Differential Evolution Made Easy

Technical report no. 2005-01

Rasmus K. Ursem
`ursem@stofanet.dk`

Notes of use: Please cite my paper [15] if you use this work in your own research.

1 Introduction

Optimization techniques for numerical problems are very well-explored and several hundred algorithms exist. Despite more than four decades of extensive research and thousands of comparative studies *no* algorithm has yet proven to be the overall best. In fact, Wolpert and Macready have shown that no algorithm is superior on all problems [17]. Although this is a strong proof, it is merely of theoretical interest – all problems literally means *all problems*, including those yielding an arbitrary and determinstic function value, i.e., problems with no correlation between neighboring solutions. Naturally, no algorithm is better than random search on such problems...

As mentioned, four decades of research has not yet nominated one single algorithm as the overall best. However, in 1995 Storn and Price suggested the "Differential Evolution" (DE) algorithm [12]. Since then, the algorithm has been tested on numerous artificial and real problems and out to be a strong candidate for the title [2; 3; 4; 6; 11; 14; 16; 18; 20].

Published work, work by colleagues, and my own recent experiences with DE on motor system identification [15], has convinced me to always try this algorithm first.

2 Differential Evolution Algorithm

As other evolutionary algorithms, DE maintains a population (a set) of solutions to the optimization problem at hand. The main idea in DE is to use vector differences in the creation of new candidate solutions, whereas traditional EAs rely on random pertubation (mutation) of a solution and mixing of two or more solutions (recombination). Another major difference is that the three phases of a standard EA (selection, recombination, and mutation) are combined to one operation, which is carried out for each individual. In the standard EA, each phase is performed on the entire population. In contrast, the DE algorithm iterates through the population and creates, for each population index i, a potential candidate $C[i]$ by vector addition (mutation) and a variant of uniform crossover (recombination). Selection is straightforward and very simple; the candidate solution $C[i]$ replaces $P[i]$ if it is better. Figure 1 illustrates the DE algorithm.

The key to success in DE is the creation of the candidate solution $C[i]$. Until now, several schemes have been suggested (for variants, see [12] and [13]). Storn and Price suggest a two-point crossover scheme where a number of consecutive genes are copied from the parent $P[i]$. I have experimented with a uniform crossover variant described by the pseudocode in figure 2, which has shown good performance in my previous work [15].

Figure 3 illustrates a two-dimensional example where the objective is to determine the parameters X1 and X2.

In the general case of N-dimensional problems, the final candidate $C[i]$ will be a corner in the hypercube spanned by $P[i]$ and the initial candidate $C_1[i]$.

The algorithm is quite robust with respect to the algorithmic parameters f and p_c. Setting $f = 0.35$ and $p_c = 0.2$ will give generally good convergence on a wide range of problems.

```
Differential Evolution
  Initialize and evaluate population $P$
  while (not done) {
    for ($i = 0$ ; $i < ps$ ; $i{+}{+}$) {
      Create candidate $C[i]$
      Evaluate $C[i]$
      if ($C[i]$ is better than $P[i]$)
        $P'[i] = C[i]$
      else
        $P'[i] = P[i]$
    }
    $P = P'$
  }
```

Figure 1: Pseudocode for DE. ps is the population size, P is the population of the current generation, and P' is the population to be formed for the next generation. The routine ***Create candidate*** *$C[i]$ is listed in figure 2.*

```
Create candidate $C[i]$
  Randomly select parents $P[i_1]$, $P[i_2]$, and $P[i_3]$ where $i$, $i_1$, $i_2$, and $i_3$ are different.
  Create initial candidate $C_1[i] = P[i_1] \oplus f \odot (P[i_2] \ominus P[i_3])$.
  Create final candidate $C[i]$ by crossing over the genes of $P[i]$ and $C_1[i]$ as follows:
  for ($j = 0$ ; $j < N$ ; $j{+}{+}$) {
    if ($U(0,1) < p_c$)
      $C[i][j] = C_1[i][j]$
    else
      $C[i][j] = P[i][j]$
  }
```

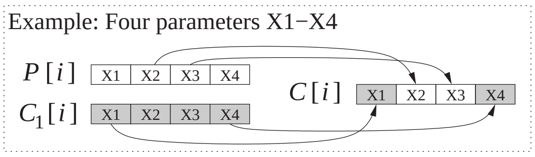


Figure 2: Pseudocode for the creation procedure in DE. $C[i]$ is the candidate solution with population index i, $C[i][j]$ is the j'th entry in the solution vector of $C[i]$, N is the problem dimensionality, $U(0,1)$ is a uniformly distributed number between 0 and 1, p_c is the probability of crossover, and f is the scaling factor. The operators $\oplus$ and $\ominus$ denote vector addition and subtraction, and $\odot$ is scalar multiplication. The example in the box illustrates the creation of the final candidate for a four-dimensional problem ($N = 4$). The values for X1 and X4 come from the initial candidate $C_1[i]$. X2 and X3 are from the parent $P[i]$.

3 Multiobjective Differential Evolution

The multiobjective DE is a bit more complex and research is still being done to adapt the algorithm to such problems [1; 7; 8; 10; 18; 19].

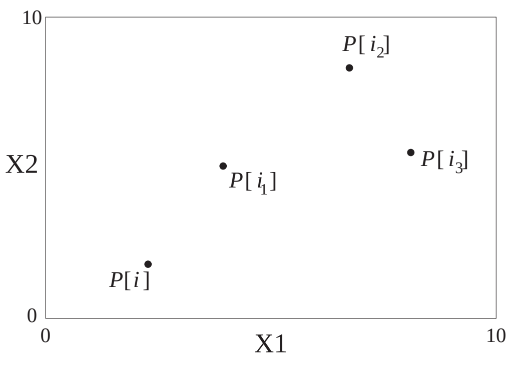
A simple and relatively good approach is to use a weighting scheme in case the candidate and the parent does not dominate each other. In this case, the algorithm is as displayed in figure 4.

For more further information on multiobjective optimization, please refer to Deb's book [5].

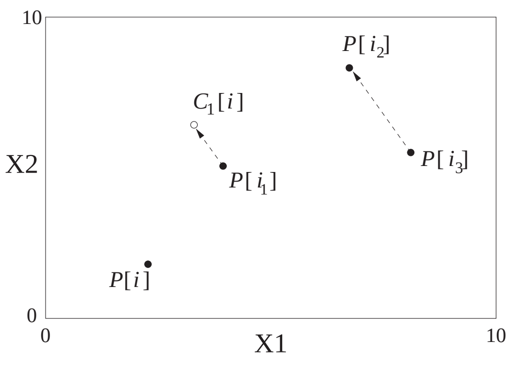
4 Constraint-handling Differential Evolution

No special extensions of the algorithm is necessary to make it suitable for handling constraints. Most constraint problems can be handled by the penalty method, which works as follows. Assume you have $p \geq 0$ inequalities and $q - p \geq 0$ equalities:

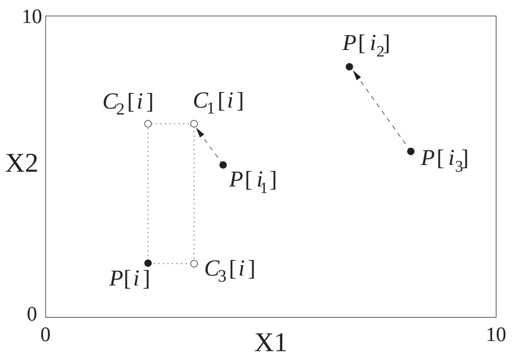
$$\begin{aligned} g_j(X) &\leq 0 \qquad & j &= 1, 2, \ldots, p \\ h_j(X) &= 0 \qquad & j &= p+1, p+2, \ldots, q \end{aligned}$$



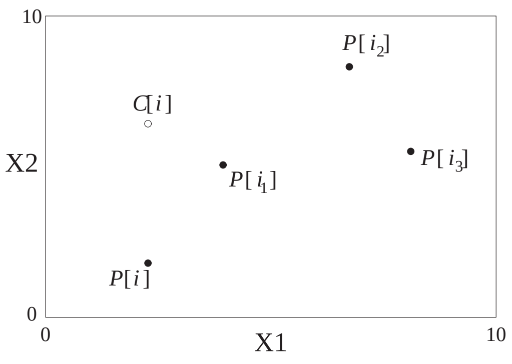
(a) Select parents $P[i_1]$, $P[i_2]$, and $P[i_3]$.



(b) Create initial candidate $C_1[i]$.



(c) All potential candidates $C_1[i]$, $C_2[i]$ and $C_3[i]$.



(d) Final candidate $C[i]$ (equal to $C_2[i]$.)

Figure 3: Example of creation procedure for one candidate on a two-dimensional problem. Subfigure (c) illustrates the three possible candidates that can be created by the for-loop in figure 2. In principle, $P[i]$ can also be generated, but this is prevented by the algorithm.

A measure of the constraint violation is often useful when handling constraints. A straightforward measure of constraint j's violation is:

$$f_j(X) = \begin{cases} \max\left(0, g_j(X)\right) & 1 \leq j \leq p \\ |h_j(X)| & p+1 \leq j \leq q \end{cases} \tag{1}$$

The weighted sum of constraint violation is often used as a measure of a solution's feasibility.

$$CV(X) = \sum_{j=1}^{q} w_j \cdot f_j(X) \tag{2}$$

The sum of all constraint violations is zero for feasible solutions and positive when at least one constraint is violated. For a minimization problem, the constraint violation is added to the fitness, whereas it is subtracted for a maximization problem.

An obvious application of the constraint violation is to use it to guide the search towards feasible areas of the search space. There was quite a lot of work on such ideas and other constraint techniques in the EA-community during the 1990's. A summary of these techniques can be found in Michalewicz's and Fogel's book [9], which also contains information on many other stochastic techniques.

Hope you'll find the information useful.

/Rasmus K. Ursem

```
**Differential Evolution**
  Initialize and evaluate population $P$
  **while** (not done) {
    **for** ($i = 0$ ; $i < ps$ ; $i$++) {
      **Create candidate** $C[i]$
      Evaluate $C[i]$
      **if** ($C[i]$ dominates $P[i]$)
        $P'[i] = C[i]$
      **elseif** (Neither dominates the other) {
        Calculate weighted sum fitness for candidate $F(C[i]) = \sum_{k=1}^{M} w_k f_k(C[i])$
        Calculate weighted sum fitness for parent $F(P[i]) = \sum_{k=1}^{M} w_k f_k(P[i])$
        **if** ($F(C[i]) < F(P[i])$)
          $P'[i] = C[i]$
        **else**
          $P'[i] = P[i]$
      }
      **else**
        $P'[i] = P[i]$
    }
    $P = P'$
  }
```

Figure 4: Pseudocode for the multiobjective DE. ps is the population size, P is the population of the current generation, P' is the population to be formed for the next generation, w_k is the weight for objective k and f_k is the k'th objective. The routine ***Create candidate*** *$C[i]$ is listed in figure 2.*

References

[1] Abbass, H. A., Sarker, R., and Newton, C. (2001). PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 971–978.

[2] Babu, B. V. and R.Angira (2001). Optimization of Thermal Cracker Operation using Differential Evolution. In *Proceedings of International Symposium and 54th Annual Session of IIChE (CHEMCON-2001).*

[3] Babu, B. V. and S.A.Munawar (2000). Differential Evolution for the Optimal Design of Heat Exchangers. In *Proceedings of All India Seminar on Chemical Engineering Progress on Resource Development - A Vision 2010 and Beyond.*

[4] Chang, C. S., Lu, L. R., and Wang, F. (2003). Application of differential evolution for harmonic worst-case identification of mass rapid transit power supply system. In Sarker et al., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 593–599.

[5] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons.

[6] Kilkki, J., Huapeng, W., and Handroos, H. (2001). Applying The Differential Evolution Algorithm To The Optimisation of Redundant Parallel Manipulator. In Giannakoglou et al., editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 223–228.

[7] Kukkonen, S. and Lampinen, J. (2004). An Extension of Generalized Differential Evolution for Multi-Objective Optimization with Constraints. In Yao et al., editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *LNCS*, pages 750–759.

[8] Madavan, N. (2002). Multiobjective Optimization Using a Pareto Differential Evolution Approach. In Fogel et al., editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1145–1150.

[9] Michalewicz, Z. and Fogel, D. B. (2000). *How to Solve It: Modern Heuristics*. Springer.

[10] Sarker, R. and Abbass, H. A. (2004). Differential Evolution for Solving Multiobjective Optimization Problems. *Asia-Pacific Journal of Operational Research*, 21(2):225–240.

[11] Sastry, K. K. N., Behera, L., and Nagrath, I. J. (1999). Differential Evolution Based Fuzzy Logic Controller for Nonlinear Process Control. *Fundamenta Informaticae*, 37(1/2):121–136.

[12] Storn, R. and Price, K. (1995). Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, International Computer Science Institute, Berkley.

[13] Storn, R. and Price, K. (1997). Differential evolution a simple and efficient heuristic for global optimisation over continuous spaces. *Journal of Global Optimization*, 11:341–359.

[14] Thomsen, R. (2003). Flexible ligand docking using differential evolution. In Sarker et al., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 2354–2361.

[15] Ursem, R. K. and Vadstrup, P. (2003). Parameter Identification of Induction Motors using Differential Evolution. In *Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003)*, pages 790–796.

[16] Vesterstroem, J. and Thomsen, R. (2004). A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 1980–1987.

[17] Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

[18] Xue, F., Sanderson, A. C., and Graves, R. J. (2003). Multi-Objective Differential Evolution and Its Application to Enterprise Planning. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'03)*, volume 3, pages 3535–3541.

[19] Xue, F., Sanderson, A. C., and Graves, R. J. (2003). Pareto-based multi-objective differential evolution. In Sarker et al., editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 862–869.

[20] Yang, J.-M., Horng, J.-T., and Kao, C.-Y. (2001). Integrating Adaptive Mutations and Family Competition with Differential Evolution for Flexible Ligand Docking. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 473–480.