Journal of Optimization in Industrial Engineering, Vol.11, Issue 2, Summer and Autumn 2018, 105-115
DOI: 10.22094/joie.2017.500.12

EMCSO: An Elitist Multi-Objective Cat Swarm Optimization

Maysam Orouskhani[a], Mohammad Teshnehlab[b,*], Mohammad Ali Nekoui[b]

[a] *Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*

[b] *Industrial Control Center of Excellence, Electrical Engineering Department, K. N. Toosi University of Technology, Tehran, Iran*

Received 11 July 2015; Revised 15 october 2016; Accepted 20 November 2017

Abstract

This paper introduces a novel multi-objective evolutionary algorithm based on cat swarm optimization algorithm (EMCSO) and its application to solve a multi-objective knapsack problem. The multi-objective optimizers try to find the closest solutions to true Pareto front (POF) where it will be achieved by finding the less-crowded non-dominated solutions. The proposed method applies cat swarm optimization (CSO), a swarm-based algorithm with ability of exploration and exploitation, to produce offspring solutions and uses the non-dominated sorting method to find the solutions as close as to POF and crowding distance technique to obtain a uniform distribution among the non-dominated solutions. Also, the algorithm is allowed to keep the elites of population in reproduction process and use an opposition-based learning method for population initialization to enhance the convergence speed. The proposed algorithm is tested on standard test functions (zitzler' functions: ZDT) and its performance is compared with traditional algorithms and is analyzed based on performance measures of generational distance (GD), inverted GD, spread, and spacing. The simulation results indicate that the proposed method gets the quite satisfactory results in comparison with other optimization algorithms for functions of ZDT1 and ZDT2. Moreover, the proposed algorithm is applied to solve multi-objective knapsack problem.

***Keywords*:** Multi-objective cat swarm optimization; Non-dominated sorting; Crowding distance; Opposition-based learning.

1. Introduction

Optimization includes finding the best values of some objective function given a defined domain or a set of constraints, including a variety of different types of objective functions and different types of domains. If the quantity to be optimized is expressed using only one objective, the problem is then referred to as a single-objective problem, while many real-world problems require the simultaneous optimization of a number of objective functions which may be in conflict with one another; these problems are called multi-objective optimization problems (MOOP) whose main aim is to find a set of solutions.

There are many optimization algorithms for solving MOOPs. Some of these algorithms, such as classical algorithms, can only find one point as the best solution, while others, such as population-based evolutionary algorithms, find a set of solutions and keep and store the best one. Evolutionary algorithms are desirable to solve multi-objective optimization problems because they deal simultaneously with a set of possible solutions which allows finding an entire set of optimal solutions in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the classical algorithms. Most of optimization algorithms are derived from meta-heuristic concepts that are a higher-level procedure or heuristic designed to find, generate, or select a heuristic that may provide a sufficiently good solution for an optimization problem (Rahmanian, Ghaderi, & Mehrabad.M, 2012) (Mirzazadeh, Shirdel, & Masoumi, 2011) (Mehdizadeh & Kivi, 2014).

Swarm intelligence algorithms, as one of the main algorithms of metaheuristics, imitates the creature's swarm behavior on finding food or mating and are able to share obtained information between particles of the population. This paper applies Cat swarm optimization that imitates the behavior of cats in seeking and tracing mode. This paper proposes a multi-objective version of cat swarm optimization where an opposition-based learning method for population initialization is applied into algorithm to increase the convergence speed. Also, in order to investigate the performance of the algorithm, simulations are done on ZDT test functions and analyzed based on multi-objective optimization performance measures. Also, due to the important role of multi-objective optimization in practical optimization problems of industrial engineering such as allocation of customers to distribution centers (Bagherinejad & Dehghani, 2016), digital convergent product network (Hassanzadeha, Mahdavib, & Mahdavi-Amiric, 2014), workflow task scheduling in utility grids (Kahejvand, Hossein, & Zandieh, 2014), and flexible flow-shop scheduling problems (Naderi & Sadeghi, 2012), this paper applies the proposed algorithm to solve multi-objective knapsack problem.

The rest of the article is organized as follows: in Section 2, multi-objective optimization, Pure Cat swarm optimization, and knapsack problem are introduced; in Section 3, the proposed algorithm for solving multi-objective problems is expressed. In Section 4, simulation results for standard test

*Corresponding author Email address: teshnehlab@eetd.kntu.ac.ir

functions based on four performance metrics are indicated. Finally, a conclusion is drawn in the last section.

2. Background Information

2.1. Multi-objective optimization

Let $S \subseteq R^{n_x}$ and $F \subseteq S$ denote n_x-dimensional search space and the feasible space, respectively. With no constraints, the feasible space is the same as the search space. Let $x = (x_1, x_2, \ldots, x_{n_x}) \in S$, be referred to as a decision vector. A single-objective function $f_k(x)$ is defined as $f_k : R^{n_x} \rightarrow R$. Let $f(x) = (f_1(x), f_2(x), \ldots, f_{nk}(x)) \in O \subseteq R^{nk}$ be an *objective vector* containing n_k objective function evaluations; O is referred to as the *objective space*. The search space, S, is also referred to as the *decision space*. Figure (1) shows these two spaces. Equation (1) defines the multi-objective optimization for minimization problem:

$$\begin{aligned} &\text{Minimize} && f(x) \\ &\text{Subject to} && g_m(x) \leq 0 \qquad m = 1, 2, \ldots, n_g \\ &h_n(x) = 0 && n = 1, 2, \ldots, n_h \\ &x \in [x_{min}, x_{max}]^{n_x} \end{aligned} \tag{1}$$

Here, g_m and h_n are respectively the inequality and equality constraints, while $x \in [x_{min}, x_{max}]$ represents the boundary constraints. Solutions x^* to the multi-objective optimization problem are in the feasible space, i.e., all $x^* \in F$. The meaning of an "optimum" has to be redefined for MOO. In terms of MOO, the main problem is the presence of conflicting objectives, where improvement in one objective may cause deterioration in another objective and the task is to find solutions that balance these trade-offs. Such a balance is achieved when a solution cannot improve any objective without degrading one or more of the other objectives. These solutions are referred to as non-dominated solutions. The objective, when solving a MOOP, is to produce a set of good compromises, instead of a single solution. This set of solutions is referred to as the non-dominated set, or the Pareto-optimal set. The corresponding objective vectors in objective space are referred to as the Pareto front (Engelbrecht., (2002).).

2.1.1 Definitions

Domination: A decision vector, x_1, dominates a decision vector, x_2 (denoted by $x_1 \prec x_2$), if and only if x_1 is not worse than x_2 in all objectives:

$$i.e., \quad \forall\ k = 1, \ldots, n_k : f_k(x_1) \leq f_k(x_2)$$

And, x_1 is strictly better than x_2 in at least one objective:

$$i.e., \quad \exists\ k = 1, \ldots, n_k : f_k(x_1) < f_k(x_2)$$

Similarly, an objective vector, f_1, dominates another objective vector, f_2, if f_1 is not worse than f_2 in all objective values, and f_1 is better than f_2 in at least one of the objective values. Objective vector dominance is denoted by $f_1 \prec f_2$. Solution x_1 is better than solution x_2 if $x_1 \prec x_2$ which happens when $f_1 \prec f_2$.

Pareto-optimal: A decision vector, $x^* \in F$, is Pareto-optimal if there does not exist a decision vector, $x \neq x^* \in F$ that dominates it. That is, $\exists k : f_k(x) < f_k(x^*)$

Objective vector, $f^*(x)$, is Pareto-optimal if x is Pareto optimal (Deb, 2002).

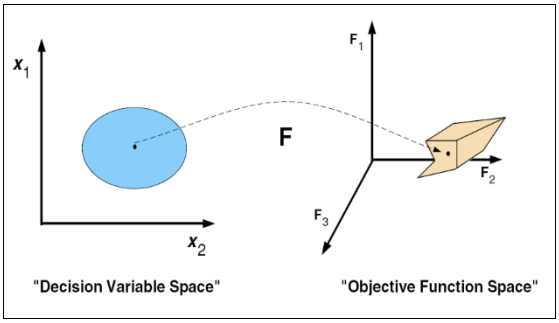

Fig. 1. Variable space and objective space

(Coello, et al, 2007)

Pareto-optimal set: The set of all Pareto-optimal decision vectors form the Pareto-optimal set, P*, that is:

$$P^* = \{x^* \in F \mid \exists\ x \in F : x \prec x^*\}$$

The Pareto-optimal set, therefore, contains the set of solutions, or balanced trade-offs, for the MOP. The corresponding objective vectors are referred to as the Pareto-optimal front.

Pareto-optimal front: Given objective vector, f(x), and Pareto-optimal solution set, P*, the Pareto-optimal front $PF^* \subseteq O$ is defined as follows:

$$PF^* = \{f = (f_1(x^*), f_2(x^*), \ldots, f_k(x^*)) \mid x^* \in P\}$$

The Pareto front, therefore, contains all the objective vectors corresponding to decision vectors that are not dominated by any other decision vector (Engelbrecht., (2002).).

2.1.2 Classification of Multi-Objective Algorithms

(Deb, 2002) and (Coello, et al, 2007) introduced two main classifications of multi-objective optimization algorithms. Deb categorized the multi-objective optimizers to classical and evolutionary algorithms including elitist and non-elitist multi-objective evolutionary algorithms. Classical algorithms, such as weighted sum method (Zadeh, 1963) and goal programming (Charnes, 1997), can only find one point as the best solution, while an evolutionary algorithm is able to deal with population of points and keeps the best solutions. Elitist algorithms apply elite-preserving operator to give opportunity to elites of population to be directly carried over to the next generation, while non-elitist algorithms do not.

The second classification is based on Coello study. Here, MOO algorithms are categorized to aggregative functions, population-based and Pareto-based. Algorithms in the first category use the techniques of combination or summation of objective functions to transform a MOP to a single-objective problem. Weighted sum method is one the common algorithm in this group. Population-based algorithms, such as Vector Evaluated Genetic Algorithm (Scheffer, 1985) without using domination concept in selection process, apply an evolutionary algorithm to make

offspring solutions. Pareto-based algorithms not only use evolutionary process in making offspring, but also apply domination concept to selection process. In this paper, multi-objective algorithms are classified in three classes: classical, evolutionary, and swam-based algorithms.

- Classical Methods

Due to the structure of the classical methods, only one Pareto-optimal solution can be expected to be found in one simulation run. Although these algorithms are easy in implementation, they require some problem knowledge such as suitable weights. Weighted Sum Method (WSM) is one of the easiest algorithms in this category. This algorithm combines a set of objectives with a single objective by pre-multiplying each objective with a user-supplied weight. It is clear that weights are the representative of the importance ratio of each objective. A MOOP can be converted to a single optimization problem as follows:

$$\begin{aligned} \text{Minimize} \quad & F(x) = \sum_{m=1}^{M} w_m f_m(x) \\ \text{Subject to} \quad & g_j(x) \geq 0 \qquad j = 1,2,\ldots,J \\ & h_k(x) = 0 \qquad k = 1,2,\ldots,K \\ & x_i^{(L)} \leq x_i \leq x_i^{(u)} \end{aligned} \tag{2}$$

where w_m is the weight of the m-th objective function.

How to select the relative weight for each objective is the most important issue in this method. Also, setting an appropriate weight vector depends on the scaling of each objective function and may take different orders of magnitude. Therefore, it is mandatory to prepare normal objectives.

- Evolutionary Algorithms

Evolutionary algorithms mimic natural principles to constitute search and optimization procedures. Fitness assignment, selection, and reproduction are three main parts of these algorithms (Weise, 2009). Genetic algorithm is the most interesting algorithm in optimization problems (Goldberg, 1989). This algorithm uses crossover and mutation operators to generate new population for next iteration. Selection is an important part of an evolutionary algorithm and can be done by many techniques such as tournament, random or deterministic methods. Many single-optimization algorithms are modified to be used in multi-objective optimization problems, such as genetic algorithm and differential evolution (Storn & Price, 1997).

Scheffer (1985) proposed the first multi-objective algorithm based on genetic algorithm called 'Vector Evaluated Genetic Algorithm'. This algorithm divides the population at every generation into M (number of objectives) equal subpopulations randomly and each of the M objective functions is used to evaluate some members in population.

Multi-objective genetic algorithm (MOGA) (Fleming & Fonseca, 1993) assigns a rank to each individual based on the number of individuals dominating it. This algorithm uses a fitness sharing technique to distinguish between solutions, and then fitness of the solutions in the same niche will be reduced.

Elitist non-dominated sorting genetic algorithm (NSGA2) is one of the most powerful algorithms proposed by Deb that uses a non-dominated sorting method for choosing points close to optimal solutions and applies the crowding distance technique in order to achieve a uniform distribution among the obtained Pareto front (Deb, 2002).

- Swarm-based Algorithms

Swarm intelligence algorithms are inspired by swarm life of creatures and simulate the behavior of animals in finding food and mating. These algorithms are based on population and try to solve the problems by using cognitive and social knowledge. Particles in swarm have a role the same as individuals in population and not only act like them, but also are able to share information between together to obtain knowledge. Particle swarm optimization *(Kennedy & Eberhart, 1995)- (Shi & Eberhart, 1998)*, ant colony optimization *(Dorigo & Gambardella, 1997)*- (Chu & Roddick, 2004), and bee colony optimization *(karaboga, 2005)* are the examples of swarm intelligence.

Multi-objective particle swarm optimization (Coello, 2004) uses an external archive to store non-dominated solutions and randomly selects a particle in a cell with fewer members as a leader of other particles, and MOCSO (Pradhan & Panda, 2012) was developed by using cat swarm optimization algorithm where the Pareto ranking scheme is incorporated and the non-dominated solutions obtained by the cats are stored in the external archive. These are the two samples of multi-objective optimizers.

2.2. Multi-objective knapsack problem

Suppose that we have a set of items whose values and weights are known and a bag with a limited capacity. To fill out the bag with the items in a way that their total value is the highest possible without exceeding the bag's capacity is known as the knapsack problem.

In a single knapsack problem, there is a fitness function as goal and a constraint to limit the solutions to the problem as follows:

$$Max \begin{cases} \sum v_i x_i, & 0 \leq x_i \leq m_i \\ St: \sum w_i x_i \leq W \end{cases} \tag{3}$$

where w,v,m,x are weight, value, permitted value, and selected value of the i-th item, respectively, and W is the total acceptable weight. This problem is known as a single-constrained problem and can be easily changed to a multi-objective minimum problem by investigating the constraint as a new function. So, the multi-objective knapsack problem is defined as follows:

$$Min \begin{cases} \sum v_i (m_i - x_i), & 0 \leq x_i \leq m_i \\ \sum w_i x_i \end{cases} \tag{4}$$

Or, it can be defined as follows:

$$Min \begin{cases} \sum v_i (m_i - x_i), & 0 \leq x_i \leq m_i \\ \max\left(\frac{\sum w_i x_i}{W} - 1.0\right) \end{cases} \tag{5}$$

Many real-life problems can be formulated such as the knapsack problem or one of its variants; for example, loading problems, project selection problems, capital budgeting problems, and cutting stock problems. Moreover,

it can be found as a sub-problem in several models, such as the partition and design of electronic circuits and the choice of a flight crew. There are several approaches that can be used to solve this kind of problems. These approaches are classified as exact methods (Luila, 2008) and approximated methods or meta-heuristics (Ghosh, 2004). The solutions obtained by the exact methods are exact solutions, while those obtained by the metaheuristics are approximate solutions. Due to the lack of computational resources and the problem sizes, the exact methods have limited application, i.e., they become ineffective, in particular in instances of great dimension. While heuristic methods, though they do not provide exact solutions, have been used as reliable alternatives, they have proven themselves as being efficient when dealing with big-sized instances, since they achieve "good" approximate solutions while using reasonable computational resources; that is what would be practically impossible to achieve through the exact methods.

2.3. Pure cat swarm optimization

Cat swarm optimization (CSO) is an optimization algorithm in the field of swarm intelligence. The CSO algorithm models the behavior of cats into two modes: 'Seeking mode' and 'Tracing mode'. Swarm is made of initial population composed of particles to search in the solution space. For example, we can simulate birds, ants, and bees and create Particle swarm optimization, Ant colony optimization, and Bee colony optimization, respectively. Here, in CSO, we use cats as particles for solving the problems (Chu, et al, 2006). In CSO, every cat has its own position composed of D dimensions, velocities for each dimension, a fitness value, which represents the accommodation of the cat to the fitness function, and a flag to identify whether the cat is in seeking mode or tracing mode. The final solution would be the best position of one of the cats. The CSO keeps the best solution until it reaches the end of the iterations (Santosa & Ningrum, 2009). Although CSO is introduced for solving single optimization, but recently is used in many applications such as clustering (Yongguo et al., 2012) and system identification (Orouskhani et al., 2013)-(Panda et al., 2011). Cat swarm optimization algorithm has two modes in order to solve the problems described below.

2.3.1. Seeking mode

For modeling the behavior of cats in resting time and being alert, we use the seeking mode. This mode is a time for thinking and deciding about next move. This mode has four main parameters which are mentioned as follows: seeking memory pool (SMP), seeking range of the selected dimension (SRD), counts of dimension to change (CDC), and self-position consideration (SPC). The process of seeking mode is described as follows:

Step1: Make j copies of the present position of cat_k, where j = SMP. If the value of SPC is true, let j = (SMP-1), then retain the present position as one of the candidates.

Step2: For each copy, according to CDC, randomly plus or minus SRD percent the present values and replace the old ones.

Step3: Calculate fitness values (FS) of all candidate points.

Step4: If all FS are not exactly equal, calculate the selecting probability of each candidate point by equation (6); otherwise, set all the selecting probabilities of each candidate point to '1'.

Step5: Randomly pick a point to move to from the candidate points and replace the position of cat_k.

$$P_i = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}} \quad (6)$$

If the goal of the fitness function is to find the minimum solution, $FS_b = FS_{max}$; otherwise, $FS_b = FS_{min}$.

2.3.2. Tracing Mode

Tracing mode is the second mode of algorithm. In this mode, cats desire to trace targets and foods. The process of tracing mode can be described as follows:

Step1: Update the velocities for every dimension according to equation (7).

Step2: Check if the velocities are in the range of maximum velocity. In case the new velocity is over-range, it is set equal to the limit.

$$V_{k,d} = V_{k,d} + r_1 c_1 (X_{best,d} - X_{k,d}) \quad (7)$$

Step 3: Update the position of cat_k according to Eq (8).

$$X_{k,d} = X_{k,d} + V_{k,d} \quad (8)$$

$X_{k,d}$ is the position of cat_k , $X_{best,d}$ is the position of the cat, who has the best fitness value, c_1 is an acceleration coefficient for extending the velocity of the cat to move in the solution space and is usually equal to 2.05, and r_1 is a random value uniformly generated in the range of [0,1].

2.3.3. Core Description of CSO

In order to combine the two modes into the algorithm, a mixture ratio (MR) is defined. This parameter decides how many cats will move into seeking and tracing. CSO is summarized below: (Orouskhani et al., 2011).

First, 'N' cats are created, and then positions, velocities, and flags of each cat should be initialized. (*) Fitness value of each cat will be evaluated according to the fitness function, and the best cat is stored into memory. In the next step, according to cat's flag, each cat will move to the seeking mode or tracing mode processes. After finishing the related process, re-pick the number of cats and set them into seeking mode or tracing mode according to MR parameter. At the end, check the termination condition; if satisfied, terminate the program, and otherwise go to (*).

3 .EMCSO: The Proposed Algorithm

Although there exists a multi-objective CSO, but no strategy is used for diversity issue that makes the algorithm unable to distinguish between solutions with the same rank

and diversity among them to be ignored. Also, this algorithm did not apply to standard test functions while using standard benchmarks to compare different algorithms is obligatory. In order to solve MOOPs and find the Pareto optimal front, pure CSO needs some changes on seeking and tracing modes to satisfy the quality and diversity of solutions. Moreover, opposition-based learning method is applied to algorithm to speed up the convergence rate.

- **Changes on Seeking Mode**

First, according to values of SRD, CDC, and SMP, positions of all cats in this mode are updated. In this mode, cats have two additional parameters: rank and crowding distance. Rank is the number of loss in competition process and crowding distance is used to demonstrate the density. So, a competition based on domination is performed among the cats; for each cat, the number of cats dominating it is assigned as rank. Then, cats are sorted: less rank, higher priority. Calculating the crowding distance metric for each cat is the second important task in this mode. Higher crowding distance value indicates that cat may be located in less density. Crowding distance calculation is shown in figure (2). For example, for an optimization problem with two objectives, the value of crowding distance for cat_i is calculated as follows:

$$d_i^1 = \frac{|f_1^{i+1} - f_1^{i-1}|}{f_1^{max} - f_1^{min}}, \quad d_i^2 = \frac{|f_2^{i+1} - f_2^{i-1}|}{f_2^{max} - f_2^{min}} \qquad (9)$$

$$\text{Crowding distance for } cat_i = d_i^1 + d_i^2$$

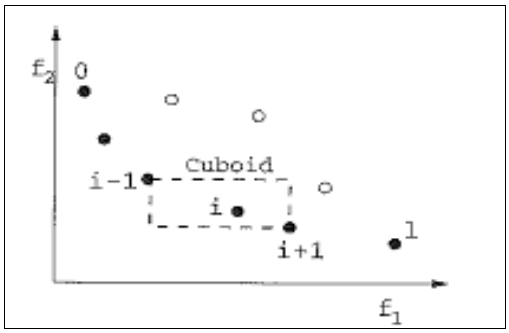

Fig. 2. The crowding distance calculation (Deb, 2002)

The last step in this mode is to select the best cats. Here, algorithm sorts the solutions (consisting of parents and offspring: cats and their copies) based on non-dominated solutions (solutions with less rank) and higher crowding distance. It is necessary to indicate that the main selection operator is rank of each cat, and crowding distance is the second criterion and should be applied for cats with the same rank.

- **Changes on Tracing Mode**

Here, cats try to move towards food to find it. So, this mode is called moving mode, and due to using the position of the best cat in velocity equation, the most important issue is how to select the leader (the best cat). In this algorithm, ten percent of the best cats are stored in an external archive and the leader of all cats is randomly selected from the archive.

- **Elitism in algorithm**

Some of the multi-objective optimization algorithms use an elite-preserving operator to favor the elites of a population by giving them an opportunity to be directly carried over to the next generation. Here, EMCSO allows the elites of population (ten percent of population size) in each epoch to participate and compete in selection process to generate the next population.

- **Using opposition-based learning (OBL)**

Generally speaking, cat swarm optimization starts with initial population and tries to improve them towards some optimal solutions. In the absence of a priori information about the solutions, random guesses are usually used. The computation time, among others, is related to the distance of these initial guesses from the optimal solution. We can improve our chance of starting with a closer (fitter) solution by simultaneously checking the opposite solution. By doing this, guess or opposite guess can be chosen as an initial solution.

Definitions:

Opposite Number: Let $x \in [a, b]$ be a real number. The opposite number $\breve{x}$ is defined as follows:

$$\breve{x} = a + b - x$$

Opposite Point: Let $P = (x_1, x_2, \ldots, x_D)$ be a point in a *D*-dimensional space, where: $x_1, x_2, \ldots, x_D \in R$, and $x_i \in [a_i, b_i]\ \forall i \in \{1, 2, \ldots, D\}$.

Opposite point $\breve{P} = (\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ is completely defined by its components: $\breve{x}_i = a_i + b_i - x_i$.

Now, by employing the opposite point definition, the opposition-based optimization can be defined as follows.

Opposition-Based Optimization: Let $P = (x_1, x_2, \ldots, x_D)$ be a point in *D*-dimensional space. Assume that 'F' is a fitness function which is used to measure the candidate's fitness. According to the definition of the opposite point, $\breve{P} = (\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ is the opposite of $P = (x_1, x_2, \ldots, x_D)$. Now, if $F(\breve{P}) \geq F(P)$, then point P can be replaced with $\breve{P}$; otherwise, we continue with P. Hence, the point and its opposite point are evaluated simultaneously in order to continue with the fitter one. So, OBL is applied in population initialization section of the proposed algorithm to enhance the speed of the convergence. Figure (3) shows the flowchart of the proposed algorithm.

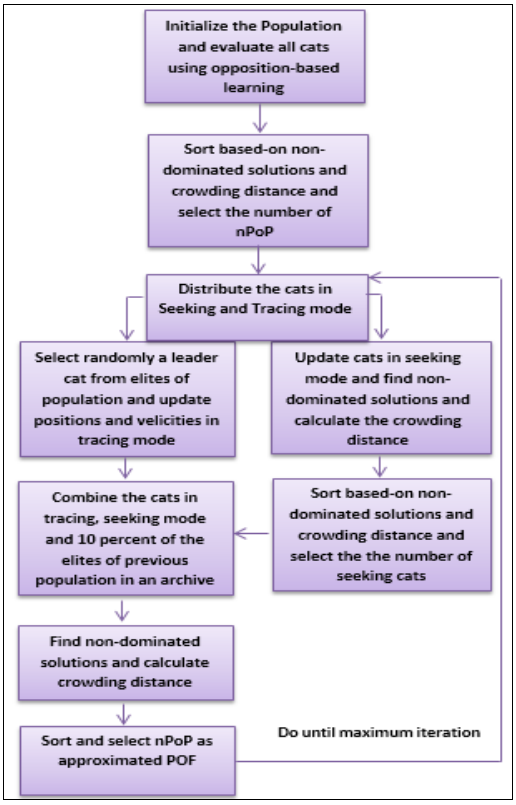

Fig. 3. Flowchart of EMCSO

4 . Experiments and Simulation Results

This section investigates the benchmarks, performance measures, and simulations of the ZDT functions.

4.1 .Multi-objective optimization problems

4.1.1. Benchmarks

ZDT1: This problem has a convex Pareto-optimal front where n = 30 and $x_i \in [0,1]$. The true POF is $f_2 = 1 - \sqrt{f_1}$ and is shown in Figure 4. (Deb, 2002)

$$f_1(x) = x_1 \tag{10}$$
$$f_2(x,g) = g(x) \times \left(1 - \sqrt{\frac{f_1}{g(x)}}\right)$$
$$g(x) = 1 + \frac{9}{n-1} \times \sum_{i=2}^{n} x_i$$

ZDT2: This problem has a non-convex Pareto-optimal front where n = 30 and $x_i \in [0,1]$. The true POF is $f_2 = 1 - f_1^2$ and is shown in Figure 5. (Deb, 2002)

$$f_1(x) = x_1 \tag{11}$$
$$f_2(x,g) = g(x) \times \left(1 - \left(\frac{f_1}{g(x)}\right)^2\right)$$
$$g(x) = 1 + \frac{9}{n-1} \times \sum_{i=2}^{n} x_i$$

ZDT3: This function has a Pareto-optimal front disconnected consisting of several noncontiguous convex parts where n = 30 and $x_i \in [0,1]$. The true POF is $f_2 = 1 - \sqrt{f_1} - f_1 \times \sin(10\pi f_1)$ and is shown in Figure 6. (Deb, 2002)

$$f_1(x) = x_1 \tag{12}$$
$$f_2(x,g) = g(x) \times \left(1 - \sqrt{\frac{f_1}{g(x)}} - \frac{f_1}{g(x)} \times \sin(10\pi f_1)\right)$$
$$g(x) = 1 + \frac{9}{n-1} \times \sum_{i=2}^{n} x_i$$

ZDT4: This problem contains 21^9 local Pareto-optimal fronts where n = 10, $x_1 \in [0,1]$, and $x_2, \ldots, x_n \in [-5,5]$. The true POF is $f_2 = 1 - \sqrt{f_1}$ and is shown in Figure 7. (Deb, 2002)

$$f_1(x) = x_1 \tag{13}$$
$$f_2(x,g) = g(x) \times \left(1 - \sqrt{\frac{f_1}{g(x)}}\right)$$
$$g(x) = 1 + 10 \times (n-1) + \sum_{i=2}^{n} x_i^2 - 10 \times \cos(4\pi f_1)$$

ZDT6: This function causes two difficulties for algorithms due to the non-uniformity of the search space: (i) the solutions are non-uniformly distributed along the global POF; (ii) the solutions are the least dense close to the POF and most dense away from the POF, where n = 10 and $x_i \in [0,1]$. The true POF is $f_2 = 1 - f_1^2$ and is shown in Figure 8. (Deb, 2002)

$$f_1(x) = 1 - \exp(-4x_1) \times \sin^6(6\pi x_1) \tag{14}$$
$$f_2(x,g) = g(x) \times \left(1 - \left(\frac{f_1}{g(x)}\right)^2\right)$$
$$g(x) = 1 + 9 \times \left[\frac{\left(\sum_{i=2}^{n} x_i\right)}{9}\right]^{0.25}$$

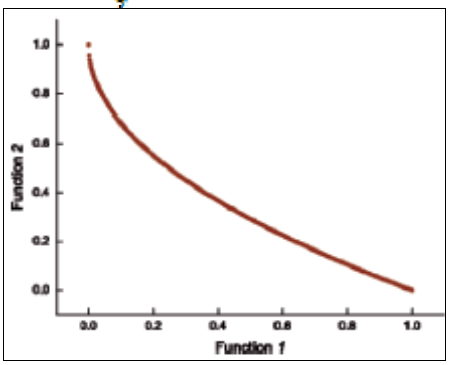

Fig. 4. The true POF of ZDT1 (Coello et al, 2007)

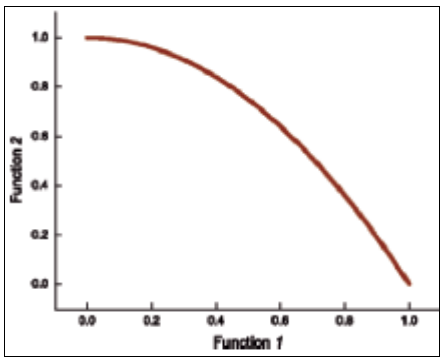

Fig. 5. The true POF of ZDT2 (Coello et al, 2007)

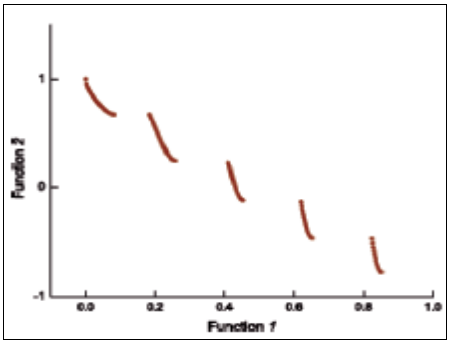

Fig. 6. The true POF of ZDT3 (Coello et al, 2007)

4.1.2. Performance measure

There are many performance metrics to quantify the performance of optimization algorithms in MOOPs. The main aim of a Multi-objective algorithm is to find the Pareto front closed to true Pareto front. There are two types of performance metrics: metrics evaluating closeness to the Pareto-optimal front and metrics evaluating diversity among non-dominated solutions. The first metric computes a measure of the closeness of set Q of N solutions from a known set of the Pareto-optimal set P*; the second metric finds the diversity among the obtained non-dominated solutions. This paper uses generational distance and inverted GD to evaluate the performance of the algorithm for closeness and spacing and spread to measure the diversity of solutions along the obtained Pareto front.

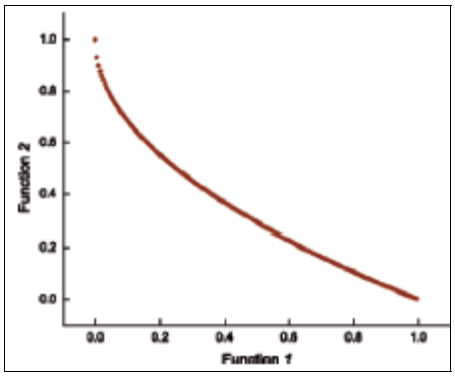

Fig. 7**. The** true POF of ZDT4 (Coello et al, 2007)

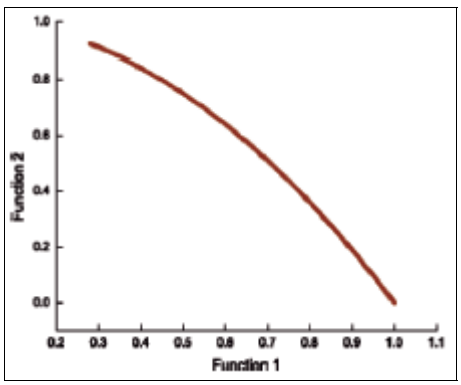

Fig. 8**. The** true POF of ZDT6 (Coello et al, 2007)

Generational distance (GD): This measure finds an average distance of the solutions of Q from P^* as follows: (Veldhuizen, 1999)

$$GD = \frac{(\sum_{i=1}^{|Q|} d_i^p)^{1/p}}{|Q|} \quad (15)$$

For p=2, parameter d_i is the Euclidean distance between solutions i in Q and the nearest member of P^*.

Inverted GD (IGD): This measure is inverted of GD and finds an average distance P^* from solutions of Q as follows: (Veldhuizen, 1999)

$$IGD = \frac{(\sum_{i=1}^{|Q|} d_i^p)^{1/p}}{|Q|} \quad (16)$$

where d_i is the Euclidean distance between the non-dominated individual of Q and the nearest individual of P^*.

Spacing: This metric is used to measure the distribution of the obtained Pareto optimal front. If the value is zero, the obtained Pareto optimal front is uniform distribution in the object space. Spacing of the POF is calculated as follows: (Schott, 1995)

$$Spacing = \sqrt{\frac{1}{|Q|-1}\sum_{i=1}^{|Q|}(\bar{d} - d_i)^2} \quad (17)$$

$$d_i = \min\left\{\sum_{m=1}^{K} |f_m(w_i) - f_m(w_j)|\right\}$$

$$w_i, w_j \in Q, \quad i, j = 1, 2, \ldots, |Q|$$

where *d* is the average value of *di*, and *K* is the number of object function.

Spread: This measures the diversity of solutions in obtained POF and can be obtained as follows: (Deb, 2002)

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}} \quad (18)$$

where d_i can be any distance measure between neighboring solutions, and $\bar{d}$ is the mean value of these distance measures. Parameter d_m^e is the distance between the extreme solutions of P^* and Q corresponding to m-th objective function. Distances for calculating the spread metric is showed in figure (9).

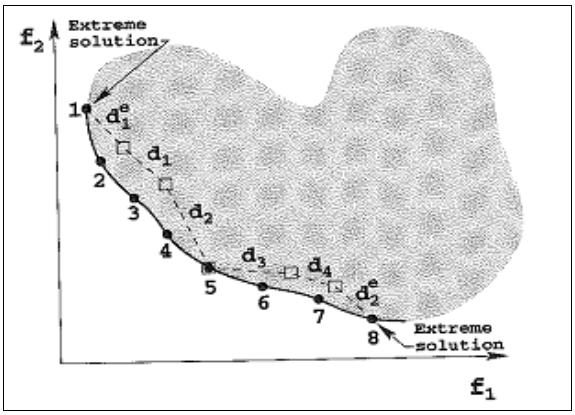

Fig. 9. Distances from the extreme solutions (Deb, 2002)

4.1.3. Simulation results

Simulation results of the proposed algorithm on ZDT functions and compared with traditional methods based on

measures of generational distance, inverted generational distance, spread, and spacing are indicated in Tables 1-4, respectively. It is notable that the best value in each benchmark is indicated in bold. Table 1 shows the results based on GD measure, and indicates that the proposed algorithm has the value of 0.00087 and 0.00035 for functions of ZDT1 and ZDT2 and gets the best performance for these two benchmarks while its rank for ZDT3-5 is forth, third, and second, respectively. Simulation results based on measure of IGD are indicated in Table 2 where the proposed method gets the best performance for ZDT1 and ZDT2 again. Results of calculating the diversity of the algorithms are shown in Tables 3 and 4. These tables indicates that the proposed algorithm does not have a good performance for measure of spread, while it achieves the best performance based on spacing metric for all functions with regard to other algorithms. Also, approximated POFs of ZDT functions by the proposed algorithm are shown in Figure 11. It is remarkable that the results shown in Tables 1-4 are obtained from (Hu & Yen, 2013)- (Ali et al., 2012)- (Abdi et al., 2012)- (Abbasian & Nezamabadipour, 2012). Moreover, in order to compare the proposed multi-objective CSO with former MOCSO, a comparison is done based on three performance metrics for a test function which is introduced in (Xue & Sanderson, 2003), and the results are indicated in Table 5. Obtained results show that the proposed algorithm has the best performance for the measures of GD and Spacing.

4.2 . Multi-objective knapsack simulation

In this section, the proposed algorithm is applied to solve the multi-objective knapsack problem. Here, the characteristics of the knapsack are as follows: it has 10 items
between
$[0,20], 10 < w < 20, W_{max} = 1800, 20 < v < 100,$ and $10 < m < 20$
. Figure 10 shows the POF of multi-objective knapsack problem when the goal is to maximize the selected value, while the weight is needed to minimize. It shows that for the maximum acceptable weight (1800), the maximum selected value is 4521, while the maximum value (10000) is selected when the weight reaches 6013.

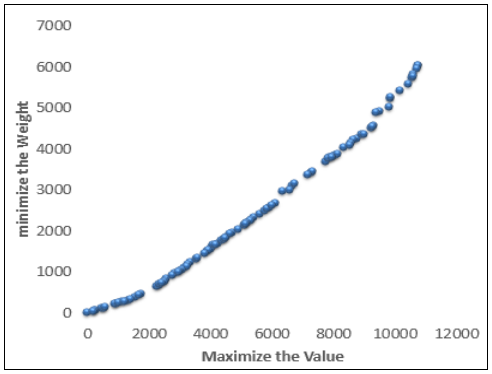

Fig. 10. Simulation of multi-objective knapsack problem: Maximum value VS Minimum weight

Table 1
Comparison between the proposed algorithm and traditional methods by measure of GD (1st row: Mean, second row: Std)

Functions	NSGA2	SPEA	PAES	MODE	ADEA	DEMO	MDEA	MOBBO/DE	EMCSO
ZDT1	0.0334	0.0017	0.0820	0.0058	0.0027	0.0011	0.0009	0.0021	**0.00084**
	0.0047	1E-6	0.0086	0	0.0003	0.0001	5E-6	0.0001	5.23E-9
ZDT2	0.0723	0.0013	0.1262	0.0055	0.0022	0.0007	0.0006	0.00083	**0.00055**
	0.0316	0	0.0368	0	0.0002	0.0000	0	0.00005	2.36E-9
ZDT3	0.1145	0.0475	0.0238	0.0215	0.0027	0.0012	**0.0011**	0.0105	0.00282
	0.0049	4.2E-5	4.7E-5	1E-5	0	0.0001	9.1E-5	0.0015	1.11E-7
ZDT4	0.5130	7.3402	0.8548	0.6389	0.1001	**0.0410**	0.0489	0.3653	0.22977
	0.1184	6.5725	0.5272	0.5002	0.4462	0.0639	0.5363	0.2566	0.09601
ZDT6	0.2965	0.2211	0.0854	0.0262	0.0006	0.0006	**0.0004**	0.0034	0.00059
	0.0131	0.0004	0.0066	0.0008	6E-5	2E-5	5.5E-5	0.0022	3.64E-9

Table 2
Comparison between the proposed algorithm and traditional methods by measure of IGD (1st row: Mean, second row: Std)

Functions	pA MPSO	Ag MPSO	Cd MPSO	Cluster MPSO	Pd MPSO	NSGA2	SPEA2	MOED/D	EMCSO
ZDT1	4.013E-3	1.18E-1	4.24E-3	1.25E-2	5.57E-1	5.05E-1	4.14E-3	4.03E-3	**8.17E-4**
	6.28E-5	8.03E-2	2.58E-4	1.78E-3	1.95E-1	7.36E-2	1.77E-8	5.58E-5	3.11E-10
ZDT2	4.09E0-3	1.07E-2	4.28E-3	1.78E-2	6.87E-2	7.57E-1	4.10E-3	3.84E-3	**1.88E-3**
	4.81E-5	7.47E-3	1.13E-4	5.09E-3	4.38E-2	1.43E-1	2.65E-8	4.34E-5	1.21E-8
ZDT3	3.323E-3	3.60E-1	**3.06E-3**	1.04E-1	3.05E-1	3.57E-1	3.16E-3	8.42E-3	2.49E-2
	9.95E-5	9.84E-2	7.13E-5	7.05E-2	1.003E-1	3.95E-2	0	7.01E-3	6.77E-8
ZDT4	7.97E-3	5.79	5.91	3.98	4.03	2.52E+1	2.49E+1	**4.86E-3**	0.21451
	1.470E-3	2.98	4.51	2.61	1.65	7.214	7.25E-5	8.41E-4	0.0650
ZDT6	3.406E03	4.68E-1	**2.98E-3**	4.39E-1	2.46	1.65	5.32E-3	3.99E-3	9.55E-3
	2.287E-4	7.67E-1	1.54E-4	2.36E-2	8.16E-1	9.80E-1	2.65E-8	6.01E-5	6.70E-10

Table 3
Comparison between the proposed algorithm and traditional methods by measure of Spread (1st row: Mean, second row: Std)

Functions	NSGA2	SPEA	PAES	PDEA	ADEA	MDEA	MODEA	MOBBO/DE	EMCSO
ZDT1	0.3903	0.7845	1.2297	0.2985	0.3828	**0.2837**	0.3332	0.58541	0.6142
	0.0018	0.0044	0.0007	0.0007	0.0014	0.0029	0.0009	0.0401	0.0008
ZDT2	0.4307	0.7551	1.1659	**0.3179**	0.3457	0.4504	0.3219	0.57439	0.6063
	0.0047	0.0045	0.0076	0.0013	0.0039	0.0042	0.0004	0.0686	0.0021
ZDT3	0.7385	0.6729	0.7899	0.6238	0.5257	**0.2993**	0.7303	0.75083	0.5631
	0.0197	0.0035	0.0016	0.0002	0.0430	0.0233	0.0000	0.0551	0.00008
ZDT4	0.7026	0.7984	0.8704	0.84085	0.43630	0.40638	**0.37268**	0.68631	0.6085
	0.0646	0.0146	0.1013	0.0357	0.1100	0.0623	0.0034	0.0563	0.0384
ZDT6	0.66802	0.84938	1.1530	0.47307	0.36110	0.30524	**0.30259**	0.75057	0.5500
	0.0099	0.0027	0.0039	0.0217	0.0361	0.0194	0.00007	0.1083	0.0010

Table 4
Comparison between the proposed algorithm and traditional methods by measure of Spacing (1st row: Mean)

Functions	SPEA2	NSGA2	TVMOPSO	MOGSA	NF-MOGSA	EMCSO
ZDT1	0.0156	0.0007	0.0041	0.0076	0.0005	**0.000115**
ZDT2	0.0167	0.0001	0.0026	0.0033	0.0007	**0.000100**
ZDT3	0.0059	0.0002	0.0094	0.0136	0.0015	**0.000129**
ZDT6	0.0132	0.0003	0.0023	0.0081	0.0007	**0.0000994**

Table 5
Comparison of the proposed algorithm with traditional methods and former MOCSO for test function (1st row: Mean)

Measure	NSGA2	MOPSO	MOCSO	EMCSO
GD	0.0265	0.001	0.0007692	**0.00072284**
Spacing	0.009	0.0089	0.009	**0.00011664**
Spread	0.6594	0.72	**0.6077**	0.6495

---

* Algorithms indicated in Tables 1-5 are referred as: NSGA2 (Deb et al, 2002), SPEA (Zitzler, 1999), PAES (Knowles & Corne, 1999), MOBBO/DE (Abdi et al, 2012), DEMO (Robic & Filipi, 2005), TVMOPSO (Tripathi et al, 2007), MOGSA (Hassanzadeh, 2010), NF-MOGSA (Abbasian & Nezamabadipour, 2012), MODE (Xue & Sanderson, 2003), ADEA (Qian & Li, 2008), MDEA (Ali et al, 2009), PAMPSO, AgMPSO, CdMPSO, ClusterMPSO, PdMPSO (Hu & Yen, 2013), SPEA2 (Zitzler et al, 2001), MOED/D (Zhang & Li, 2007), PDEA (Madavan, 2003), MODEA (Ali et al, 2012), MOPSO (Coello, 2004)

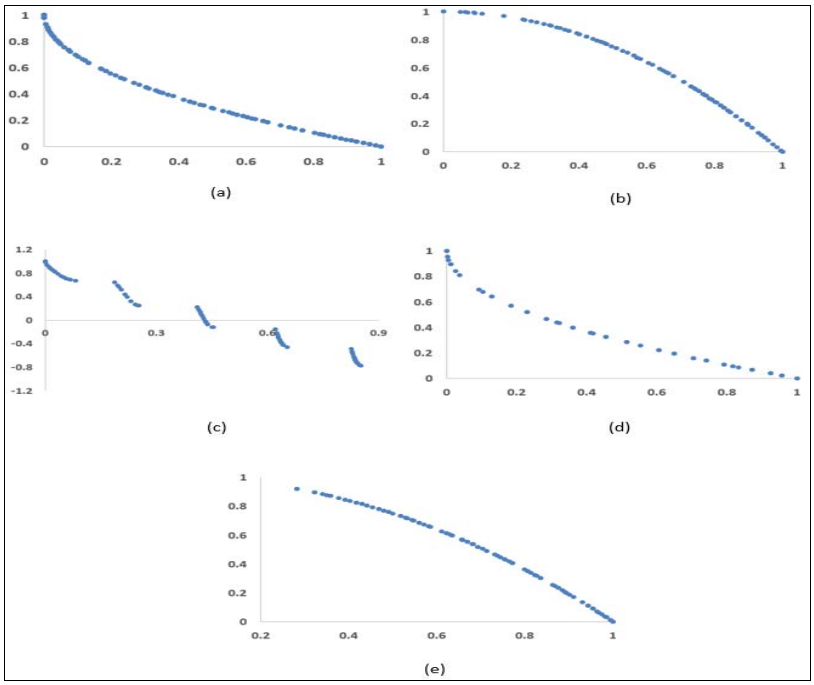

Fig. 11. Obtained POF by the proposed algorithm. a: ZDT1, b: ZDT2, c:ZDT3, d:ZDT4, e:ZDT6

5. Conclusion

This paper proposes a novel multi-objective optimization algorithm. The proposed algorithm uses cat swarm for making offspring solutions and applies the non-dominated sorting method and crowding distance technique to satisfy closeness to the true Pareto front and diversity of the solutions. Also, an elite-preserving operator is incorporated to keep the elites of population to participate in the next generation and an opposition search-based technique is applied to increase the rate of the convergence. The simulation is done on ZDT test functions and analyzed based on closeness and diversity-based performance measures. Moreover, the proposed method is applied to solve a multi-objective knapsack problem. Simulation results show that the proposed algorithm in comparison with traditional methods has better performance for functions of ZDT1 and ZDT2. Adapting the parameters of the proposed method in order to move from exploration to exploitation, global to local, and using the obtained knowledge of particles by expert systems, such as fuzzy approaches, can be investigated in future studies.

References

Abbasian, M., & Nezamabadipour, H. (2012). Multi-objective gravitational search algorithm using non-dominated fronts. *Journal of electrical engineering, Vol 41, No 1 (in Persian)*.

Abdi, S., Teshnehlab, M., Aliyari, M., & Golahmadi, H. (2012). Designing a multi-objective optimization algorithm with BBO/DE (in persian). *Intelligent systems in electrical engineering, No3*.

Ali, M., Pant, M., & Abraham, A. (2009). A modified differential evolution algorithm and its application to engineering problems. *In: Proceedings of International Conference of Soft Computing and Pattern Recognition (SoCPaR-2009), ,* (pp. pp.196–201).

Ali, M., Siarry, P., & Pant, M. (2012). An efficient Differential Evolution based algorithm for solving multi-objective optimization problems. *European Journal of Operational Research 217*, 404–416.

Bagherinejad, J., & Dehghani, M. (2016). A Non-dominated Sorting Ant Colony Optimization Algorithm Approach to the Bi-objective Multi-vehicle Allocation of Customers to Distribution Centers. *Journal of Optimization in Industrial Engineering*, Volume 9, Issue 19, Page 61-74.

Charnes, A. (1997). Goal programming and multiple objective optimization,. *European journal of operational research. 1, ,* 39–54.

Chu, S., & Roddick, J. (2004). Ant colony system with communication strategies. *Information Sciences 167*, 63-76.

Chu, S., Tsai, P., & Pan, J. (2006). Cat Swarm Optimization. *LNAI 4099, 3 (1), Berlin Heidelberg: Springer-Verlag*, (pp. pp. 854–858).

Coello, A. C., Gary, B., & Veldhuisen, A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems.*

Coello, C. (2004). Handling multi objectives with pso, . *Evolutionary Computation, IEEE Transactions on (Volume:8 , Issue: 3).*

Deb, K. (2002). A Fast and Elitist Multiobjective Genetic Algorithm. *IEEE Transactions On Evolutionary Computation, Vol. 6, NO. 2*.

Deb, K. (2002). *MultiObjective optimization using evolutionary algorithms.* Wiely.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation, 6(2)*, 182–197.

Dorigo, M., & Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. On Evolutionary Computation. 26 (1) ,* 53-66.

Engelbrecht., A. ((2002).). *Computational Intelligence.* John Wiley & Sons Ltd.

Fleming, C., & Fonseca, P. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms,* (pp. 416–42).

Ghosh, A. (2004). Evolutionary Algorithms for Multi-Criterion Optimization: A Survey. *International Journal of Computing & Information Sciences, 2 (1),* pp. 38-57.

Goldberg, D. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. *Kluwer Academic Publishers*.

Hassanzadeh, H. (2010). A Multi-objective Gravitational Search Algorithm. *Computational Intelligence, Communication Systems and Networks (CICSyN), Second International Conference on,* (pp. pages 7-12).

Hassanzadeha, R., Mahdavib, I., & Mahdavi-Amiric, N. (2014). Ant Colony Optimization for Multi-objective Digital Convergent. *Journal of Optimization in Industrial Engineering*, Volume 7, Issue 16, Page 1-19.

Hu, W., & Yen, G. (2013). Adaptive Multiobjective Particle Swarm Optimization Based on Parallel Cell Coordinate System. *IEEE Transactions on Evolutionary Comutation*.

Kahejvand, V., Hossein, P., & Zandieh, M. (2014). Multi-objective and Scalable Heuristic Algorithm for Workflow Task Scheduling in Utility Grids. *Journal of Optimization in Industrial Engineering*, Volume 7, Issue 14, Page 27-36.

karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical report, .* Erciyes University.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks.*

Knowles, J., & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. *in Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press,* (pp. pp. 98–105).

Luila, E. P. (2008). *Problema da Mochila Multi-critério, Aspectos Algorítmicos e Implementação Informática. .* Tese de Mestrado, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Madavan, N. (2003). Multi-objective optimization using a Pareto differential evolution approach. *Proceeding of the Congress on Evolutionary Computation", Vol.2,* (pp. pp.862-869).

Mehdizadeh, E., & Kivi, F. (2014). Three Metaheuristic Algorithms for Solving the Multi-item Capacitated Lot-sizing Problem with Product Returns and Remanufacturing. *Journal of optimization in industrial engineering*, Volume 7, Issue 16, Page 41-53.

Mirzazadeh, M., Shirdel, G., & Masoumi, B. (2011). A Honey Bee Algorithm To Solve Quadratic Assignment Problem. *Journal of optimization in industrial engineering*, Volume 4, Issue 9, Page 27-36 .

Naderi, B., & Sadeghi, H. (2012). A Multi-objective simulated annealing algorithm to solving flexible no-wait flowshop scheduling problems with transportation times. *Journal of Optimization in Industrial Engineering*, Volume 5, Issue 11, Page 33-41.

Orouskhani, M., Mansouri, M., & Teshnehlab, M. (2011). Average- Inertia weighted Cat swarm optimization. *LNCS, Berlin Heidelberg: Springer-Verlag*, (pp. pp 321– 328).

Orouskhani, M., Mansouri, M., Orouskhani, Y., & Teshnehlab, M. (2013). A hybrid method of modified cat swarm optimizationand gradient descent algorithm for training anfis. *International Journal of Computational Intelligence and Applications Volume 12, Issue 02,*.

Panda, G., Pradhana, P., & Majhib, B. (2011). IIR system identification using cat swarm optimization. *Expert Systems with Applications, Volume 38, Issue 10*, Pages 12671–12683.

Pradhan, P., & Panda, G. (2012). Solving multiobjective problems using cat swarm optimization. *Expert Systems with Applications, Volume 39, Issue 3*, Pages 2956–2964.

Qian, W., & Li, A. (2008). Adaptive differential evolution algorithm for multiobjective optimization problems. *Applied Mathematics and Computation 201*, 431–440.

Rahmanian, R., Ghaderi, A., & Mehrabad.M. (2012). A Simulated Annealing Algorithm within the Variable Neighbourhood Search Framework to Solve the Capacitated Facility Location-Allocation. *Journal of Optimization in Industrial Engineering*, Volume 5, Issue 10, Page 45-54.

Robic, T., & Filipi, B. (2005). DEMO: Differential Evolution for Multiobjective Optimization. *LNCS 3410*, (pp. pp. 520–533).

Santosa, B., & Ningrum, M. (2009). Cat Swarm Optimization for Clustering . *International Conference of Soft Computing and Pattern Recognition*, (pp. pp 54-59).

Scheffer, M. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proceedings of the 1st International Conference on Genetic Algorithms.

Schott, J. R. (1995). *Fault tolerant design using single and multi-criteria genetic algorithms. Master's thesis.*

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *Proceedings of IEEE International Conference on Evolutionary Computation*, (pp. pp. 69–7).

Storn, R., & Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 341–359.

Tizhoosh, H. (2005). Opposition-Based Learning. *Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation.*

Tripathi, P., Bandyopadhyay, S., & Pal, S. (2007). Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients. *Information sciences 177*, 5033-5049.

Veldhuizen, D. V. (1999). *Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations, PhD thesis.*

Weise, T. (2009). *Global Optimization Algorithms, Theory and applications.*

Xue, F., & Sanderson, A. (2003). Pareto-based multi-objective differential evolution. *Proceedings of the 2003 Congress on Evolutionary Computation*, (pp. pp.420-431).

Yongguo, L., Xindong, W., & Yidong, S. (2012). Cat swarm optimization clustering (KSACSOC): A cat swarm optimization clustering algorithm . *Scientific Research and Essays Vol. 7(49)*, pp. 4176-4185.

Zadeh, L. (1963). Optimality and Non-Scalar-Valued Performance Criteria. *IEEE Trans Autom Control 8*, 59–60.

Zhang, Q., & Li, H. (2007). MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput., Vol. 11, no. 6*, pp. 712-731.

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications.* Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology (ETH).

Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: improving the strength pareto evolutionary algorithm.* Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, Technical report TIK- report 103.

This article can be cited: Orouskhani, M., Teshnehlab, M. & Nekoui, M.A. (2018). EMCSO: An Elitist Multi-Objective Cat Swarm Optimization . *journal of Optimization in Industrial Engineering*. 11(2), 2018, 105-115.

URL: http://qjie.ir/article_538170.html
DOI: 10.22094/joie.2017.500.12