**CEE 5290/ CS 5722/ ORIE 5340 Heuristic Methods for Optimization**
**Homework 7: DDS and Parallel Computing**
**Assigned: Tuesday, October 17, 2011**
**Due: Wednesday, October 26, 2011**
**TA's Office Hours: Tuesday, Thursday 3:00pm – 4:30pm**
**Prof. Shoemaker Office Hours MWF 2:30pm – 3:30pm**

1. **Relative performances of SA, DDS and GA on the "bump" function.**
Recall the bump function from HW 5 with $n = 20$:

$$f(\vec{x}) = \begin{cases} \left| \dfrac{\sum_{i=1}^{n} \cos^4(x_i) \; - \; 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|, & if \ (\forall i, 0 \le x_i \le 10) \ and \ (\prod_{i=1}^{n} x_i \ge 0.75) \\ 0, & Otherwise \ (i.e. not \ feasible) \end{cases}$$

(a) Implement the SA algorithm to *maximize* the 20-variable bump function. In this exercise, define the neighborhood for SA as a perturbation of a randomly selected element of the current solution with a Gaussian number with mean 0 and variance 5. Make sure this perturbation satisfies the *bound* constraints of the bump function, i.e. the returned element is in the interval [0,10]. Hence if your randomly generated point is outside [0,10] then generate another point randomly until you get a point inside [0,10]. Other neighborhood definitions will be accepted, but they should be described.

Run 20 trials of 10,000 cost function evaluations. Save all the best solutions at the end of each trial. Plot the average best bump function value over the 20 trials vs. function evaluations, and save the data for this plot as you will add to it in subsequent parts.

(b) Implement the DDS algorithm to maximize 20-variable bump function. *You may write your own DDS code, or use the code provided on the course website The pseudocode for DDS is given at the end of this assignment.* Run 20 trials of 10,000 cost function evaluations. Save all the best solutions at the end of each trial.
   i. Add to the figure in part (a) a plot of the average best bump function value vs *function evaluations* for DDS.
   ii. Suggest some ways you might modify DDS to make it better. Assume the parameter = 0.2 is fixed. Explain why your modification might be good on certain set (type) of problems. You do not need to implement this modification. There are many reasonable answers.

(c) Add to the figure in part (b) a plot of the average best bump function value vs. function evaluations for GA. From just looking at the plot, does it appear as if one algorithm dominates another?

(d) Perform at least 3 algorithm comparison tests (e.g. boxplot, empirical cdf, hypothesis test, etc.) for 10,000 cost function evaluations for the best solutions at the end of each of the 20 trials for all algorithms SA, DDS and GA. Which algorithm performed the best based on the tests you performed? Is this consistent with what you concluded in the part (c)? Why or why not?

**2. Parallel Computing**

Assume you want to compute GA in parallel, and you want to produce **20 offspring** in each generation and do a total of **100 generations**. Assume the time to compute the fitness function (e.g. the objective function) is $\mathbf{T_F}$ seconds for each offspring. Assume also that you want to distribute the computation of the fitness functions among all the processors as efficiently as possible, but that you cannot split the fitness calculation for one offspring between two processors. Assume also that the time for doing the GA serial calculations in one generation (which include all algorithm calculations except the evaluation of J(S) for the offspring S) takes **50 seconds**. Assume that the one-way communication time between processors is **10 seconds**. Assume that initially that you have 20 offspring and their fitness values, so that the first thing that you need to do is to do the serial part of GA.

a. Assume $\mathbf{T_F}$ is 50 seconds and that you have 20 processors.
What is the wall clock time for a run of this GA algorithm (100 generations)?
What is the "*speedup*" on 20 processors?

b. Same as in part (a), but you have 22 processors.

**c.** "*Efficiency*" is the speed up divided by the number of processors. Assume that your goal is 80% efficiency.
For what range of values of $\mathbf{T_F}$ will you have at least 80% efficiency on **20 processors**?
For what range of values of $\mathbf{T_F}$ will you have at least 80% efficiency on **22 processors?**
Explain your answer for **22 processors.**

**Below is the pseudocode for DDS for problem 1(b) (if you want to code it yourself):**

Step 1: Define DDS inputs:

- Neighborhood perturbation size parameter, r (set at 0.2)
- Maximum number of function evaluations, *m*
- Vectors of lower, $x_{min}$ and upper, $x_{max}$ bounds for all D decision variables
- Initial solutions $x_0 = [x_1,\ldots,x_D]$

Step 2. Set counter to 1, i=1, and evaluate objective function F at initial solution $F(x_0)$:

- $F_{best} = F(x_0)$, $x_{best} = x_0$

Step 3: Randomly select J of the D decision variables for inclusion in neighborhood, {N}:

- Calculated probability each decision variable is included in {N} as a function of the current iteration count: $P(i) = 1 - \ln(i)/\ln(m)$
- For d = 1,...,D decision variables, add d to {N} with probability P
- IF {N} empty, select one random d for {N}

Step 4: FOR j = 1, …, J decision variables in {N}, perturb $x_{j,best}$ using a standard normal random variable, N(0,1), reflecting at decision variable bounds if necessary:

- $x_{j,new} = x_{j,best} + \sigma_j N(0,1)$, where $\sigma_j = r(x_{j,max} - x_{j,min})$
- IF $x_{j,new} < x_{j,min}$, reflect perturbation:
  - $x_{j,new} = x_{j,min} + (x_{j,min} - x_{j,new})$
  - IF $x_{j,new} > x_{j,max}$, set $x_{j,new} = x_{j,min}$
- IF $x_{j,new} > x_{j,min}$, reflect perturbation:
  - $x_{j,new} = x_{j,min} - (x_{j,new} - x_{j,max})$
  - IF $x_{j,new} < x_{j,max}$, set $x_{j,new} = x_{j,max}$

Step 5: Evaluate $F(x_{new})$ and update current best solution if necessary:

- IF $F(x_{new}) \leq F_{best}$, update new best solution:
  - $F_{best} = F(x_{new})$ and $x_{best} = x_{new}$

Step 6: Update iteration count, i=i+1, and check stopping criterion:

- IF i=m, STOP, print output (e.g. $F_{best}$ & $x_{best}$)
- ELSE go to STEP 3

Citation:
Tolson, B.A., and C.A. Shoemaker (2007), Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, *Water Resour. Res., 43*, W01413, doi:10.1029/2005WR004723