Investigating multi-view differential evolution for solving constrained engineering design problems

Abstract

Several constrained and unconstrained optimization problems have been adequately solved over the years thanks to advances in the metaheuristics area. In the last decades, different metaheuristics have been proposed employing new ideas, and hybrid algorithms that improve the original metaheuristics have been developed. One of the most successfully employed metaheuristics is the Differential Evolution. In this paper it is proposed a Multi-View Differential Evolution algorithm (MVDE) in which several mutation strategies are applied to the current population to generate different views at each iteration. The views are then merged according to the winner-takes-all paradigm, resulting in automatic exploration/exploitation balance. MVDE was tested to solve a set of well-known constrained engineering design problems and the obtained results were compared to those from many state-of-the-art metaheuristics. Results show that MVDE was very competitive in the considered problems, largely outperforming several of the compared algorithms.

Keywords: Evolutionary computations, Global optimization, Constrained optimization, Metaheuristics, Differential Evolution.

1. Introduction

Over the years, several metaheuristics have been proposed to solve constrained problems by finding global optimum solutions. Some are hybrid approaches (Michalewicz and Schoenauer, 1996; Liu et al., 2010; He and Wang, 2007b; Pedamallu and Ozdamar, 2008), many are classical algorithms with new operators or improvements (Coelho, 2010; He and Wang, 2007a; Pant et al., 2009; Coello and Mezura-Montes, 2002; Kimbrough et al., 2008), others are self-adaptive version of classical algorithms (Coello Coello, 2000; Michalewicz and Schoenauer, 1996; Mezura-Montes et al., 2006; Noman and Iba, 2008).

An improved EA, named Differential Evolution (DE) (Storn and Price, 1997), was presented as an effective, robust, and simple global optimization algorithm which has only a few control parameters. Many works have shown that DE outperforms many other optimization methods, in terms of convergence speed and robustness, in solving hard benchmark functions and real-world problems (Chakraborty, 2008). A recent and very complete review can be seen in (Das and Suganthan, 2011).

DE has only three parameters: population size, the amplification factor ($F$) and the crossover probability ($CR$). Choosing an adequate configuration depends on the problem and on the mutation and crossover operators. Based on these issues, several variants have been proposed to improve DE in a self-adaptive way, employing several strategies at the same time and dynamic adjustment of parameter control over the generations to perform better exploration and exploitation of the search-space (Qin et al., 2009; Zhang and Sanderson, 2009; Wang et al., 2011).

In general those approaches combine several strategies with several control parameter settings, according to percentage of success in replacing the parent solution, to generate a single population.
After the population is evaluated, the percentages of selection from the possible combinations are updated. Thus, in the following iterations some combinations get a higher possibility of generating children than others.

In this paper a similar approach was taken, but without calculating percentages and using a different combination of the trial solutions. Also, for the current algorithm there is no self-adaptation of the control parameters. The proposed approach, called Multi-View differential evolution (MVDE) is a simple modification of the DE algorithm and thus requires a low effort to be implemented.

To evaluate MVDE’s performance, the algorithm was employed to solve five well-known constrained engineering design problems, and the results were compared to those from several state-of-the-art algorithms.

This paper is organized as follows. In Section 2 the Differential Evolution algorithm is briefly introduced. The new algorithm, MVDE, is elaborated in Section 3. Section 4 introduces constrained optimization and presents the numerical examples (engineering problems), details of the experiments, the results obtained, and the discussion. Finally, in Section 5 some conclusions are drawn about the results.

2. Differential Evolution

Differential Evolution was introduced by Storn and Price (1995). It is a real-valued populational metaheuristic that works like evolutionary algorithms, successfully used to solve several benchmarks and real-world problems (Neri et al., 2011; Pan et al., 2011; Wang et al., 2010; Chakraborty, 2008; Melo and Delbem, 2009).

The basic functioning is as follows. A population $P$ with $N$ vectors of $D$ dimensions is randomly initialized (using a uniform distribution) inside the problem’s bounds and evaluated using the objective/fitness function for the problem. Then, until a stop condition is satisfied, the algorithm performs an iterative evolutionary process of mutation, crossover and selection operations.

For each vector $x_i$ from $P$, the mutation operator uses the weighted difference of parent solutions to generate trial vectors $v_i$. In this work the following mutation strategies were selected:

1. **rand/1**

   \[ v_i = x_{r1} + F \times (x_{r2} - x_{r3}) \]  

2. **best/1**

   \[ v_i = x_{best} + F \times (x_{r2} - x_{r3}) \]  

3. **current-to-best/1**

   \[ v_i = x_i + F \times (x_{best} - x_i) + F \times (x_{r1} - x_{r2}) \]  

4. **best/2**

   \[ v_i = x_{best} + F \times (x_{r1} - x_{r2}) + F \times (x_{r3} - x_{r4}) \]  

5. **rand/2**

   \[ v_i = x_{r1} + F \times (x_{r2} - x_{r3}) + F \times (x_{r4} - x_{r5}) \]  

where $x_{r1}$, $x_{r2}$, $x_{r3}$, $x_{r4}$, and $x_{r5}$ are five distinct and randomly chosen vectors from $P$, $x_{best}$ is the best solution from $P$, and $F \in [0, 2]$ is the mutation or amplification factor. Classically, the binomial crossover operator is applied on $v_i$ to generate the final offspring vector $u_i$ according to
\[ u_{i,j} = \begin{cases} v_{i,j}, & \text{if } U \sim (0, 1) \leq CR \text{ or } j = \text{trunc}(U \sim (1, D)), \\ x_{i,j}, & \text{otherwise} \end{cases} \]

where \( j = 1, \ldots, D \); \( U(a, b) \) is a random floating-point number from a uniform distribution between \( a \) and \( b \) generated for each \( j \), and \( CR \in [0, 1] \) is the crossover probability.

Finally, the selection step selects the best evaluated vector between \( x_i \) and \( u_i \). The offspring replaces the parent if its fitness value is better. Otherwise, the parent is maintained in the population.

3. Multi-View Differential Evolution

In this work, the Multi-View learning (Chen and Yao, 2008; Crammer et al., 2008) is proposed as a metaheuristic enhancement and is directly applied to improve DE. In Machine Learning Mitchell (1997), Multi-View learning is a paradigm in which a learning algorithm uses the agreement among multiple learners to decide about a prediction. Multiple hypotheses are trained from the same dataset - where the instances are known (labeled) - to generate predictions on one or more unlabeled examples. In a classification process each hypothesis (called view) may be a different algorithm (neural networks, SVM, decision-trees, etc) or the same algorithm with different settings. Each view presents a response/prediction for the unlabeled example to be classified. A voting procedure, for instance, is then employed to decide the winner prediction among the views.

Based on that idea, the algorithm proposed in this work is named Multi-View Differential Evolution (MVDE). Instead of using several populations, sub-populations, or co-evolution, the idea proposed in this work consists of employing different strategies to generate new trial solutions from the same population, thus providing different views for the same problem. Different views lead to exploration of different regions. Some strategies generate solutions toward a local optimum whereas other strategies are better in escaping from a local optimum areas. No self-adaptation is employed.

However, differently from the approaches employed in other similar algorithms (Qin et al., 2009; Zhang and Sanderson, 2009; Wang et al., 2011) that change the number new of trial vectors for each strategy to maintain the population size, in MVDE \( N \) trial vectors are always generated for each view which are then merged to be the selected population of trial vectors. The proposed algorithm is presented in Algorithm 1.

Three modifications in the classical DE are proposed. First of all, a large sample of points in the search-space is created. The fitness is calculated and the \( N \) best solutions are selected to be MVDE's population, as proposed in (Melo and Delbem, 2008). This allows for a better initial coverage and possibly unbiased starting sampling of the search-space.

The second and main modification is the Multi-View approach shown in Figure 1. Starting from the current population, several views with \( N \) trial vectors are created. In this paper five views are employed, each one corresponding to a strategy presented in Section 2. One important characteristic of the Multi-View approach is that the use of the views may provide better exploration of the search-space, allowing it to escape from local optima more often than when only a aggressive search is employed. This can be achieved by using mutation strategies with such characteristics (\( rand/1 \), for instance, to perform exploration) while performing local exploitation in views guided toward the best solution found (\( best/1 \), for instance).
Algorithm 1 Algorithm of MVDE.

Generate a large population
Evaluate the population
Select best $N$ solutions to be the actual initial population
Initialize $views(1\to v)$

Do

/* Generate the views: */
For each $view$
    Apply, in the current population, the mutation strategy corresponding to the current $view$ and save it in the $view$'s population
    Evaluate the vector generated
End for

/* Elitism: */
Save current best solution and the best vector of all $views$ in the merged mutated population

/* Merge the views: */
For $index = 1$ to $N$ in the merged mutated population
    For each $view$
        Get the vector from position $index$
    End for
    Apply tournament-selection in the taken vectors
    Save the winner in the merged mutated population
End for
Remove the two worst vectors from the merged mutated population

/* Usual crossover */
Apply crossover to the merged mutated population

Until the termination condition is reached
The third and last modification is that after all views are generated and evaluated, the best vector of all views in the current generation and the current best solution ($x_{best}$), are saved (elitism). Then, tournament selection is applied to choose the remaining best vectors of the same index from different views: $view_1^i$, $view_2^i$, ..., $view_v^i$, $i = 1...N$. To keep the correct population size ($N$), one must remove the worst solutions. This selected population is then sent to the usual crossover operator which is, in our case, the binomial crossover presented in Section 2.

In this process, sometimes the best solution of a generation is generated by one view, while at another times a different view may find better solutions. Depending on the problem function, an initial local exploitation phase may be performed for several generations until the local-search strategy becomes unable to generate better solutions, possibly because it found a local optimum. In this case, another strategy may escape from the local optimum. After that, the local-search can start again at another place. The interesting part is that all this process is automatically performed by the tournament selection based on the pressure selection concept, without hard-coded heuristics or self-adaptation to chose the more adequate mutation strategy to be applied according to some probability, history or learning.

4. Experimental studies

In order to investigate the performance of the proposed algorithm, some real-world constrained engineering design problems were selected. The chosen problems have been well studied before as benchmark functions by several authors. First of all, the constrained optimization task is briefly described. Then the problems, the configuration of the algorithm, and the numerical results are presented. For each studied problem the obtained results are compared with some results of EA-based methods previously reported in literature.

4.1. Constrained Optimization

In real-world, several engineering design problems can be formulated in a nonlinear programming way. In this problem, one wants to find $\vec{x}$ that optimizes $f(\vec{x})$ which is

$$\begin{align*}
\text{subject to} & \quad h_i(\vec{x}) = 0 & i = 1, 2, ..., m \\
& \quad g_i(\vec{x}) \leq 0 & i = 1, 2, ..., p
\end{align*}$$

where $f(\vec{x})$ is the objective function to be optimized, and $\vec{x} \in \mathbb{R}^n$ is an n-dimensional vector $\vec{x} = [x_1, x_2, ..., x_n]^T$. This vector may contain mixed variables such as integer, discrete and continuous ones. Each $x_k$, $k = 1, ..., n$ can be bounded by lower and upper limits $L_i \leq x_k \leq U_k$;
4.1.1. Constraints handling

There is an important aspect in constrained optimization that directly affects the search, which is the method employed to handle the constraints. The method must guide the optimization to feasible regions and be able to reach the bounds of the search-space. An unfeasible solution may be removed from the population or repaired to become a feasible one. A common approach is the use of penalties (Michalewicz and Schoenauer, 1996) (see Equation 8) and it is the same technique chosen for this paper. A penalty function is applied to unfeasible solutions to generate a poor function value. The penalty may be a fixed value or changed during the optimization process. If the solution ($\vec{x}$) is feasible ($F$), then the penalty is 0.

$$f(\vec{x}) = \begin{cases} \text{objective\_function}(\vec{x}) & \text{if } \vec{x} \in F \\ \text{objective\_function}(\vec{x}) + \text{penalty}() & \text{otherwise} \end{cases} \quad (8)$$

This way the unfeasible solution may be removed from the population in the selection step of the MVDE algorithm. No repair method is applied. Using this approach, the constrained problem is treated as an unconstrained one. This simple yet effective approach was adopted in this work to evaluate the performance of MVDE. As the problems studied in this work are minimization ones, the penalty function just adds a very high value ($1e10$) to the objective function. This penalty value was empirically chosen based on the objective function values generated by the tested problems. It is known that the constraints handling method directly affects the search. However, if this simple penalty function present good results further work may be the study of better methods.

4.2. Numerical Examples

Five well explored engineering design problems, commonly studied in literature, are presented next.

1. Design of a Welded Beam

   In this problem, a welded beam must be designed for minimum cost subjected to some constraints named: shear stress ($\tau$), bending stress in the beam ($\sigma$), blocking load on the bar ($P_b$), end deflection of the beam ($\delta$), and side constraints. This problem presents four design variables: $h(x_1)$, $l(x_2)$, $t(x_3)$, and $b(x_4)$, where $0.1 \leq x_1 \leq 2.0$, $0.1 \leq x_2 \leq 10.0$, $0.1 \leq x_3 \leq 10.0$ and $0.1 \leq x_4 \leq 2.0$. See the mathematical model of the optimization problem, and the constraints, in (Coello Coello and Becerra, 2004; Mezura-Montes et al., 2006; He and Wang, 2007b; Coello Coello, 2000; He and Wang, 2007a);

2. Design of a Speed Reducer

   The weight of a speed reducer must be minimized subject to 11 constraints regarding the following characteristics: bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft. The problem has 7 variables to be minimized ($x_1, \ldots, x_7$): the face width, module of the teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings, and the diameter of the first and second shafts (Ray and Liew, 2003), with the ranges $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.3 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, and $5 \leq x_7 \leq 5.5$. This is
a mixed integer programming problem, where all variables are continuous, except $x_3$ that is integer. Constraints and mathematical definition are in (Ray and Liew, 2003; Zhang et al., 2008; Liu et al., 2010; Mezura-Montes et al., 2006; Cagnina et al., 2008; Brajevic et al., 2011).

3. Design of a Three-bar truss
This problem considers a 3-bar planar truss structure (Ray and Liew, 2003). The volume of a loaded 3-bar truss must be minimized subject to stress ($\sigma$) constraints on each of the truss members. The design optimization problem presents two continuous variables ($x_1 = A_1$ and $x_2 = A_2$) and three nonlinear inequality constraints with the following ranges: $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$ (see constraints and definition in (Ray and Liew, 2003; Gandomi et al., 2011; Kashan, 2011; Liu et al., 2010));

4. Design of a Pressure Vessel
A cylindrical pressure vessel with two hemispherical heads must be designed for minimum fabrication cost (Coello Coello, 2000; Ray and Liew, 2003; He and Wang, 2007a; Mezura-Montes et al., 2006; Shen et al., 2009; Coelho, 2010). Four variables are identified: thickness of the pressure vessel ($x_1 = T_s$, $1 \leq x_1 \leq 99$); thickness of the head ($x_2 = T_h$, $1 \leq x_2 \leq 99$); inner radius of the vessel ($x_3 = R$, $10 \leq x_3 \leq 200$), and length of the vessel without heads ($x_4 = L$, $10 \leq x_4 \leq 200$). $T_s$ and $T_h$ are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates. As MVDE uses floating-point, the values for $x_1$ and $x_2$ are truncated to integers. $R$ and $L$ are treated as continuous variables;

5. Minimization of the weight of a Tension/Compression Spring
A tension/compression spring must be designed for minimum weight subject to the following constraints: shear stress, minimum deflection, surge frequency, limits on outside diameter and on design variables. The design variables are: the wire diameter ($x_1 = D$), the mean coil diameter ($x_2 = d$), and the number of active coils ($x_3 = N$). The design optimization problem presents three continuous variables and four nonlinear inequality constraints with the following ranges: $0.25 \leq x_1 \leq 1.3$; $0.05 \leq x_2 \leq 2.0$, and $2 \leq x_3 \leq 15$ (equations and constraints are presented in (Coello Coello, 2000; Ray and Liew, 2003; He and Wang, 2007a; Mezura-Montes et al., 2006; Shen et al., 2009; Coelho, 2010)).

4.3. Configuration of the Algorithm
The control parameters were configured as $F = 0.9$ and $CR = 0.5$ for all tests. The population sizes ($N$) are 40, 50, 30, 50, and 50; and the number of function evaluations were limited to 15,000, 30,000, 7,000, 15,000, and 10,000 for Problems 1 to 5, respectively. These amounts of function evaluations were sufficient to achieve high-quality solutions and to present competitive mean results, which is the focus of this paper.

For all experiments it was adopted $\pi = 3.14159265358979$. The results are presented using high-precision numbers to allow a correct manual calculation of the objective function values found in the experiments.

4.4. Experimental Analysis
For each problem 1000 runs were performed because the usual 30 runs was considered unsatisfactory to offer an adequate statistical distribution of the final results. For each Problem evaluated in this section some information are provided: a plot (called profile) of the percentage of selection of each view over the generations, a summary of statistics from the runs, the best solution found, and a table with results of some state-of-the-art approaches for comparison.
4.4.1. Problem 1: Welded Beam

Figure 2 presents the profile of each view (strategies 1 to 5) during the optimization process for this problem. The plot presents the percentage of times (from 1000 runs) a solution from a specific view was selected in the tournament, by generation. The plot is smoothed to better show the differences among the curves.

All curves present similar behavior, with almost constant percentage of use. The separation of the global exploration phase from the local exploration is not clear in the plot. However, the profile shows that view 1 (the classical rand/1 strategy) generated the highest quality solutions over the generations, being the most selected, whereas view 4 had very low participation in the optimization process.

As can be seen in the statistics of Figure 3, this Problem is not complex enough to MVDE as the worst solution is close to the optimum value. The best reported rounded VTR (value-to-reach) for this Problem is 1.724852 (Coello Coello and Becerra, 2004; Mezura-Montes et al., 2006; He and Wang, 2007b). MVDE achieved a mean value close to 1.724862. A larger amount of generations could be sufficient to reach even better results, but the current configuration makes MVDE a very efficient approach.

Table 1 presents the comparison with state-of-the-art algorithms, which are among the best results reported in literature. MVDE presents a mean performance similar to that of the best approach found in literature (Mezura-Montes et al., 2006), but requiring almost 40% less function evaluations.

4.4.2. Problem 2: Speed Reducer

For this Problem, the best known rounded VTR reported in literature is 2994.471066 (Zhang et al., 2008). Other results found in literature that are lower than that were violating constraints
Table 1: Comparison of our results (MVDE) and results of some state-of-the-art approaches in the design of a welded beam. “NA” means not available.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.Dev.</th>
<th>Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVDE</td>
<td>1.7248527</td>
<td>1.7248621</td>
<td>1.7249215</td>
<td>7.88359e-6</td>
<td>15,000</td>
</tr>
<tr>
<td>(Coello Coello, 2000)</td>
<td>1.748309</td>
<td>1.771973</td>
<td>1.785835</td>
<td>0.011220</td>
<td>NA</td>
</tr>
<tr>
<td>(Coello and Mezura-Montes, 2002)</td>
<td>1.728226</td>
<td>1.792654</td>
<td>1.993408</td>
<td>0.074713</td>
<td>24,000</td>
</tr>
<tr>
<td>(Coello Coello and Becerra, 2004)</td>
<td>1.724852</td>
<td>1.971809</td>
<td>3.179709</td>
<td>0.443131</td>
<td>NA</td>
</tr>
<tr>
<td>(Mezura-Montes et al., 2006)</td>
<td>1.724852</td>
<td>1.724853</td>
<td>1.724854</td>
<td>1.0e-15</td>
<td>24,000</td>
</tr>
<tr>
<td>(He and Wang, 2007a)</td>
<td>1.728024</td>
<td>1.748831</td>
<td>1.782143</td>
<td>0.012926</td>
<td>&gt;30,000</td>
</tr>
<tr>
<td>(He and Wang, 2007b)</td>
<td>1.724852</td>
<td>1.749040</td>
<td>1.814295</td>
<td>0.040049</td>
<td>75,000</td>
</tr>
</tbody>
</table>

and were ignored.

The profile for this Problem is presented in Figure 4. In the first 5 generations two strategies that perform some sort of local-search were more strongly employed. At the same time, the strategies that explore the search-space were not very used. After a few generations MVDE started to look around by using rand/1 (view 1), and then the profile became similar to the one presented for Problem 1: view 1 explores the search-space whereas views 2 and 3 try to improve the population. It seems that views 4 and 5 may be employed to give some diversity in the population.

With 30,000 evaluations, MVDE was able to reach a mean of 2994.47106645397 (see summary in Figure 5), which corresponds to VTR, and the 3rd Quartile was also very close to VTR. Probably a few more generations could make MVDE reach VTR in all runs.

![Figure 4: Percentage of selection of each view for Problem 2.](image)

Table 2 presents the comparison with other state-of-the-art approaches. In this table, MVDE was the second-best performer algorithm because the mean was affected by the worst solutions found. However, the other approaches, including particle swarm optimization, artificial bee colony, and improved differential evolution, required more evaluations and achieved worse results. A different configuration for MVDE could lead to even better results.
Table 2: Comparison of results for the design of a speed reducer.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.Dev.</th>
<th>Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVDE</td>
<td>2994.471066</td>
<td>2994.471066</td>
<td>2994.471069</td>
<td>2.819316e-7</td>
<td>30,000</td>
</tr>
<tr>
<td>(Wang et al., 2009)</td>
<td>2994.516778</td>
<td>2994.585417</td>
<td>2994.630797</td>
<td>3.3e-2</td>
<td>40,000</td>
</tr>
<tr>
<td>(Zhang et al., 2008)</td>
<td>2994.471066</td>
<td>2994.471066</td>
<td>2994.471066</td>
<td>3.58e-12</td>
<td>30,000</td>
</tr>
<tr>
<td>(Liu et al., 2010)</td>
<td>2996.348167</td>
<td>2996.348174</td>
<td>2996.348204</td>
<td>6.4e-6</td>
<td>54,350</td>
</tr>
<tr>
<td>(Ray and Liew, 2003)</td>
<td>2994.744241</td>
<td>3001.758264</td>
<td>3009.964736</td>
<td>4.0</td>
<td>54,456</td>
</tr>
<tr>
<td>(Mezura-Montes et al., 2006)</td>
<td>2996.356689</td>
<td>2996.367220</td>
<td>2996.3890137</td>
<td>8.2e-3</td>
<td>24,000</td>
</tr>
<tr>
<td>(Cagnina et al., 2008)</td>
<td>2996.348165</td>
<td>2996.3482</td>
<td>NA</td>
<td>NA</td>
<td>24,000</td>
</tr>
<tr>
<td>(Brajevic et al., 2011)</td>
<td>2996.348165</td>
<td>2996.3482</td>
<td>NA</td>
<td>NA</td>
<td>240,000</td>
</tr>
</tbody>
</table>

Figure 6: Percentage of selection of each view for Problem 3.

Best: 2994.47106616197, 1st Quartile: 2994.47106627408, Median: 2994.4711395078993518837,

Best solution: x_1 = 3.50000000001147, x_2 = 0.7, x_3 = 17, x_4 = 7.3, x_5 = 7.715399115729, x_6 = 3.35021466610515, x_7 = 5.2866499022.

Figure 5: Summary statistics and best solution for Problem 2.

4.4.3. Problem 3: Three-bar truss

The best known value for Problem 3 in literature is 263.895843376468 (Liu et al., 2010), rounded to 263.89584338 (8-digit precision). In Figure 6 one can see the profile of MVDE’s views for this Problem. This profile is quite different from the previous two. While in Problems 1 and 2 the profile of each view was clearly separated from the others, in this case there is a mixture. Views 1, 2, and 3 present almost the same percentage of selection, corresponding to about 75%. On the other hand, views 4 and 5 show participation of 15% or less for each one. Another interesting point is that, for this Problem, view 3 was the most selected (view 1 was the most selected in Problems 1 and 2), ending with an increase for view 1 (exploration) and decrease for view 2 (local-search) in the last 5 iterations. This can be seen as an attempt to escape from the current region.

Figure 7 presents a summary of the results obtained by MVDE, and Table 3 shows the results of MVDE and of some state-of-the-art algorithms. MVDE found VTR with the same mean reported by two state-of-the-art approaches ((Liu et al., 2010) and (Kashan, 2011)), but using only 7,000
Table 3: Comparison of results for the design of a three-bar truss.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.Dev.</th>
<th>Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVDE</td>
<td>263.89584337</td>
<td>263.89584338</td>
<td>263.8958548</td>
<td>2.576062e-7</td>
<td>7,000</td>
</tr>
<tr>
<td>(Wang et al., 2009)</td>
<td>263.8958435</td>
<td>263.8966</td>
<td>263.90041</td>
<td>1.1e-3</td>
<td>17,000</td>
</tr>
<tr>
<td>(Gandomi et al., 2011)</td>
<td>263.97156</td>
<td>264.0669</td>
<td>NA</td>
<td>0.00009</td>
<td>15,000</td>
</tr>
<tr>
<td>(Kashan, 2011)</td>
<td>263.8958434</td>
<td>263.8958434</td>
<td>263.8958498</td>
<td>5.68e-14</td>
<td>10,000</td>
</tr>
<tr>
<td>(Zhang et al., 2008)</td>
<td>263.8958434</td>
<td>263.8958436</td>
<td>263.8958498</td>
<td>9.72e-7</td>
<td>15,000</td>
</tr>
<tr>
<td>(Ray and Liew, 2003)</td>
<td>263.8958466</td>
<td>263.9033</td>
<td>263.96975</td>
<td>1.257e-2</td>
<td>17,610</td>
</tr>
<tr>
<td>(Liu et al., 2010)</td>
<td>263.89584338</td>
<td>263.89584338</td>
<td>263.8958498</td>
<td>4.5e-10</td>
<td>17,600</td>
</tr>
</tbody>
</table>

evaluations, compared to 10,000 and 17,600 required by the other two. The standard deviation of MVDE's runs is a bit higher because in some runs MVDE was not able to reach VTR in time (see Worst column in Table 3). This could be solved by extending the running time of the algorithm. However, this procedure was not necessary to reach a high-quality mean solution, outperforming several state-of-the-art metaheuristics.

\[
\text{Best: 263.89584337, 1^{st} Quartile: 263.89584338, Median: 263.89584339, 3^{rd} Quartile: 263.89584340, Worst: 263.89584341.}
\]

\[
\text{Best solution: x1 = 0.78867513744977, x2 = 0.408248282388824.}
\]

Figure 7: Summary statistics and best solution for Problem 3.

4.4.4. Problem 4: Pressure Vessel

Mezura-Montes et al. (2006) report a VTR for this Problem as 6059.701660. With the current configuration MVDE was unable to find the VTR, getting trapped in a specific local optimum (with function value around 6059.714) from where it tried to escape without success. This behavior can be seen in the profile (Figure 8) where view 1 is strongly employed during all generations and view 2 (local-search) comes in third, with a smooth increase over the time. Similar increase is shown for view 3. View 5 has an important participation in the beginning but decreases until the end. View 4 is stable around 12% and could probably be discarded for this Problem.

Figure 8: Percentage of selection of each view for Problem 4.
The summary of the experiment is shown in Figure 9 along with the best solution found. The worst solution was a local optimum with function value close to 6090. As MVDE got stuck in this local optimum just a few times from 1000 runs, it did not present a large contribution to the mean.

In Table 4 the results of MVDE are compared with those from other approaches. The best known result was obtained by Mezura-Montes et al. (2006) and their approach found that best solution in all runs, which also gives the lowest mean of all six algorithms. In terms of mean value, MVDE was the second-best for a small difference, but performing 37.5% less function evaluations than Mezura-Montes’ approach. Nevertheless, MVDE can be considered very competitive when compared to the other algorithms presented in the comparison.

### 4.4.5. Problem 5: Tension/Compression Spring

For this Problem, VTR is defined as 0.012665 as shown in (Shen et al., 2009). In this paper, the results are presented with higher precision for greater accuracy.

The profile of the views in this Problem is exhibited in Figure 10. Views that perform local-search were more used in the first generations whereas the exploration was less employed. Then a peak of exploration occurred around generation 10 and stabilized by generation 15 to the end. At the same time, local-search decreased and then presented a small variance of use. All views present some stability after generation 13 and are well-distributed. View 1 was the most used, with more than 25% of selections and, once more, view 4 was the least selected in the tournament.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.Dev.</th>
<th>Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVDE</td>
<td>6059.714387</td>
<td>6059.99735696</td>
<td>6090.5335277869</td>
<td>2.9102896082</td>
<td>15,000</td>
</tr>
<tr>
<td>(Wang et al., 2009)</td>
<td>6059.7255</td>
<td>6061.9878</td>
<td>6090.8022</td>
<td>4.7e-0</td>
<td>30,000</td>
</tr>
<tr>
<td>(Coello Coello, 2000)</td>
<td>6288.7445</td>
<td>6293.84323196</td>
<td>6308.14965192</td>
<td>7.41328537</td>
<td>NA</td>
</tr>
<tr>
<td>(Akhtar et al., 2002)</td>
<td>6171.00</td>
<td>6335.05</td>
<td>6453.65</td>
<td>NA</td>
<td>20,000</td>
</tr>
<tr>
<td>(He and Wang, 2007a)</td>
<td>6061.077700</td>
<td>6147.133200</td>
<td>6363.804100</td>
<td>86.45450</td>
<td>32,500</td>
</tr>
<tr>
<td>(Mezura-Montes et al., 2006)</td>
<td>6059.701600</td>
<td>6059.701600</td>
<td>6059.701600</td>
<td>1.0e-12</td>
<td>24,000</td>
</tr>
<tr>
<td>(Shen et al., 2009)</td>
<td>6059.71400</td>
<td>6298.801000</td>
<td>6820.410000</td>
<td>194.3150</td>
<td>26,000</td>
</tr>
<tr>
<td>(Coello, 2010)</td>
<td>6059.7110</td>
<td>6464.8166</td>
<td>7544.4925</td>
<td>465.1386</td>
<td>8,000</td>
</tr>
</tbody>
</table>
Table 5: Comparison of results for the design of a tension/compression spring.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std.Dev.</th>
<th>Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVDE</td>
<td>0.01266527172</td>
<td>0.012667324</td>
<td>0.012719055</td>
<td>2.451838e-6</td>
<td>10,000</td>
</tr>
<tr>
<td>(Wang et al., 2009)</td>
<td>0.01266826198</td>
<td>0.012708075</td>
<td>0.012861375</td>
<td>4.5e-05</td>
<td>25,000</td>
</tr>
<tr>
<td>(Coello Coello, 2000)</td>
<td>0.0127048</td>
<td>0.012769</td>
<td>0.012822</td>
<td>3.939000e-5</td>
<td>NA</td>
</tr>
<tr>
<td>(Ray and Liew, 2003)</td>
<td>0.01266924934</td>
<td>0.012922669</td>
<td>0.016717272</td>
<td>5.9e-4</td>
<td>25,167</td>
</tr>
<tr>
<td>(He and Wang, 2007a)</td>
<td>0.0126747</td>
<td>0.012730</td>
<td>0.012924</td>
<td>5.1985e-5</td>
<td>32,800</td>
</tr>
<tr>
<td>(Mezura-Montes et al., 2006)</td>
<td>0.012665</td>
<td>0.012666</td>
<td>0.012674</td>
<td>2.0e-6</td>
<td>24,000</td>
</tr>
<tr>
<td>(Shen et al., 2009)</td>
<td>0.012665</td>
<td>0.012708</td>
<td>0.012994</td>
<td>5.1e-5</td>
<td>26,000</td>
</tr>
<tr>
<td>(Coelho, 2010)</td>
<td>0.012666</td>
<td>0.012996</td>
<td>0.015869</td>
<td>0.000628</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Figure 10: Percentage of selection of each view for Problem 5.

Figure 11 presents the statistics regarding this experiment. One can notice that VTR was reached at least one time (best) and that the mean value was close to it. Moreover, the worst value presents an error of 5.4e-05, which can be considered a small value. Using only 10,000 evaluations, MVDE reach VTR in a large percentage of runs, but a few bad runs increased the mean value enough to place MVDE as the second-lowest mean in Table 5, which contains results of other algorithms for comparison.

The worst result found by MVDE is better than the mean value of several approaches in the table. The best overall result was obtained by Mezura-Montes et al. (2006), but using more than the double number of evaluations. Allowing MVDE to perform more evaluations could improve the results. However, with the present configuration it is capable of finding high-quality solutions in low computational time, outperforming more elaborated algorithms.
5. Conclusions

In this paper, a novel approach for metaheuristic enhancement - called Multi-View - is proposed to improve Differential Evolution. The new algorithm originates the Multi-View Differential Evolution (MVDE). MVDE was developed to automatically balance exploration and exploitation of the search-space, improving the chances to escape from local optima, to finding high-quality solutions, and with fast convergence. Instead of using several populations, sub-populations, or co-evolution, the idea consists of employing different strategies to generate new vector solutions from the same population, which provides different views for the same problem.

MVDE was employed to solve five well-known engineering design problems and the results were compared to the results obtained by several state-of-the-art algorithms for constrained global optimization. Experiments were conducted on the design of: a Welded Beam; a Speed Reducer; a Three-bar truss; a Pressure Vessel; and a Tension/Compression Spring).

The profile plots show that strategies 4 and 5 presented very low contribution to the optimization process. They performed thousands of function evaluations but generated good solutions in lower frequency than the other strategies. This indicates that those two strategies could be removed or that the parameter setting for MVDE was not well-chosen for them. One possibility is to define different settings for each view, which may be investigated in future works.

Nevertheless, results show that MVDE achieved VTR in four of the five problems after a relatively small number of function evaluations. Moreover, the small values for standard deviations may indicate the robustness and stability of the proposed technique.

When compared to some newer and more sophisticated approaches MVDE found better solutions, with lower mean value (over 1000 runs) and standard deviation, and requiring less function evaluations. In terms of mean value and number of function evaluations, MVDE was the second-best performer of eight approaches in Problems 1, 2, 4, and 5, and the second-best of seven in Problem 3. It is important to notice that MVDE was executed 1000 times for each function to allow for a better distribution of the results found and more confident statistics. On the other hand, the algorithms used in comparison were executed only a few times (30 or 50 runs) - according to their authors - which could avoid local optima results and improve their statistics.

These overall results confirm that the proposed approach - MVDE - can adequately solve constrained engineering design problems. The final solutions are of high-quality and stable under a low computational effort, positioning MVDE as a top performer for these five engineering design problems.

To achieve better results with MVDE one could test a bigger population, change MVDE's parameters (F and CR), or use other mutation strategies. As commonly happens to metaheuristics, there are several ways to improve MVDE, opening various topics for future works.

6. Acknowledgments


