Adv Syst Sci Appl 2023; **04**; 156-178
Published online at https://ijassa.ipu.ru.

A Novel Two-Stage Hybrid Multi-Objective Differential Evolution with Opposition Based-Learning

Noureddine Boukhari[1*], Mohamed Amine Nemmich[2], Fatima Debbat[3], Nicolas Monmarché[4], Mohamed Slimane[4]

[1)] *Department of Mathematics, Evolutionary Engeneering & Distributed Information Systems Laboratory (EEDIS), Djillali Liabes University of Sidi Bel-Abbes, Sidi Bel-Abbes, Algeria*

[2)] *Department of computer science, Mathematics Laboratory, Djillali Liabes University of Sidi Bel-Abbes, Sidi Bel-Abbes, Algeria*

[3)] *Department of Computer Science, University Mustapha Stambouli of Mascara, Mascara, Algeria*

[4)] *Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT), Tours, France*

Abstract: Evolutionary algorithms have been shown to be powerful for solving multi-objective optimization problems, where non-dominated sorting is a widely adopted selection method. In this paper, a new two stage hybridized multi-objective evolutionary algorithm, are evolved. Firstly, multi-objective differential evolution based on ranking mutation is applied using non-dominated sorting and crowding distance. Secondly, jumping probability is used in second stage in order to meet the objective of balancing the precision of the solution and the rate of convergence while maintaining the diversity of the population by opposition-based learning technique. Through the validation of the new approach using a suite of carefully selected test benchmarks problems with complex Pareto sets and solve a large portfolio complex problem , it is observed that our hybridization achieves overall better performance in terms of convergence and diversity compared to other algorithms of literature.

Keywords: evolutionary multi-objective optimization, differential evolution, Pareto dominance, opposition-based learning, hybrid algorithm, portfolio optimization problem

1. INTRODUCTION

Among the recent advances in optimization algorithms, evolutionary algorithms (EAs) have proven successful in overcoming difficulties with traditional optimization techniques either in their standard version, or hybridized [1]. EAs work with a set of solutions, called population. This feature is particularly suitable for solving multi-objective problems, as it enables approximating the efficient frontier in a single run. A Multi-Objective Evolutionary Algorithm (MOEA) is a modified version of the traditional Genetic Algorithm (GA) - also known as the simple GA, designed to solve multi-objective optimization problems (MOPs). MOEAs have been recognized to be suitable for solving multi-objective optimization problems because of their ability to find good solutions for competing objective functions simultaneously and to search for multiple non-dominated solutions simultaneously in the population of candidate solutions. This capability enables them to find several trade-off solutions for all objectives in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the traditional techniques such as weighted sum, goal programming and weighted min-max methods. Ideally, a MOEA returns a Pareto-optimal

[*] Corresponding author: boukhari.noureddine@gmail.com

set, the solutions not dominated by any other solution in the search space. In addition, MOEAs are less susceptible to the shape or continuity of the Pareto-optimal front, whereas these two issues pose a problem for the mentioned traditional techniques.

In the past two decades , multi-objectives optimization has attracted increasing interests in the evolutionary computation community, and a large number of multi-objective optimization algorithms have been developed on the basis of different population based meta-heuristics, such as genetic algorithm [2], differential evolution algorithm [3], firefly algorithm [4], particle swarm optimization algorithm [5], SMS-EMOA[6]. Although various approaches have been adopted for selection [7], most MOEAs adopt the Pareto-based approach, i.e., the qualities of the candidate solutions are compared using Pareto dominance. Among various dominance comparison mechanisms, non-dominated sorting has been shown to be very effective for finding Pareto-optimal solutions. Much work has also been done to efficiently store non dominated solutions found during search in an archive. Non-dominated sorting is a procedure where solutions in the population are assigned to different fronts based on their dominance relationships.

Differential Evolution (DE) is a simple and efficient population-based EA that has been reported in several studies for its high robustness, fast convergence speed, and good solution quality, making it a very popular EA in the evolutionary computing community. Due to these strengths observed in DE, the use of this EA may provide an answer to some, but not all, of the above limitations. In order to overcome these limitations, several improvement works were carried out on DE in order to allow it to meet the objective of balancing the precision of the solution and the rate of convergence while maintaining the diversity of the population for different types of MOOP. In order to simultaneously achieve fast convergence speed and efficient global search capacity, researchers explored the use of hybridization of different EAs for MOOPs as well as the formulation of memory algorithms that integrate local search into EAs, and these efforts can also be found in DE [8][9]. Researchers have also introduced the use of adaptive or self-adaptive DE variants [10][11] which eliminate the need to undergo the tedious process of trial and error setting the control parameters in DE to an optimum setting for the problems tested.

Differential evolution as one of the most evolutionary algorithms, has been widely applied to solve multi-objective optimization problems due to its simple implementation and fast convergence. In order to apply DE to multi-objective optimization, there are at least two fundamental issues to be addressed. The first issue is how to define the best individual, given that there does not exist any individual which can perform the best on all objectives of an MOP. The second issue is how to balance convergence and diversity of the population. Since the target of multi-objective optimization is to obtain a set of trade-off solutions, diversity maintenance is particularly important. A DE based multi-objective algorithm is very likely to be trapped in local optimum (or one of the many optima) of an MOP due to its fast convergence. There-fore, striking a balance between convergence and diversity is crucial to the performance of multi-objective DE algorithm.

2. DIFFERENTIAL EVOLUTION FOR MOP

Several researchers have applied DE to solve multi-objective problems [12], with superior and impressive performance to other multi-objective algorithms in benchmarks and widely practical applications. Reported and verified [13][14], the general DEMO procedure is shown in Algorithm 1. Like DE, the algorithm starts with a population P randomly created solutions. For each generation, the following steps are repeated. A candidate is constructed from its parent (and other solutions of P) using the DE/rand/1/bin strategy described previously. After that, the candidate is assessed and compared to his parent. At this point, DEMO differs from DE (see step 2.1 (c) in Algorithms 1). In DEMO, the candidate replaces the parent only if he dominates him. If the parent dominates the candidate, the candidate is

discarded. Otherwise (when the candidate and the parent are incomparable), the candidate is added to the population. After repeating this step p times, we obtain a population of size between p and 2p. If the population has grown in size, it should be truncated to size p using one of the environmental selection approaches. At the end of the generation, the solutions of P are listed at random (as in DE). The end result of DEMO consists of non-dominated solutions of P.

General MODE algorithm 1 [15]
Input: parameters of the algorithm.
Output: Non-dominated solutions of *P*.
1. Evaluate the initial population *P* of *p* random solutions.
2. While the stop criterion has not been met, proceed as follows:
   2.1. For each solution x_i $(i = 1, \dots, p)$ of *P* repeat:
      (a) Create candidate c from parent x_i
      (b) Calculate the candidate's goals.
      (c) If the candidate dominates the parent, the candidate replaces the parent. If the parent dominates the candidate, the candidate is discarded. Otherwise, the candidate is added to the population.
   2.2. If the population has more p solutions, apply an environment Selection procedure to obtain the best solutions *P*.
   2.3. Randomly list the solutions in *P*.

Note that newly created candidates who enter the population (either by replacement or by addition) instantly participate in the creation of subsequent candidates. This makes it possible to achieve rapid convergence towards the optimal Pareto front.

The extension of DE to MOPs has also given promising results [16]. In what follows, we will review the multi-objective DE (MODE) under five aspects, namely adaptive strategies, selection operator, diversity control, constraint management and practical applications.

For adaptive mutation type and MODE parameterization strategies, [17] have proposed a self-adaptive MODE (MOSaDE) capable of adaptively selecting the appropriate mutation strategies. Later, a new version of the MOSaDE hybrid with learning strategies by objectives (OWMOSaDE) was developed by [18], who can adaptively select mutation strategies and crossing parameters appropriate for each objective separately. [3] presented a self-adapting MODE in which direction information from solutions archived below the current population is used in the mutation process. [19] used information from the evolution of solutions along each search direction to tailor MODE control parameters. [20] proposed a variant of MOEA/D with DE operators (MOEA/DFRRMAB) where rates of mutation operators were adaptively determined by a multi-armed bandit scheme.

For the selection operator in MODE, traditional modes [18], and Pareto differential evolution [21] used the one-to-one selection scheme. one, that is, the test vector was allowed into the next generation if and only if it could dominate its target solution. Different from this scheme, some MODEs preferred to store the new test vectors and then select good solutions from the union of test vectors and the original population to build the next population. The application of such a scheme is found in the MODE named PDEA proposed by [22], DEMO proposed by (Robič & Filipič, 2005), ADEMO proposed by [23], MODEA proposed by [24], and the DEMON proposed by [25].

Regarding the diversity control in MODE, many studies have preferred to adopt the Pareto ranking and the overpopulation distance of NSGA-II [26] to truncate the elitist archive. Instead of crowding distance, [27] used a measure of diversity entropy of crowding in the self-adaptive MODE to preserve the diversity of elite archives. [28] developed a measure of diversity based on harmonic distance to maintain the diversity of external archives and dominance has been integrated into MODE to ensure the diversity of external archives. In addition to metrics for measuring diversity, the multiple populations strategy has also been adopted in MODE to maintain diversity. A cooperative MODE (CMODE) with multiple populations has been presented by [29], in which each population treated only one

goal. Another version of the multi-population mode has been proposed in [30], in which each subpopulation adopted its own mutation strategy and competed for evolutionary resources.

For the MOP with constraints, [31] incorporated local research based on approximate set theory into MODE. [32] developed a new MODE by designing a new constraint management method to guide the search for individuals using a number of good infeasible solutions.

For the practical application of the MODE, [33] presented a MODE for the optimization of the operation of the pyrolysis of naphtha. [34] presented a MODE incorporating a taboo list for chemical engineering applications. [35] Studied the optimization of the functioning of the p-xylene oxidation reaction process and developed a MODE with a self-adaptive strategy for the generation of test vectors and the control of settings. [36] proposed a MODE with an adaptive parameter control strategy and non-dominated ranking for POM of electromagnetic problems. [37] proposed an adaptive MODE for the problem of defining the operating point with four objectives focusing on the cost of fuel consumption and reduction of emissions.

The work [38] proposed a DE variant to solve multi-objective optimization problems called MyODEMR (Multiple-objective DE with mutation restriction). The algorithm uses the concept of Pareto dominance coupled with the inverted generational distance metric to select the population for the next generation from a combination of the parent and offspring populations. The algorithm also uses a strategy of restriction of the difference vector in the DE mutation to improve the convergence characteristics on multimodal fitness landscapes. Recently, [39] proposed a multi-goal optimization algorithm that periodically rearranges goals based on their conflict status and selects a subset of conflicting goals for further processing. The authors used DEMO as the underlying metaheuristic algorithm, and implemented the technique of selecting a subset of conflicting goals using an order based on the correlation of goals. The resulting method is called α-DEMO, where α is a parameter determining the number of conflicting objectives to select. DE has also been used as a core optimizer in recently developed decomposition-based MOEAs for multi-objective optimization such as [40].

3. OPPOSITION BASED LEARNING

Opposition based learning (OBL) is a new concept of machine learning, inspired by the opposing relationship between entities. In 2005, for the first time, the concept of opposition was introduced, which has attracted a lot of research efforts over the past decade. A variety of soft computer algorithms such as optimization methods, reinforcement learning, artificial neural networks, and fuzzy systems have already used the concept of OBL to improve their performance. The concept of computational opposition [41] was inspired by the concept of opposition in the real world and opposing numbers were simply defined as follows:

Definition 3.1: (opposite number). [41] Let $\boldsymbol{x} \in [\boldsymbol{a}, \boldsymbol{b}]$ be a real number. Its opposite, $\breve{x}$, is defined as follows:

$$\breve{x} = a + b - x \tag{3.1}$$

Definition 3.1: (opposite point in space D). [41] Let $x\,(x_1, \ldots, x_D)$ be a point in dimensional space D and $x_i \in [a_i, b_i]$, $i = 1, 2, \ldots, D$. The opposite of x is defined by $\breve{x}$ $(\breve{x}_i, \ldots, \breve{x}_D)$ as follows:

$$\breve{x}_i = a_i + b_i - x_i \tag{3.2}$$

In fact, they indicate that to find the unknown optimal solution, the simultaneous search for a random direction and its opposite gives a higher chance of finding the promising regions. It is reasonable that if the current estimates (assume) are far from the unknown optimal solution, the computation of their opposites leads in the opposite direction to the unknown optimal solution. Note that the base opposite point is calculated the same as a reflected point when it is calculated through the center point $((x_1 + x_2 + \cdots + x_D)/2)$.

OBL was introduced as an attempt to increase the rate of learning in EAs. Since evolution is a slow process while revolution is a fast process, the simulation of revolution in EAs might speed up their convergence. It has also been implemented in many other optimization algorithms as mentioned in section 01. Several research works have been conducted in utilizing OBL concept and extending the new schemes of OBL to enhance EA algorithms. Also, mathematical theorems were derived to demonstrate that opposite candidate solutions have higher probability to be closer to an unknown optimal solution than the randomly generated candidate solutions.

In [42], some theorems are mathematically proved to conclude the advantage of using the OBL concept. Also, they conducted some experiments by utilizing the OBL concept in the framework of EA algorithms to enhance their performance.

In fact, according to probability theory, 50% of the time, a guess is further from the solution than its opposite guess. Therefore, it will generate the opposite individual of the current individual, evaluate the fitness of both individuals, and select the better one as a new individual which can consequently improve the quality of the search population and accelerate convergence.

The motivation for this work was twofold. First, we wanted to design an efficient algorithm for multi-objective optimization, which would use DE for decision space exploration in a simple way. Although DE-based algorithms for multi-objective optimization have already been proposed in the past, they have either ignored the basic feature of DE of comparing each new solution. to its parent, or applied it too strictly for multi-objective optimization. In addition, the existing approaches only use the environmental selection method of NSGA-II, while our objective was to allow combinations of exploration of the decision space to boost the algorithm to select better solutions with a good diversity compared to the algorithms proposed in the literature. The second objective is to apply an efficient hybridization of the DE-based multi-objective optimization algorithm with other training technique in order to reduce the complexity of the compute and that the results are precise and simple. Our hybrid approach could help by exploring space.

4. PROPOSED HYBRID ALGORITHM MODE-OBL

This section describes the multi-objective differential evolution based on the opposition MODE-OBL developed in this study on the basis of the MODE-RMO algorithm [43] (see algorithm 2). In MODE-OBL, the candidate potential solutions are processed at initialization and the exploration/exploitation capacities during the various optimization processes are mainly concerned. OBL can be used in two steps of MODE-OBL. First, at the initialization stage in order to arrive at candidate solutions for the most appropriate fit under conditions where there is no a priori knowledge about the initial individuals; Second, during the implementation of MODE-OBL in order to force the current population to embark on new candidate solutions more suitable than current ones. These two steps are called, respectively, opposition-based population initialization and opposition-based generation leap. In this way, the proposed algorithm can converge faster while maintaining good diversity. Figure 4.2 shows the overall operational architecture of the proposed algorithm. The majority of works have used OBL in the context of mono-objective optimization and recently very few attempts are invested to hybridize OBL with multi-objective evolutionary algorithms [44] [45]. Hence our approach is inspired to incorporate a two-phase (serial) learning technique into multi-objective differential evolution based ranking mutation.

4.1. Multi-Objective Differential Evolution with Ranking-Based Mutation Operator

In this section, the proposed MODE-RMO algorithm is presented in detail. In MODE-RMO, the ranking-based mutation operator is introduced in MODE to improve its performance by accelerating the speed of convergence. Therefore, in the next part, we will first describe the

ranking-based mutation operator for multi-objective optimization; then MODE-RMO is minutely elucidated.

In nature, good people always hold good information, and they are more likely to be used to guide other people. Based on these considerations, [46] proposed a ranking-based mutation operator for DE in single-objective optimization, in which parents are selected proportionally based on their ranking in the current population. The higher a parent ranks, the more they will be selected. The authors incorporated the proposed rank-based mutation operator in some DE algorithms, and experimental results indicated that the rank-based mutation operator is able to improve the performance of DE algorithms in the optimization of one goal. The ranking-based mutation operator was introduced in MODE for MOP. However, the challenge of this extension is that the population can be directly sorted from best to worst in single objective optimization, while there are many solutions that are not dominated among themselves in multi-objective optimization. Whereas the MODE algorithm should generate approximate Pareto solutions with both good convergence and good propagation, therefore, non-dominated fast sort and saturation distance are incorporated into the rank-based mutation operator to treat MOPs.

4.1.1. Fast sorting not dominated and crowding distance

The unexpressed rapid sorting and crowding distance is proposed by [47] in the NSGA-II algorithm. In the quick sort procedure without markup, for each solution, two entities are calculated:

1) n_p : the dominance account, i.e. the number of solutions which dominate the solution p;
2) S_p : a set of solutions that the solution dominates. All solutions of the first non-dominated front will have their dominance count at zero. Now, for each solution p with $n_p = 0$, we visit each member (q) of its set S_p and reduce its domination count by 1. If for a member the domination count becomes zero, we put it in a separate list Q, these members belong to the second non-dominated front. Now the above procedure continues with each limb and the third front is identified. This process continues until all fronts are identified.

The crowding distance is used to get an estimate of the density of solutions surrounding a particular individual i in the population; it calculates the average distance of its two neighboring solutions on along each of the goals. The computation of the gathering distance requires sorting the population according to each objective function value in an increasing order of magnitude. Then, for each objective function, the boundary solutions (solutions with the smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference of the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall staging distance value is calculated as the sum of the individual distance values corresponding to each goal. Each objective function is normalized before calculating the crowding distance.

4.1.2. Ranking assignment and probability of selection

After obtaining the number of non-naming fronts i_{front} and the gathering distance i_{cd} for each solution i, we can define a partial order for the whole population. Solution i is better than solution j, if one of the two conditions is met:

(i) $i_{front} < j_{front}$;

(ii) $i_{front} = j_{front}\ and\ i_{cd} > j_{cd}$

Now the population can be sorted in ascending order based on the partial order defined above. An individual's ranking is assigned as follows:

$$R_i = N_p - 1, \qquad i = 1,2,\dots,N_p \tag{4.1}$$

where N_p is the size of the population.

According to the equation. (4.1), the best individual in the current population will obtain the highest rank. After assigning the ranking for each individual, the selection probability P_i of the i^{th} individual is calculated as follows:

$$P_i = \frac{R_i}{N_p}, \qquad i = 1,2, \dots, N_p \tag{4.2}$$

After calculating the probability of selecting each individual by the equation. (4.2), the mutation operator based on the ranking of DE/rand/1 for multi-objective optimization can be presented in step 5 on Algorithm 2, where the individual with a higher ranking will have a greater probability of being selected as the base vector or the terminal vector in the mutation operator; hence, it is beneficial for the propagation of good information in the population to the offspring. In the ranking-based mutation operator for multi-objective optimization, only the base vector and the terminal vector are selected based on their selection probabilities, while the start vector is selected randomly. Indeed, if the two vectors of the difference vector are both chosen from among better vectors, the search step size of the difference vector can decrease rapidly and lead to premature convergence [48].

4.1.3. Selection operator

In MODE, the crossover is performed in the same way as in single objective optimization. However, the selection needs to be rethought, as the test vector and the target vector are often non-dominated over each other. In MODE-RMO, we use the following selection operator, which has three steps:

(1) If the test vector dominates the target vector, use the test vector to replace the target vector.

(2) If the target vector dominates the test vector, the test vector is discarded.

(3) Otherwise, the test vector is added to the population.

Thus, at the end of a generation, the total population size is between NP and 2NP. This population is truncated for the next step of the algorithm. The truncation process involves sorting and rating individuals on the same front with the crowding distance. The truncation procedure retains only the best NP vectors in the population.

Algorithm 2 general procedure of MODE-RMO

Step 1: define the number of generations $\boldsymbol{g = 0}$,

Randomly initialize the population $\boldsymbol{P^g = \{x_1^g, x_2^g, \dots, x_{Np}^g\}}$

with $\boldsymbol{x_i^g = \{x_{i1}^g, x_{i2}^g, \dots, x_{iD}^g\}}$ $\boldsymbol{(i = 1, \dots, Np)}$ uniformly distributed in the search space

Define the mutation scale factor F, the crossbreeding constant CR and the maximum number of Maxgen generations.

Step 2: evaluate the fitness value of each target vector $\boldsymbol{x_i^g}$.

Step 3: For each individual vector, perform the following steps from step 4 to step 7.

Step 4: determine the vector indices selected using the method described in 4.1.2

Step 5: Use the ranking-based mutation operator to generate a vector $\boldsymbol{m_i^{g+1}}$ corresponding to the target vector $\boldsymbol{x_i^g}$ according to equation (5).

Step 6: Use the crossover operation to generate a test vector $\boldsymbol{v_i^{g+1}}$ for each target vector $\boldsymbol{x_i^g}$ according to equation (6).

Step 7: Evaluate the test vector and use the following selection operation:

(i) if $\boldsymbol{v_i^{g+1}}$ dominates $\boldsymbol{x_i^g}$, $\boldsymbol{x_i^{g+1} = v_i^{g+1}}$.

(ii) if $\boldsymbol{x_i^g}$ dominates $\boldsymbol{v_i^{g+1}}$, $\boldsymbol{v_i^{g+1}}$ is rejected.

(iii) if $\boldsymbol{x_i^g}$ and $\boldsymbol{v_i^{g+1}}$ are not dominated with each other, add $\boldsymbol{v_i^{g+1}}$ to the population.

Step 8: After step 7, the size of the population varies from Np to 2Np.

Sorting population based on non-dominated sorting and crowding distance, and best individual $\boldsymbol{Np}$ survive the next generation.

Step 9: Set g = g + 1, go back to step 3 until the $\boldsymbol{Maxgen}$ be reached.

4.2. Initialization based on opposition learning

Random initialization, in the absence of a priori knowledge, reduces the chances of sampling better regions in population-based algorithms. However, the use of OBL can obtain more suitable starting candidates even in the absence of a priori knowledge and increase the probability of detecting better regions. OBL is an effective mechanism in the field of optimization because of the promising potential to improve the performance of various optimization algorithms, including KH [49], DE [50] and PSO [51].

The Optimization based on the opposition: Let $P\ (x_1, x_2, \ldots, x_D)$, a point in a space of dimensions D with $x_i \in [a_i, b_i]\ (i = 1, 2, \ldots, D)$, or a candidate solution. Suppose $f\ (x)$ is a fitness function used to measure candidate optimality. According to the definition of the opposite point, $\breve{P}\ (\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ is the opposite of $P\ (x_1, x_2, \ldots, x_D)$. If $f\ (\breve{P})$ is better than $f\ (P)$, then point P can be replaced by $\breve{P}$; otherwise, we continue with P. Therefore, the point and its opposite point are evaluated simultaneously to continue with the adjuster [52]. Inspired by the idea of [49], we conduct here a new method for initializing MODE with OBL, which is different from the survival selection in previous OBL-based algorithms (including OBL strategies choose the best N_p individuals among the original N_p individuals and the opposite N_p individuals in initialization). The main steps to explain this procedure are given as shown in Figure 4.1:

Step 1: Divide the population $P\ (N_p)$ into two parts, the first half of the population P_1 is generated by a random distribution.

Step 2: The remaining half of the population P_2 is initialized in terms of OBLs, as shown in section 4.2:

$$P_2 = a_i + b_i - P_1, \tag{4.3}$$

where $(i = 1, 2, \ldots, D)$

Step 3: The set $P_1 \cup P_2$ is restructured as the initial population N_p.

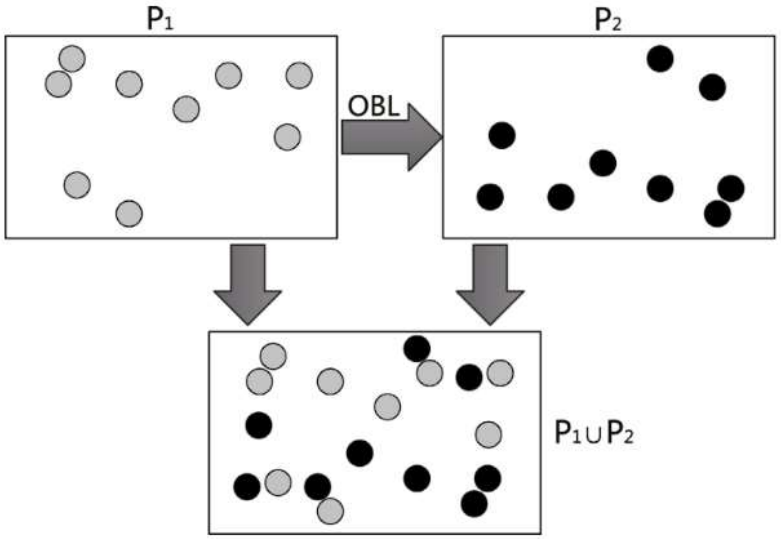

Fig 4.1. OBL-based initialization strategy in MODE

The OBL strategy used in our proposed algorithm is different from the traditional OBL based algorithm. Regarding initialization, the traditional OBL strategy first initializes the population at random, then calculates the opposite population. While the OBL operation in our proposed algorithm first divides the population into two parts, then randomly generates and calculates the inverse respectively, which can obtain more suitable starting candidates when there is no a priori knowledge of the solution. Also, after the above steps in generation and opposite calculation, the two subpopulations are made up of one population, which can

make the population size unchanged in the optimization process and help the algorithm to operate efficiently.

As for the use of OBL in the evolutionary phase, traditional OBL-based algorithms can apply the OBL in the evolutionary phase with a jump rate or a jump probability. However, the proposed MODE OBL directly calculates the opposite without the jump rate, which can increase the likelihood of effectively detecting better regions and reduce complex setting, especially for the real-world problem. This strategy is used to accelerate convergence when there is no prior knowledge of solutions, thus achieving better solutions more quickly.

4.3. Jumping Generation based on opposition

In order to improve overall convergence and avoid sub-optimal solutions, the OBL technique is reapplied to the current population. At this point, if the jump condition is satisfied:
j_r

$$rand(\quad) \leq -\left(\frac{g}{g_{max}}\right)^2 + 2\left(\frac{g}{g_{max}}\right), \tag{4.4}$$

where Gen and G_{max} are the current and maximum generations respectively [45]. The corresponding opposition population is calculated by forcing current one to pass to a new solution. After that, the fittest N_p individuals are selected from the combined population of the current population and the opposition as the current population for the next generation. Unlike the process of the opposition-based initialization phase, the generation jump computes the opposition population dynamically. Instead of using the predefined interval limits of the variables $[LB_j, UB_j]$, the generation jump calculates the opposite of each variable based on the minimum values Min_j^p and maximum Max_j^P for this variable in the current population.

$$X_{i,j}^{0,current} = Min_j^p + Max_j^P - x_{i,j}^{current}, where\ (j = 1, \dots, D; i = 1, \dots, N_p) \tag{4.5}$$

By remaining within the static limits of the interval of variables, we jump outside the already reduced search space and lose knowledge of the current reduced space (converged population). Therefore, we calculate opposite points using the current range of variables in the population which is, as we search, smaller and smaller than the corresponding initial range.

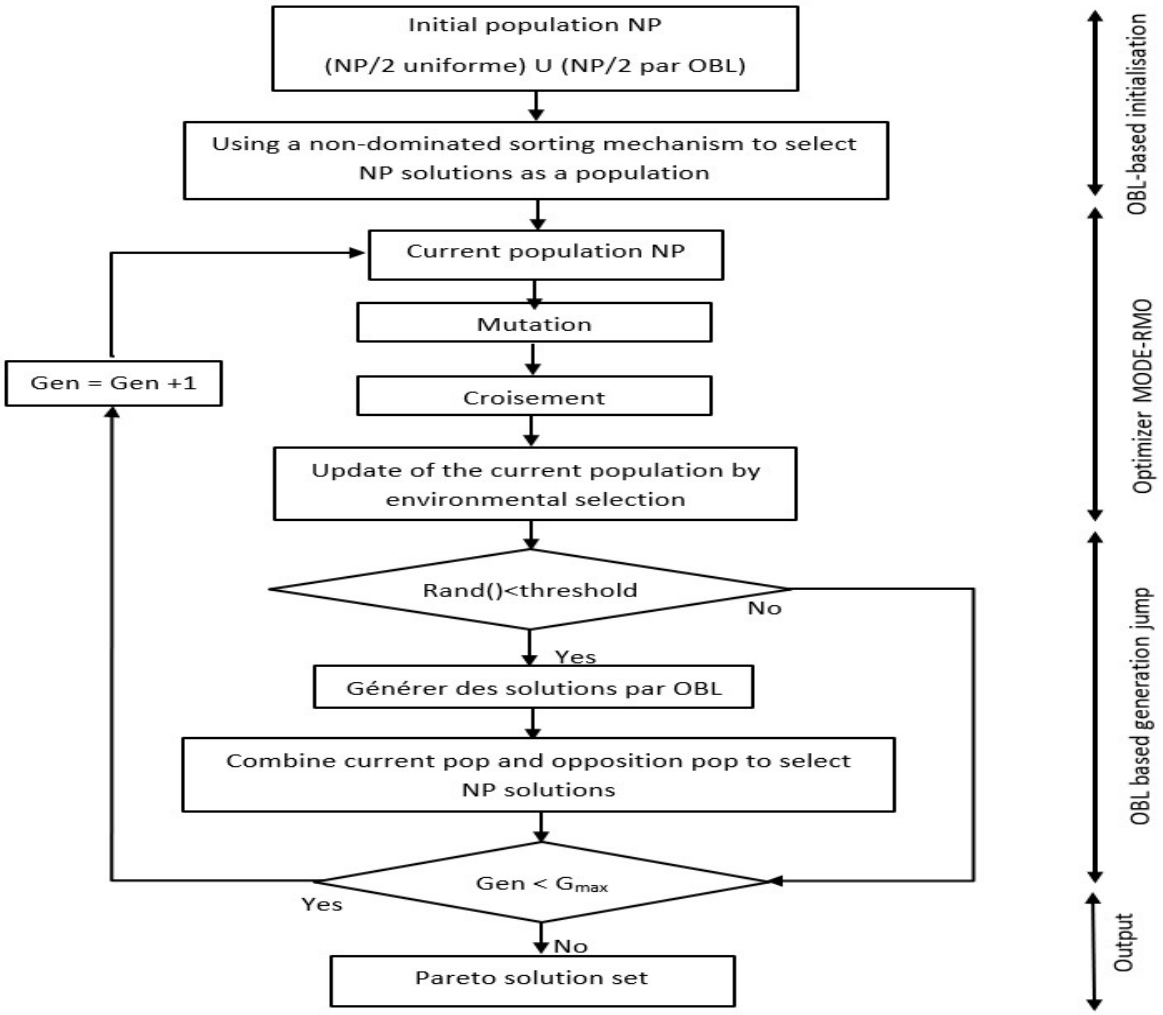

Fig 4.2. Flowchart of the proposed hybrid algorithm MODE-OBL

5. EXPERIMENTAL RESULTS AND COMPARISONS

Five state-of-the-art algorithms, namely NSGA-II, MOPSO, NSDE, MOEA/D-DE and MODE-RMO were chosen for the performance comparison with the proposed MODE-OBL. NSGA-II [53] is a popular algorithm in evolutionary multi-objective optimization because it has the ability to achieve promising solutions for most MOOPs. This algorithm uses Pareto rank and crowding distance as the operators of fitness assignment, binary tournament selection, uniform crossover, bit-flip mutation, and parent-offspring archiving. The Non-Dominated Sorting Differential Evolution (NSDE) [54] is an extension of the Basic Differential Evolution solving multi-objective optimization. It adopts the non-dominated sorting, ranking, and elitism techniques found in NSGA-II, but the main difference between them is that the NSDE uses the differential evolution mutation operator instead of the SBX operator. For MOEA/D-DE [55], it is an evolutionary algorithm that decomposes any given MOP into a number of single objective subproblems. Each subproblem is simultaneously optimized during the evolutionary research process. For the decomposition of MOOP, Tchebycheff's approach is used in this algorithm and the difference between MOEA/D-SBX and MOEA/D-DE lies in their genetic operators whereby the SBX crossover operator is used with a polynomial mutation. for MOEA/D-SBX while MOEA/D-DE uses the DE/rand/1 crossover with polynomial mutation. Finally, for MOPSO [56], it is extended from the particle swarm optimization algorithm (PSO), and it essentially combines the strong characteristics of PSO. For fair comparisons, all parameters of the compared algorithms are set to recommended values as in their original articles.

In the following subsections, the test issues and quality indicators used in our comparative experiments are first presented. Then, the experimental parameters adopted in this study are provided. In addition, thirty independent tests are performed for each test problem in order to avoid the stochastic phenomenon, and the Wilcoxon rank sum test is adopted with significance level of 0.05 to compare the results obtained by MODE-OBL and the five algorithms compared to determine if the best performing algorithm differs from competitors results in a statistically significant way. Where the symbols "+", "-" and "≈" indicate that the result is significantly better, significantly worse and statistically similar to that obtained by the MODE-OBL, respectively.

5.1. Test problems

A total of 12 benchmark test problems were chosen to test the optimization performance of the proposed hybrid algorithm MODE-OBL in terms of convergence towards the true Pareto front as well as the ability to maintain a set various solution. The test issues used included ZDT and DTLZ issues. For test problems, they can have two, three or five objective functions and have an evolving number of decision variables. These problems have been chosen because they cover different characteristics of multi-objective optimization, namely the convex Pareto front, the non-convex Pareto front, the discrete Pareto front, multimodality, and non-uniformity of the distribution of solutions. The presence of these characteristics may pose challenges to a multi-objective optimization algorithm.

Table 5.1. Multi-objective testing issues. S (scalability), M (the number of objective functions), K (scalar parameter), N (the number of decision variables), SP (separable), NS (non-separable).

Instance	*M*	*N*	*Interval*	*Geometry*	*SP/NS*	*U/M*
ZDT1	2	30(S)	$[0,1]^n$	Convex	SP	U
ZDT2	2	30(S)	$[0,1]^n$	Concave	SP	U
ZDT3	2	30(S)	$[0,1]^n$	Deconnected	SP	M
ZDT4	2	30(S)	$[0,1]^n \times [-5,5]^{n-1}$	Convex	SP	M
ZDT6	2	30(S)	$[0,1]^n$	Concave	SP	M
DTLZ1	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Lineair	SP	M
DTLZ2	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Concave	SP	U
DTLZ3	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Concave	SP	M
DTLZ4	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Concave	SP	U
DTLZ5	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Degenerated	NS	U
DTLZ6	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Degenerated	NS	U
DTLZ7	3(S)	$\boldsymbol{m} + \boldsymbol{K} - 1(S)$	$[0,1]^n$	Deconnected	SP	M

5.2. Performance indicators

In our experimental study, in order to make a fair comparison of the different optimization algorithms, performance measures relevant and applicable to the optimization objectives of convergence and distribution must be used. Two widely used measures are chosen to evaluate the performance of each algorithm, which are called the generational distance [57], and the inverted generational distance (IGD) [58]. GD and IGD can effectively measure the convergence and diversity of the obtained solutions, respectively. Convergence describes the degree of approximation of the result obtained by the algorithm to the true Pareto front (PF). The stronger the convergence of the algorithm, the closer the set of solutions is to the true optimal solution and the more precise the result. The distribution describes the distribution characteristics of the result obtained in the objective space. On the one hand, the results should be distributed as much as possible over the entire PF, and on the other hand, the results should be distributed as evenly as possible. The stronger the distribution of the algorithm, the better the overall exploration capacity of the algorithm.

5.2.1. Generational distance (GD)

Generational distance (GD) is a unary performance indicator that is defined as:

$$GD = \sqrt{\frac{\sum_{i=1}^{N} d(P_i^*, P)_i^2}{N}}, \tag{5.1}$$

where N is the number of solutions in PF^*, $p \in PF$, $p^* \in PF^*$ and $d(p^*, p)_i$ the minimum Euclidean distance in the objective space between p^* and p for each member i. GD illustrates the convergence capacity of the algorithm by measuring the convergence between the optimal Pareto front and the evolved Pareto front. Thus, a lower value of GD shows that the evolved Pareto front is closer to the optimal Pareto front. This indicator is a representative metric that provides a quantitative measure of the multi-objective optimization convergence goal.

5.1.2. Inverted generational distance (IGD):

IGD is a unary indicator by which the distance of each solution in the optimal Pareto front to the obtained Pareto front is calculated. Let P^* be a set of solutions uniformly distributed in objective space along the Pareto front. P is an approximation of the PF, which is obtained by the algorithm. IGD is described as:

$$IGD(P, P^*) = \frac{\sum_{|P^*|}^{i=1} dist(P_i^*, P)}{|P^*|}, \tag{5.2}$$

where dist(P_i^*, P) is the Euclidean distance between a point $x^* \in P^*$ and its nearest neighbor in P, and $|P^*|$ is the cardinality of P^*. We can see from the definition of IGD that for a large $|P^*|$, it can cover approximately the entire Pareto front, which is another aspect of the metric in terms of diversity.

5.3. Results

For the other algorithms compared, their parameter settings used in this study followed those used in their original studies. The experimental settings and overall parameters are summarized in Table 5.2. The comparison was made to examine their optimization performance in the test problems described above. All algorithms were implemented in MATLAB and run on an Intel® Core ™ i3 2.53 GHz computer with 6 GB memory capacity.

Table 5.2. Parameter values used in this comparison

Parameter Settings	*Parameter values*
Population size for all algorithms	100 for problems with 2 objectives 300 for problems with 3 objectives 500 for problems with 5 objectives
Stop criterion	50,000 evaluations function with 2 objectives 150,000 evaluations function with 3 goals 250,000 evaluations fonction with 5 goals
Number of decision variables for ZDT problems	300 for ZDT1, ZDT2 and ZDT3, and 100 for ZDT4 and ZDT6
Number of decision variables for DTLZ problems	12 for DTLZ1 and DTLZ3, and 120 for all the other problems DTLZ
Number of independent executions	30 for each algorithm
Mutation rate	1/n (where n is the number of decision variables)
Crossover rate for NSGA-II and NSDE	0.8
Mutation scale factor for NSDE	0.5
Neighborhood size for MOEA/D algorithms	20
Inertia weight for the MOPSO algorithm	0.5
(C_1, C_2) coéfficients for MOPSO algorithm	$C_1 = 1, C_2 = 2$

Comparative studies were conducted to evaluate the performance of the six algorithms as part of a comprehensive suite of benchmark test functions. The simulation results in terms of measurement of the mean and standard deviation values of the generational distance (GD) and the inverted generational distance (IGD) over 30 simulation cycles are presented in Tables 5.3, 5.4 and 5.5. The parentheses next to the test problems indicate the number of goals (M) and decision variables (D) for the problems.

ZDT test problems [59] are a set of simple bi-objective optimization problems that are scalable in the number of decision variables and have different characteristics in the Pareto optimal front such as the convexity, concavity, discontinuity, local optimality and non-uniformity. Since most evolutionary algorithms are able to solve ZDT problems without difficulty, the number of decision variables was set at ten times its original parameters in this study. This would then pose greater challenges to the algorithms due to a larger search space. The results indicate that MODE-OBL has the best overall performance. A notable achievement of MODE-OBL is its ability to achieve convergence for ZDT4 when all other algorithms cannot for this problem. The ZDT4 problem is an extremely multimodal problem with the presence of many local optima, and so it is likely that the other algorithms have had difficulty being trapped in the local optima. The good overall performance obtained by MODE-OBL can be attributed to the complementary effects of DE based on opposition.

Table 5.3. Results obtained by the algorithms for ZDT problems

Test problem	*Algorithm*	*IGD*	*GD*
ZDT1(2,300)	NSGA-II	0.118 (7.5e-03) -	0.066 (3.1e-03) -
	MOPSO	0.072 (8.9e-03) -	0.488 (1.5e-01) -
	NSDE	0.716 (1.9e-01) -	0.286 (4.9e-02) -
	MOEA/D-DE	0.339 (4.4e-02) -	0.189 (4.5e-02) -
	MODE-RMO	0.048 (1.1e-03) -	0.054 (3.3e-03) -
	MODE-OBL	**0.047 (6.8e-04)**	**0.052 (3.1e-04)**
ZDT2(2,300)	NSGA-II	0.079 (1.5e-02) -	0.366 (1.9e-02) -
	MOPSO	0.073 (1.8e-03) -	0.652 (7.1e-02) -
	NSDE	0.060 (2.2e-03) ≈	0.400 (2.2e-02) -
	MOEA/D-DE	0.099 (2.9e-03) -	2.889 (1.1e-01) -
	MODE-RMO	0.232 (6.8e-02) -	1.594 (2.1e-01) -
	MODE-OBL	**0.059 (1.0e-02)**	**0.339 (1.8e-02)**
ZDT3(2,300)	NSGA-II	0.447 (2.3e-02) +	0.507 (7.5e-01) -
	MOPSO	0.587 (7.1e-02) -	0.714 (4.9e-01) -
	NSDE	0.810 (1.2e-01) -	0.870 (1.2e-01) -
	MOEA/D-DE	0.669 (2.6e-01) -	0.402 (3.8e-01) -
	MODE-RMO	0.454 (8.2e-02) -	0.859 (1.4e-01) -
	MODE-OBL	**0.429 (5.4e-02)**	**0.283 (3.8e-03)**
ZDT4(2,100)	NSGA-II	0.053 (4.2e-03) -	0.011 (1.3e-03) -
	MOPSO	0.076 (4.6e-04) -	0.347 (2.0e-02) -
	NSDE	0.067 (1.2e-03) -	0.021 (2.6e-03) -
	MOEA/D-DE	0.399 (2.0e-02) -	0.742 (1.0e-05) -
	MODE-RMO	0.067 (7.3e-03) -	0.024 (5.0e-03) -
	MODE-OBL	**0.048 (7.0e-04)**	**0.006 (3.8e-05)**
ZDT6(2,100)	NSGA-II	0.034 (1.1e-03) ≈	0.101 (1.2e-02) -
	MOPSO	0.047 (4.1e-04) -	0.143 (6.4e-03) -
	NSDE	0.090 (1.9e-02) -	0.087 (2.9e-03) -
	MOEA/D-DE	0.321 (5.5e-02) -	1.156 (3.3e-01) -
	MODE-RMO	**0.034 (6.3e-04)** +	**0.068 (1.5e-03)** +
	MODE-OBL	0.042 (6.4e-04)	0.108 (9.8e-03)

The suite of DTLZ problems created by [2] can be extended to any number of objectives and decision variables. Therefore, for this study, the DTLZ problems consisted of three and five objective functions. The number of decision variables in DTLZ1 and DTLZ3 was fixed at 12 because they are highly multimodal problems and therefore more difficult. For the other DTLZ problems, they are generally easier to solve, so the number of decision variables has been set at 120 instead. For the case of DTLZ problems with three objective functions, MODE-OBL achieves competitive performances compared to the other algorithms of this study. Based on the simulation results, MODE-OBL achieves the lowest IGD and GD values or approaches the best values for most DTLZ problems. However, for the case of DTLZ5 and DTLZ6, MODE-OBL did not perform as well compared to other algorithms. For DTLZ5, we observe that the NSGA-II algorithm which incorporates the use of the SBX operator as well as MOPSO gives better results compared to algorithms with the DE operator. This suggests that using the DE operator may not be as powerful in solving degenerate Pareto front problems.

In DTLZ problems with five objective functions, the results demonstrate that MODE-OBL achieves the best overall performance for DTLZ1, DTLZ3 and DTLZ7 when opposed to all the algorithms compared. For DTLZ2, MODE-OBL reaches the lowest IGD value but not for the GD metric. For DTLZ4, DTLZ5 and DTLZ6, we observe that the decomposition-based algorithm generally performs better in terms of better IGD and GD values than the others for these three problems. This demonstrates the better ability of decomposition-based algorithms to solve multi-objective problems compared to domination-based algorithms. This is attributed to the fact that algorithms based on decomposition allow a better selection of promising solutions by using aggregated fitness values. For the other dominance-based algorithms in this study, dominance behavior between solutions must be determined before deciding which solutions are superior. However, as the number of objective functions increases, dominance behavior will be weakened. Therefore, it will be more difficult for domination-based algorithms to select the best solutions in a higher objective space.

We observe that MODE-OBL generally shows better performance for DTLZ1, DTLZ3 and DTLZ7, and competitive performance for DTLZ2, with three and five objective functions. DTLZ1 and DTLZ3 are highly multimodal problems, and the success of MODE-OBL in dealing with these problems is probably attributed to the strong exploration capabilities inherent in its DE operator which allows the algorithm to escape the optimal local. As with DTLZ2, the opposition-based generation jump phase in MODE-OBL helps produce adequate selection pressure towards the large spherical Pareto front in the large objective domain. The good performance shown by MODE-OBL for DTLZ7 could also be attributed to the strong exploratory nature of its DE operator complemented by the ranking-based mutation operator (RMO) as this helps the algorithm to discover distributed subpopulations. in all disconnected Pareto-optimal regions. Moreover, the environmental selection method used is also effective in keeping the solutions found in Pareto-optimal regions disconnected. These factors may explain why MODE-OBL is able to handle the DTLZ7 problem well.

Table 5.4. Results obtained by algorithms for DTLZ problems (3 objectives)

Test problem	*Algorithm*	*IGD*	*GD*
DTLZ1(3,12)	NSGA-II	0.200 (2.3e-02) -	2.802 (9.8e-01) -
	MOPSO	0.198 (3.0e-02) -	2.711 (9.2e-02) +
	NSDE	0.254 (7.1e-02) -	4.069 (3.0e-01) -
	MOEA/D-DE	0.764 (9.1e-02) -	3.147 (1.3e-01) -
	MODE-RMO	0.439 (2.8e-02) -	3.716 (1.6e+00) -
	MODE-OBL	**0.151 (4.9e-03)**	**2.431 (5.6e-02)**
DTLZ2(3,120)	NSGA-II	0.201 (1.5e-01) -	0.012 (1.9e-03) -
	MOPSO	0.178 (8.4e-04) -	**0.006 (1.4e-05)** +
	NSDE	0.2719 (5.1e-01) -	2.130 (5.7e+00) -
	MOEA/D-DE	**0.153 (1.1e-03)** +	0.049 (1.4e-02) -
	MODE-RMO	0.254 (4.3e-02) -	0.742 (5.8e-05) -
	MODE-OBL	0.154 (8.8e-03)	0.111 (2.4e-02)
DTLZ3(3,12)	NSGA-II	0.766 (4.0e-02) -	0.3013 (2.1e-01) -
	MOPSO	1.288 (4.1e-01) -	**0.2931 (3.7e-02)** +
	NSDE	0.948 (2.1e-02) -	1.0623 (6.5e-01) -
	MOEA/D-DE	**0.519 (1.0e-01)** +	0.4865 (2.4e-02) -
	MODE-RMO	0.877 (3.5e-01) -	0.8574 (9.2e-01) -
	MODE-OBL	0.565 (1.2e-01)	0.3021 (5.8e-01)
DTLZ4(3,120)	NSGA-II	**0.146 (1.5e-03)** +	0.387 (5.0e-04) -
	MOPSO	0.394 (2.9e-02) -	0.368 (1.1e-02) ≈
	NSDE	0.463 (8.5e-02) -	2.648 (4.1e-01) -
	MOEA/D-DE	0.180 (3.7e-02) -	0.391 (2.1e-02) -
	MODE-RMO	0.185 (8.2e-05) -	0.431 (5.2e-02) -
	MODE-OBL	0.158 (6.9e-03)	**0.368 (8.2e-03)**
DTLZ5(3,120)	NSGA-II	0.100 (1.5e-03) +	0.253 (5.3e-03) -
	MOPSO	0.177 (5.7e-03) -	0.046 (3.2e-03) +
	NSDE	0.537 (6.8e-03) -	0.134 (6.6e-01) -
	MOEA/D-DE	**0.099 (4.6e-06)** +	**0.026 (1.5e-02)** +
	MODE-RMO	0.231 (1.6e-02) -	0.668 (8.1e-01) -
	MODE-OBL	0.124 (2.7e-03)	0.211 (4.7e-03)

DTLZ6(3,120)	NSGA-II	0.413 (7.1e-02) -	1.874 (6.3e-01) -
	MOPSO	9.086 (4.8e+00) -	1.272 (3.6e-02) -
	NSDE	0.201 (4.1e-02) -	2.063 (6.1e-02) -
	MOEA/D-DE	0.112 (3.1e-03) -	**0.246 (1.3e-03)** +
	MODE-RMO	0.113 (2.5e-03) -	1.346 (5.5e-02) -
	MODE-OBL	**0.099 (2.9e-03)**	1.080 (7.3e-03)
DTLZ7(3,120)	NSGA-II	**2.547 (2.1e-01)** +	0.813 (3.8e-01) -
	MOPSO	11.18 (9.6e+00) -	0.791 (4.2e-02) -
	NSDE	18.07 (2.5e+02) -	0.955 (2.1e-01) -
	MOEA/D-DE	6.061 (6.1e-01) -	0.744 (6.4e-01) -
	MODE-RMO	7.645 (1.1e+00) -	0.669 (8.6e-02) -
	MODE-OBL	2.614 (7.7e-02)	**0.668 (2.2e-03)**

Table 5.5. Results obtained by algorithms for DTLZ problems (5 objectives)

Test problem	*Algorithm*	*IGD*	*GD*
DTLZ1(5,12)	NSGA-II	1.355 (2.6e-02) -	1.814 (1.6e-02) -
	MOPSO	8.413 (3.7e+00) -	3.297 (1.0e+00) -
	NSDE	1.686 (1.1e-01) -	2.278 (3.2e-01) -
	MOEA/D-DE	1.344 (5.0e-06) -	1.745 (1.4e-02) -
	MODE-RMO	1.567 (5.7e-02) -	1.793 (2.2e-02) -
	MODE-OBL	**1.227 (3.8e-03)**	**1.726 (2.5e-02)** +
DTLZ2(5,120)	NSGA-II	0.410 (5.9e-02) -	0.194 (7.2e-01) -
	MOPSO	1.397 (5.0e-05) -	0.438 (3.2e-02) -
	NSDE	3.561 (3.2e-02) -	4.152 (4.2e-01) -
	MOEA/D-DE	0.257 (1.1e-03) -	**0.156 (1.2e-03)** +
	MODE-RMO	0.658 (7.4e-02) -	0.306 (4.6e-02) -
	MODE-OBL	**0.224 (4.7e-03)**	0.218 (2.1e-02) -
DTLZ3(5,12)	NSGA-II	1.548 (5.9e-02) -	2.852 (4.4e-03) -
	MOPSO	3.157 (2.5e+00) -	4.043 (8.7e-02) -
	NSDE	4.699 (2.1e+00) -	2.973 (2.2e+00) -
	MOEA/D-DE	0.881 (1.1e-01) -	**1.663 (2.1e-02)** +
	MODE-RMO	1.978 (4.1e-01) -	2.528 (5.8e-02) -
	MODE-OBL	**0.682 (2.1e-01)**	1.906 (4.5e-03) -
DTLZ4(5,120)	NSGA-II	2.245 (8.9e-03) -	0.956 (1.4e-02) -
	MOPSO	4.398 (2.2e-02) -	1.030 (8.0e-02) -
	NSDE	6.544 (3.9e-02) -	3.680 (8.7e-01) -
	MOEA/D-DE	**2.224 (1.4e-03)** +	**0.899 (2.0e-02)** +
	MODE-RMO	2.365 (1.1e-02) -	3.258 (3.9e-01) -
	MODE-OBL	2.330 (1.6e-02) -	1.269 (4.4e-02) -
DTLZ5(5,120)	NSGA-II	1.267 (1.4e-02) -	0.475 (1.2e-01) -
	MOPSO	3.441 (4.7e-02) -	8.762 (5e+05) -
	NSDE	2.811 (4.3e-02) -	4.75 (1.4e-01) -
	MOEA/D-DE	**1.169 (5.0e-03)** +	**0.097 (5.7e-02)** +
	MODE-RMO	1.345 (2.8e-02) -	1.682 (5.7e+01) -
	MODE-OBL	1.219 (1.0e-02) -	1.104 (2.0e-03) -
DTLZ6(5,120)	NSGA-II	2.2e+09 (6e+09) -	15.735 (1.1e+00) -
	MOPSO	5.9e+10 (3e+10) -	17.512 (3.7e+01) -
	NSDE	2.946 (2.7e+01) -	17.23 (1.3e+02) -
	MOEA/D-DE	1.061 (5.5e-02) -	**4.98 (2.1e+00)** +
	MODE-RMO	1.402 (8.5e-02) -	12.887 (3.2e+00) -
	MODE-OBL	**0.927 (2.8e-03)**	9.216 (6.2e+00) -
DTLZ7(5,120)	NSGA-II	1.498 (1.7e-03) -	2.565 (4.6e-01) -
	MOPSO	1.463 (4.3e-01) -	7.127 (3.8e+02) -
	NSDE	12.02 (2.7e+01) -	7.982 (2.2e-03) -
	MOEA/D-DE	1.320 (2.3e-02) -	3.287 (6.8e-00) -
	MODE-RMO	1.946 (4.3e-01) -	3.889 (1.4e+01) -
	MODE-OBL	**1.053 (4.0e-03)**	**1.996 (5.2e-01)**

6. APPLICATION ON PORTFOLIO OPTIMIZATION PROBLEM

Computational finance is an emerging application field of metaheuristic algorithms. These optimization methods are becoming the solving approach alternative when dealing with realistic versions of several decision-making problems in finance.

The portfolio selection problem can be defined as the optimal allocation of wealth among a finite number of assets that follows careful processing of all available information about

both investors and markets. Markowitz's mean-variance model is by far the most popular procedure in asset allocation [60].

There are a few key concepts in portfolio optimization. First, reward and risk are measured by expected return and variance of a portfolio. Expected return is calculated based on historical performance of an asset, and variance is a measure of the dispersion of returns. Second, investors are exposed to two types of risk: unsystematic risk and systematic risk. Unsystematic risk is an asset's intrinsic risk which can be diversified away by owning a large number of assets. These risks do not present enough information about the overall risk of the entire portfolio. Systematic risk, or the portfolio risk, is the risk generally associated with the market which cannot be eliminated. Third, the covariance between different asset returns gives the variability or risk of a portfolio. Therefore, a well-diversified portfolio contains assets that have little or negative correlations [61].

EAs work with a set of solutions, called population. This feature is particularly suitable for solving multi-objective problems, as it enables approximating the efficient frontier in a single run. As a result, multi-objective evolutionary algorithms (MOEAs) have received growing attention to financial applications [62]. In fact, portfolio optimization was one of the first successful applications of MOEAs in economics and finance.

Increasing complexity of practical applications has led researchers to develop heuristic procedures for solving their portfolio optimization problems. These techniques require less domain information to be considered than the standard gradient-based mathematical programming methods do. Moreover, they guarantee satisfactory approximations to solutions in a fair computational time even when they deal with non-convexity, discontinuity, and integer decision variables. The approaches that have been proposed in the soft-computing literature can be categorized into the following two groups. On one hand, single objective methods optimize a weighted sum of the portfolio objectives. On the other hand, multi-objective evolutionary algorithms (MOEAs) attempt to tackle the allocation problem directly in its multi-objective form by simultaneously optimizing risk and reward. In the first case, the complete set of risk-return profiles is obtained by varying a parameter that represents the risk aversion of the investor [63] [64]. In the second case, the complete efficient frontier is represented in a single run [65] [66]. Both categories pay great attention to encoding types and constraint–handling techniques [67].

While single objective optimization methods consider either a minimal risk for a given return or a maximum risk for a given expected return or an objective function that weights the two goals and thus have to be run several times with the respective weights [68], multi-objective optimization methods find a set of Pareto solutions, while balancing two or more objective functions simultaneously.

6.1. Problem formulation

The key to achieving investors' objectives is to provide an optimal portfolio strategy which shows investors how much to invest in each asset in a given portfolio. Therefore, the decision variable of portfolio optimization problems is the asset weight vector $\bar{x} = [x_1, x_2, ...x_n]^T$ with x_i as the weight of asset i in the portfolio. The expected return for each asset in the portfolio is expressed in the vector form $\bar{p} = [p_1, p_2, ...p_n]^T$ with p_i as the mean return of asset i. The portfolio expected return is the weighted average of individual asset return (Eq 6.2) Variance and covariance of individual asset are characterized by a variance-covariance matrix $v = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{1n} & \cdots & \sigma_{nn} \end{bmatrix}$, where $\sigma_{i,i}$ is the variance of asset i and $\sigma_{i,j}$ is the covariance between asset i and asset j. The portfolio variance is defined in Eq (6.1).

The mathematical representation of portfolio optimization was introduced by Markowitz in the fifties and he was rewarded with a Nobel Prize in Economics in 1990 [69]. The Markowitz model assumes the investors would like to maximize the return under certain risk level or minimize the risk with certain return level and this model makes use of the mean and variance of normalized historical asset price to measure the expected portfolio return and risk [70].The model can be expressed as a bi-objective problem and formulated as following:

$$Min\text{: Risk } \sigma^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{6.1}$$

$$Max\text{: Return } \mathrm{r}_p = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i r_i \tag{6.2}$$

subject to

$$\sum_{i=1}^{n} w_i = 1\,; w_i \geq 0, \tag{6.3}$$

where n is the number of assets in the portfolio and the dimension of the problem, w_i is the weight of i^{th} asset. σ^2 stands for the portfolio risk and σ_{ij} is the covariance between asset i and asset j. If $i = j$, σ_{ij} is just the variance of that particular asset. r_p is the portfolio return, while r_i is the individual return of asset i.

6.2. Experimental Settings and dataset

All the experiments are conducted using standard Markowitz (mean-variance) model. 100 days' closing prices of 100 and 500 stocks were downloaded and used as the historical stock data in the simulation and used as the test data used in the simulation. Stocks' monthly returns and close prices are picked from [71]. A monthly return calculation is formulated as: $M_t = \ln\frac{P_t}{P_{t-1}}$; Where, $P(t)$ is the closing price, $P(t-1)$ is the closing price in the day before and $M(t)$ is the monthly return. Experiments are conducted on 100 and 500 stocks. Thus, the chromosome size is 100 and 500, respectively. And the number of function evaluation is 100000 and 300000 respectively for all the three algorithms and all algorithms are run 25 times with random initialization.

Except MODE-OBL, two other multi-objective evolutionary algorithms are also tested on this portfolio optimization problem for comparison due to their superiority respected to GD and IGD metrics in different benchmarks problems tested previously. For real-coded NSGA-II, a population size of 100 is used, crossover probability of 0.9 and mutation probability of 1/n , where n is the number of decision variables, distribution indexes for crossover and mutation operators as presented in [72]. MOEA/D-DE uses a population size of 100, F is set to 0.5 and CR is set to 0.1.

To check the robustness of the results, 25 simulations for each algorithm and for each test problem are used. The algorithms are implemented in MATLAB R2019b and the experiments are carried out on a 2.6 GHz Intel Core i5 7300U laptop with 8 GB RAM.

6.3. Experimental Results

A set of non-dominated solutions generated by MOEAs can be measured with different performance metrics available in the literature. In case, we don't have information regarding true Pareto front, then coefficient of variation measure CV defined as follow (Eq 6.4) is used:

$$CV = \frac{m}{\sigma} = mean\,(J(x), x \in X \mid \sqrt{Var\{J(x), x \in X\}} \tag{6.4}$$

which allows to determine how much volatility, or risk, is assumed in comparison to the amount of expected return from investments and spacing metric (SP) suggested by [73] is

calculated with a relative distance measure between consecutive solutions in the obtained non-dominated set, as follows:

$$SP = \sqrt{\frac{1}{|S|-1}\sum_{i=1}^{|S|}\left(\bar{d}-d_i\right)^2}, \tag{6.5}$$

where $d_i = min_{(s_i,s_j)\in S,\ s_i\neq s_j}\left\|F(s_i)-F(s_j)\right\|_1$ is the l_1 distance between a point $s_i \in S$ and the closest point of the Pareto front approximation produced by the same algorithm, and $\bar{d}$ the mean of the d_i.

From the results obtained in Table 6.1, it is observed that MODE-OBL obtains better pareto-optimal fronts with better convergence and diversity comparing with NSGA-II and MOEA/D-DE on 100-stocks problem. The differences among various algorithms are small compared with the results obtained in terms of distribution metric with slight superiority of MODE-OBL. Note that MOEA/D-DE showed good performance when optimizing the 500-stock portfolio indicating that multi-objective algorithm-based decomposition approach has a stronger ability in solving large-scale optimization problem.

A better insight into the nature of the found solutions can be obtained by analyzing Fig 6.1 and Fig 6.2. The plots presented in these figures show the efficient frontiers. As can been seen from these plots, the best results (Risk/Return) are generated by MOEA/D-DE in 500-stocks and quasi-consistent with the results obtained by MODE-OBL in 100-stocks. Although NSGAII performs the worst, it can still generate distributed and satisfactory fronts. The experiments conducted on portfolio optimization problems with 100 and 500 stocks show that the Multi-objective differential algorithm based ranking mutation combined with opposition-based learning presented an excellent performance comparing with other two multi-objective evolutionary algorithms and was a potential solution for this kind of problems.

Table 6.1. The comparison results of algorithms MODE-OBL, MOEA/D-DE, NSGA-II

Dataset	*Metrics*	*Statistics*	*MODE-OBL*	*MOEA/D-DE*	*NSGA-II*
100 stocks	Risk	Best	**0.0281e-04(1.0019)**	0.0697e-04(1.0017)	0.1351e-04(1.0016)
		Avg	0.0742e-04(**1.0093)**	**0.0665e-04**(1.0087)	0.6894e-04(1.0083)
	Return	Best	**1.1056(2.0036e-04)**	1.1036(2.004e-04)	1.1036(2.004e-04)
		Avg	**1.0093** (0.0742e-04)	1.0087(**0.0665e-04**)	1.0083(0.0689e-04)
	Coefficient of variation (CV)	Best	**6.7795e-03**	2.5005e-01	5.2053e-01
		Worst	**1.4993e-02**	2.9526e-01	5.6592e-01
		Avg	**1.1725e-02**	2.7482e-01	5.4216e-01
	Spacing metric (SP)	Best	**6.9662e-06**	8.9935e-06	6.6883e-05
		Worst	**1.4376e-05**	1.9869e-05	1.4525e-04
		Avg	**1.0603e-05**	1.2816e-05	9.8367e-05
500 stocks	Risk	Best	0.2107e-04(1.0023)	**0.1890e04(1.0025)**	0.3772e-04(1.0024)
		Avg	**0.4890**e-**04**(1.0053**)**	0.5156e-04(**1.0057)**	0.6135e-04 (1.0043)
	Return	Best	1.0067(1.0956e-04)	**1.0071(1.0253**e-04**)**	1.0051(1.2715 e-04)
		Avg	1.0053**(0.4890e-04)**	**1.0057(**0.5156e-04)	1.0043(0.6135e-04)
	Coefficient of variation (CV)	Best	**1.3851e-02**	1.5036e-02	5.6942e-01
		Worst	3.5104e-01	**3.4263e-01**	8.9172e-01
		Avg	2.8757e-01	**2.6981e-01**	6.7819e-01
	Spacing metric (SP)	Best	2.1776e-04	**1.4915e-04**	1.3038e-02
		Worst	**2.5627e-04**	5.1583e-04	4.1017e-02
		Avg	**2.3606e-04**	2.7524e-04	2.9133e-02

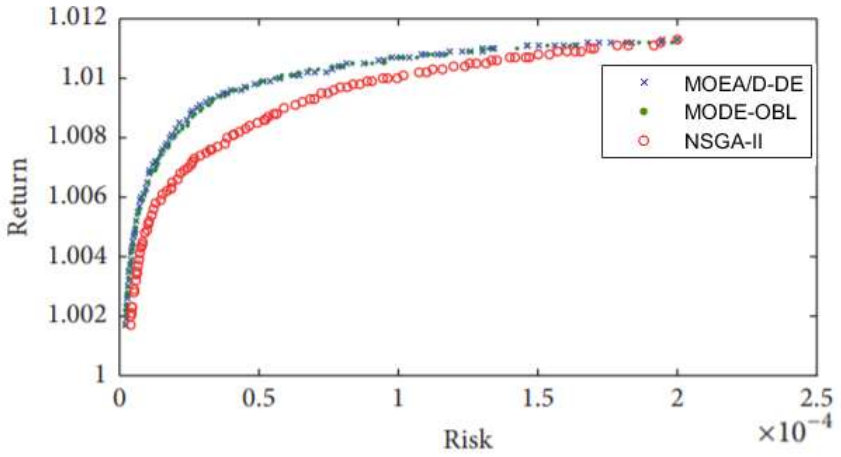

Fig 6.1 pareto frontier generated of 100 stocks problem

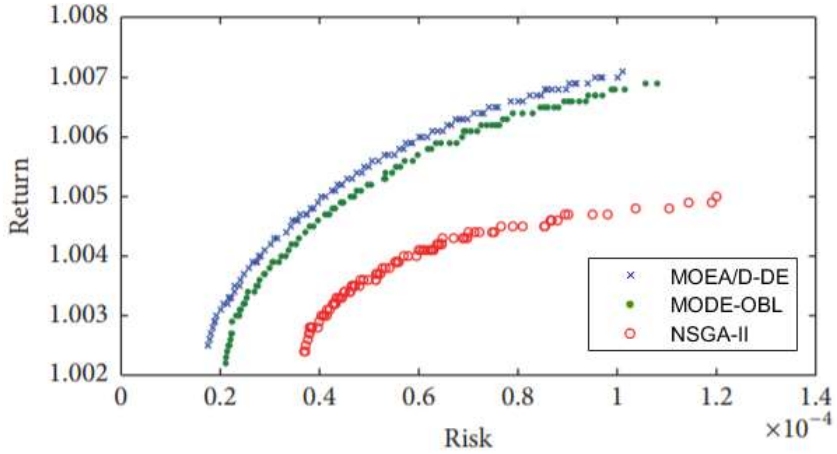

Fig. 6.2. Pareto frontier generated of 500 stocks problem

This work applied Multi-objective differential algorithm based ranking mutation combined with opposition-based learning to solve the large-scale portfolio optimization problems. The experiments conducted on portfolio optimization problems with 100 and 500 stocks show that the proposed algorithm presented a better performance comparing with other two multi-objective evolutionary algorithms and was a potential solution for this kind of problems.

7. CONCLUSION

This paper proposed a hybridized multi-objective differential evolution variant based on opposition-based learning. The proposed algorithm MODE-OBL inherited two operators from the original MODE: selection count dominance and crowding distance. Ranking based mutation from MODE-RMO is used to accelerate the convergence rate, then a new generated population passed to the second stage in order to maintain diversity of solution by the opposition-based learning technique. Experiments on mathematical benchmark functions and financial portfolio optimization problem were performed to make an extensive comparison of MODE-OBL with a series of multi-objective algorithms and MODE-OBL was considered to be competitive with other methods. However, there is still a lot of room for improvement in the improvement aspect of MODE, and this paper aims to explore feasible ways to improve the basic MODE algorithm to handle a wide range of MOP.

Future research will be directed to applying others selection techniques and verifying the MODE-OBL on more real-world problems such as telecommunication network design.

REFERENCES

1. Abbass, H. A., Sarker, R., & Newton, C. (2001). PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems, *Proc. of the IEEE Conference on Evolutionary Computation (ICEC)* (Chicago, IL).
2. Ahandani, M. A., & Alavi-Rad, H. (2012). Opposition-based learning in the shuffled differential evolution algorithm, *Soft Computing*, **16**, 1303–1337.
3. Ali, M. M. (2011). Differential evolution with generalized differentials, *Journal of Computational and Applied Mathematics*, **235**(8), 2205–2216.
4. Ali, M., Siarry, P., & Pant, M. (2012). An efficient Differential Evolution based algorithm for solving multi-objective optimization problems, *European Journal of Operational Research*, **217**(2), 404–416.
5. Angira, R., & Babu, B. V. (2005). Non-dominated sorting differential evolution (NSDE): An extension of differential evolution for multi-objective optimization, *Proc. of the 2nd Indian International Conference on Artificial Intelligence (IICAI)* (Tumkur, Karnataka, India).
6. Baatar, N., Jeong, K. Y., & Koh, C. S. (2014). Adaptive parameter controlling non-dominated ranking differential evolution for multi-objective optimization of electromagnetic problems, *IEEE Transactions on Magnetics*, **50**(2), 709–712.
7. Bandyopadhyay, S., & Mukherjee, A. (2015). An algorithm for many-objective optimization with reduced objective computations: A study in differential evolution, *IEEE Transactions on Evolutionary Computation*, **19**(3), 400–413.
8. Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume, *European Journal of Operational Research*, **181**(3), 1653–1669.
9. Caponio, A., Neri, F., & Tirronen, V. (2009). Super-fit control adaptation in memetic differential evolution frameworks, *Soft Computing*, **13**, 811–831.
10. Chen, X., Du, W., & Qian, F. (2014). Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization, *Chemometrics and Intelligent Laboratory Systems*, **136**, 85–96.
11. Coello, C. A. C., & Cortés, N. C. (2005). Solving multiobjective optimization problems using an artificial immune system, *Genetic Programming and Evolvable Machines*, **6**, 163–190.
12. Coello Coello, C. A., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, **8**(3), 256–279.
13. Coello Coello, C. A., & Reyes-Sierra, M. (2006). Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art, *International Journal of Computational Intelligence Research*, **2**(3), 287–308.
14. Cura, T. (2009). Particle swarm optimization approach to portfolio optimization. Nonlinear Analysis: Real World Applications. https://doi.org/10.1016/j.nonrwa.2008.04.023
15. Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution-An updated survey, *Swarm and Evolutionary Computation*, **27**, 1–30.
16. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002a). A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, **6**(2), 182–197.
17. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002b). A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, **6**(2), 182–197.
18. Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002). Scalable multi-objective optimization test problems, *Proc. of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI).
19. Denysiuk, R., Costa, L., & Santo, I. E. (2013). Many-objective optimization using

differential evolution with variable-wise mutation restriction, *Proc. of the 2013 Genetic and Evolutionary Computation Conference (GECCO 2013)* (Amsterdam, The Netherlands).

20. Dong, N., & Wang, Y. (2014). A memetic differential evolution algorithm based on dynamic preference for constrained optimization problems, *Journal of Applied Mathematics*, **2014**, 606019.
21. Duan, Y. C. (2004). A Multi-Objective Approach to Portfolio Optimization, *Rose-Hulman Undergraduate Mathematics Journal*, **8**, 207–227.
22. Gong, W., & Cai, Z. (2013). Differential evolution with ranking-based mutation operators, *IEEE Transactions on Cybernetics*, **43**(6), 2066–2081.
23. Gong, W., Cai, Z., & Liang, D. (2014). Engineering optimization by means of an improved constrained differential evolution, *Computer Methods in Applied Mechanics and Engineering*, **268**, 884–904.
24. Guerard, J. (2016). The theory of risk, return, and performance measurement, *Portfolio Construction, Measurement, and Efficiency: Essays in Honor of Jack Treynor*, 1–38.
25. Huang, V. L., Qin, A. K., Suganthan, P. N., & Tasgetiren, M. F. (2007). Multi-objective optimization based on self-adaptive differential evolution algorithm, *Proc. of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)* (Singapore).
26. Huang, V. L., Zhao, S. Z., Mallipeddi, R., & Suganthan, P. N. (2009). Multi-objective optimization using self-adaptive differential evolution algorithm, *Proc. of 2009 IEEE Congress on Evolutionary Computation (CEC 2009)* (Trondheim, Norway).
27. Janga Reddy, M., & Nagesh Kumar, D. (2012). Computational algorithms inspired by biological processes and evolution, *Current Science*, **103**(4), 370–380.
28. Kumar, R. (2016). Risk and return, *Valuation: Theories and Concepts*, **2016**, 47–72.
29. Li, K., Fialho, A., Kwong, S., & Zhang, Q. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition, *IEEE Transactions on Evolutionary Computation*, **18**(1), 114–130.
30. Liang, J. J., Zheng, B., Xu, F. Y., Qu, B. Y., & Song, H. (2014). Multi-objective differential evolution algorithm based on fast sorting and a novel constraints handling technique, *Proc. of the 2014 IEEE Congress on Evolutionary Computation (CEC 2014)* (Beijing, China).
31. Lin, P. C. (2012). Portfolio optimization and risk measurement based on non-dominated sorting genetic algorithm, *Journal of Industrial and Management Optimization*, **8**(3), 549–564.
32. Luong, D. L., Tran, D. H., & Nguyen, P. T. (2018). Optimizing multi-mode time-cost-quality trade-off of construction project using opposition multiple objective difference evolution, *International Journal of Construction Management*, **21**(3), 271–283.
33. Madavan, N. K. (2002). Multiobjective optimization using a Pareto differential evolution approach, *Proc. of the 2002 Congress on Evolutionary Computation (CEC 2002)* (Honolulu, HI).
34. Mahdavi, S., Rahnamayan, S., & Deb, K. (2018). Opposition based learning: A literature review, *Swarm and Evolutionary Computation*, **39**, 1–23.
35. Markowitz, H. (2014). Mean-variance approximations to expected utility, *European Journal of Operational Research*, **234**(2), 346–355.
36. Meghwani, S. S., & Thakur, M. (2017). Multi-criteria algorithms for portfolio optimization under practical constraints, *Swarm and Evolutionary Computation*, **37**, 104–125.
37. Metaxiotis, K., & Liagkouras, K. (2015). The solution of the 0-1 multi-objective knapsack problem with the assistance of multi-objective evolutionary algorithms based on decomposition: A comparative study, *Proc. of the 5th International Workshop on Computer Science and Engineering: Information Processing and Control Engineering (WCSE 2015-IPCE)* (Moscow, Russia).
38. Mishra, S. K., Panda, G., & Majhi, R. (2014). A comparative performance assessment of a set of multiobjective algorithms for constrained portfolio assets selection, *Swarm and*

Evolutionary Computation, **16**, 38–51.

39. P-N-Suganthan. (2017). *Large-Scale Portfolio Optimization Using Multiobjective Evolutionary Algorithms and Preselection Methods*, [Online]. Available: https://github.com/P-N-Suganthan/CODES/blob/master/2017-MPE-Portfolio.rar

40. Pai, G. A. V. (2019). Multi-objective Metaheuristics for Managing Futures Portfolio Risk, *Proc. of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI 2018)* (Bangalore, India).

41. Pant, M., Thangaraj, R., & Abraham, A. (2011). De-PSO: a new hybrid meta-heuristic for solving global optimization problems. *New Mathematics and Natural Computation*, **07**(03) 363–381.

42. Park, S. Y., & Lee, J. J. (2016). Stochastic Opposition-Based Learning Using a Beta Distribution in Differential Evolution, *IEEE Transactions on Cybernetics*, **46**(10), 2184–2194.

43. Ponsich, A., Jaimes, A. L., & Coello, C. A. C. (2013). A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications, *IEEE Transactions on Evolutionary Computation*, **17**(3), 321–344.

44. Qian, W., & Li, A. (2008). Adaptive differential evolution algorithm for multiobjective optimization problems, *Applied Mathematics and Computation*, **201**(1–2), 431–440.

45. Quintana, D., Denysiuk, R., Garcia-Rodriguez, S., & Gaspar-Cunha, A. (2017). Portfolio implementation risk management using evolutionary multiobjective optimization, *Applied Sciences*, **7**(10), 1079.

46. Rahnamayan, R. S., Tizhoosh, H. R., & Salama, M. M. A. (2008). Opposition-based differential evolution, *IEEE Transactions on Evolutionary Computation*, **12**(1), 64–79.

47. Rakshit, P., Konar, A., Das, S., Jain, L. C., & Nagar, A. K. (2014). Uncertainty management in differential evolution induced multiobjective optimization in presence of measurement noise, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **44**(7), 922–937.

48. Robič, T., & Filipič, B. (2005). DEMO: Differential Evolution for Multiobjective Optimization, *Proc. of the 3rd International Conference Evolutionary Multi-Criterion Optimization (EMO 2005)* (Guanajuato, Mexico).

49. Santana-Quintero, L. V., Hernández-Díaz, A. G., Molina, J., Coello Coello, C. A., & Caballero, R. (2010). DEMORS: A hybrid multi-objective optimization algorithm using differential evolution and rough set theory for constrained problems, *Computers and Operations Research*, **37**(3), 470–480.

50. Schott, J. R. (1995). Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. *Ph.D Thesis*, Massachusetts Institute of Technology.

51. Sharma, S., & Rangaiah, G. P. (2013). An improved multi-objective differential evolution with a termination criterion for optimizing chemical processes, *Computers and Chemical Engineering*, **56**, 155–173.

52. Shen, Y., & Wang, Y. (2017). Operating Point Optimization of Auxiliary Power Unit Using Adaptive Multi-Objective Differential Evolution Algorithm, *IEEE Transactions on Industrial Electronics*, **64**(1), 115–124.

53. Tang, L., Wang, X., & Dong, Z. (2019). Adaptive Multiobjective Differential Evolution with Reference Axis Vicinity Mechanism, *IEEE Transactions on Cybernetics*, **49**(9), 3571–3585.

54. Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence, *Proc. of International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)* (CIMCA 2005) (Vienna, Austria).

55. V. L. Huang, P. N. Suganthan, A. K. Qin, & B. Subramanian (2005). *Multiobjective*

differential evolution with external archive and harmonic distance-based diversity measure, [Online]. Available: https://www.researchgate.net/publication/228967624_Differential_Evolution_with_External_Archive_and_Harmonic_Distance-Based_Diversity_Measure.

56. Van Veldhuizen, D. A., & Lamont, G. B. (1998). Evolutionary Computation and Convergence to a Pareto Front. In: J.R. Koza (Eds.), *Late Breaking Papers at the Genetic Programming 1998 Conference*. Stanford, CA: Stanford University Bookstore.

57. Wang, G. G., Deb, S., Gandomi, A. H., & Alavi, A. H. (2016). Opposition-based krill herd algorithm with Cauchy mutation and position clamping, *Neurocomputing*, **177**, 147–157.

58. Wang, H., Wu, Z., Rahnamayan, S., Liu, Y., & Ventresca, M. (2011). Enhancing particle swarm optimization using generalized opposition-based learning, *Information Sciences*, **181**(20), 4699–4714.

59. Wang, J., Zhang, W., & Zhang, J. (2016). Cooperative Differential Evolution with Multiple Populations for Multiobjective Optimization, *IEEE Transactions on Cybernetics*, **46**(12), 2848–2861.

60. Wang, X., & Tang, L. (2013). Multiobjective operation optimization of naphtha pyrolysis process using parallel differential evolution, *Industrial and Engineering Chemistry Research*, **52**(40), 14415–14428.

61. Wang, X., & Tang, L. (2016). An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization, *Information Sciences*, **348**, 124–141.

62. Wang, Y. N., Wu, L. H., & Yuan, X. F. (2010). Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure, *Soft Computing*, **14**, 193–209.

63. Woodside-Oriakhi, M., Lucas, C., & Beasley, J. E. (2011). Heuristic algorithms for the cardinality constrained efficient frontier, *European Journal of Operational Research*, **213**(3), 538–550.

64. Xu, B., Qi, R., Zhong, W., Du, W., & Qian, F. (2013). Optimization of p-xylene oxidation reaction process based on self-adaptive multi-objective differential evolution, *Chemometrics and Intelligent Laboratory Systems*, **127**, 55–62.

65. Yang, S., Jiang, S., & Jiang, Y. (2017). Improving the multiobjective evolutionary algorithm based on decomposition with new penalty schemes, *Soft Computing*, **21**, 4677–4691.

66. Yang, X. S. (2013). Multiobjective firefly algorithm for continuous optimization, *Engineering with Computers*, **29**, 175–184.

67. Zhang, J., & Sanderson, A. C. (2008). Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions, *Proc. of IEEE Congress on Evolutionary Computation (CEC 2008)* (Hong Kong, China).

68. Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, **13**(5), 945–958.

69. Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Transactions on Evolutionary Computation*, **11**(6), 712–731.

70. Zhang, X., Tian, Y., Cheng, R., & Jin, Y. (2015). An Efficient Approach to Nondominated Sorting for Evolutionary Multiobjective Optimization, *IEEE Transactions on Evolutionary Computation*, **19**(2), 201–213.

71. Zhang, Y., Wang, S., & Ji, G. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications, *Mathematical Problems in Engineering*, **2015**, 931256.

72. Zhou, A., Qu, B. Y., Li, H., Zhao, S. Z., Suganthan, P. N., & Zhangd, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art, *Swarm and Evolutionary Computation*, **1**(1), 32–49.

73. Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results, *Evolutionary Computation*, **8**(2), 173–195.

