Fuzzy Rule-based Design of Evolutionary Algorithm for Optimization

Saber Elsayed[1], Ruhul Sarker[1] and Carlos Coello Coello[2]

***Abstract*—During the last two decades, many multi-operator- and multi-method-based evolutionary algorithms for solving optimization problems have been proposed. Although, in general terms, they outperform single-operator-based traditional ones, they do not perform consistently for all the problems tested in the literature. The designs of such algorithms usually follow a trial and error approach that can be improved by using a rule-based approach. In this paper, we propose a new way for two algorithms to cooperate as an effective team, in which a heuristic is applied using fuzzy rules of two complementary characteristics, the quality of solutions and diversity in the population. In this process, two sub-populations are used, one for each algorithm, with greater emphasis placed on the better-performing one. Inferior algorithms learn from trusted ones and a fine-tuning procedure is applied in the later stages of the evolutionary process. The proposed algorithm was analyzed on the CEC2014 unconstrained problems and then tested on other three sets (CEC2013, CEC2005 and 12 classical problems), with its results showing a high success rate and that it outperformed both single-operator-based and different state-of-the-art algorithms.**

***Keywords*—Multi-method, multi-operator, fuzzy logic, optimization**

I. Introduction

Continuous optimization problems involve finding the values of continuous decision variables so that one or more objective functions is optimized (either maximized or minimized). They can be found in many fields including, but not limited to, science, engineering and business [17]. Generally, the mathematical model of a single objective problem with continuous search domains can be formulated as

$$\text{minimize or maximize } f(\overrightarrow{x})$$

$$\text{subject to: } \underline{x}_j \leq x_j \leq \overline{x}_j,\ j = 1, 2, \ldots, D \qquad (1)$$

where $f(\overrightarrow{x})$ is the objective function, $\overrightarrow{x} = [x_1, x_2, ..., x_D]$ a vector with D decision variables with each x_j has lower and upper limits $\underline{x}_j$ and $\overline{x}_j$, respectively.

During the last decades, evolutionary algorithms (EAs) (such as genetic algorithms (GAs) [12], differential evolution (DE) [52] and evolution strategy (ES) [29]), and swarm intelligence (SI) methods (such as ant colony optimization (ACO) [51] and particle swarm optimization (PSO) [13]) have demonstrated their success in solving such problems. However, it has been found that no single optimization algorithm (OA) performs consistently well for all types of problems; for instance, (1) a GA performs well in solving noisy problems but its convergence is slow compared with that of DE [22], (2) the covariance matrix adaptation ES (CMA-ES) is very good at solving uni-modal problems but becomes trapped in local solutions when solving multi-modal functions [28], (3) DE is a good choice when feasible patches are parallel to the axes, but when solving multi-modal functions, it could become stuck in local optima [22], and (4) PSO is characterized by its high convergence rate in the early stages of the optimization process, but becomes slow in its refinement stage, and may move away from the global optima [22].

As a single OA design might not perform well for many problems, several methods that utilize the search capabilities of different algorithms and/or search operators in a single algorithm framework have been proposed. They have different names, such as (1) ensemble-based (which use a mix of methods [38]), (2) hyper-heuristic (a heuristic that selects other heuristics for an effective search [8]), (3) multi-method (more than one OA is used in one framework [60][16]), (4) multi-operator (emphasis is placed adaptively on the best search operator, of many, in a single OA [19, 20, 17]), (5) heterogeneous (e.g., heterogeneous PSO in which particles in a swarm are allocated different search behaviors from a behavior pool) [23] and (6) population-based algorithm portfolios (PAP) (a combination of multiple EAs [44, 56]). Although the abovementioned methods are different in their designs, they share the common concept of using a pool of different algorithms and/or operators and a selection procedure for placing emphasis on the best-performing one during the optimization process. However, they use different selection mechanisms to rank the algorithms in the pool, multiple populations rather than a single one and may consider different OAs in the pool.

Generally, although such designs usually lead to better performances than those of all OAs in a pool, they do not guarantee consistent results for all test problems in the literature; for example, in [44], PAP was statistically outperformed by its constituent algorithms for many problems, as was the case in [39]. The apparent weaknesses of these algorithms are that: (i) their designs are based mainly on trial and error approaches; and (ii) they may not follow any design principles that may either ensure an improvement in performance or reduce the risk of failing.

Based on research on the organizational behaviour [31], an effective organizational team is made up of group members who possess complementary skills, which can lead to a diverse group's ability to generate decision making alternatives. Therefore, in our design of a successful framework, we introduce a definition of a team of OAs (TOAs) as "*a group of optimization methods/operators organized to work interdependently as well as cooperatively to accomplish a common goal, i.e., optimize an objective function, under predefined guidelines*"; in other words, an appropriate mix of different algorithms and/or operators that utilizes the individual components' strengths in the best possible way.

Motivated by these points, in this paper, we propose a TOAs

[1]The authors are with the School of Engineering and Information Technology, University of New South Wales Canberra, Australia, emails: s.elsayed@unsw.edu.au, r.sarker@unsw.edu.au.

[2]The author is with the Depto. de Computación, CINVESTAV-IPN, Mexico, email: ccoello@cs.cinvestav.mx.

framework composed of a set of algorithms which possesses complementary characteristics (CC). As CC measurements in EAs are vague and imprecise, the TOAs framework uses a novel heuristic with fuzzy rules to place the right emphasis on the algorithm with the appropriate CC. This complementarity (also called effectiveness) is calculated using the two basic characteristics of EAs, (1) solution quality and (2) diversity, with information shared between a trusted OA and inferior ones. Also, as the measurement of the trust of an OA is vague, fuzzy rules are useful and, in the later stages of a team life-cycle, the group's outcome is polished through a dynamic fine-tuning procedure. Note that, in the proposed design, the emphasis on each constituent algorithm (/operator) within the framework may change adaptively for every problem under consideration as the evolution progresses.

The framework was adopted with two well-known complementary algorithms, with its performance firstly analyzed on 30 unconstrained problems introduced in [34], with 10, 30 and 50 dimensions. The results demonstrated that the TOAs was able to reach success rates (statistically better or the same as those of its team members) of 100%, 100% and 97%, respectively, for these problems and also outperformed well-known state-of-the-art algorithms. TOAs was then tested on other 65 unconstrained problems taken from three different benchmark sets, with TOAs outperforming several state-of-the-art algorithms.

The rest of this paper is organized as follows: an overview of related work is presented in Section II; the proposed algorithm is described in Section III; and the experimental results and conclusions are discussed in Sections IV and V, respectively.

II. Brief Review

As previously mentioned, OAs refer to many methods including, but not limited to, EAs and SI. EAs are population-based search methods that employ some form of selection to bias the search toward good solutions. Their steps are almost the same, with variations in only their sequences and the ways in which they generate new individuals with environmental pressure usually causing natural selection (survival of the fittest) to focus on the more promising search space. New offspring are then created by executing crossover and/or mutation operators or vice versa [14]. On the other hand, SI algorithms, which were inspired by the behaviors of insects, birds and fish, have unique capabilities for solving complex tasks in the form of swarms, i.e., PSO starts with initial particles which fly through the problem's hyperspace at given velocities and are then updated in each generation [13].

To effectively solve a wide range of test problems using a single algorithm, concepts of combining more than one method and/or operator in a single algorithmic framework have been proposed, a brief review of which is provided below.

Vrugt et al. [59] introduced an algorithm known as a multi-algorithm genetically adaptive multi-objective (AMALGAM) that proved to be a powerful approach for solving multi-objective problems. Later, it was modified to solve single-objective ones (AMALGAM-SO) [60] which used a GA, CMA-ES and PSO, and automatically tuned the number of offspring these three OAs were allowed to contribute during each generation. It obtained similar efficiencies as existing algorithms for relatively simple problems but was increasingly superior for more complex and higher-dimensional multi-modal optimization ones. However, it was noted that, if DE was included in the framework, the performance of AMALGAM-SO could deteriorate [60].

Peng et al. [44] proposed the PAP framework which used multiple EAs as its constituent algorithms, each of which was run for a given number of test problems for a part of the given fitness evaluations (time budget), and a migration scheme among the algorithms. It showed its superiority to other algorithms on a set of unconstrained problems but, interestingly, was statistically outperformed by some of the OAs in the pool. This work was then extended by Tang et al. [56], who used an estimated performance matrix (EPM-PAP) module for automatic selection of the constituent algorithms, each of which was applied to each problem for a predefined number of runs. Then, an EPM was constructed for each algorithm based on the quality of solutions obtained in each run and, subsequently, the risk of using each algorithm was determined, with the subset with the smallest risk used to run in parallel and periodically share information. This method was tested on a set of unconstrained problems and obtained good results. However, calculating risk considering only the quality of solutions might not be sufficient, i.e., complementary measures could be a better option. Also, the calculations were based on running the algorithms for a predefined number of generations in the early stages of the evolutionary process which meant that a bias could occur as one algorithm might perform better in the later stages [18, 17].

In [25], a multi-method hyper-heuristic algorithm using seven common meta-heuristics in the lower level of a hyper-heuristic framework was proposed. Different strategies for selecting the most appropriate meta-heuristic in each generation of the optimization process were tested. Its performance was evaluated on a few real parameter benchmark problems and obtained promising results. This algorithm was then extended to investigate the impact of different heuristic space diversity (HSD) strategies [26], with the exponentially increasing one outperforming the others.

Elsayed et al. [16] proposed united multi-operator EAs (UMOEAs), where multiple OAs were used, each of which was a self-adaptive multi-operator algorithm. UMOEAs was used to solve a set of unconstrained problems, with the results showing significant improvements in comparison with existing algorithms. However, its performance could be further improved with a careful design of the algorithmic framework.

Masegosa et al. [40] proposed a centralized cooperative strategy, where a set of trajectory-based methods were controlled by a rule-driven coordinator. The algorithm consisted of a set of agents that were run in parallel, with a coordinator receiving information about their performance and sending orders to them. The algorithm was tested on small-and large-scale problems and showed competitive results, though for some test problems it dit not outperform those of the other methods. Xue et al. [64] integrated three self-adaptive learning OAs by dividing the population into three sub-populations each of which was evolved using a different OA, with an information exchanging manners (IEM) used during the optimization process. The algorithm with different IEMs was tested on a set of unconstrained problems and showed a competitive performance to those of other OAs.

López-Ibáñez et al. [36] introduced the irace package which implements a general iterated racing procedure, including I/FRace as a special case [4]. The package involved (1) sampling from a truncated normal distribution; (2) a parallel implementation; (3) a restart strategy; (4) and an elitist

racing. López-Ibáñez and Stützle [37] used the hypervolume measure to compare the performances of OAs in terms of Pareto-optimality and then integrated this measure in irace. In [35], an automatic algorithm configuration tool was applied to to improve the performance of ACMA-ES algorithm, by separating the tuning and testing sets. The improved version of CMA-ES was superior to algorithms in solving unconstrained problems. However, doing the automatic configuration during the evolutionary process would be interesting and more practical.

Considering multi-operator-based algorithms, Qin et al. [45] proposed a self-adaptive DE algorithm (SaDE) that used four mutation strategies, to one of which each individual in the population was assigned based on a given probability. Then, the selection probability of each operator was updated based on its success and failure rates during previous generations (a learning period). Zamuda and Brest [72] introduced an algorithm that employed two mutation strategies in their earlier algorithm proposed [6], with the population size adaptively reduced during the evolutionary process. Its performance on some real-world problems was better than that of two other algorithms. An adaptive DE algorithm [9], which utilized four mutation strategies in a sequential manner, i.e., one mutation in every predefined number of generations, was introduced, with a mechanism also used to reduce the population size. Wang et al. [63] introduced a composite DE algorithm (CoDE) in which, in each generation, a trial vector was generated by randomly combining three DE variants with three control parameter settings. This algorithm performed well on a set of unconstrained test problems.

Regarding SI approaches, a mix of different PSO variants, each of which evolved with a different number of individuals, was proposed in [21]. In each generation, the algorithm assigned more individuals to the better- and fewer to the worse-performing variants. It was tested on a reasonable number of constrained problems, with the results showing its effectiveness. Nepomuceno and Engelbrecht [42] proposed a frequency-based heterogeneous PSO for solving real-parameter optimization functions. This algorithm kept track of the frequency of success of the behaviors of the particles for a number of iterations in order to use it as a selection criterion, and demonstrated better performances than other heterogeneous PSOs.

A. Fuzzy Theory

As fuzziness is involved in our daily conversation, Zadeh [70] introduced the field of fuzzy theory in the mid 60's . In it, to represent uncertainty, Zadeh defined the term "*fuzzy sets*" as those sets whose boundaries are not clear [41]. Generally, a fuzzy set y in S (S is a space of objects whose elements are denoted by s, i.e., $S = \{s\}$) is described by a membership function $\mu_y(s)$ of a real number $\in [0, 1]$ associated with each point in S, where $\mu_y(s)$ is the grade of membership s in y. The closer the value of $\mu_y(s)$ is to 1, the more s belongs to y [70]. There are many types of membership functions, such as Gaussian, generalized bell curve, triangular and trapezoidal.

As a natural language is fuzzy (i.e., involves vague and imprecise expressions) [48], fuzzy logic was developed for computing with words [69]. The idea behind it is to map an input space to an output one. To do this, the core mechanism creates a list of 'if-then' rules (i.e., if the antecedent then the consequent) which are then converted by a fuzzy system to their mathematical equivalents.

Generally, to create a fuzzy logic system, the following steps are required [50].

1) Fuzzification determines the degree to which a crisp number (a system input) belongs to each of the appropriate fuzzy sets via a membership function.
2) An inference engine imitates a human's thinking by making fuzzy inferences regarding the inputs and if-then rules, i.e., applying a fuzzy operator (AND or OR) in the antecedent, which infers from the antecedent to consequent and aggregates the consequent across the rules (max, probabilistic OR and sum of each rule's output set).
3) Defuzzifucation transforms the fuzzy set obtained by the inference engine into a crisp value. There are different defuzzification methods, such as (*i*) centroid which returns the center of an area under the curve, (*ii*) bisector which divides the curve into two equal parts, (*iii*) middle of maximum which is the average of the maximum values of the output set, (*iv*) largest of maximum which is the largest value of the domain with the maximal degree of membership and (*v*) smallest of maximum which is the smallest value of the domain with the maximal degree of membership. Of these methods, the centroid calculation is the most popular.

Over the years, fuzzy logic has shown its benefits which include, but are not limited to, (1) its ease of understanding, (2) more intuitive approach, (3) flexibility, (4) tolerance of imprecise data and (5) suitability for many applications ranging from the basic sciences to engineering, social and biomedical systems, and consumer products [71].

B. Fuzzy logic in OAs

Generally, uses of fuzzy logic in OAs were mainly to adapt their parameters which is not considered in this paper. Below is a brief review on uses of fuzzy logic in OAs.

Bernal et al. [3] proposed a method of using fuzzy logic to find the optimal values of the imperialist competitive algorithm (ICA) parameters (β, ξ). Three fuzzy systems were used, with the first conducted to determine the best value of β, second ξ and third β and ξ. All the fuzzy systems were of a Mamdani type with the input defined as the decades. On only six unconstrained problems solved, the results were competitive with other algorithms. Similarly, Valdez et al. [58] applied fuzzy logic to dynamically adapt the inertia weight and learning factors of a PSO. The algorithm was tested on a set of unconstrained problems and showed better results than those of the same algorithm with different adaptation mechanisms. A similar mechanism was also carried out to adapt the parameters of a grey wolf optimizer (GWO) [47] and DE [43]. For GAs, Herrera and Lozano [30] presented a GA variant with adaptive genetic operators based on coevolution with fuzzy behaviors. In it, the adaptation took place at the individual level by means of fuzzy logic and the fuzzy rule bases used by fuzzy logic come from a separate GA. The algorithm was tested on a set of unconstrained problem with promising results achieved.

III. Team of Optimization Algorithms (TOAs)

As previously discussed, the current frameworks which combine more than one operator and/or OA, may not outperform all their individual algorithms/operators for all test

problems. Therefore, as solving an unknown problem using such an algorithm will provide no confidence about the quality of solutions, we propose a new heuristic in the design of a TOAs framework.

In the beginning, let $\mathrm{A}_{set} = \{\mathrm{A}_i | i = 1, 2, .., N_{alg}\}$ be a set of N_{alg} OAs, and $F_{set} = \{F_k | k = 1, 2, ...N_{fun}\}$ be a set of problems to solve.

The first rule is that A_{set} should be a set of complementary algorithms and the second that, to determine which algorithm to apply or the probability of evolving individuals using an OA, the heuristic should be based on useful complementary criteria, such as the quality of solutions and diversity.

For a minimization problem, assuming that $\mathrm{A}_{set} = \{\mathrm{A}_1, \mathrm{A}_2\}$, A_1 and A_2 can be considered complementary algorithms if:

1) $N_{fun,1}\left(\overline{f}_{A_1} < \overline{f}_{A_2}\right) > 0$ and $N_{fun,2}\left(\overline{f}_{A_1} > \overline{f}_{A_2}\right) > 0$, where $\overline{f}_{A_j}$ is the average fitness value obtained by running the i^{th} algorithm a few times and $\left(N_{fun,1} \cup N_{fun,2} \cup N_{fun}\left(\overline{f}_{A_1} = \overline{f}_{A_2}\right)\right) = N_{fun}$. Note that the best fitness value can be used instead of $\overline{f}$, but the latter may give a better indication about the algorithm's performance.
2) the search capabilities of A_{set}, during the evolutionary process, are complementary, i.e., $N_{g,1}\left(\overline{f}_{A_1,g} < \overline{f}_{A_2,g}\right) \geq g_\epsilon$ and $N_{g,2}\left(\overline{f}_{A_1,g} > \overline{f}_{A_2,g}\right) \geq g_\epsilon$, where $N_{g,1}\left(\overline{f}_{A_1,g} < \overline{f}_{A_2,g}\right)$ denotes the number of generations that A_1 performs better than A_2, $g_\epsilon \in [1, N_{g_{max}}]$ a defined number of generations, $\left(N_{g,1} \cup N_{g,2} \cup N_g\left(\overline{f}_{A_1,g} = \overline{f}_{A_2,g}\right)\right) = N_{g_{max}}$, and $N_{g_{max}}$ the maximum number of generations. Figure 1 depicts an example that fulfils this rule, in which A_2 has better average fitness values for the first $g_\epsilon = 350$ generations, then A_1 converges faster in later generations.
3) A_{set} holds CC, i.e., A_1 is good based on the quality of solutions, while A_2 has a better diversity rate (div), such that $\overline{f}_{A_1} < \overline{f}_{A_2}$ and $div_{A_1} < div_{A_2}$ or vice versa.

Note that the 2^{nd} and 3^{rd} conditions may hold for only one problem.

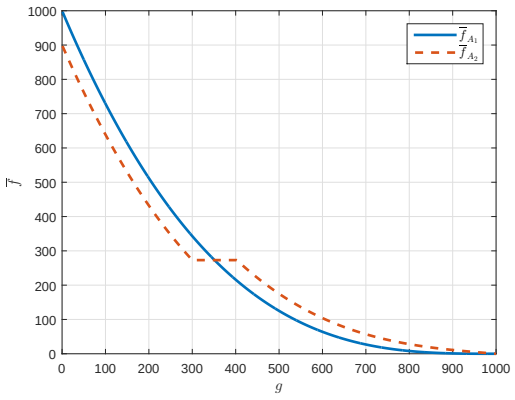


Figure 1. Example of complementary algorithms during evolutionary process

In the literature, it is recognized that, although sharing information among sub-populations (/individual algorithms) is important for improving performance [44, 17], it may deteriorate the team's performance if it is not appropriately conducted. Therefore, we propose rules for determining how to share information between a trusted (high-performing) algorithm and low-performing ones, with fuzzy rules used to measure the effectiveness of an algorithm based on its results in terms of diversity and quality of solutions, as discussed in section III-B.

Finally, in practice, finely tuning the performance of a team remains central to the systematic improvement of the final product (result). Therefore, having a dynamic fine-tuning procedure may ensure the achievement of quality, i.e., keep using it if it is worthwhile.

Based on the abovementioned rules, a general framework is generated, with its steps described in Algorithm 1. Firstly, an initial population (X) of size PS is randomly generated $\left(X = \left\{\overrightarrow{X}_1, \overrightarrow{X}_2, ..., \overrightarrow{X}_{PS}\right\}\right)$ with the probability of each algorithm being applied set to 1, i.e., $prob_i = 1 \forall i = \{1, 2, ..N_{alg}\}$. Also, the initial diversity of each algorithm and the best fitness value in X is recorded. Then, X is divided into N_{algo} sub-populations $\left(X = \{x_1, x_2, ..., x_{N_{alg}}\}\right)$, each of which is conditionally evolved by a different A, i.e., based on its $prob_i$, and each x_i is of size PS_i.

In each generation, N_{alg} random numbers are generated, i.e., $rand_i \in [0, 1] \; \forall i = \{1, 2, ..N_{alg}\}$, if $rand_i \leq prob_i$, x_i is updated using A_i. Note that in the first cycle (a predefined number of generations) all the algorithms are used to evolve their corresponding sub-populations. This process is repeated for a cycle (CS) and, when it is finished, every $prob_i$ is updated based on its effectiveness, as discussed in section III-A.

For each generation in the 2nd cycle, N_{alg} random numbers are generated, with at least one $rand$ having to be less than its corresponding $prob_i$ to make sure that at least one algorithm is applied in each generation. Similar to the first cycle, if $rand_i \leq prob_i$, then A_i is used to update x_i $\forall i = \{1, 2, ..N_{alg}\}$. Then, at the end of the second cycle, the effectiveness of each algorithm is calculated. Subsequently, two actions occur: (1) information is shared among algorithms if a condition is satisfied (see section III-B); (2) each $prob_i$ is set to 1, i.e., returns to the first cycle. The reasons for the second action are, (1) if the information-sharing procedure has taken place, all algorithms are provided with a chance to evolve equally with the new individuals which may help to change their search capabilities, and (2) the search capabilities of OAs may vary during the optimization process, i.e., one algorithm may be good in the early stages but perform poorly in later generations (this characteristic is shown in Figure 3 in the supplementary materials and will be discussed later).

Finally, a local search is used as a fine tuning of the team performance during the last stages of the optimization algorithm (section III-C). The algorithm continues until a stopping criterion is met.

In the following subsections, each component of the TOAs framework is discussed in detail.

A. Fuzzy rules-based heuristic

As previously mentioned, it is crucial to select an OA during the optimization process based on CC. In this paper, two factors are considered CC: (1) the quality of solutions obtained; and (2) the diversity of a population generated by an algorithm. As these factors can be described by linguistic variables, e.g., the diversity is low and the quality of solutions

Algorithm 1 General framework of TOAs

Define PS, $cy \leftarrow 0$, $prob_i \leftarrow 1$, $g \leftarrow 1$, and all other parameters required (Section IV).
At $g = 1$, generate random individuals (X), and devide them into N_{algo} groups, i.e., $X = \{x_1, x_2, ..., x_{N_{alg}}\}$.
while $cfe < FFE_{max}$ **do**
  $cy \leftarrow cy + 1$.
  if $cy = CS$ **then**
    Measure the effectiveness of each A_i (Section III-A).
    Update $prob_i$.
  end if
  if $cy = 2 \times CS$ **then**
    Share information (Section III-B).
    $prob_i \leftarrow 1$.
    $cy \leftarrow 0$; .
  end if
  Generate $rand_i \in [0, 1]$, with at least one satisfies $rand_i \in [0, 1] \leq prob_i$.
  if $rand_i \leq prob_i$ **then**
    Evolve x_i using A_i.
    Update cfe and sort x_i.
  end if
  if $finetuningstage$ **then**
    if $rand \in [0, 1] \leq prob_{ls}$ **then**
      Apply local search.
      Update $prob_{ls}$.
    end if
  end if
  $g \leftarrow g + 1$, and go to step 4.
end while

high, fuzzy logic can help in the decision-making process for determining the most effective algorithm to use. In this paper, five levels (subsets) of each fuzzy set are considered, that is, {very low (VL), low (L), medium (M), high (H) and very high (VH)}.

Firstly, the quality of solutions is represented as the difference between the optimal (or best known fitness value if the optimal one is unknown) and best fitness values in each sub-population such that

$$Q_i = f^* - f_{g,i}^{best} \, \forall \, i = 1, 2, ...N_{alg} \tag{2}$$

where f^* is the optimal or best known solution and $f_{t,i}^{best}$ the best objective value of the i^{th} sub-population. This means that the closer the value of Q_i to 0, the better quality it is.

The diversity rate is calculated as the average distance of each individual in x_i to the best solution among them:

$$div_i = \frac{\sum_{z=1}^{PS_i} div\left(\overrightarrow{x}_{i,z}, \overrightarrow{x}_{best}\right)}{PS_i}, \, \forall \, i = 1, 2, ...N_{alg} \tag{3}$$

where $div\left(\overrightarrow{x}_z, \overrightarrow{x}_{best}\right)$ is the Euclidean distance between the z^{th} individual and best individual in x_i. Note that the best solution is $\overrightarrow{x}_1$, as, in every generation, each sub-population is sorted based on the fitness values.

Subsequently, it is crucial to define the scale of each fuzzy set (universal space), i.e., its lower and upper limits. Based on equation 2, the upper bound (UB_Q) of the quality of solutions is zero. Regarding the lower bound (LB_Q), although the simplest way is to fix it to the maximum (Q_i) at $g = 0$, during the optimization process, it is possible that the initial value will be very large and the algorithm will become trapped in a local solution in later generations which may be far from the optimal solution but falls within the 'VH' level of quality. To clarify this, assuming that the max min $f_{g=0,i} \, \forall \, i = \{1, 2, ...N_{alg}\}$ is $1e + 10$ and $f^* = 0$, the limits will be $[-1e + 10 \quad 0]$. If the membership follows a trapezoidal function, as discussed later, any fitness value better than $2.5e + 08$ will take a high membership value. However, this value is too far from the optimal solution. Therefore, we propose an automatic update of this value over generations, i.e., changing it every CS generation.

Algorithm 2 Steps of updating lower and upper bounds of quality of solutions and diversity

At $g = 0$, $cy \leftarrow 0$; calculate Q_i (eq. 2) and div_i (eq. 3) $\forall i = 1, 2, ..N_{alg}$;
$UB_Q \leftarrow 0$; $LB_Q \leftarrow \text{Max}\left(Q_{i \;\; \forall i=1,2,..N_{alg}}\right)$;
$LB_{div} \leftarrow 0$; $UB_{div} \leftarrow \text{Max}\left(div_{i \;\; \forall i=1,2,..N_{alg}}\right)$;
while $cfe < FFE_{max}$ **do**
  $cy \leftarrow cy + 1$;
  if $cy = CS$ or $cy = 2 \times CS$ **then**
    calculate Q_i (eq. 2) and div_i (eq. 3) $\forall i = 1, 2, ..N_{alg}$;
    $LB_Q \leftarrow \text{Min}\left(LB_Q, \text{Max}\left(Q_{i \;\; \forall i=1,2,..N_{alg}}\right)\right)$;
    $UB_{div} \leftarrow \text{Max}\left(UB_{div}, \text{Max}\left(div_{i \;\; \forall i=1,2,..N_{alg}}\right)\right)$;
  end if
  if $cy = 2 \times CS$ **then**
    $cy \leftarrow 0$;
  end if
end while

Considering diversity, the lower bound (LB_{div}) is zero and, similar to (LB_Q), the upper bound (UB_{div}) is automatically updated in every CS generations, with its initial value set to $UB_{div} = \max(div_i) \, \forall \, i = \{1, 2, ...N_{alg}\}$. Bearing in mind, if the diversity at any stage is increased, UB_{div} is updated as $\max\left(UB_{div}, \;\; \max(div_i) \, \forall \, i = \{1, 2, ...N_{alg}\}\right)$. The steps for updating the lower bounds of both the diversity and quality of solutions are shown in Algorithm 2.

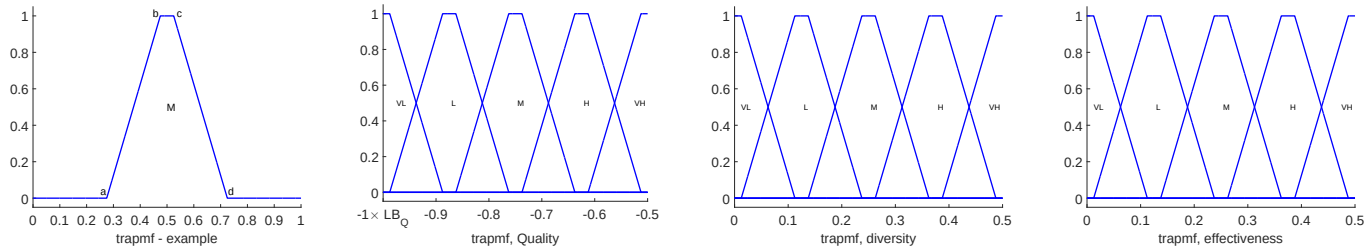
Also, the trapezoidal membership function is considered to map each point in the input space to a membership value. The trapezoidal curve of a point y depends on four scalar values, a, b, c, and d, such that

$$\mu_y(s) = \begin{cases} 0 & y < a \\ \frac{y-a}{b-a} & a \leq y \leq b \\ 1 & b \leq y \leq c \\ \frac{d-y}{d-c} & c \leq y \leq d \\ 0 & y \geq d \end{cases} \tag{4}$$

where a, b, c, and d of each level for the inputs (quality, diversity) and output (effectiveness) are shown in Figure 2.

Once the fuzzy sets are defined, the if-then rules are used to codify the conditional statements that encompass fuzzy logic and formulate the effectiveness based on the inputs (diversity and quality), as shown in Table I; for instance, ***if*** quality is high and diversity low, ***then*** the effectiveness of an OA is medium.

As described in section II-A, an if-then rule involves two different processes: (1) evaluating the antecedent (which involves fuzzifying the input and applying any necessary



(a) a, b, c, and d parameters for the "M" level (b) Membership function for quality factor (c) Membership function for diversity factor (d) Membership function for effectiveness factor

Figure 2. Membership function for inputs (quality, diversity) and output (effectiveness)

Table I. IF-THEN RULES

	levels	Quality				
		VL	L	M	H	VH
Diversity	VL	VL	VL	L	M	H
	L	VL	L	M	M	H
	M	L	M	H	H	VH
	H	M	M	H	VH	VH
	VH	H	H	VH	VH	VH

fuzzy operators); and (2) applying that result to the consequent (known as implication). Note that (1) the logical AND is represented as the minimum of the membership values $(prob\,(O_1 \text{ AND } O_2) = \min\,(p(O_1), p(O_2)))$; (2) the implication function, which modifies that fuzzy set to the degree specified by the antecedent, is the min function; (3) the output fuzzy sets for each rule are aggregated into a single output fuzzy set, using the max function; and (4) the defuzzification technique used is the centroid method. For a trapezoidal fuzzy number, the centroid can be calculated as: $\bar{s}(y) = \frac{\int_a^d s \times \mu_y(s) ds}{\int \mu_y(s) ds} = \frac{1}{3}\left[a + b + c + d - \frac{d \times c - a \times b}{(d+c)-(a-b)}\right]$[62].

The abovementioned steps are carried out for every OA, with the value obtained after defuzzification $(\in [0, 1])$ considered its effectiveness, or simply called $prob_i \, \forall i = \{1, 2, .., N_{alg}\}$.

B. Information sharing

A team of OAs without trust is not really an effective team, that may not perform consistently well. One way to build the trust is to share vital information among them in an effective manner. To perform this crucial task, a few questions need to be answered: (1) among which algorithms should we share information? (2) when to share information? (3) how to share information? and (4) what information we should share?

Regarding the first question, a trusted algorithm can pass information to inferior ones as it has above-average effectiveness (as described in III-A), with an algorithm considered inferior if its effectiveness is below average, i.e., $(prob < 0.5)$. The reason for avoiding sharing information among well-performing algorithms is to give them space to search independently as long as they continue to perform well.

As, for the same reason, sharing information should be carried out periodically rather than in each generation [17], it is conducted at the end of the second cycle ($cy = 2 \times CS$ in Algorithm 1).

Considering the third and fourth questions, a simple way of sharing information is to replace the worst individual in an inferior algorithm's sub-population by the best one in that of a trusted algorithm. However, this does not add any benefit to CMA-ES if it is used and found to be among the inferior algorithms. To clarify this, the main elements in CMA-ES are $\overrightarrow{x}_m$, σ_t and $N\,(0, C_t)$, as described later.If we replace one individual in its sub-population by the best individual in that of the trusted algorithm without updating one of these factors based on the new information, CMA-ES will evolve according to its previous values. Therefore, in this paper, information sharing for CMA-ES is considered a process of restarting its individuals, i.e., replacing its entire sub-population by random individuals from that of the trusted algorithm. Also, its parameters are reset to their initial values, except that σ is updated as $\sigma = \sigma_{initial} \times \left(1 - \frac{cfe}{FFE_{max}}\right)$ to avoid huge perturbations that may occur.For any algorithm in the group of inferior algorithms, its worst individual is substituted by the best one in the trusted algorithm's sub-population.

C. Fine-tuning technique

In this research, a fine-tuning process is adopted which applies a local search procedure during the final stages of the optimization process, i.e., in each generation of the final 15% of the evolutionary process, with a probability of $prob_{ls}$, sequential quadratic programming (SQP) [5] applied to the best individual found so far and for up to cf e LS fitness evaluations. However, to avoid applying SQP without obtaining any benefit, $prob_{ls}$ is dynamically changed based on SQP's performance. To clarify this, if SQP does not successfully obtain a better result, $prob_{ls}$ is set to a small value, otherwise to its initial one.

The solution obtained by SQP is shared with all the algorithms by replacing the worst individual in the sub-population. For CMA-ES, if its effectiveness is below average, its parameters are reset to their initial values, except thatσ is set to $\sigma_{initial} \times \left(1 - \frac{cfe}{FFE_{max}}\right)$.

D. OAs

Research studies have indicated that a diverse group, that is, its members complement each other, can provide the benefit of generating good decision-making alternatives [31]. A diverse group means that the group members should complement each other. Although the framework proposed in this paper is general and can be applied to any diverse group of algorithms, two $(N_{alg} = 2)$ powerful algorithms considered are (1) multi-operator DE (MODE) which combines three DE mutation operators to overcome the shortcoming that one may work well for one problem but not another as such algorithms having proven their capability to perform well [17, 49], and (2) CMA-ES.

1) MODE : MODE starts with PS_1 individuals which are randomly taken from the entire PS individuals. It uses three DE variants: (1) DE_1: current-to-$p$best/bin with archive; (2) DE_2: current-to-$p$best/bin without archive; and (3) DE_3: wighted-rand-to-ϕbest/bin. Descriptions of these operators are give in the supplementary materials.

Firstly, any individual (x_z) in X_1 can be evolved using DE_1 or DE_2 or DE_3 with a predefined probability, i.e., $Pr_{DE_1} = Pr_{DE_2} = Pr_{DE_3} = \frac{1}{3}$. To clarify, if $rand_z \in [0,1] \leq 0.33$, then $\overrightarrow{x}_z$ is evolved using DE_1, if $0.33 \leq rand_z \in [0,1] \leq 0.667$, use DE_2, otherwise DE_3 is used. Then, each probability is updated based on the fitness improvements rate (I_{DE_κ}) achieved by each DE:

$$I_{DE_\kappa} = \frac{\sum_{z=1}^{PS_1} max\left(0, f_{new_z} - f_{old_z}\right)}{\sum_{z=1}^{PS_1} f_{old_{z,\kappa}}}, \quad \forall \overrightarrow{x}_\kappa \text{ updated by } DE_\kappa \text{ and } \kappa = 1,2,3 \quad (5)$$

where f_{new} and f_{old} are the new and old fitness values, respectively.

Then, each probability is updated as

$$Pr_{DE_\kappa} = \max\left(0.1, \min\left(0.9, \frac{I_{DE_\kappa}}{I_{DE_1} + I_{DE_2} + I_{DE_3}}\right)\right), \quad \forall \kappa = 1,2,3 \quad (6)$$

Note that, as one operator may perform good at different stages of the evolutionary process, and perform badly in others, a minimum value of $prob_{DE_\kappa}$ is considered [17]. Furthermore, if $\left(\sum_{\kappa=1}^{3} I_\kappa\right) = 0$, $Pr_\kappa = \frac{1}{3} \forall i = 1,2,3$.

Also, for maintaining diversity within the early evolutionary process, while enhancing the exploitation ability in later ones [54], a linear reduction of PS_1 is carried out at the end of each generation by removing the worst individual, such that

$$PS_{1,t+1} = \text{round}\left(\left(\left(\frac{PS_{1,min} - PS_{1,max}}{FFE_{max}}\right) \times cfe\right) + PS_{1,max}\right) \quad (7)$$

where $PS_{1,max}$ and $PS_{1,min}$ are the maximum and minimum values of PS_1, respectively, and FFE_{max} the maximum number of fitness evaluations.

Adaptation of F and Cr : In this paper, the mechanism proposed in [54] is adopted. Its main idea is to use two historical memories for F and Cr which showed good performance during the previous generations. Then, new ones are generated by sampling around those stored in the memory. Details about such a mechanism are discussed in supplementary materials.

2) CMA-ES: Over the last two decades, CMA-ES has shown its capability to efficiently solve diverse types of optimization problems [28]. It was derived from the concept of self-adaptation in ES, which adapts the covariance matrix of a multivariate normal search distribution. In it the new individuals are generated by sampling from a Gaussian distribution, and instead of using a single mutation step, it considers the path the population takes over generations [29]. The main steps in CMA-ES used in this paper are described in the supplementary materials.

IV. Experimental Results

This section presents, discusses and analyzes the computational results obtained by TOAs on a set of unconstrained problems (the CEC2014 problems) [33]. Then, it shows the results of testing TOAs on 28 problems taken from the CEC2013 problems [34], 25 ones from the CEC2005 benchmark [53], and other 12 classical unconstrained ones [67][1].

A. Analysis of TOAs on CEC2014 problems

The benchmark consists of 30 test problems, i.e., $F_{set} = \{1,2,..,30\}$, with $F_1 \sim F_3$ are uni-modal, $F_4 \sim F_{16}$ multi-modal, $F_{17} \sim F_{22}$ hybrid, and $F_{23} \sim F_{30}$ composite functions.

All the algorithms were run 51 times for each test problem, and the stopping criterion $10,000$D fitness evaluations, with D = 10, 30, and 50, or $\left|f\left(\overrightarrow{x}_{best}\right) - f\left(\overrightarrow{x^*}\right)\right| \leq 1e{-}08$, where $f\left(\overrightarrow{x^*}\right)$ is the optimal solution. Note that all the algorithms started with the same initial population, which changed in each run.

For MODE, $PS_{1,max}$ was 18D individuals and $PS_{1,min}$ 4, $H = 6$ [54]. For CMA-ES, $PS_2 = 4 + \lfloor(3log\left(D\right))\rfloor$ [28], $\mu = \frac{PS}{2}$ and $\sigma = 0.3$. CS was set to 50, 100 and 150, for the 10, 30, and 150D problems, respectively and cfe_{LS} to $0.2 \times FFE_{max}$ fitness evaluations.

The non-parametric Wilcoxon test [10] was carried out to determine if there was a statistical difference between TOAs and the other algorithms based on two options: (1) each problem (51 results); and (2) a whole set of problems, i.e., 30 results, each of which is the average of 51 runs. Using a significance level of 5%, one of three symbols ($+$, $-$ and $\approx$) was used, where $+$, $-$ and $\approx$ meant TOAs was statistically superior, inferior and similar to the other algorithm, respectively. Also, the Friedman test was undertaken to rank all the algorithms based on their average fitness errors. In addition, to visually compare the results, the performance profiles tool was considered [2]. To use such a technique, a goal had to be defined, which, in this paper, was the average fitness error obtained obtained.

1) 10D Results: The average fitness errors of the best solutions compared with the optimal ones and standard deviation results are presented in Table 1 in the supplementary materials. Note that, if $\left|f\left(\overrightarrow{x}_{best}\right) - f\left(\overrightarrow{x^*}\right)\right| \leq 1e-08$, the value is considered 0.

Firstly, among 30 problems, the algorithm was able to obtain optimal solutions for all the uni-modal and 7 multi-modal ones, fitness errors very close to zero for the remaining 6 and performed well in solving the hybrid functions. However, for the complex composition functions, it converged to local solutions.

Based on the quality of solutions obtained, it was clear that TOAs was always able to obtain better results than both MODE and CMA-ES. To clarify, considering the best solutions achieved, TOAs was better than, equal to and worse than MODE and CMA-ES for 17, 12 and 1, and 23, 6 and 1 problems, respectively. Regarding the average results, neither MODE nor CMA-ES outperformed TOAs for any problem; in particular, TOAs was superior and similar to MODE for 22

[1] due to the number of pages limitation, some results and figures are moved to the supplementary materials attached with the paper

and 8 test problems, respectively, and to CMA-ES for 27 and 3, respectively.

Based on the Wilcoxon test, it was found that TOAs was statistically better than MODE and CMA-ES for 13 and 26 test problems, respectively, with no significant difference between them for the remaining 17 and 4, respectively. Furthermore, when this test was conducted considering the second option previously mentioned, the results showed that TOAs was statistically superior to both these algorithms. Also, the Friedman test was carried out to rank all the algorithms, with the results demonstrating that TOAs was 1^{st}, followed by MODE and CMA-ES.

Finally, using the performance profiles graphical tool, it was clear that TOAs was the best as it was able to reach a probability of 1.0 with a value of $\tau = 1$, as depicted in Figure $1.a$ in the supplementary materials.

2) 30D Results: Generally, the TOAs was able to obtain optimal solutions for the $F_1 \sim F_3$ problems. For the multi-modal functions, TOAs was able to attain the optimality in F_4 and $F_6 \sim F_{10}$. The best solutions obtained for $F_{12} \sim F_{15}$ were very close to the optimal ones, but the algorithm became stuck in local solutions when solving F_5 and F_{11}. For the hybrid functions, although the best solutions obtained for $F_{18} \sim F_{22}$ were close to 0, the best for F_{17} was slightly worse. This was also noticeable for the average results obtained for F_{17} and F_{21}. For the composition functions, although TOAs was not able to reach optimality, its fitness errors, especially for $F_{23} \sim F_{28}$, were reasonably close to 0.

In comparison with the other two algorithms, TOAs continued to perform well, as evident from the results presented in Table 2 in the supplementary materials, with the summary provided in Table II showing that it was clearly better than the other algorithms for the majority of test problems.

Table II. COMPARISON SUMMARY OF TOAS AGAINST MODE AND CMA-ES FOR 30D PROBLEMS

Algorithms	Criteria	Better	Similar	Worse
TOAs vs. MODE	**Best fitness values**	12	13	5
	Average fitness values	16	12	2
TOAs vs. CMA-ES	**Best fitness values**	22	7	1
	Average fitness values	26	3	1

However, it was crucial to check whether TOAs was statistically inferior to the other algorithms for the problems for which it obtained slightly worse results. To do this, the Wilcoxon test was used, with the results showing that no statistical difference could be found between TOAs and those algorithms for those problems. The Wilcoxon test was also carried out based on option 2 previously mentioned, with the results demonstrating that it was statistically superior to MODE and CMA-ES for solving the 30D test problems. Again, considering the Friedman test, TOAs was ranked 1^{st}, with MODE and CMA-ES 2^{nd} and 3^{rd}, respectively.

Based on the plot generated by the performance profiles, depicted in Figure $1.b$ in the supplementary materials, TOAs was found to be the best, followed by MODE and CMA-ES.

3) 50D Results: From the results presented in Table 3 in the supplementary materials, regarding the quality of solutions obtained, it was noted that TOAs was able to reach optimal solutions for 2 uni-modal problems and very close to optimality for the remaining 1 (F_{01}). For the multi-modal problems, it achieved optimality for 3, very close to 0 for 7, close to optimality for F_5 and F_{16} and poor results for only F_{11}, with its performances for solving hybrid functions reasonable. For the composition functions, it was able to converge to a local solution which was close to optimality but its average result for F_{30} was far from the global solution.

As the complexity of a problem increases with an increasing number of decision variables, it was expected that TOAs would not perform as well as it did for smaller-dimensional problems but would be better than, or similar to, both MODE and CMA-ES. This was achieved for the majority of test problems as it was able to obtain better results (based on the best fitness values) than MODE and CMA-ES for 16 and 22 problems, respectively, similar results for 10 and 6, respectively, and inferior for only 4 and 2, respectively. Based on the average fitness values, TOAs was superior, similar and inferior to MODE for 16, 9 and 5, test problems, respectively, and to CMA-ES for 26, 2 and 2, respectively.

As TOAs performed worse than the other algorithms for a few test problems, it was vital to check their statistical differences. Based on the recorded results, it was found that TOAs was statistically superior or similar to the other algorithms for the majority of test problems. Unfortunately, although the differences between average results were not large, it was statistically inferior to MODE in 2 problems (F_{10}, F_{16}), and to CMA-ES for (F_1, F_{29}).

As it was important to understand this drawback, an investigation showed that the population size of MODE was the main cause, i.e., MODE's population size (PS_1) was 18D=900 individuals while CMA-ES used $4 + \lfloor 3log\,(\text{D}) \rfloor = 15$, as suggested in [28], which affected the convergence rate of TOAs. We noticed that CMA-ES converged very slowly for F_{29} during the early generations and then improved during the later stages of the optimization process, as shown by the average fitness errors of both MODE and CMA-ES for in Figure 3 in the supplementary materials. This meant that TOAs always preferred using MODE in every decision step, and re-initialized CMA-ES due to its poor performance.

For those problems affected by the first reason, i.e., a large PS_1, we easily managed that by setting its value to D for the uni-modal problem (F_1), 3D for F_{10} and 6D for F_{16}. To compare algorithms in a fair manner, both MODE, and TOAs with the new initial PS were run for 51 runs, and their results recorded, as presented in Table III. It is clear now that TOAs was able to obtain better, or same, fitness errors to those of the other algorithms. Furthermore, TOAs was always statistical superior, or similar to the other algorithms.

Moreover, from the Wilcoxon test based on the second option, TOAs was statistically superior to both algorithms and ranked 1^{st} based on the Friedman test, with MODE and CMA-ES 2^{nd} and 3^{rd}, respectively.

Unfortunately as, for F_{29}, for which CMA-ES outperformed TOAs, the population size was not a remedy, further investigations will be required.

Finally, the performance profiles tool was used to graphically compare all the algorithms, as shown in Figure $1.c$ in the supplementary materials. The plots show that TOAs was in first place, as it achieved a probability of 1.0 at $\tau = 0.2 \times 10^5$ whereas until $\tau = 2 \times 10^5$, neither MODE nor CMA-ES was able to reach $Rho = 1$.

4) Computational Times: In this subsection, the computational times of TOAs, MODE and CMA-ES are compared.

For each method, the average computational times taken to solve all the test problems were calculated if one of the following two criteria was met: (1) the maximum number of fitness

Table III. FITNESS ERRORS $\left(\left|f\left(\overrightarrow{x}_{best}\right)-f\left(\overrightarrow{x^*}\right)\right|\right)$ OBTAINED BY MODE, CMA-ES AND TOAS FOR F_1, F_{10} AND F_{16} WITH 50D (BETTER MEAN FITNESS ERRORS SHOWN IN BOLDFACE)

Prob.	Best fitness errors			mean (std.) fitness errors			Stat. Test (p,Dec.)	
	MODE	CMA-ES	TOAs	MODE	CMA-ES	TOAs	MODE	CMA-ES
F_1	2.5519E+03	**0.0000E+00**	**0.0000E+00**	1.7806E+04(2.0051E+04)	**0.0000E+00** (0.000E+00)	2.0663E-04 (1.476E-03)	0.0000(+)	1.000(≈)
F_{10}	**0.0000E+00**	6.0102E+03	**0.0000E+00**	**2.4493E-04**(1.749E-03)	7.7879E+03 (7.233E+02)	1.2247E-03 (3.752E-03)	0.2188(≈)	0.0000(+)
F_{16}	1.5494E+01	2.1740E+01	**1.5000E+01**	**1.6567E+01**(4.689E-01)	2.2693E+01 (5.590E-01)	1.6615E+01 (4.650E-01)	0.4647(≈)	0.0000(+)

evaluations was reached; or (2) $\left|f\left(\overrightarrow{x}_{best}\right)-f\left(\overrightarrow{x^*}\right)\right| \leq 1e-08$. Note that all the experiments were run on a PC with a Core(TM) i7-3770 CPU @ 3.40GHz (8 CPUs), 16 GB RAM and Windows 7 using MATLAB 8.5.0.197613 (R2015a).

Based on the results presented in Table IV, MODE was the fastest algorithm, consuming only slightly less time than TOAs but significantly less than CMA-ES. In fact, it was expected that TOAs would be slower than MODE as it used CMA-ES in its process which is computationally expensive. Generally, as the solutions obtained by TOAs were significantly better than those by MODE, this small increase in the computational time can be ignored.

Table IV. AVERAGE COMPUTATIONAL TIMES, IN SECONDS, FOR TOAS, MODE AND CMA-ES FOR 10D,30D AND 50D

	CMA-ES	MODE	TOAs
10D	7.97E+00	**2.72E+00**	3.45E+00
30D	3.15E+01	**1.06E+01**	1.26E+01
50D	8.07E+01	**2.93E+01**	3.10E+01

5) Benefits of using Complementary Selection Characteristics : In this subsection, we analyze the benefits of using complementary criteria to determine the effectiveness of an algorithm. To do this, TOAs was run by setting the selection method to only (1) the quality of solutions (var_1), (2) diversity (var_2), and (3) random (var_3). All the variants were compared based on only the 30D problems, with the summary presented in Table V demonstrating the benefits of using complementary selection characteristics.

Table V. COMPARISON SUMMARY OF TOAS AGAINST var_1 FOR 30D PROBLEMS (WHERE STATE. TEST REFERS TO RESULTS BASED ON WILCOXON TEST)

Algorithms	Criteria	Better	Similar	Worse	Stat. test
TOAs vs. var_1	**Best fitness values**	11	13	6	0.356(≈)
	Average fitness values	15	11	4	0.022(+)
TOAs vs. var_2	**Best fitness values**	12	13	5	0.019(+)
	Average fitness values	16	10	4	0.001(+)
TOAs vs. var_3	**Best fitness values**	12	13	5	0.136(≈)
	Average fitness values	19	10	1	0.001(+)

Considering the Wilcoxon test, although no statistical difference was found between TOAs and both var_1 and var_3 regarding the best fitness values achieved. TOAs was statistically better considering the average results and always statistically superior to var_2. Also, the Friedman test ranked TOAs 1^{st} with a mean rank of 1.95, and var_1, var_2, and var_3 came 2^{nd}, 3^{rd} and 4^{th} with scores of 2.18, 2.80 and 3.07, respectively. The performance profiles method also produced consistent results, as TOAs was able to reach a probability of 1 first with $\tau \approx 2.85$ (Figure 2 in the supplementary materials).

6) Effect of CS: In this section, the effect of CS is analyzed by running TOAs with different CS values (i.e., 10, 50, 100, 150) to solve the 10, 30 and 50D problems. Subsequently, the Friedman test was carried out to rank all variants, with a summary given in Table VI. Based on the results obtained, it was noticed that it would be good to increase CS with the increase of dimensionality, i.e., CS=50, 100 and 150 were the best for the 10, 30 and 50D problems, respectively.

Table VI. RANKS OF TOAS WITH WITH DIFFERENT CS VALUES BASED ON FRIEDMAN TEST

CS	10D	30D	50D
10	2.85	2.85	2.70
50	**2.23**	2.75	2.83
100	2.55	**2.02**	2.40
150	2.37	2.36	**2.07**

7) Effect of LS: To analyze the effect of LS, MODE and CMA-ES (with and without LS) were run to solve the 30D test problems and compared to the proposed algorithm with and without LS. A comparison summary is given in Table VII, with the results showing that the proposed algorithm was the best even when LS was incorporated into the opponent algorithms.

Table VII. COMPARISON AMONG ALGORITHMS WITH AND WITHOUT LS FOR THE 30D PROBLEMS

Algorithms	Better	Similar	Worse
TOAs without LS vs. MODE	13	11	6
TOAs vs. MODE with LS	13	11	6
TOAs without LS vs. CMA-ES	25	3	2
TOAs vs. CMA-ES with LS	26	3	1

Also, the Friedman test was carried out to rank all six variants. The results showed that TOAs with LS was the best followed by MODE with LS, TOAs without LS, MODE without LS, CMA-ES with LS and CMA-ES without LS, with scores, 2.03, 2.63, 2.93, 3.27, 5.03 and 5.1, respectively. Generally, incorporating LS into any of the abovementioned variants was better than the same without LS.

8) Comparison with State-of-the-art Algorithms: TOAs was compared with two well-known algorithms, (1) LSHADE [54] and (2) UMOEAs [16], with the detailed results shown in Tables 4 and 5 in the supplementary materials.

Based on the quality of solutions, a comparison summary is presented in Table VIII, which indicates that TOAs was the best for the majority of test problems. Also, it was observed that all the algorithms were able to obtain optimal solutions for all the uni-modal problems, except the 50D ones for which LSHADE could not achieve it for F_1. Also, Also, for the multi-modal problems, TOAs showed its superiority to the other algorithms, except for F_{12} and F_{13}, for which UMOEAs was slightly better. TOAs also performed well for the majority of hybrid functions, and for most the composition functions, was able to obtain significantly better results than all the other algorithms.

Statistically, the Wilcoxon test showed that TOAs was significantly better than LSHADE and UMOEAs for all dimensions. Furthermore, Table IX shows the ranking of each

algorithm according to the Friedman test in which it is clear that TOAs was ranked 1^{st} for all dimensions while UMOEAs was better than LSHADE for the 10D ones, but had the lowest rank for all other dimensions.

Table IX. RANKS OF ALL ALGORITHMS BASED ON FRIEDMAN TEST

Algorithms	10**D**	30**D**	50**D**
LSHADE	2.37	2.22	2.13
UMOEAs	2.22	2.33	2.43
TOAs	1.42	1.45	1.43

Finally, the performance profiles graphical tool showed consistent conclusions for all dimensions, as depicted in Figure 4 in the supplementary materials.

9) Comparison with other Algorithms: In this section, TOAs was compared with other 4 well-known algorithms, (1) DE with self-adaptation of its control parameters (jDE) [6]; (2) DE with an ensemble of parameters and mutation strategies (EPSDE) [39]; (3) DE with composite trial vector generation strategies (CoDE) [63]; and (4) success-history parameter adaptation of DE (SHADE) [55]. The results were taken from [15]. Due to the number of pages limitation, the comparison was conducted only on the 30D problems, with the average fitness errors reported in Table 6 in the supplementary materials. Regarding the quality of solutions, a comparison summary is given in Table X, where the non-parametric Wilcoxon test was carried based on option 2 previously discussed. The results clearly confirm the superiority of the proposed algorithm. Furthermore, the Friedman test ranked TOAs 1^{st} with a mean rank of 1.28, as reported in Table XI. Also, the performance profile tool showed the superiority of TOAs, as depicted in Figure 5 in the supplementary materials.

Table X. COMPARISON SUMMARY OF TOAS AGAINST JDE, EPSDE, CODE AND SHADE ON 30D PROBLEMS (DEC. STATISTICAL DECISION TAKEN BASED ON WILCOXON TEST)

Algorithms	30**D**			
	Better	**Similar**	**Worse**	**Dec.**
TOAs vs. jDE	26	4	0	+
TOAs vs. EPSDE	25	5	0	+
TOAs vs. CoDE	29	0	1	+
TOAs vs. SHADE	25	4	1	+

Table XI. RANKS OF ALL ALGORITHMS BASED ON FRIEDMAN TEST

jDE	EPSDE	CoDE	SHADE	TOAs
3.08	3.73	4.35	2.55	**1.28**

B. Testing TOAs on additional benchmark problems

In this section, three other benchmark sets are solved, taken from the CEC2013 and CEC 2005 special sessions on real-parameter optimization [34][53], and 12 classical ones [67] described in Table 7 in the supplementary materials. The performance of TOAs was evaluated against several state-of-the-art algorithms based their capability of obtaining high-quality solutions, non-parametric Wilcoxon test, their rankings based on the Friedman test and the performance profile tool.

1) CEC2013 problems: 28 problems were solved with each ran 51 times with $D = 30$ variables and the stopping criterion $10,000$D or the fitness error between the best and optimal solutions reached $1\text{E}-08$. The results obtained by TOAs were compared with those of other 8 algorithms, (1) DE with an individual-dependent mechanism (IDE) [57]; (2) JADE with eigenvector-based mutation (JADE/eig) [27]; (3) DE with an evolution path (JADEEP) [32]; (4) CoDE; (5) EPSDE; (6) dynNP_jDE [7]; (7) SHADE; and (8) collective information-powered DE (CIPDE)[73]. Note that the results of the first three algorithms and CIPDE were taken from their corresponding papers, while the rest from [55]. The average fitness errors of all algorithms are shown in Table 8 in the supplementary materials. The comparison summary presented in Table XII demonstrates that TOAs was able to achieve the best results for the majority of test problems. Also, TOAs was statistically better than all the other algorithm, and ranked 1^{st} based on the Friedman test. In addition, the performance profile tool clearly showed the superiority TOAs to all the other algorithms, as depicted in Figure 6 in the supplementary materials.

Table XII. COMPARISON SUMMARY OF TOAS AGAINST 8 STATE-OF-THE-ART ALGORITHMS ON CEC2005 UNCONSTRAINED PROBLEMS WITH 30 VARIABLES

Algorithms	30**D**			
	Better	**Similar**	**Worse**	Stat. Test (p,Dec.)
TOAs vs. IDE	22	3	3	$(0.001, +)$
TOAs vs. JADE/eig	24	4	0	$(0.000, +)$
TOAs vs. JADEEP	22	4	2	$(0.000, +)$
TOAs vs. CoDE	21	5	2	$(0.000, +)$
TOAs vs. EPSDE	25	3	0	$(0.000, +)$
TOAs vs. dynNP_jDE	21	4	3	$(0.000, +)$
TOAs vs. SHADE	21	4	3	$(0.000, +)$
TOAs vs. CIPDE	21	4	3	$(0.000, +)$

Table XIII. RANKS OF ALL ALGORITHMS BASED ON FRIEDMAN TEST BASED ON CEC2013 PROBLEMS

IDE	JADE/eig	JADEEP	CoDE	EPSDE	dynNP_jDE	SHADE	CIPDE	TOAs
4.54	6.02	4.98	5.39	7.89	5.39	4.39	4.25	**2.14**

2) CEC2005 problems: In this subsection, considering the CEC2005 problems, TOAs is evaluated against (1) self-adaptive DE with discrete mutation control parameters (DMPSADE) [24]; (2) JADE with auto-enhanced population diversity (AEPD-JADE) [65]; (3) DE with a hybrid mutation operator and self-adapting control parameters (HSDE) [68]; (4) efficient player selection strategy based diversified PSO ((EPS-dPSO) [1]; (5) a trajectory-based centralized cooperative strategy based on an approaching action (TCCS-AC) [40]; (6) DE with dynamic parameters selections (DE-DPS) [49]; (7) CoDE. Each algorithm was run 25 times, except DMPSADE ran 50 times, with $D = 30$ variables. The average fitness errors and standard deviation values were taken from their corresponding papers and compared to those of TOAs. The detailed results are shown in Table 9 in the supplementary materials, with a comparison summary presented in XIV. The results demonstrated that TOAs outperformed all the other algorithms for the majority of test problems. Statistically speaking, ToAs was better than all the other algorithms, except DE-DPS, as both algorithms were statistically similar. However, if we conduct the Wilcoxon test with a significance level of 10%, ToAs will statistically outperform DE-DPS.

Table VIII. COMPARISON SUMMARY OF TOAS AGAINST LSHADE AND UMOEAS (DEC. STATISTICAL DECISION TAKEN BASED ON WILCOXON TEST)

Algorithms	Criteria	10D				30D				50D			
		Better	Similar	Worse	Dec.	Better	Similar	Worse	Dec.	Better	Similar	Worse	Dec.
TOAs vs. LSHADE	**Average fitness values**	25	4	1	+	21	6	3	+	22	3	5	+
TOAs vs. UMOEAs	**Average fitness values**	18	5	7	+	20	5	5	+	22	3	5	+

Considering the mean rank of each algorithm calculated by the Friedman test, TOAs was the best, as reported in Table XV. The performance profile tool also showed the superiority of TOAs (Figure 7 in the supplementary materials).

Table XIV. COMPARISON SUMMARY OF TOAS AGAINST DMPSADE, AEPD-JADE, HSDE, EPS-DPSO, TCCS-AC, DE-DPS AND CODE ON CEC2005 UNCONSTRAINED PROBLEMS WITH 30 VARIABLES

Algorithms	30D			
	Better	Similar	Worse	Stat. Test (p,Dec.)
TOAs vs. DMPSADE	17	7	1	$(0.002, +)$
TOAs vs. AEPD-JADE	20	3	2	$(0.001, +)$
TOAs vs. HSDE	17	5	3	$(0.008, +)$
TOAs vs. EPS-dPSO	19	4	2	$(0.000, +)$
TOAs vs. TCCS-AC	20	3	2	$(0.001, +)$
TOAs vs. DE-DPS	14	7	4	$(0.052, \approx)$
TOAs vs. CoDE	19	5	1	$(0.000, +)$

Table XV. RANKS OF DMPSADE, AEPD-JADE, HSDE, EPS-DPSO, TCCS-AC AND TOAS BASED ON FRIEDMAN TEST ON CEC2005 PROBLEMS

DMPSADE	AEPD-JADE	HSDE	EPS-dPSO	TCCS-AC	DE-DPS	CoDE	TOAs
4.90	6.12	4.34	5.62	5.46	3.22	4.06	**2.28**

3) 12 *classical problems:* In this subsection, TOAs is evaluated against 5 of the-state-of-the-art algorithms, (1) jDE; (2) DE with neighborhood search (NSDE) [66]; (3) Opposition-based DE(ODE) [46]; (4) DEGL with self-adaptive weight factor (DEGL/SAW) [11]; and (5) Gaussian bare-bones DE (MGBDE) [61]. For each problem, every algorithm was run 50 times with $D = 25$ variables. The results of these algorithms were taken from Table V in [61]. Based on the results obtained (Table 10 in the supplementary materials), a comparison summary is presented in XVI, with the results showing that TOAs was always better than, or similar to, all the algorithms, except DEGL/SAW, in which, for only two problems, TOAs was slightly inferior. Also, TOAs was statistically better than all the algorithms, except DEGL/SAW, in which both algorithms were statistically similar. The rank of TOAs was 1.56 which put it in the 1st place, as presented in Table XVII. Regarding the performance profile tool, Figure 8 in the supplementary materials demonstrated that TOAs had the capability of outperforming all other state-of-the-art algorithms considered. TOAs was also run with $D = 30$ and showed better performance compared with other two algorithms, with more details given in Tables 11 and 12 in the supplementary materials.

Table XVI. COMPARISON SUMMARY OF TOAS AGAINST JDE, NDE, ODE, DEGL/SAW AND MGBDE ON 12 UNCONSTRAINED PROBLEMS WITH 25 VARIABLES

Algorithms	Better	Similar	Worse	Stat. Test (p,Dec.)
TOAs vs. jDE	11	1	0	$(0.003, +)$
TOAs vs. NSDE	11	1	0	$(0.003, +)$
TOAs vs. ODE	9	3	0	$(0.008, +)$
TOAs vs. DEGL/SAW	9	1	2	$(0.248, \approx)$
TOAs vs. MGBDE	8	4	0	$(0.012, +)$

Table XVII. RANKS OF ALL ALGORITHMS BASED ON FRIEDMAN TEST

jDE	NSDE	ODE	DEGL/SAW	MGBDE	TOAs
4.42	5.00	4.04	3.00	2.96	**1.58**

V. CONCLUSION AND FUTURE WORK

Many EAs and SI methods for solving optimization problems have been introduced. As no single optimization method has proven to be the best for all types of problems, researchers have started developing frameworks that use a mix of algorithms and/or operators. Although these frameworks have shown more success than single-based methods or operators, their designs were based on trial and error approaches. Also, their performances might be statistically outperformed by those of single-based methods.

Therefore, based on organizational behavior, this research study introduced a set of rules for designing such frameworks. As a consequence, a new one was proposed which could be considered a step toward better designs of teams of OAs. Our TOAs was constructed using powerful and complementary OAs and then, based on complementary search characteristics, a fuzzy rules system was used to place emphasis on the best-performing one. Also, based on the effectiveness of this algorithm, an information-sharing scheme was implemented with a fine-tuning procedure used in the latter stages of the optimization search process.

This framework was adopted with MODE and CMA-ES as OAs, and SQP as a fine-tuning procedure. Then, TOAs was used to solve a set of real-parameter benchmark problems with 10, 30 and 50 dimensions. The results showed that it was 100% successful in obtaining statistically better, or similar, results to its individual algorithms for the 10D and 30D problems while achieving the same performance for 29 of 30 of the 50D ones. The results were also compared in terms of the quality of solutions using the Friedman test and performance profiles tool, both of which showed the superiority of the proposed method. Furthermore, TOAs was statistically competitive with different state-of-the-art algorithms. TOAs was also evaluated on the CEC2013, CEC2005 and 12 standard problems and was found better than the state-of-the-art algorithms.

Generally, this paper can offer new directions for designing multi-operator and multi-method frameworks using a single population rather than multiple sub-populations, and dynamically selecting their parameters although any new algorithms should be carefully developed. Another vital future work is developing a remedy for the shortcoming encountered when solving F_{29} (in the CEC2014 benchmark) with 50 dimensions. Also, extending the proposed fuzzy system for constrained problems will be beneficial. The idea of a trusted algorithm can be adapted to trusted operators and or parameters, which may be an interesting future direction to explore.

ACKNOWLEDGMENT

This research is supported by the Australian Research Council Discovery Project DP150102583 awarded to R. Sarker and CONACyT project no. 221551 awarded to C. Coello Coello.

REFERENCES

[1] Prativa Agarwalla and Sumitra Mukhopadhyay. Efficient player selection strategy based diversified particle swarm optimization algorithm for global optimization. *Information Sciences*, 397:69–90, 2017.

[2] Helio JC Barbosa, Heder S Bernardino, and André Barreto. Using performance profiles to analyze the results of the 2006 cec constrained optimization competition. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[3] Emer Bernal, Oscar Castillo, José Soria, and Fevrier Valdez. Imperialist competitive algorithm with dynamic parameter adaptation using fuzzy logic applied to the optimization of mathematical functions. *Algorithms*, 10:18, 2017.

[4] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. *F-Race and Iterated F-Race: An Overview*, pages 311–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[5] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[6] Janez Brest, Saso Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[7] Janez Brest and Mirjam Sepesy Maučec. Population size reduction for the differential evolution algorithm. *Applied Intelligence*, 29(3):228–247, 2008.

[8] Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. *International series in operations research and management science*, pages 457–474, 2003.

[9] Tae Jong Choi and Chang Wook Ahn. An adaptive cauchy differential evolution algorithm with population size reduction and modified multiple mutation strategies. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2*, pages 13–26. Springer, 2015.

[10] Gregory W Corder and Dale I Foreman. *Nonparametric statistics for non-statisticians: a step-by-step approach*. John Wiley & Sons, 2009.

[11] Swagatam Das, Ajith Abraham, Uday K Chakraborty, and Amit Konar. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, 13(3):526–553, 2009.

[12] Lawrence Davis et al. *Handbook of genetic algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.

[13] Russ C Eberhart, James Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[14] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[15] Saber Elsayed, Ruhul Sarker, and Carlos A Coello Coello. Sequence-based deterministic initialization for evolutionary algorithms. *IEEE transactions on cybernetics*, 2016.

[16] Saber M Elsayed, Ruhul Sarker, Daryl L Essam, Noha M Hamza, et al. Testing united multi-operator evolutionary algorithms on the cec2014 real-parameter numerical optimization. In *IEEE Congress on Evolutionary Computation*, pages 1650–1657. IEEE, 2014.

[17] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Computers & operations research*, 38(12):1877–1896, 2011.

[18] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Adaptive configuration of evolutionary algorithms for constrained optimization. *Applied Mathematics and Computation*, 222:680–711, 2013.

[19] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. An improved self-adaptive differential evolution algorithm for optimization problems. *IEEE Transactions on Industrial Informatics*, 9(1):89–99, 2013.

[20] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Self-adaptive differential evolution incorporating a heuristic mixing of operators. *Computational Optimization and Applications*, 54(3):771–790, 2013.

[21] Saber M Elsayed, Ruhul A Sarker, and Efrén Mezura-Montes. Self-adaptive mix of particle swarm methodologies for constrained optimization. *Information Sciences*, 277:216–233, 2014.

[22] Saber Mohamed Elsayed. *Evolutionary Approach for Constrained Optimization*. PhD thesis, University of New South Wales at the Australian Defence Force Academy at Canberra, 2012.

[23] Andries P Engelbrecht. Heterogeneous particle swarm optimization. In *Swarm Intelligence*, pages 191–202. Springer, 2010.

[24] Qinqin Fan and Xuefeng Yan. Self-adaptive differential evolution algorithm with discrete mutation control parameters. *Expert Systems with Applications*, 42(3):1551–1572, 2015.

[25] Jacomine Grobler, Andries P Engelbrecht, Graham Kendall, and VSS Yadavalli. Alternative hyper-heuristic strategies for multi-method global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[26] Jacomine Grobler, Andries P Engelbrecht, Graham Kendall, and VSS Yadavalli. Heuristic space diversity control for improved meta-hyper-heuristic performance. *Information Sciences*, 300:49–62, 2015.

[27] Shu-Mei Guo and Chin-Chang Yang. Enhancing differential evolution utilizing eigenvector-based crossover operator. *IEEE Transactions on Evolutionary Computation*, 19(1):31–49, 2015.

[28] Nikolaus Hansen. Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2389–2396. ACM, 2009.

[29] Nikolaus Hansen, Sibylle Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003.

[30] F. Herrera and M. Lozano. Adaptive genetic operators based on coevolution with fuzzy behaviors. *IEEE Transactions on Evolutionary Computation*, 5(2):149–165, Apr 2001.

[31] Jon R Katzenbach and Douglas K Smith. *The wisdom of teams: Creating the high-performance organization*. Harvard Business Press, 1993.

[32] Yuan-Long Li, Zhi-Hui Zhan, Yue-Jiao Gong, Wei-Neng Chen, Jun Zhang, and Yun Li. Differential evolution with an evolution path: A deep evolutionary algorithm. *IEEE transactions on cybernetics*, 45(9):1798–1810, 2015.

[33] JJ Liang, BY Qu, and PN Suganthan. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2013.

[34] JJ Liang, BY Qu, PN Suganthan, and Alfredo G Hernández-Díaz. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 201212:3–18, 2013.

[35] Tianjun Liao, Marco A. Montes de Oca, and Thomas Stützle. Computational results for an automatically tuned cma-es with increasing population size on the cec'05 benchmark set. *Soft Computing*, 17(6):1031–1046, Jun 2013.

[36] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[37] Manuel López-Ibánez and Thomas Stützle. Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, 235(3):569–582, 2014.

[38] Rammohan Mallipeddi and Ponnuthurai N Suganthan. Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14(4):561–579, 2010.

[39] Rammohan Mallipeddi, Ponnuthurai N Suganthan, Quan-Ke Pan, and Mehmet Fatih Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696, 2011.

[40] Antonio David Masegosa, David Alejandro Pelta, and José Luis Verdegay. A centralised cooperative strategy for continuous optimisation: The influence of cooperation in performance and behaviour. *Information Sciences*, 219:73–92, 2013.

[41] Masao Mukaidono. *Fuzzy logic for beginners*. World Scientific, 2001.

[42] Filipe V Nepomuceno and Andries P Engelbrecht. A self-adaptive heterogeneous pso for real-parameter optimization. In *IEEE Congress on Evolutionary Computation*, pages 361–368. IEEE, 2013.

[43] Patricia Ochoa, Oscar Castillo, and José Soria. Differential evolution using fuzzy logic and a comparative study with other metaheuristics. In *Nature-Inspired Design of Hybrid Intelligent Systems*, pages 257–268. Springer, 2017.

[44] Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800, 2010.

[45] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[46] Shahryar Rahnamayan, Hamid R Tizhoosh, and Magdy MA Salama. Opposition-based differential evolution. *IEEE Transactions on Evolutionary computation*, 12(1):64–79, 2008.

[47] Luis Rodríguez, Oscar Castillo, José Soria, Patricia Melin, Fevrier Valdez, Claudia I Gonzalez, Gabriela E Martinez, and Jesus Soto. A fuzzy hierarchical operator in the grey wolf optimizer algorithm. *Applied Soft Computing*, 57:315–328, 2017.

[48] Timothy J Ross. *Fuzzy logic with engineering applications*. John Wiley & Sons, 2009.

[49] R.A. Sarker, S.M. Elsayed, and T. Ray. Differential evolution with dynamic parameters selection for optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(5):689–707, Oct 2014.

[50] Yuhui Shi, Russell Eberhart, and Yaobin Chen. Implementation of evolutionary fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 7(2):109–119, 1999.

[51] Krzysztof Socha and Marco Dorigo. Ant colony optimization for continuous domains. *European journal of operational research*, 185(3):1155–1173, 2008.

[52] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[53] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL report*, 2005005:2005, 2005.

[54] R. Tanabe and A.S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *IEEE Congress on Evolutionary Computation*, pages 1658–1665, July 2014.

[55] Ryoji Tanabe and Alex Fukunaga. Success-history based parameter adaptation for differential evolution. In *IEEE Congress on Evolutionary Computation*, pages 71–78. IEEE, 2013.

[56] Ke Tang, Fei Peng, Guoliang Chen, and Xin Yao. Population-based algorithm portfolios with automated constituent algorithms selection. *Information Sciences*, 279:94–104, 2014.

[57] Lixin Tang, Yun Dong, and Jiyin Liu. Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation*, 19(4):560–574, 2015.

[58] Fevrier Valdez, Juan Carlos Vazquez, Patricia Melin, and Oscar Castillo. Comparative study of the use of fuzzy logic in improving particle swarm optimization variants for mathematical functions using co-evolution. *Applied Soft Computing*, 52:1070–1083, 2017.

[59] Jasper A Vrugt and Bruce A Robinson. Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences*, 104(3):708–711, 2007.

[60] Jasper A Vrugt, Bruce A Robinson, and James M Hyman. Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Transactions on Evolutionary Computation*, 13(2):243–259, 2009.

[61] Hui Wang, Shahryar Rahnamayan, Hui Sun, and Mahamed GH Omran. Gaussian bare-bones differential evolution. *IEEE Transactions on Cybernetics*, 43(2):634–647, 2013.

[62] Ying-Ming Wang. Centroid defuzzification and the maximizing set and minimizing set ranking based on alpha level sets. *Computers & Industrial Engineering*, 57(1):228–236, 2009.

[63] Yong Wang, Zixing Cai, and Qingfu Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55–66, 2011.

[64] Yu Xue, Shuiming Zhong, Yi Zhuang, and Bin Xu. An ensemble algorithm with self-adaptive learning techniques for high-dimensional numerical optimization. *Applied Mathematics and Computation*, 231:329–346, 2014.

[65] Ming Yang, Changhe Li, Zhihua Cai, and Jing Guan. Differential evolution with auto-enhanced population diversity. *IEEE transactions on cybernetics*, 45(2):302–315, 2015.

[66] Zhenyu Yang, Xin Yao, and Jingsong He. Making a difference to differential evolution. In *Advances in metaheuristics for hard optimization*, pages 397–414. Springer, 2007.

[67] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2):82–102, 1999.

[68] Wenchao Yi, Liang Gao, Xinyu Li, and Yinzhi Zhou. A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. *Applied Intelligence*, 42(4):642–660, 2015.

[69] L. A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103–111, May 1996.

[70] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[71] Lotfi A Zadeh. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy sets and systems*, 90(2):111–127, 1997.

[72] Aleš Zamuda and Janez Brest. Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In Leszek Rutkowski, Marcin Korytkowski, Rafał Scherer, Ryszard Tadeusiewicz, LotfiA. Zadeh, and JacekM. Zurada, editors, *Swarm and Evolutionary Computation*, pages 154–161. Springer, 2012.

[73] Li Ming Zheng, Sheng Xin Zhang, Kit Sang Tang, and Shao Yong Zheng. Differential evolution powered by collective information. *Information Sciences*, 399:13–29, 2017.

Saber Elsayed (M'10) received the Ph.D. degree in Computer Science from the University of New South Wales Canberra, Australia, in 2012. Currently, Saber is a Research Fellow with the School of Engineering and Information Technology, University of New South Wales Canberra. His research interests include the areas of evolutionary algorithms, constraint-handling techniques for evolutionary algorithms, scheduling, big data and cybersecurity using computational intelligence. Dr. Elsayed was the winner of several IEEE-CEC competitions. He is an editorial board member of the International Journal of Business Intelligence and Data Mining and serves as a reviewer in several international journals. Saber was a member of the program committee of several international conferences.

Ruhul Sarker (M'03) received his Ph.D. degree from Dalhousie University, Halifax, Canada, in 1992. He is currently a Professor in the School of Engineering and Information Technology and the Director of Faculty Postgraduate Research, University of New South Wales, Canberra Campus, Australia. His main research interests are evolutionary optimization and applied operations research. He is the lead author of the book Optimization Modelling: A Practical Approach (CRC, Boca Raton, FL, 2007). He has published more than 250 refereed articles in the international journals, edited books, and conference proceedings. Prof. Sarker is currently an Associate Editor of the Memetic Computing Journal, Journal of Industrial and Management Optimization (JIMO), and Flexible Service and Manufacturing Journal (FSMJ)..

Carlos A. Coello Coello (M'98–SM'04–F'11) received PhD degree in computer science from Tulane University, USA, in 1996. He is currently Professor (CINVESTAV-3F Researcher) at the Computer Science Department of CINVESTAV-IPN, in Mexico City, México. Dr. Coello has authored and co-authored over 450 technical papers and book chapters. He has also co-authored the book Evolutionary Algorithms for Solving Multi-Objective Problems (Second Edition, Springer, 2007). His publications currently report over 35,800 citations in Google Scholar (his h-index is 76). Currently, he is associate editor of the IEEE Transactions on Evolutionary Computation and serves in the editorial board of 12 other international journals. His major research interests are: evolutionary multi-objective optimization and constraint-handling techniques for evolutionary algorithms. He received the 2007 National Research Award from the Mexican Academy of Sciences in the area of Exact Sciences, the 2013 IEEE Kiyo Tomiyasu Award and the 2012 National Medal of Science and Arts in the area of Physical, Mathematical and Natural Sciences. He is a Fellow of the IEEE, and a member of the ACM and the Mexican Academy of Sciences.