Evolutionary Computation for Optimization and Model Discovery in Biological Systems

by

Cedric Gondro

BSc. Hons. Federal University of Parana
Grad. Dip. Comp. Sci. Federal University of Parana
MSc. Genetics Federal University of Parana

October 2005

A thesis submitted for the degree of Doctor of Philosophy of the University of New England

Acknowledgements

Initially I would like to thank my principal supervisor Brian Kinghorn. His comments and suggestions were always right on the mark. I should have just followed the path he laid out instead of stubbornly steering down dead ends just to make sure they would lead nowhere. But I do appreciate the freedom with which he allowed me to carry out my research. I am greatly indebted to him for so freely sharing his knowledge and friendship.

Brian still accepted the ultimate challenge of trying to get me to play soccer at a passable level. Anyone who knows me would deem the task at best quixotic. I would not say he was entirely successful, nobody would be, but at least I'm not laughed out of the field anymore.

Brian, thank you for everything. It's been great working under your supervision.

Special thanks to my co-supervisors Anatoly Ruvinski and John Gibson for their insightful comments and for assisting me while Brian was on study leave. I am also indebted to Anatoly for keeping me on track and dragging me back down to the ground when I enthusiastically started to heed the call of those blue skies.

This thesis covers a broad range of subjects and as such would not have been viable without the valuable inputs from many experts in these different fields. For their collaboration in chapter 3 I would like to thank my good friends Alvaro Simeone and Virginia Beretta, co-authors of this chapter who took me through the ropes of cattle nutrition models. In chapter 4 I would like to acknowledge Cedric Notredame for his comments on using genetic algorithms for multiple sequence alignment. I would also like to thank Julie Thompson for her help with BaliScore. Thanks are due to Sarah Blott from Sygen International in Cambridge for her assistance in chapter 5. I am greatly indebted to Thomas Sauter, Thomas Eissing and the Systems Biology group at the University of Stuttgart for welcoming me so warmly at their Institute and patiently coaching me on how to work with differential equations models. This background knowledge was indispensable for the latter chapters of this thesis.

Thanks to Bruce Tier, Alex Woolaston, Robert Murison and Julius Van der Werf, whose doors I was frequently knocking down in search of answers. Special thanks to Brad Walmsley, my 'cellmate' during this work, I know we frequently got under each others feet and quite often got sidetracked in our work with not particularly relevant discussions (which just made them even more interesting); but I also learnt a lot from you – I appreciate your help and friendship. Thanks mate.

This work could not have been carried out without the financial support of the Cooperative Research Centre for Cattle and Beef Quality. I greatly appreciate the scholarship which I was granted with. Even though most of the chapters of this thesis are not directly related to the beef industry, I hope the methods presented are, in the future, applied to real life scenarios and contribute to the growth of the industry.

I am grateful to the University of New England for providing the research grant and infrastructure for this project.

I would still like to thank Sygen International for funding my travel expenses at the University of Cambridge, Roslin Institute and the University of Stuttgart. It was a unique opportunity and I appreciate it.

I have a debt I will never be able to repay to my mentor and friend Joao Carlos Magalhaes who sent me down this fascinating path. I still remember the first time I approached him to see if he would supervise a Human Behavior Honors project I was thinking of. In reply he asked me if I knew anything about computers and if I had ever heard of genetic algorithms. Joao, you changed my life.

I would like to thank my parents for their unyielding support in my endeavors. I am sorry I am literally on the other side of the world and we see so little of each other. I miss you both.

Finally, Simone Valle de Souza, my wife and friend for so many years. Thanks for your courage, optimism and willingness to embrace new things. Life would not make much sense without you.

For Sean – may life be good to you.
"42"

Abstract

Evolutionary Computation encompasses a large group of stochastic problem solving methods loosely inspired on biological evolutionary processes such as selection, mutation and recombination. These methods are commonly referred to as Evolutionary Algorithms and all have in common the use of populations of candidate solutions which reproduce, compete, and are subjected to selective pressures and random variation – the four basic elements of evolution. Some of the best known Evolutionary Algorithms include Genetic Algorithms, Genetic Programming and Evolution Strategies. Evolutionary Algorithms are suited to optimization of complex non linear problems, making them appropriate to optimization of biological problems which are usually complex and non linear. In this thesis Evolutionary Algorithms were developed and used to optimize biological problems. For the first problem, Differential Evolution was used to optimize nutrition model parameters in beef cattle, predict weight gains and formulate rations; results indicate a good fit of predicted values to observed measurements. A second problem addressed was the multiple sequence alignment of amino acid and nucleotide residues. A Genetic Algorithm was developed and results indicate that the method can improve on Clustal-W alignments. Genetic Algorithms were also used to optimize the experimental design of spotted microarrays. Optimization is treated as a multicriteria problem and the heuristic uses a weighted multi objective function to balance conflictive design considerations. Results evidence that, for a given set of constraints, near optimal designs can be found that are more efficient than common designs. Differential Evolution is also used to parameterize biochemical systems modelled as S-systems differential equations, which adhere to a fixed functional form, yet fit well to diverse complex systems. Results show that time series data can efficiently parameterize the evaluated models. The last problem evaluated was full model discovery and parameterization from time series data. A hybrid Differential Evolution and Gene Expression Programming algorithm was developed and compared against other heuristic methods. Alongside the development of methods, a major component of this project was the production of computational tools that implement the methods; five tools – one for each research topic – were developed and are freely available. Evolutionary Computation is a powerful approach to optimization of complex biological problems; this is clearly evidenced through the different problems that were addressed in this thesis.

***Keywords**: evolutionary computation; evolutionary algorithms; gene expression programming; differential evolution; genetic algorithms; cattle nutrition models; bioinformatics; computational biology; multiple sequence alignment; microarray experimental design; s-systems; delay differential equations; biochemical systems; genetic networks.*

Declaration

I certify that the substance of this thesis has not already been submitted for any degree and is not currently being submitted for any other degree or qualification.

I certify that any help received in preparing this thesis, and all sources used, have been acknowledged in this thesis.

[redacted]

Table of Contents

Acknowledgements ii
Abstract iv
Declaration v
Table of Contents vi
List of Figures viii
List of Tables xiv
Abbreviations xvii
1 Introduction 1
2 Literature Review 4
2.1 Computational Biology and Bioinformatics 4
2.2 Systems Biology 5
2.2.1 Characteristics of Biological Systems 6
2.2.2 Methods in Systems Biology 7
2.3 Modelling of Biological Systems 8
2.3.1 Modelling Approaches 9
2.3.2 Computational Tools for Modeling Biological Systems 11
2.4 Microarrays 11
2.4.1 Types of Microarrays 12
2.4.2 Analysis of DNA Microarrays 13
2.5 Evolutionary Computation 13
2.5.1 Evolution of Evolutionary Computation 14
2.5.2 Overview 14
2.5.3 Pros and Cons 15
2.5.4 Types of Evolutionary Computation 17
2.5.5 Computational Representation and Implementation 20
2.5.6 Selection 20
2.5.7 Recombination 22
2.5.8 Mutation 22
2.5.9 Applications in Bioinformatics 23
2.5.10 Summary 23
2.6 Conclusion 24
3 Nutrition Parameter Optimization in Beef Cattle Using Differential Evolution 25
3.1 Introduction 25
3.1.1 Authorship Note 27
3.2 Mathematical Model of Cattle Nutrition 27
3.3 Differential Evolution 28
3.3.1 Differential Evolution Search Operators 29
3.4 LaGraZ 30
3.4.1 Cattle Parameters 31
3.4.2 Feed Parameters 32
3.4.3 Optimization 34
3.4.4 Visualization of Results 36
3.4.5 A Simple Optimization Example Using LaGraZ 37
3.5 Experimental Results from LaGraZ 38
3.6 Conclusion and Future Work 39
4 A Simple Genetic Algorithm for Multiple Sequence Alignment 41
4.1 Introduction 41
4.2 Sequence Alignment 42
4.2.1 Dynamic Programming 43
4.2.2 Progressive Alignment 43
4.2.3 Scoring Systems and Objective Functions 44
4.3 Multiple Sequence Alignment with Evolutionary Computation 46
4.3.1 A Simple Genetic Algorithm for Multiple Sequence Alignment 47
4.3.2 MSA-GA 50
4.4 Evaluation of MSA-GA for Multiple Sequence Alignment of Proteins 52
4.4.1 Discussion 55

4.5 Conclusion and Future Work 56
5 Optimization of Microarray Experimental Design Using Genetic Algorithms 58
5.1 Introduction 58
5.2 Experimental Design of Microarrays 59
5.2.1 Sources of Variation 60
5.2.2 Replication 60
5.2.3 Graphical Representation of Designs 61
5.2.4 Types of Design 61
5.2.5 Design Optimization 63
5.3 A Genetic Algorithm for Optimization of Microarray Experimental Designs 64
5.3.1 Design Representation 64
5.3.2 Genetic Operators 65
5.3.3 Fitness Function 67
5.3.4 ArrayDesigner – Software for Microarray Design Optimization 69
5.4 Optimization of Microarray Experimental Designs 72
5.5 Conclusion and Future Work 75
6 SimArray, Software for Modelling Genetic and Biochemical Networks 76
6.1 Introduction 76
6.2 Directed Graphs 77
6.3 S-Systems 79
6.3.1 Characteristics of S-Systems 79
6.4 SimArray 82
6.4.1 Graphical Design of Networks 83
6.4.2 Dynamic Modelling of Networks 85
6.4.3 Steady-States and Stability/Sensitivity Analyses 86
6.4.4 Parameter Estimation using Differential Evolution 89
6.4.5 Simulation of Gene Expression Data 90
6.5 An Example of Model Building and Parameter Estimation Using SimArray 90
6.5.1 Parameter Estimation for the Fermentation Pathway in Saccharomyces cerevisae 93
6.6 Conclusion and Future Work 95
7 Hybrid Evolutionary Computation Algorithm 98
7.1 Introduction 98
7.2 Lac Operon in Escherichia coli 99
7.3 Gene Expression Programming 101
7.4 Hybrid Evolutionary Algorithm 103
7.4.1 Hybrid Differential Evolution and Gene Expression Programming Algorithm 104
7.4.2 Selection and Search Operators in Gene Expression Programming 105
7.4.3 Fitness and Objective Functions 105
7.4.4 Bloat 106
7.4.5 Computational Tool for Evolutionary Algorithms 107
7.5 Hybrid Algorithm in Model Discovery and Parameterization 108
7.5.1 Comparison of Differential Evolution with Other Heuristic Methods 110
7.5.2 Comparison of the Hybrid Algorithm with Gene Expression Programming 112
7.5.3 Evolved System of Differential Equations and Parameters for *Model I* 113
7.5.4 Evolved System of Differential Equations and Parameters for *Model II* 114
7.6 Conclusion and Future Work 115
8 Conclusions 117
8.1 Nutrition Parameters in Beef Cattle 118
8.2 Multiple Sequence Alignment 118
8.3 Microarray Experimental Design 119
8.4 S-Systems Modelling of Genetic and Biochemical Networks 119
8.5 Reconstruction of Genetic and Biochemical Pathways 119
8.6 Final Remarks 120
References 121

List of Figures

Figure 2.1 19

Recombination in a tree GP. Recombination between parents A and B generate offspring C, the function set is {*,+,/,-} and the terminal set is {2,0,1,5}. The trees code for A=(5*0)-(1*2); B=(5+0)+(1/2); C=(5*0)-(5+0).

Figure 2.2 22

One-point crossover in a binary genetic algorithm. A breakpoint is randomly selected and the two chromosomes swap bitstrings after the breakpoint. Recombination is a search operator which explores available population variability by testing new combinations. No new allelic variability is generated through recombination but it does generate new variation in fitness values.

Figure 2.3 23

Point-mutation bit-flip in a binary genetic algorithm. New offspring are produced by a random change to the parent. In the example allele at position 4 mutated and was flipped from zero to one. Mutation is a source of new variability in a population.

Figure 3.1 31

Screenshot of LaGraZ. The panel on the left hand side is used to set the parameters of the model. These are grouped into two classes: *Cattle* and *Feed* parameters, which are accessed by clicking on the tabs on the upper side of the panel. The lower box on the panel provides a brief description of the highlighted parameter. The window on the right hand side describes the selected parameters and the results of the simulation. The arrow on the tool bar runs the optimization routine.

Figure 3.2 33

Feed parameters in LaGraZ. Parameters not relevant for a given scenario are greyed out. Shaded fields are only informative and automatically updated when a parameter is changed.

Figure 3.3 34

Screenshot of *Feed Table.* Feeds from three different classes (concentrates, roughage and pasture) can be selected from the dropdown list on the right hand side. The user defines the percentage of each selected feed for the three classes. The daily allowances in kilograms are automatically calculated from the parameters. On the left hand side is the Feed Editor. Feeds and their parameters can be created, modified and deleted by the user.

Figure 3.4 35

Screenshot of *Optimization Parameters*. User defined DE parameters are the maximum number of generations and population size (number of chromosomes) in a run. Any combination of *targets* and *parameters* can be selected. *Targets* are used in the fitness evaluation of the chromosomes. *Parameters* are the model parameters that are optimized by the DE.

Figure 3.5 37

Screenshot of the *Energy Plot*. It describes the total required metabolizable energy, the energy requirements for gain and maintenance, the available energy and the balance between supplied and required energy.

Figure 3.6 39

Comparison of predicted daily live weight gain with LaGraZ and experimental data from trials in Uruguay.

Figure 4.1 49

Horizontal (A) and *vertical* (B) recombination in the MSA Genetic Algorithm. (A) Offspring are generated by randomly selecting entire sequences from either of the parents. (B) A randomly defined cut point splits the sequences of the parents in two; offspring are generated by selecting one substring from each parent.

Figure 4.2 51

Screenshot of MSA-GA. The upper pane shows FASTA format sequences to be aligned. The lower pane displays the pairwise alignments of all sequences. The *MSA-GA Settings* window allows selecting the run parameters of the GA and the alignment settings.

Figure 4.3 52

Screenshot of MSA-GA showing the best alignment at the end of a run.

Figure 5.1 61

Microarray loop design with 4 arrays and 4 samples. Nodes represent the samples and the edges are the arrays. The arrow is used to represent the dye color: the head is the red dye (Cy5) and the tail is the green dye (Cy3). The samples are represented by the letters (A, B, C and D) and the arrays are numbered from 1 to 4. The samples co-hybridized on each array are: 1 – A/B, 2 – B/C, 3 – C/D and 4 – D/A.

Figure 5.2 62

A Dye-swap loop design. For each pair of samples two hybridizations are performed with the dyes reversed in the second hybridization. In this example, for all three samples a direct comparison is possible.

Figure 5.3 63

Reference Design. Four samples of interest are hybridized to a common reference. All comparisons are indirect with a step size of 2.

Figure 5.4 63

Loop Design. Four samples are hybridized one to another. The design is balanced – each sample is used twice, once with each dye. There are 4 possible direct comparisons (A/B, B/C, C/D, D/A).

Figure 5.5 65

A GA chromosome with 5 varieties (numbered s_0 – s_4) used to represent a loop design. Dimension *n* is used for dye assignment and dimension *m* represents the number of arrays in the design.

Figure 5.6 66

Recombination operators. A) One-point recombination – offspring is generated by copying one of the parents from the start of the array up to the breakpoint and copying the other parent from the breakpoint to the end of the array. B) Block recombination with 2 blocks – the tournament loser is copied into the offspring and blocks of variable size from the winner are grafted into the offspring.

Figure 5.7 67

Mutation operators. A) Sample swap mutation – a sample in the array (in bold) in replaced by a new sample, with this single change the design becomes a loop design. B) Dye Swap mutation – the dye channels of a hybridization pair are swapped (in bold), the change turned the unbalanced design into a balanced one.

Figure 5.8 70

Screenshot of ArrayDesigner – a simple example with four samples and a fixed number of four slides. The optimization criteria were: balance dyes and reduce number of steps; equal weights (*1.0*) were used for both criteria. The upper pane presents result of the run, the optimized design in text format and dye balance information. The lower pane shows the optimized design as a graph.

Figure 5.9 71

Screenshot of the parameter settings window in ArrayDesigner.

Figure 5.10 74

Evolved designs for six example cases. 1) Time course design. 2) Comparisons of equal interest. 3) Minimize number of arrays. 4) Minimize number of steps. 5) Balance dye. 6) Comparisons of different interest.

Figure 6.1 78

Graphical symbols used to represent reactions in a directed graph. A) irreversible reaction; B) reversible reaction; C) independent divergence branch point; D) independent convergence branch point; E) dependent divergence branch point; F) dependent convergence branch point; G) co-enzyme activation; H) modulation; I) inhibition; J) activation. Adapted from Voit (2000:26).

Figure 6.2 79

An example of a directed graph – glycolysis pathway, modified from Voit (2000:29).

Figure 6.3 80

A simple linear pathway. A) graph with names of variables. B) graph with indexed notation; dependent variables are listed first and then the independent variables.

Figure 6.4 84

Screenshot of SimArray with a simple feedback pathway. On the left-hand side is graphic toolbar with a pointer to select and modify the layout of nodes, four node representations, six flow and modulator arrows and zoom tools. The node and edge editor on the left-hand side allows provides graph editing capabilities. The parameter editor on the right-hand side provides editing capabilities of initial values, variable settings, rate constants and kinetic orders.

Figure 6.5 85

S-systems equations for the simple feedback pathway.

Figure 6.6 86

Red and green channel plot for the feedback pathway. The simulation was run over a 10 minute interval, solved using Euler with a 0.001 step size. Two bolus of 0.2 of *X1* were added at one and five minutes on the red channel.

Figure 6.7 87

Steady State and Stability window for the feedback pathway (steady-states view). The steady-states of all dependent variables are shown. The material flow is the same for *X1* and *X2* following from the precursor-product relationship. The turnover time of *X1* is over four times the turnover time of *X2* (first-order values). The system is locally stable since the real parts of all the eigenvalues are negative. Non-zero imaginary parts of eigenvalues indicate possibility of oscillations.

Figure 6.8 88

Steady State and Stability/Sensitivity window for the feedback pathway (sensitivities view). The sensitivities of the dependent variables in relation to changes in the independent variables, rate constants and kinetic orders are shown. Sensitivities can be seen as a measurement of the percentage of change in a dependent variable that is caused by a 1% change to the independent variables and the parameters of the system.

Figure 6.9 91

Snapshot of directed graph in SimArray of the fermentation pathway of *Saccharomyces cerevisae* adapted from Voit (2000:261).

Figure 6.10 92

S-systems equations of the dependent variables in the fermentation pathway of *Saccharomyces cerevisae.*

Figure 6.11 95

Fit of the best evolved parameters in relation to the original parameters in the yeast model.

Figure 7.1 100

Schematic representation of the *lac operon* in *E. coli.* In the absence of lactose the repressor binds to the operator and inhibits transcription (upper half). In the presence of lactose, the repressor binds to allolactose, changes its conformation and can no longer bind to the operator, thus freeing transcription of the structural genes *Z*, *Y* and *A* by RNA polymerase (lower half).

Figure 7.2 102

Expression tree of equation *y = (a+b)*(c-d)*. Terminal set T={a,b,c,d) and function set F=(+,*,-}.

Figure 7.3 102

GEP gene of size 21 with head (h) of size 10 and tail (t) of size 11. *t = h(n-1)+1*. The highest arity of the function set is 2, thus *n* = 2.

Figure 7.4 103

Effect of mutation on the GEP structures. A single point mutation can completely change the size of ORFs, the ETs and the final product. In position 4 a terminal (b) mutated to a function (*) as shown in Gene B. From this mutation a new ORF was created of size 15 (shaded grey) replacing the original ORF of size 7, and a new tree (new segments shaded grey). The final products are shown under *Equation.* Tails in bold font.

Figure 7.5 106

Figure 7.5 Bloat. The shaded tree is inert and does not alter the final value of the organism. A) A bloated parse-tree with 15 nodes. B) The same tree can be reduced to 5 nodes.

Figure 7.6 108

Figure 7.6 Evolutionary Computation Methods Tool. A C# tool that implements the hybrid DE-GEP algorithm. The tool also allows comparison of the efficiency of the hybrid approach to other heuristic methods. In the example Differential Evolution was used to find the values of five variables that, solving for the given equations fit the data in the columns (columns with no asterisks – given values, columns with asterisks – predicted values).

Figure 7.7 **113**

Fit of predicted and simulated data points for *model I*.

Figure 7.8 **114**

Fit of predicted and simulated data points for *model II*. Allolactose levels are scaled down on a 1:1000 ratio.

List of Tables

Table 3.1 32

Breeds and Standard Reference Weights (SRW) used in LaGraZ.

Table 3.2 37

Cattle and Feed model parameters used in the optimization example.

Table 3.3 38

Optimized ration formulation for a daily gain of 1kg in Nelore cattle.

Table 4.1 45

BLOSUM62 Substitution Scoring Matrix. The BLOSUM 62 is a 20x20 matrix in which every possible conservation and substitution is assigned a score based on the actual observed frequencies in alignments of proteins with 62% similarity. Identities are assigned the most positive scores. Frequently observed substitutions are assigned positive scores and infrequent substitutions are assigned negative scores (Henikoff & Henikoff 1992).

Table 4.2 45

Example of a simple 4x4 substitution matrix for nucleic acids.

Table 4.3 48

Population initialization algorithm.

Table 4.4 53

GA Parameter settings used in MSA-GA.

Table 4.5 54

BaliScore score results of MSA-GA, MSA-GA with prealigned sequences and Clustal W for 28 test cases selected from BaliBase 2. In reference 1 the abbreviations are short (*S*), medium (*M*) and long (*L*). References from groups 6 and 7 were not included since BaliScore returned an error when trying to score the Clustal W alignments. The figures in *bold* show a higher score compared to Clustal W.

Table 5.1 72

GA parameters used for design optimization. Tournament size is hardcoded in ArrayDesigner.

Table 5.2 73

Criteria and design parameters for 6 experimental designs. In test case 3 the value for slides refers to the maximum possible number of slides. The values for the criteria are the weights used – a value of 0.0 means that the criterion was not used.

Table 5.3 74

Average variances for all possible parameterizations of GA evolved designs and A-optimal designs from Kerr & Churchill (2001). *V* represents the number of varieties and *A* the number of arrays in the design.

Table 6.1 80

Table of production and degradation constants, variables and kinetic orders for pathway of figure 6.3B.

Table 6.2 91

Dependent variables in the fermentation pathway of *Saccharomyces cerevisae.*

Table 6.3 91

Independent variables in the fermentation pathway of *Saccharomyces cerevisae.*

Table 6.4 92

Steady-states and Stability/Sensitivity values for the independent variables of the *Saccharomyces cerevisae model.*

Table 6.5 94

Optimization of rate constants using steady-states in the yeast model.

Table 6.6 94

Optimization of rate constants and kinetic orders for *internal glucose* using steady-states for model fitting in the yeast pathway.

Table 7.1 104

Simplified algorithm of the hybrid method using Differential Evolution and Gene Expression Programming.

Table 7.2 109

Parameters for *Model I* of the lac operon from Mahaffy and Savev (1999). The parameters are not expressed in units since they are not derived from biological experimental data.

Table 7.3 110

Parameters for *Model II* of the lac operon from Yildirim *et al.* (2004). The parameters used were estimated from experimental data and closely reflect biological values.

Table 7.4 **110**

Parameter settings for the heuristic methods.

Table 7.5 **111**

Comparison of efficiency of heuristic methods.

Table 7.6 **112**

Parameter settings for GEP and the hybrid algorithm. The symbol @ in the function set is the time delay function. The concentrations in the variables set are the values of the concentrations over time of each component of the model (4 in *model I* and 3 in *model II*) as estimated by the algorithm.

Abbreviations

BLOSUM – Blocks Amino Acid Substitution Matrix

Cy3 – Cyanine-3

Cy5 – Cyanine-5

DE – Differential Evolution

DP – Dynamic Programming

EA – Evolutionary Algorithms

EC – Evolutionary Computation

EP – Evolutionary Programming

ERDP – Effective Rumen Degradable Protein

ES – Evolution Strategies, Effective Size

EST – Expressed Sequence Tag

ET – Expression Tree

FME – Fermentable Metabolizable Energy

GAs – Genetic Algorithms

GEP – Gene Expression Programming

GP – Genetic Programming

GUI – Graphical User Interface

HC – Stochastic Hill Climbing

HDE-GEP – Hybrid Differential Evolution and Gene Expression Programming

KEGG – Encyclopedia of Genes and Genomics

LCS – Learning Classifier Systems

LP – Linear Programming

MEP – Multi Expression Programming

MOP – Multiple-Objective Programming

MSA – Multiple Sequence Alignment

NCBI – National Center for Biotechnology Information

NFL – No-free-lunch Theorem

NIH – National Institutes of Health

NP – No Polynomial

ORF – Open Reading Frame

OF – Objective Function

PAM – Percent of Accepted Mutation

RS – Random Search

SA – Simulated Annealing

SAGE – Serial Analysis of Gene Expression

SBML – Systems Biology Markup Language

SDE – System of Differential Equations

XML – Extensible Markup Language