5

Cat Swarm Optimization - Modifications and Application

Dorin Moldovan
Department of Computer Science
Technical University of Cluj-Napoca, Romania

Adam Slowik
Department of Electronics and Computer Science
Koszalin University of Technology, Koszalin, Poland

Viorica Chifu
Department of Computer Science
Technical University of Cluj-Napoca, Romania

Ioan Salomie
Department of Computer Science
Technical University of Cluj-Napoca, Romania

CONTENTS

5.1 Introduction

Cat Swarm Optimization (CSO) was introduced in [1] and it is a part of a larger family of algorithms that are nature inspired. This family of algorithms includes algorithms such as Particle Swarm Optimization (PSO) [2], Cuckoo Search (CS) [3], Bat Algorithm (BA) [4], Ant Colony Optimization (ACO) [5], Elephant Herding Optimization (EHO) [6] and Lion Optimization Algorithm (LOA) [7]. The CSO algorithm is inspired by the behavior of the cats in nature and it is an algorithm that can be used for solving complex engineering problems in which the search space has many dimensions. In the case of the CSO algorithm each solution of the optimization problem is represented by a cat that has a velocity and a position and the cats are characterized by two types of behavior namely, seeking mode and tracing mode. The objective of the algorithm is to determine the best cat according to a fitness function that depends on the optimization problem that will be solved. The main idea of optimization in the case of the CSO algorithm is the movement of the cats towards a local best *Lbest* in the case of the local version of the algorithm or towards a global best *Gbest* in the case of the global version of the algorithm. In literature there are many varieties of CSO algorithm such as: Crazy Cat Swarm Optimization (CCSO) [8], Harmonious Cat Swarm Optimization (HCSO) [9], Binary Cat Swarm Optimization (BCSO) [10] and Parallel Cat Swarm Optimization (PCSO) [11]. The chapter is organized as follows: in Section 5.2 is given a short presentation of the CSO algorithm, of the global version of CSO (GCSO) and of the local version of CSO (LCSO), in Section 5.3 are illustrated modifications of CSO such as velocity clamping, inertia weight, mutation operators, acceleration coefficient c_1 and adaptation for diets recommendation, in Section 5.4 is presented the application of the CSO algorithm for generation of diets and in Section 5.5 are presented the main conclusions.

5.2 Original CSO algorithm in brief

The original version of the CSO algorithm in the global version (GCSO) and in the local version (LCSO) can be represented using the pseudo-code from Algorithm 6. The code that is used only for the local version is underlined with a dotted line, while the code that is used only for the global version is underlined with a solid line.

Algorithm 6 Pseudo-code of the original CSO.

```
Input $M$ - the number of dimensions, $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$ - range of variability for
$i$-th cat and $j$-th dimension, $MR$, $SMP$, $SRD$, $CDC$, $SPC$, $N$ - the total number
of cats, $max$ - maximum number of iterations, $c_1$ - a constant for updating the
velocity of the cats in tracing mode, number of returned results $Re \in [1, N]$, the
topology and the number of nearest neighbors $Ne$
Output $Gbest$ - the best position achieved by a cat; select $Re$ the best cats
among the $Lbest$ cats and return these cats as results
1: determine the $M$-th dimensional objective function $OF(.)$
2: randomly create the population of cats $X_i (i = 1, 2, ..., N)$
3: create the $M$-dimensional $Gbest$ vector
4: while stopping criterion not satisfied or $I < I_{max}$ do
5:     set the cats in tracing mode or seeking mode according to $MR$
6:     for each $i$-th cat $X_i$ do
7:         create a $Xbest_i$ cat equal to $i$-th cat $X_i$
8:         create a $Lbest_i$ cat for each cat $X_i$
9:         create a velocity $V_i$ for each cat $X_i$
10:        evaluate the cat $X_i$ using the $OF(.)$ function
11:    end for
12:    assign the best cat $X_i$ to the $Gbest$; for each cat $Xbest_i$ assign the best cat
   among $Ne$ nearest neighbors of cat $X_i$
13:    for $i = 1 : N$ do
14:        if $X_i$ is in seeking mode then
15:            create $SMP$ copies
16:            update the position of each copy using the formula
17:            $X_{cn} = X_c \times (1 \pm SRD \times R)$
18:            pick randomly a position to move to from the set of $SMP$ copies
19:        else
20:            update the velocity of the cat using the formula
21:            $v_{i,d} = v_{i,d} + R \times c_1 \times (Gbest_d - X_{i,d})$
22:            $v_{i,d} = v_{i,d} + R \times c_1 \times (Lbest_{i,d} - X_{i,d})$
23:            update the position of the cat using the formula
24:            $X_{i,d,new} = X_{i,d,old} + v_{i,d}$
25:        end if
26:        select the best cat among $Ne$ nearest neighbors of cat $X_i$ and assign it
   to the cat $T$
27:        if $OF(T)$ better than $OF(Lbest_i)$ then
28:            assign the cat $T$ to the cat $Lbest_i$
29:        end if
30:    end for
31:    select the best cat and assign it to cat $T$
32:    if $OF(T)$ better than $OF(Gbest)$ then
33:        assign the cat $T$ to the cat $Gbest$
34:    end if
35: end while
36: return the $Gbest$ as a result; the best $Re$ cats among the $Lbest$ cats
```

5.2.1 Description of the original CSO algorithm

The pseudo code of the original CSO algorithm is presented in Algorithm 6 and in this subsection the algorithm is described in more detail. The inputs of the algorithm are M - the number of dimensions, $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$ - the range of variability for the i-th cat for the j-th dimension, MR - the mixture ratio, a numerical value that indicates how many cats are in tracing mode and how many cats are in seeking mode, SMP - the seeking memory pool, SRD - the seeking range of the selected dimension, CDC - the count of dimensions to change, SPC - the self-position consideration, N - the total number of cats, max - the maximum number of iterations and c_1 - a constant that is used for updating the velocity of the cats. In addition to these inputs, the local version of the algorithm uses as inputs the number of returned results $Re \in [1, N]$, the topology and Ne which is the number of nearest neighbors. The output of the algorithm in the case of the global version is represented by $Gbest$, while the output of the algorithm in the case of the local version is represented by the best Re cats among the $Lbest$ cats.

In step 1 of the algorithm is created the M-th dimensional objective function $OF(.)$ and in step 2 of the algorithm the initial population of N cats is created randomly. For the GCSO algorithm in step 3 is created the M-dimensional $Gbest$ vector. While the stopping criterion that is set initially is not satisfied or the number of iterations is less than a threshold max, a number of steps is repeated. In step 5 of the algorithm the cats are set either in tracing mode or in seeking mode according to a numerical value MR which is the mixture ratio. Then for each cat X_i the following operations are performed: an equal copy $Xbest_i$ is created, in the LCSO version of the algorithm a $Lbest_i$ cat is created, a velocity vector V_i is initialized and the cat X_i is evaluated using the objective function $OF(.)$. In step 12 of the algorithm, in the GCSO version the best cat X_i according to the value returned by $OF(.)$ is assigned to $Gbest$ while in the LCSO version of the algorithm for each cat $Xbest_i$ the best cat among the Ne nearest neighbors of cat X_i is assigned.

The steps from step 13 to step 30 are executed for each cat X_i. If the cat X_i is in seeking mode then in step 15 of the algorithm SMP copies of that cat are created, the position of each copy is updated using a formula that considers the SRD - the seeking range of the selected dimension and R a random numerical value from the interval $[0, 1]$ and in step 18 a position to move to is picked randomly from the set of SMP copies. Otherwise, if the cat is in tracing mode, in step 20 the velocity of the cat is updated using a different formula depending on whether the algorithm is GCSO or LCSO and in step 23 the position of the cat is updated using a formula that considers the computed velocity. If the algorithm is LCSO then steps 26-29 are performed: the best cat from the Ne nearest neighbors is selected and assigned to cat T and if the value of $OF(T)$ is better than the value of $OF(Lbest_i)$, the new value of $Lbest_i$ becomes T. The steps 31-34 are performed only in the GCSO version of the algorithm: the best cat is assigned to T and if $OF(T)$ is better than $OF(Gbest)$ then cat T is assigned to $Gbest$. Finally in step 36 of the algorithm, in the GCSO version the $Gbest$ is returned as a result and in the $LCSO$ version the best Re cats among the $Lbest$ cats are returned.

5.3 Modifications of the CSO algorithm

5.3.1 Velocity clamping

In the original version of the CSO algorithm the velocity explodes quickly to large values. One solution for this problem is represented by the introduction of limited step sizes as shown in the following equation:

$$V_{i,j} = \begin{cases} V_{i,j} & \text{if } |V_{i,j}| < V_j^{max} \\ V_j^{max} & \text{if } |V_{i,j}| \geq V_j^{max} \end{cases} \tag{5.1}$$

where V_j^{max} is the maximum value of velocity for the j-th decision variable. This value can be computed using the formula:

$$V_j^{max} = \delta \times \left(P_j^{max} - P_j^{min}\right) \tag{5.2}$$

where $\delta \in [0, 1]$ is a numerical value that can be chosen by trial-and-error and it usually depends on the optimization problem [12], P_j^{max} is the maximum value from the search domain for the j-th decision variable and P_j^{min} is the minimum value from the search domain for the j-th decision variable.

5.3.2 Inertia weight

The application of inertia weight in the case of CSO is inspired by the application of inertia weight in the case of Particle Swarm Optimization (PSO) [13]. In literature there are various types of inertia weights such as the constant inertia weight [14], the random inertia weight [15] and the adaptive inertia weight [16]. The equation for velocity update in the case of GCSO when inertia weight is used is:

$$V_{i,j} = \omega \times V_{i,j} + R \times c_1 \times (Gbest_j - X_{i,j}) \tag{5.3}$$

and the equation for velocity update in the case of LCSO when inertia weight is used is:

$$V_{i,j} = \omega \times V_{i,j} + R \times c_1 \times (Lbest_{i,j} - X_{i,j}) \tag{5.4}$$

where $V_{i,j}$ is the value of velocity of the i-th cat for the j-th dimension, ω is the inertia weight, R is a random numerical value from the interval $[0, 1]$, $Gbest_j$ is the position of the global best cat for the j-th dimension, $Lbest_{i,j}$ is the position of the local best for the i-th cat and the j-th dimension and $X_{i,j}$ is the position of the i-th cat for the j-th dimension. The value of the inertia factor ω is usually in the interval $[0.4, 0.9]$ and there are two major cases: when $\omega \geq 1$ the swarm of cats diverges and when $0 < \omega < 1$ the cats decelerate. The value of ω depends on the optimization problem that is solved.

5.3.3 Mutation operators

The mutation operators [17] are used in order to improve the performance of the CSO algorithm and to escape from the local minima. In some cases the global best cat is mutated while in other cases the local best cat is mutated. The mutation operator that is presented in this chapter is adapted after the one presented in [18]. First, a weight vector is computed using the following formula:

$$W_i = \frac{\sum_{j=1}^{N} V_{j,i}}{N} \tag{5.5}$$

where N is the size of the population, $V_{j,i}$ is the i-th velocity value of the j-th cat from the swarm and W_i is the value at position i of the weights vector W. The value of W_i is from an interval $[-W_{max}, W_{max}]$ and the value of W_{max} is usually equal with 1. Second, the value of the global best cat is mutated using the following formula:

$$Gbest_i^{'} = Gbest_i + W_i \times \mathcal{N}\left(X_{min}, X_{max}\right) \tag{5.6}$$

where $Gbest_i$ is the i-th value of the $Gbest$ vector, $Gbest_i^{'}$ is the i-th value of the mutated $Gbest_i$ vector, W_i is the value at the i position of the weights vector and $\mathcal{N}\left(X_{min}, X_{max}\right)$ is a random number from a Cauchy distribution that has the scale parameter equal with 1 where X_{min} and X_{max} are the minimum and the maximum values that can be taken by the positions of the cats for each dimension.

Using a similar approach, the Cauchy operator can be used in order to mutate the value of the local best cat using the following formula:

$$Lbest_i^{'} = Lbest_i + W_i \times \mathcal{N}\left(X_{min}, X_{max}\right) \tag{5.7}$$

where $Lbest_i$ is the i-th value of the $Lbest$ vector and $Lbest_i^{'}$ is the i-th value of the mutated $Lbest$ vector.

5.3.4 Acceleration coefficient c_1

The coefficient c_1 is named acceleration coefficient because it is responsible for the social part of the algorithm. We can introduce the adaptive change of the acceleration coefficient using the following formula:

$$c_1 = \left(c_1^{min} - c_1^{max}\right) \times \frac{t}{t^{max}} + c_1^{max} \tag{5.8}$$

where c_1^{min} is the minimum value that can be taken by c_1, c_1^{max} is the maximum value that can be taken by c_1, t is the current number of generations and belongs to the interval $[0, t^{max}]$ and t^{max} is the maximum number of generations. When $t = 0$ the value of c_1 is equal with c_1^{max} and when $t = t^{max}$ the value of c_1 is equal with c_1^{min}.

5.3.5 Adaptation of CSO for diets recommendation

In this subsection is described how the classical CSO algorithm can be adapted for discrete optimization problems, in particular for recommendation of diets. Discrete optimization problems are a special subset of the optimization problems in which the search space is discrete, or in other words for each dimension there is a finite set of states. In the approach that is used in this chapter the search space is represented by dishes that have well known nutritional values.

As in the classical version of the CSO algorithm the cats are initialized to random values. Those values are in intervals described by the minimum and the maximum values of the nutritional properties of the dishes from the search space. Each cat corresponds to a diet that is composed from a number of dishes and in order to determine the dishes from the search space that are the closest to the numerical values that describe the positions of the cats, the squared Euclidean distance is applied.

If the position vector of a cat is $\{(x_{1,1}, ..., x_{1,M}), ..., (x_{K,1}, ..., x_{K,M})\}$ where K is the number of dishes and M the number of nutritional values then the dish from the search space that corresponds to the k-th dish of the cat, where $k = 1, ..., K$, is the one for which the value of the function presented below has the minimum value:

$$F(cat, k, dish) = \sum_{i=1}^{M} (x_{k,i} - dish(nutrient_i))^2 \tag{5.9}$$

where $dish(nutrient_i)$ is the value of the i-th nutrient of the dish. The cat from the function $F(cat, k, dish)$ is from the swarm of cats and the dish is from the search space of dishes.

5.4 Application of CSO algorithm for recommendation of diets

5.4.1 Problem description

The generation of diets is a complex discrete optimization problem due to the fact that the search space is very large and for each person the recommended nutritional values for a day are different. These recommended nutritional values are computed by taking into consideration several properties of the person such as height, weight, age, food preferences and so on. Just to create an image of how many different types of foods exist, in [19] is listed information about approximately 250,000 foods. Additionally the world population is projected to reach 9.8 billion people by 2050 and due to the last technological advancements the people are expected to live longer than ever before. The variety of foods is also expected to be greater in the next years due to technological advancements and one of the major consequences is represented by the fact that the people are able to select diets that fit the best for their particular needs such as health improvement, weight loss or muscle gain.

5.4.2 How can the CSO algorithm be used for this problem?

In this subsection we present how the CSO algorithm can be used for recommendation of diets. The recommendation of diets using CSO was treated before by us in [20] but compared to the approach presented in that article, in this chapter we perform the following major changes: (1) we use another dataset of dishes which contains dishes for different types of diets such as diabetic and Mediterranean, (2) we consider only the nutritional values of the dishes when computing the fitness values of the diets and (3) we also consider how the tuning of the CSO parameters might influence the homogeneity of the diets. In this chapter we consider that a diet consists of four principal meals in a day namely, breakfast, lunch, dinner and supper. In addition each meal has three dishes: a main dish, a secondary dish and a dessert. In Figure 5.1 is presented the template of a diet. As can be seen in the figure a cat corresponds to a diet for one day and consists of one breakfast, one lunch, one dinner and one supper.

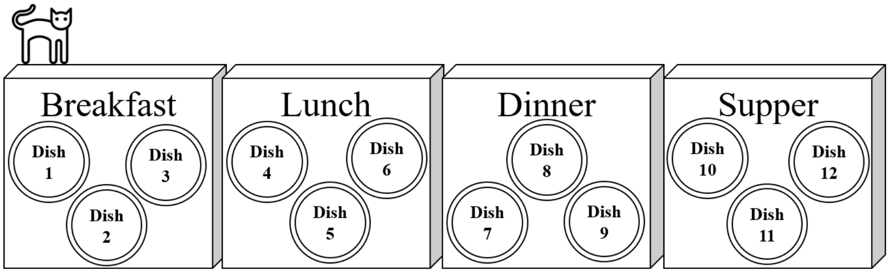

FIGURE 5.1
Illustrative template of a diet for the recommendation of diets optimization problem.

A brute force approach that considers all combinations of dishes is clearly too expensive in terms of computational resources such as time and memory allocation. The application of CSO for recommendation of diets is a great fit because it generates a near-optimal solution very fast, it produces different results at each run of the algorithm, thus ensuring the diversity of the diets, and it also has two major types of searching in the space of solutions namely, the seeking mode and the tracing mode, which overlap very well with the situations when the algorithm is applied for searching homogeneous diets and when the algorithm is applied for searching heterogeneous diets. By tuning the Mixture Ratio (MR) parameter, a number that indicates how many cats are in tracing mode and how many cats are in seeking mode, CDC which is the number of dimensions that will be mutated in the seeking mode and SPC, a flag used in seeking mode which specifies if the current position of the cat will be considered or not, it is possible to vary the degree of homogeneity of the generated recommendations of diets.

The mapping between the concepts used by CSO and the concepts used in the optimization problem is presented in Table 5.1.

TABLE 5.1
Mapping Between CSO Concepts and Optimization Problem Concepts

CSO Concept	Optimization Problem Concept
cat	a diet that consists of breakfast, lunch, dinner and supper
position	a vector that consists of $12 \times 4 = 48$ values
velocity	a vector that consists of $12 \times 4 = 48$ values
topology	ring topology
objective function	formula 5.10

The concepts are described in more detail next. A cat is mapped to a diet that has four main components namely, a breakfast, a lunch, a dinner and a supper, the position of the cat is represented by a vector that has 48 values (12 dishes $\times$ 4 nutritional values per dish), the velocity of the cat is represented similarly by a vector that has 48 values and the topology that is used in the local version of the algorithm is the ring topology which considers the left and the right neighbor cats. We consider that each dish has values for the following properties: (1) proteins (g), (2) lipids (g), (3) carbohydrates (g) and (4) energy (cal). The equation of the objective function is:

$$OF(cat) = OF(cat_{breakfast}) + OF(cat_{lunch}) + OF(cat_{dinner}) + OF(cat_{supper}) \quad (5.10)$$

where:

$$OF(cat_{breakfast}) = \sum_{i=1}^{3} OF\left(dish_{breakfast,i}\right) \quad (5.11)$$

$$OF(cat_{lunch}) = \sum_{i=1}^{3} OF\left(dish_{lunch,i}\right) \quad (5.12)$$

$$OF(cat_{dinner}) = \sum_{i=1}^{3} OF\left(dish_{dinner,i}\right) \quad (5.13)$$

$$OF(cat_{supper}) = \sum_{i=1}^{3} OF\left(dish_{supper,i}\right) \quad (5.14)$$

and the objective function for a dish is:

$$OF(dish) = \sum_{i=1}^{4} \left(dish(nutrient_i) - dish(optimal_{value}(nutrient_i))\right)^2 \quad (5.15)$$

The proposed method presented in this chapter is based on the local version of the CSO algorithm and it consists of the following steps.

Step 1 The initial population of cats X which consists of N cats X_i with $i = 1, ..., N$ is created randomly. Each cat is a D dimensional vector where $D = 12 \times 4 = 48$. For each i-th cat X_i the vectors $Xbest_i$, $Lbest_i$ and V_i are created and then the vector $Gbest$ is created for the whole swarm of cats. Vector $Xbest_i$ indicates the best position of the cat X_i in the search space among the positions that were obtained so far. Vector $Lbest_i$ determines the best position reached by another cat among the NN nearest neighbors of cat X_i found until now. Vector V_i represents the velocity of cat X_i. $Gbest$ determines the global best solution that was found by the CSO algorithm. In the beginning the vector X_i is assigned to $Xbest_i$ vector for the i-th cat, and the vectors $Lbest_i$, V_i and $Gbest$ are initialized to zero.

Step 2 Each cat X_i is evaluated using the objective function $OF(.)$ described by the formula 1.10 and the objective of the CSO algorithm is to minimize the value of the objective function $OF(.)$. When the value of the computed $OF(.)$ function for a cat X_i is lower than the value of $Xbest_i$, the values from cat X_i are assigned to vector $Xbest_i$.

Step 3 For each cat X_i the best local cat $Lbest_i$ is selected from the NN nearest neighbors of that cat X_i. The similarity between pairs of cats is determined using the values returned by $OF(.)$ and two cats are more similar when their $OF(.)$ values are closer.

Step 4 In this step the positions of the cats are updated using the equations that correspond either to the seeking mode or to the tracing mode. For the seeking mode SMP copies are created, the position of each copy is updated using the formula:

$$X_{cn} = X_c \times (1 \pm SRD \times R) \quad (5.16)$$

and the position to move to is selected randomly from the set of SMP copies. For the tracing mode the velocities of the cats are updated using the formula:

$$v_{i,d} = v_{i,d} + R \times c_1 \times (Lbest_{i,d} - X_{i,d}) \quad (5.17)$$

and the positions of the cats are updated using the formula:

$$X_{i,d,new} = X_{i,d,old} + v_{i,d} \quad (5.18)$$

In the formulas presented above the parameters have the following significance: SMP is the seeking memory pool, X_{cn} is the new position of the copy, X_c is the position of the copy, SRD is the seeking range of the selected dimensions, R is a random number from the interval $[0, 1]$, $v_{i,d}$ is the value of the velocity of the i-th cat for the d-th dimension, c_1 is a constant, $Lbest_{i,d}$ is the value of the local best position of the i-th cat for the d-th dimension, $X_{i,d}$ the value of the position of the i-th cat for the d-th dimension, $X_{i,d,new}$ is the new position of the cat i for the d-th dimension, $X_{i,d,old}$ is the old position of the cat i for the d-th dimension and $v_{i,d}$ is the value of velocity of the i-th cat for the d-th dimension.

Step 5 The worst cat from the swarm is replaced by a cat that is created randomly. The worst cat is the one for which the value that is returned by $OF(.)$ has the highest value. The scope of this operation is to increase the convergence of the algorithm.

Step 6 If a better cat is found considering the value returned by $OF(.)$ then the value of the *Gbest* cat is updated with that cat. If the number of the current generation is less than the number of generations given as input then the algorithm jumps to the first step, otherwise the loop stops and the algorithm continues with the next step.

Step 7 The algorithm returns the diet that corresponds to the *Gbest* cat. The diet that corresponds to the *Gbest* cat is calculated using the procedure presented in more detail in the subsection that describes how CSO is adapted for diets recommendation.

5.4.3 Description of experiments

In the approach presented in this chapter is considered a dataset that contains dishes for two types of diets namely, the diabetic diet and Mediterranean diet. Some examples of dishes are Zucchini Saute and Italian Stewed Tomatoes. The main challenge is represented by the initial configuration of the parameters of the CSO algorithm in order to be applied successfully on the discrete optimization problem described in this chapter. Since some parameters depend very much on the dataset that is used as support, we present justifications for each choice of parameters.

We consider that the number of dimensions of the search space is equal to $12 \times 4 = 48$ (12 dishes and 4 nutritional values for each dish). The number of cats N is equal to 10 and the value of MR is equal to 0.3, thus at each generation of the algorithm 7 cats are in seeking mode and 3 cats are in tracing mode. The number of generations is chosen as 100 after a series of experiments, but the optimal value of this number can be determined experimentally if we consider as stopping condition the state when the generation of the algorithm for which the absolute value of the difference between the global best at that generation and the global best at the previous p-th generation is less than a threshold ϵ.

The optimal values for p and ϵ should also be determined experimentally. The value of p is from the set $\{1, 2, 3, ...\}$ and the value of ϵ should be a very small numerical value. A possible approach to determine the value of ϵ is to consider the values from the set $\{0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, ...\}$.

The value of SMP is 5 and that means that 5 copies are generated for each cat in the seeking mode, the value of SRD is equal to 0.2, the value of CDC is $2 \times 11 = 22$ and the value of c_1 is 0.5.

The range of variability $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$ for the i-th cat for each j-th dimension corresponds to the minimum and the maximum values of the nutritional properties that correspond to that dimension. In addition we consider that the range of variability of the velocities is $\left[-2 \times P_{i,j}^{max}, 2 \times P_{i,j}^{max}\right]$.

5.4.4 Results obtained

In this subsection are presented experimental results for two types of diets namely, a diabetic diet and a Mediterranean diet. We chose these particular diets because they consider the health condition and this dimension will have a great importance in the future when selecting diets as the world population will increase significantly in the next years. For example in the case of the diabetic diet the prediction of diet recommendations is a hot research topic that was treated in literature using approaches such as type 2 fuzzy logic [21] and ontology and semantic matching [22]. In the case of the Mediterranean diet, some ingredients such as extra-virgin olive oil lower the incidence of Alzheimer's disease [23]. The optimal nutritional values for each type of diet are computed for a single day.

5.4.4.1 Diabetic diet experimental results

We consider the profile of a woman named Anna Smith that has diabetes. She is 60 years old, has a height of 179 centimeters and a weight of 75 kilograms. The recommended nutritional values for one day are computed considering the age, the height and the weight, and their values are: 136 proteins (g), 65 lipids (g), 203 carbohydrates (g) and 1936 energy (calories).

In Table 5.2 are listed the minimum and the maximum values of the nutrients of the dishes that compose the diabetic diet and in addition the optimal values of these nutrients for a day for Anna Smith.

The dishes that correspond to the global best cat are summarized in Table 5.3.

TABLE 5.2
Minimum and Maximum Values of the Nutrients of the Diabetic Diet Dishes and the Optimal Values of the Nutrients for a Day for Anna Smith.

Nutrient	Minimum Value / Dish	Maximum Value / Dish	Optimal Value / Day
proteins (g)	0.3	30	136
lipids (g)	0.1	21.3	65
carbohydrates (g)	6.6	36.7	203
energy (cal)	27	363	1936

TABLE 5.3
Diabetic Diet Dishes Recommendation for Anna Smith.

Dish	**Proteins (g)**	**Lipids (g)**	**Carbs (g)**	**Energy (cal)**
Breakfast				
Shiraz Salad	2.2	0.4	14.6	61.0
Grilled Chicken Citrus Salad	30.0	2.2	26.1	238.0
White Beans and Peppers	8.5	1.8	26.6	150.0
Lunch				
Banana Oat Energy Bars	3.6	4.0	20.0	124.0
Easy Roasted Peppers	1.8	0.5	10.8	55.0
Bean Soup With Kale	11.0	2.5	31.0	182.0
Dinner				
Curried Israeli Couscous	4.9	3.7	29.7	174.0
Curried Cottage Fries	3.8	4.1	28.5	162.0
Meyer Lemon Avocado Toast	3.6	1.2	11.8	72.0
Supper				
Black Bean Spread	6.8	0.7	16.3	96.0
Black Beans and Rice	6.3	0.9	27.1	140.0
Chickpea and Couscous Delight	5.4	0.9	29.3	147.0
Total	87.89	22.9	271.8	1601.0

5.4.4.2 Mediterranean diet experimental results

We consider the profile of a man named John Smith who is 80 years old, has a height of 176 centimeters and a weight of 85 kilograms. The corresponding nutritional values for one day are: 154 proteins (g), 71 lipids (g), 220 carbohydrates (g) and 2138 energy (cal).

In Table 5.4 are listed the minimum and the maximum values of the nutrients of the dishes that compose the Mediterranean diet and in addition the optimal values of these nutrients for a day for John Smith.

TABLE 5.4
Minimum and Maximum Values of the Nutrients of the Mediterranean Diet Dishes and the Optimal Values of the Nutrients for a Day for John Smith.

Nutrient	**Minimum Value / Dish**	**Maximum Value / Dish**	**Optimal Value / Day**
proteins (g)	1.6	41.7	154
lipids (g)	0.5	77.8	71
carbohydrates (g)	0.6	95	220
energy (cal)	49	853	2138

TABLE 5.5
Mediterranean Diet Dishes Recommendation for John Smith.

Dish	**Proteins (g)**	**Lipids (g)**	**Carbs (g)**	**Energy (cal)**
		Breakfast		
Chef John's Tzatziki Sauce	2.2	3.4	2.5	49.0
Real Hummus	1.6	2.5	6.8	54.0
Kefir Yogurt (Sana)	5.9	2.9	8.4	84.0
		Lunch		
Mediterranean Roast Chicken	39.8	26.6	81.6	724.0
Mediterranean Kale	4.6	3.2	14.5	91.0
Easy Greek Yogurt	7.9	2.3	10.7	95.0
		Dinner		
Whole Wheat Pita Bread	7.4	1.1	17.1	101.0
Slow Cooker Mediterranean Stew	3.4	0.5	30.5	122.0
Cauliflower Tabbouleh	4.9	7.1	15.1	128.0
		Supper		
Air Fryer Potato Wedges	2.3	5.3	19.0	129.0
Salmon Tartare	14.3	7.7	0.6	133.0
Asparagus Sformato	9.6	9.8	4.0	139.0
Total	103.9	72.4	210.79	1849.0

The dishes that correspond to the global best cat are summarized in Table 5.5.

5.5 Conclusions

In this paper the CSO algorithm was presented in detail, both in local version and in global version. Some modifications of the CSO algorithm such as velocity clamping, inertia weight, mutation operators, acceleration coefficient c_1 and adaptation for diets recommendation were demonstrated and discussed. Finally the CSO algorithm was applied to a real world problem namely, the recommendation of diets. Using the CSO algorithm we obtain near optimal results fast and we can also control the homogeneity of the recommended diets by varying several parameters such as MR which is the mixture ratio or CDC which is the count of dimensions to change. In this chapter at the beginning of the algorithm the cats are initialized randomly but as future research work we would like: (1) to consider cats that are initialized to specific diets that are recommended by nutritionists and to search similar diets in the neighborhood of those cats, (2) to parallelize the algorithm using big data frameworks such as Apache Spark [24] and (3) to include more parameters when recommending diets such as the food preferences, the quality of the foods and other characteristics of the foods like smell, taste or price.

References

1. S.-C. Chu, P.-W. Tsai, J.-S. Pan. "Cat Swarm Optimization" in *Yang Q., Webb G. (eds) PRICAI 2006: Trends in Artificial Intelligence. PRICAI 2006. Lecture Notes in Computer Science*, vol. 4099, 2006, pp. 854-858.
2. J. Kennedy, R. Eberhart. "Particle Swarm Optimization" in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942-1948.
3. X.-S. Yang, S. Deb. "Cuckoo Search via Levy flights" in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210-214.
4. X.-S. Yang. "A New Metaheuristic Bat-Inspired Algorithm" in *Gonzalez J.R., Pelta D.A., Cruz C., Terrazas G., Krasnogor N. (eds) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Studies in Computational Intelligence*, 2010, pp. 65-74.
5. M. Dorigo, M. Birattari, T. Stutzle. "Ant colony optimization" in *IEEE Computational Intelligence Magazine*, vol. 1, 2006, pp. 28-39.
6. G.-G. Wang, S. Deb, L. dos S. Coelho. "Elephant Herding Optimization" in *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, 2015, pp. 1-5.
7. M. Yazdani, F. Jolai. "Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm" in *Journal of Computational Design and Engineering*, vol. 3, 2016, pp. 24-36.
8. A. Sarangi, S. K. Sarangi, M. Mukherjee, S. P. Panigrahi. "System identification by Crazy-cat swarm optimization" in *2015 International Conference on Microwave, Optical and Communication Engineering (ICMOCE)*, 2015, pp. 439-442.
9. K. C. Lin, K. Y. Zhang, J. C. Hung. "Feature Selection of Support Vector Machine Based on Harmonious Cat Swarm Optimization" in *2014 7th International Conference on Ubi-Media Computing and Workshops*, 2014, pp. 205-208.

10. B. Crawford, R. Soto, N. Berrios, F. Johnson, F. Paredes. "Binary cat swarm optimization for the set covering problem" in *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*, 2015, pp. 1-4.

11. P.-W. Tsai, J.-S. Pan, S.-M. Chen, B.-Y. Liao, S.-P. Hao. "Parallel Cat Swarm Optimization" in *2008 International Conference on Machine Learning and Cybernetics*, 2008, pp. 3328-3333.

12. R. A. Jamous, A. A. Tharwat, E. El-Seidy, B. I. Bayoum. "Modifications of Particle Swarm Optimization Techniques and Its Application on Stock Market: A Survey" in *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 6, no. 3, 2015, pp. 99-108.

13. J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, A. Abraham. "Inertia Weight strategies in Particle Swarm Optimization" in *Proc. of the 2011 Third World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 633-640.

14. Y. Shi, R. Eberhart. "A modified particle swarm optimizer" in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 69-73.

15. R. C. Eberhart, Y. Shi. "Tracking and optimizing dynamic systems with particle swarms" in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 2001, pp. 94-100.

16. A. Nikabadi, M. Ebadzadeh. "Particle swarm optimization algorithms with adaptive Inertia Weight : A survey of the state of the art and a Novel method" in *IEEE Journal of Evolutionary Computation*, 2008.

17. M. Imran, R. Hashim, N. E. A. Khalid. "An overview of particle swarm optimization variants" in *Procedia Engineering*, vol. 53, 2013, pp. 491-496.

18. H. Wang, C. Li, Y. Liu, S. Zeng. "A hybrid particle swarm algorithm with Cauchy mutation" in *2007 IEEE Swarm Intelligence Symposium*, 2007, pp. 356-360.

19. United States Department of Agriculture, Agricultural Research Service, USDA Food Composition Databases, https://ndb.nal.usda.gov

20. D. Moldovan, P. Stefan, C. Vuscan, V. R. Chifu, I. Anghel, T. Cioara, I. Salomie. "Diet generator for elders using cat swarm optimization and wolf search" in *International Conference on Advancements of Medicine and Health Care through Technology*, 2016, pp. 238-243.

21. H. A. Mohammed, H. Hagras. "Towards developing Type 2 fuzzy logic diet recommendation system for diabetes" in *2018 10th Computer Science and Electronic Engineering (CEEC)*, 2018, pp. 56-59.

22. A. Arwan, M. Sidiq, B. Priyambadha, H. Kristianto, R. Sarno. "Ontology and semantic matching for diabetic food recommendations" in *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2013, pp.

23. Y. Batarseh, H. Qosa, K. Elsayed, J. N. Keller, A. Kaddoumi. "Extra-virgin olive oil and oleocanthal reduce amyloid ß load in Alzheimer's disease mouse model" in *2016 32nd Southern Biomedical Engineering Conference (SBEC)*, 2016, pp. 92-92. 170-175.

24. B. Akil, Y. Zhou, U. Rohm. "On the usability of Hadoop MapReduce, Apache Spark & Apache flink for data science" in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 303-310.