Department of Computer Science
University of Pretoria

Artificial Intelligence
COS710

Assignment 3: Differential Evolution & Coevolution

Due date: 22 May 2017, at 23:00

For this assignment you will implement a number of cooperative approaches to differential evolution (DE) in order to improve the scalability of DE on large-dimensional optimization problems. Be warned that this assignment will require much computational time, so start early.

You will submit a pdf report (using the specifications and style files as for the previous assignments) and all of your code. Create a compressed archive to contain your report and your code. The archive should expand to a folder named `????????`, where the question marks are replaced with your student number. In this folder, the pdf, named `????????.pdf`, should be extracted to the sub-folder `report`, and your code to the sub-folder, `code`. In the `code` subfolder, you have to add a `readme` file with details on how to compile and run each of your DE implementations. The last sub-folder, `data` will contain a spreadsheet of your data. Download the spreadhseet, `results.ods`, and use this to enter your data.

For this assignment, you need to do the following:

- Do a literature search on cooperative DE, and provide a review (survey) of existing approaches to scale DE using cooperative coevolution. In the review, focus also on approaches implemented to cope with variable dependencies.
- Implement a standard DE/rand/bin/1.
- Implement the following very basic cooperative DE: Split the problem into $k < n$ sub-components, where n is the dimension of the problem. Each sub-component is then optimized using one sub-population, executing a standerd DE/rand/bin/1. You will therefore have n/k sub-populations. Use a context vector as in the cooperative particle swarm optimization algorithm to quantify the quality of particles in sub-populations. You will evaluate performance for different values of k.
- Select one of the approaches discussed in your review above, specifically an approach that were developed to cope with variable dependencies. Implement this approach.
- Then implement the following approaches:
  - Random grouping cooperative DE: Split the problem into $1 < k < n$ sub-components, to produce k/n sub-populations. For each sub-population, at each iteration, randomly select which dimensions/decision variables will be grouped in which sub-population. Make sure that each dimension/decision variable occurs in only one sub-population. You will evaluate perfromance for different values of k.

  - Random grouping cooperative DE with sub-population overlap: This will be the same as the random grouping cooperative DE, but each decision variable will have a probability, p_o, to also be included in each of the other populations. This will allow for some overlap between populations. You will evaluate performance for different values of k and p_o.
  - Decomposition cooperative DE: In this case, the DE starts on the entire n-dimensional problem, and every t iterations, each sub-population is split in two. After each split, decision variables are randomly assigned to the new sub-populations. This process continues until n 1-dimensional sub-populations have been formed. You will evaluate performance with different values of t.
  - Decomposition cooperative DE with random grouping: This is the same as the decomposition DE, but each iteration decision variables are randomly re-assigned to sub-populations.
  - Merging cooperative DE: A similar approach to the decomposition DE, but here we start with n 1-dimensional sub-populations, and at each t iterations randomly merge two sub-populations. This process continues until we get one n-dimensional population optimizing the entire problem. You will evaluate performance for different values of t.
  - Merging cooperative DE with random grouping: This is the merging cooperative DE, but each iteration, decision variables are randomly re-assigned to the sub-populations.

- Compare the above approaches on the 11 scalable optimization problems given in the appendix, for the following dimensions: 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000. For each algorithm, for each problem, for each dimension, execute 30 independent runs, and report your results as averages ± standard deviations. As performance measure, consider the quality of the best solution at the end of 5000 iterations, using 30 individuals in each population/sub-populations. For each of the above algorithms, find the best values for the crossover probability, p_r, and for the scaling factor β. The comparison above must be for these best found values, and the best values for k and t. Present and discuss all of your results, and conclude on which approach is best.
- For the data spreadsheet, each function has its own sheet. The entries are the quality of the best solution found after the 5000 iterations, for each of the independent runs. Please do not change any of the formatting or formulas in this spreadsheet.

Appendix: Functions to Use

Use the functions listed below for your empirical analysis of these algorithms:

f_3, the alpine function, defined as

$$f_3(\mathbf{x}) = \left(\prod_{j=1}^{n_x} \sin(x_j)\right) \sqrt{\prod_{j=1}^{n_x} x_j} \tag{1}$$

with each $x_j \in [-10, 10]$.

f_4, the egg holder function, defined as

$$\begin{aligned} f_4(\mathbf{x}) &= \sum_{j=1}^{n_x-1} \Big(-(x_{j+1} + 47)\sin(\sqrt{|x_{j+1} + x_j/2 + 47|}) \\ &+ \sin(\sqrt{|x_j - (x_{j+1} + 47)|})(-x_j) \Big) \end{aligned} \tag{2}$$

with each $x_j \in [-512, 512]$.

f_6, the griewank function, defined as

$$f_6(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{j=1}^{n_x} x_j^2 - \prod_{j=1}^{n_x} \cos\left(\frac{x_j}{\sqrt{j}}\right) \tag{3}$$

with each $x_j \in [-600, 600]$.

f_9, the norwegian function, defined as

$$f_9(\mathbf{x}) = \prod_{j=1}^{n_x} \left(\cos(\pi x_j^3) \left(\frac{99 + x_j}{100} \right) \right) \tag{4}$$

with each $x_j \in [-1.1, 1.1]$.

f_{13}, the rosenbrock function, defined as

$$f_{13}(\mathbf{x}) = \sum_{j=1}^{n_x-1} \left(100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2\right) \tag{5}$$

with $x_j \in [-30, 30]$.

f_{14}, the salomon function, defined as

$$f_{14}(\mathbf{x}) = -\cos(2\pi \sum_{j=1}^{n_x} x_j^2) + 0.1\sqrt{\sum_{j=1}^{n_x} x_j^2} + 1 \tag{6}$$

with $x_j \in [-100, 100]$.

f_{15}, the schaffer 6 function, defined as

$$f_{15}(\mathbf{x}) = \sum_{j=1}^{n_x-1} \left(0.5 + \frac{sin^2(x_j^2 + x_{j+1}^2) - 0.5}{(1 + 0.001(x_j^2 + x_{j+1}^2))^2}\right) \tag{7}$$

with each $x_j \in [-100, 100]$.

f_{20}, the schwefel 2.22 function, defined as

$$f_{20}(\mathbf{x}) = \sum_{j=1}^{n_x} |x_j| + \prod_{j=1}^{n_x} |x_j| \tag{8}$$

with each $x_j \in [-10, 10]$.

f_{21}, the shubert function, defined as

$$f_{21}(\mathbf{x}) = \prod_{j=1}^{n_x} \left(\sum_{i=1}^{5} (i\cos((i+1)x_j + i)\right) \tag{9}$$

with each $x_j \in [-10, 10]$.

f_{24}, the vincent function, defined as

$$f_{24}(\mathbf{x}) = -\left(1 + \sum_{j=1}^{n_x} \sin(10\sqrt{x_j})\right) \tag{10}$$

with each $x_j \in [0.25, 10]$.

f_{25}, the weierstrass function, defined as

$$\begin{aligned} f_{25}(\mathbf{x}) &= \sum_{j=1}^{n_x} \left(\sum_{i}^{20} (a^i \cos(2\pi b^i (x_j + 0.5)))\right) \\ &- n_x \sum_{i=1}^{20} (a^i \cos(\pi b^i)) \end{aligned} \tag{11}$$

with each $x_j \in [-0.5, 0.5]$, $a = 0.5$ and $b = 3$.