Nature Inspired Computing

Differential Evolution

Optimization Algorithm

Introduction

- Differential evolution (DE) is a stochastic, population-based search strategy developed by Storn and Price in 1995

- While DE shares similarities with other evolutionary algorithms (EA), it differs significantly in the sense that distance and direction information from the current population is used to guide the search process

Basic Differential Evolution

- DE differs from the evolutionary algorithms:
  - Mutation is applied first to generate a trial vector, which is then used within the crossover operator to produce one offspring, (While in EAs crossover is applied first and then mutation)
  - Mutation step sizes are not sampled from a prior known probability distribution function (while in EAs mutation and crossover both are sampled from some probability function)
- In DE, mutation step sizes are influenced by differences between individuals of the current population

Forming basis of DE

- The positions of individuals provide valuable information about the fitness
- Over time, as the search progresses, the distances between individuals become smaller, with all individuals converging to the same solution
- The magnitude of the initial distances between individuals is influenced by the size of the population
- The more individuals in a population, the smaller the magnitude of the distances.

$$\text{Distance} \, \alpha \, \frac{1}{\text{PopSize}}$$

Forming basis of DE (Cont.)

- Distances between individuals are a very good indication of the diversity of the current population
- If there are large distances between individuals, it stands to reason that individuals should make large step sizes in order to explore as much of the search space as possible
- On the other hand, if the distances between individuals are small, step sizes should be small to exploit local areas

Forming basis of DE (Cont.)

- It is this behaviour that is achieved by DE in calculating mutation step sizes as weighted differences between randomly selected individuals

- The first step of mutation is therefore to first calculate one or more difference vectors, and then to use these difference vectors to determine the magnitude and direction of step sizes

DE Operators (Mutation)

The DE mutation operator produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential. This trial vector will then be used by the crossover operator to produce offspring. For each parent, $\mathbf{x}_i(t)$, generate the trial vector, $\mathbf{u}_i(t)$, as follows: Select a target vector, $\mathbf{x}_{i_1}(t)$, from the population, such that $i \neq i_1$. Then, randomly select two individuals, $\mathbf{x}_{i_2}$ and $\mathbf{x}_{i_3}$, from the population such that $i \neq i_1 \neq i_2 \neq i_3$ and $i_2, i_3 \sim U(1, n_s)$. Using these individuals, the trial vector is calculated by perturbing the target vector as follows:

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

where $\beta \in (0, \infty)$ is the scale factor, controlling the amplication of the differential variation.

DE Operators (Crossover)

The DE crossover operator implements a discrete recombination of the trial vector, $\mathbf{u}_i(t)$, and the parent vector, $\mathbf{x}_i(t)$, to produce offspring, $\mathbf{x}_i^{'}(t)$. Crossover is implemented as follows:

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

where $x_{ij}(t)$ refers to the j-th element of the vector $\mathbf{x}_i(t)$, and $\mathcal{J}$ is the set of element indices that will undergo perturbation (or in other words, the set of crossover points). Different methods can be used to determine the set, $\mathcal{J}$.

When two organisms mate they share their genes. The resultant offspring may end up having half the genes from one parent and half from the other. This process is called recombination.

DE Operators (Binomial Crossover)

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

- The crossover points are randomly selected from the set of possible crossover points, {1, 2,...,n_x},where n_x is the problem dimension
- p_r is the probability that the considered crossover point will be included
- The larger the value of p_r, the more crossover points will be selected compared to a smaller value
- This means that more elements of the trial vector will be used to produce the offspring, and less of the parent vector

DE Operators (Exponential Crossover)

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

From a randomly selected index, the exponential crossover operator selects a sequence of adjacent crossover points, treating the list of potential crossover points as a circular array. The pseudocode in Algorithm shows that at least one crossover point is selected, and from this index, selects the next until $U(0, 1) \geq p_r$ or $|\mathcal{J}| = n_x$.

DE Operators (Selection)

Selection is applied to determine which individuals will take part in the mutation operation to produce a trial vector, and to determine which of the parent or the offspring will survive to the next generation.

With reference to the mutation operator, a number of selection methods have been used. Random selection is usually used to select the individuals from which difference vectors are calculated. For most DE implementations the target vector is either randomly selected or the best individual is selected.

To construct the population for the next generation, deterministic selection is used: the offspring replaces the parent if the fitness of the offspring is better than its parent; otherwise the parent survives to the next generation. This ensures that the average fitness of the population does not deteriorate.

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
        else
            Add $\mathbf{x}_i(t)$ to $\mathcal{C}(t+1)$;
        end
    end
end
Return the individual with the best fitness as the solution;