University of Ljubljana, Faculty of Computer and Information Science

Differential evolution and its variants



Prof Marko Robnik-Šikonja

Analysis of Algorithms and Heuristic Problem Solving Version 2025

Idea of differential evolution

- Storn and Price, 1997
- Metaheuristic
- Optimization in R^a
- No need for gradient vector
- Combines ideas from evolutionary computation

Some slides taken from Hossein Talebi and Hassan Nikoo.

Template of evolutionary program

generate a population of agents (objects, data structures)
do {

compute fitness (quality) of the agents select candidates for the reproduction using fitness create new agents by combining the candidates replace old agents with new ones

} while (not satisfied)

• immensely general -> many variants

Crossover

- Single point/multipoint
- Shall preserve individual objects

Crossover: bit representation

Parents:11010111000111000101Children:11010101010111001100

A fitness function



Crossover: vector representation

Simplest form Parents: (6.13, 4.89, 17.6, 8.2) (5.3, 22.9, 28.0, 3.9) Children: (6.13, 22.9, 28.0, 3.9) (5.3, 4.89, 17.6, 8.2)

In reality: linear combination of parents

Linear crossover

- The linear crossover simply takes a linear combination of the two individuals.
- Let $x = (x_1, ..., x_N)$ and $y = (y_1, ..., y_N)$
- Select *α* in *(0, 1)*
- The results of the crossover is $\alpha x + (1 \alpha)y$.
- Possible variation: choose a different α for each position.

Mutation

- Adding new information
- Random search?
- Binary representation: 0111001100 --> 0011001100
- Single point/multipoint
- Lamarckian (searching for locally best mutation)

Gaussian mutation

- When mutating one gene, selecting the new value by choosing uniformly among all the possible values is not the best choice (empirically).
- The mutation selects a position i in the vector of floats and mutates it by adding a Gaussian error: a value extracted according to a normal distribution with mean 0 and variance depending on the problem.

DE introduction

- The original DE was developed for continuous value problems
- Individuals are vectors x_i , $i \in [1., n_s]$ of dimension n_s
- Distance and direction information from current population is used to guide the search process

DE background

- DE/rand/1
- Emphasizes mutation but still uses cross-over
- Generate trial vectors (u, mutant, donor) using the following formula:

$$u_i = x_{r1} + \beta \ (x_{r2} \ - x_{r3})$$

• Self-organizing ability

Illustration



Difference of DE with other EAs

- 1. Mutation is applied first to generate trial vectors, then cross-over is applied to produce offspring
- 2. Mutation step size is not sampled from prior known PDF (probability density function), it is influenced by difference between individuals of the current population

Difference Vector

- Positions of individuals provide valuable information about fitness landscape.
- At first, individuals are distributed over the search space, and over the time they converge to the same solution
- Differences are large in the beginning of evolution; therefore, we have bigger step size (exploring)
- Differences are smaller at the end of search process; therefore we have smaller step size (exploiting)

DE mutation

- Mutation produces a trial vector for each individual
- This trial vector is then used by the crossover operator to produce offspring
- For each parent $x_i(t)$, we make a trial vector $u_i(t)$

🛪 Target vector

$$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t))$$

Weighted Differential

$$i_2, i_3 \sim U(1, n_s)$$

$$\beta \in (0, \infty)$$

$$i \neq i_1 \neq i_2 \neq i_3$$

General classical DE Algorithm

Set the generation counter, t = 0; Initialize the control parameters, β and p_r ; Create and initialize the population, $\mathcal{C}(0)$, of n_s individuals; while stopping condition(s) not true do for each individual, $\mathbf{x}_i(t) \in \mathcal{C}(t)$ do Evaluate the fitness, $f(\mathbf{x}_i(t))$; Create the trial vector, $\mathbf{u}_i(t)$ by applying the mutation operator; Create an offspring, $\mathbf{x}'_{i}(t)$, by applying the crossover operator; if $f(\mathbf{x}'_{i}(t))$ is better than $f(\mathbf{x}_{i}(t))$ then Add $\mathbf{x}'_{i}(t)$ to $\mathcal{C}(t+1)$; end $\mathbf{u}_{i}(t) = \mathbf{x}_{i_{1}}(t) + \beta(\mathbf{x}_{i_{2}}(t) - \mathbf{x}_{i_{3}}(t))$ else Add $\mathbf{x}_i(t)$ to $\mathcal{C}(t+1)$; $x_{ij}^{'}(t) = \begin{cases} u_{ij}(t) & \text{if } j \in \mathcal{J} \\ x_{ij}(t) & \text{otherwise} \end{cases}$ end end end

Return the individual with the best fitness as the solution;

Geometrical Illustration (mutation)



Crossover

• DE crossover is a recombination of trial vector $u_i(t)$ and parent vector $x_i(t)$ to produce offspring $x_i'(t)$

$$x_{ij}'(t) = \begin{cases} u_{ij}(t) & \text{if } j \in \mathcal{J} \\ x_{ij}(t) & \text{otherwise} \end{cases}$$

Methods to determine ${\cal J}$

- Binomial crossover:
- we add at least one, each dimension is independent from others

 $j^* \sim U(1, n_x);$ $\mathcal{J} \leftarrow \mathcal{J} \cup \{j^*\};$ for each $j \in \{1, \dots, n_x\}$ do if $U(0, 1) < p_r$ and $j \neq j^*$ then $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\};$ end end

Methods to determine ${\cal J}$

- Exponential crossover:
- we add a connected sequence of dimensions (good for e.g., permutations)

$$\mathcal{J} \leftarrow \{\}; \\ j \sim U(0, n_x - 1); \\ \textbf{repeat} \\ \mathcal{J} \leftarrow \mathcal{J} \cup \{j + 1\}; \\ j = (j + 1) \bmod n_x; \\ \textbf{until } U(0, 1) \geq p_r \text{ or } |\mathcal{J}| = n_x; \end{cases}$$

Geometrical Illustration (crossover)



Selection

- For mutation to make the trial vector, it selects
 - A random individual
 - A target vector
 - The best individual
 - One of the best individuals
- Selection between a parent and offspring for the next generation
 - The better survives

Control Parameters

Scaling factor β also called F $\beta \in (0,\infty)$

$$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t))$$

- The smaller the value of $\boldsymbol{\beta}$ the smaller the step size
- Shall be small enough to allow differentials to exploit tight valleys, and large enough to maintain diversity.
- Empirical results suggest that β =0.5 generally provides good performance

Control Parameters

Recombination probability p_r also called P_{CR} (crossover probability)

- The higher p_r the more variation is introduced in the new population
- Increasing p_r often results in faster convergence, while decreasing p_r increases search robustness

Notation DE/x/y/z

- x the vector to be mutated
 - rand (randomly choosen from population)
 - best (from current population)
 - current-to-best (linear combination of current and best)
 - pbest (one of the best, randomly selected
- y the number of difference vectors used, i.e. 1 or 2, or more
- z the crossover scheme:
 - bin binomial (nearly binomial distribution of selected components from donors in random selection from U(0,1) < P_{CR})
 - exp exponential (selection of components from donor following the random dimension from 1..a, and additional number of components which is a random number from 1..a (circular); useful when nearby components are related)
 - arithmetic recombination $u_i = x_i + k_i (v_i x_i)$, k_i being the same for all components (line recombination) or different for each component
- We previously discussed: DE/rand/1/bin
- population size: typically between 5d and 10d, some variants use dynamic reduction of population size

DE/best/1/z

- Target vector is the best individual in current population $\widehat{x(t)}$,
- One differential vector is used.
- Any of the crossover methods.

$$\mathbf{u}_i(t) = \hat{\mathbf{x}}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t))$$

 $\mathbf{DE}/x/n_v/z$

- Any method for the target vector selection
- More than one difference vector
- Any of the crossover methods

$$\mathbf{u}_{i}(t) = \mathbf{x}_{i_{1}}(t) + \beta \sum_{k=1}^{n_{v}} (\mathbf{x}_{i_{2},k}(t) - \mathbf{x}_{i_{3},k}(t))$$

• The larger the value of n_v , the more directions can be explored per generation.

$\mathbf{DE}/\mathbf{rand-to-best}/n_v/z$

- $x_{i_1}(t)$ is randomly selected
- The closer γ is to 1, the more greedy the search process
- Value of γ close to 0 favors exploration.

$$\mathbf{u}_i(t) = \gamma \hat{\mathbf{x}}(t) + (1 - \gamma) \mathbf{x}_{i_1}(t) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_2,k}(t) - \mathbf{x}_{i_3,k}(t))$$

$\mathbf{DE}/\mathbf{current-to-best}/1 + n_v/z$

- At list two difference vectors.
- 1. Calculated from the best vector and the parent vector
- 2. While the rest of the difference vectors are calculated using randomly selected vectors

$$\mathbf{u}_i(t) = \mathbf{x}_i(t) + \beta(\hat{\mathbf{x}}(t) - \mathbf{x}_i(t)) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_1,k}(t) - \mathbf{x}_{i_2,k}(t))$$

• Empirical studies have shown **DE/current-to-best/2/bin** shows **good convergence characteristics**

Popular mutation strategies

• DE/rand/1

$$u_i = x_{r1} + \beta \ (x_{r2} \ - x_{r3})$$

• DE/best/1

$$u_i = x_{best} + \beta \ (x_{r1} \ - x_{r2})$$

- DE/current-to-best/1 $u_i = x_i + \beta(x_{best} - x_i) + \beta(x_{r1} - x_{r2}), \beta < 1$
- DE/best/2

$$u_i = x_{best} + \beta(x_{r1} - x_{r2}) + \beta(x_{r3} - x_{r4})$$

• DE/rand/2

$$u_i = x_{r1} + \beta(x_{r2} - x_{r3}) + \beta(x_{r4} - x_{r5})$$

Hybridization of DE

- Combinations with Particle Swarm Optimization (PSO):
 - Mixtures of populations
 - Mixtures of PSO and DE runs
- Combinations with Genetic Algorithms (GA):
 - Applying GA mutation or Gaussian noise
 - Applying DE mutation and/or crossover in GA
 - Rank based selection in DE
 - Etc.
- Dynamic parameter tuning: dynamic decrease, dependence on fitness, etc.

Many application

- Multiprocessor synthesis
- Neural network learning
- Synthesis of modulators
- Heat transfer parameter estimation
- Radio network design

• ...

DE for discrete problems

- For integers: round continuous values
- For binary discrete problems (bit vectors)
 - Limit continuous values to [0, 1] and treat them as probabilities
 - Use angle modulation
- For constraints: use penalization in objective function

Angle modulation DE 1/2

• Bit generator function, example

 $g(x) = \sin(2\pi(x-a) \times b \times \cos(2\pi(x-a) \times c)) + d$

a = horizontal shift b = maximum frequency of sin c=maximum frequency of cos d=verical shift

a=0, b=1, c=1, d=0

x=[-2, 2] sample at equal width points if value > 0, set bit to 1, otherwise to 0



Angle modulation DE 2/2

• use DE to learn good a, b, c, and d

Generate a population of 4-dimensional individuals;

repeat

Apply any DE strategy for one iteration;

 $\mathbf{for} ~\mathit{each}~\mathit{individual}~\mathbf{do}$

Substitute evolved values for coefficients a, b, c and d into equation

Produce n_x bit-values to form a bit-vector solution;

Calculate the fitness of the bit-vector solution in the original bit-valued space; \mathbf{end}

until a convergence criterion is satisfied;

Modern DE variants

- Idea: record history of successful parameters ($\beta, p_r,$ and $n_s)$ and sample from it for future generations
- SHADE (success-history based adaptive differential evolution)
- Uses DE/current-to-pbest/1/bin DE-strategy, archive A, and an adaptation of control parameters $\beta, \, p_r,$
- DE/current-to-pbest/1

$$u_{i} = x_{i} + \beta \left(x_{pbest} - x_{i} \right) + \beta \left(x_{r1} - x_{r2} \right), \beta < 1$$

- Where point x_{pbest} is randomly chosen from p*100% of best points
- Archive A is initialized as an empty set
- Each point x_i , which is replaced by its better trial point u_i , is included into archive A during the search process.
- The archive A is adjusted after each generation to have maximal size of n_s , where members removed from A are chosen randomly
- x_{pbest} is chosen from population P, x_{r1} and x_{r2} are chosen from the union of P and A
- L-SHADE (success-history based adaptive differential evolution with linear reduction of population size)

L-SHADE pseudocode

- M_{CR} and M_F are memories of successful P_{CR} and F (i.e. β) parameters
- S_{CR} and S_F are temporal memories of P_{CR} and F parameters
- H is history size

1: $q \leftarrow 1$, Archive $\mathbf{A} \leftarrow \emptyset$ 2: Initialize population $\mathbf{P}_g = (\vec{x}_{i,g}, \dots, \vec{x}_{NP,g})$ randomly 3: Set all values in M_{CR} , M_F to 0.5; 4: $k \leftarrow 1 \parallel$ index counter 5: while the termination criatera are not meet do $S_{CR} \leftarrow \emptyset, S_F \leftarrow \emptyset$ 6: for i = 1 to NP do 7: $r_i \leftarrow$ select from [1, H] randomly // H = 6 8: if $M_{CR,r_i} = \perp$ then 9: $CR_{i,a} \leftarrow 0$ 10: 11: else $CR_{i,q} \leftarrow \mathcal{N}_i(M_{CR,r_i}, 0.1)$ // Normal distribution 12: 13: end if $F_{i,q} \leftarrow C_i(M_{F,r_i}, 0.1)$ // Cauchy distribution 14: $\vec{u}_{i,a} \leftarrow current-to-pBest/1/bin$ 15: end for 16: for i = 1 to NP do 17: if $f(\vec{u}_{i,g}) \leq f(\vec{x}_{i,g})$ then 18: 19: $\vec{x}_{i,a+1} \leftarrow \vec{u}_{i,a}$ else 20: $\vec{x}_{i,q+1} \leftarrow \vec{x}_{i,q}$ 21: 22: end if if $f(\vec{u}_{i,q}) < f(\vec{x}_{i,q})$ then 23: $\vec{x}_{i,g} \to \mathbf{A}, CR_{i,g} \to S_{CR}, F_{i,g} \to S_F$ 24: 25: end if Shrink A, if necessary 26: 27: Update M_{CR} and M_F (Algorithm 2) Apply LPSR strategy *// linear population size reduction* 28: 29: end for 30: $q \leftarrow q + 1$ 31: end while

Memory update in L-SHADE

1: if
$$S_{CR} \neq \emptyset$$
 and $S_F \neq \emptyset$ then
2: if $M_{CR,k,g} = \bot$ or $max(S_{CR}) = 0$ then
3: $M_{CR,k,g} \leftarrow \bot$
4: else
5: $M_{CR,k,g+1} \leftarrow mean_{WL}(S_{CR})$
6: end if
7: $M_{F,k,g+1} \leftarrow mean_{WL}(S_F)$
8: $k \leftarrow k + 1$
9: if $k > H$ then
10: $k \leftarrow 1$
11: end if
12: else
13: $M_{CR,k,g+1} \leftarrow M_{CR,k,g}$
14: $M_{F,k,g+1} \leftarrow M_{F,k,g}$
15: end if

mean_{WL} is weighted Lehmer mean

$$\operatorname{mean}_{WL}(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k}$$
$$w_k = \frac{\Delta f_k}{\sum_{l=1}^{|S_{CR}|} \Delta f_l}$$
$$\Delta f_k = |f(\boldsymbol{u}_{k,G}) - f(\boldsymbol{x}_{k,G})$$

Lehmer mean

In mathematics, the **Lehmer mean** of a tuple x of positive real numbers, named after Derrick Henry Lehmer,^[1] is defined as:

$$L_p(\mathbf{x}) = rac{\sum_{k=1}^n x_k^p}{\sum_{k=1}^n x_k^{p-1}}.$$

The weighted Lehmer mean with respect to a tuple w of positive weights is defined as:

$$L_{p,w}(\mathbf{x}) = rac{\sum_{k=1}^n w_k \cdot x_k^p}{\sum_{k=1}^n w_k \cdot x_k^{p-1}}.$$

The Lehmer mean is an alternative to power means for interpolating between minimum and maximum via arithmetic mean and harmonic mean.

Population in L-SHADE

• Linear population size reduction

$$NP_{G+1} = round \left[NP^{init} - \frac{FES}{MaxFES} \left(NP^{init} - NP^{min} \right) \right]$$

- NP_{G+1} = population size in generation G+1
- Np^{init} initial population size
- Np^{min}– minimal population size
- FES current number of fitness evaluations
- MaxFES maximally allowed number of fitness evaluations