Continuous optimisation:
The basu's problem

PALLAMIDESSI Joseph, ERSFELD Thomas

April 28, 2015

1 Abstract

2 Keywords

3 Introduction

The goal of this project is to provided a set of the 22 constants to the function derived from article[] to fit a gaussian function of $\mu = 1.5 and \phi = 0.2$.

4 Implementation

4.0.1 Genome definition

The genome is defined as a array of the 22 doubles, so each value is mapped to a specific product value as explained by the following table:

1	2	3	4	5	6	7	8	9	10	11
kTR1	KA1	nA1	dmRNA1	kTL1	dLaclI	KTL1'	dCI	kTR2	KR2	nR2

12	13	14	15	16	17	18	19	20	21	22
dmRNA2	kTL2	dLacl	kTR3	KR3	nR3	KR3'	nR3'	dmRNA3	kTL3	dGFP

For the result of this report we will refer to the each cell (ie. parameter that we tried to optimise) of the array as genes.

4.0.2 Initialization

For each genes, we defined theirs bounds as given by this project subject. Here a quick reminder

Parameter	min	max
kTR1	0,001	1000000
KA1	0,0001	1000
nA1	1	4
dmRNA1	0,001	0,1
kTL1	0,0000001	0,00001
dLacII	0,0001	0,01
KTL1'	0,0000001	0,00001
dCI	0,0001	0,01
kTR2	0,001	1000000
KR2	0,0001	1000
nR2	1	4
dmRNA2	0,001	0,1
kTL2	0,0000001	0,00001
dLacl	0,0001	0,01
kTR3	0,001	1000000
KR3	0,0001	1000
nR3	1	4
KR3'	0,0001	1000
nR3'	1	4
dmRNA3	0,001	0,1
kTL3	0,0000001	0,00001
dGFP	0,0001	0,01i

4.0.3 Evaluation

We decided to use a sample the target function and the function to optimized with a sampling rate of 100, in the hope of having a very close approximation. The fitness function used is a Root-mean-square deviation of the two vectors resulting from the sampling.

$$\text{RMSD} = \sqrt{\frac{\sum_{t=1}^{n}(x_{1,t}-x_{2,t})^2}{n}}$$

4.0.4 Crossover

Since we are dealing with real value parameters, the crossover function is a is a simple barycentric one, which is commonly found is a lot of continous optimization by genetic algorithm.

Algorithm 1 Barycentric crossover

1: **procedure** Crossover
2:     **for all** gene in Genome **do**
3:         $alpha \leftarrow random(0., 1.)$
4:         $Child.gene \leftarrow alpha * Parent1.gene + (1. - alpha) * Parent2.gene$

4.0.5 Mutator

Once again a simple technique was used. Giving a constant mutation parameter, for each gene of the genome a chance is given to add or substract a random value within by this gene limit. We then avoid gene value going outside their bound by capping the value by the gene value bounds.


Algorithm 2 Mutator operator

---

    procedure MUTATOR
2:       **for all** gene in Genome **do**
          if $tossCoin(mutation_per_gene)$ **then**
4:              $alpha \leftarrow random(-0.5, 0.5)$
             $Individual.gene \leftarrow Individual.gene + alpha * Individual.gene$
6:              $Individual.gene \leftarrow MAX(Individual.gene, gene.min)$
             $Individual.gene \leftarrow MIN(Individual.gene, gene.max)$

---

5 Preliminary results

To much of ours surprise, when we ran the optimization with an ahl concentration from 0.0001 to 10, the resulting function was quite bad. So bad, in fact, that we quickly realized that the optimized function was constant to the average of the gaussian values.

By empirecally modifying the range of ahl, we discover that given the basu's function definition, there was no way that with only one set of parameters we will have a good approxiamation on the whole definition range specified in the subject (i.e [0.0001,10]).

However we did have some pretty good results on smaller definition range, fortunately, for the most interresting range of [1,1.8] as demonstrated by the following figure.
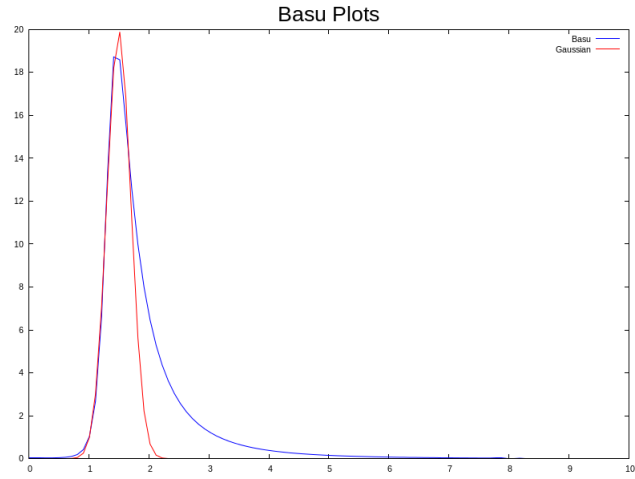
6   Results

Here are the observed results, averaged on 30 runs:

Parameter		
Nb of generations	2000	
Population size	2048	
Crossover probability	1	
Mutation probability	0.02	Parameters for basu optimisation
Surviving parents	100%	
Surviving offspring	100%	
Elitism	Strong	
Elite	1	
Result	1.17e+00	

7   Further analysis and development

In the end, we felt that multiples set of parameters, by breaking the problem definition range into several continuous ones, must be used in order to have a good approximate of the target function.