# A Collective Intelligence Strategy for Enhancing Population-based Optimization Algorithms

Azam Asilian Bidgoli*, Shahryar Rahnamayan, SMIEEE
Nature Inspired Computational Intelligence (NICI) Lab
Department of Electrical, Computer, and Software Engineering
Ontario Tech University, Oshawa, Canada
azam.asilianbidgoli@uoit.ca, shahryar.rahnamayan@uoit.ca

***Abstract*—Population-based algorithms are a well-established category of metaheuristic optimization algorithms in which individuals collaborate with each other to find the optimal solution in a search space. During the search process, each individual provides a partial intelligence which can assist the population movement toward promising regions. In this paper, a dimension-wise strategy is proposed to collect the intelligence of whole population to generate a new trial candidate solution. For new individual, the value of each variable is calculated using the votes of a more-crowded cluster of individuals obtained on each dimension (one-dimensional clustering). Accordingly, a group of candidate solutions in the population collaborate to determine a variable value of new individual. Utilizing this strategy, collective intelligence (CI) aims the algorithm to find better candidate solutions. Since the proposed method keeps untouched all other parts of the algorithm, it can be used with any population-based algorithm. This paper presents the modification of two well-known population-based algorithms based on the proposed strategy in utilizing Collective Intelligence (CI), Differential Evolution (CIDE) and Particle Swarm Optimization (CIPSO). In conducted experiments, two proposed algorithms are compared with classical version of DE and PSO on 30 functions of CEC-2017 benchmark. The results indicate that the proposed method generates an individual with better objective function value than many of the individuals in the population which leads totally better results in overall.**

***Index Terms*—Collective Intelligence, Population-based Algorithms, Differential Evolution (DE), Particle Swarm Optimization (PSO), Metaheuristics, Voting, Dimension-wise Clustering**

## I. Introduction

Population-based metaheuristic algorithms are one of the well-known family of algorithms for solving the global optimization problems [1], [2]. These algorithms are often biological or social inspired algorithms which represent a range of universal problem solvers. They begin the optimization search by using an initial population which consists of candidate solutions; then they try to find the optimal solutions using an iterative and stochastic processes. Despite the importance of optimization task and designing numerous algorithms, the capability of solving many global optimization problems is still not satisfactory. Hence, researchers try to utilize various strategy to improve the existing algorithm or design new algorithms.

Two most popular families of population-based algorithms are evolutionary algorithms (EAs) [3] and swarm intelligence algorithms [4]. In both families of algorithms, the use of some sort of implicit or explicit collective intelligence of individuals in the population can be recognized. Collective intelligence is an integration process of collective behaviors of individuals in social groups or collective functions of components in computational intelligent systems [5]. A transdisciplinary study on metaheuristic computing and social psychology may explain a set of key mechanisms of collected intelligence.

Evolutionary algorithms work based on generating new individuals using crossover and mutation operations during the optimization process. Then, elite solutions are selected to be transferred to the next generation [6]. Regarding evolutionary algorithms, one of the operations which is defined as use of the collective intelligence of the population is crossover. This operation generates new offspring using combining two or more individuals of the population. When the number of parents increases, in fact the participants on collective intelligence increase. Based on some research investigations, increasing the number of parents in the Genetic Algorithm (GA) [7] improves the convergence rate of the classical GA [8]. In all multi-parent crossovers, the parents number plays a key role to keep high the population diversity and to reduce the risk of premature convergence [9]. In [10], authors proposed taking advantage of the collective information from the $m$ best candidate solutions in continuous DE by combining them to use the combinational solution crossover and mutation operations. In [5], the properties of collective intelligence and its applications in metaheuristic computing are represented.

Swarm-based algorithms, second group of metaheuristics, are made up of simple particles which cooperate with each other and interact with their environment to move toward the region of the optimal solution [11]. The movement of each particle is updated based on some rules to improve its behavior in terms of getting closer to the target. Since the position of the best particle is a part of rules' definition, it can be stated that the collective intelligence of population affects movement of whole population. However, in the classical version of the algorithm, the intelligence doesn't come from the whole population, instead it is from the best members of the population or the top best particles. In [12], the global best solution of the PSO was modified by taking into account all local optimums of particles multiplied by their objective function values. As another example for collaborating of individuals of the population to find the optimal solution in the search space is Ant Colony Algorithm (ACO) [13]. The main principle founding ant-based algorithms is to leave traceable signs for letting other to follow individuals guided through them [14]. However, other algorithms such as Particle Swarm intelligence Optimization(PSO) [15] and Differential Evolution (DE) [16] use the implicit memory for their collaboration; but ACO uses an explicit memory.

All mentioned perspectives of collective intelligence in metaheuristic algorithms involve only a part of the population or use the its benefit implicitly in their operations. There has been conducted a study on utilizing whole population intelligence but on discrete optimization problems. The voting operator has been used as collective intelligence on discrete problems in [17]. The authors generate a new individual by voting on variables' value of individuals. Increasing the participation of individuals to utilize the intelligence of more components in any type of optimization problem is the main idea of this paper.

The proposed method generates a new individual using the collective intelligence of whole population. Each variable of the new vector is obtained using a continuous voting scheme on a crowded cluster of individuals which is earned by applying a clustering algorithm. By considering this idea, whole population contribute to generate a high quality candidate solution. Because the clustering task is dimension-wise, it doesn't suffer increasing the number of dimensions. Two popular metaheuristic algorithms, DE and PSO algorithms are modified based on the proposed scheme. In order to evaluate the performance of proposed method, some experiments are conducted on functions of CEC-2017 benchmark set to compare the collective intelligence-based versions of DE and PSO (called CIDE and CIPSO, respectively) with their parents.

The remaining sections of this paper are organized as the follows. Section 2 presents a background review on applied algorithms. Section 3 provides a detailed explanations about the proposed method. The performance of the proposed method is investigated over 30 benchmark functions in the Section 4. The paper is concluded in the Section 5.

## II. Background review

In this section, a brief explanation of parent algorithms which the proposed method are compared with, is provided. DE and PSO are two well-known population-based optimization algorithms which are described. In addition, the k-means algorithm as utilized clustering algorithm for collecting the intelligence is explained in this section.

### A. Differential Evolution

DE is an evolutionary algorithm which is originally designed to solve continuous optimization problems. The main part of DE is its mutation operator, otherwise, all other steps of algorithm is kept untouched rather than other evolutionary algorithms. Similar to other population-based algorithm, DE starts with a uniform random initialization of the population. Objective function is calculated for each individual in the population. Next generation should be constructed using selecting the best individuals from current and produced individuals. In order to generate new individuals, an evolutionary mutation is conducted. For each individual, an offspring is generated using three randomly selected individuals from population based on Eq. 1.

$$v_{j,i} = x_{j,i_1} + F.(x_{j,i_2} - x_{j,i_3}), \tag{1}$$

where $x_{j,i_1}$, $x_{j,i_2}$, and $x_{j,i_3}$ are three randomly selected individuals. For each variable of new individual ($v_i$), the value of variables of two selected individuals are subtracted using a mutation factor, $F$, then the result is added to the third selected individual. This operator generates new trial candidate solution to get away from the $x_{i_3}$ candidate and move toward the $x_{i_1}$ and $x_{i_2}$ candidate solutions. Some or all genes of parent are selected based on Crossover Rate ($C_R$) to be replaced by genes of generated individual. After producing offsprings in number of population size, DE selects best individuals using comparing parent and its corresponding offspring. Each of them with better objective value is remained in the population to be transefered to the next generation. This steps continue to meet a stopping criterion.

### B. Particle Swarm Optimization

PSO is one of the well-known metaheuristic algorithm. It works based on movement of swarm of individuals which called particles. Initial population is generated uniform randomly. Each particle is defined using a position which means the value of variables and a velocity of their movement. Optimization forces all particles to move toward a global optimum solution. At each generation, PSO updates the position and velocity of particles based on Eq.s 2and 3.

$$v_{id} = w.v_{id} + c_1.r_1.(p_{id} - x_{id}) + c_2.r_2.(p_{gd} - x_{id}) \tag{2}$$

$$x_{id} = x_{id} + v_{id}, \tag{3}$$

where the position of an particle, $x_{id}$, is updated by $v_{id}$ which shows the particle velocity. The position of the best candidate solution in a predefined neighborhood ($p_{gd}$) along with the position of the best movement achieved so far associated with each particle ($p_{id}$) changes the particle's velocity toward the global and local best positions. $r_1$ and $r_2$ are two uniform random vectors in range $(0, 1)$. $c_1$, $c_2$ are two parameters representing the particle's confidence in itself and in the swarm experiences, respectively and $w$ determines the effect of previous velocity, called inertia weight. There are variant kinds of topology schemes [18] which are defined for PSO to identify $p_{gd}$. The most commonly used topology is a 2-neighborhood undirected regular ring topology [19]. In the ring topology, each particle, $i$, has two neighbors, $i - 1$ and $i + 1$, such that the overall network has the form of a ring. Global best is selected among particle's neighbors. In another variant of PSO which is named fully informed PSO (FIPs) [20], each particle is not influenced by the best particle among its neighbors. Instead, all its neighbors are used to update the velocity. After updating velocity and position of each particle, it is expected that swarm moves toward global solution. The generation updating continues to meet a stopping criterion such as predefined number of objective function evaluations.

### C. $k$-means clustering

Clustering is a task of dividing the data points into a number of clusters such that data points in the same clusters have high similarity while they are very dissimilar to entities in other clusters [21]. A clustering task needs two components to be determined, clustering algorithm and utilizing similarity measure. $k$-means clustering [22] is a well-known and simple clustering algorithm which puts data points into $k$ groups based on a similarity measure. This method needs a predefined number of clusters to start clustering. At the first, $k$ points are randomly selected among data points as initial centres of clusters. Then, a similarity measure like Euclidean distance is applied to assign each data point to corresponding nearest center. So $k$ initial clusters have been constructed. Next step is the updating of centers and renewing the clusters. So that the mean of data points of each cluster are computed as new centers. In order to update clusters based on new centroids, the distance of each data point is computed from all centers and it is assigned to nearest cluster. These steps continues until the convergence criterion such as predefined iterations or no change met. The pseudo-code of the algorithm is presented in the Algorithm 1.

```
Input  : $D = \{x_1, x_2, x_3, ..., x_n\}$: Set of data points
         which should be clustered, $K$: Number of
         clusters
Output: $C = \{c^{(1)}, c^{(2)}, c^{(3)}, ..., c^{(K)}\}$: Set of clusters
Initialize the centroids of clusters, $\mu_1, \mu_2, \mu_3, ..., \mu_K$
  randomly
repeat
    Assign each data point to closest cluster based on
      Euclidean distance: $c^{(i)} = \arg\min_j ||x^{(i)} - \mu_j||^2$
    Calculate new centroid for each cluster:
        $\mu_j = (\sum_{c^{(i)}=j} x^{(i)})/n_j$; // $n_j$ is the
        number of members in jth cluster
until Convergence criteria met (no change in clusters);
```

**Algorithm 1:** Pseudo-code for K-means algorithm

## III. Proposed method

Collective intelligence of population in metaheuristic algorithms can yield to produce more elite solutions. In population-based algorithms, it is desirable that the population moves toward region of optimal solution but in most of them, each individual tries to enhance its own position. Accordingly, after producing better candidate solution, the previous one is replaced with new one. However, in some algorithms there is collaboration between a part of individuals to assist each other to produce better individual, utilizing collective intelligence of whole population is not proposed. As mentioned previously, as a sample of collaboration between individuals, we can mention producing offspring using two or three individuals in most of evolutionary algorithms such as DE or Genetic Algorithms [23]. In PSO algorithm, collaboration can be defined as effect of global and local best solutions to update the position of each individual, but in all these cases, there is a partial collaboration of individuals. In this section, the details of the proposed scheme and its embedding in population-based algorithms are provided.

### A. Collective intelligence strategy

In this paper, collective intelligence of population is employed to increase elitism by producing generally a better candidate solution (CI-based trial vector). The method utilizes continues voting between individuals. The value of each variable of CI-based vector is produced using averaging between individuals that are more similar to each other in corresponding variable. Since in population-based algorithms, the individuals move toward better region of search space i.e, close to optimum point, so it is expected that among generations, the variables achieve better values to optimize objective function. Thus, voting among the variables of individuals creates a novel candidate solution which collects the intelligence of the whole population. In other words, it is preferred to construct individual with variable values which are obtained based on sort of aggregation established by all members of the population. The regular operators of algorithms (e.g., mutation and crossover in evolutionary algorithms) assemble good solutions in a slow manner (if they do) because a part of intelligence of population construct new individuals while by using generating CI-based vector (e.i., collecting valuable knowledge of all individuals at once), the process will be accelerated.

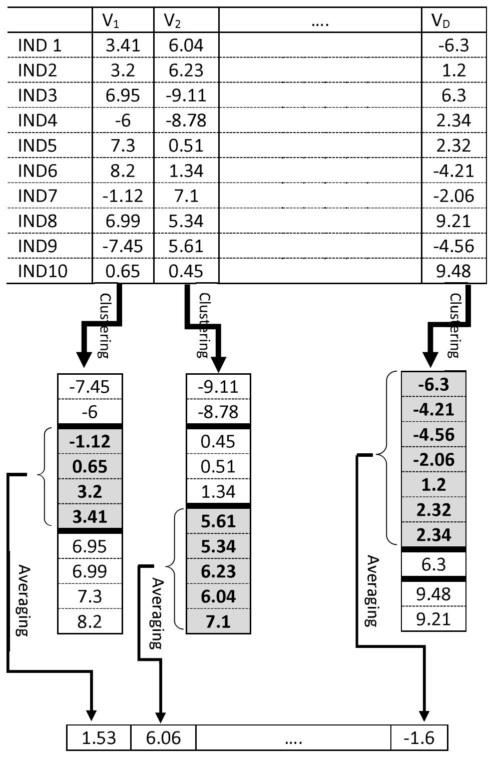
For this purpose, a new candidate solution is generated as follows. Suppose population includes the individuals, $< x_1, x_2, x_3, ..., x_N >$ and each individual, $x_i$, is a $D$ dimensional vector, $x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D}$. For each dimension, $j$, $k$-means clustering is performed on individuals in the population based on only the value of corresponding variable. It means that a one dimensional $k$-means clustering is performed on $x_{1,j}, x_{2,j}, x_{3,j}, ..., x_{N,j}$ which are the one dimensional points related to $j$-th variable. So that for each variable $j$, the population is grouped into $K$ clusters based on the similarity of individuals in corresponding variable. Therefore, individuals with similar values of variable are collected in the same cluster. The cluster with maximum number of members shows strong agreement on that variable. Accordingly, creating a new candidate based on the values of individuals in the most crowded cluster will help the search algorithm in exploration phase. The value of variable $j$ in new candidate solution is acquired by averaging the values of variable $j$ in members of most crowded cluster. It means that the continuous voting between individuals decides for the value of variables in new trial vector. Consequently, on each dimension, one-dimensional $k$-means clustering is applied to construct clusters of individuals and then selecting most crowded cluster determines the value of corresponding variable in CI-based individual. It is worth to mention that since $k$-means clustering is done per dimension, however, the time complexity increases by increasing the dimension of the problem, it doesn't affects the $k$-means efficiency. In other words, this is a clustering for each dimension not a clustering method for whole population in the search space. For whole process, one-dimensional clustering would be applied. Fig. 1 illustrates the process of generating a trial vector using proposed collective intelligence-based method. In this example, population has 10 individuals and the most crowded cluster is indicated by gray color. In the case with same number of members in the clusters, a random cluster would be selected. As another perspective, the clustering process and generating CI-based vector in two dimensional search space is presented in Fig. 2. As can be seen, the value of variables on each dimension are clustered to three clusters, average of values in most crowded cluster would be the value of CI-based vector.

Similar to other population-based algorithms, optimization process begins with a randomly generated initial population. At each iteration, new individuals are generated by using operators or current individuals are updated. Based on the selection strategy of each algorithm, updated population as the next generation should move to the next iteration. According to proposed method, at the end of each iteration, a new trial vector is produced based on the collective intelligence scheme. The generated individual can be entered into population as a candidate solution if it is good enough. The trial vector is compared with the worst candidate solution in the population and it is replaced by the new vector if it has worse objective value. The detail of replacement strategy will be discussed in the following subsection.

In the optimization algorithms, as process progresses, the population is expected to move toward the more promising regions; therefore, by trusting on the only available knowledge distributed over the population, the variables of generated vector would gain values which gradually become closer to the optimal values. In other word, the density of individuals in each dimension shows the promisness of each region in search space. It means that we expect that majority of voters can take the correct decision especially when the number of iterations increases (in fact that is a common assumption for any democratic system) . It is worth mentioning that the proposed method does not affect on any other part of optimization algorithm. Therefore, the proposed method can be applied in any population-based algorithm and any type of optimization problem. It means, the proposed CI-based scheme is population level approach not an operation level one.

### B. *Embedding proposed scheme in a population-based optimization algorithm: DE and PSO as case studies*

The CI-based candidate solution can influence movement of whole population toward better regions. In this paper, DE and PSO algorithms are modified using our proposed method. Since mutation operator in DE algorithm generates a new individual based on other candidate solutions in population, the CI-based individual can affect on whole population when



| | $V_1$ | $V_2$ | .... | $V_D$ |
|---|---|---|---|---|
| IND 1 | 3.41 | 6.04 | | -6.3 |
| IND2 | 3.2 | 6.23 | | 1.2 |
| IND3 | 6.95 | -9.11 | | 6.3 |
| IND4 | -6 | -8.78 | | 2.34 |
| IND5 | 7.3 | 0.51 | | 2.32 |
| IND6 | 8.2 | 1.34 | | -4.21 |
| IND7 | -1.12 | 7.1 | | -2.06 |
| IND8 | 6.99 | 5.34 | | 9.21 |
| IND9 | -7.45 | 5.61 | | -4.56 |
| IND10 | 0.65 | 0.45 | | 9.48 |

Fig. 1: An example of generating CI-based vector in a population with 10 individuals and $D$ dimension.

it is selected as one of three randomly selected vectors for mutation operation. According to the proposed method, at each iteration of DE algorithm, the CI-based vector is constructed by collaborative intelligence of population and replace the worst candidate solution if the trial vector has a better objective value.

In the PSO algorithm, since each individual updates its own movement, the only effect which they get from others is related to updating the position based on the best candidate solution in their neighborhood. According to PSO updating equations, each individual determines its next position and velocity based on the best candidate solution in its neighborhood, so if it is desired to have more influence of the new CI-based vector on others, it is required to use a more contributed topology scheme to define neighborhood concept (like ring topology). For applying our method on PSO algorithm, at the end of each iteration, a new CI-based solution is generated based on the proposed scheme. Then, the new individual will be compared with the particle with worst personal best movement ($x_w$). If the CI-based vector has better objective value comparing to worst personal best movement, the trail vector enters to the population and replaces $x_w$. The components of new particle (position and velocity) should be updated accordingly. CI-based particle gets the position based on voted variables (according to proposed scheme) and its personal best movement is initialized with its current position. But its velocity should

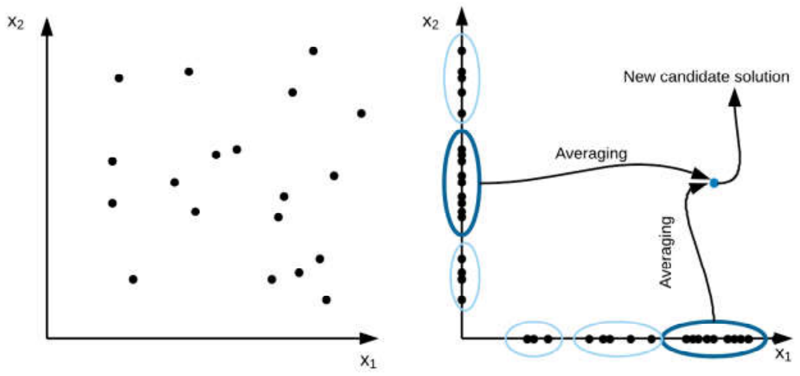


Fig. 2: Illustrating example of generating of CI-based vector in 2-dimensional space based on proposed scheme

be calculated at the next iteration according to the Eq. 2. However as a new member, its velocity be affected by only the global best. All steps of the proposed algorithm are provided in Algorithm 2.

## IV. Experiments results

In order to evaluate the performance of the proposed scheme, a series of experiments are conducted on CEC-2017 benchmark functions [24]. This test suite offers 30 scalable minimization problems, which can be categorized into: unimodal functions (F1, F2, F3), simple multi-modal functions (F4, F5,..., F10), hybrid functions (F11, F12,..., F20), and composition functions (F21, F22,..., F30) [25]. The details of benchmark functions are provided in Table I.

As mentioned in the proposed method section, two meta-heuristic algorithms, DE and PSO are modified based on collective intelligence scheme, namely, CIDE and CIPSO. The proposed algorithms are compared with their parents (DE and PSO) on different dimensions which are valid for this benchmark set, D=10, 30, 50, and 100. The same parameters are considered for all algorithms. The population size and number of function evaluations are set to 100 and $30 \times D$, respectively. The crossover rate, $CR$, and scaling factor, $F$, for DE and CIDV algorithms set to 0.9 and 0.5. Optimal values of constriction coefficients and inertia weights for PSO proposed in [26] are 0.7298 and $C_0$=$C_1$=1.49618. The number of clusters ($K$) in proposed scheme would be 10. Each algorithm run 30 times. Then, the mean, median, best, worst and standard deviation of fitness function values for each algorithm are reported. To investigate the meaningful difference between algorithms, a Wilcoxon statistical test [27] with a confidence interval of 95% is performed on mean value of fitness functions. Symbols '†' and '$\approx$' denote the CI version algorithms are better than and similar to compared algorithms, respectively. "w/t/l" indicates the number of "wins", "ties", and "loses" of proposed algorithm comparing to their parent version.

Tables II and III represent the resulted fitness values using DE algorithm and its CI-based version. Mean, median, best, worst, and standard deviation of the results of 30 algorithm's runs are reported. All functions are solved in dimensions 10, 30, 50, and 100. The results confirm the significance improvement of proposed algorithm comparing to parent DE. The last row of the table indicates the result of statistical results. As it is presented, on dimension 10, CIDE could outperform DE on 18 functions out of 30 while it loses on only 3 functions and it has same results with CIDE on 8 out of 30 functions. On other dimensions, the results are almost as dimension 10. The difference among the results of dimension 10 and two higher dimensions (30 and 50) is on the number of ties and loses. Regarding dimension 100, the results of CIDE deteriorates slightly rather than lower dimensions while it is still has significant better results comparing to DE. According to obtained results, it is clear that CIDE performed almost same as DE for the unimodal problems (F1-F3) while for the multi-modal problems (F4-F10), it was able to obtain a solution with less distance to optimal solution for most of the functions (F5, F7, F8, F10) regard to the mean solutions comparing to DE. It means that the effect of injecting CI-based vector to population in such problems is more than unimodal ones. In the case of hybrid and composite functions, the CIDE obtained the closer solution to the optimal for some functions (F11, F17, F23, F24, F29) on all experimental dimensions. However, for other functions, CIDE performs differently on various dimensions.

The computational results for PSO and CIPSO are demonstrated in Tables IV and V. Similar to CIDE, CIPSO also outperforms its parent algorithm, PSO on most of the experimental functions. However, comparing to CIDE, CIPSO wins on less functions, it is still better on most of the functions compared to PSO. The important point about this comparison is that the improvement on results of CIPSO increases when dimension increases. So that on dimension 10, CIPSO has better results on 12 functions out of 30, while on dimension 18, the number of wins is 18. CIPSO ties with PSO on highest experimental dimension, 100, on 8 functions out of 30 while the number of loses is only 3. Regarding to type of functions,

```
input : $NP$: Population size, $ITmax$: Maximum
        number of iterations, $D$: Number of
        dimensions of the problem $K$: Number of
        clusters
output: Best solution
// Initialization
Generate $NP$ uniform random individuals to construct
  $POP_0$;
$t = 1$;
Evaluate each individual in terms of fitness function;
while $t < ITmax$ do
    // Run optimization algorithm on
       $POP_{t-1}$
    $POP_t \leftarrow$ Optimize$(POP_{t-1})$;
    for $d \leftarrow 1$ to $D$ do // For each dimension
        // Performing the K-means
           clustering
        $C =$ K-means$(POP_t(:, d))$;// $C =$
           $\{c^{(1)}, c^{(2)}, c^{(3)}, ..., c^{(K)}\}$
        // Calculating the size of each
           cluster
        $NC = \{|c^{(1)}|, |c^{(2)}|, |c^{(3)}|, ..., |c^{(K)}|\}$;
        $MC =$ max$(NC)$;
        // Averaging on variables in
           most crowded cluster
        $CIV(d) =$ Average$(POP_t(c^{MC}, d))$;
    end
    $F_p =$ Evaluate$(CIV)$;
    if DE is applied then
        Select candidate solution with worst fitness
          value ($X_w$);
        if $F_p <$ Evaluate$(X_w)$ then
            $POP_t(w) = CIV$
        end
    end
    if PSO is applied then
        Select candidate solution with worst personal
          best fitness value ($X_{pw}$);
        if $F_p <$ Evaluate$(X_{pw})$ then
            $POP(pw).position = CIV$;
            $POP(pw).pbest = CIV$;
        end
    end
    t=t+1;
end
```

**Algorithm 2:** Pseudo-code of proposed method; embedding proposed CI-based scheme in DE and PSO algorithms

TABLE I: Summary of the CEC 2017 test functions [24]. $N$ is the number of basic functions to construct the components of hybrid or composition functions.

| | No. | Functions | $F_i(x^*)$ |
|---|---|---|---|
| Unimodal Functions | 1 | Shifted and Rotated Bent Cigar Function | 100 |
| | 2 | Shifted and Rotated Sum of Different Power Function* | 200 |
| | 3 | Shifted and Rotated Zakharov Function | 300 |
| Simple Multimodal Functions | 4 | Shifted and Rotated Rosenbrock's Function | 400 |
| | 5 | Shifted and Rotated Rastrigin's Function | 500 |
| | 6 | Shifted and Rotated Expanded Scaffer's F6 Function | 600 |
| | 7 | Shifted and Rotated Lunacek Bi_Rastrigin Function | 700 |
| | 8 | Shifted and Rotated Non-Continuous Rastrigin's Function | 800 |
| | 9 | Shifted and Rotated Levy Function | 900 |
| | 10 | Shifted and Rotated Schwefel's Function | 1000 |
| Hybrid Functions | 11 | Hybrid Function 1 (N=3) | 1100 |
| | 12 | Hybrid Function 2 (N=3) | 1200 |
| | 13 | Hybrid Function 3 (N=3) | 1300 |
| | 14 | Hybrid Function 4 (N=4) | 1400 |
| | 15 | Hybrid Function 5 (N=4) | 1500 |
| | 16 | Hybrid Function 6 (N=4) | 1600 |
| | 17 | Hybrid Function 6 (N=5) | 1700 |
| | 18 | Hybrid Function 6 (N=5) | 1800 |
| | 19 | Hybrid Function 6 (N=5) | 1900 |
| | 20 | Hybrid Function 6 (N=6) | 2000 |
| Composition Functions | 21 | Composition Function 1 (N=3) | 2100 |
| | 22 | Composition Function 2 (N=3) | 2200 |
| | 23 | Composition Function 3 (N=4) | 2300 |
| | 24 | Composition Function 4 (N=4) | 2400 |
| | 25 | Composition Function 5 (N=5) | 2500 |
| | 26 | Composition Function 6 (N=5) | 2600 |
| | 27 | Composition Function 7 (N=6) | 2700 |
| | 28 | Composition Function 8 (N=6) | 2800 |
| | 29 | Composition Function 9 (N=3) | 2900 |
| | 30 | Composition Function 10 (N=3) | 3000 |
| Search Range: $[-100, 100]^D$ | | | |

CIPSO is able to reach near to optimal solutions in unimodal and multi-modal functions on all dimensions. However, in hybrid and compositions functions, the difference of CIPSO performance is more explicit on higher dimensions.

For better illustration of performance evaluation, the convergence plots of proposed methods and parent algorithms are displayed in Fig. 3 and Fig. 4 on some selected functions. As it is presented, after the first few iterations, the CIDE and CIPSO algorithms surpass DE and PSO algorithms in terms of minimizing the fitness function. The injection CI-based vector to population improves the performance of the algorithm; so that the difference on some functions such as F12 to decreasing the error is very remarkable.

Table VI shows some impressive results to clarify how collective intelligence can generate a beneficial candidate solution. The table gives a comparison on quality of CI-based individual with best and worst individuals in the population. Each number indicates the percentage of the iterations that generated individual would be better than best or worst candidate solution in the population. The other column in this table is related to average rank of created individual in whole population. Comparing the quality of generated solution with best solution indicates that in some iterations of the algorithm, the new member will be injected to the population as the best solution (e.g., 8.3% in CIPSO on dimension 100). As it is presented, by increasing the dimension of the problem, the goodness of created individual comparing to worst solution increases. For example, in dimension 100, the result of voting is better than worst individual in 68% of iterations. These

TABLE II: Comparison of DE and CIDE on functions F1-F20.

| | | D=10 | | D=30 | | D=50 | | D=100 | |
|---|---|---|---|---|---|---|---|---|---|
| Function | | DE | CIDE | DE | CIDE | DE | CIDE | DE | CIDE |
| | Mean | 1.27e+05 | 7.94e+03$^{\dagger}$ | 3.03e+04 | 8.53e+03$^{\dagger}$ | 7.97e+03$^{\approx}$ | 7.10e+03$^{\approx}$ | 8.49e+03$^{\approx}$ | 9.59e+03$^{\approx}$ |
| | Best | 1.14e+04 | 4.08e+02 | 2.11e+03 | 1.08e+02 | 2.23e+02 | 1.23e+02 | 1.24e+02 | 1.12e+02 |
| F1 | Worst | 8.02e+05 | 1.26e+04 | 2.48e+05 | 2.06e+04 | 2.48e+04 | 3.12e+04 | 3.04e+04 | 4.40e+04 |
| | Median | 9.34e+04 | 8.40e+03 | 1.44e+04 | 5.32e+03 | 5.79e+03 | 4.57e+03 | 5.20e+03 | 4.21e+03 |
| | Std | 1.43e+05 | 3.95e+03 | 4.73e+04 | 7.77e+03 | 7.41e+03 | 8.64e+03 | 9.02e+03 | 1.23e+04 |
| | Mean | 2.00e+02$^{\approx}$ | 2.00e+02$^{\approx}$ | 1.99e+23$^{\dagger}$ | 2.02e+24 | 11.342e+76 | 9.47e+54$^{\dagger}$ | 2.09e+93$^{\dagger}$ | 1.02e+133 |
| | Best | 2.00e+02 | 2.00e+02 | 1.99e+23 | 4.52e+14 | 4.231e+34 | 9.70e+25 | 2.09e+93 | 8.75e+68 |
| F2 | Worst | 2.00e+02 | 2.00e+02 | 1.99e+23 | 2.51e+25 | 3.943e+80 | 1.28e+56 | 2.09e+93 | 3.06e+134 |
| | Median | 2.00e+02 | 2.00e+02 | 1.99e+23 | 2.33e+23 | 3.548e+ 64 | 3.43e+51 | 2.09e+93 | 3.51e+92 |
| | Std | 0.00e+00 | 0.00e+00 | 0.00e+00 | 4.96e+24 | 3.294e+60 | 3.06e+55 | 0.00e+00 | 5.58e+133 |
| | Mean | 1.58e+03 | 1.32e+03$^{\dagger}$ | 1.12e+05 | 1.05e+05$^{\dagger}$ | 2.57e+05$^{\approx}$ | 2.56e+05$^{\approx}$ | 6.78e+05$^{\approx}$ | 6.83e+05$^{\approx}$ |
| | Best | 8.18e+02 | 4.81e+02 | 7.91e+04 | 6.44e+04 | 1.98e+05 | 1.92e+05 | 5.63e+05 | 5.46e+05 |
| F3 | Worst | 2.82e+03 | 2.68e+03 | 1.48e+05 | 1.68e+05 | 3.02e+05 | 2.96e+05 | 7.81e+05 | 8.23e+05 |
| | Median | 1.53e+03 | 1.24e+03 | 1.19e+05 | 1.05e+05 | 2.62e+05 | 2.63e+05 | 6.72e+05 | 6.99e+05 |
| | Std | 5.12e+02 | 5.42e+02 | 1.74e+04 | 2.34e+04 | 2.55e+04 | 2.61e+04 | 5.31e+04 | 6.50e+04 |
| | Mean | 4.01e+02$^{\dagger}$ | 4.01e+02 | 4.86e+02$^{\approx}$ | 4.87e+02$^{\approx}$ | 5.37e+02$^{\approx}$ | 5.51e+02$^{\approx}$ | 6.28e+02$^{\approx}$ | 6.36e+02$^{\approx}$ |
| | Best | 4.00e+02 | 4.01e+02 | 4.85e+02 | 4.85e+02 | 5.01e+02 | 4.29e+02 | 5.99e+02 | 5.91e+02 |
| F4 | Worst | 4.02e+02 | 4.02e+02 | 4.89e+02 | 4.99e+02 | 6.06e+02 | 6.03e+02 | 6.91e+02 | 7.09e+02 |
| | Median | 4.01e+02 | 4.01e+02 | 4.86e+02 | 4.86e+02 | 5.20e+02 | 5.52e+02 | 6.23e+02 | 6.30e+02 |
| | Std | 4.26e-01 | 5.22e-01 | 7.00e-01 | 2.44e+00 | 3.65e+01 | 4.14e+01 | 2.54e+01 | 2.89e+01 |
| | Mean | 5.33e+02 | 5.13e+02$^{\dagger}$ | 7.00e+02 | 5.52e+02$^{\dagger}$ | 8.93e+02 | 5.82e+02$^{\dagger}$ | 1.38e+03 | 7.20e+02$^{\dagger}$ |
| | Best | 5.20e+02 | 5.03e+02 | 6.63e+02 | 5.21e+02 | 8.69e+02 | 5.42e+02 | 1.33e+03 | 6.37e+02 |
| F5 | Worst | 5.43e+02 | 5.36e+02 | 7.21e+02 | 6.79e+02 | 9.22e+02 | 6.25e+02 | 1.42e+03 | 1.03e+03 |
| | Median | 5.33e+02 | 5.10e+02 | 7.03e+02 | 5.40e+02 | 8.91e+02 | 5.81e+02 | 1.38e+03 | 7.09e+02 |
| | Std | 4.49e+00 | 9.38e+00 | 1.36e+01 | 3.83e+01 | 1.42e+01 | 1.89e+01 | 2.26e+01 | 7.20e+01 |
| | Mean | 6.00e+02 | 6.00e+02$^{\dagger}$ | 6.00e+02 | 6.00e+02$^{\dagger}$ | 6.00e+02 | 6.00e+02$^{\dagger}$ | 6.00e+02$^{\dagger}$ | 6.01e+02 |
| | Best | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 |
| F6 | Worst | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.03e+02 |
| | Median | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.00e+02 | 6.01e+02 |
| | Std | 1.40e-03 | 8.21e-06 | 3.90e-02 | 7.69e-03 | 3.51e-02 | 3.53e-02 | 1.06e-02 | 7.72e-01 |
| | Mean | 7.45e+02 | 7.16e+02$^{\dagger}$ | 9.37e+02 | 7.70e+02$^{\dagger}$ | 1.15e+03 | 8.33e+02$^{\dagger}$ | 1.70e+03 | 1.05e+03$^{\dagger}$ |
| | Best | 7.36e+02 | 7.11e+02 | 9.14e+02 | 7.50e+02 | 1.09e+03 | 8.06e+02 | 1.66e+03 | 9.82e+02 |
| F7 | Worst | 7.52e+02 | 7.29e+02 | 9.54e+02 | 9.16e+02 | 1.18e+03 | 8.65e+02 | 1.77e+03 | 1.15e+03 |
| | Median | 7.45e+02 | 7.15e+02 | 9.39e+02 | 7.62e+02 | 1.15e+03 | 8.27e+02 | 1.70e+03 | 1.04e+03 |
| | Std | 4.69e+00 | 5.11e+00 | 1.16e+01 | 2.94e+01 | 2.14e+01 | 1.71e+01 | 2.75e+01 | 5.07e+01 |
| | Mean | 8.35e+02 | 8.08e+02$^{\dagger}$ | 1.00e+03 | 8.47e+02$^{\dagger}$ | 1.19e+03 | 8.86e+02$^{\dagger}$ | 1.68e+03 | 1.01e+03$^{\dagger}$ |
| | Best | 8.25e+02 | 8.00e+02 | 9.73e+02 | 8.21e+02 | 1.14e+03 | 8.41e+02 | 1.64e+03 | 9.30e+02 |
| F8 | Worst | 8.43e+02 | 8.28e+02 | 1.03e+03 | 8.71e+02 | 1.23e+03 | 9.54e+02 | 1.71e+03 | 1.15e+03 |
| | Median | 8.36e+02 | 8.06e+02 | 1.01e+03 | 8.47e+02 | 1.19e+03 | 8.84e+02 | 1.68e+03 | 9.97e+02 |
| | Std | 4.69e+00 | 7.33e+00 | 1.25e+01 | 1.45e+01 | 1.94e+01 | 2.44e+01 | 1.92e+01 | 4.85e+01 |
| | Mean | 9.00e+02 | 9.00e+02$^{\dagger}$ | 9.00e+02 | 9.00e+02$^{\dagger}$ | 9.00e+02$^{\dagger}$ | 9.04e+02 | 9.02e+02$^{\dagger}$ | 1.03e+03 |
| | Best | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.24e+02 |
| F9 | Worst | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.01e+02 | 9.17e+02 | 9.15e+02 | 1.36e+03 |
| | Median | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.00e+02 | 9.03e+02 | 9.01e+02 | 1.00e+03 |
| | Std | 1.61e-06 | 8.78e-10 | 1.12e-01 | 1.72e-01 | 1.50e-01 | 4.05e+00 | 2.95e+00 | 1.03e+02 |
| | Mean | 1.83e+03 | 1.50e+03$^{\dagger}$ | 8.10e+03 | 4.97e+03$^{\dagger}$ | 1.46e+04 | 9.42e+03$^{\dagger}$ | 3.17e+04 | 1.87e+04$^{\dagger}$ |
| | Best | 1.37e+03 | 1.12e+03 | 7.08e+03 | 2.43e+03 | 1.38e+04 | 5.85e+03 | 3.00e+04 | 1.22e+04 |
| F10 | Worst | 2.43e+03 | 2.20e+03 | 8.78e+03 | 7.79e+03 | 1.51e+04 | 1.53e+04 | 3.24e+04 | 3.23e+04 |
| | Median | 1.83e+03 | 1.45e+03 | 8.20e+03 | 4.53e+03 | 1.47e+04 | 7.61e+03 | 3.18e+04 | 1.61e+04 |
| | Std | 2.74e+02 | 2.92e+02 | 4.36e+02 | 1.65e+03 | 3.28e+02 | 3.55e+03 | 4.45e+02 | 6.17e+03 |
| | Mean | 1.11e+03 | 1.10e+03$^{\dagger}$ | 1.21e+03 | 1.16e+03$^{\dagger}$ | 1.30e+03 | 1.16e+03$^{\dagger}$ | 4.11e+04 | 2.25e+04$^{\dagger}$ |
| | Best | 1.11e+03 | 1.10e+03 | 1.17e+03 | 1.11e+03 | 1.27e+03 | 1.14e+03 | 2.39e+04 | 7.37e+03 |
| F11 | Worst | 1.11e+03 | 1.11e+03 | 1.26e+03 | 1.21e+03 | 1.35e+03 | 1.22e+03 | 5.75e+04 | 3.93e+04 |
| | Median | 1.11e+03 | 1.10e+03 | 1.21e+03 | 1.17e+03 | 1.30e+03 | 1.16e+03 | 4.20e+04 | 1.89e+04 |
| | Std | 1.55e+00 | 2.15e+00 | 3.08e+01 | 3.06e+01 | 1.26e+01 | 1.86e+01 | 8.52e+03 | 9.20e+03 |
| | Mean | 2.88e+05$^{\approx}$ | 1.93e+05$^{\approx}$ | 2.91e+06 | 7.02e+05$^{\dagger}$ | 1.73e+07 | 1.75e+06$^{\dagger}$ | 1.24e+07 | 2.72e+06$^{\dagger}$ |
| | Best | 4.40e+04 | 4.83e+03 | 8.13e+05 | 7.19e+04 | 5.64e+06 | 2.71e+05 | 4.21e+06 | 9.33e+05 |
| F12 | Worst | 7.05e+05 | 5.48e+05 | 6.93e+06 | 3.79e+06 | 5.03e+07 | 4.32e+06 | 2.50e+07 | 8.02e+06 |
| | Median | 2.54e+05 | 1.37e+05 | 2.60e+06 | 2.46e+05 | 1.56e+07 | 1.58e+06 | 1.06e+07 | 2.55e+06 |
| | Std | 1.79e+05 | 1.54e+05 | 1.59e+06 | 1.03e+06 | 8.83e+06 | 1.17e+06 | 5.19e+06 | 1.49e+06 |
| | Mean | 1.54e+03 | 1.39e+03$^{\dagger}$ | 2.07e+04$^{\approx}$ | 2.58e+04$^{\approx}$ | 1.45e+04$^{\dagger}$ | 2.32e+04 | 7.98e+03$^{\approx}$ | 1.42e+04$^{\approx}$ |
| | Best | 1.35e+03 | 1.32e+03 | 1.47e+03 | 1.47e+03 | 1.78e+03 | 1.54e+03 | 1.81e+03 | 1.50e+03 |
| F13 | Worst | 1.99e+03 | 1.74e+03 | 6.14e+04 | 6.21e+04 | 3.78e+04 | 3.89e+04 | 1.93e+04 | 3.63e+04 |
| | Median | 1.48e+03 | 1.37e+03 | 1.10e+04 | 7.40e+03 | 1.30e+04 | 2.19e+04 | 7.19e+03 | 9.62e+03 |
| | Std | 1.58e+02 | 8.58e+01 | 2.05e+04 | 2.68e+04 | 1.08e+04 | 1.31e+04 | 5.39e+03 | 1.19e+04 |
| | Mean | 1.44e+03$^{\approx}$ | 1.43e+$^{\approx}$ | 4.67e+03$^{\approx}$ | 3.86e+03$^{\approx}$ | 5.92e+04$^{\approx}$ | 5.47e+04$^{\approx}$ | 2.37e+06$^{\approx}$ | 2.75e+06$^{\approx}$ |
| | Best | 1.43e+03 | 1.43e+03 | 2.04e+03 | 1.66e+03 | 1.65e+04 | 5.12e+03 | 8.55e+05 | 6.45e+05 |
| F14 | Worst | 1.45e+03 | 1.45e+03 | 9.25e+03 | 1.75e+04 | 1.43e+05 | 1.59e+05 | 6.77e+06 | 5.65e+06 |
| | Median | 1.44e+03 | 1.43e+03 | 4.08e+03 | 3.00e+03 | 4.99e+04 | 4.70e+04 | 1.87e+06 | 2.52e+06 |
| | Std | 4.78e+00 | 4.54e+00 | 1.79e+03 | 2.84e+03 | 3.59e+04 | 3.07e+04 | 1.37e+06 | 1.46e+06 |
| | Mean | 1.58e+03 | 1.55e+03$^{\dagger}$ | 8.75e+03$^{\approx}$ | 1.17e+04$^{\approx}$ | 1.63e+04$^{\dagger}$ | 2.21e+04 | 1.36e+04$^{\approx}$ | 1.66e+04$^{\approx}$ |
| | Best | 1.52e+03 | 1.52e+03 | 1.69e+03 | 1.52e+03 | 2.26e+03 | 1.93e+03 | 1.89e+03 | 1.71e+03 |
| F15 | Worst | 1.65e+03 | 1.58e+03 | 2.52e+04 | 2.87e+04 | 3.29e+04 | 3.28e+04 | 2.78e+04 | 2.89e+04 |
| | Median | 1.58e+03 | 1.55e+03 | 4.22e+03 | 1.00e+04 | 1.72e+04 | 2.58e+04 | 1.35e+04 | 1.07e+04 |
| | Std | 3.00e+01 | 2.02e+01 | 8.29e+03 | 8.63e+03 | 9.96e+03 | 1.01e+04 | 9.50e+03 | 1.17e+04 |
| | Mean | 1.60e+03$^{\approx}$ | 1.61e+03$^{\approx}$ | 3.08e+03 | 2.23e+03$^{\dagger}$ | 4.82e+03 | 3.22e+03$^{\dagger}$ | 9.85e+03 | 6.01e+03$^{\dagger}$ |
| | Best | 1.60e+03 | 1.60e+03 | 2.86e+03 | 1.63e+03 | 4.45e+03 | 1.79e+03 | 9.32e+03 | 3.83e+03 |
| F16 | Worst | 1.62e+03 | 1.64e+03 | 3.27e+03 | 3.04e+03 | 5.13e+03 | 5.13e+03 | 1.05e+04 | 9.29e+03 |
| | Median | 1.60e+03 | 1.60e+03 | 3.10e+03 | 2.15e+03 | 4.82e+03 | 3.05e+03 | 9.87e+03 | 5.99e+03 |
| | Std | 4.58e+00 | 8.39e+00 | 1.20e+02 | 3.21e+02 | 1.80e+02 | 7.58e+02 | 2.96e+02 | 1.07e+03 |
| | Mean | 1.74e+03 | 1.73e+03$^{\dagger}$ | 2.19e+03 | 1.91e+03$^{\dagger}$ | 3.65e+03 | 2.70e+03$^{\dagger}$ | 7.02e+03 | 4.37e+03$^{\dagger}$ |
| | Best | 1.73e+03 | 1.70e+03 | 1.81e+03 | 1.76e+03 | 3.32e+03 | 2.00e+03 | 6.57e+03 | 2.76e+03 |
| F17 | Worst | 1.75e+03 | 1.75e+03 | 2.51e+03 | 2.20e+03 | 3.99e+03 | 3.79e+03 | 7.30e+03 | 5.70e+03 |
| | Median | 1.73e+03 | 1.73e+03 | 2.25e+03 | 1.87e+03 | 3.63e+03 | 2.62e+03 | 7.05e+03 | 4.31e+03 |
| | Std | 4.62e+00 | 1.18e+01 | 2.09e+02 | 1.21e+02 | 1.73e+02 | 4.62e+02 | 1.82e+02 | 7.14e+02 |
| | Mean | 6.58e+03 | 3.51e+03$^{\dagger}$ | 9.32e+05$^{\approx}$ | 1.04e+06$^{\approx}$ | 2.92e+06$^{\approx}$ | 2.87e+06$^{\approx}$ | 1.70e+07$^{\approx}$ | 1.60e+07$^{\approx}$ |
| | Best | 2.52e+03 | 1.90e+03 | 3.16e+05 | 2.58e+05 | 1.23e+06 | 4.96e+05 | 7.95e+06 | 2.51e+06 |
| F18 | Worst | 1.49e+04 | 9.80e+03 | 2.12e+06 | 2.25e+06 | 5.63e+06 | 5.86e+06 | 2.63e+07 | 2.65e+07 |
| | Median | 6.17e+03 | 2.76e+03 | 8.64e+05 | 9.82e+05 | 2.73e+06 | 2.72e+06 | 1.64e+07 | 1.65e+07 |
| | Std | 2.85e+03 | 2.04e+03 | 4.44e+05 | 4.38e+05 | 1.10e+06 | 1.45e+06 | 4.57e+06 | 5.32e+06 |
| | Mean | 1.94e+03 | 1.92e+03$^{\dagger}$ | 1.52e+04$^{\dagger}$ | 2.70e+04 | 5.47e03+$^{\dagger}$ | 1.35e+04 | 1.71e+04$^{\approx}$ | 2.09e+04$^{\approx}$ |
| | Best | 1.91e+03 | 1.91e+03 | 1.99e+03 | 1.92e+03 | 1.99e+03 | 2.41e+03 | 2.08e+03 | 2.03e+03 |
| F19 | Worst | 2.07e+03 | 1.94e+03 | 5.20e+04 | 5.41e+04 | 2.68e+04 | 4.23e+04 | 3.91e+04 | 4.13e+04 |
| | Median | 1.94e+03 | 1.92e+03 | 9.58e+03 | 2.46e+04 | 3.48e+03 | 4.26e+03 | 1.26e+04 | 1.41e+04 |
| | Std | 2.91e+01 | 7.51e+00 | 1.55e+04 | 2.01e+04 | 5.32e+03 | 1.44e+04 | 1.25e+04 | 1.57e+04 |
| | Mean | 2.02e+03 | 2.00e+034$^{\dagger}$ | 2.27e+03 | 2.18e+03$^{\dagger}$ | 3.81e+03 | 2.98e+03$^{\dagger}$ | 7.11e+03 | 5.12e+03$^{\dagger}$ |
| | Best | 2.00e+03 | 2.00e+03 | 2.06e+03 | 2.03e+03 | 3.22e+03 | 2.28e+03 | 6.63e+03 | 3.48e+03 |
| F20 | Worst | 2.03e+03 | 2.03e+03 | 2.60e+03 | 2.47e+03 | 4.03e+03 | 3.93e+03 | 7.62e+03 | 7.44e+03 |
| | Median | 2.02e+03 | 2.00e+03 | 2.21e+03 | 2.16e+03 | 3.85e+03 | 2.78e+03 | 7.10e+03 | 4.63e+03 |
| | Std | 6.56e+00 | 7.73e+00 | 1.85e+02 | 1.38e+02 | 1.68e+02 | 5.38e+02 | 2.19e+02 | 1.30e+03 |

TABLE III: Comparison of DE and CIDE on functions F21-F30.

| | | D=10 | | D=30 | | D=50 | | D=100 | |
|---|---|---|---|---|---|---|---|---|---|
| Function | | DE | CIDE | DE | CIDE | DE | CIDE | DE | CIDE |
| | Mean | 2.29e+03$^{\approx}$ | 2.30e+03$^{\approx}$ | 2.50e+03 | 2.36e+03$^{\dagger}$ | 2.69e+03 | 2.40e+03$^{\dagger}$ | 3.21e+03 | 2.53e+03$^{\dagger}$ |
| | Best | 2.20e+03 | 2.20e+03 | 2.46e+03 | 2.33e+03 | 2.66e+03 | 2.35e+03 | 3.17e+03 | 2.47e+03 |
| F21 | Worst | 2.34e+03 | 2.33e+03 | 2.52e+03 | 2.50e+03 | 2.72e+03 | 2.65e+03 | 3.24e+03 | 2.62e+03 |
| | Median | 2.33e+03 | 2.31e+03 | 2.50e+03 | 2.35e+03 | 2.69e+03 | 2.39e+03 | 3.22e+03 | 2.54e+03 |
| | Std | 6.57e+01 | 4.90e+01 | 1.47e+01 | 4.32e+01 | 1.35e+01 | 5.27e+01 | 1.92e+01 | 3.63e+01 |
| | Mean | 2.30e+03$^{\dagger}$ | 2.30e+03 | 4.68e+03$^{\approx}$ | 4.53e+03$^{\approx}$ | 1.59e+04 | 1.13e+04$^{\dagger}$ | 3.34e+04 | 1.98e+04$^{\dagger}$ |
| | Best | 2.20e+03 | 2.30e+03 | 2.30e+03 | 2.30e+03 | 1.54e+04 | 6.68e+03 | 3.26e+04 | 1.47e+04 |
| F22 | Worst | 2.30e+03 | 2.30e+03 | 9.70e+03 | 9.69e+03 | 1.66e+04 | 1.61e+04 | 3.41e+04 | 3.32e+04 |
| | Median | 2.30e+03 | 2.30e+03 | 2.30e+03 | 2.30e+03 | 1.59e+04 | 9.95e+03 | 3.33e+04 | 1.78e+04 |
| | Std | 2.63e+01 | 2.53e-01 | 3.43e+03 | 2.82e+03 | 3.46e+02 | 3.43e+03 | 4.06e+02 | 5.60e+03 |
| | Mean | 2.63e+03 | 2.62e+03$^{\dagger}$ | 2.85e+03 | 2.72e+03$^{\dagger}$ | 3.12e+03 | 2.84e+03$^{\dagger}$ | 3.70e+03 | 3.09e+03$^{\dagger}$ |
| | Best | 2.62e+03 | 2.61e+03 | 2.82e+03 | 2.67e+03 | 3.07e+03 | 2.77e+03 | 3.66e+03 | 3.00e+03 |
| F23 | Worst | 2.64e+03 | 2.64e+03 | 2.87e+03 | 2.85e+03 | 3.15e+03 | 2.91e+03 | 3.77e+03 | 3.26e+03 |
| | Median | 2.63e+03 | 2.62e+03 | 2.85e+03 | 2.71e+03 | 3.12e+03 | 2.83e+03 | 3.69e+03 | 3.08e+03 |
| | Std | 5.09e+00 | 8.91e+00 | 1.18e+01 | 4.05e+01 | 1.64e+01 | 3.36e+01 | 2.98e+01 | 6.05e+01 |
| | Mean | 2.75e+03 | 2.74e+03$^{\dagger}$ | 3.01e+03 | 2.93e+03$^{\dagger}$ | 3.28e+03 | 3.14e+03$^{\dagger}$ | 4.14e+03 | 3.58e+03$^{\dagger}$ |
| | Best | 2.50e+03 | 2.50e+03 | 2.99e+03 | 2.87e+03 | 3.26e+03 | 3.00e+03 | 4.05e+03 | 3.45e+03 |
| F24 | Worst | 2.77e+03 | 2.77e+03 | 3.03e+03 | 3.02e+03 | 3.31e+03 | 3.29e+03 | 4.20e+03 | 3.70e+03 |
| | Median | 2.76e+03 | 2.76e+03 | 3.01e+03 | 2.89e+03 | 3.28e+03 | 3.08e+03 | 4.15e+03 | 3.58e+03 |
| | Std | 4.81e+01 | 6.55e+01 | 1.05e+01 | 5.77e+01 | 1.23e+01 | 1.24e+02 | 3.50e+01 | 6.20e+01 |
| | Mean | 2.91e+03$^{\approx}$ | 2.91e+03$^{\approx}$ | 2.89e+03$^{\approx}$ | 2.89e+03$^{\approx}$ | 2.99e+03$^{\approx}$ | 3.00e+03$^{\approx}$ | 3.28e+03$^{\approx}$ | 3.26e+03$^{\approx}$ |
| | Best | 2.90e+03 | 2.90e+03 | 2.89e+03 | 2.89e+03 | 2.98e+03 | 2.96e+03 | 3.18e+03 | 3.11e+03 |
| F25 | Worst | 2.95e+03 | 2.95e+03 | 2.89e+03 | 2.89e+03 | 3.06e+03 | 3.07e+03 | 3.39e+03 | 3.35e+03 |
| | Median | 2.90e+03 | 2.90e+03 | 2.89e+03 | 2.89e+03 | 2.98e+03 | 2.98e+03 | 3.28e+03 | 3.27e+03 |
| | Std | 1.77e+01 | 2.18e+01 | 6.58e-02 | 1.20e-01 | 2.79e+01 | 3.42e+01 | 5.02e+01 | 5.37e+01 |
| | Mean | 2.90e+03$^{\dagger}$ | 2.90e+03 | 5.55e+03 | 4.28e+03$^{\dagger}$ | 7.58e+03 | 5.01e+03$^{\dagger}$ | 1.45e+04 | 8.92e+03$^{\dagger}$ |
| | Best | 2.90e+03 | 2.90e+03 | 5.33e+03 | 3.76e+03 | 7.28e+03 | 4.37e+03 | 1.38e+04 | 7.74e+03 |
| F26 | Worst | 2.90e+03 | 2.95e+03 | 5.76e+03 | 5.27e+03 | 7.82e+03 | 6.07e+03 | 1.51e+04 | 1.08e+04 |
| | Median | 2.90e+03 | 2.90e+03 | 5.57e+03 | 4.24e+03 | 7.58e+03 | 4.97e+03 | 1.46e+04 | 8.84e+03 |
| | Std | 1.04e-04 | 8.61e+00 | 1.00e+02 | 3.25e+02 | 1.43e+02 | 3.82e+02 | 3.36e+02 | 6.33e+02 |
| | Mean | 3.09e+03$^{\approx}$ | 3.09e+03$^{\approx}$ | 3.20e+03$^{\dagger}$ | 3.21e+03 | 3.24e+03$^{\dagger}$ | 3.35e+03 | 3.35e+03$^{\dagger}$ | 3.47e+03 |
| | Best | 3.09e+03 | 3.09e+03 | 3.18e+03 | 3.19e+03 | 3.20e+03 | 3.26e+03 | 3.31e+03 | 3.39e+03 |
| F27 | Worst | 3.09e+03 | 3.09e+03 | 3.22e+03 | 3.24e+03 | 3.32e+03 | 3.61e+03 | 3.42e+03 | 3.57e+03 |
| | Median | 3.09e+03 | 3.09e+03 | 3.20e+03 | 3.21e+03 | 3.23e+03 | 3.32e+03 | 3.35e+03 | 3.47e+03 |
| | Std | 2.00e-01 | 3.22e-01 | 9.56e+00 | 1.22e+01 | 3.32e+01 | 8.57e+01 | 3.12e+01 | 4.08e+01 |
| | Mean | 3.15e+03$^{\approx}$ | 3.15e+03$^{\approx}$ | 3.21e+03$^{\approx}$ | 3.21e+03$^{\approx}$ | 3.26e+03$^{\dagger}$ | 3.29e+03 | 3.39e+03$^{\dagger}$ | 9.82e+03 |
| | Best | 3.10e+03 | 3.10e+03 | 3.20e+03 | 3.11e+03 | 3.26e+03 | 3.26e+03 | 3.34e+03 | 3.34e+03 |
| F28 | Worst | 3.41e+03 | 3.41e+03 | 3.25e+03 | 3.27e+03 | 3.31e+03 | 3.37e+03 | 3.50e+03 | 1.62e+04 |
| | Median | 3.10e+03 | 3.10e+03 | 3.21e+03 | 3.21e+03 | 3.26e+03 | 3.31e+03 | 3.38e+03 | 1.13e+04 |
| | Std | 1.04e+02 | 1.03e+02 | 1.58e+01 | 3.46e+01 | 1.12e+01 | 2.77e+01 | 3.78e+01 | 6.09e+03 |
| | Mean | 3.18e+03 | 3.16e+03$^{\dagger}$ | 3.91e+03 | 3.52e+03$^{\dagger}$ | 5.05e+03 | 3.78e+03$^{\dagger}$ | 8.98e+03 | 5.97e+03$^{\dagger}$ |
| | Best | 3.16e+03 | 3.14e+03 | 3.44e+03 | 3.36e+03 | 4.39e+03 | 3.29e+03 | 7.85e+03 | 4.84e+03 |
| F29 | Worst | 3.21e+03 | 3.21e+03 | 4.28e+03 | 4.06e+03 | 5.66e+03 | 5.54e+03 | 9.57e+03 | 7.88e+03 |
| | Median | 3.18e+03 | 3.16e+03 | 3.97e+03 | 3.46e+03 | 5.06e+03 | 3.65e+03 | 8.96e+03 | 5.75e+03 |
| | Std | 1.27e+01 | 1.65e+01 | 2.44e+02 | 1.62e+02 | 2.28e+02 | 4.54e+02 | 3.81e+02 | 7.87e+02 |
| | Mean | 3.68e+04$^{\approx}$ | 6.94e+04$^{\approx}$ | 2.77e+04 | 1.09e+04$^{\dagger}$ | 3.47e+06 | 1.46e+06$^{\dagger}$ | 1.40e+04$^{\approx}$ | 1.40e+04$^{\approx}$ |
| | Best | 4.15e+03 | 4.71e+03 | 1.12e+04 | 5.87e+03 | 8.88e+05 | 6.12e+05 | 8.00e+03 | 7.22e+03 |
| F30 | Worst | 7.46e+05 | 8.21e+05 | 7.65e+04 | 1.62e+04 | 1.11e+07 | 3.57e+06 | 2.38e+04 | 3.97e+04 |
| | Median | 9.26e+03 | 8.27e+03 | 2.39e+04 | 1.04e+04 | 2.86e+06 | 1.33e+06 | 1.37e+04 | 1.03e+04 |
| | Std | 1.34e+05 | 2.05e+05 | 1.69e+04 | 3.33e+03 | 2.36e+06 | 6.13e+05 | 4.17e+03 | 8.33e+03 |
| w/t/l | | - | 18/9/3 | - | 19/8/3 | - | 18/6/6 | - | 15/10/5 |

values even are more significant regarding CIPSO algorithm. So that, in 79.12% of iterations for dimension 100, the CI-based individual is better than worst member of population. Considering the rank of voting individual, in CIPSO, the new member is better than more than half of the members in population in all dimensions. This clarifies how injection the new member push whole population to more promising regions in the search space.

## V. Conclusion remarks

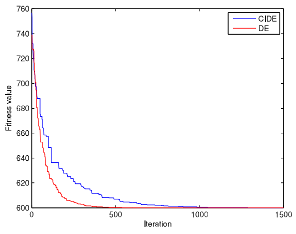
In population-based algorithms, individuals can collaborate implicitly or explicitly with each other to find more promising regions in the search space. Accordingly, this paper presents a collective intelligence method to generate a new trial vector which is expected to have a better quality. Each variable's value of CI-based individual is obtained using voting among a cluster of individuals. The average value of variable of members in the most crowded cluster is considered as value of trail's variable. This method leads to collect the best part of subset of population to generate new candidate solution. The experiments show that in the average, the created vector has better fitness value rather than 60% of individuals. Two population-based algorithms, DE and PSO are modified based on this scheme. The results of experiments confirm that injec-

TABLE IV: Comparison of PSO and CIPSO on functions F1-F20.

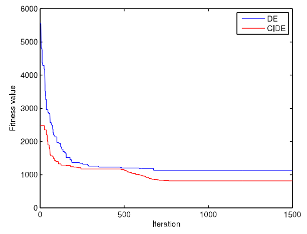
| | | D=10 | | D=30 | | D=50 | | D=100 | |
|---|---|---|---|---|---|---|---|---|---|
| Function | | PSO | CIPSO | PSO | CIPSO | PSO | CIPSO | PSO | CIPSO |
| | Mean | 1.65e+03$^{\approx}$ | 1.31e+03$^{\approx}$ | 4.54e+03$^{\approx}$ | 4.15e+03$^{\approx}$ | 1.54e+04 | 2.95e+03$^{\dagger}$ | 6.29e+04 | 1.55e+04$^{\dagger}$ |
| | Best | 1.26e+02 | 1.12e+02 | 2.95e+02 | 1.65e+02 | 3.56e+03 | 5.36e+02 | 3.00e+04 | 7.82e+03 |
| F1 | Worst | 1.09e+04 | 5.47e+03 | 1.18e+04 | 2.18e+04 | 5.16e+04 | 7.84e+03 | 1.38e+05 | 3.56e+04 |
| | Median | 9.30e+02 | 7.07e+02 | 4.30e+03 | 3.14e+03 | 1.25e+04 | 2.48e+03 | 5.70e+04 | 1.53e+04 |
| | Std | 2.31e+03 | 1.39e+03 | 3.17e+03 | 4.51e+03 | 1.08e+04 | 2.18e+03 | 2.60e+04 | 6.35e+03 |
| | Mean | 2.86e+03 | 8.62e+02$^{\dagger}$ | 4.23e+21 | 2.06e+17$^{\dagger}$ | 3.78e+50 | 1.48e+41$^{\dagger}$ | 1.14e+138 | 4.16e+113$^{\dagger}$ |
| | Best | 2.00e+02 | 2.00e+02 | 3.91e+15 | 4.00e+13 | 7.87e+37 | 2.55e+32 | 8.68e+103 | 1.74e+95 |
| F2 | Worst | 1.88e+04 | 2.39e+03 | 4.38e+22 | 2.04e+18 | 1.12e+52 | 4.11e+42 | 3.43e+139 | 1.14e+115 |
| | Median | 1.08e+03 | 5.15e+02 | 3.80e+20 | 1.78e+16 | 2.78e+45 | 2.16e+36 | 1.20e+120 | 1.53e+104 |
| | Std | 3.96e+03 | 7.27e+02 | 8.91e+21 | 4.70e+17 | 2.04e+51 | 7.50e+41 | 6.26e+138 | 2.08e+114 |
| | Mean | 5.63e+03 | 4.21e+03$^{\dagger}$ | 1.56e+05 | 1.26e+05$^{\dagger}$ | 3.64e+05 | 2.58e+05$^{\dagger}$ | 8.06e+05 | 5.94e+05$^{\dagger}$ |
| | Best | 1.83e+03 | 1.63e+03 | 6.31e+04 | 7.88e+04 | 2.52e+05 | 1.77e+05 | 6.35e+05 | 4.77e+05 |
| F3 | Worst | 1.15e+04 | 9.54e+03 | 2.08e+05 | 1.83e+05 | 4.49e+05 | 3.50e+05 | 1.01e+06 | 7.49e+05 |
| | Median | 4.97e+03 | 3.97e+03 | 1.65e+05 | 1.18e+05 | 3.63e+05 | 2.55e+05 | 8.08e+05 | 5.96e+05 |
| | Std | 2.71e+03 | 1.90e+03 | 3.45e+04 | 3.12e+04 | 4.97e+04 | 4.65e+04 | 8.93e+04 | 7.41e+04 |
| | Mean | 4.03e+02$^{\dagger}$ | 4.05e+02 | 4.42e+02$^{\dagger}$ | 4.72e+02 | 5.09e+02$^{\approx}$ | 5.21e+02$^{\approx}$ | 7.72e+02$^{\approx}$ | 7.88e+02$^{\approx}$ |
| | Best | 4.00e+02 | 4.00e+02 | 4.17e+02 | 4.24e+02 | 4.41e+02 | 4.39e+02 | 6.14e+02 | 7.09e+02 |
| F4 | Worst | 4.07e+02 | 4.07e+02 | 4.81e+02 | 4.97e+02 | 5.98e+02 | 6.03e+02 | 8.87e+02 | 8.72e+02 |
| | Median | 4.04e+02 | 4.06e+02 | 4.30e+02 | 4.77e+02 | 5.05e+02 | 5.23e+02 | 7.68e+02 | 7.83e+02 |
| | Std | 2.22e+00 | 1.95e+00 | 2.17e+01 | 1.91e+01 | 4.19e+01 | 3.78e+01 | 6.78e+01 | 4.41e+01 |
| | Mean | 5.12e+02 | 5.08e+02$^{\dagger}$ | 6.12e+02 | 5.94e+02 | 7.55e+02$^{\approx}$ | 7.37e+02$^{\approx}$ | 1.34e+03 | 1.23e+03$^{\dagger}$ |
| | Best | 5.06e+02 | 5.02e+02 | 5.65e+02 | 5.46e+02 | 6.97e+02 | 6.73e+02 | 1.15e+03 | 1.09e+03 |
| F5 | Worst | 5.20e+02 | 5.16e+02 | 6.50e+02 | 6.37e+02 | 8.05e+02 | 8.08e+02 | 1.63e+03 | 1.42e+03 |
| | Median | 5.12e+02 | 5.08e+02 | 6.19e+02 | 5.96e+02 | 7.57e+02 | 7.37e+02 | 1.32e+03 | 1.23e+03 |
| | Std | 3.52e+00 | 3.44e+00 | 2.27e+01 | 1.61e+01 | 3.12e+01 | 3.50e+01 | 1.00e+02 | 9.74e+01 |
| | Mean | 6.00e+02 | 6.00e+02$^{\dagger}$ | 6.06e+02$^{\approx}$ | 6.06e+02$^{\approx}$ | 6.28e+02$^{\approx}$ | 6.25e+02$^{\approx}$ | 6.65e+02 | 6.51e+02$^{\dagger}$ |
| | Best | 6.00e+02 | 6.00e+02 | 6.02e+02 | 6.01e+02 | 6.17e+02 | 6.10e+02 | 6.48e+02 | 6.43e+02 |
| F6 | Worst | 6.00e+02 | 6.00e+02 | 6.17e+02 | 6.13e+02 | 6.42e+02 | 6.37e+02 | 6.75e+02 | 6.59e+02 |
| | Median | 6.00e+02 | 6.00e+02 | 6.05e+02 | 6.06e+02 | 6.27e+02 | 6.25e+02 | 6.65e+02 | 6.50e+02 |
| | Std | 8.39e-02 | 6.92e-03 | 3.35e+00 | 3.18e+00 | 7.33e+00 | 6.99e+00 | 7.23e+00 | 3.63e+00 |
| | Mean | 7.24e+02 | 7.21e+02$^{\dagger}$ | 8.34e+02$^{\approx}$ | 8.33e+02$^{\approx}$ | 1.01e+03$^{\approx}$ | 1.02e+03$^{\approx}$ | 1.69e+03$^{\approx}$ | 1.74e+03$^{\approx}$ |
| | Best | 7.13e+02 | 7.14e+02 | 7.93e+02 | 7.95e+02 | 9.30e+02 | 9.42e+02 | 1.41e+03 | 1.55e+03 |
| F7 | Worst | 7.34e+02 | 7.28e+02 | 8.81e+02 | 8.90e+02 | 1.07e+03 | 1.09e+03 | 2.10e+03 | 1.92e+03 |
| | Median | 7.23e+02 | 7.20e+02 | 8.39e+02 | 8.29e+02 | 1.01e+03 | 1.02e+03 | 1.67e+03 | 1.72e+03 |
| | Std | 5.00e+00 | 3.65e+00 | 2.23e+01 | 2.07e+01 | 3.66e+01 | 3.72e+01 | 1.41e+02 | 1.12e+02 |
| | Mean | 8.12e+02 | 8.07e+02$^{\dagger}$ | 9.05e+02 | 8.75e+02$^{\dagger}$ | 1.06e+03 | 1.03e+03$^{\dagger}$ | 1.65e+03 | 1.56e+03$^{\dagger}$ |
| | Best | 8.06e+02 | 8.01e+02 | 8.57e+02 | 8.44e+02 | 9.96e+02 | 9.52e+02 | 1.45e+03 | 1.41e+03 |
| F8 | Worst | 8.17e+02 | 8.14e+02 | 9.44e+02 | 8.99e+02 | 1.12e+03 | 1.16e+03 | 1.85e+03 | 1.76e+03 |
| | Median | 8.12e+02 | 8.07e+02 | 9.06e+02 | 8.77e+02 | 1.06e+03 | 1.03e+03 | 1.65e+03 | 1.58e+03 |
| | Std | 2.95e+00 | 3.09e+00 | 1.92e+01 | 1.51e+01 | 3.03e+01 | 4.43e+01 | 8.42e+01 | 8.28e+01 |
| | Mean | 9.00e+02 | 9.00e+02$^{\dagger}$ | 1.73e+03 | 1.46e+03$^{\dagger}$ | 9.87e+03 | 6.84e+03$^{\dagger}$ | 6.55e+04 | 3.35e+04$^{\dagger}$ |
| | Best | 9.00e+02 | 9.00e+02 | 1.08e+03 | 9.97e+02 | 5.38e+03 | 3.44e+03 | 4.32e+04 | 1.83e+04 |
| F9 | Worst | 9.04e+02 | 9.00e+02 | 2.41e+03 | 2.94e+03 | 1.79e+04 | 1.25e+04 | 9.50e+04 | 5.11e+04 |
| | Median | 9.00e+02 | 9.00e+02 | 1.67e+03 | 1.34e+03 | 1.04e+04 | 6.70e+03 | 6.23e+04 | 3.46e+04 |
| | Std | 6.65e-01 | 1.79e-02 | 3.16e+02 | 4.32e+02 | 3.34e+03 | 2.22e+03 | 1.36e+04 | 7.53e+03 |
| | Mean | 1.58e+03$^{\approx}$ | 1.56e+03$^{\approx}$ | 5.12e+03$^{\approx}$ | 5.26e+03$^{\approx}$ | 9.04e+03$^{\approx}$ | 9.12e+03$^{\approx}$ | 2.17e+04$^{\approx}$ | 2.01e+04$^{\approx}$ |
| | Best | 1.16e+03 | 1.13e+03 | 3.88e+03 | 4.19e+03 | 7.24e+03 | 6.03e+03 | 1.85e+04 | 1.17e+04 |
| F10 | Worst | 1.90e+03 | 1.86e+03 | 6.35e+03 | 6.28e+03 | 1.04e+04 | 1.14e+04 | 2.52e+04 | 2.59e+04 |
| | Median | 1.58e+03 | 1.58e+03 | 5.09e+03 | 5.28e+03 | 9.01e+03 | 9.19e+03 | 2.16e+04 | 2.03e+04 |
| | Std | 1.56e+02 | 1.86e+02 | 6.45e+02 | 5.11e+02 | 8.05e+02 | 1.08e+03 | 1.64e+03 | 3.78e+03 |
| | Mean | 1.11e+03$^{\approx}$ | 1.11e+03$^{\approx}$ | 1.23e+03 | 1.20e+03$^{\dagger}$ | 1.62e+03 | 1.43e+03$^{\dagger}$ | 6.37e+04 | 3.14e+04$^{\dagger}$ |
| | Best | 1.10e+03 | 1.10e+03 | 1.16e+03 | 1.15e+03 | 1.41e+03 | 1.26e+03 | 2.73e+04 | 1.05e+04 |
| F11 | Worst | 1.11e+03 | 1.11e+03 | 1.29e+03 | 1.29e+03 | 1.96e+03 | 1.71e+03 | 1.16e+05 | 6.02e+04 |
| | Median | 1.11e+03 | 1.11e+03 | 1.23e+03 | 1.19e+03 | 1.60e+03 | 1.42e+03 | 6.29e+04 | 3.11e+04 |
| | Std | 3.02e+00 | 2.13e+00 | 3.56e+01 | 2.98e+01 | 1.28e+02 | 1.09e+02 | 1.93e+04 | 1.03e+04 |
| | Mean | 2.18e+04$^{\approx}$ | 2.58e+04$^{\approx}$ | 1.32e+06 | 8.45e+05$^{\dagger}$ | 6.04e+06 | 2.86e+06$^{\dagger}$ | 5.23e+07 | 2.06e+07$^{\dagger}$ |
| | Best | 4.81e+03 | 4.41e+03 | 1.13e+05 | 4.94e+04 | 1.34e+06 | 8.02e+05 | 2.33e+07 | 8.41e+06 |
| F12 | Worst | 6.42e+04 | 2.55e+05 | 3.25e+06 | 2.52e+06 | 1.25e+07 | 9.17e+06 | 8.80e+07 | 5.28e+07 |
| | Median | 1.90e+04 | 1.36e+04 | 1.14e+06 | 7.23e+05 | 5.96e+06 | 2.41e+06 | 5.14e+07 | 1.90e+07 |
| | Std | 1.44e+04 | 4.58e+04 | 8.29e+05 | 5.97e+05 | 2.82e+06 | 1.73e+06 | 1.46e+07 | 8.67e+06 |
| | Mean | 1.95e+03$^{\approx}$ | 2.33e+03$^{\approx}$ | 3.33e+03$^{\dagger}$ | 1.12e+04 | 3.52e+03$^{\approx}$ | 2.98e+03$^{\approx}$ | 3.89e+03$^{\dagger}$ | 5.16e+03 |
| | Best | 1.33e+03 | 1.39e+03 | 1.52e+03 | 1.43e+03 | 1.74e+03 | 1.47e+03 | 1.90e+03 | 1.61e+03 |
| F13 | Worst | 3.77e+03 | 7.15e+03 | 1.07e+04 | 4.72e+04 | 9.29e+03 | 8.94e+03 | 1.61e+04 | 1.22e+04 |
| | Median | 1.88e+03 | 1.96e+03 | 2.58e+03 | 1.00e+04 | 2.76e+03 | 1.96e+03 | 3.19e+03 | 4.96e+03 |
| | Std | 5.34e+02 | 1.15e+03 | 2.01e+03 | 9.51e+03 | 2.03e+03 | 1.85e+03 | 2.51e+03 | 2.74e+03 |
| | Mean | 1.53e+03$^{\approx}$ | 1.53e+03$^{\approx}$ | 3.54e+04$^{\approx}$ | 2.63e+04$^{\approx}$ | 2.89e+05 | 1.87e+05$^{\dagger}$ | 1.73e+06 | 1.21e+06$^{\dagger}$ |
| | Best | 1.46e+03 | 1.46e+03 | 8.34e+03 | 2.69e+03 | 2.75e+04 | 2.90e+04 | 5.67e+05 | 9.46e+04 |
| F14 | Worst | 1.68e+03 | 1.71e+03 | 1.12e+05 | 8.15e+04 | 6.15e+05 | 4.66e+05 | 3.85e+06 | 3.59e+06 |
| | Median | 1.52e+03 | 1.52e+03 | 2.73e+04 | 2.22e+04 | 2.72e+05 | 1.39e+05 | 1.57e+06 | 1.01e+06 |
| | Std | 5.43e+01 | 6.23e+01 | 2.79e+04 | 2.10e+04 | 1.62e+05 | 1.22e+05 | 8.71e+05 | 7.29e+05 |
| | Mean | 2.00e+03$^{\approx}$ | 2.08e+03$^{\approx}$ | 2.56e+03$^{\approx}$ | 3.48e+03$^{\approx}$ | 3.45e+03$^{\dagger}$ | 1.01e+04 | 2.72e+03$^{\approx}$ | 3.31e+03$^{\approx}$ |
| | Best | 1.61e+03 | 1.57e+03 | 1.62e+03 | 1.56e+03 | 1.75e+03 | 1.74e+03 | 1.87e+03 | 1.73e+03 |
| F15 | Worst | 2.77e+03 | 4.20e+03 | 5.84e+03 | 1.32e+04 | 6.19e+03 | 2.77e+04 | 4.88e+03 | 1.38e+04 |
| | Median | 1.96e+03 | 1.93e+03 | 2.42e+03 | 2.35e+03 | 3.35e+03 | 8.00e+03 | 2.51e+03 | 2.59e+03 |
| | Std | 3.16e+02 | 5.45e+02 | 8.68e+02 | 2.97e+03 | 1.14e+03 | 7.28e+03 | 7.37e+02 | 2.37e+03 |
| | Mean | 1.62e+03 | 1.61e+03$^{\dagger}$ | 2.41e+03$^{\approx}$ | 2.35e+03$^{\approx}$ | 3.27e+03 | 2.98e+03$^{\dagger}$ | 5.48e+03$^{\approx}$ | 5.17e+03$^{\approx}$ |
| | Best | 1.60e+03 | 1.60e+03 | 1.97e+03 | 1.89e+03 | 2.72e+03 | 2.25e+03 | 3.73e+03 | 4.35e+03 |
| F16 | Worst | 1.65e+03 | 1.62e+03 | 2.79e+03 | 2.67e+03 | 4.01e+03 | 3.65e+03 | 6.81e+03 | 6.47e+03 |
| | Median | 1.62e+03 | 1.60e+03 | 2.40e+03 | 2.39e+03 | 3.25e+03 | 3.02e+03 | 5.47e+03 | 5.08e+03 |
| | Std | 1.44e+01 | 7.58e+00 | 2.38e+02 | 2.14e+02 | 3.43e+02 | 3.61e+02 | 6.42e+02 | 6.03e+02 |
| | Mean | 1.73e+03$^{\approx}$ | 1.73e+03$^{\approx}$ | 2.11e+03 | 1.93e+03$^{\dagger}$ | 3.26e+03 | 2.87e+03$^{\dagger}$ | 6.77e+03 | 4.89e+03$^{\dagger}$ |
| | Best | 1.71e+03 | 1.71e+03 | 1.82e+03 | 1.76e+03 | 2.93e+03 | 2.53e+03 | 5.39e+03 | 3.98e+03 |
| F17 | Worst | 1.76e+03 | 1.78e+03 | 2.35e+03 | 2.22e+03 | 3.64e+03 | 3.30e+03 | 7.88e+03 | 6.02e+03 |
| | Median | 1.73e+03 | 1.73e+03 | 2.11e+03 | 1.92e+03 | 3.25e+03 | 2.90e+03 | 6.87e+03 | 4.88e+03 |
| | Std | 1.07e+01 | 1.32e+01 | 1.16e+02 | 9.95e+01 | 1.80e+02 | 1.76e+02 | 7.10e+02 | 5.00e+02 |
| | Mean | 3.79e+03$^{\approx}$ | 3.91e+03$^{\approx}$ | 2.96e+05 | 2.12e+05$^{\dagger}$ | 1.42e+06$^{\approx}$ | 1.28e+06$^{\approx}$ | 4.00e+06 | 2.26e+06$^{\dagger}$ |
| | Best | 2.08e+03 | 1.98e+03 | 5.76e+04 | 4.48e+04 | 2.90e+05 | 3.91e+05 | 1.58e+06 | 5.83e+05 |
| F18 | Worst | 9.65e+03 | 1.14e+04 | 6.65e+05 | 5.97e+05 | 4.15e+06 | 3.24e+06 | 6.45e+06 | 6.57e+06 |
| | Median | 3.41e+03 | 3.03e+03 | 2.83e+05 | 1.63e+05 | 1.21e+06 | 1.19e+06 | 4.04e+06 | 2.03e+06 |
| | Std | 1.61e+03 | 2.21e+03 | 1.66e+05 | 1.34e+05 | 8.92e+05 | 6.68e+05 | 1.24e+06 | 1.37e+06 |
| | Mean | 2.14e+03$^{\approx}$ | 2.15e+03$^{\approx}$ | 2.48e+03$^{\dagger}$ | 4.06e+03 | 3.26e+03$^{\dagger}$ | 1.46e+04 | 2.80e+03$^{\dagger}$ | 3.74e+03 |
| | Best | 1.93e+03 | 1.92e+03 | 1.97e+03 | 1.94e+03 | 2.02e+03 | 4.90e+03 | 2.14e+03 | 2.05e+03 |
| F19 | Worst | 3.46e+03 | 2.79e+03 | 5.65e+03 | 1.76e+04 | 7.73e+03 | 2.72e+04 | 7.22e+03 | 7.63e+03 |
| | Median | 2.08e+03 | 2.09e+03 | 2.29e+03 | 2.77e+03 | 2.66e+03 | 1.50e+04 | 2.55e+03 | 3.29e+03 |
| | Std | 3.05e+02 | 2.25e+02 | 7.00e+02 | 3.41e+03 | 1.43e+03 | 6.26e+03 | 9.92e+02 | 1.71e+03 |
| | Mean | 2.03e+03 | 2.02e+03$^{\dagger}$ | 2.37e+03$^{\approx}$ | 2.33e+03$^{\approx}$ | 3.14e+03 | 2.90e+03$^{\dagger}$ | 5.74e+03 | 4.93e+03$^{\dagger}$ |
| | Best | 2.01e+03 | 2.00e+03 | 2.15e+03 | 2.11e+03 | 2.75e+03 | 2.34e+03 | 4.74e+03 | 3.84e+03 |
| F20 | Worst | 2.04e+03 | 2.03e+03 | 2.58e+03 | 2.56e+03 | 3.46e+03 | 3.44e+03 | 6.43e+03 | 5.97e+03 |
| | Median | 2.03e+03 | 2.02e+03 | 2.37e+03 | 2.31e+03 | 3.20e+03 | 2.86e+03 | 5.79e+03 | 4.91e+03 |
| | Std | 7.84e+00 | 9.84e+00 | 9.99e+01 | 1.10e+02 | 2.10e+02 | 3.13e+02 | 4.83e+02 | 5.03e+02 |

TABLE V: Comparison of PSO and CIPSO on functions F21-F30.

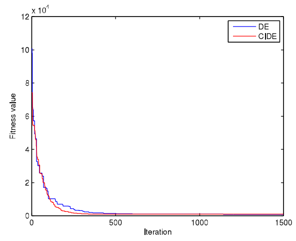
| | | D=10 | | D=30 | | D=50 | | D=100 | |
|---|---|---|---|---|---|---|---|---|---|
| Function | | PSO | CIPSO | PSO | CIPSO | PSO | CIPSO | PSO | CIPSO |
| | Mean | 2.24e+03$^{\approx}$ | 2.25e+03$^{\approx}$ | 2.40e+03 | 2.38e+03$^{\dagger}$ | 2.55e+03 | 2.50e+03$^{\dagger}$ | 3.20e+03 | 3.03e+03$^{\dagger}$ |
| | Best | 2.20e+03 | 2.20e+03 | 2.36e+03 | 2.34e+03 | 2.48e+03 | 2.45e+03 | 3.01e+03 | 2.87e+03 |
| F21 | Worst | 2.33e+03 | 2.32e+03 | 2.43e+03 | 2.41e+03 | 2.61e+03 | 2.59e+03 | 3.37e+03 | 3.17e+03 |
| | Median | 2.21e+03 | 2.24e+03 | 2.40e+03 | 2.38e+03 | 2.56e+03 | 2.50e+03 | 3.19e+03 | 3.04e+03 |
| | Std | 4.90e+01 | 4.47e+01 | 1.97e+01 | 1.74e+01 | 3.92e+01 | 3.48e+01 | 1.05e+02 | 7.50e+01 |
| | Mean | 2.29e+03$^{\approx}$ | 2.30e+03$^{\approx}$ | 4.85e+03$^{\approx}$ | 4.08e+03$^{\approx}$ | 1.09e+04$^{\approx}$ | 1.10e+04$^{\approx}$ | 2.38e+04$^{\approx}$ | 2.42e+04$^{\approx}$ |
| | Best | 2.23e+03 | 2.22e+03 | 2.30e+03 | 2.30e+03 | 8.60e+03 | 8.67e+03 | 1.96e+04 | 1.98e+04 |
| F22 | Worst | 2.30e+03 | 2.30e+03 | 7.71e+03 | 7.76e+03 | 1.25e+04 | 1.30e+04 | 2.71e+04 | 2.70e+04 |
| | Median | 2.30e+03 | 2.30e+03 | 6.16e+03 | 2.31e+03 | 1.09e+04 | 1.12e+04 | 2.39e+04 | 2.40e+04 |
| | Std | 2.63e+01 | 1.69e+01 | 2.15e+03 | 2.11e+03 | 1.10e+03 | 1.20e+03 | 1.84e+03 | 2.21e+03 |
| | Mean | 2.62e+03 | 2.61e+03$^{\dagger}$ | 2.77e+03 | 2.76e+03$^{\dagger}$ | 2.98e+03 | 2.91e+03$^{\dagger}$ | 3.74e+03 | 3.51e+03$^{\dagger}$ |
| | Best | 2.61e+03 | 2.61e+03 | 2.72e+03 | 2.71e+03 | 2.90e+03 | 2.81e+03 | 3.48e+03 | 3.34e+03 |
| F23 | Worst | 2.63e+03 | 2.62e+03 | 2.80e+03 | 2.82e+03 | 3.11e+03 | 3.01e+03 | 4.03e+03 | 3.70e+03 |
| | Median | 2.61e+03 | 2.61e+03 | 2.78e+03 | 2.76e+03 | 2.97e+03 | 2.92e+03 | 3.76e+03 | 3.52e+03 |
| | Std | 4.71e+00 | 3.60e+00 | 1.98e+01 | 2.07e+01 | 4.26e+01 | 5.66e+01 | 1.36e+02 | 1.03e+02 |
| | Mean | 2.72e+03$^{\approx}$ | 2.73e+03$^{\approx}$ | 2.96e+03 | 2.93e+03$^{\dagger}$ | 3.24e+03 | 3.17e+03$^{\dagger}$ | 4.36e+03 | 4.13e+03$^{\dagger}$ |
| | Best | 2.53e+03 | 2.60e+03 | 2.90e+03 | 2.89e+03 | 3.15e+03 | 3.10e+03 | 3.98e+03 | 3.92e+03 |
| F24 | Worst | 2.76e+03 | 2.75e+03 | 3.01e+03 | 2.97e+03 | 3.31e+03 | 3.25e+03 | 4.61e+03 | 4.38e+03 |
| | Median | 2.75e+03 | 2.74e+03 | 2.96e+03 | 2.93e+03 | 3.25e+03 | 3.17e+03 | 4.36e+03 | 4.13e+03 |
| | Std | 6.84e+01 | 3.11e+01 | 2.22e+01 | 1.86e+01 | 4.30e+01 | 3.66e+01 | 1.41e+02 | 1.09e+02 |
| | Mean | 2.89e+03$^{\dagger}$ | 2.91e+03 | 2.88e+03$^{\approx}$ | 2.88e+03$^{\approx}$ | 2.98e+03$^{\dagger}$ | 3.01e+03 | 3.41e+03 | 3.38e+03$^{\dagger}$ |
| | Best | 2.65e+03 | 2.90e+03 | 2.88e+03 | 2.88e+03 | 2.94e+03 | 2.94e+03 | 3.28e+03 | 3.30e+03 |
| F25 | Worst | 2.90e+03 | 2.95e+03 | 2.89e+03 | 2.89e+03 | 3.03e+03 | 3.07e+03 | 3.52e+03 | 3.48e+03 |
| | Median | 2.90e+03 | 2.90e+03 | 2.88e+03 | 2.88e+03 | 2.98e+03 | 3.01e+03 | 3.41e+03 | 3.37e+03 |
| | Std | 4.60e+01 | 1.81e+01 | 3.70e+00 | 4.64e+00 | 2.48e+01 | 2.80e+01 | 5.48e+01 | 4.25e+01 |
| | Mean | 2.83e+03$^{\approx}$ | 2.87e+03$^{\approx}$ | 4.43e+03 | 4.31e+03$^{\dagger}$ | 5.73e+03 | 5.37e+03$^{\dagger}$ | 1.74e+04 | 1.52e+04$^{\dagger}$ |
| | Best | 2.60e+03 | 2.60e+03 | 3.02e+03 | 3.87e+03 | 4.82e+03 | 4.43e+03 | 1.28e+04 | 1.28e+04 |
| F26 | Worst | 2.94e+03 | 3.00e+03 | 4.76e+03 | 4.78e+03 | 6.52e+03 | 6.38e+03 | 2.03e+04 | 1.69e+04 |
| | Median | 2.90e+03 | 2.90e+03 | 4.53e+03 | 4.31e+03 | 5.77e+03 | 5.35e+03 | 1.77e+04 | 1.52e+04 |
| | Std | 1.04e+02 | 7.08e+01 | 3.30e+02 | 2.42e+02 | 4.51e+02 | 4.95e+02 | 1.57e+03 | 9.26e+02 |
| | Mean | 3.08e+03 | 3.07e+03$^{\dagger}$ | 3.20e+03$^{\approx}$ | 3.20e+03$^{\approx}$ | 3.20e+03$^{\approx}$ | 3.20e+03$^{\approx}$ | 3.20e+03$^{\approx}$ | 3.20e+03$^{\approx}$ |
| | Best | 3.07e+03 | 3.07e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 |
| F27 | Worst | 3.09e+03 | 3.09e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 |
| | Median | 3.08e+03 | 3.07e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 | 3.20e+03 |
| | Std | 2.98e+00 | 3.53e+00 | 1.03e-04 | 1.03e-04 | 1.18e-04 | 1.19e-04 | 1.26e-04 | 1.30e-04 |
| | Mean | 3.27e+03 | 3.27e+03$^{\dagger}$ | 3.30e+03$^{\approx}$ | 3.30e+03$^{\approx}$ | 3.30e+03$^{\approx}$ | 3.30e+03$^{\approx}$ | 3.30e+03$^{\approx}$ | 3.30e+03$^{\approx}$ |
| | Best | 3.27e+03 | 3.27e+03 | 3.29e+03 | 3.29e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 |
| F28 | Worst | 3.27e+03 | 3.27e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 |
| | Median | 3.27e+03 | 3.27e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 | 3.30e+03 |
| | Std | 5.38e-03 | 7.69e-05 | 1.81e+00 | 2.71e+00 | 1.24e-04 | 1.15e-04 | 1.44e-04 | 1.56e-04 |
| | Mean | 3.18e+03$^{\approx}$ | 3.18e+03$^{\approx}$ | 3.61e+03 | 3.52e+03$^{\dagger}$ | 4.43e+03 | 4.24e+03$^{\dagger}$ | 6.97e+03 | 6.27e+03$^{\dagger}$ |
| | Best | 3.14e+03 | 3.16e+03 | 3.23e+03 | 3.33e+03 | 3.95e+03 | 3.84e+03 | 5.77e+03 | 4.87e+03 |
| F29 | Worst | 3.21e+03 | 3.20e+03 | 3.85e+03 | 3.74e+03 | 4.98e+03 | 4.64e+03 | 7.71e+03 | 7.03e+03 |
| | Median | 3.18e+03 | 3.18e+03 | 3.61e+03 | 3.49e+03 | 4.39e+03 | 4.23e+03 | 7.00e+03 | 6.32e+03 |
| | Std | 1.94e+01 | 1.48e+01 | 1.29e+02 | 1.19e+02 | 2.58e+02 | 1.97e+02 | 4.20e+02 | 4.96e+02 |
| | Mean | 4.15e+03$^{\approx}$ | 4.40e+03$^{\approx}$ | 3.54e+03$^{\approx}$ | 3.87e+03$^{\approx}$ | 3.96e+03$^{\approx}$ | 4.08e+03$^{\approx}$ | 4.74e+03$^{\dagger}$ | 7.61e+03 |
| | Best | 3.26e+03 | 3.24e+03 | 3.26e+03 | 3.23e+03 | 3.39e+03 | 3.31e+03 | 3.54e+03 | 3.36e+03 |
| F30 | Worst | 7.27e+03 | 9.09e+03 | 4.35e+03 | 7.34e+03 | 7.26e+03 | 7.41e+03 | 1.35e+04 | 1.92e+04 |
| | Median | 3.97e+03 | 3.80e+03 | 3.42e+03 | 3.35e+03 | 3.80e+03 | 3.81e+03 | 4.11e+03 | 5.74e+03 |
| | Std | 8.84e+02 | 1.46e+03 | 3.02e+02 | 1.10e+03 | 7.19e+02 | 9.40e+02 | 1.90e+03 | 4.33e+03 |
| w/t/l | | - | 12/16/2 | - | 14/13/3 | - | 16/11/3 | - | 18/8/3 |



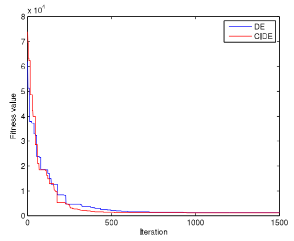
(a) F6



(b) F7

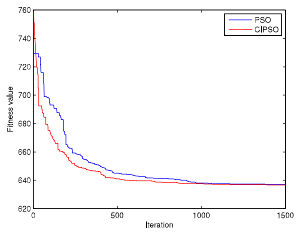


(c) F9

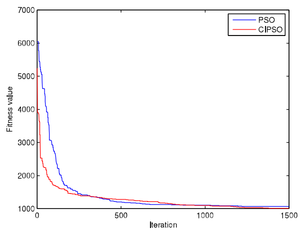


(d) F11

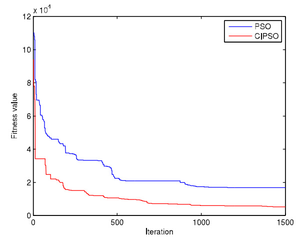
Fig. 3: Convergence plot for F6, F7, F9, F11 on D=50 for DE and CIDE.



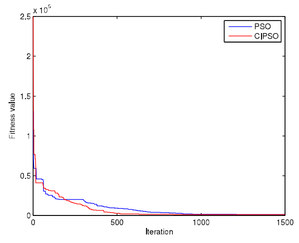
(a) F6



(b) F7



(c) F9



(d) F11

Fig. 4: Convergence plot for F6, F7, F9, F11 on D=50 for PSO and CIPSO.

TABLE VI: The quality of resulted candidate solutions using collective intelligence compared with best, worst, and whole population.

| | CIDE | | | CIPSO | | |
|---|---|---|---|---|---|---|
| D | Better than Best | Beter than Worst | AvgRank | Better than Best | Beter than Worst | AvgRank |
| 10 | 4.47 | 58.01 | 36.29 | 4.13 | 68.45 | 43.45 |
| 30 | 1.49 | 58.31 | 39.04 | 2.05 | 63.39 | 45.23 |
| 50 | 0.8945 | 54.43 | 42.12 | 1.4 | 70.02 | 60.82 |
| 100 | 7.4 | 68.51 | 53.67 | 8.3 | 79.12 | 65.34 |

tion of the new vector to population have a significant effect on improvement of acceleration rate of algorithms. In the present study, collective intelligence are applied in single-objective optimization while multi-objective case can be considered in future work. Moreover, as a future work, the effect level of intelligence for each individual can be defined according to its fitness value (e.g., utilizing weighted voting in generating new candidate solution).

## References

[1] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2191–2233, 2019.

[2] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Information Sciences*, vol. 295, pp. 407–428, 2015.

[3] A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," in *Advances in evolutionary computing*. Springer, 2003, pp. 819–845.

[4] A. Chakraborty and A. K. Kar, "Swarm intelligence: A review of algorithms," in *Nature-Inspired Computing and Optimization*. Springer, 2017, pp. 475–494.

[5] Y. Wang, "A sociopsychological perspective on collective intelligence in metaheuristic computing," *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 1, no. 1, pp. 110–128, 2010.

[6] R. Kicinger, T. Arciszewski, and K. De Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art," *Computers & structures*, vol. 83, no. 23-24, pp. 1943–1978, 2005.

[7] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[8] S. P. T. P. Phyu and G. Srijuntongsiri, "Effect of the number of parents on the performance of multi-parent genetic algorithm," in *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*. IEEE, 2016, pp. 1–6.

[9] E. Bonilla-Huerta, J. C. H. Hernández, and R. Morales-Caporal, "A specialized random multi-parent crossover operator embedded into a genetic algorithm for gene selection and classification problems," in *Electrical Engineering and Control*. Springer, 2011, pp. 559–566.

[10] L. M. Zheng, S. X. Zhang, K. S. Tang, and S. Y. Zheng, "Differential evolution powered by collective information," *Information Sciences*, vol. 399, pp. 13–29, 2017.

[11] M. Elbes, S. Alzubi, T. Kanan, A. Al-Fuqaha, and B. Hawashin, "A survey on particle swarm optimization with emphasis on engineering and network applications," *Evolutionary Intelligence*, pp. 1–17, 2019.

[12] O. Moslah, Y. Hachaïchi, and Y. Lahbib, "Democratic inspired particle swarm optimization for multi-robot exploration task," 2016.

[13] N. Monmarché, F. Guinand, and P. Siarry, *Artificial ants*. Wiley-ISTE Hoboken, 2010.

[14] M. E. Aydin, J. Wu, and L. Zhang, "Swarms of metaheuristic agents: A model for collective intelligence," in *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. IEEE, 2010, pp. 296–301.

[15] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4. Citeseer, 1995, pp. 1942–1948.

[16] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[17] S. Mahdavi, S. Rahnamayan, and A. Mahdavi, "Majority voting for discrete population-based optimization algorithms," *Soft Computing*, vol. 23, no. 1, pp. 1–18, 2019.

[18] T. Tsujimoto, T. Shindo, T. Kimura, and K. Jin'no, "A relationship between network topology and search performance of pso," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–6.

[19] Y.-X. Wang and Q.-L. Xiang, "Particle swarms with dynamic ring topology," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 419–423.

[20] C. W. Cleghorn and A. Engelbrecht, "Fully informed particle swarm optimizer: Convergence analysis," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 164–170.

[21] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[22] V. Faber, "Clustering and the continuous k-means algorithm," *Los Alamos Science*, vol. 22, no. 138144.21, p. 67, 1994.

[23] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *computer*, vol. 27, no. 6, pp. 17–26, 1994.

[24] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 special sessionand competition on single objective bound constrained real-parameter numerical optimization," *Technical Report*, 2016.

[25] V. Kreischer, T. T. Magalhaes, H. J. Barbosa, and E. Krempser, "Evaluation of bound constraints handling methods in differential evolution using the cec2017 benchmark," in *XIII Brazilian Congress on Computational Intelligence, Rio de Janeiro, Brazil*, 2017.

[26] J. Kennedy, R. Poli, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[27] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.