
Presented by David Craft
September 15, 2003

The highlights of Differential Evolution (DE)

A population of solution vectors are successively updated by addition, subtraction, and component swapping, until the population converges, hopefully to the optimum.

No derivatives are used.

Very few parameters to set.

A simple and apparently very reliable method.
DE: the algorithm

Start with $NP$ randomly chosen solution vectors.

For each $i$ in $(1, \ldots NP)$, form a ‘mutant vector’

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3})$$

Where $r_1$, $r_2$, and $r_3$ are three mutually distinct randomly drawn indices from $(1, \ldots NP)$, and also distinct from $i$, and $0 < F \leq 2$. 
DE: forming the mutant vector

\[ v_i = x_{r1} + F \cdot (x_{r2} - x_{r3}) \]
DE: From old points to mutants
DE: Crossover $x_i$ and $v_i$ to form the trial vector

Possible trial vectors
DE: Crossover $x_i$ and $v_i$ to form the trial vector $u_i$

\[
x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})
\]
\[
v_i = (v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5})
\]
\[
u_i = (__, __, __, __, __)
\]

For each component of vector, draw a random number in $U[0,1]$. Call this $\text{rand}_j$. Let $0 \leq CR < 1$ be a cutoff. If $\text{rand}_j \leq CR$, $u_{ij} = v_{ij}$, else $u_{ij} = x_{ij}$.

To ensure at least some crossover, one component of $u_i$ is selected at random to be from $v_i$. 
DE: Crossover $x_i$ and $v_i$ to form the trial vector $u_i$

$x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})$

$v_i = (v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5})$

So, for example, maybe we have

$u_i = (v_{i1}, x_{i2}, x_{i3}, x_{i4}, v_{i5})$

Index 1 randomly selected as definite crossover

rand$_5$ <= CR, so it crossed over too
DE: Selection

If the objective value \( \text{COST}(u_i) \) is lower than \( \text{COST}(x_i) \), then \( u_i \) replaces \( x_i \) in the next generation. Otherwise, we keep \( x_i \).
Numerical verification

Much of the paper is devoted to trying the algorithm on many functions, and comparing the algorithm to representative algorithms of other classes. These classes are:

• Annealing algorithms
• Evolutionary algorithms
• The method of stochastic differential equations

Summary of tests: *DE is the only algorithm which consistently found the optimal solution, and often with fewer function evaluations than the other methods.*
Numerical verification: example

The fifth De Jong function, or “Shekel’s Foxholes”

(See equation 10 on page 348 of the *Differential Evolution* paper.)
The rest of the talk…

• Why is DE good?

• Variations of DE.

• How do we deal with constraints?

• An example from electricity load management.
Why is DE good?

• Simple vector subtraction to generate ‘random’ direction.
• More variation in population (because solution has not converged yet) leads to more varied search over solution space.

\[ \Delta = (x_{r2} - x_{r3}) \]  
[discuss: size and direction]

• Annealing versus “self-annealing”.
Variations of DE

$x_{r1}$: instead of random, could use **best**

$(x_{r2}-x_{r3})$: instead of single difference, could use more vectors, for more variation.

for example $(x_{r2}-x_{r3}+x_{r4}-x_{r5})$

Crossover: something besides bernoulli trials…
Dealing with constraints

• Penalty methods for ‘difficult’ constraints.
• Simple projection back to feasible set for $l \leq x \leq u$ type constraints.
• Or, random value $U[l, u]$ (when, why?)
Example: Appliance Job Scheduling

Hourly electricity prices (cents/kWh):

Power requirements for 3 different jobs (kW):

Start time constraints.
Example: Appliance Job Scheduling

Objective: find start times for each job which minimize cost.

Cost includes a charge on the maximum power used throughout the day. \textit{This couples the problems!}

\[
\min \sum_{i=1}^{J} t_i(x_i) + D(x)
\]

s.t. \(a_i \leq x_i \leq u_i\) \(i=1,\ldots,J\)

where

\[
t_i(x_i) = \int_{x_i}^{x_i + l_i} p(t)e_i(t, x_i) \, dt\quad \text{Cost of job } i \text{ started at time } x_i
\]

\[
D(x) = r \cdot \max_{t\in[0,T]} \sum e_i(t, x_i)\quad \text{Demand charge}
\]
Convergence for different F

Other settings: CR=0.3, NP=6
Appliance Job Scheduling: Solution

Solution

Total energy profile

Electricity price over time
Wrap-up

• DE is widely used, easy to implement, extensions and variations available, but no convergence proofs.

• More information:
  DE homepage: practical advice (e.g. start with NP=10*D and CR=0.9, F=0.8), source codes, etc.
  http://www.icsi.berkeley.edu/~storn/code.html

  http://www.lut.fi/~jlampine/debiblio.htm