L (2005) General particle swarm optimization model. Chin J Comput 28(12):1980–1987

Guo WZ, Chen GL, Peng SJ (2011) Hybrid particle swarm optimization algorithm for VLSI circuit partitioning. J Softw 22(5):833–842

Guo T, Lan JL, Li YF (2014) Adaptive fractional-order darwinian particle swarm optimization algorithm. J Commun 35(4):130–140

Ibrahim I, Yusof ZM, Nawawi SW et al (2012) A Novel multi-state particle swarm optimization for discrete combinatorial optimization problems. In: Proceedings of the 4th international conference on computational intelligence, modelling and simulation, Kuantan, Malaysia, 25–27 Sept

Iordache GV, Boboila MS, Pop F, Stratan C, Cristea V (2006) A decentralized strategy for genetic scheduling in heterogeneous environments. On the move to meaningful Internet systems 2006: CoopIS, DOA, GADA, and ODBASE: OTM confederated international conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, October 29–November 3, 2006. Proceedings, Part II, Springer Berlin, Heidelberg, pp 1234–1251

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN, Perth, Australia, pp 1942–1948

Kumari N, Jha AN (2014) Frequency control of multi-area power system network using PSO based LQR. In: Proceedings of the 6th international conference power India Delhi, India, 5–7 Dec

Lee KB, Kim JH (2013) Multi objective particle swarm optimization with preference-based sort and its application to path following footstep optimization for humanoid robots. IEEE Trans Evolut Comput 17(6):755–766

Lei KY (2014) A highly efficient particle swarm optimizer for super high-dimensional complex functions optimization. In: Proceedings of the 5th international conference software engineering and service science, Beijing, China, 27–29 June

Liao YF, Yau DH, Chen CL (2012). Evolutionary algorithm to traveling salesman problems. Comput Math Appl. 64(5):788–797. Available: http://www.sciencedirect.com/science/article/pii/S089812211101073X

Lorenz EN (2005) Designing chaotic models. J Atmos Sci 62(5):1574–1587

Nancharaiah B, Mohan BC (2013) MANET link performance using ant colony optimization and particle swarm optimization algorithms. In: Proceedings of the 2013 international conference of communications and signal processing, Melmaruvathur, Tamil, 3–5 April

Pop CB, Chifu VR, Salomie I et al (2013) Particle swarm optimization-based method for generating healthy lifestyle recommendations. In: Proceedings of the international conference of intelligent computer communication and processing, Cluj-Napoca, Romania, 5–7 Sept

Sfrent A, Florin Pop F (2015) Asymptotic scheduling for many task computing in big data platforms. Inf Sci 319:71–91

Sharma J, Singhal RS (2015) Comparative research on genetic algorithm, particle swarm optimization and hybrid GA-PSO. In: Proceedings of the 2015 conference of computing for sustainable global development, New Delhi, India, 11–13 March

Shi YH, Eberhart RA (1998) Modified particle swarm optimizer. In: Proceedings of the IICOEC, Anchorage, AK, pp 69–73

Sorkunlu N, Sahin U, Sahin F (2013) Block matching with particle swarm optimization for motion estimation. In: Proceedings of the 2013 international conference on systems, man, and cybernetics, Manchester, England, 13–16 Oct

Wang TJ, Wu YC (2011) Study on optimization of logistics distribution route based on chaotic PSO. Comput Eng Appl 47(29):218–221

Wen T, Sheng GJ, Guo Q et al (2013) Web service composition based on modified particle swarm optimization. Chin J Comput 36(5):1031–1046

Xu XB, Zhang KG, Li D et al (2012) New chaos-particle swarm optimization algorithm. J Commun 33(1):24–30

Yang Q, Chen Q, Li ZZ (2015) A chaos particle swarm optimization algorithm of vehicle routing problem with time windows. Comput Technol Dev 25(8):119–122

Yao JJ, Li J, Wang LM et al (2012) Wireless sensor network localization based on improved particle swarm optimization. In: Proceedings of the 2012 international conference of computing, measurement, control and sensor network, Taiyuan, China, 7–9 July

Zhao XC, Liu GL, Liu HQ (2014) Particle swarm optimization algorithm based on non-uniform mutation and multiple states perturbation. Chin J Comput 37(9):2058–2070

Zhu XH, Li YG, Li N et al (2014) Improved PSO algorithm based on swarm prematurely degree and nonlinear periodic oscillating strategy. J Commun 35(2):182–189