A Comparison on the Search of Particle Swarm Optimization and Differential Evolution on Multi-Objective Optimization

Jorge S. Hernández Domínguez and Gregorio Toscano Pulido
Information Technology Laboratory, CINVESTAV-Tamaulipas
Parque Científico y Tecnológico TECNOTAM
Km. 5.5 carretera Cd. Victoria-Soto La Marina
Cd. Victoria, Tamaulipas 87130, MÉXICO
(jhernandez@tamps.cinvestav.mx, gtoscano@cinvestav.mx)

Abstract—Particle swarm optimization (PSO) and differential evolution (DE) are meta-heuristics which have been found to be successful in a wide variety of optimization tasks. The high speed of convergence and the relative simplicity of PSO make it a highly viable candidate to be used in multi-objective optimization problems (MOPs). Therefore, several PSO approaches capable to handle MOPs (MOPSOs) have appeared in the past. There are some problems, however, where PSO-based algorithms have shown a premature convergence. On the other hand, multi-objective DEs (MODE) have shown lower speed of convergence than MOPSOs but they have been successfully used in problems where MOPSO have mistakenly converged. In this work, we have developed experiments to observe the convergence behavior, the online convergence, and the diversity of solutions of both meta-heuristics in order to have a better understanding about how particles and solutions move in the search space. To this end, MOPSO and MODE algorithms under (to our best effort) similar conditions were used. Moreover, the ZDT test suite was used on all experiments since it allows to observe Pareto fronts in two-dimensional scatter plots (more details on this are presented on the experiments section). Based on the observations found, modifications to two PSO-based algorithms from the state of the art were proposed resulting in a rise on their performance. It is concluded that MOPSO presents a poor distributed scheme that leads to a more aggressive search. This aggressiveness showed to be detrimental for the selected problems. On the other hand, MODE seemed to generate better distributed points on both decision and objective space allowing it to produce better results.

I. INTRODUCTION

Several multi-objective evolutionary algorithms (MOEAs) have been proposed for a wide range of multi-objective problems (MOPs). Some of these proposals include multi-objective particle swarm optimization algorithms (MOPSOs) [1]–[3] and differential evolution approaches (MODE) [4]–[6] which solve non-trivial MOPs in a low number of function evaluations. Nevertheless, few researchers have studied the search mechanisms of MOPSO and MODE, aiming to observe these search engines working at the low level. In our opinion, it will be fructuous to understand how MOEAs generate points in the search space and how these points guide the search. This aspect has been addressed by some theoretical studies for single-objective optimization [7, 8] in the past. However, an empirical approach in regards to multi-objective optimization may result in knowledge not found previously. This knowledge may allow to take full advantage of the search characteristics of a MOEA and to solve MOPs at a low number of function evaluations. To this end, a comparison seems to be appropriate to obtain knowledge such as the one described.

This article presents a comparison between the MOEAs (i) multi-objective particle swarm optimization and (ii) multi-objective differential evolution. These two MOEAs present a key characteristic on MOPs. This is that while MOPSO fails abruptly on multifrontal problems such as ZDT4, MODE has shown good results on this problem (see [9]). Even though this aspect is used as an incentive for this work, the presented comparison is not constrained to multi-frontal problems. Rather, the complete ZDT test suite [10] (with the exception of the binary problem ZDT5) is used to compare the search performed by these two algorithms. Due to space limitations, however, we only show graphical results of ZDT2, ZDT4 and ZDT6 test functions.

This article is organized as follows. Section II introduces the multi-objective optimization problem as well as the PSO and DE algorithms. On Section III, experiments are presented. First, some of the DE variants found in the literature are compared for multi-objective optimization. Then, a similar comparison is made in regards to different flight formulas of PSO. Then, an online convergence experiment is performed to compare a MODE and a MOPSO under similar conditions. This led to another experiment where the distribution of points generated by MODE and MOPSO is evaluated. On Section IV, modifications (that abide to observations of previous experiments) to the Bare Bones Differential Evolution (BBDE) [11] and Bare Bones Particle Swarm Optimization (BBPSO) [12] algorithms are proposed. These modifications promoted a rise on the performance of both algorithms. At the end, conclusions are drawn on Section V.

II. BACKGROUND

In this study, we assume that all the objectives are to be minimized and are equally important. We are interested in
solving multi-objective problems with the following form:

\[
\text{Minimize } \quad f(X_i) = (f_1(X_i), f_2(X_i), \ldots, f_M(X_i))^T \\
\text{subject to } \quad X_i \in \mathcal{F}
\]  

where \( X_i \) is a decision vector (containing our decision variables), \( f(X_i) \) is the \( M \)-dimensional objective vector, \( f_m(X_i) \) is the \( m \)-th objective function, and \( \mathcal{F} \) is the feasible region delimited by the problem’s constraints.

### A. Particle Swarm Optimization

The flight of particles on PSO is typically directed by three components. These components are: (i) velocity, (ii) cognitive, and (iii) social. These components are next described:

- **Velocity** - This component is conformed by a velocity vector which aids in moving the particle to its next position. Moreover, an inertia weight \( W \) is used to control the amount of previous velocity to apply.

- **Cognitive** - This component represents the “memory” of a particle and holds the best position achieved by the particle until the current iteration. This is achieved with a \( P_{best} \) vector which remembers the best position found so far by a particle. Also, a learning factor \( c_1 \) is used to control the level of attraction to \( P_{best} \).

- **Social** - This component represents the position of another particle known as the leader which is also used to guide the fly of the current particle. The leader is the particle with the best performance on the neighborhood of the current particle. Another learning factor \( c_2 \) is used to control the level of attraction to the leader.

The already mentioned three components found on PSO can be seen on its flying formula. When a particle is about to update its position, a new velocity vector is computed on the following way:

\[
v_i(t+1) = w v_i(t) + c_1 r_1(t)(y_i(t) - x_i(t)) + c_2 r_2(t)(\hat{y}_i(t) - x_i(t)) 
\]

Thereafter, the position of the particle is calculated using the new velocity vector:

\[
x_i(t + 1) = x_i(t) + v_i(t + 1)
\]

For 2 and 3, \( x_i(t) \) denotes the position of particle \( i \) at time \( t \), \( v_i(t) \) denotes the velocity of particle \( i \) at time \( t \), \( w \) is the inertia weight, \( c_1 \) and \( c_2 \) are the cognitive and social factors, respectively, and \( r_1 , r_2 \sim U(0,1) \). Moreover, \( y_i \) is the best position found so far by particle \( i \), and \( \hat{y}_i \) is the neighborhood best position for particle \( i \).

### B. Differential Evolution

Differential evolution was proposed under the idea that a convenient source for perturbation is the population itself. Therefore, on DE, the step size is obtained from the current population. In this manner, for each parent vector \( x_i \), a vector differential \( x_{i_1} - x_{i_2} \) is used to perturb another vector \( x_{i_3} \). This can be seen on the mutation equation,

\[
z_i(t) = x_{i_3} + F \cdot (x_{i_1} - x_{i_2})
\]

where \( x_{i_1} , x_{i_2} , x_{i_3} \) are random vectors, and \( F \) is a scaling factor.

Recombination on DE is achieved by combining elements from a parent vector \( x_i(t) \) with elements from \( z_i(t) \),

\[
\mu_{i,j}(t) = \begin{cases} 
  z_{i,j}(t) & \text{if } U(0,1) < p_r \text{ or } j = r \\
  x_{i,j}(t) & \text{otherwise}
\end{cases}
\]

where \( r \) is a random integer from \([1,2,\ldots,\text{Dimensions}]\) and \( p_r \) is the recombination probability.

### III. PERFORMED EXPERIMENTS

### A. Differential Evolution variants

Several differential evolution variants are found in the literature. Each of these vary the manner in which new candidates are created on DE \(^1\). Attempting to observe the behavior of these variants on MOPs and to use this information on further experiments, a comparison among the variants was performed. The following variants were studied:

1. Demo/Rand/1/Exp
2. Demo/Best/1/Exp
3. Demo/CurrentToRand/1/Exp
4. Demo/CurrentToBest/1/Exp
5. Demo/Rand/1/Exp
6. Demo/Best/1/Exp
7. Demo/CurrentToRand/1/Exp
8. Demo/CurrentToBest/1/Exp

As it can be seen on the previous listing, some of the variants require a “best” solution to be used. Since the concept of a best solution in multi-objective optimization is blurred by the existence of more than one objective function, this experiment aids from the DEMO algorithm [4] (all compared DE variants are based on the DEMO structure and hence the word Demo on the variants name). DEMO uses a non-dominated sorting [14] which creates layers of non-dominated solutions. For this comparison, when a best solution is needed, any individual from the first layer of non-dominated solutions was used.

Moreover, in order to obtain more detailed information on the performance of these variants, the comparison was made at 2000, 4000, 8000, and 16000 function evaluations with 30 runs on each experiment. As previously mentioned, the ZDT suite was used. The metric used is the hypervolume ratio (HVR) [15] which measures both convergence and diversity of the solution set obtained. Figures 1.1 to 1.4 show a boxplot for each of the function evaluation numbers tested.

From Figures 1.1 to 1.4, it can be seen that the best performance is achieved by Demo/Rand/1/Exp and Demo/Best/1/Exp. These two variants have better (bigger) HVR values on ZDT1, ZDT2, ZDT3, and ZDT6 at all four plots. As for ZDT4, they show competitive results for 2000, 4000, and 8000 evaluations and outperform all others when reaching 16000 evaluations.

\(^1\)Details on how these variants work can be found at [13]
flight formulas were evaluated. Some of these flight formulas are taken from single-objective proposals while others come from multi-objective PSO. The evaluated formulas come from algorithms:

1) BBPSO [12]
2) BBDE [11]
3) OMOPSO [16]
4) SMPSO [17]

In order to make a fair comparison, all formulas were adapted to the OMOPSO algorithm structure (therefore, one of the algorithms is the original OMOPSO proposal). Again, 30 runs on each experiment were performed and results are shown at 2000, 4000, 8000, and 16000 function evaluations. Figures 2.1 to 2.4 show the resulting boxplots.

Several statements can be made from Figures 2.1 to 2.4. The most noticeable is that the only algorithm able to solve ZDT4 is SMPSO. In fact, SMPSO had been previously compared with other MOPSOs (BBDE and BBPSO were not included in that study) being the only one able to solve ZDT4 [9]. It is surprising that BBDE shows the poorest performance even when its flight formula uses a recombination scheme similar to the one on DEMO which is able to solve all test problems. Furthermore, BBPSO and OMOPSO are the fastest schemes on all problems except ZDT4 where they fail. SMPSO, on the other hand, is the slowest of all. However, on the long run (16000 evaluations) SMPSO is able to reach (and sometimes pass) all algorithms. This results illustrate that for the selected problems, an aggressive MOPSO strategy is not necessarily best.

C. Online convergence of MOPSO and MODE

The two previous experiments provide information on the produced solution sets at four different function evaluation numbers. Even when these experiments allowed to observe the behavior of MODE and MOPSO under certain circumstances, it would be best to see full convergence through time for these metaheuristics. Moreover, a fair comparison between the MODE algorithms from the first experiment and the MOPSOs from the second can not be made since the algorithm structures (DEMO for the first experiment and OMOPSO for the second) are different. Therefore, on this experiment we compare the convergence of MODE and MOPSO under (to our best effort) similar conditions. For MODE, the used algorithm is a DE/Best/1/Bin variant a little different from the Demo/Best/1/Bin presented on experiment 1 (which showed the best results). This algorithm is named mode and is shown on Algorithm 1. As for MOPSO, the algorithm is similar to OMOPSO with only a small change. The main reason for using OMOPSO is that its flight formula is the most similar to the original PSO proposal and it showed a convergence speed between BBPSO (the fastest) and SMPSO (the slowest) on experiment III-B. This scheme was named mopso and is presented on Algorithm 3. Moreover, Demo/Rand/1/Bin was added as a reference since this algorithm showed good results for all test problems on experiment III-A.

B. Particle Swarm Optimization flight formulas

As in MODE, some studies are available in which the most relevant MOPSOs are compared [9]. However, these comparisons are mainly focused on the structure of the algorithms (e.g. how leaders are selected and when a Pbest is updated). For this work, rather than comparing the algorithm structures, the focus is on the flight of particles. In this respect, four PSO
The comparison was performed by running mode, mopso, and Demo/Rand/1/Bin 30 times on each test problem and recording the non-dominated solution sets at each iteration/generation on each run. These non-dominated solution sets were measured using the HVR metric with the same reference point at all iterations. For each iteration, the average HVR value (of the 30 runs) is calculated. A plot is presented were the average HVR value is shown on the y axis while the time is shown on the x axis. It is important to note that for each run, the algorithms initialize the population with the same seed yielding the initial population and non-dominated solutions sets to be the same for mode, mopso and Demo/Rand/1/Bin. Moreover, the $p_r$ and $F$ values used in mode were 0.3 and 0.5 respectively which showed the best results on DEMO. For mopso, parameters are the same as in the OMOPSO proposal. The results of this experiment are shown in Figures 3.1 to 3.3.

Figures 3.1 to 3.3 show an interesting aspect about the search performed by mode and mopso. For all test problems,

### Algorithm 1 mode algorithm used on experiment III-C

Initialize Population
Find non-dominated solutions (archive)
\[ g = 0 \]
while \( g < g_{Max} \) do
  for each individual \( i \) do
    Create candidate using a DE/Best/1/1 scheme (see Algorithm 2)
    Evaluate candidate
    Replace individual \( i \) with candidate if candidate dominates or its non-dominated with respect to solution \( i \)
  end for
  Update non-dominated solutions
end while

### Algorithm 2 Create candidate used on experiment III-C

Require: Parent \( P \)
Randomly select two individuals \( x, y \)
Randomly select a non-dominated solution \( n \) from non-dominated solutions archive
\[ r \leftarrow U \sim (0, DIM) \]
for \( j = 1 \) to \( DIM \) do
  if \( U \sim (0,1) < p_r \) or \( r == j \) then
    \[ Candidate_j \leftarrow n_j + F \times (x_j - y_j) \]
  else
    \[ Candidate_j \leftarrow P_j \]
  end if
end for

### Algorithm 3 mopso algorithm used on experiment III-C

Initialize Swarm
Find leaders (non-dominated solutions)
\[ g = 0 \]
while \( g < g_{Max} \) do
  for each Particle \( i \) do
    Flight particle \( i \) (see Algorithm 4)
    Evaluate Particle
    Update PBest if new position dominates or its non-dominated with respect to current PBest
  end for
  Update leaders
end while
mopso starts with bigger (better) HVR values than mode (the HVR values for the initial non-dominated solution sets are not shown in the plots). Since mopso and mode start with the same population and therefore with the same non-dominated solution set, this finding implies that the flight formula used on mopso is more aggressive than the candidate creation scheme of mode. However, even though mopso starts with a better HVR, it can also be seen that for all test problems, mopso starts to stagnate around iteration 50. Therefore, even though mopso seems to be more aggressive than mode for the test problems used, this aggressiveness may present a drawback on the long run.

D. MODE and MOPSO point generation

The previous experiment raises a question about where the points generated by mode and mopso land on the search space. With the aim to obtain a better understanding on this issue, another experiment was performed. The aim is to observe the distribution of points (in the objective and decision space) generated by the mopso flight formula and the mode candidate generation scheme. In this experiment, the mode and mopso algorithms are run with 1000 particles/solutions on one iteration only.

Again, aiming for a fair comparison, the mode and mopso algorithms were modified. The main reason behind these modifications is that the cognitive component on mopso subtracts the PBest from the current position of the particle. Since this experiment uses one iteration only, the cognitive component would be 0 for all particle flights (the PBest is initialized with the same values as the initial position of the particle). In order to avoid this, an archive of PBest was used. This archive stores particles that were non-dominated with respect to their current PBest. In this manner, when a particle flies, it selects a PBest that can be either its current PBest or any of the PBests of the archive. This minimizes the possibility of a 0 valued cognitive component. For the case of mode, the algorithm was also changed to reflect the changes of mopso. This time mode will add child solutions to the current population when these are non-dominated with respect to their parents. Then, when creating a candidate, recently added solutions are also taken into account. This scheme is very similar to DEMO. The modified mode and mopso algorithms are presented on Algorithms 5 and 7 respectively. Finally, both algorithms work with the same initial population and non-dominated solution set.

The results are shown using a key characteristic of the ZDT test suite. Since ZDT1, ZDT2, ZDT3, and ZDT6 have all their variables in the range $[0, 1]$, it is possible to use a single histogram that shows the frequencies for the generated values for the variables of each solution/particle (in the decision space). For the case of ZDT4, two histograms are used since this problem has the first variable in the range $[0, 1]$ while the rest lie on $[-5, 5]$. As for the generated points in objective space, a scatter plot is used where the $x$ axis shows the first objective, while the $y$ axis shows the second objective (the ZDT test suite is bi-objective). Moreover, the objective space plots show the non-dominated points (squares in red) used in mode and mopso.

Figures 4.1 to 4.8 show the obtained histograms for the values on decision space. From these histograms, it can be observed that mode generated points with a much better distribution than mopso for all test problems. Even when mode
Algorithm 5: mode algorithm used on experiment III-D

Initialize Population
Find non-dominated solutions (archive)
for each individual \( i \) do
    Create candidate using a DE/Best/1/bin scheme (see Algorithm 6)
    Evaluate candidate
    Replace individual \( i \) with candidate if candidate dominates solution \( i \)
    Add candidate to Population if non-dominated with respect to current solution \( i \)
end for

Algorithm 6: Create candidate for experiment III-D

Require: Parent \( P \)
Randomly select two individuals \( x, y \) from Population (including recently added solutions)
Randomly select a non-dominated solution \( n \) from non-dominated solutions archive
\( r \leftarrow U \sim (0, DIM) \)
for \( j = 1 \) to \( DIM \) do
    if \( U \sim (0, 1) < p_r \) or \( r == j \) then
        \( \text{Candidate}_j \leftarrow n_j + F \ast (x_j - y_j) \)
    else
        \( \text{Candidate}_j \leftarrow P_j \)
    end if
end for

Algorithm 7: mopso algorithm used on experiment III-D

Initialize Swarm
Find leaders (non-dominated solutions)
for each Particle \( i \) do
    Flight particle \( i \) (see Algorithm 8)
    Evaluate Particle
    Update particle \( i \) PBest if new position dominates PBest
    Add new particle to archive of PBests
end for

Algorithm 8: Flight Particle for experiment III-D

Require: Particle \( P \)
Randomly select a leader \( L \) from leaders archive
Randomly select a PBest \( P \) (either PBest from current particle or one from the PBests archive) with same probability
Flight particle \( P \) using \( L \) as leader and \( P \) as PBest

Histograms show high frequencies on the variable bounds, these frequencies are not as high as the ones presented on mopso. In a sense, mopso is wasting all these values that land on the variable bounds. In order to see how these frequencies translate into decision space, Figures 5.1 to 5.6 show plots where the density of the generated points is appreciated. The plots show that mode generates a better distribution of points for all test problems. Another interesting aspect is that the points generated by mopso are (for all test problems) always closer to the true Pareto front than the ones generated by mode. In other words, mopso produces more points that dominate the points in red. Again, these results imply the aggressiveness of mopso.

IV. PROPOSALS

A. Bare Bones Particle Swarm and Bare Bones Differential Evolution modification

From the observations made on previous experiments, it seems that for the selected test problems, two key aspects are necessary to achieve good results. First, values in the
recombination was added to
the only change is
seen on “xperiment ---s0ot The ..=ˇ@ and ..,” schemes.

Figure 5. Plots of generated points (objective space) by mopso and mode on ZDT2, ZDT4, and ZDT6.

decision space need to be well distributed among the variable ranges (as observed for Experiment III-D). Second, enough selection pressure needs to be present in the optimizer (as seen on Experiment III-A). The BBPSO and BBDE schemes were modified such that they abide to these two observations. BBPSO was added a recombination scheme that uses a random leader (non-dominated solution) instead of a PBest as in the original proposal. This in hope that generated values on decision space are well distributed while adding more selection pressure to the algorithm. For the case of BBDE where recombination is already in the original proposal, the only change is the particle used. Again, this particle is now a leader instead of a PBest. Moreover, borrowing from SMPSO, polynomial recombination was added to 1 out of every 6 particles on both schemes with the aim to promote exploration. This schemes were named BBDE2 and BBPSOREcombination. Figures 6.1 to 6.4 show the obtained results when compared against OMOPSO, SMPSO, NSGA-II [14] ², and Demo/Best/1/Bin.

The results show that both schemes improved with respect

²the implementation was taken from the KANGAL research laboratory website and used 0.9 for crossover, 1/D1M for mutation, 15 for distribution index for crossover, and 20 for distribution index for mutation
to the initial proposals. Both schemes seem to be a little less aggressive while giving better results as iterations increase. Furthermore, it is strange that on some problems (i.e ZDT1 and ZDT3) both BBDE2 and BBPSOREcombination are still the fastest of all algorithms. However, on ZDT2 they are the slowest. As for ZDT4, even when both schemes are not the best on 16000 evaluations they improved considerably with
respect to the original proposals.

V. CONCLUSION

In conclusion, the experiments performed on this work imply that MOPSO and MODE have big differences on their search. While MOPSO seems to be a much more aggressive scheme, this has shown to be a detrimental as generations elapse. On the decision space, new points may be out of range which most of the time will not help the search. On objective space, the new positions may be much closer to the true Pareto front but this seems to deter exploration and promote very abrupt movements through the search space (particles flying long distances in the search space). This behavior may be the opposite to the equilibrium that is found on other metaheuristics such as simulated annealing and that has shown to be key in their search. On the other hand, MODE seems to be a more passive metaheuristic that succeeds in finding diverse points both in decision and objective space. This may be due to the recombination (that allows solutions to be close to their parents) scheme that provides the equilibrium needed. This effect seems to delay the search in comparison with MOPSO but this appears to be a better strategy for the selected problems. Finally, it is important to note that the observations presented on this document are restricted to the selected test problems and that this may not be the case for other MOPs. However, this work may serve as an incentive for further research on these two metaheuristics.

ACKNOWLEDGMENT

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Information Technology Laboratory, CINVESTAV-Tamaulipas. The second author gratefully acknowledges support from CONACyT through project 105060. Also, this research was partially funded by project number 51623 from “Fondo Mixto Conacyt-Gobierno del Estado de Tamaulipas”. Finally, we would like to thank to “Fondo Mixto de Fomento a la Investigación científica y Tecnológica CONACyT - Gobierno del Estado de Tamaulipas" for their support to publish this paper.

REFERENCES


