On Differential Evolution

Alexander G. Ororbia II

Biologically-Inspired Intelligent Systems

CSCI-633

2/12/2026

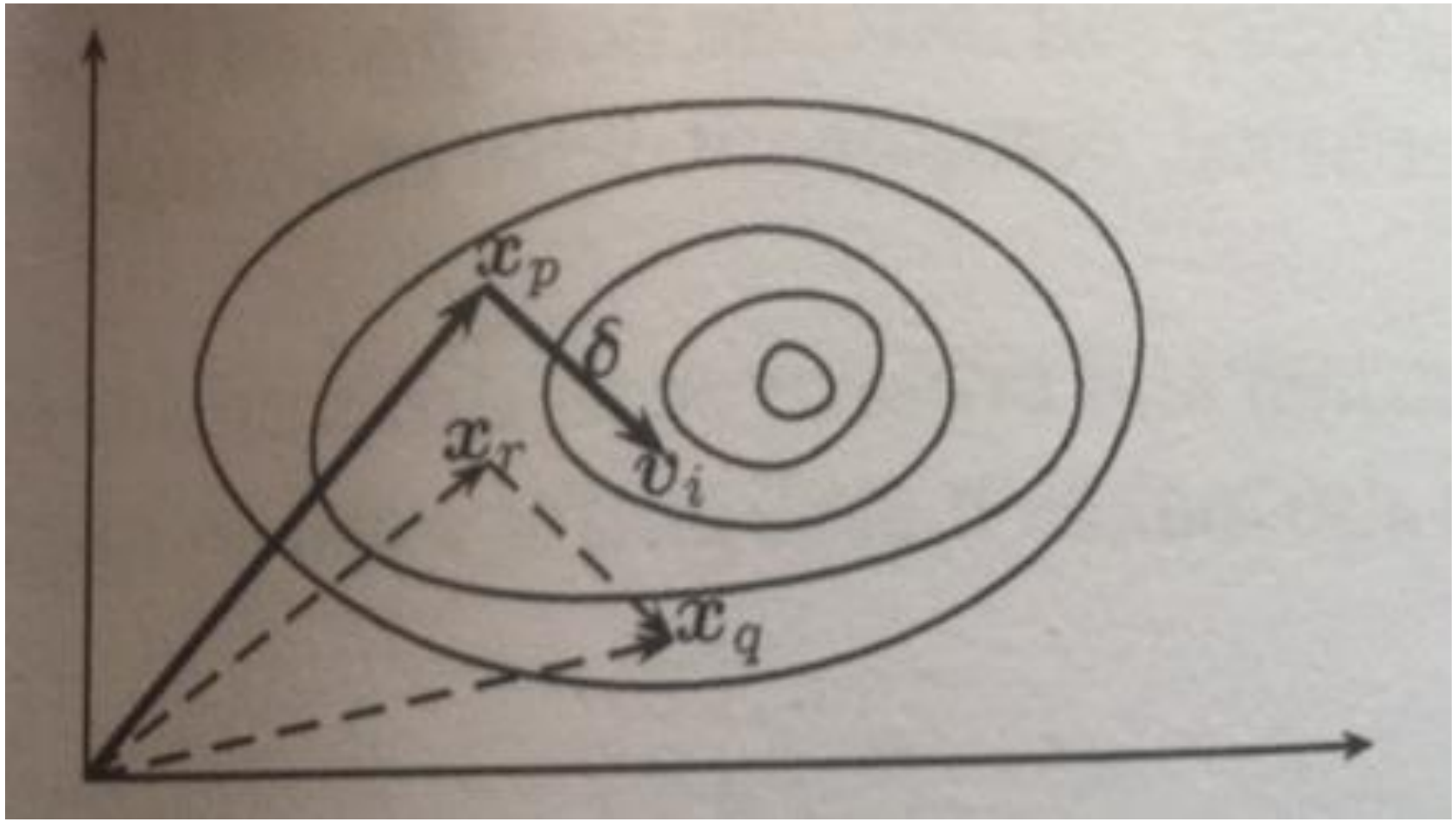
Metaheuristic: Differential Evolution (DE)

- Vector-based (population-based) algorithm, Storn & Price (1996/1997)
  - Individuals ‘evolve’ by recombination w/ other individuals & differentials between other individuals
- Devised for continuous search spaces, derivative-free
- No encoding/decoding required – real numbers are solution strings/chromosomes
- *DE/rand/1/bin*

DE General Mechanics

- Builds on the idea of genetic algorithms
- Three primary steps:
  - Mutation, crossover, selection

- Name convention: DE/x/y/z – x is mutation scheme, e.g., random (Rand) or best (Best), y is number of difference vectors, z is crossover scheme, e.g., binomial (Bin) or exponential (Exp) or either/agnostic (*)
  - **We will start with**: DE/Rand/1/*



Schematic representation the application of a perturbation/mutation in DE, according to perturbation $\boldsymbol{\delta} = \alpha(\boldsymbol{x}_q^t - \boldsymbol{x}_r^t)$ (movement along function space).

DE: A Crossover Scheme

- Crossover $C_r \in [0,1]$ controls probability/rate for crossover

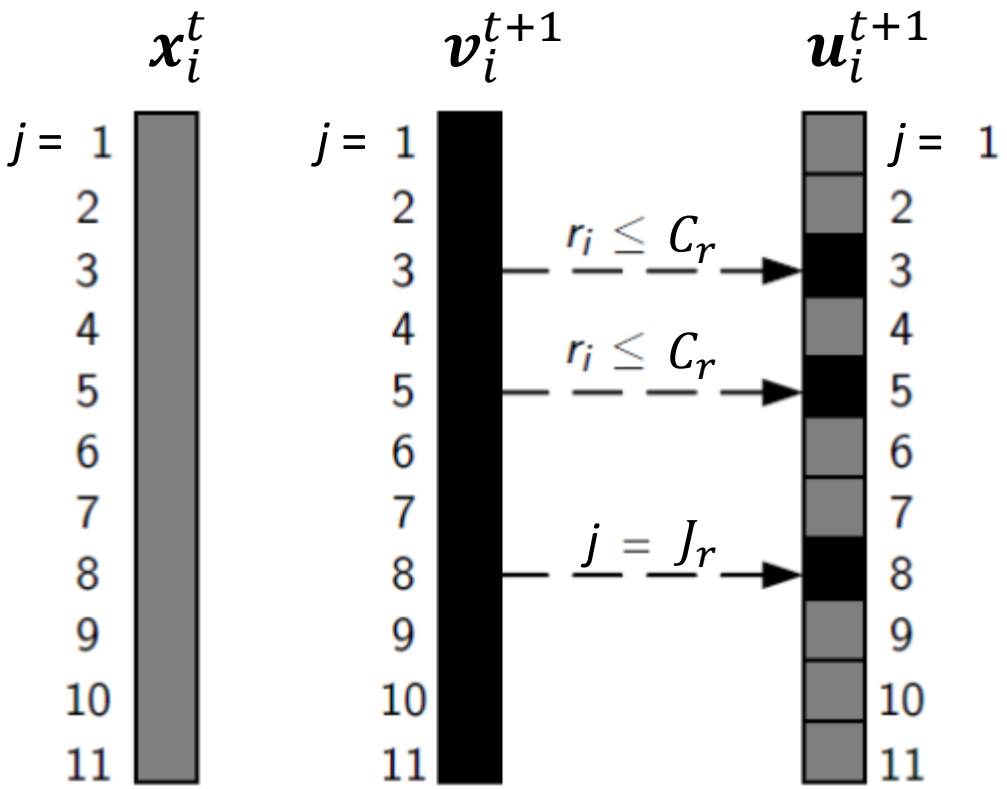
- ***Binomial*** scheme:
  - Crossover on each of *d* variables/parameters or components
  - Generate a random uniformly distributed number $r_i \in [0,1]$ and alter the *j*th dimension of $\boldsymbol{v}_i$ as follows:

$$\boldsymbol{u}_{j,i}^{t+1} = \begin{cases} \boldsymbol{v}_{j,i}, & \text{if } r_i \leq C_r, \\ \boldsymbol{x}_{j,i}^{t}, & \text{otherwise,} \end{cases} \quad j = 1, 2, \ldots, d \qquad (6.3)$$

  - Essentially, carry over dimension *j* if probability threshold not met

Binomial Crossover

- *Vector crossover visualization*



DE: Another Crossover Scheme

- ***Exponential*** scheme:
  - Segment of donor vector selected of random size *k* and random length *L* (could include multiple variables/dims)
  - Choose $k \in [0, d-1]$ and $\mathrm{L} \in [1, d]$ (uniformly), and alter dimensions/segments as follows:

$$\boldsymbol{u}_{j,i}^{t+1} = \begin{cases} \boldsymbol{v}_{j,i}, & \text{for } j = k, \ldots, k-L+1 \in [1, d], \\ \boldsymbol{x}_{j,i}^{t}, & \text{otherwise.} \end{cases} \tag{6.4}$$

- Binomial is simpler to implement, hence usually preferred

DE: Selection

- Same as in standard genetic algorithms – select the "fittest" (i.e., minimum objective value(s) of $f(\ .\ )$)
- Selection proceeds as:

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^{t+1}, & \text{if } f\left(\boldsymbol{u}_i^{t+1}\right) \leq f\left(\boldsymbol{x}_i^{t}\right), \\ \boldsymbol{x}_i^{t}, & \text{otherwise.} \end{cases} \qquad (6.5)$$

- Control search efficiency by manipulating hyperparameters α and C_r

Pseudocode

Differential Evolution

Initialize the population $\boldsymbol{x}$ with randomly generated solutions
Set the weight $F \in [0, 2]$ and crossover probability $C_r \in [0, 1]$
while (stopping criterion)
  for $i = 1$ to n,
  For each $\boldsymbol{x}_i$, randomly choose 3 distinct vectors $\boldsymbol{x}_p$, $\boldsymbol{x}_r$ and $\boldsymbol{x}_r$
  Generate a new vector $\boldsymbol{v}$ by DE scheme (6.2)
  Generate a random index $J_r \in \{1, 2, ..., d\}$ by permutation
  Generate a randomly distributed number $r_i \in [0, 1]$
  for $j = 1$ to d,
  For each parameter $\boldsymbol{v}_{j,i}$ (jth component of $\boldsymbol{v}_i$), update

$$\boldsymbol{u}_{j,i}^{t+1} = \begin{cases} \boldsymbol{v}_{j,i}^{t+1} & \text{if } r_i \le C_r \text{ or } j = J_r \\ \boldsymbol{x}_{j,i}^{t} & \text{if } r_i > C_r \text{ and } j \ne J_r, \end{cases}$$

  end
  Select and update the solution by (6.5)
  end
end
Post-process and output the best solution found

Figure 6.2 Pseudo code of differential evolution.

Pseudocode

Differential Evolution

Initialize the population $\boldsymbol{x}$ with randomly generated solutions
Set the weight $F \in [0, 2]$ and crossover probability $C_r \in [0, 1]$
while (stopping criterion)
  for $i = 1$ to n,
  For each $\boldsymbol{x}_i$, randomly choose 3 distinct vectors $\boldsymbol{x}_p$, $\boldsymbol{x}_r$ and $\boldsymbol{x}_r$
  Generate a new vector $\boldsymbol{v}$ by DE scheme (6.2)
  Generate a random index $J_r \in \{1, 2, ..., d\}$ by permutation
  Generate a randomly distributed number $r_i \in [0, 1]$
  for $j = 1$ to d,
  For each parameter $\boldsymbol{v}_{j,i}$ (jth component of $\boldsymbol{v}_i$), update

$$\boldsymbol{u}_{j,i}^{t+1} = \begin{cases} \boldsymbol{v}_{j,i}^{t+1} & \text{if } r_i \le C_r \text{ or } j = J_r \\ \boldsymbol{x}_{j,i}^{t} & \text{if } r_i > C_r \text{ and } j \ne J_r, \end{cases}$$

  end
  Select and update the solution by (6.5)
  end
end
Post-process and output the best solution found

Figure 6.2 Pseudo code of differential evolution.

Improves search efficiency by ensuring at least one dimension of perturbed solution is different from the original

Meta-parameter Selection

- Extensive work shows that meta-params/hyper-params should be tuned to problem
- DE is most sensitive to scale factor α, with $\alpha \in [0.4,0.95]$ an empirically good range with a starting point of $\alpha \in [0.7,0.9]$
- $C_r \in [0.1,0.8]$ an empirically good range w/ $C_r = 0.5$ as a starting point
- For population size *n*, value should reflect dimensionality *d* of the problem, so something such as $n = 5d$ (or $10d$) → issues for high-dimensional problems
  - Can start with fixed value as starting point: $n = 40$ or 100

Some Convergence Thoughts / Issues

- Xue et al [10]* showed that λ should be large to yield better convergence
- Zaharie [13,14] – to avoid premature convergence (for any population-based algorithm), must maintain good degree of diversity [1]
  - Generally analyze/characterize the variance of DE variants (usually w/o selection) → generally lead to conclusions about meta-params such as α and how they affect variance of population solutions
- In general: when *var*(P) is decreasing/going down, DE is converging (or when *var*(P) → 0, DE has converged)
  - This convergence may be premature (i.e., not a global optima)
- ***Issue***: Population (P) diversity also depends on initial P
- ***Issue***: combinatorial problems – difficult to say if DE works well given how hard it is to discretize differential ops, etc.

* Reference/citation numbers match those in textbook ;-)

Implemented / Used In:

- **Knapsack Problem**

  - KRAUSE, Jonas; Parpinelli, R. S.; Lopes, H.S. **Proposta de um algoritmo inspirado em Evolução Diferencial aplicado ao Problema Multidimensional da Mochila**, 2012, Curitiba. Anais do Encontro Nacional de Inteligência Artificial – ENIA.

  - KRAUSE, Jonas; Cordeiro, J.A.; Lopes, H.S.. **Comparação de Métodos de Computação Evolucionária para o Problema da Mochila Multidimensional.** In: H.S. Lopes; L.C.A. Rodrigues; M.T.A. Steiner. (Org.). Meta-Heurísticas em Pesquisa Operacional. 1ed.Curitiba: Omnipax, 2013, p. 87-98.

  - KRAUSE, Jonas; Lopes, H.S. **A comparison of differential evolution algorithm with binary and continuous encoding for the MKP.** In: BRICS - Conference on Computational Intelligence, 2014, Recife. Proceedings of BRICS-CCI, 2013.

- **Scheduling Problem**

  - KRAUSE, Jonas; Sieczka, E.; Lopes, H.S. **Differential Evolution Variants and MILP for the Pipeline Network Schedule Optimization Problem.** In: LA-CCI - Congress on Computational Intelligence, 2015, Curitiba.

Differential Evolution (Naming)

Many Variations:

- best/n/bin
- best/n/exp
- random/n/bin
- random/n/exp
- current/n/bin
- current/n/exp

Parent Selection / Number of Pairs / Recombination

- In general:
  - Perform binary or exponential recombination between the current individual and another individual modified by a scaled difference between *n* pairs of other individuals

Questions?