Differential Evolution.

By Luke Maurer and Pavel Govyadinov

Resources:


Grew out of Price and Storn’s attempt to solve the Chebychev Polynomial fitting problem, around 1994.

Simple, yet powerful. Finished 3rd at the 1st ICEO in Mayb 1996

Why not use the vector difference to perturb the vector population?
- DE does not use the gradient of the problem optimized.
- The problem does not need to be differentiable! Cool!
Definition.

\[ X_{i,g} = (x_{j,i,g}), \quad j = 0,1,2, \ldots, D - 1. \]

\[ P_{x,g} = (X_{i,g}), \quad i = 0,1, \ldots, N_p - 1, \]

\[ g = 0,1,2, \ldots, g_{\text{max}}. \]

Where: \( N_p \) is the number of population vectors,
\( g \) defines the generation counter (from 1 to \( g_{\text{max}}. \)
\( D \) is the dimensionality.

Initial Vector

\[ x_{j,i,0} = x_{j,min} + \text{rand}_{i,j}[0,1] \times (x_{j,min} - x_{j,max}) \]
Difference Vector aka Difference Based Mutation.

**DE, rand, 1, bin**

\[ V_{i,g} = X_{r_1,g} + F \times (X_{r_2,g} - X_{r_3,g}) \]

- \( V_{i,g} \) is called the donor vector.
- \( X_{r_1,g} \) is called the base vector.
- \( (X_{r_2,g} - X_{r_3,g}) \) is called a difference vector.
- In the rand implementation, \( X_{r_1,g}, X_{r_2,g}, X_{r_3,g} \) are all chosen randomly from our population.
- There are other implementations, that will be discussed later.
Diversity Enhancement aka Crossover.

\[ u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } (\text{rand}_j[0,1] \leq Cr) \\ x_{j,i,g} & \text{otherwise} \end{cases} \]

- \( u_{i,g} \) is the trial vector.
- \( v_{i,g} \) is the donor vector.
- \( x_{i,g} \) is the target vector.
- Cross over is done by mixing the parameters of \( v_{j,i,g} \) and \( x_{j,i,g} \).
- In order to prevent the case where the target vector is equal to the trial vector, at least one component is taken from the mutation vector \( V_{i,g} \).
Selection.

\[ x_{i,g+1} = \begin{cases} u_{i,g} & \text{if } f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} & \text{otherwise} \end{cases} \]

- One-to-one survivor selection where the trial vector competes against the target vector for the lowest objective function.
\[ V_{i,g} = X_{r_1,g} + F \times (X_{r_2,g} - X_{r_3,g}) \]
Pseudo-Code/Flow.

1) Choose target vector and base vector
2) Random choice of two population members
3) Compute weighted difference vector
4) Add to base vector
5) \( x_{0,g+1} = u_{0,g} \) if \( f(u_{0,g}) \leq f(x_{0,g}) \), else \( x_{0,g+1} = x_{0,g} \)

population \( P_{x,g} \)
parameter vector \( x_{Np-1,g} \)
objective function value \( f(x_{Np-1,g}) \)
mutant population \( P_{v,g} \)
new population \( P_{x,g+1} \)
DE/x/y/z
- X denotes how we choose the base vector.
- Y denotes the number of difference vectors (think minus signs).
- Z denotes the cross over method.

Only important parameters are:
- \( Cr = \) crossover rate.
- \( F = \) Scalar parameter to control evolution.
- \( Np = \) size of the population.
Perturbation of the Mutation Method.

\[ V_{i,g} = X_{\text{best},g} + F \left( X_{r_1,g} - X_{r_2,g} \right) \]

DE/best/1/bin

\[ V_{i,g} = X_{r_1,g} + F \left( X_{\text{best},g} - X_{r_1,g} \right) + F \left( X_{r_2,g} - X_{r_3,g} \right) \]

DE/current-to-best/1/bin

\[ V_{i,g} = X_{\text{best},g} + F \left( X_{r_1,g} - X_{r_2,g} + X_{r_3,g} - X_{r_4,g} \right) \]

DE/best/2/bin

\[ V_{i,g} = X_{r_1,g} + F \left( X_{r_1,g} - X_{r_2,g} - X_{r_3,g} \right) \]

DE/rand/2/dir
Example.

\[ f(x_1, x_2) = 3(1 - x_1)^2 \cdot \exp(x_1^2 + (x_2 + 1)^2) - 10 \left( \frac{x_1}{5} - x_1^3 - x_2^5 \right) \cdot \exp(x_1^2 + x_2^2) - \frac{1}{3} \cdot \exp((x_1 + 1)^2 + x_2^2) \]
Example Cont.
Parameters

- $NP$ — size of population
- $F$ — scaling factor for difference vectors
- $Cr$ — per-gene frequency of crossover
Parameters

- **NP** — size of population
  - Somewhere between $3D$ and $10D$ for $D$-dimensional problem
- **$F$** — scaling factor for difference vectors
  - Sensible range of values is $0.4 < F < 1$
  - Higher $F$ means bigger mutations
- **$Cr$** — per-gene frequency of extra crossover
  - $0 < Cr < 0.2$ if objective function is separable
  - $0.9 < Cr < 1$ otherwise
  - At $Cr = 1$, algorithm is rotationally invariant

Effect of $Cr = 0, 0.5, 1$
Adaptive DE

- SaDE
  - Adjust $F$ and $Cr$ over time for the whole population, tracking what values have been most successful

- jDE
  - Associate $F$ and $Cr$ values with each vector in the population
  - Before mutation, re-pick $F$ with probability $\tau_1$ and $Cr$ with probability $\tau_2$
  - Better values more likely to propagate to next generation
Randomized $F$

- Allows either exploration or exploitation to occur at any time
- Allows searching over larger areas while keeping $NP$ small
- Two possibilities
  - **Dither**: Add random noise to $F$ once for each mutation
  - **Jitter**: Add random noise to $F$ for each dimension in each mutation
Time-Varying $F$

- Let $F$ vary from 1 to 0.5 over time
- Encourages exploration at first, exploitation later
- Similar in purpose to simulated annealing
Arithmetic Recombination

- For each target vector $\vec{X}_i$, pick $k_i \in [0.1]$ randomly.
- After generating donor vector $\vec{V}_i$ as before, compute trial vector $\vec{U}_i$ by

$$\vec{U}_i = \vec{X}_i + k_i(\vec{V}_i - \vec{X}_i)$$

- Can explore more space by picking weight per component, but loses rotational invariance.
Mutation

\[ \vec{V}_i = \vec{X}_{r_1} + \frac{F}{2} (\vec{X}_{r_1} - \vec{X}_{r_2} - \vec{X}_{r_3}) \]

where \( f(X_{r_1}) \leq f(X_{r_2}), f(X_{r_3}) \)

- Use objective function to direct selection of mutation vectors, giving weight to the best one
Combined Mutation and Crossover

\[
\tilde{U}_i = \begin{cases} 
\vec{X}_{r_1} + F(\vec{X}_{r_2} - \vec{X}_{r_3}) & \text{with probability } p_F \\
\vec{X}_{r_0} + k(\vec{X}_{r_1} + \vec{X}_{r_2} - 2\vec{X}_{r_0}) & \text{otherwise}
\end{cases}
\]

- Randomly generate either a pure recombinant or a pure mutant
- Recommended \( k = 0.5(F + 1) \)
- Works well on a variety of functions
SaDE also selects among four different strategies:
- DE/rand/1/bin
- DE/rand-to-best/2/bin
- DE/rand/2/bin
- DE/current-to-rand/1 (with arithmetic crossover)

Strategies that produce more successful offspring are more likely to be chosen in successive generations.
Comparative Analysis

- From Mezura-Montes et al. 2006

- For unimodal, separable functions:

<table>
<thead>
<tr>
<th>Variant</th>
<th>f01</th>
<th>f02</th>
<th>f04</th>
<th>f06</th>
<th>f07</th>
</tr>
</thead>
<tbody>
<tr>
<td>rand/1/bin</td>
<td>0.0</td>
<td>0.0</td>
<td>1.9521</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>rand/1/exp</td>
<td>0.0</td>
<td>0.0</td>
<td>3.7584</td>
<td>0.843360</td>
<td>0.0</td>
</tr>
<tr>
<td>best/1/bin</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0017</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>best/1/exp</td>
<td>407.972</td>
<td>3.291</td>
<td>1.701872</td>
<td>2737.8458</td>
<td>0.070545</td>
</tr>
<tr>
<td>current-to-best/1</td>
<td>0.54148</td>
<td>4.842</td>
<td>4.233736</td>
<td>1.394</td>
<td>0.0</td>
</tr>
<tr>
<td>current-to-rand/1</td>
<td>0.69966</td>
<td>3.503</td>
<td>3.298563</td>
<td>1.767</td>
<td>0.0</td>
</tr>
<tr>
<td>current-to-rand/1/bin</td>
<td>0.0</td>
<td>0.0</td>
<td>0.149514</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>rand/2/dir</td>
<td>0.0</td>
<td>0.0</td>
<td>0.044199</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
For unimodal, nonseparable functions ($f_{03}$) and multimodal, separable functions ($f_{08}$ and $f_{09}$):

<table>
<thead>
<tr>
<th>Variant</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_{03}$</td>
</tr>
<tr>
<td>rand/1/bin</td>
<td>0.024305</td>
</tr>
<tr>
<td>rand/1/exp</td>
<td>0.000004</td>
</tr>
<tr>
<td>best/1/bin</td>
<td>0.0</td>
</tr>
<tr>
<td>best/1/exp</td>
<td>10.607806</td>
</tr>
<tr>
<td>current-to-best/1</td>
<td>0.471730</td>
</tr>
<tr>
<td>current-to-rand/1</td>
<td>0.903563</td>
</tr>
<tr>
<td>current-to-rand/1/bin</td>
<td>0.000232</td>
</tr>
<tr>
<td>rand/2/dir</td>
<td>30.112881</td>
</tr>
</tbody>
</table>
### Comparative Analysis

- For multimodal, nonseparable functions:

<table>
<thead>
<tr>
<th>Variant</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_{05}$</td>
</tr>
<tr>
<td>rand/1/bin</td>
<td>19.577895</td>
</tr>
<tr>
<td>rand/1/exp</td>
<td>6.696064</td>
</tr>
<tr>
<td>best/1/bin</td>
<td>30.390870</td>
</tr>
<tr>
<td>best/1/exp</td>
<td>132621.5</td>
</tr>
<tr>
<td>current-to-best/1</td>
<td>30.984666</td>
</tr>
<tr>
<td>current-to-rand/1</td>
<td>31.702063</td>
</tr>
<tr>
<td>current-to-rand/1/bin</td>
<td>24.260535</td>
</tr>
<tr>
<td>rand/2/dir</td>
<td>30.654916</td>
</tr>
</tbody>
</table>
Comparative Analysis

- best/1/bin, rand/1/bin, rand/2/bin, and current-to-rand/1/bin are best among those tried
- */*/exp is bad
- best/1/bin only weak when problem is both multimodal and non-separable
- rand/2/dir performs well everywhere, but especially on multimodal, non-separable problems