THESIS REPORT

Ph.D.

Evolving Population-Based Search Algorithms through Thermodynamic Operation: Dynamic System Design and Integration

by R-L. Sun
Advisors: W.A. Weigand
J.E. Dayhoff

Ph.D. 95-2

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Abstract

Title of Dissertation: Evolving Population-Based Search Algorithms through Thermodynamic Operation : Dynamic System Design and Integration

Ray-Long Sun, Doctor of Philosophy, 1995

Dissertation directed by: Professor William A. Weigand
Department of Chemical Engineering
and
Dr. Judith E. Dayhoff
Institute for Systems Research

During the last three decades there has been a growing interest in algorithms which rely on analogies to natural processes. The guided random search techniques, genetic algorithms (GAs) and simulated annealing (SA), are very promising strategies, and both techniques are analogs from physical and biological systems. These two algorithms are stochastic relaxation search methods especially suitable for applications to a wide variety of complex optimization problems. Each produces a sequence of candidate solutions to the underlying problems, and the purpose of both algorithms is to generate sequences biased toward solutions which optimize the objective function.

Limitations, however, occur in performance because optimization may take a large number of iterations, and final parameter values may be found that are not at the global extremum points. In this thesis, a population-based search algorithm that combines approaches from GAs and SA is proposed. The combined approach, called GASA, maintains a population of individuals over a period of generations. In the GASA

technique, simulated annealing is used in choices regarding a subset of individuals that undergo crossover and mutation. This thesis shows that the GASA technique has superior performance when compared to a genetic algorithm for the nonlinear function optimization problem.

Temperature plays an important role in the GASA algorithm, The temperature change results in the moving of populations. When arriving at equilibrium, the individuals in the same population also have an assumed unique critical temperature. This phenomenon follows the rules of general thermodynamic laws. Schema theory and simulated phase transition are used to explore the search mode of GASA. Energy changes affect structural change by mutation or crossover.

GASA is utilized for parameter optimization on dynamic system design and integration. The GASA technique can be used for different tasks like control, detection, and computation. For fed-batch bioprocesses, an optimized input function (substance feed rate) with GASA can increase the quantity of product. GASA is applied to pH control in combination with the classical PID control. A convergence analysis of GASA explores the characteristics of this evolutionary controller.

Evolving Population-Based Search Algorithms through Thermodynamic Operation : Dynamic System Design and Integration

by

Ray-Long Sun

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1995

Advisory Committee:

Professor William A. Weigand, Chairman/Advisor
Research Scientist Dr. Judith E. Dayhoff (Co-advisor)
Associate Professor Nam S. Wang
Associate Professor William E. Bentley
Professor Bilal M. Ayyub

Dedication

To my parents

Acknowledgements

First, I would express my profound appreciation to my advisors, Professor William Weigand and Dr. Judith Dayhoff, for their advice and patience in this research. This work took shape from many, many discussions we had, and their advice was invaluable. Specifically, Dr. Dayhoff contributed her time to review the algorithm design and other theoretical parts like Chapter 4 and Chapter 5 as best as she can. The application parts, Chapter 6 and Chapter 7, were reviewed by Prof. Weigand mostly. Also, I would like to thank my doctoral examination committee, Drs. Nam Wang, William Bentley and Bilal Ayyub, for valuable comments and suggestions on this work.

Special thanks are extended to my worldwide Internet friends such as Dr. D. Fogel, P. Harmut and Dr. S. Forrest for sharing their valuable knowledge and providing me the very useful materials, even though we only met on-line via terminals. The other valuable discussion, laughter and sense of humor from my ISR friends and other colleagues are memorized in this study. In addition, I also acknowledge the computer facilities supported from the two grants of the National Science Foundation, CDR-88-03012 and BIR9309169.

Lastly, I wish to thank my parents for their support and encouragement of my overseas study at the University of Maryland. The other concern from my brother and sister is appreciated also.

Table of Contents

Section			Page
List of Tables			**viii**
List of Figures			**ix**
Chapter			**1**
1	**Introduction**		**1**
	1.1	Motivation of the Thesis	2
	1.2	Organization of the Thesis	6
2	**Claims Section**		**9**
3	**Search Algorithms and Nonlinear Programming**		**12**
	3.1	An Overview of Search Algorithms	12
	3.2	Genetic Algorithms	14
		3.2.1 Description	15
		3.2.2 Genetic operators	17
		3.2.3 Other genetic alternatives	20
	3.3	Simulated Annealing	21
		3.3.1 Description	22

3.3.2 Thermal equilibrium in simulated data transformation 23
3.3.3 Annealing process 24
3.4 Applications of Evolutionary Computation and Further Challenges . . 26
3.4.1 Applications of evolutionary computation 26
3.4.2 Challenges for evolutionary computation 27
3.5 Miscellaneous Evolutionary Computation Algorithms 29
3.5.1 Evolutionary algorithms (EAs) 29
3.6 Structured Evolutionary Computation for Nonlinear Programming . . 31
3.6.1 Nonlinear programming Environment 31
3.6.2 A structured genetic algorithm 32
3.6.3 A structured simulated annealing 35
3.6.4 A structured evolutionary algorithm 37

4 A Population-Based Search Algorithm through Thermodynamic Operation — GASA Algorithm Design 39

4.1 An Overview 39
4.2 The Preliminary Unified Concepts 42
4.2.1 A mapping inspired from nature — biological and physical systems 42
4.2.2 The premise of the combination of GAs and SA 43
4.3 A Unified Approach 45
4.3.1 GASA algorithm design 45
4.4 Results on Function with Multiple Optima 54

5 Temperature Effect and Search Mode 70

5.1 Annealing Schedule 71

5.1.1 Initial temperature 73
5.1.2 Temperature scale 77
5.2 The Effect of Partial Annealing Mechanism 82
5.3 Search Phase and Schema Theory 89
5.3.1 Schema theory 91
5.4 Simulated Phase Transition 94
5.4.1 Qualitative analysis 95
5.4.2 Quantitative analysis 97
5.5 Population Turbulence 100
5.6 Summary 103

6 Fed-Batch Bioreactor Optimization with the Genetic-Annealing Approach 112

6.1 Optimal Control and Generalized Genetic-Thermodynamic Algorithm (GASA) 114
6.1.1 Dynamic optimization 115
6.2 Fed-Batch Bioreactor Optimization 116
6.3 Dynamic Optimization of Fed-Batch Bioreactor with GASA algorithm 117
6.3.1 GASA algorithm and fed-batch optimization 118
6.4 Results and Discussion 119

7 Control of a pH Plant Using Genetic-Annealing Approach 133

7.1 Genetic-Annealing Approach 134
7.2 Problem Formulation 136
7.2.1 A pH plant model 136
7.3 Control Algorithms 139
7.3.1 PID controller 139

7.4 Control Tuning of the Plant 140
7.4.1 PID tuning by genetic-annealing approach 141
7.5 Experiment Result 142
7.6 Convergence Analysis 147
7.7 Summary 148

8 Conclusions and Future Perspectives **154**
8.1 Conclusions from the Dissertation 154
8.2 Future Perspectives 158

Appendix **160**

A Nomenclature **160**

Bibliography **164**

List of Tables

Number		Page
4.1	The basic units in different algorithms	44
4.2	Bochachevsky Function: Statistic Analysis of the Number of Cutoff Generations to Achieve a Degree of Global Optimization with GAs	58
5.1	The turning points for the larger initial temperatures T_0s	76
5.2	The spent generations and time for GAs and GASAs	85
6.1	The basic units in different algorithms	122
7.1	The PID parameters and performance function (PF) for ZN, GAs and GASA	147

List of Figures

Number		Page
3.1	Classes of search algorithms	13
3.2	The GAs cycle	16
3.3	An Outline of the Genetic Algorithms	18
4.1	The behavior of the participating individuals	43
4.2	A multiple-optima function	54
4.3	The contour plot of Bohachevsky function	55
4.4	Annealing schedule	57
4.5	An Interpretation of GASA operation in generations: (a) The locations of all individuals in generation 28. (b) The objective values of all individuals in generation 28. (c) Best and mean objective values of individuals in generations. (d) Best individuals in generations.	60
4.6	Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.	61

4.7 An Interpretation of GAs operation in generations: (a) The locations of all individuals in generation 69. (b) The objective values of all individuals in generation 69. (c) Best and mean objective values of individuals in generations. (d) Best individuals in generations. 62
4.8 Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. 63
4.9 Mean objective value in generations 64
4.10 Best objective value in generations 64
4.11 Standard deviation of population vs. generation 65
4.12 Variance of population vs. generation 65
4.13 The minimum of indviduals vs. generations in 1$\sim$ 5 sets of GASA in round 1 66
4.14 The minimum of indviduals vs. generations in 6$\sim$ 10 sets of GASA in round 2 66
4.15 The minimum of indviduals vs. generations in 11$\sim$15 sets of GASA in round 3 67
4.16 The minimum of indviduals vs. generations in 16$\sim$20 sets of GASA in round 4 67
4.17 The minimum of indviduals vs. generations in 1$\sim$5 sets of GAs of round 1 68
4.18 The minimum of indviduals vs. generations in 6$\sim$10 sets of GAs in round 2 68
4.19 The minimum of indviduals vs. generations in 11$\sim$1 5 sets of GAs of round 3 69

4.20 The minimum of indviduals vs. generations in 16~2 0 sets of GAs in round 4 69

5.1 The distribution of the state density function ω 72

5.2 The min energy E_{min} vs. generation for different initial temperatures . 75

5.3 The mean energy $\bar{E}$ vs. generation for different initial temperatures . . 75

5.4 The population's standard deviation σ through generations for different initial temperatures 76

5.5 Minimum energy vs. generation for different annealing coefficients . . 79

5.6 Average energy vs. generation 80

5.7 Average energy vs. $\log(T_t)$ for different annealing coefficients 80

5.8 Standard deviation vs. temperature (semilog) 81

5.9 Mean objective value vs. generation for GAs and GASAs 86

5.10 Min objective value vs. generation for GAs and GASAs 86

5.11 Standard deviation vs. generation for GAs and GASAs 87

5.12 Variance vs. generation for GAs and GASAs 88

5.13 Annealing schedule for GASAs 88

5.14 Schemata $H_1, H_2, \cdots, H_m$ and their genetic models 92

5.15 Visualization of schemata as hyperplanes in 3-dimensional space . . . 94

5.16 An intuitive sketch of the relationship of populations 97

5.17 The min of individuals vs. generation 105

5.18 The mean of individuals vs. generation 105

5.19 The standard deviation vs. generation 106

5.20 The average energy <E> vs. temperature 106

5.21 The standard deviation vs. temperature (semilog) 107

5.22 The standard deviation of individuals vs. temperature (log-log) 107

5.23 Average energy <E> vs. min energy min(E) 108
5.24 Turbulence "average energy <E> - min energy min(E)" vs. temperature (semilog) 108
5.25 Turbulence "average energy <E> - min energy min(E)" vs. temperature (log-log) 109
5.26 Average energy <E> vs. min energy min(E) 109
5.27 Turbulence "average energy <E> - min energy min(E)" vs. temperature 110
5.28 Standard deviation vs. generation 110
5.29 Average energy <E> vs. min energy min(E) 111

6.1 An Illustration of GASA Feedback Controller 119
6.2 Specific growth rate μ and yield Y vs. substrate concentration s . . . 120
6.3 The mesh plot for fed-batch optimization with different ramp input patterns 123
6.4 The contour plot for fed-batch optimization with different ramp feed patterns 124
6.5 Different feed patterns 124
6.6 The volume profile 125
6.7 The substrate conc. profile 125
6.8 The cell mass profile 126
6.9 An Interpretation of GAs operation in generations: (a) The objective values of all individuals in generation 106. (b) Best and mean objective values of individuals in generations. 126
6.10 Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. 127

6.11 The profiles of volume and feed rate with GAs 128
6.12 The profiles of substrate conc. and cell mass with GAs 128
6.13 The product profile with GAs 129
6.14 An Interpretation of GASA operation in generations: (a) The objective values of all individuals in generation 133. (b) Best and mean objective values of individuals in generations. 129
6.15 Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. 130
6.16 The profiles of volume and feed rate with GASA 131
6.17 The profiles of substrate conc. and cell mass with GASA 131
6.18 The product profile with GASA 132

7.1 A neutralization process 137
7.2 Titration curve for neutralization of strong acid ($10^{-3}mol/l$) and strong base ($10^{-3}mol/l$) 138
7.3 A classical PID control system 139
7.4 A PID + GAs control system 140
7.5 A PID + GASA control system 141
7.6 pH vs. time for ZN tuning method 144
7.7 Performance function (PF) vs. time for ZN tuning method 144
7.8 pH vs. Time 145
7.9 The base flow rate profile 145
7.10 PF vs. Time 146
7.11 The profile of $[H^+] - [OH^-]$ 146

7.12 An Interpretation of GAs operation in generations: (a) PID parameters of best individuals in generations. (b) The objective values of all individuals in generation 201. (c) Best and mean objective values of individuals in generations. 150
7.13 Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. (d) Best pH vs. generation. 151
7.14 An Interpretation of GASA operation in generations: (a) PID parameters of best individuals in generations. (b) The objective values of all individuals in generation 21. (c) Best and mean objective values of individuals in generations. 152
7.15 Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. (d) Best pH vs. generation. 153

Chapter 1

Introduction

With today's challenge in the high-tech industries, and the continuing need for energy competitiveness, materials conversion, hazard free operation, and environmentally safe discharges, it has become increasingly important to develop and apply new methodologies and techniques that lead to improved performance and operation in chemical engineering. Whether an engineer is concerned with design, production planning, operation, or control, he or she needs to make predictions about the process and plant behavior. However, the complexity of chemical processes makes building accurate mathematical models difficult. In general, the dynamic characteristics of the chemical plant structure are unknown or poorly modeled at best. If we have some knowledge of the plant, the plant model can be determined by using system identification techniques for the plant coefficients and parameters. Mostly, it is only an approximation to the real plant, since the plant is not known exactly and is nonlinear.

The emergence of the concept "evolutionary computation" provides a totally different method to analyze the past data, and can be developed into a model-based search algorithm. This new novel search algorithm combines genetic algorithms and simu-

lated annealing, both of which are very promising search strategies. This algorithm is called **GASA** throughout this work. The search skill of GASA is extended to a powerful optimization technique. Also, it can be used to help improve another attractive learning method "artificial neural networks" by establishing new learning rules. The learning ability can help to reduce human effort required in analyzing complex dynamical process operations and it even suggests a potential for discovering better system schemes than presently known. Connectionists study the learning behavior of artificial neural networks in order to establish this relationship via the configurations of neural networks. The natural evolution in genetics and thermodynamics each provide the different configurations. This research tries to explore a new interdigitation: a hybrid technique for process dynamics by employing Genetic Algorithms (from genetics and evolution), and Simulated Annealing (from physical metallurgy). The combination of algorithms derived from natural principles can result in different learning rules. Finally, it is important that our new technique can be applied to the analysis and system design of dynamical chemical and biochemical processes.

1.1 Motivation of the Thesis

In the thesis proposal [96, Sun, 1993], it was indicated that this research intended to study the use of combination of genetic algorithms and simulated annealing, and other connectionist methods for a very broad spectrum of applications in dynamic system design and integration. The applications considered include dynamic optimization and control. However, the discussion below concentrates mainly on the algorithm design and dynamic system design.

Originally, this thesis was initiated to create a novel search algorithm from the

integration of algorithms from natural systems. It was hoped to develop alternatives which possibly offer some advantages over the conventional evolutionary and genetic computation methods [32, Fogel, 1993]. The main difference between genetic and evolutionary algorithms is at the genotype and phenotype. Genetic algorithms emphasize genotypic transformation, while evolutionary algorithms emphasize phenotypic adaptation. After a survey of the open literature on genetic algorithms, the population-based search was found to be an excellent candidate among all possible alternatives. It has been pointed out repeatedly by several researchers that a genetic algorithm has the capability of tracing a path to the global optimum using the information provided. Initially, this research intended to utilize the search characteristics of genetic algorithms for optimizing a dynamic process. However, as pointed out by several researchers on the convergence of genetic algorithms [82, Rudolph, 1990], the genetic operators are executed in these algorithms without any specific selection. However, another stochastic search algorithm, simulated annealing, offers a totally different search mode from genetic algorithms. The objective is to determine how to make a more effective selection in populations with the annealing schedule supplied by simulated annealing. Therefore, effort in this research was directed toward finding out whether a new search algorithm can offer a more stable and precise solution to the problem encounted when using the genetic algorithms.

The consideration of temperature schedule cannot be neglected in the explanation of theory. A typical annealing algorithm starts at some high temperature. The system is allowed to approach equilibrium, at which time the temperature is reduced, and the system is allowed to equilibrate repeatedly. Its basic feature is the possibility of exploring the configuration space by allowing "hill-climbing" moves, i.e., the generation of new configurations of the problem which increase the cost. These moves are

controlled by a parameter, which is analogous to temperature in the annealing. These moves become less and less likely towards the end of process where the temperature is low. This change of configuration is helpful to divide the individuals in populations by the location of their energy levels. This division pushes our genetic approach to do a second selection after they are replicated. This combination of algorithms is investigated in the search phase of individuals. Their movement is similar to a phase transition in their domain. Schema theory is combined with our simulated phase transition to find the individual's moving trajectory.

Later, the use of GASA in dynamic system design is shown to be equivalent to one of the two major optimization approaches in classical static optimization where functions are used. The other dynamic optimization for nonlinear systems are investigated with this parallel, global search technique. The separate applications of genetic algorithms and simulated annealing in nonlinear programming problems and several engineering problems have been successfully investigated by many researchers. In terms of practical applications, the challenging problem of the control of fed-batch fermentation is investigated. Fed-batch processes are technically challenging to control: the process variables are difficult to measure, the "quantity" of the product is difficult to define yet it is very important, the process model usually only approximates and contains strongly time-varying parameters, etc. But, above all, the challenge arises because optimization of the feed rate is a **dynamical** problem. Many researchers use the maximum principle to solve these problems. When this method is applied, the consideration of singularity becomes very important especially for the certain system models and applied constraints. The other application in this thesis is to develop a new control strategy by combining GASA and PID control. When controllers are designed without having an accurate mathematical model of the system to be controlled, two

questions arise: first, how is the **structure** of the controller is **establish**, and second, how are the **numerical values** of the controller parameters chosen. In solving the first problem, many techniques have proved successful, for example, expert systems and machine learning with neural networks have been used. In contrast, the second problem is solved on an ad hoc. The GASA technique can be used for both **learning the control structure** and **tuning the control parameters**. Stability is an important part in the analysis and design of a control system. A standard approach for robustness analysis is to examine the characteristics of the control system in the presence of parametric uncertainties. This type of problem solving scheme is addressed in the thesis.

As the research proceeded, the GASA approach was found to have significant advantages over the genetic algorithms. For example, the GASA approach can lead to a much higher precision and less oscillation than the genetic algorithms. In addition, further investigation has revealed that the developed GASA approach holds great promise in many situations, such as when the genetic algorithm is used in static and dynamic optimization, or when singularity is introduced in the model. Inevitably, however, the proposed GASA algorithm was found to have a significant drawback. In particular, the training speed is extremely slow in many cases because of the addition of annealing compared with genetic algorithms. Training with GASA takes as much as 50 CPU time units to converge on a Sun SPARC station for a simple function optimization problem. In order to circumvent this drawback while attempting to retain the advantages of this proposed method, there are at least three approaches that can be taken:

- Use a real-value coding approach,

- Use an effective annealing schedule, or
- Employing a "quench" schedule.

The first approach is proposed because presently there is no appropriate approach for binary coding other than using a set of real-valued coding. The second approach has often been used by several researchers when solving combinatorial problems. The last approach was also considered at an earlier stage of this research [51, Ingber, 1992]. However, it is only mentioned briefly in this thesis because the quench approach is a form of annealing and takes many few steps than annealing. Thus it is not discussed here.

Finally, the goal of developing a GASA architecture is to use the genotypic transformation to code the state variables or control parameters. While the process industries have faced increasing competition from abroad, control or dynamic optimization has become one of the most effective approachs to keep up with the changing demand and unpredictable marketplace. In most of the fermentation industries, optimal conditions can be found by minimizing (or maximizing) a cost criterion subject to the constraints imposed by mathematical models. These models contain many singular points which cause difficulty when trying to control them.

1.2 Organization of the Thesis

The organization of this thesis is described below. Chapter 3 starts with a brief introduction to the basic algorithms — genetic algorithms, simulated annealing and other search algorithms. The discussion is mainly focused on how to combine these two structured algorithms. These combined architectures establish the basis of the entire dissertation. A unified approach for these two algorithms is proposed, and an

associated learning algorithm is presented as well. Chapter 4 describes the architecture of this combined technique (GASA) and how to use it for design. A simple application in function optimization is also included in this chapter to show the efficiency of the proposed algorithm. This chapter discusses the topic of algorithm design using a division approach, which is an alternative to the simulated annealing. As to the key role of temperature and the search mode of GASA algorithm, Chapter 5 discuss the search phase theory which comes from the schema theory and the proposed simulated phase transition. This search mode results from the exchange of different phases. A phase is similar to a population here. When individuals change their location, they shift to another plane (phase) in the convex surface. The individuals change from one phase to another as the temperature moves toward equilibrium. While moving towards equilibrium, the individuals become the parents of their next generations. Schema theory were proposed by John Holland in his benchmark book in 1975 [46, Holland, 1975] to explain how GAs perform so well. This theory can partially explain the characteristics of phase transition by introducing the temperature in the search mode. Chapter 6 and Chapter 7 describe the application of the GASA technique in system design. In our example problems, we only consider dynamic systems, whether they are linear or nonlinear. Time-varying factors and the given constraints sometimes result in complex models. Here we choose validated models from chemical and biochemical processes, involving fed-batch fermentation, where the feed-rate optimization problem is considered in Chapter 6. In Chapter 7, feedback tuning achieved with the GASA approach helps to stabilize a unstable pH plant. In this case, a GASA PID controller is applied to this system. The convergence analysis is considered to characterize the genetic-annealing control design. Finally, Chapter 8 summarizes the accomplishments made in the thesis. Some recommendations for

future work are also given. A promising design of parallelized GASA may possibly be used in automata or cell placement to design a parallel machine. Additionally, a notation list is provided in Appendix A.

Chapter 2

Claims Section

The original contributions of the thesis are:

1. Developed a new search algorithm — GASA

   This thesis is based on my newly developed search algorithm, called **GASA**. The algorithm uses a combined scheme from genetic algorithms (GAs) and simulated annealing (SA). The characteristics of simulated annealing are used to classify the populations created from a genetic approach. In this algorithm, **individual** is the same as the **chromosome** in biology. It is part of an evolutionary computation method. The individuals are unconditionally forced to undergo the genetic operations of crossover and mutation in traditional genetic algorithms. The GASA approach uses a division method for the individuals in the whole population. During division, a simulated annealing approach is used. The energy level of individuals determines whether they should be processed by the genetic operators. In turn, temperature determines the energy level. The design and performance of this population-based search approach are discussed in Chapter 4. The high precision and speed of this new algorithm are demonstrated

by practical problems used as examples.

2. Examined and explained a new search scheme

   The schema theory and simulated phase transition are used to discuss the search trajectory of GASA. The schema theory originally comes from the discussion of human social learning in psychology. In 1975, John Holland defined the term to discuss the search ability of his first algorithm. The simulated phase transition is a new description of the population behavior similar to the physical phase transition (e.g. solid, liquid, gas). The **phase** is defined as the generated population. They will move toward equilibrium until they meet the Gibbs distribution. The principles applied for the exploration of the new search scheme are inspired from the observation of Nature.

3. Designed and tuned the algorithm developed from genetic and annealing mechanism

   The design and tuning mechanism extracts genetic operators from biological genetics and annealing from physical metallurgy. The tuning mechanism has two stages: one from genetic operation including crossover and mutation, the other is annealing. The genetic operators generate new diversity for the population. Too much diversity would result in difficulty of convergence for all genetic approaches. With the adjustment of temperature, GASA can converge easier when compared with classical genetic algorithms. Thus, it is important to be able to design of the temperature schedule, because it is related to the running speed and precision in simulation.

4. Developed a new scheme for dynamic optimization with the GASA Algorithm

   The approach (GASA) for genetic algorithms through an annealing process

is used to design a new scheme of dynamic optimization. The objective is to optimize the input function for dynamic optimization problems. The application is the feed rate optimization of a fed-batch bioreactor. After searching generation by generation, the optimum input profile is found by the strong search ability of GASA.

5. Developed a new control strategy with GASA Algorithm

   The new control strategy is developed with GASA which determines optimized contol parameter by its search capability. The GASA technique is applied to conventional PID control. The PID parameters are tuned with GASA through the temperature schedule.

Chapter 3

Search Algorithms and Nonlinear Programming

3.1 An Overview of Search Algorithms

The basic search algorithms as illustrated in Fig. 3.1 can be classified in general into three broad classes [38, 28, Goldberg, 1989;Filho et al, 1994] which are "enumerative", "calculus-based" and "guided random" search. Their relationship can be found from the family tree of general search methods.

Enumerative search algorithms search each point within an objective function's domain space (finite or discrete) one point at a time. They are very simple to implement but may require significant computation. The domain space of many applications are too large to search by using these techniques. A good example of an enumerate technique is **dynamic programming**.

Calculus-based search algorithms apply a set of sufficient/necessary conditions to be satisfied by the optimal solutions of an optimization problem. These approaches sub-divide into **direct** and **indirect** methods. The direct methods (e.g., Newton and Fibonacci) look for the extrema by "hopping" around the search space and assessing

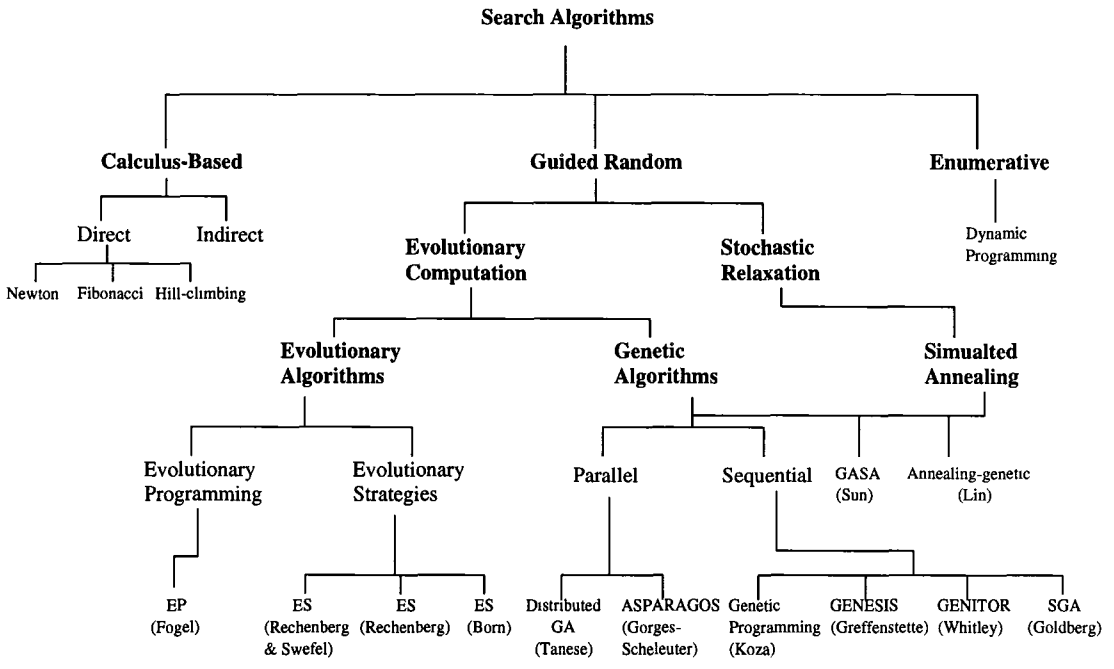


Figure 3.1: *Classes of search algorithms*

the gradient of the new point, which guides the direction of the search. This concept **hill-climbing** which finds the best local points by "climbing" the steepest permissible gradient. The indirect methods seek for the local extrema by solving the usually nonlinear equations resulting from setting the gradient of the objective function equal to zero. The search for possible possible solutions (function peaks) starts by restricting the search to points with zero slope in all directions. The hill-climbing methods suffer from the problem that the first peak found will be climbed, and this may not be the highest peak. Having reached the top of the local extrema, no further progress can be made. Thus, the algorithms can only succeed on a restricted set of "well behaved" problems.

Guided random search algorithms are based on enumerative techniques, but use additional information to guide the search. They are quite general in their scope,

being able to solve very complex problems. Most of them are stochastic processes resulting from the use of random probability. Two major sub-groups are: **evolutionary computation (EC)** and **stochastic relaxation (SR)**. Stochastic relaxation uses a stochastic process and a relaxation method to search. Simulated annealing is one of the best known algorithms, which uses a thermodynamic evolution process to search for minimum energy states. Evolutionary computation, on the other hand, is based on the natural selection principles. Furthermore, evolutionary computation can be sub-classified into **genetic algorithms (GAs)** and **evolutionary algorithms (EAs)** (i.e. evolutionary strategies [79, 86, Rechenberg, 1973; Schwefel, 1977] and evolutionary programming [33, Fogel et al., 1966]). The former emphasize the genotypic transform, the latter emphasize the phenotypic adaptation. The computation search approach evolves through **generations**, improving the features of potential solutions by means of biologically inspired operations.

The details of simulated annealing and genetic algorithms will be introduced in the next sections.

3.2 Genetic Algorithms

Evolution is a remarkable problem solving scheme. The field of evolutionary computation has approached a stage of maturity in the last few decades. It is an attractive class of computational models that attempts to mimic the mechanisms of natural evolution or genetics to solve problems in a wide variety of domains.

Using a simulation of the evolution process with a digital computer, daunting search problems has been attempted since the 1960s [14, 33, 80, Bremermann, 1962; Fogel & Owens & Walsh ,1966; Reed & Toombs & Barricell, 1967].

In the mid-1970s, Holland [46, Holland, 1975] introduced and analyzed the expected behavior of a particular genetic algorithm that employed a population of trial solutions, fitness-proportional reproduction and the production of offspring by crossover (asexual recombination), inversion and mutation. Theoretical analyses suggest that genetic algorithms can quickly locate high performance regions in extremely large and complex search spaces. Furthermore, some natural insensitivity to noisy feedback is expected because of the distributed and repeated sampling in the population. These expectations are based on the insight that genetic algorithms simultaneously balance a preservation of **building blocks** (constituent parts of the objects being tested) that have consistently been found in the better than average parents while they generate and test new building blocks in the offspring [38, 47, Goldberg, 1989; Holland, 1992]. Following Holland's original genetic algorithms, many variations of the basic algorithms has been introduced [38, Goldberg, 1989]. Genetic algorithms are often viewed as function optimizers, although the range of problems to which genetic algorithms have been applied is quite broad [107, Whitley, 1993].

3.2.1 Description

Genetic Algorithms (GAs) can be defined as optimization techniques based on the mechanisms of natural selection and genetics. Their search sampling is not limited to a simple point but to a pool or population. Though they used randomized techniques, they differ substantially from random searches.

Genetic algorithms differ from traditional optimization in four ways [38, Goldberg, 1989]:

- GAs work with a coding of the parameter set, not the parameters themselves.

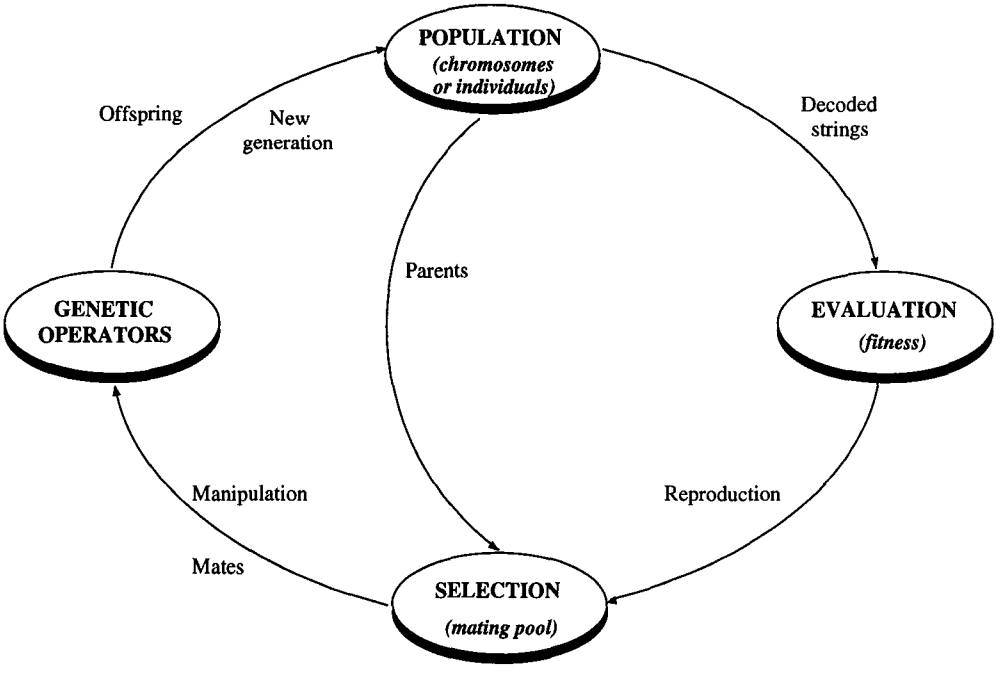


Figure 3.2: *The GAs cycle*

- GAs search from a population of points, not a single point.
- GAs use payoff (objective or cost function) information, not derivatives or other auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules.

Fig. 3.2 illustrates the four stages of the process using the biologically inspired GAs terminology. In each cycle, a new **generation** of the candidate solutions for a given problem is produced. At the first stage, an **initial population** of the potential solutions is created as a starting point for the search process. Each element of the population is **encoded** into a string **chromosome**, to be manipulated by the genetic operators. In the next stage, the performance (or **fitness**) of each individual of the population is **evaluated**, with respect to the constraints imposed by the problem. Based on

the each individual's fitness a **selection** mechanism chooses "mates" for the **genetic manipulation** process. The selection policy is ultimately responsible for assuring survival of the best fitted individuals. The applied genetic operators are "crossover" and "mutation", both of which result in the structural change for offsprings in the new generation.

In Genetic Algorithms the natural parameter set of the optimization is coded into a finite-length string(s) over a finite alphabet. These **strings** can be viewed as the **chromosomes** in the natural systems analogy. The set of strings or **structure** is the **genotype**.

In natural systems the organism formed by the interaction of the genotype with its environment is the **phenotype**, where in artificial systems it is called **parameter set**, **solution alternative**, or **point** (in the solution space). The chromosomes in natural systems are composed of **genes** (bits) which may take on values called **alleles** (0 or 1) in artificial system. The position of the gene is called the **locus** (plural **loci**). In artificial systems the strings are composed of **features** or **detectors** which take on different **values** located in different **positions**.

3.2.2 Genetic operators

The basic structure processed by GAs is the **string**. Strings (generally called **individuals**) are composed of a sequence of characters of finite length λ composed over some alphabet V. Strings can be represented as follows:

$$A = a_1 a_2 \cdots a_\lambda$$

Strings of the current population are then manipulated to generate a new population in the next time step. This is done by the use of the transition rules, namely, reproduction,

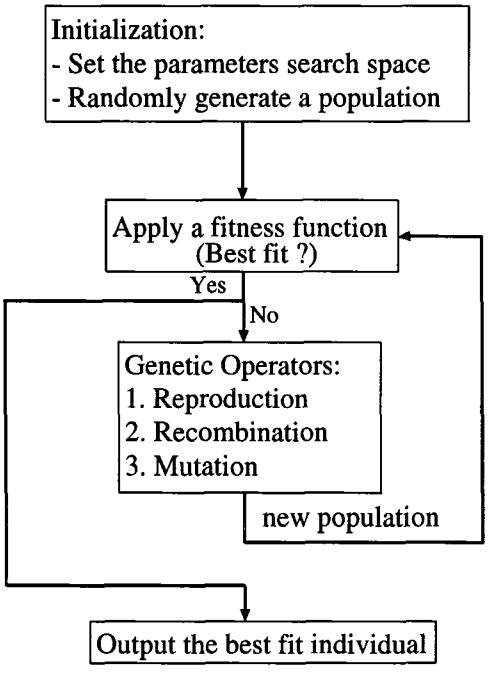


Figure 3.3: *An Outline of the Genetic Algorithms*

crossover, mutation [25, DeJong, 1975]. The three main genetic operators are:

- **Reproduction:**

  A process determines the number of copies of an existing string to be made during a new population. The selection procedure probabilistically selects an individual i to remain in the population and reproduce with probability $p_i = f_i / \sum_{j=1}^{n} f_j$. The fitness f_i is calculated from the objective function. Those states not selected are culled from the population. Because the average fitness of the population is defined as $\bar{f} = \sum_{j=1}^{n} f_j / n$, if there are $b_i(t)$ copies of an individual i at time t, the new population will have $b_i(t+1) = b_i(t) f_i / \bar{f}$ copies of f_i. The effect of this reproduction scheme is that above average performing individuals reproduce, replacing poorly performing individuals.

The test, select, and reproduce operators are applied at each iteration (generation), and replace the standard generate and test paradigm from Darwin's benchmark work. That means that the strings (coding points in the search space) with a higher fitness value have a higher probability of contributing to one or more offsprings in the next generation. This is the artificial version of Darwin's natural selection, which guarantees improvement over generations.

- **Crossover (Recombination):**

  This is a process in which a substring is swapped with the corresponding substring of another string (its mate). By this process, new structures (genotypes) are generated with new fitness values (hopefully better in average). The length of the string to be crossed-over is selected at random. Then, an integer k along the string is selected at random, with uniform distribution, in the interval, $[1, \lambda - 1]$. Then the two strings are created at random, by exchanging all the characters between positions k and λ inclusively. As an example consider:

$$\begin{aligned} A &= a_1\, a_2\, a_3 | a_4\, a_5\, a_6 \\ B &= b_1\, b_2\, b_3 | b_4\, b_5\, b_6 \end{aligned}$$

  For $k = 3$, the resulting crossover will be:

$$\begin{aligned} A' &= a_1 a_2 a_3 b_4 b_5 b_6 \\ B' &= b_1 b_2 b_3 a_4 a_5 a_6 \end{aligned}$$

  Crossover results in a randomized yet structured information exchange. Each solution created combines the characteristics of both parents.

- **Mutation:**

Although this operation plays a secondary role in GAs, it has proven very useful and necessary (both in nature and in GAs). It is needed to introduce noise in the system and to "kick" it periodically so that it does not get stuck in local optima (a difference with the annealing process). Mutation randomly flips a bit in an individual's bit's string representation. The resulting effect allows the population to sample states away from its means, preventing the population from converging and stagnating at any minima. Its real purpose is the recreation of good genes that were lost by change through poor selection of mates. In other words, the probability of mutation should be tiny enough (not zero) to protect the valuable gene pool from wanton destruction.

3.2.3 Other genetic alternatives

Other operators such as inversion have been used in PMX (Partial Matched Crossover) [39, Goldberg & Lingle, 1985] and Frantz's study of eptistasis [35, Frantz, 1972]. Inversion is also the primary natural mechanism responsible for recoding a problem. Inversion is defined by taking a random section of code and reversing its position index, i.e.

$$\begin{aligned} A &= a_1|a_2a_3a_4a_5|a_6 \\ A' &= a_1|a_5a_4a_3a_2|a_6 \end{aligned}$$

The mathematical foundations of GAs are based on the concepts of **schema** (plural **schemata**) and **building blocks**. Loosely speaking, the schema is a collection of genes in a chromosome having certain specified values. For our purposes, a schema is a small set of the genes in a chromosome which, when they take on specified values, act as a unit to produce an effect [48, 46, Holland, 1968; 1975]. The picture of GAs'

performance is much clearer with the perspective afforded by schemata. Because highly fit schemata of low defining length and low order play such an important role in the actions of GAs, they have a special name: **building block**.

The reproduction, crossover and mutation cycles continue until an acceptable solution is found, until the average population fitness ($\bar{f}$) converges to a stable fixed set of points, or until a fixed number of generations have evolved. In the mean time, it completes the generations' cycles (see Fig. 3.3). However, GAs are not guaranteed to find the global functional optima because: (1) the precision limits in the encoding process can substantially reduce the solution accuracy, and (2) the search process does not ergodically (weakly) cover and search the state space [51, Ingber, 1992]. The next introduced algorithm can elude local minima and reach global minima statistically.

3.3 Simulated Annealing

Simulated Annealing (SA) is a stochastic computational technique derived from statistical mechanics for finding a near globally-minimum-cost solution to large optimization problems. Kirkpatrick et al. (1983) were the first to propose and demonstrate the application of simulation techniques from statistical physics to optimization problems, specifically to the problems of wire routing and component placement in VLSI design.

In general, finding the global minimum value of an objective function with many degrees of freedom subject to conflicting constraints is a NP (NonPolynomial)-problem with many local minima. A procedure for solving difficult optimization problems should sample values of the objective function in such a way as to have a high probability of finding a near-optimal solution and should also lend itself to efficient implementation. Over the past few years, Simulated Annealing has emerged as a

viable technique which meets the computational model criteria above. Ackley [5, Ackley et al., 1985] provided a comprehensive assessment of the relative performance of simulated annealing applied to a variety of optimization problems.

3.3.1 Description

Simulated Annealing accepts and rejects randomly generated **moves** on the basis of a probability related to an **annealing temperature**. It can accept moves which change the value of an objective function in the direction opposite to that of the desired long-term trend. Thus, for a global minimization problem, a move that increases the value of the objective function (an **uphill** move) may be accepted as part of the full series of the moves for which the general trend is to decrease the value of the objective function. In this way, Simulated Annealing is able to explore the full set of solutions which are independent of the starting point. This means that one does not become trapped in a far from optimal local minimum as is possible with algorithms based on a gradient-descent method.

Simulated Annealing differs from conventional optimization, because it can:

- process objective functions possessing quite arbitrary degrees of nonlinearities, discontinuities, and stochasticity.
- process quite arbitrary boundary conditions and constraints imposed on the objective function.
- be implemented quite easily with the degree of coding quite minimal relative to other nonlinear optimization algorithms.
- statistically guarantee finding an optimal solution.

Sometimes Simulated Annealing (SA) has been misused and transformed into Simulated Quench (SQ) [52, Ingber, 1992] for which there is no statistical guarantee of finding an optimal solution.

3.3.2 Thermal equilibrium in simulated data transformation

Statistical mechanics is the base for studying the behavior of annealing, such as atoms (data points) in a fluid (a data set), in thermal equilibrium at a finite temperature. Suppose that the state of the system is identical with the set of spatial positions of the components. Here, the phase transition is defined as data transform between **ordered** (low energy) and **disordered** (high energy) phases. The **ordered phases** obey certain types of data arrangements. If the system is in thermal equilibrium at a given temperature T, then the probability of a given state s depends upon the energy $E(s)$ of the state and follows the Boltzmann distribution:

$$\pi_T(s) = \frac{e^{\frac{-E(s)}{KT}}}{\sum_{\omega \in S} e^{\frac{-E(\omega)}{KT}}} \tag{3.1}$$

where K is Boltzmann constant and S is the set of all possible states.

One can simulate the behavior of a system of particles in thermal equilibrium at temperature T using a stochastic relaxation technique developed by Metropolis et al (1953) . Suppose that at time measure t, the system is in state q. A candidate r for the state at time $t+1$ is generated randomly. The criterion for selecting or rejecting state r depends on the difference between the energies of states r and q. Specially, one computes the ratio $h(\triangle E)/T$ (reset $T = KT$), and $\triangle E = (E(r) - E(q))$, where:

$$h = \frac{\pi_T(r)}{\pi_T(q)} = e^{\frac{-\triangle E}{T}} \tag{3.2}$$

If $h > 1 (\triangle E < 0)$, that is the energy of r is strictly less than the energy of q, then the state is automatically accepted as the new state for time $t+1$. If $h \leq 1$, that is,

$E(r) \geq E(q)$, then the state r is accepted as the new state with probability h. Thus, states of higher energy can be attained. It can be shown that as $t \to \infty$, the probability that the system is in a given state s equals $\pi_T(s)$, regardless of starting state, and thus the distribution of states generated converges to the Boltzmann distribution [36, Geman & Geman, 1984].

3.3.3 Annealing process

In studying the moves of particles, one often seeks to determine the nature of the low-energy state, for example, whether freezing produces glassy or crystalline solids. Very low energy states are not common, when considering the set of all states. To achieve low-energy states, it is not sufficient to simply lower the temperature of the system. The temperature must be gradually lowered, spending enough time at each temperature to reach thermal equilibrium. If insufficient time is spent at each temperature, especially near the freezing point, then the probability of attaining a very low energy state is greatly reduced.

Before applying the simulated annealing procedure to optimization, the following preparatory steps are required. One must

- identify the analogues of the physical concepts in the optimization problem itself: the energy function becomes the cost (objective) function, the states of particles become the states of the parameter values, finding a low-energy state becomes seeking a near-optimal solution, and the temperature becomes the control parameter for the process.

- select an annealing schedule consisting of a decreasing set of temperatures together with the amount of time to spend at each temperature.

- have a method of generating and selecting new states.

Kirkpatrick's annealing algorithm [55, Kirkpatrick, 1983] consists of running a Metropolis Monte Caro integration algorithm at each temperature in the annealing schedule for the amount of time prescribed by the schedule, and selecting the final state generated as a near-optimal solution. The Metropolis algorithm, which accepts states that increase cost as well as those that decrease cost, is the mechanism for avoiding entrapment at a local minimum.

The annealing process is inherently slow. Geman and Geman (1984) determine an annealing schedule sufficient for convergence. Specially, for a given sequence of temperatures $\{T_t\}$ such that $T_t \rightarrow 0$ as $t \rightarrow \infty$ and $T_t \geq \frac{T_0}{logt}$ for a large constant T_0 (start temperature), then the probability that the system is in state s as $t \rightarrow \infty$ is equal to $\pi_0(s)$. Others have worked on improving this bound [44, 37, 50, Hajeck, 1985; Gidas, 1985; Ingber, 1989].

At the outset it must be stated that SA is not without its critics. The primary criticism is that it is too slow, which comes from the annealing process itself. Another criticism is that it is **overkill** for many of the problems on which it is used [52, Ingber & Rosen, 1992]; this is partially addressed here by summarizing a lot of work which demonstrates that it is not insignificant that many researchers are using SA because of the ease in which constraints and complex cost functions can easily be approached and coded [16, 75, 52, Charnes & Wolfe, 1989; Pincus, 1970; Ingber & Rosen, 1992].

3.4 Applications of Evolutionary Computation and Further Challenges

Some examples of GAs and SA applications were mentioned in the Chapter 1. To illustrate the flexibility of GAs, here we list some more. Some of these applications have been used in practice, while others remain as research topics.

3.4.1 Applications of evolutionary computation

This section will discuss the main area of applications of GAs and SA. Here we review some of the related applications.

The major application areas of genetic algorithms (GAs) are:

- Numerical function optimization:

  Most traditional research has concentrated in this area. GAs and SA have been shown to be able to outperform conventional optimization techniques on difficult, discontinuous, multimodal, noisy functions [25, 21, DeJong, 1975;Davis, 1991].

- Dynamic system design:

  The GAs attract many engineers and scientists working on the design of dynamic systems. The application of genetic algorithms for system identification and control were recently proposed by many researchers [58, 85, Kristinsson, 1992;Schmitendorf et al, 1992]. Work combining GAs and Fuzzy control can be found by Park et al[74, Park et al, 1994]. Controller design and tuning with GAs was done by Varšek et al[101, Varšek et al, 1993]. Recursive adaptive filter design uses genetic algorithms [27, Etter et al, 1982]. Optimal design of

PID process controllers has also been based on genetic algorithms [103, Wang & Kwok, 1993].

- Combinatorial optimization:

  These tasks require solutions to problems involving arrangements of discrete objects. This is quite unlike function optimization, and different coding, recombination, and fitness function techniques are required. Probably the most widely studied combinatorial task is the **travelling salesperson problem** (TSP) [39, 40, 64, Golderg, 1985;Gorges-Schleuter, 1989;Liepins & Hillard, 1989].

- Machine learning:

  There are many applications of GAs to learning systems, the usual paradigm being that of a **classifier system**. The GAs tries to evolve (i.e. learn) a set of **if** $\cdots$ **then** rules to deal with some particular situation. This has been applied to game playing [9, Axelrod, 1987].

Other applications using GAs are image processing [38, Goldberg, 1989], molecular prediction [98, 104, 20], and genetic algorithms for training neural networks [56].

Simulated annealing has been applied to problems in computer design [55, 102], imagine restoration and segmentation [36], combinatorial optimization such as TSP [54, Kirpatrick, 1984], and artificial intelligence [45, Hinton & Sejnowski, 1983].

3.4.2 Challenges for evolutionary computation

Many chemical processes are strongly nonlinear. For example, a process involves physics or chemistry of a process such as in supercritical extraction, in which complex phase behavior leads to sensitive dependence of operation on operating conditions and

control. The case of nonlinear time-varying plants the problem can be more difficult, with few available results. In most cases, the actual dynamical performance during process operation typically will not match the desired performance, due to model uncertainty, or drift and aging of components. For these reasons, there is a need for design techniques that will (i) overcome the uncertainty in conventional design and, (ii) account for modeling uncertainties that are exhibited during system operation. Most of the above problems can be solved with the aid of nonlinear programming. This in turn brings us to the study of this evolutionary computation.

Most of these problems are concentrated on how to reach the objective value of a given problem. These problems may occur in process control, dynamic optimization or steady state optimization. The most challenging problem is the dynamic optimization of an unknown input function especially for nonlinear dynamic systems. The popular methods for solving these problems, like Pontraygin maximum principle, are not sufficient conditions for the solution. While solving the applied problem, gradient optimization methods still use the traditional steepest descent method. This method easily converge to a local minima if they exist.

The idea of incorporating problem specific knowledge into genetic algorithms is not new and has been recognized for some time. Several researchers have discussed initialization techniques, different representations, and the use of heuristics for genetic operators. Many researchers try to use other methods, like neural networks, to overcome problems without a **priori** knowledge. Unfortunately, their weighting methods (the learning rules) still mostly use hill-climbing methods. That is, their weights may not be the optimal ones for the neurons. Thus, finding a new optimization method is the inevitable task for people who want to achieve high precision. This is the goal of how we will utilize the combined guided random search method to improve it. As De

Jong observed [24, De Jong, 1994] recently:

> "... the field had pushed the application of simple GAs well beyond our initial theories and understanding, creating a need to revisit and extend them.

3.5 Miscellaneous Evolutionary Computation Algorithms

Other formats of evolutionary computation algorithms shown in Fig. 3.1 will be briefly introduced in this section.

3.5.1 Evolutionary algorithms (EAs)

Evolutionary algorithms use computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. A variety of evolutionary computational models have been proposed. They share a common conceptual base of simulating the evolution of individual structures via the processes of **selection**, **mutation**, and **reproduction**. The processes depend on the perceived **performance** of the individual structures as defined by an **environment**.

More precisely, EAs maintain a POPULATION of structures, that evolve according to rules of **selection** and other operators, that are referred to as "search operators", (or genetic operators), such as mutation. Each individual in the population receives a measure of it's fitness in the environment. Reproduction focuses attention on high fitness individuals, thus exploiting (cf. EXPLOITATION) the available fitness information. Only mutation perturbs those individuals, providing general heuristics for

EXPLORATION. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

The main difference between EAs and GAs is with/without the genetic operator — **crossover**. We can find this from the above EAs algorithm and GAs. The basic EA method involves 3 steps (Repeated until a threshold for iteration is exceeded or an adequate solution is obtained):

1. Choose an initial population of trial solutions at random. The number of solutions in a population is highly relevant to the speed of optimization, but no definite answers are available as to how many solutions are appropriate (other than > 1) and how many solutions are just wasteful.

2. Each solution is replicated into a new population. Each of these offspring solutions are mutated according to a distribution of mutation types, ranging from minor to extreme with a continuum of mutation types in between.

3. Each offspring solution is assessed by computing its fitness. The N best solutions, or "stochastically" N of the best solutions, are retained for the next population of solutions.

3.6 Structured Evolutionary Computation for Nonlinear Programming

3.6.1 Nonlinear programming Environment

The general nonlinear programming problems NLP can be considered to find $\vec{X}$ so as to

$$\begin{aligned} &optimize\ f(\vec{X}),\ \vec{X} = \{x_1, x_2, \cdots, x_q\} \in \mathcal{R}^q \\ &\qquad subject\ to\ p \geq 0\ equations: \\ &\qquad c_i(\vec{X}) = 0, i = 0, \cdots, p. \end{aligned} \tag{3.3}$$

This is the best known method of determining the global maximum (or minimum) for a global nonlinear programming problem. Only if the objective function f and the constraints c_i satisfy certain properties, can the global optimum be found. These nonlinear programming problems are found in the area of control, modeling, optimization, detection and pattern recognition [11, 12, 7, Betts, 1977;Biggs, 1975;Bryson & Ho, 1975]. Many of the algorithms mentioned previously were developed for unconstrained problems (i.e. direct search method, gradient method) and constrained problems (the algorithms usually classified as direct and indirect methods). An indirect method attacks the problem by extracting one or more linear problems from the original one, whereas a direct method tries to determine successive points. Despite the active research and progress in global optimization in recent years [30, Floudas & Pardalos, 1992], it is probably fair to say that no efficient solution procedure is in sight for the general nonlinear programming problems NLP.

There are many other problems connected with traditional optimization techniques. In fact, many proposed methods are local in scope, they depend on the existence of

derivatives, and they are insufficiently robust in discontinuous, vast multimodal, or noisy search spaces [38, Goldberg, 1989]. It is important to investigate other (heuristic) methods, which, for many real world problems, may prove useful. In the following sections, structured algorithms are proposed from the genetic algorithms , evolutionary algorithms and simulated annealing.

3.6.2 A structured genetic algorithm

Genetic algorithms are used for a number of different application areas. An example of this would be multidimensional OPTIMIZATION problems in which the character string of the CHROMOSOME can be used to encode the values for the different parameters being optimized.

In practice, therefore, we can implement this genetic model of computation by having arrays of bits or characters to represent the chromosomes. Simple bit manipulation operations allow the implementation of CROSSOVER, MUTATION and other operations. Although a substantial amount of research has been performed on variable- length strings and other structures, the majority of work with Genetic algorithms is focussed on fixed-length character strings. We should focus on both this aspect of fixed-lengthness and the need to encode the representation of the solution being sought as a character string, since these are crucial aspects that distinguish Genetic Algorithms from **Genetic Programming**, which does not have a fixed length representation and there is typically no encoding of the problem.

When the genetic algorithm is implemented it is usually done in a manner that involves the following cycle in Fig. 3.2: evaluate the fitness of all of the individuals in the population, create a new population by performing operations such as crossover,

fitness-proportionate reproduction and mutation on the individuals whose fitness has just been measured. Discard the old population and iterate using the new population.

One iteration of this loop is referred to as a generation. There is no theoretical reason to use this as an implementation model. Indeed, we are not able to observe the "nonstop" behavior in populations in nature as a whole, even though it is occurring, but it is a convenient implementation model.

The first generation (generation 0) of this process operates on a population of randomly generated individuals. From there on, the genetic operations, in concert with the fitness measure, operate to improve the population.

The pseudo code of GA is described as below.

procedure Algorithm GA

$t = 0$;

/* initialize a usually random population of individuals */

initialize population $P(t)$;

/* evaluate fitness of all initial individuals of population */

evaluate $P(t)$;

/* test for termination criterion (time, fitness, etc.) */

while (**not** termination-condition)

/* increase the time counter */

$t = t + 1$;

/* select a sub-population for offspring production */

$P(t) =$ selectparents $P(t-1)$;

/* recombine the "genes" of selected parents */

recombine $P(t)$;

/* perturb the mated population stochastically */

mutate $P(t)$;

/* evaluate its new fitness */

evaluate $P(t)$;

end;

During iteration t each solution, $\vec{X}_i$, from the population (the population size remains fixed through the evolution space) is evaluated by computing $f(\vec{X}_i)$, a measure of its fitness. A new population, $P(t+1)$, is then formed: we select the candidate solutions to reproduce on the basis of their relative fitness, and the selected solution are recombined using genetic operators to form the new population.

3.6.3 A structured simulated annealing

Simulated annealing generalizes the hill-climbing methods and eliminates their main disadvantage: dependence of the solution on the starting point, and statistically promises to deliver an optimal solution. This is achieved by a probability, p_T, of acceptance: $p_T = 1$, if the new point provides a better value of the objective function; however, $p_T > 0$, otherwise. In the latter case, the probability of acceptance p_T is a function of the values of the objective function for the current point and the new point, and an additional control parameter, "temperature", T. In general, the lower temperature T is, the smaller the chances for the acceptance of a new point are. During execution of the algorithm, the temperature of the system, T, is lowered in steps. The algorithm terminates for some small values of T, for which virtually no changes are accepted any more. The structure of the procedure simulated annealing is given as follows.

procedure Algorithm SA

$t = 0$;

initialize temperature T;

select a current string $\vec{X}_c$ at random;

evaluate $\vec{X}_c$;

while (**not** stop-criterion)

  while (**not** termination-condition)

    select a new string $\vec{X}_n$

      in the neighborhood of $\vec{X}_c$

      by flipping a single bit of $\vec{X}_c$;

    if $f(\vec{X}_c) < f(\vec{X}_n)$

      $\vec{X}_c \leftarrow \vec{X}_n$;

      else if random$[0, 1) < exp\{(f(\vec{X}_n) - f(\vec{X}_c))/T\}$

      $\vec{X}_c \leftarrow \vec{X}_n$;

      end;

    end;

  end;

  $T \leftarrow g(T, t)$;

  $t \leftarrow t + 1$;

end;

The random number generator random$[0, 1)$ returns a random number from the range $[0, 1)$. The termination-condition checks whether "thermal equilibrium" is reached, i.e., whether the probability distribution of the selected new strings approaches

the Boltzmann distribution [1, Aart & Korst, 1989]. However, in some methods of implementation [4, Ackley, 1987], this repeat loop is executed just k times (k is a user-defined parameter).

The temperature T is lowered in many steps ($g(T,t) < T$ for all t). This algorithm terminates for some small values of T: the (stop-criterion) checks whether the system is "frozen", i.e., virtually no changes are accepted when T is small enough.

This algorithm is able to incorporate penalties for violated constraints or reject nonfeasible strings $\vec{X}_n$. For example, the code for **VFSR** (Very Fast Annealing) [50, Ingber, 1989] rejects nonfeasible points. Moreover, quite often the simulated annealing is modified to permit adaptive changes in the ranges of the parameters to take into account new information that might make it efficient to cut down the size of the search space.

3.6.4 A structured evolutionary algorithm

For EA, like GA, there is an underlying assumption that a "fitness" landscape can be characterized in terms of variables, and that there is an optimum solution in terms of those variables. For example, if one were trying to find the shortest path in a Traveling Salesman Problem, each solution would be a path. The length of the path could be expressed as a number, which would serve as the solution's "fitness". The "fitness landscape" for this problem could be characterized as a hypersurface proportional to the path lengths in a space of possible paths. The goal would be to find the globally shortest path in that space.

The EA algorithm can be summarized as below. $P(t)$ is the generated population at generation t. $P(t+1)$ is the selected subpopulation. The pseudo code of EA can

be illustrated as below.

```
procedure Algorithm EA
t = 0;
/* initialize a usually random population of individuals */
initialize population P(t);

/* evaluate fitness of all initial individuals of population */
evaluate P(t);

/* test for termination criterion (time, fitness, etc.) */
while (not termination-condition)
/* increase the time counter */
   t = t + 1;

   /* select a sub-population for offspring production */
   P(t) = selectparents P(t - 1);

   /* perturb the mated population stochastically */
   mutate P(t);

   /* evaluate its new fitness */
   evaluate P(t);
end;
```

Chapter 4

A Population-Based Search Algorithm through Thermodynamic Operation – GASA Algorithm Design

4.1 An Overview

A general search algorithm of unified evolutionary computation and simulated annealing is proposed in this chapter. The combined algorithm from evolutionary computation methods is a classical genetic algorithm. This created algorithm is called **GASA**. Depending on the setting of its control parameters, GASA executes as a genetic algorithm, a simulated annealing algorithm, or a mixture of these. In our preliminary investigations, we have developed a method that merges the approaches of genetic algorithms and simulated annealing (GASA). This combined algorithm represents an application's independent approach to optimization, and the resulting search process is highly adaptive. In GASA, objective function parameters are coded into a "chromosome" and a population of individuals is maintained and updated over a period of generations. When genetic algorithms are used alone, usually the most fit individuals are preserved for the next generation and the least fit individuals undergo crossover

and mutation.

One of the main problems of the genetic algorithms (GAs) is its convergence behavior [83, Rudolph, 1994]. Initially, the cost values of the population improves quickly. But then it becomes very difficult to obtain further improvement. Most of the run time is spent in the later phase of the process in which only small improvements are obtained very slowly. Simulated annealing is a high-precision optimization technique, but with lower running speed. In the meantime, the evolutionary computation provides a population-based method to speed up the annealing algorithm.

The annealing technique was invented by Kirkpatrick in 1983 [55, Kirkpatrick, 1983]. It is essentially a modified version of hill-climbing. Starting from a random point in the search space, a random move is made. If the move takes us to a higher point, it is accepted. If it takes us to a lower point, it is accepted only with the probability calculated at that time. The probability begins close to 1, but gradually reduces towards zero — the analog being with the cooling of a solid. A description of the cooling of molten metals motivates this algorithm. After slow cooling (annealing), the metal arrives at a low energy state. In such circumstances, thermal equilibrium at a given temperature is characterized by a Boltzmann distribution function of the energy states. Under these conditions, even at low temperature, a transition may occur from a low to high energy level, albeit with a small probability. Such transitions are assumed to be responsible for the system reaching a minimum energy state instead of being trapped in a local meta-stable state.

Like other random search techniques, simulated annealing only deals with one candidate at a time, and so does not build up an overall picture of the search space. No information is saved from previous moves to guide the selection of new moves. To overcome this limitation, we use the population-based guided search in combination

with the annealing technique to create a new search algorithm. Sirag and Weisser [92, 1987] use a similar approach with a "thermally" motivated adaptation of inversion, mutation and crossover on the TSP problem; however, our approach specifically creates subpopulations that are subject to mutation or crossover and are created by an annealing algorithm.

In the GASA technique, objective function parameters are coded into a "chromosome" and a population of individuals is maintained and updated over a period of generations. When genetic algorithms are used alone, usually the most fit individuals are preserved for the next generation but any individuals can be selected to undergo crossover and mutation. In the GASA technique, simulated annealing is used in choices regarding the subset of individuals to undergo crossover and mutation. At each generation, some individuals are accepted for a special gene pool that contains mostly the fittest individuals. A simulated annealing approach, with a probability calculation based on a Boltzmann distribution using an energy value parameter, is used to decide which individuals are accepted or rejected. Those set aside from the special gene pool are subjected to crossover. A similar procedure is then used to decide which individuals will undergo mutations.

The GASA approach is intuitively appealing because it allows the population to sustain the most fit individuals in most cases, but in a few cases, due to the probabilistic nature of the simulated annealing, highly fit individuals are crossed over or mutated. The less fit individuals are more often subjected to the perturbations of crossover and mutation, and these are the individuals who could benefit the most from such changes. Fewer iterations are required with GASA to arrive at the final solution at the global minimum of the objective function. Thus the combined technique appears promising for difficult problems with many local minima. Potential applications

include controller design and training [101, Varšek et al., 1993], molecular prediction [98, 104, 20, Unger & Moult, 1987; Wehrens et al., 1993; Dandekar & Argos, 1994], fuzzy reasoning and control [74, Park et al., 1994], and genetic algorithms to train neural networks [56, Kitano, 1990].

4.2 The Preliminary Unified Concepts

4.2.1 A mapping inspired from nature — biological and physical systems

We have developed a unified method (GASA) which uses genetic algorithms (GAs) and simulated annealing (SA), described in the section, as the basis for optimization. The moving of points at the search plane is similar to that of particles in the crystals of materials. For example, a group of individuals of a varying environment when cooled to a freezing temperature will tend to assume relative change vs. the position in their domain in such a way as to minimize the potential energies of the systems. Because of a large number of individuals and possible arrangements, the final state will most likely correspond to only a local minimum energy instead of a global minimum energy. That is, energy change makes the structural changes of the participating individuals. Their state should be totally different from their beginning positions even in the various generations.

This method extracts the characteristics of SA to make the participating individuals reach the global convergence. It is same as moving particles in the annealing process whose movement is shown in Fig. 4.1. Many individuals are initialized in GAs and then are treated with our annealing process. The annealing schedule is defined by the

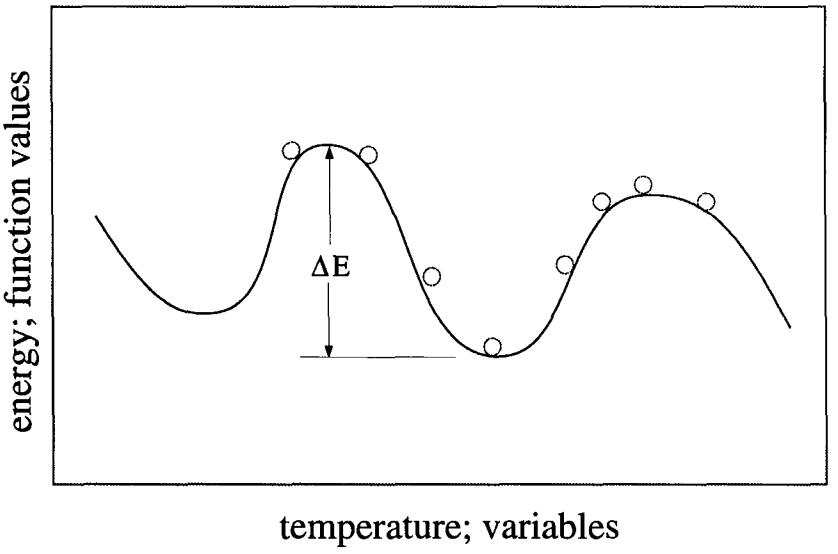


Figure 4.1: *The behavior of the participating individuals*

user. The participating individuals will change their structures if their energy goes beyond their genetic threshold energy. Once their structures change, the individuals in next generation will appear at another position and not at their parent's locations as shown in Fig. 4.1.

4.2.2 The premise of the combination of GAs and SA

The concept of constructing the pathway between genetic algorithms and simulated annealing is based on **Energy**. This "energy" concept is only an imagination instead of existence in reality. Many search problems can be regarded as linear or nonlinear programming problems. They all have one or many objective (cost) functions. The energy can be defined as the objective value from the applied problems.

$$objective\ \ function\ \ f(\vec{X}),\ \vec{X} = \{x_1, x_2, \cdots, x_q\} \in \mathcal{R}^q \tag{4.1}$$

Table 4.1: *The basic units in different algorithms*

	Phenotype	Genotype
GAs	chromosome	gene
SA	macro particle	atom
EA	individual	none
GASA	individual (chromosome)	gene

The participating basic unit is **an individual** $\in \vec{X}$ which is not limited to the so-called **particle** in simulated annealing or **chromosome** in genetic algorithms.

Basic unit The most important concept behind to be presented is to look upon **individuals (chromosomes)** and particles into one basic unit: individuals. They are the real "data points" in practical operation. Chromosomes are comprised of genes, and particles are comprised of atoms in general, too. The genes are the values (0 or 1) in the binary bits. We suggest **pseudogenes** to be the basic unit of the phenomenon, here we use the term "pseudogenes" to cover the Holland's canonical GAs work on bit string of fixed length. These pseudogenes will be processed by Darwin's evolutionary selection scheme, i.e. only individuals with high fitness can survive. Their structures are changed for the sake of existence in the varying environments.

Environment The environments here are provided by the different functions or models. They become constraints to the practical supplied problems.

The comparison of the basic units in these algorithms is described in Table 4.1. The unit in genetic operation for GASA is genes which are located at chromosomes.

Genotype and Phenotype In nature, the physical expression of the genotype

is phenotype. For example, a person has a specific genetic structure that perhaps someday can be mapped in a graphic display. That apparent structure is the person's **genotype** which is the explicit genetic structure of the person's chromosomes. The actual person is characteristics (tall, black hair, etc.) is the **phenotype**. Likewise, the atom is the genotype of the particle. The combined action of the atoms of the particle can be described as the phenotype of the particles. This is able to explain the mapping between biological and physical systems. In this task, the Boolean binary number is applied, a chromosome may have the genotype "000001011" which decodes to a parameter value (phenotype) of 11.

4.3 A Unified Approach

4.3.1 GASA algorithm design

Before we describe how to design Genetic Algorithms + Simulated Annealing (GASA), let's formulate a general statement of searching optimum problem. Consider a general function minimization problem, with function $f : \vec{X} \rightarrow \mathcal{R}^{\mathcal{D}}$, $\vec{X}$ is the set of all variables,

$$\begin{aligned} J &= \min\ f(\vec{X}) \\ \vec{X} &= \{x_i, x_2, \cdots, x_D\} \\ x_i &\in [u_i, v_i] \end{aligned} \tag{4.2}$$

where u_i and v_i are the lower and upper bounds, respectively, and D is the number of the variables x_i in the defined domain of the function f.

When we apply the generation mechanism of GAs, the x_i is replaced by $x_i(t)$, where t is the number of generations. Our goal for this problem is to search for the global minimum f^*, where x_i^* is the minimum location.

1. Temperature schedule

   Simulated annealing offers a strategy very similar to iterative improvement with one major difference: annealing allows perturbations to move in a controlled fashion. We now refer to individual perturbations as **moves**. Each move can transform one configuration into an either **better** or **worse** configuration, so it is possible to jump out of a local minima and potentially fall into a more promising path. The moves are controlled by temperature. Thus, the basic requirement for simulating this process is the ability to simulate how the system reaches thermodynamic schedule at each temperature. However, for the temperature schedule, the initial temperature should be set large enough to be successful in the annealing process. We fix the annealing schedule as $T_{m+1} = \alpha \cdot T_m$, where α is a temperature reduction parameter usually assigned to a value between 0.99 and 0.85, and m is the number of moves of individuals attempted to reach equilibrium.

2. Population generating

   First, we have to create an initial population of individuals, where each individual is a binary vector of l bits. The initial population with D individuals is noted as $\vec{P}(0) = \{\vec{\nu}_i(0) | i = 1, 2, \cdots, D\}$. All of the l bits for each individual are initialized. The creating mode is broadly applied in the following operation. The vector, $\vec{\nu}(0)$ is the initial encoded Boolean binary population vector which can be decoded into genes x_i^t. The initial population is prescribed at $t = 0$. It

can be seen as an encoded population matrix $\vec{P}(t)$ at generation t and can be formulated as

$$\vec{P}(t) = \{\vec{\nu}_i(t)\} \triangleq \left(\begin{array}{c|c|c|c} \overbrace{\nu_{111}\nu_{112}\cdots\nu_{11l}}^{x_{11}} & \overbrace{\cdots}^{x_{12}} & \cdots & \overbrace{\nu_{1D1}\cdots\nu_{1Dl}}^{x_{1D}} \\ \overbrace{\nu_{211}\nu_{212}\cdots\nu_{21l}}^{x_{21}} & \overbrace{\cdots}^{x_{22}} & \cdots & \overbrace{\nu_{2D1}\cdots\nu_{2Dl}}^{x_{2D}} \\ \multicolumn{4}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ \overbrace{\nu_{n11}\nu_{n12}\cdots\nu_{n1l}}^{x_{n1}} & \overbrace{\cdots}^{x_{n2}} & \cdots & \overbrace{\nu_{nD1}\cdots\nu_{nDl}}^{x_{nD}} \end{array} \right)_t \tag{4.3}$$

where $\{n, D, l\}$ are the total number of individuals, dimension (the number of chromosomes per individual), and bits (the number of genes per chromosome), respectively. The vector set, $\{\vec{\nu}_i(t)\}$, is the set of all binary values of the parameters of individual i in a population. For convenience, we can define a binary matrix $\underline{\underline{\nu}}$ as

$$\underline{\underline{\boldsymbol{\nu}}}(\boldsymbol{t}) \stackrel{\text{def}}{=} \{\nu_{ijk}(t) | i = 1, \ldots, n, j = 1, \ldots, D, k = 1, \cdots, l\}$$

Its decoded real-value population matrix $\underline{\underline{X}}(t)$ is obtained by applying the decoding operator Γ

$$\Gamma\{\vec{\nu}(t)\} = \underline{\underline{X}}(t) \triangleq \left(\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \multicolumn{4}{c}{\cdots\cdots\cdots\cdots\cdots} \\ x_{n1} & x_{n2} & \cdots & x_{nD} \end{array} \right)_t \tag{4.4}$$

where n is the total number of individuals and Γ is a decoding function. Each line notates the parameters of an individual.

3. Function evaluation

   As noted in GAs, the functional evaluation plays an important role, rating potential candidate solutions in terms of their objective function value, which

ultimately determines fitness. The individuals need to be decoded because of the binary bits. The function evaluation mode can be defined as

$$eval(\underline{\underline{X}}(t)) \stackrel{\text{def}}{=} f(\{\vec{X}_i(t)\}) \triangleq \begin{pmatrix} f(\vec{X}_1(t)) \\ f(\vec{X}_2(t)) \\ \vdots \\ f(\vec{X}_n(t)) \end{pmatrix} \tag{4.5}$$

where n is the number of the individuals in the population at generation t. The vector set, $\{\vec{X}_i(t)\}$, is the real-value of individual i at generation t. Then it still requires the scaling of fitness from the evaluated function values.

4. Fitness scaling

   The scaling function δ calculates the fitness from the objective function and it scales the fitness. The function δ is able to assure the best individual receives the largest fitness. Normally, fitness evaluation transforms the cost function value f into a measure of relative fitness. Most commonly, a **linear dynamic scaling** is used which takes into account the worst individual of the population $\vec{P}(t-\omega)$ over ω time steps before (if $t-\omega<0$ then replace $\vec{P}(t-\omega)$ with $\vec{P}(0)$). If ω is a scaling window, the scaling function δ is defined by

$$\delta(f(\Gamma(\underline{\nu}(t))),\omega) = a \cdot f(\Gamma(\underline{\nu}(t))) + b(\omega) \tag{4.6}$$

   where t is the current generation, a is a scaling factor ($a=-1$ usually). The $b(w)$ is a baseline reflecting the worst fitness value which occurred within the last ω generations. Each individual's fitness is subtracted from $b(\omega)$. With linear dynamic scaling, negative fitnesses do not occur and thus do not cause any problem for the calculation of selection probabilities according to Equation (1).

5. Selection

   The selection procedure probabilistically selects an individual i to remain in the population and reproduce with probability

$$p_i = \frac{\Phi_i}{\sum_{j=1}^{n} \Phi_j} \tag{4.7}$$

   The fitness Φ_i is calculated from the objective function and defined as below.

$$\Phi(\underline{\nu}) \stackrel{\text{def}}{=} \delta(f(\Gamma(\underline{\nu})), \omega) \tag{4.8}$$

   where Φ is a fitness conversion operator. When the individuals are selected, the high-fitness individuals will be reproduced according to their fitness function Φ.

6. Genetic Alterations through Thermodynamic Operation

   Before the selected individuals process crossover, the modified Metropolis criterion is applied to decide which individuals are to be put into a subpopulation that does crossover. This decision is made according to an acceptance probability. First, choose the minimum point of the individuals as the comparison center. The minimum objective function value and the optimum location (selection center) are given by $\{f^*, \vec{X}^*\}$ for the current population.

   It is possible to define each individual's energy function from the objective function value. Thermodynamics indicates that thermal equilibrium at temperature T is a probability distribution in which a state with **energy function** $\{E_i(t) = f(\vec{X}_i(t)) | i = 1, \cdots, n\}$, whose n is the total number of individuals in generation t. The corresponding **threshold probability** p_i is

$$\frac{e^{-\frac{E_i}{kT}}}{Z(T)} \tag{4.9}$$

   where $Z(T)$ is a partition factor.

From the above definition, the probability p between $\vec{X}_i$ and $\vec{X}_j$ is

$$\begin{aligned} \frac{p_i}{p_j} &= \frac{e^{-\frac{E_i}{kT}}}{e^{-\frac{E_j}{kT}}} \\ &= e^{-\frac{\triangle E}{kT}} \end{aligned} \tag{4.10}$$

where $\triangle E = E_i - E_j$. The k is 1 in this simulation.

Modified Metropolis Criterion (MMC)

A modified Metropolis criterion is used here. The method applied here is to replace E_j with the population's minimum energy E^*, and to have $\vec{X}_j$ replaced by $\vec{X}^*$. This modification results in $\triangle E$ being greater than or equal to 0. Thus, in classic simulated annealing, sometimes the uphill climbing is retained and the move **may** still occur. Therefore, a new, higher energy state will be accepted by the individual, according to a threshold probability. In GASA, an entire population of individuals is divided into two classes, say Ξ_c & Ξ_{nc}. Usually, the motivation of division is from the individual's energy (expressed with its threshold probability). For example, the crossover operation increases the diversities in the low-energy class Ξ_c. Occasionally because of the SA aspect, the higher-energy individual may move uphill or downhill according to their crossover mating. When crossover is done, the resulting individuals may end up with a higher energy level, that allows them to climb over a hill and into another basin. Since the new basin may have a lower minimum, there can be beneficial to let the individuals reach the valley.

For each individual i, p_i is calculated. If $p_i > random[0, 1)$, then the individual is accepted and put in a subpopulation pool Ξ_c. Otherwise the individual is not accepted into Ξ_c and is put into subpopulation Ξ_{nc}. We process the set of unaccepted individuals by the crossover operator $C_{\{p_c\}}$, where p_c is the applied

crossover rate, on population Ξ_{nc}. The mates of individuals are crossed-over by selecting a cross-point randomly. The crossover cannot always guarantee the offspring's fitness would be higher than that of their parents, but it can increase diversity and sometimes makes recombinations that have superior fitness. The typical feature of this algorithm is that, besides accepting individuals with high fitness in population Ξ (after selection), it to a large extent puts individuals with high fitness into population Ξ_c and puts individuals with low fitness into population Ξ_{nc}, according to the temperature. Initially, at large values of T, exceptions are more likely; as T decreases only smaller exceptions will be made to the general rule that high fitness individuals are preserved in population Ξ_c.

The mutation operation will process with similar action as that in crossover. It is processed by the mutation operator $M_{\{p_m\}}$, where p_m is the applied mutation rate. For each individual, we calculate q_i. If the individual passes the acceptance criterion, it will be kept in a gene pool Ξ_m. The remaining individuals are put into subpopulation Ξ_{nm}, then are mutated at a randomly selected bit, and kept. Thus, mutation is usually applied to the inferior genes of the running chromosomes.

By successively lowering the temperature and running this algorithm, we can simulate the individuals coming into equilibrium at each newly reduced temperature, and thus effectively find the solution candidates according to the energy transitions.

In this technique, **equilibrium** is only a conceptual criterion instead of a real physical phenomenon. It can be described as **while the generation process is repeated, a sequence of temperatory candidate solutions is produced until the solution space occupancy is described by the Eqs. 4.9 & 4.10**.

7. Insert the individuals in next generation

   The individuals kept in the gene pools are combined into a single pool and beco me the parents of the next generation $t+1$. The initial temperature T_0 is chosen together with the procedures of thermal equilibrium. Then T is updated and held constant, and the parameters that simulate the thermal dynamics are set. The choice of these parameters is referred to as a **cooling schedule** as recommended in step 1.

The moving of the participating individuals vs. temperature is shown in Fig. 4.4. The balls are the individuals climbing upwards and downwards according to the temperature change. Fig. 4.1 also reveals that the individuals' function values change with the variables.

This GASA algorithm can be described as follows:

Parameters declaration:
$t = 0$; (the initial generation);
t_{max} is the allowed maximal number of generations;
$T_0 =$ initial temperature;
$Tm =$ number of temperature moves to attempt;
C is the crossover operator;
M is the mutation operator;
Ξ is the total gene pool per generation;
Initial $\vec{P}(0) = \{\underline{\underline{\nu}}(0)\} \in U$ where $U = \{0,1\}^l$;
Evaluate $\vec{P}(0) = \{\Phi(\nu_1(0)), \cdots, \Phi(\nu_l(0))\}$;
where $\Phi(\nu_k(0)) = \delta(f(\Gamma(\nu_k(0))), P(0))$;
For m =1,Tm
while ($t < t_{\max}$)
For each individual i,

Evaluate the change of energy $(E_i - E^*)$ and find a probability p_i;

if threshold probability p_i > random [0, 1),

keep the accepted individual in a gene pool Ξ_c;

else

put i into Ξ_{nc};

Crossover: $\nu'_k(t) = C_{\{p_c\}}(\vec{P}_c(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_c(t) \in \Xi_{nc}(= \Xi - \Xi_c)$, p_c is the crossover rate;

replace Ξ with $\Xi_c + \Xi_{nc}$;

For each individual i,

Evaluate the change of energy $(E_i - E^*)$ of the individuals from Ξ and $\nu'_k(t)$,

and find a probability p_i;

if accepted probability p_i > random [0, 1),

keep the accepted individual in a gene pool Ξ_m;

else

put i into Ξ_{nm};

Mutate: $\nu''_k(t) = M_{\{p_m\}}(\vec{P}_m(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_m(t) \in \Xi_{nm}(= \Xi - \Xi_m)$, p_m is mutation rate;

replace Ξ with $\Xi_m + \Xi_{nm}$;

Until equilibrium;

$T = update(T)$;

Evaluate $\vec{P}(t) = \{\Phi(\nu_1(t), \cdots, \Phi(\nu_l(t))\}$,

where $\Phi(\nu_k(t) = \delta(f(\Gamma(\nu_k(t))), P(t - \omega))$;

Select $\vec{P}(t + 1) = sel(P(t))$;

Reproduce $\vec{P}(t + 1)$;

}; end t

$t = t + 1$;

}; end move

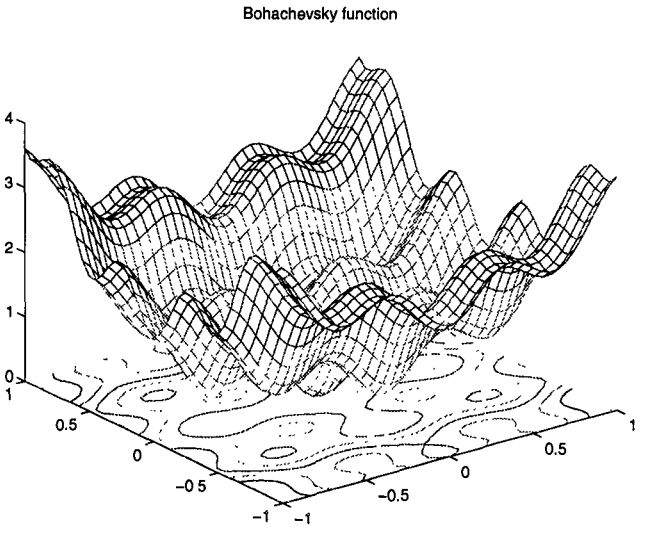


Figure 4.2: *A multiple-optima function*

4.4 Results on Function with Multiple Optima

To compare the GASA approach with GAs, we performed an experiment on a function with multiple optima. The function chosen is shown in Fig. 4.2, and is generated by the following equation.

$$f(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7 \qquad (4.11)$$

This function was examined by Bohachevsky et al. (1986)[13, Bohachevsky, 1986]. It has numerous local minima and a global minimum at the origin. This function is illustrated in Fig. 4.2 with $x_1, x_2 \in [-1, 1]$. There are many local minima indicated in the contour plot as in Fig. 4.3. "*" shows the location of global minimum. When (x_1, x_2) is far from the origin, the quadratic terms of $f(x_1, x_2)$ dominate the cosine terms and the overall shape of f is quadratic. If (x_1, x_2) is close to the origin, the cosine functions dominate the quadratic terms and $f(x_1, x_2)$ displays many hills and valleys. It is very difficult to discover the global minimum for the function using a

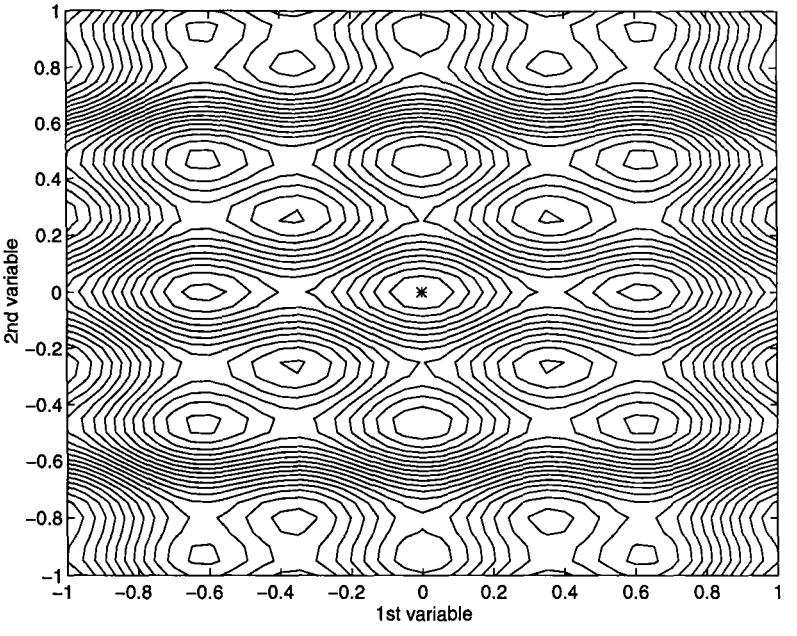


Figure 4.3: *The contour plot of Bohachevsky function*

gradient technique.

We have applied the GASA technique to find the global optimum of the Bohachevsky function. First, we initialize the parent population randomly, and then encode the values x_1 and x_2. Secondly, using the decoded variables, we select the most fit individuals from their function values. Then, we apply the MMC (Modified Metropolis Criterion) to choose the individuals either crossed-over/mutated or kept in a gene pool. Thus, very probable moves can be rejected, and very improbable moves can be accepted – at least occasionally. However, while lowering the temperature successively, we can simulate the individuals until thermodynamic equilibrium is achieved in each generation.

Fig. 4.4 is the annealing schedule in this experiment. The starting temperature is 4.5 and, over 28 iterations, the temperature relaxed to zero. This number of generations was sufficient to find the global minimum for this function.

Fig. 4.5(a) shows the locations of the individuals in the 28th generation. There are two variables on each chromosomes, plotted on the horizontal and vertical axes. The objective values of all individuals in generation 28 can be found in Fig. 4.5(b).

The best and mean score of objective values are shown in Fig. 4.5(c) as functions of the generations. The best and mean objective values are reaching the global minimum at generation 8 and 12, respectively. The best objective values begins just below 0.3 and the mean objective value begins at about 1.4. Then there is initially a wide spread in the objective values for different individuals. After 15 generations have gone by, the mean objective value is under 0.0058, close to the global minimum of zero.

Fig. 4.5(d) shows the value of the first and second variables over the 28 generations. These values are taken from the individual that ultimately has the best objective function. After generation 25, both variables have reached approximately 0, the location for the global minimum.

Fig. 4.6 shows the three results from GASA, for comparison. In Fig. 4.6(a), the fitness of the best individual is plotted over a period of generations. The highest fitness value 2 corresponds to the objective function minimum at 0. Fig. 4.6(b) shows the standard deviation of the objective values over the entire population as a function of the generation. The min & mean of the objective values as shown as a number of function calls for this algorithm in Fig. 4.6(c).

Fig. 4.7 and Fig. 4.8 are generated with GAs by applying the same crossover and mutation rate and same initial population size as those used in GASA. The same plots appear in Fig. 4.7 & Fig. 4.8 for GAs as in Fig. 4.5 & Fig. 4.6 for GASA. GAs become convergent after the 69th generation (see Fig. 4.7(c) & (d)). The fittest individuals can also be found at different generations from Fig. 4.8(a). Moreover, Fig. 4.8(b) indicates GAs's standard deviation is an irregular oscillation. GAs use more function

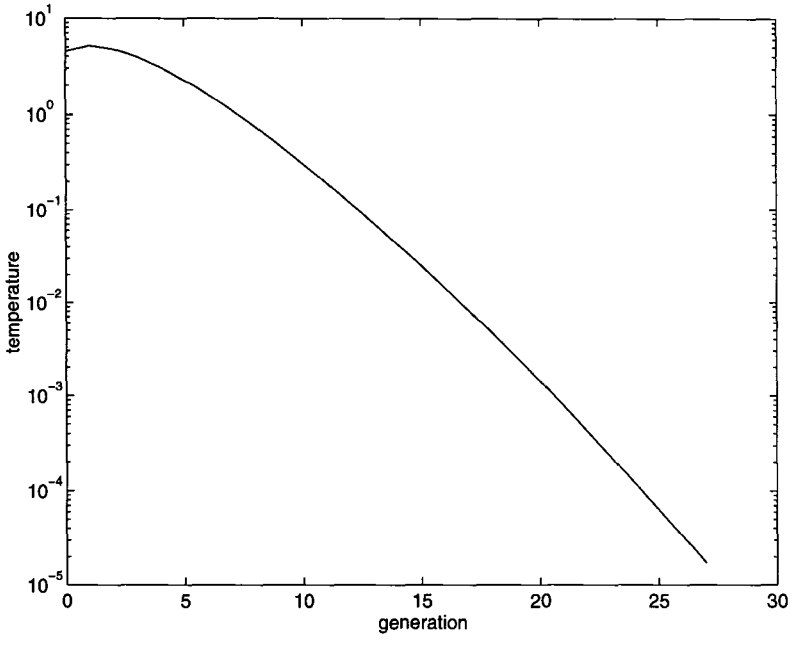


Figure 4.4: *Annealing schedule*

calls than GASA (Fig. 4.8(c)). Thus GAs need more generations to reach convergence compared with GASA.

The comparisons of the performance of GASA and GAs with the same parameters can be found in Figs. 4.9~4.12. The objective value of the mean individual in GASA decreases faster than in GAs, as shown in Fig. 4.9. Fig. 4.10 shows that GASA has a worse min objective value in the beginning, but sooner GASA reaches a faster convergence compared with GAs. These results illustrate that GASA can achieve better global convergence than GAs. Fig. 4.11 shows the standard deviation of the population versus the generation, and Fig. 4.12 shows the variance versus generation. These figures reflect decrease in population variance for GASA as the GASA method converges.

This experiment was run on SUN Sparc 10 workstation. The computation time

Table 4.2: *Bochachevsky Function: Statistic Analysis of the Number of Cutoff Generations to Achieve a Degree of Global Optimization with GAs*

	GAs			GASA		
	generation		obj. value	generation		obj. value
cutoff	mean	st . dev.	st. dev.	mean	st. dev.	st. dev.
$\leq 10^{-1}$	3.45	4.55	0.5144	3.80	5.20	0.3130
$\leq 10^{-2}$	11.05	13.95	0.4933	8.40	7.60	0.2802
$\leq 10^{-3}$	19.05	12.95	0.4251	14.60	5.40	0.1579
$\leq 10^{-4}$	29.70	19.30	0.4030	18.80	8.20	0.1176
$\leq 10^{-5}$	40.80	16.20	0.3867	24.10	2.90	0.0884
$\leq 10^{-6}$	49.30	16.70	0.3806	29.30	3.70	0.0722

of GASA and GAs are 32.90602 and 19.03456 CPU time, respectively. The time for GASA was longer because of annealing in each generation's population.

We have also compared twenty different runs of GASA and GAs , to show variations between runs. Twenty independent runs of GASA are able to be found in four rounds in Figs. 4.13$\sim$4.16. The other twenty runs in four rounds of GAs are shown in Figs. 4.17$\sim$4.20. From these graphs, it can be seen that it is much better for the GASA method to approach the global minimum.

In order to explore the performance of GASA and GAs, we list the compared resul ts of the cutoff generation for differing criteria in Table 4.2. The criterion applied depends on the precision requirement of the applications problems. In general, higher precision needs more generations, as can be seen in Table 4.2. Column 1 shows the cutoff value of the object function, Column 2 shows the average generation at which cutoff occurred, Column 3 shows the standard deviation of the generation at which

cutoff occurred, Column 4 shows the standard deviation of the objective value for the population, Column 2$\sim$4 apply to GAs, Column 5$\sim$7 show the same calculation for GASA. Twenty runs were asked for each tehnique. For example, at 10^{-4} cutoff, on average GASA and GAs spend 18.80 and 29.7 generations, respectively. At cutoff of 10^{-2} and below, GASA spends less generations than GAs to achieve the global minimum.

The standard deviations in the number of generations, shown in column 3 & 6 in Table 4.2, for GAs are higher than those for GASA. For example, when cutoff is 10^{-3}, for the objective value, GASA standard deviation is 0.1579 and GAs standard deviation is 0.4251. As to the average generation at cutoff 10^{-3}, GAs' and GASA's are 19.05 and 14.60, respectively. Thus, obviously, the search with GAs is in high oscillation.

Moreover, GASA keeps less deviation in all cutoff generations, which means the operation of GASA is more stable. GAs' operation shows high oscillation, as in Fig. 4.8(b), thus both the mean and standard deviation of GAs is larger than that of GASA. The fast decreasing of standard deviation shows how annealing is helpful to the genetic operations in GASA. The annealing is helpful to stabilize the GASA algorithm.

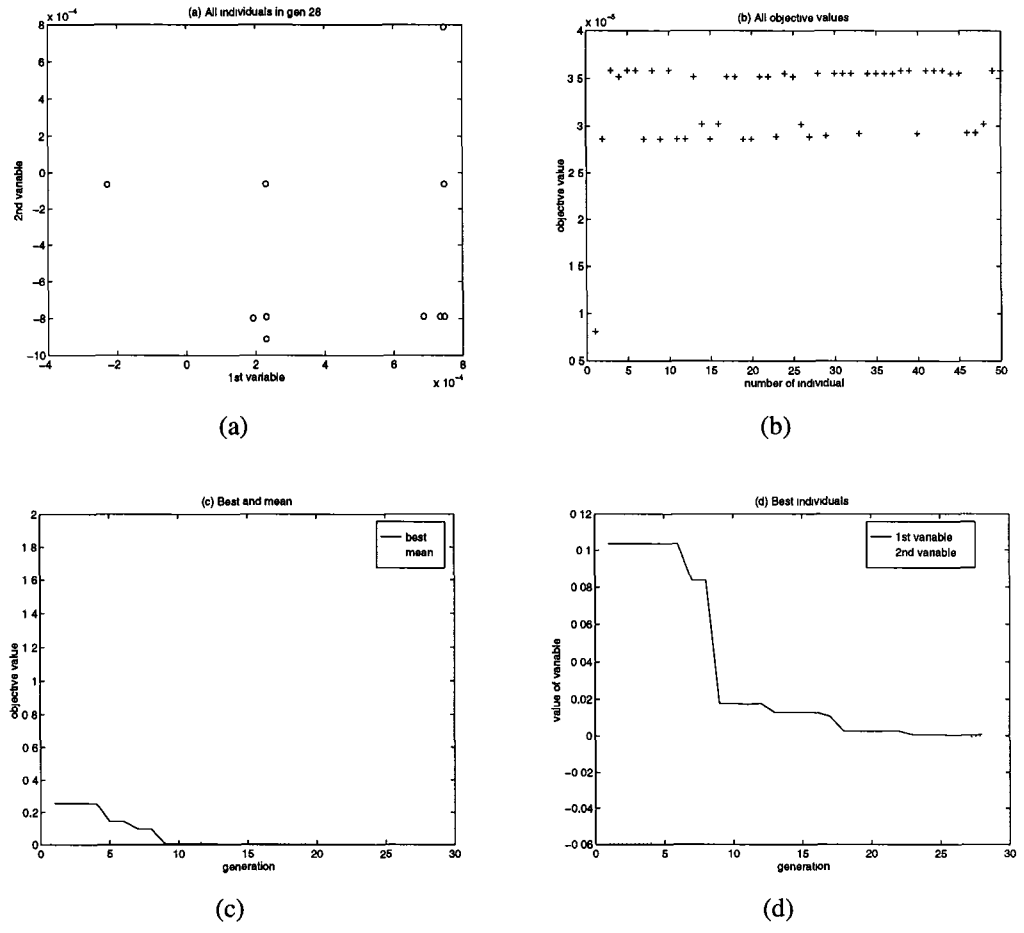


Figure 4.5: *An Interpretation of GASA operation in generations: (a) The locations of all individuals in generation 28. (b) The objective values of all individuals in generation 28. (c) Best and mean objective values of individuals in generations. (d) Best individuals in generations.*

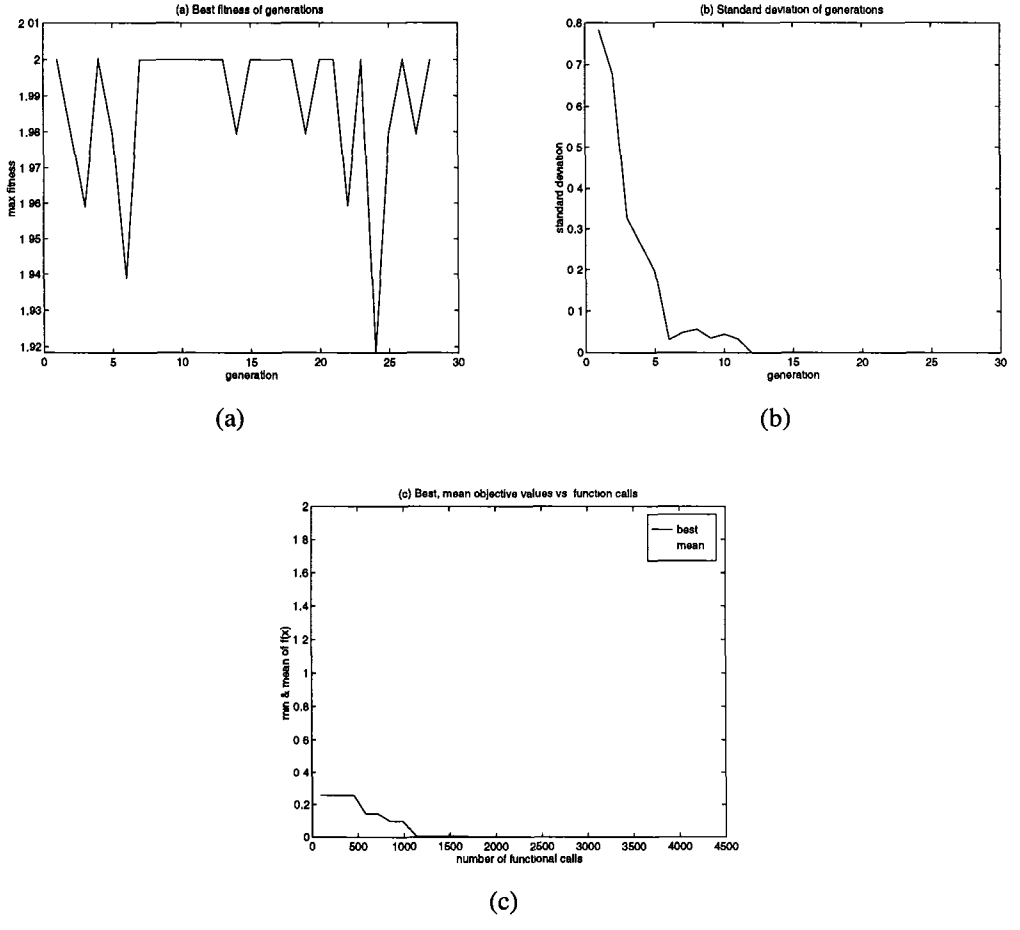


Figure 4.6: *Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.*

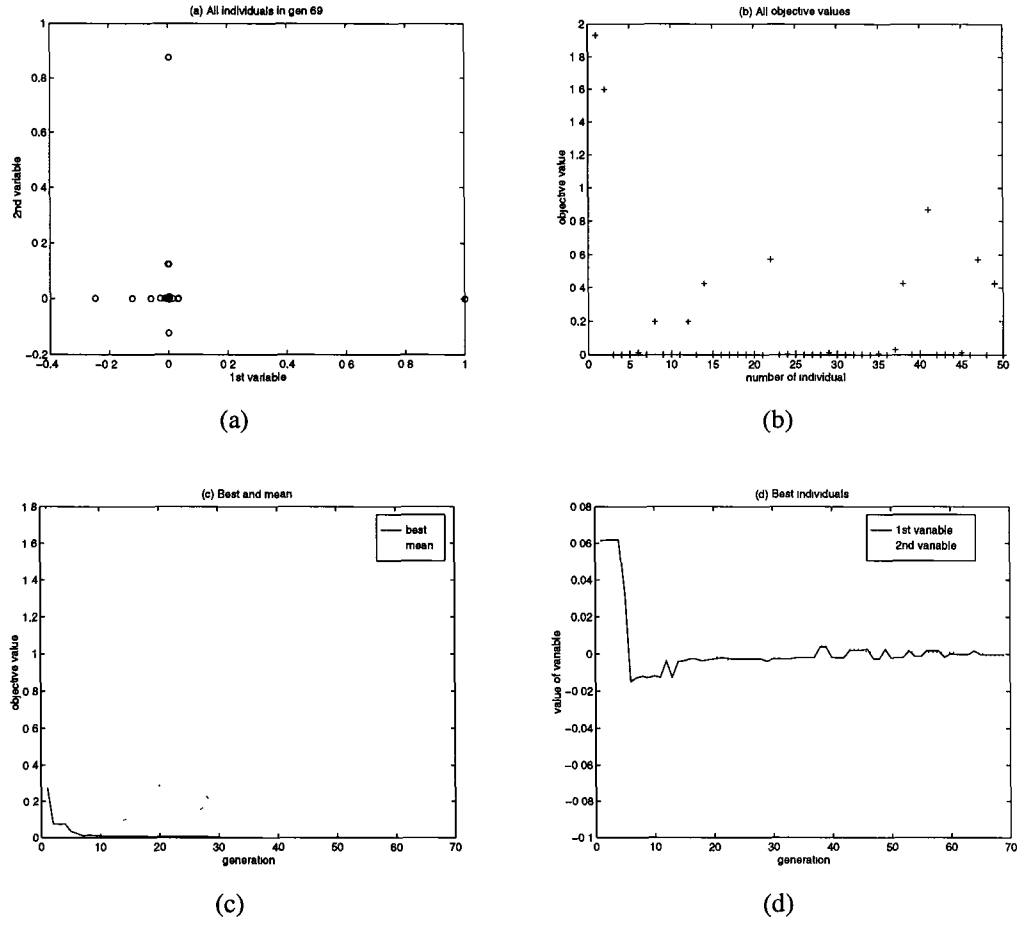


Figure 4.7: *An Interpretation of GAs operation in generations: (a) The locations of all individuals in generation 69. (b) The objective values of all individuals in generation 69. (c) Best and mean objective values of individuals in generations. (d) Best individuals in generations.*

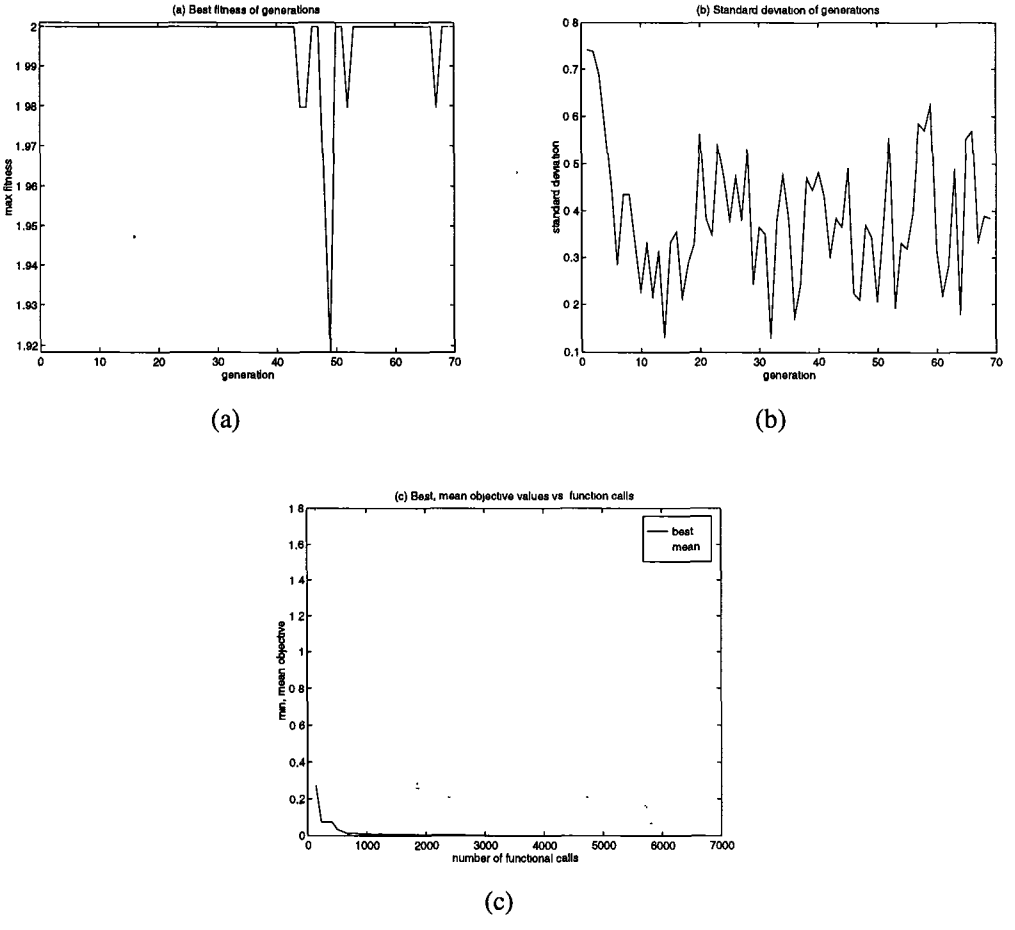


Figure 4.8: *Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.*

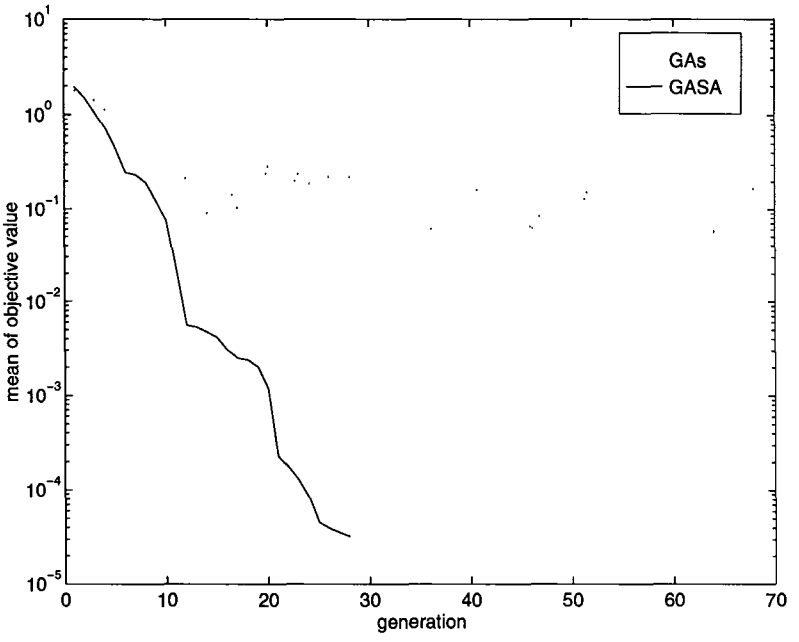


Figure 4.9: *Mean objective value in generations*

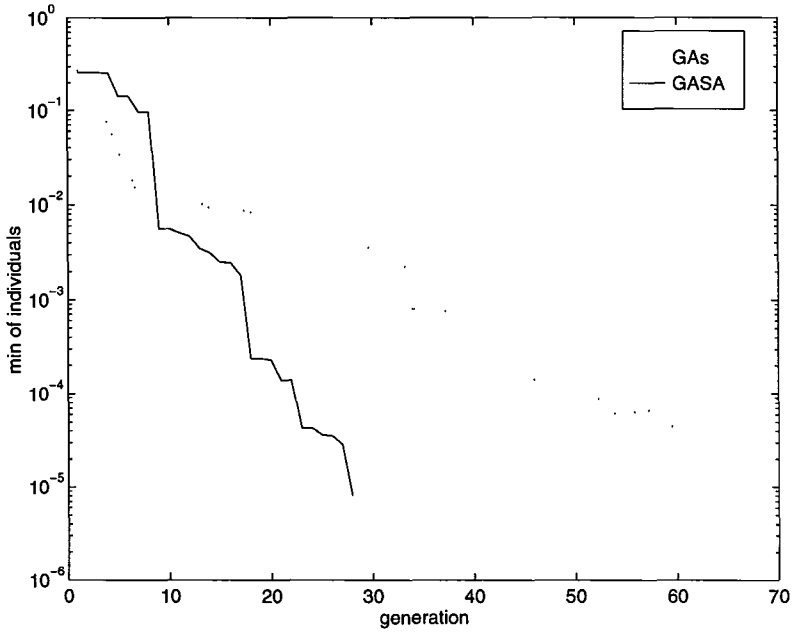


Figure 4.10: *Best objective value in generations*

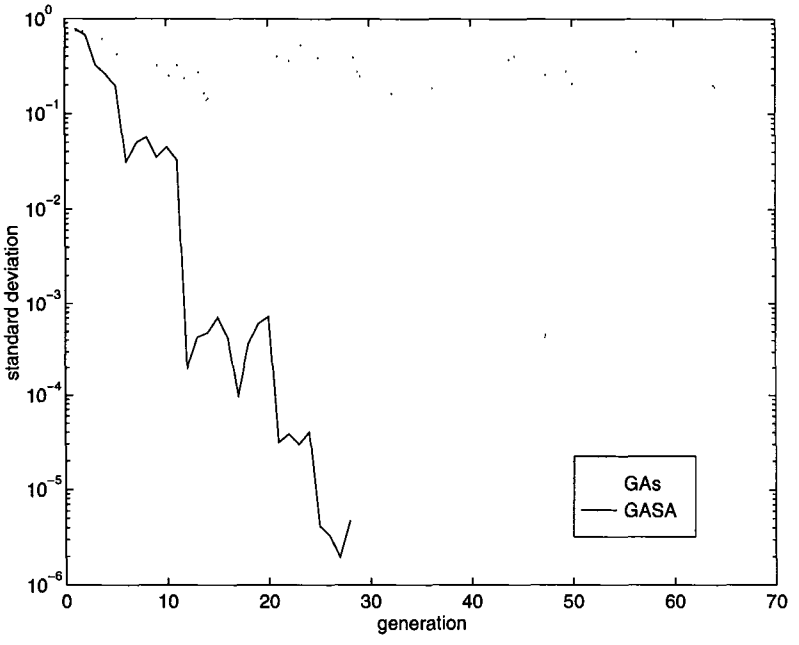


Figure 4.11: *Standard deviation of population vs. generation*

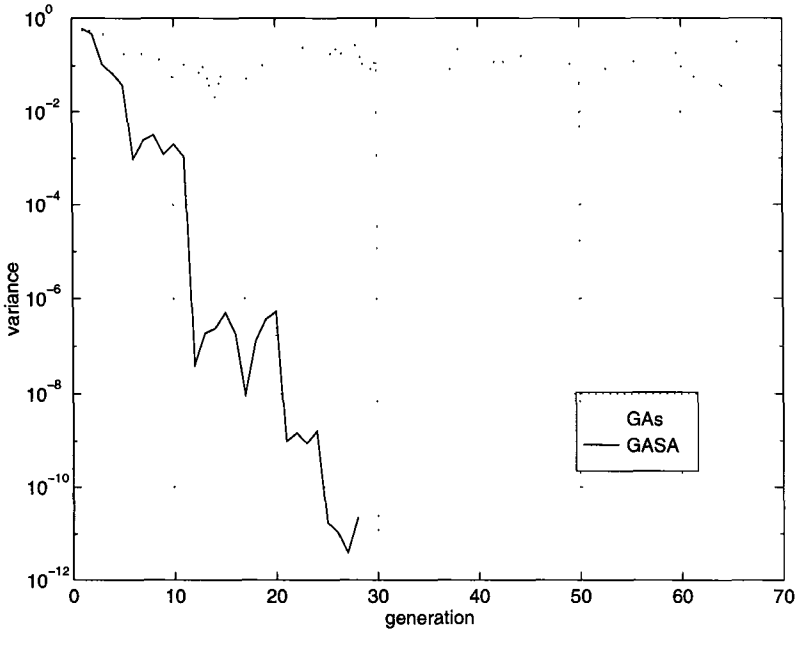


Figure 4.12: *Variance of population vs. generation*

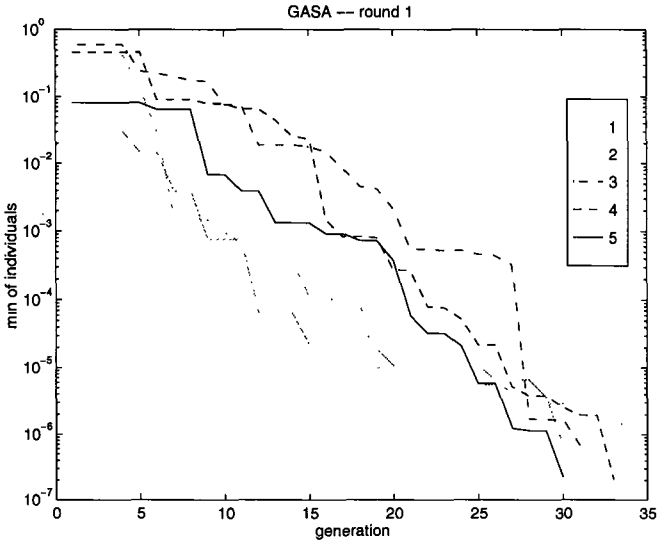


Figure 4.13: *The minimum of indviduals vs. generations in 1∼ 5 sets of GASA in round 1*

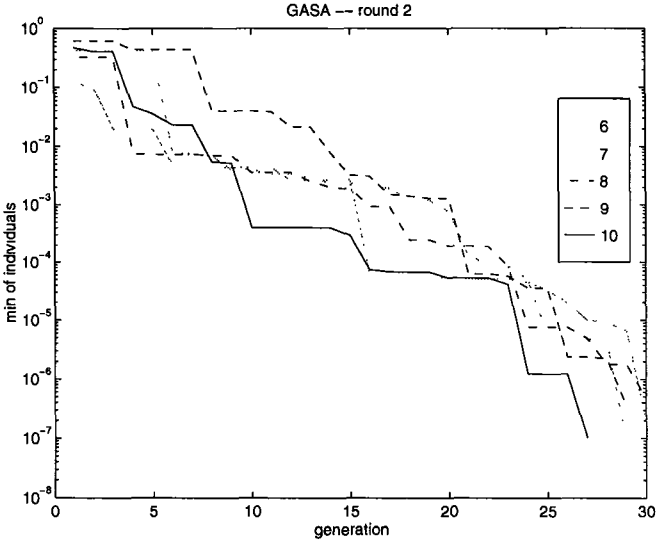


Figure 4.14: *The minimum of indviduals vs. generations in 6∼ 10 sets of GASA in round 2*

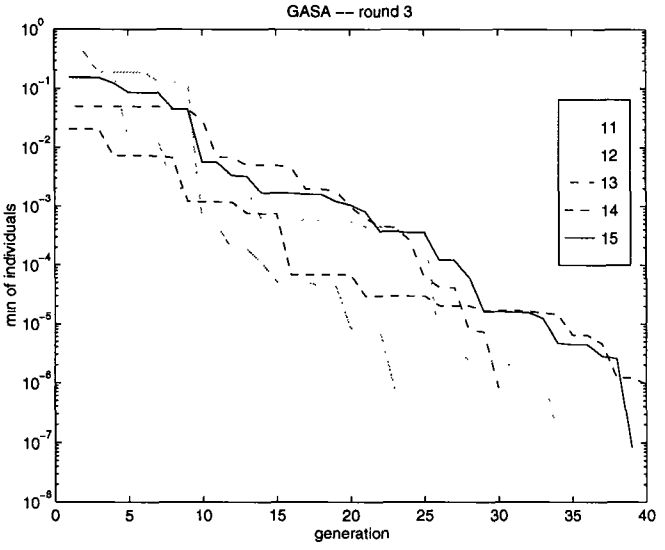


Figure 4.15: *The minimum of indviduals vs. generations in 11~15 sets of GASA in round 3*

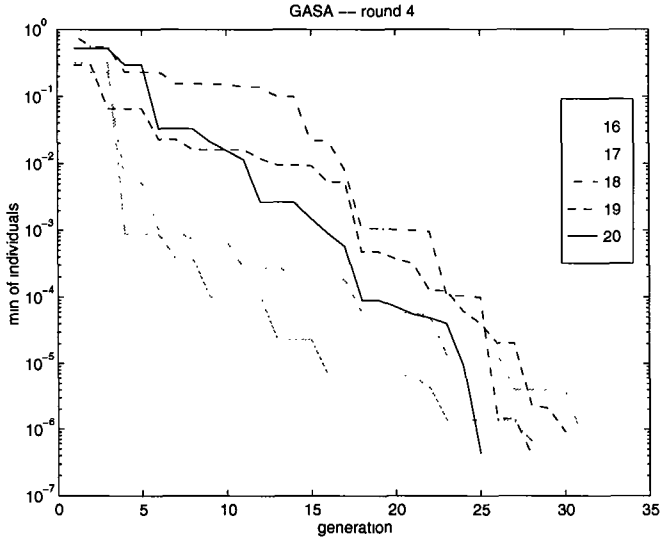


Figure 4.16: *The minimum of indviduals vs. generations in 16~20 sets of GASA in round 4*

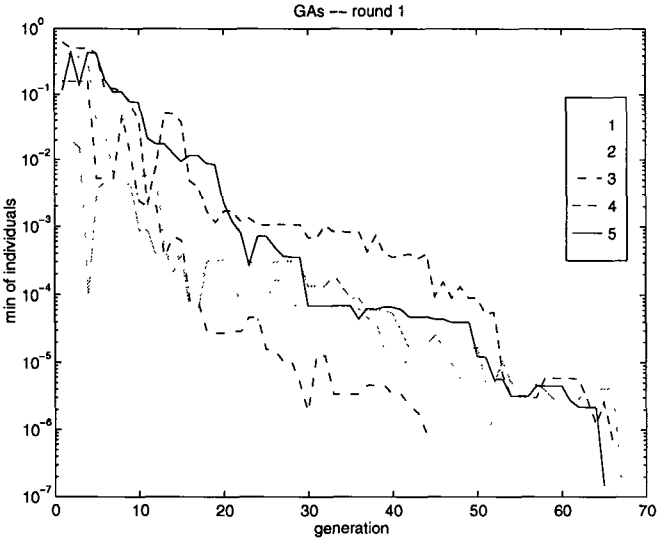


Figure 4.17: *The minimum of indviduals vs. generations in 1∼5 sets of GAs of round 1*

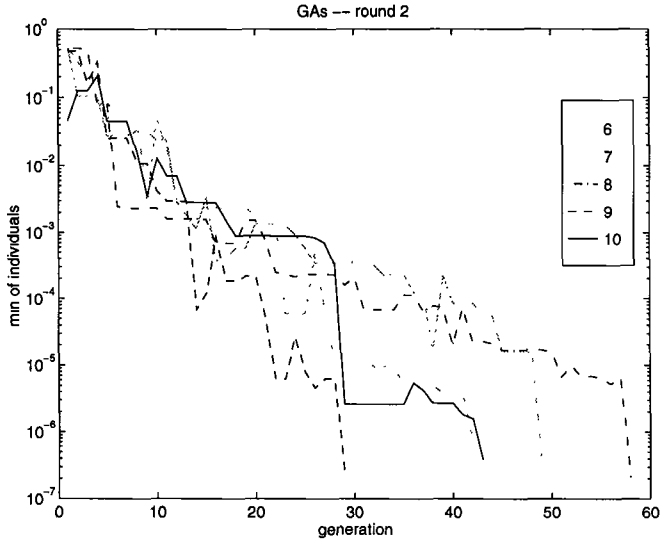


Figure 4.18: *The minimum of indviduals vs. generations in 6∼10 sets of GAs in round 2*

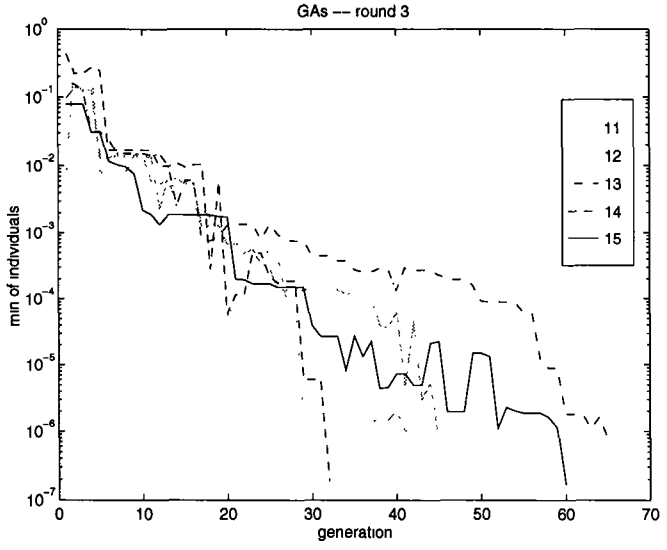


Figure 4.19: *The minimum of indviduals vs. generations in 11~1 5 sets of GAs of round 3*

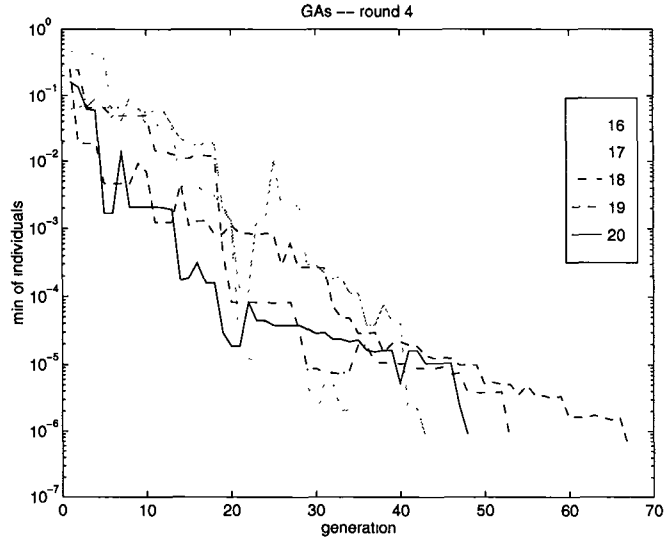


Figure 4.20: *The minimum of indviduals vs. generations in 16~2 0 sets of GAs in round 4*

Chapter 5

Temperature Effect and Search Mode

The annealing mechanism is a big concern for the simulated annealing algorithm. In general, the annealing mechanism includes the annealing (cooling) schedule or temperature scale which is used to decide the annealing process, and the initial temperature. Many researchers proposed their annealing schedule to define the temperature scale, but still no one can design a general scale for all problems [106, 66, 1, 67, White, 1984;Lundy & Mees, 1986;Aarts & Korst, 1989;Margarida et al., 1994].

The temperature scale is also related to the GASA algorithm, because temperature affects the division of the participating individuals. In thermodynamics, we use phase transition to explain the behavior of particles in different physical phases (e.g. gas, liquid, solid). Here **simulated phase transition** is proposed to explore the behavior of the search trajectory of the individuals. Along with theory from Holland's **schema theory** [46, Holland, 1975], the characteristics of GASA are explained with correlated theorems. Schema theory is used to explain the computational power of genetic operation on hyperplane (schemata) sampling [107, Whitley, 1993]. In population-based models, short schemata have higher probabilities of surviving crossover each generation. A new concept of population equilibrium is discussed from simulated behavior by combining the correlated genetic and thermodynamic phenomena for an

artificial environment.

Another extension of the annealing mechanism applied to GASA can make the partial population-based annealing algorithms on the genetic operators **crossover** or **mutation** alone. Thus, we can issue two other GASA algorithms: GASA_C (with annealing on crossover only) and GASA_M (with annealing on mutation only). The oscillations and diversity from the two alternative algorithms are discussed in this chapter.

In this chapter, Sec.5.1 illustrates the design of the annealing schedule including the evaluation of initial temperature and temperature scale. Sec.5.2 describes the effects of the partial annealing mechanisms and the two partial annealing GASA algorithms — GASA_C and GASA_M. Sec.5.3 overviews the behavior of the search phase and schema theory. Sec.5.4 is created and used to explain the transformation of individuals because of the temperature change. A new equilibrium rule is broadly applied to denote the equilibrium condition for the population-based search algorithms in Sec.5.5. Finally, the summary concludes the last section.

5.1 Annealing Schedule

A typical annealing algorithm starts at a high temperature. The temperature is reduced repeatedly, and the system is allowed to approach equilibrium. The procedure will stop whenever no further useful improvement can be reached, or when a cutoff value is reached. This protocol for cooling the system is called **the annealing schedule**.

Two frequently asked questions about the annealing schedule are discussed here. First, how high should the beginning temperature be for the system at the start ? Second, how low should the final temperature be, at which the system stops ? How

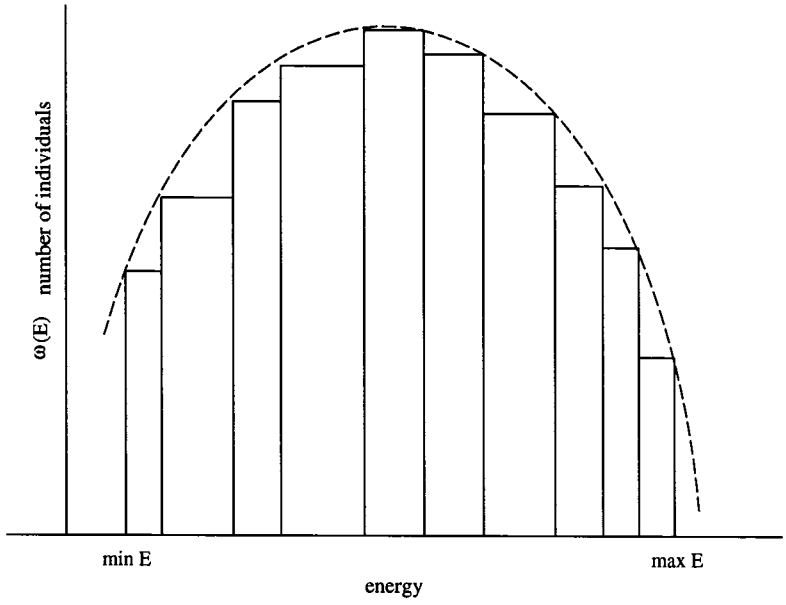


Figure 5.1: *The distribution of the state density function* ω

should the intermediate temperature be chosen ? These questions are still open to research. Because these solutions could be dependent on the applied problem and its dimension, and other factors, these questions arise in our new algorithm, GASA.

In order to discuss the temperature effect, let's define the **behavior**, **location** and **state** for each individual first.

Definition 1 *The* **behavior** *of an individual is how it moves in the multi-dimensional space according to how its variable* $\vec{X}$ *changes over time.*

Definition 2 *The* **location** *of an individual is described by its variable vector* $\vec{X}$ *in* R^n.

Definition 3 *A* **state** *of an individual (candidate solution) is the energy level which appears at the current arrangement of its genes.*

5.1.1 Initial temperature

The problem here is to minimize an objective function $f(\vec{X})$, which is described as follows.

$$E(\vec{X}(T_t)) = f(\vec{X}(T_t)) \tag{5.1}$$

where T_t is the temperature at generation t and $\vec{X}$ is the location of an individual, and $\vec{X}$ is a variable vector of the function f. E is the energy function at time (generation) t. From statistical analysis, the state density function $\omega(E(T_t))$ is **the number of possible states of the system per unit energy in the population at time** t (see Fig. 5.1). This description reveals the information retained in equilibrium properties of the system at all temperatures, because one of the possible states can be at equilibrium. The density of states can be found by collecting the states with energy $E(T)$ at temperature T. In annealing, for energies E near $\bar{E}$ may be taken to be continuous, to a good approximation [106, 81, Whtte, 1984;Reif, 1965]. The average energy $< E >$ for population $\vec{P}(t)$ after separating by GASA annealing at a fixed temperature T is calculated by:

$$< E >= \frac{\int_{\Omega_E} E\omega(E)e^{-E/T}dE}{\int_{\Omega_E} \omega(E)e^{-E/T}dE} \tag{5.2}$$

and

Definition 4 $< E >_t$= *average E at time t*

where $E = E(T)$ at $T = T_t$, and Ω_E is the energy domain which include all the generated energy by all the individuals. $exp(-E/T)$ is probability of acceptance into "no crossover" subpopulation. Given an analytical expression for the state density, it is possible to evaluate the integral Eq. (5.2). But to estimate $\omega(E)$ for a particular search problem is very hard in most cases. If $\omega(E)$ is given by a Gaussian (normal)

distribution, then we can approximate E "near" $\bar{E}$ with this distribution.

$$\omega(E) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(E-\bar{E})^2}{2\sigma^2}) \tag{5.3}$$

and the standard deviation of the populations energy, σ, can be represented as:

$$\sigma = \sqrt{\frac{n\sum_{i=1}^{n} E_i^2 - (\sum_{i=1}^{n} E_i)^2}{n(n-1)}} \tag{5.4}$$

where n is the population size. The numerical value for the average energy $\bar{E}$ is evaluated from the following expression:

$$\bar{E} := \frac{1}{n}\sum_{i=1}^{n} E_i(T) \tag{5.5}$$

where $E_i(T)$ is the energy of individual i at temperature T. Thus, the solution [1, Aart & Korst, 1989] of Eq. (5.3) becomes

$$\begin{aligned} < E > &= E_{min} + \sqrt{2}\sigma[\chi + \frac{e^{-\chi^2}}{\sqrt{\pi}(1+erf(\chi))}] \\ \chi &= \frac{1}{\sqrt{2}}[\frac{\bar{E}-E_{min}}{\sigma} - \frac{\sigma}{T}] \end{aligned} \tag{5.6}$$

If T is very high, that is χ is a very large number. So we can find

$$< E > \approx \bar{E} - \frac{\sigma^2}{T} \tag{5.7}$$

In order to reach the pseudo-equilibrium, i.e. $< E > \approx \bar{E}$, meanwhile from Eq. (5.7) the temperature of the system is denoted as $T >> \sigma$.

Fig. 5.2 and Fig. 5.3 display the minimum (E_{min}) and mean energy ($\bar{E}$) for the six different initial temperatures in the previous Sec. 4.4 (Function optimization of Bohachevsky function) in Chapter 4 respectively. When $T_0 = 4000$, the spent generations is much more than those of the other initial temperatures.

On the contrary, a very small initial temperature ($T_0 = 1e-8$) would take more time to get to the same performance, because $T_0 < 0.0006$ ($= \sigma$), which contrasts

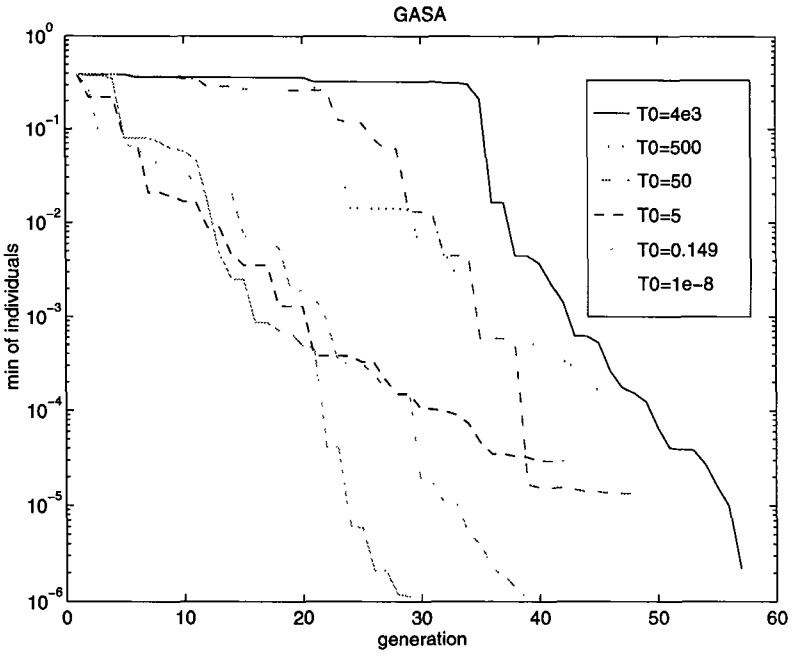


Figure 5.2: *The min energy E_{min} vs. generation for different initial temperatures*

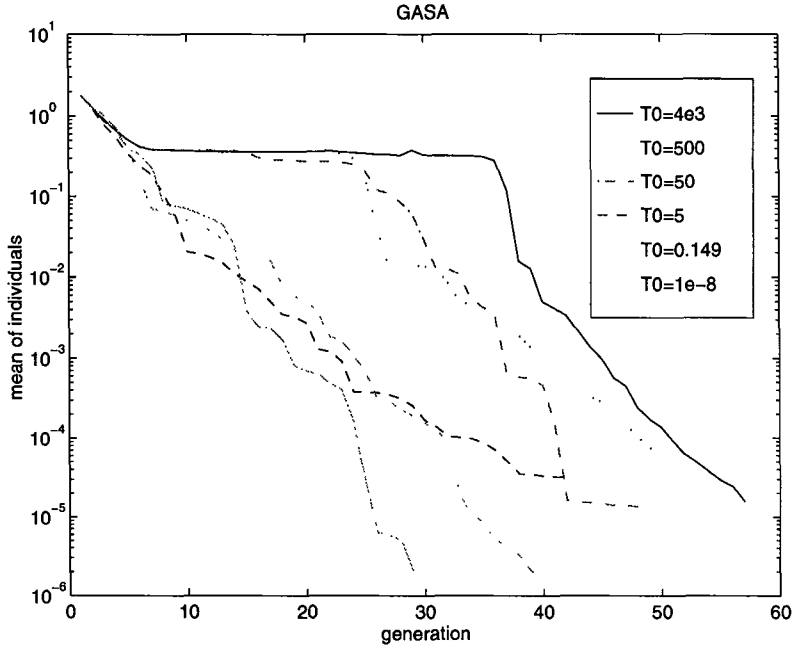


Figure 5.3: *The mean energy $\bar{E}$ vs. generation for different initial temperatures*

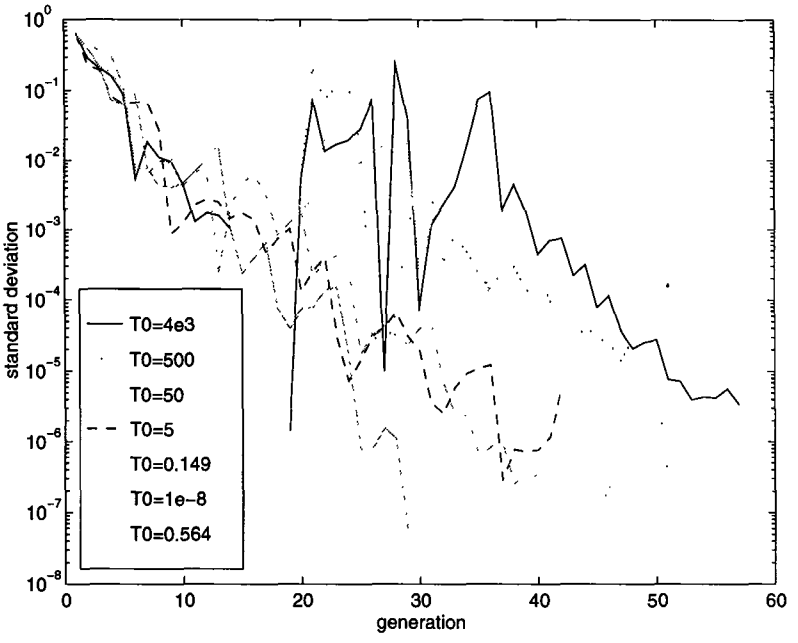


Figure 5.4: *The population's standard deviation σ through generations for different initial temperatures*

Table 5.1: *The turning points for the larger initial temperatures T_0s*

T_0	Gen. t_T	Temp. T_T
4000	35	0.00966
500	23	0.47829
50	24	0.02998

with Eq. (5.7). The very small initial temperatures make the acceptance probability $exp(-(E_i - E_{min})/T) \approx 0$. That means all the individuals will "never" be accepted, and they need to be processed by the genetic operation. At this time, our **GASA** algorithm is turned into a **genetic algorithm**.

When T_0 is applied with large values (see Fig. 5.3), a plateau is stretched over the three curves for $T_0 = 4000, 500, 50$ individually. The turning points for them occur at turning temperature T_T and generation t_T which can be found from Table 5.1. When T goes to infinity, the acceptance probability $\approx$ 1, and all the individuals are retained in the gene pool and skip the genetic operations. This makes an absolute "time waste" for the GASA algorithm. So the individuals in the plateaux have to keep reducing the annealing temperature until a genetic operation happens. Therefore, it is still difficult to point out the upper bound for the initial temperature. As to precision, at the same generation, the individuals starting at the higher initial temperature found it not easy to reach higher precision. The population size $\lambda = 50$ is applied here and all simulations are according to the same initial population. However, in fact, the system with the higher initial temperature spends more generation (time) to reach the global optimum.

5.1.2 Temperature scale

The other concern about a simulated annealing algorithm is how to make a temperature scale to reach the highest efficiency. The considered temperature scale is also known as the **cooling schedule**. Several approaches have been proposed by using the standard deviation to determine the next temperature decrement [2, 3, 73, 88, Aarts & Laarhoven, 1985a; 1985b; Otten & Ginneken, 1984; Sechen, 1988]. The advantage of these approaches is that the temperature is controlled dynamically by the annealing process itself, which is applicable to many different optimization problems. No reliably general proof for methods of the annealing scale across all kinds of search problems have been reported in the literatures [52, Ingber, 1992].

Huang et al. [49, Huang et al., 1986] applied the so-called **annealing curve**, a plot

of average cost $< E(T_t) >$ versus the log of the temperature, to guide the temperature decrement. The slope of the annealing curve is

$$\frac{d < E >}{dln(T_t)} = T\frac{d < E >}{dT_t} \tag{5.8}$$

From the well known relationship [81, Reif, 1965],

$$\frac{d < E >}{dT} = \frac{\sigma^2}{T^2} \tag{5.9}$$

Replace Eq. (5.8) with Eq. (5.9), and we have

$$\frac{d < E >}{dln(T_t)} = \frac{\sigma^2}{T} \tag{5.10}$$

Using a linear approximation of the slope,

$$\frac{\Delta E}{ln(T_{t+1}) - ln(T_t)} = \frac{\sigma^2}{T} \tag{5.11}$$

and ΔE is the difference in cost at temperature T_{t+1} and T_t. The analysis of search phase can help us find out the initial temperature of the system, we have to define the search phase and the cooling schedule. Rearrangement of the previous equation results in

$$T_{t+1} = T_t \exp(\frac{T\Delta E}{\sigma^2}) \tag{5.12}$$

A pseudo-equilibrium could be maintained by requiring the expected decrease in average energy to be less than σ, let $\Delta E = -\alpha\sigma$. Thus, the above equation can also written as

$$T_{t+1} = T_t \exp(\frac{-\alpha T}{\sigma}) \tag{5.13}$$

The above equation can be simplified as follows.

$$T_{t+1} = c(T_t) * T_t \tag{5.14}$$

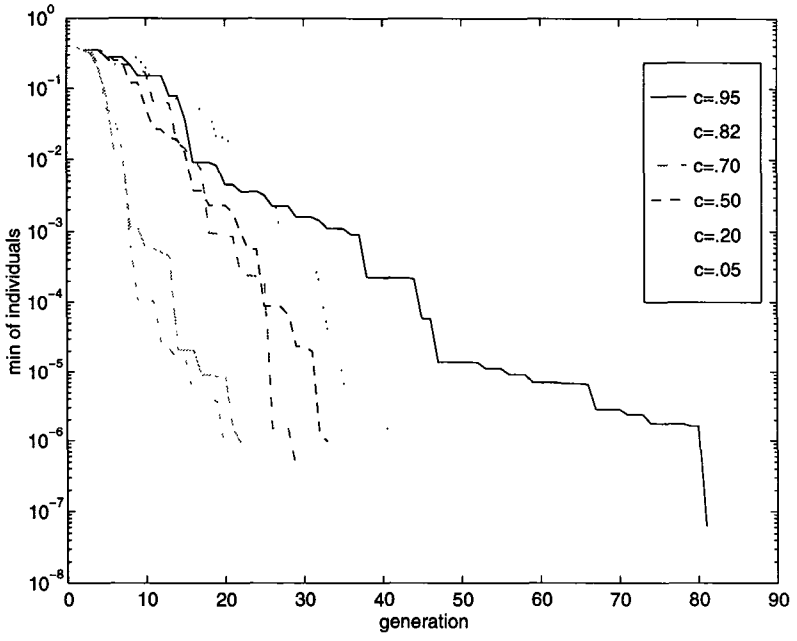


Figure 5.5: *Minimum energy vs. generation for different annealing coefficients*

where $c(T_t)$ is the temperature decrement control parameter or so-called **annealing coefficient**. Fig. 5.5 $\sim$ Fig. 5.8 are considered with six different annealing coefficients of c. The annealing speed runs as fast as the decrease of the annealing parameter c (see Fig. 5.5 & Fig. 5.6).

But for $c \leq 0.5$, the annealing process is so fast that a **quench** condition occurs [50, 52, Ingber, 1989;1992]. Thus, "quench" means that a fast temperature decrement (i.e. fast cooling) makes the objective function reach the optimum globally. In metallurgy, coarser crystals may be found from "quench" instead of "annealing". That is, the quench process usually makes unstable situations happen which is discussed in Sec.5.5. Several researchers used **simulated quench** (**SQ**) on complex circuits design problems, including several layers of logic hierarchy [52, Ingber, 1992].

Fig. 5.5 describes the minimum energy through generations for different annealing

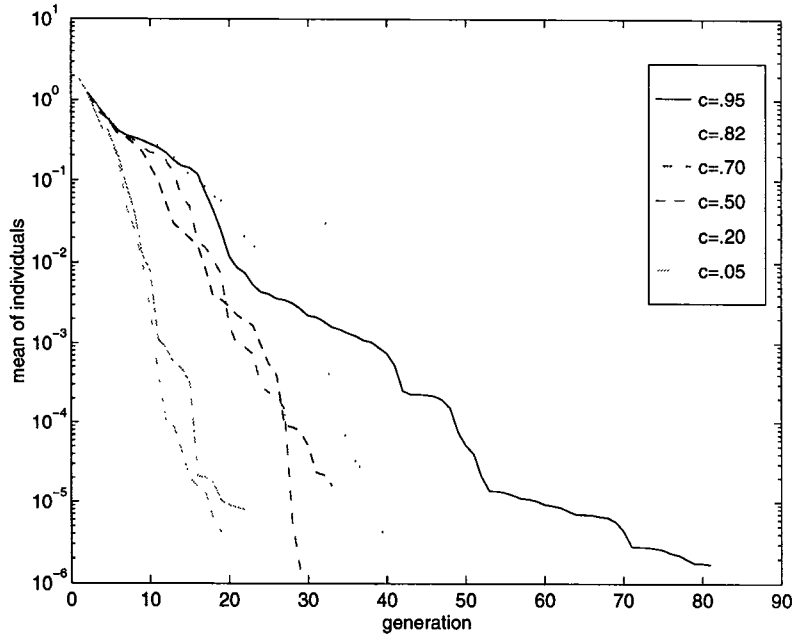


Figure 5.6: *Average energy vs. generation*

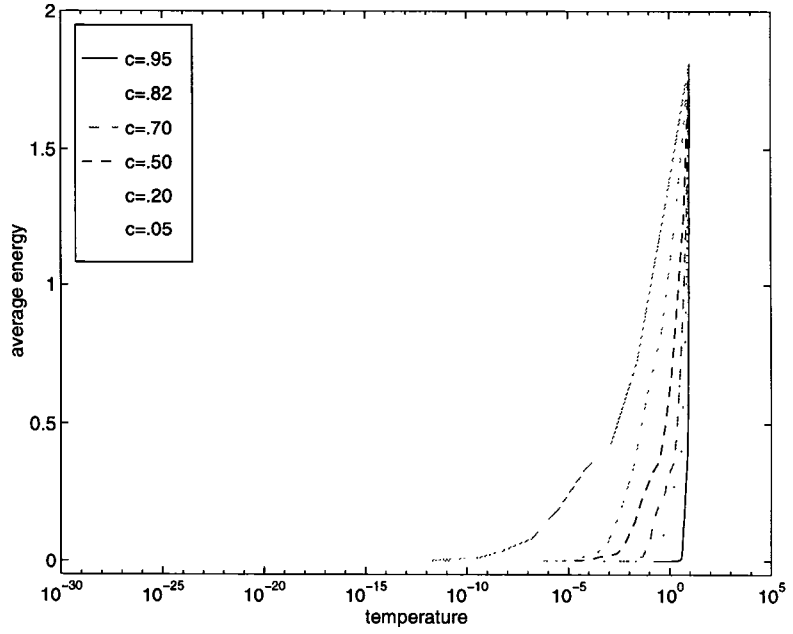


Figure 5.7: *Average energy vs.* $\log(T_t)$ *for different annealing coefficients*

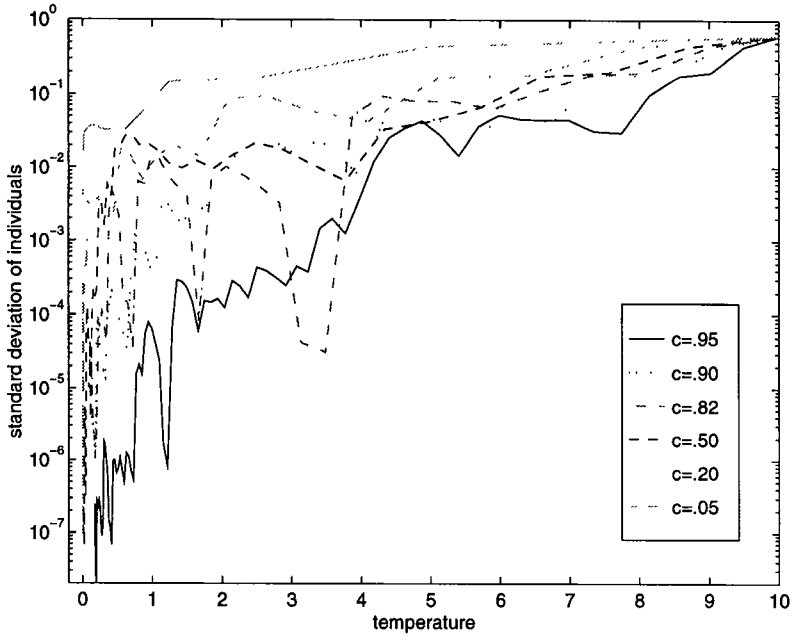


Figure 5.8: *Standard deviation vs. temperature (semilog)*

schedules. GASA spends the most and least generations to reach the global minimum for $c = 0.95$ and $c = 0.20$, respectively. Fig. 5.6 shows the same tendency for the average energy as the min energy in Fig. 5.5. Fig. 5.7 illustrates the average energy through different temperatures. The falling speed of the average energy follows the increase of the annealing coefficient c. The standard deviation of individuals versus temperature can be found in Fig. 5.8. When $c = 0.95$, a slowest but very smooth annealing is operated in GASA. The standard deviation becomes the lowest while the final temperature is close to 0. That is, the diversity of the participating individuals becomes less and all the individuals come to be uniform. This matches the physical phenomenon of annealing for the fine crystals which are found by slow annealing. If $c = 0.05$, a fast "quench" occurs in the operation. The diversity of the joining individuals is still large even through the final temperature is pretty small. This type

of analysis can help improve the performance of the annealing algorithms.

In general, when **quench** occurs in the annealing process, the arrival of equilibrium, on the contrary, sometimes takes more time than the regular annealing coefficient ($0.7 \leq c \leq 0.99$) such as $c = 0.50$. In physical metallurgy, this condition would usually produce coarse crystals. Similarly, **quench** results in more time to find a better solution. When c is 0.20, compared to other curves, it has a higher performance. So how to make an ideal annealing scale is very important in the annealing algorithm even with the "quench" schedule. From the analysis of Fig. 5.5 and Fig. 5.6, the slower cooling scale spends more time (generations) to reach the pseudo-equilibrium. Fig. 5.8 reveals that a smaller standard deviation value can be found from the curve $c = 0.95$ for the same temperature. So the curve $c = 0.20$ shows the most unstable condition in this plot.

5.2 The Effect of Partial Annealing Mechanism

The annealing environment on the GASA algorithm can be altered by assuming no annealing (equilibrium) either for crossover or mutation. From these alterations, two other GASA algorithms are found with the names of GASA_C (annealing on crossover only) and GASA_M (annealing on mutation only). According Sec. 4.3, the GASA_C algorithm can be illustrated as the follows.

Algorithm GASA_C
Parameters declaration:
$t = 0$; (the initial generation);
t_{max} is the allowed maximal number of generations;
T_0 = initial temperature;
Tm = number of temperature moves to attempt;

C is the crossover operator;

M is the mutation operator;

Ξ is the total gene pool per generation;

Initial $\vec{P}(0) = \{\underline{\nu}(0)\} \in U$ where $U = \{0,1\}^l$;

Evaluate $\vec{P}(0) = \{\Phi(\nu_1(0)), \cdots, \Phi(\nu_l(0))\}$;

where $\Phi(\nu_k(0)) = \delta(f(\Gamma(\nu_k(0))), P(0))$;

For m =1,Tm

while $(t < t_{\max})$

For each individual i,

Evaluate the change of energy $(E_i - E^*)$ and find a probability p_i;

if threshold probability $p_i >$ random [0, 1),

keep the accepted individual in a gene pool Ξ_c;

else

put i into Ξ_{nc};

Crossover: $\nu'_k(t) = C_{\{p_c\}}(\vec{P}_c(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_c(t) \in \Xi_{nc} (= \Xi - \Xi_c)$, p_c is the crossover rate;

replace Ξ with $\Xi_c + \Xi_{nc}$;

Mutate: $\nu''_k(t) = M_{\{p_m\}}(\vec{P}_m(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_m(t) \in \Xi$, p_m is mutation rate;

Until equilibrium;

$T = update(T)$;

Evaluate $\vec{P}(t) = \{\Phi(\nu_1(t), \cdots, \Phi(\nu_l(t))\}$,

where $\Phi(\nu_k(t) = \delta(f(\Gamma(\nu_k(t))), P(t - \omega))$;

Select $\vec{P}(t+1) = sel(P(t))$;

Reproduce $\vec{P}(t+1)$;

}; end t

$t = t + 1$;

}; end move

From the above description, the main change in GASA_C algorithm is that we don't have to calculate the energy change before the mutation operation, because the individuals $\vec{P}_m(t)$ in the whole gene pool Ξ are forced to process mutation.

Likewise, when we only anneal the individuals in mutation, the other alternative of GASA is found as GASA_M. The algorithm design is similar to GASA_M and can be illustrated as the following:

Algorithm GASA_M

Parameters declaration:

$t = 0$; (the initial generation);

t_{max} is the allowed maximal number of generations;

T_0 = initial temperature;

Tm = number of temperature moves to attempt;

C is the crossover operator;

M is the mutation operator;

Ξ is the total gene pool per generation;

Initial $\vec{P}(0) = \{\underline{\nu}(0)\} \in U$ where $U = \{0,1\}^l$;

Evaluate $\vec{P}(0) = \{\Phi(\nu_1(0)), \cdots, \Phi(\nu_l(0))\}$;

where $\Phi(\nu_k(0)) = \delta(f(\Gamma(\nu_k(0))), P(0))$;

For m =1,Tm

while ($t < t_{\max}$)

Crossover: $\nu'_k(t) = C_{\{p_c\}}(\vec{P}_c(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_c(t) \in \Xi$, p_c is the crossover rate;

For each individual i,

Evaluate the change of energy $(E_i - E^*)$ of the individuals from Ξ and $\nu'_k(t)$, and find a probability p_i;

if accepted probability p_i > random [0, 1),

keep the accepted individual in a gene pool Ξ_m;

else

Table 5.2: *The spent generations and time for GAs and GASAs*

	Generation
GAs	46
GASA	40
GASA_C	39
GASA_M	43

put i into Ξ_{nm};

Mutate: $\nu''_k(t) = M_{\{p_m\}}(\vec{P}_m(t)), \forall k \in \{1, \cdots, l\}$

$, \forall \vec{P}_m(t) \in \Xi_{nm} (= \Xi - \Xi_m)$, p_m is mutation rate;

replace Ξ with $\Xi_m + \Xi_{nm}$;

Until equilibrium;

$T = update(T)$;

Evaluate $\vec{P}(t) = \{\Phi(\nu_1(t), \cdots, \Phi(\nu_l(t))\}$,

where $\Phi(\nu_k(t) = \delta(f(\Gamma(\nu_k(t))), P(t - \omega))$;

Select $\vec{P}(t+1) = sel(P(t))$;

Reproduce $\vec{P}(t+1)$;

}; end t

$t = t + 1$;

}; end move

The comparison of GASA algorithms (GASA, GASA_M and GASA_C) and GAs (genetic algorithms) are illustrated in Fig. 5.9~Fig. 5.12. The test example is the Bohachevsky function. The applied population size is 50 and the annealing coefficient c is 0.82. The initial populations for these algorithms are all the same.

The spent generations are listed in Table 5.2. From this table, we can find GAs needs the most generations to reach the stop criterion ($1e-6$). For GASA_M, it takes

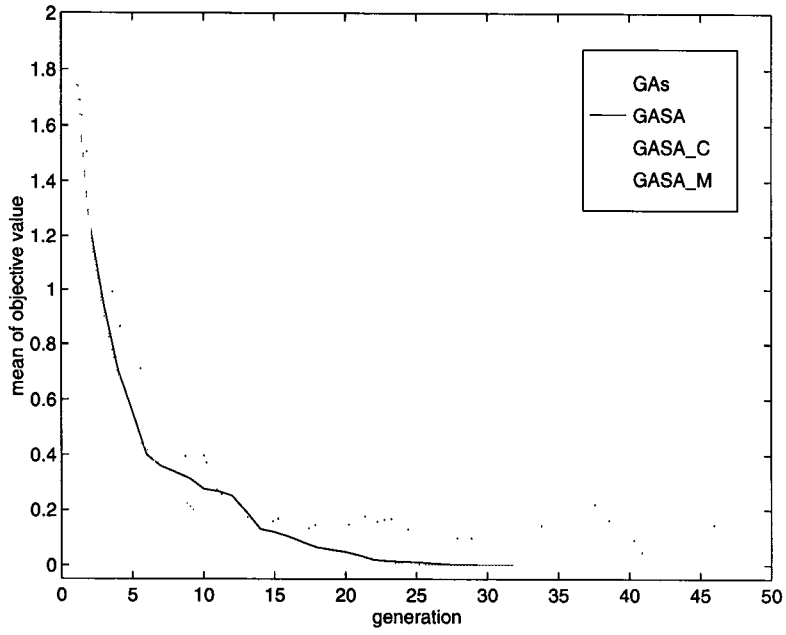


Figure 5.9: *Mean objective value vs. generation for GAs and GASAs*

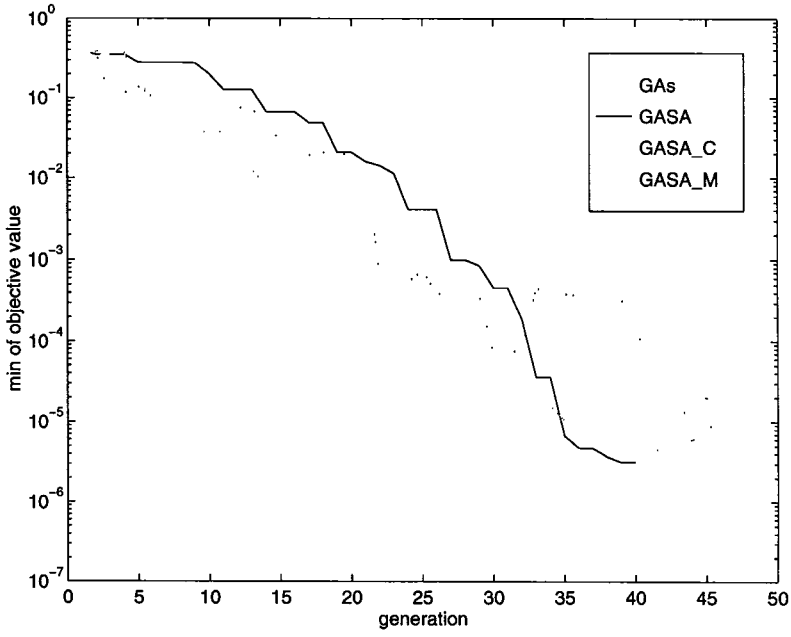


Figure 5.10: *Min objective value vs. generation for GAs and GASAs*

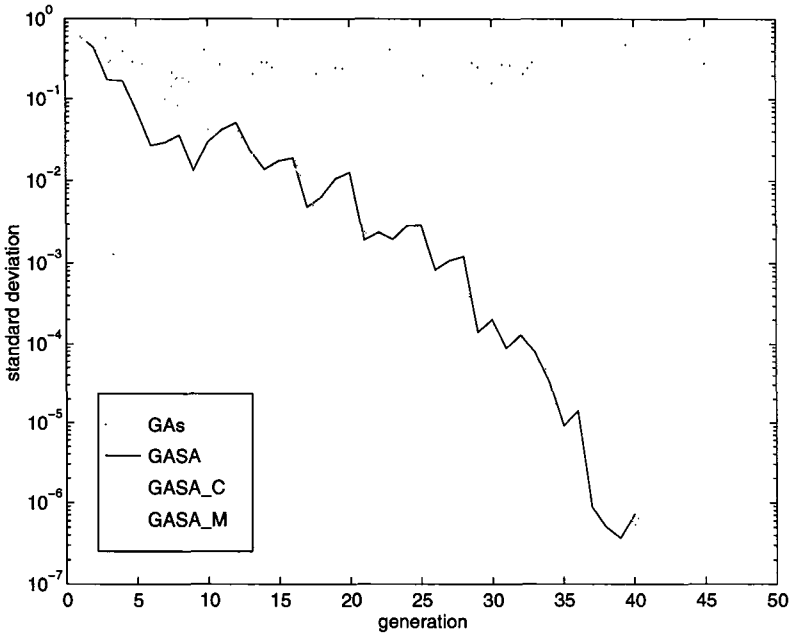


Figure 5.11: *Standard deviation vs. generation for GAs and GASAs*

43 generations which are more than those of the other GASAs. GASA and GASA_C almost take the same generations to reach this cutoff value.

From Fig. 5.9, GASA algorithms almost converge at generation 27, which are superior to GAs. Fig. 5.10 microscopes the calculation of GAs and GASAs. When at cutoff $> 1e-4$, GASA_M does the best job compared with others. But for cutoff $< 1e-4$, GASA and GASA_C converge faster than GASA_M especially for GASA_C. As for the analysis of these algorithms, Fig. 5.11 & Fig. 5.12 shows GAs still stay at a higher oscillation. GASA_M converges fastest, GASA second, then GASA_C is the third. The lower and lower standard deviation curves of GASAs reveal that GASAs have the strong self-adaptive ability compared to GAs. Fig. 5.13 is the annealing schedule for this simulation.

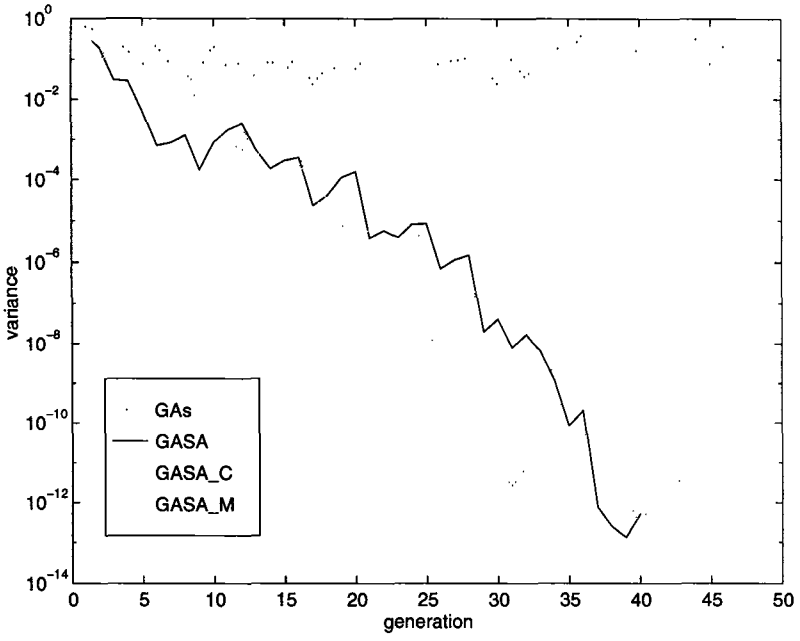


Figure 5.12: *Variance vs. generation for GAs and GASAs*

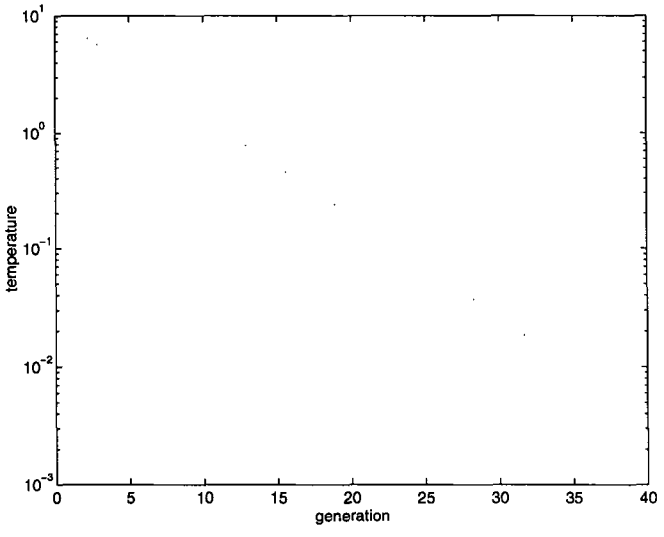


Figure 5.13: *Annealing schedule for GASAs*

5.3 Search Phase and Schema Theory

The other important analytical characteristics of GASA algorithms is how it can search. The search of this algorithm can be analyzed according to hyperplane exchange. One hyperplane is composed of a schema. The hyperplane exchange means the change of the population through generations in the search space.

The search phase of a genetic-annealing algorithm as outlined, in which each succeeding solution in the solution sequence is determined stochastically based on the current solution, suggesting that the behavior of algorithms can be described as a Markov chain. A search problem can be represented by the pair (Ω, E), whereby Ω is the search space and E is the objective (energy) function. Assume that Ω is a finite space, so GASA algorithms can be formulated by a 9-tuple:

$$GASA \;=\; (\vec{P}(0), \lambda, l, s, P_T, \rho, \Phi, \tau, \zeta) \tag{5.15}$$

where

$\vec{P}(0) \in U^\lambda, U = \{0,1\}^l$	initial population
$\lambda \in \mathbf{N}$	population size
$l \in \mathbf{N}$	length of representation of each individual
$s : U^\lambda \to U^\lambda$	selection operator
P_T :	a stochastic matrix which describes state transition
$\rho = \{C, M\}$	genetic operators (C: crossover,M: mutation)
$\Phi : U^\lambda \to \mathcal{R}$	fitness function
$\tau = \{T_t, t = 0, 1, 2, \cdots, f\}$	a finite length which decreases temperature
$\zeta =$ the stop criterion monotonously.	

Definition 5 (Aarts & Korst, 1989) *A* **transition state** *is a combined action resulting*

in the transformation of a current solution into a subsequent one. The action consists of the following steps: (1) application of generation mechanism, (2) application of the acceptance criterion.

The GASA algorithm design has been discussed in Chapter 4. The abstracted description can be illustrated as follows. In the beginning, an initial population $\vec{P}(t)$ is created randomly. Then let $\vec{P}(t)$ be selected by s, this selection is according to the fitness function Φ. After that, the probability P_T will determine the qualified individuals to be retained in the gene pool, and the unqualified ones will be crossed-over. Again the procedure is repeated by the mutation operation. Following the decreasing temperature, the new population will be put into the same procedures until the stop criterion (ζ) is met.

Here we only consider the analysis of its characteristics. The GASA algorithm still uses the main property of the genetic algorithm by applying binary encoded individuals. The temperature factor is also an important factor in the algorithm.

The mathematical analysis of the population-based search can be followed by two ways, the first one could be from schema theory [46, 38, 107, Holland, 1975; Goldberg, 1989; Whitley, 1993], the other one from simulated phase transition. Here we introduce schema theory first. In fact, the search mode of the genetic-annealing algorithms is a hyperplane exchange.

5.3.1 Schema theory

The term **schema** (plural schemata) refers to a similarity template which describes a subset of strings with similarity at certain string positions (in a more straight way schemata are hyperplanes of varying dimensions in the $l-$ dimensional space). The

hyperplanes that are relevant in the case are defined by schemata over binary strings. A schema is a string defined to be elements of the set $\{0, 1, *\}^l$ and for given schemata $H \in \{0, 1, *\}^l$. The $*$ symbol is a wild card that allows the specified bit position to take on the value 0 or 1. Therefore, the schema $1 * * * * * *$ would represent a hyperplane partition that contains all strings of length $l = 7$ with a "1" bit in the initial position. So a schema is an element of a hyperplane.

Each individual of length l contains 2^l schemata. A population size λ contains between 2^l and $\lambda \cdot 2^l$. The order of schema, denoted by $o(H)$ is defined as the number of fixed positions (0's or 1's). The **defining length** of the schema, $\delta(H)$, is defined as the distance between the first and last final position in a string. For example, the schema $01 * 0 * 10*$ has order 5 and length 6.

Intuitively, we can observe that schemata with low order and length have a better chance to survive **crossover** than the others. This is because the probability of being cut by the crossover operator is less for them. Let's consider the following two schemata H_1 and H_2

$$H_1 = * 1 * * * 0$$

$$H_2 = * * 1 0 * *$$

From Fig. 5.14, a certain arrangement of the schemata H is similar to the double-ellipsoidal structure of the chromosomes in biology. But in fact, a schema can be a number only in our mathematical analysis. Schema H_1 is less likely to survive than H_2. Specifically, H_1 has five possible crossover sites. The probability of it being destroyed is $\delta(H_1)/(l-1) = 4/5$ and its survival probability is $1/5$. Similarly H_2 is destroyed with a probability of $1/5$ and has a survival probability of $4/5$. In Fig. 5.14,

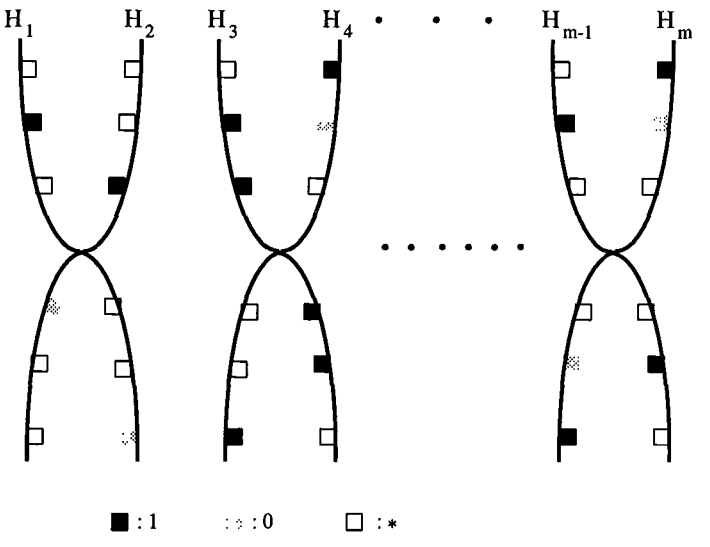


Figure 5.14: *Schemata $H_1, H_2, \cdots, H_m$ and their genetic models*

the schemata are the analogs from the double-helix chromosomes with in biology.

Let $\vec{P}(t)$ represent the population at time t. Denote by $m(H,t)$ the expected number of chromosomes containing a hyperplane (schemata) H within population $\vec{P}(t)$. The expected number of chromosomes containing the schemata H within population $\vec{P}(t+1)$ is represented by the following theorem.

Holland (1975) proposed the famous **schema theorem** as the fundamental of genetic algorithms. Here we restate it as below.

Theorem 1 (John Holland, 1975) *In a canonical genetic algorithm using a proportional selection and single-point selection, and for a single-position mutation, the following holds for each schemata (hyperplane) H represented in $\vec{P}(t)$:*

$$m(H,t+1) \geq m(H,t)\frac{\Phi(H,t)}{\bar{\Phi}(t)}[1 - p_c\frac{\delta(H)}{l-1}](1-p_m)^{o(H)} \qquad (5.16)$$

where $\Phi(H,t)$ is the average fitness of the chromosomes containing the schemata H at time t, $\bar{\Phi}(t)$ is the average fitness for all chromosomes in $\vec{P}(t)$ at time (generation)

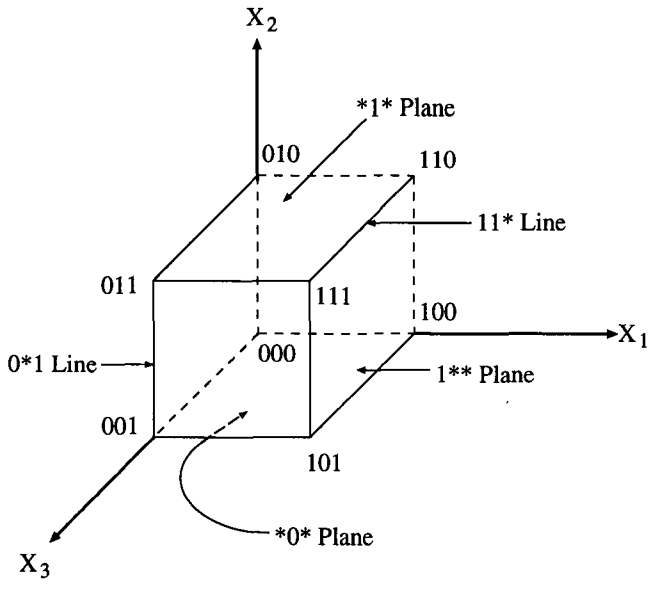


Figure 5.15: *Visualization of schemata as hyperplanes in 3-dimensional space*

t, p_c and p_m are the crossover and mutation rate respectively, $\delta(H)$ is the defining length of hyperplane H, and l is the length of each chromosome.

For small values of p_m ($p_m << 1$), $(1 - p_m)^{o(H)}$ can be approximated to $1 - o(H)p_m$, so we can rewrite Eq. (5.16) as

$$m(H, t+1) \geq m(H, t)\frac{\Phi(H, t)}{\bar{\Phi}(t)}[1 - p_c\frac{\delta(H)}{l-1}](1 - p_m \cdot o(H)) \qquad (5.17)$$

What this equation implies is that above-average schemata that survive crossover and mutation (usually short length and low-order) receive exponentially increasing instantiations in subsequent generations. Due to the importance of this theorem it is also called the Fundamental Theorem of Genetic Algorithms [38, Goldberg, 1989] or so-called **schema theory**.

Perhaps the best way for us to understand how a population-based algorithm can sample hyperplane partitions is to consider a 3-D space (see Fig. 5.15) [38, Goldberg, 1989]. Suppose we have a problem encoded with just 3 bits, this can be exactly

represented as a simple cube with the string 000 at the origin. For example, the string 011 not only samples its own corner in the hypercube (011) but also the hyperplanes represented by the schemata 0**, *1*, **1, 01*, 0*1, *11, and even ***. So a population of sample points provides numerous hyperplanes; furthermore, low order hyperplanes should be sampled by numerous points in the population. Thus the hyperplane sampling is only responsible for "information exchange".

5.4 Simulated Phase Transition

The other corresponding explanation for the moves of the individuals through the energy change is illustrated by **simulated phase transition**. In metallurgy, it is well known that there are two classes of alloys: homogeneous (one phase) alloys and eutectics (two phase systems). In the former, the concentration of each component is constant through the alloy, whereas, in the latter, there are small crystals. Obviously, the homogeneous one can be seen as an **order** phase. Generally speaking, the phase transition is one of the cooperative phenomena; it is a process which depends upon the interplay of a large number of atoms rather than on the individual atoms. To simplify the physical phenomena to our artificial environment, we can take two phases (order & disorder) to simulate them. In GASA algorithms, the genes (binary bits) of the chromosomes (individuals) are similar to atoms. The corresponding behavior of genes and atoms is our simulations create the structure that alters in the applied genetic operation.

However, when the individuals (chromosomes) follow a certain arrangement which results from the applied mathematical models and the constraints. At this time, the individuals are in **order phase**. Otherwise, they are in **disorder phase**. But

several factors can change this arrangement, and make the new created individuals go into a new phase because of the genetic operation and temperature change. The artificial environment which the individuals should make self-adaptation comes from the complexity of our problems.

5.4.1 Qualitative analysis

The concept of the simulated phase transition is extracted from the phase transition in thermodynamics. First, we have to clarify several important terms in the search problem. The elements of the simulated phase are, in fact, the candidate solutions to the applied problems. They are the individuals which join this computation. One phase is composed of many elements. About the **phase**, we can describe as below.

Definition 6 *A* **phase** *is a hyperplane composed of many schemata. The behavior of phase is limited by the supplied problems and their constraints and computational paradigm which make the artificial environment.*

The above definition of the phase indicates that one phase is one population at time t, and a phase can be a collection of lines, planes or hypercubes, and even a more complex surface in multi-dimensional space. In GASA, individuals with high energy (low fitness) tend to undergo mutation and crossover. For those that undergo crossover, longer schemata disrupt more often.

The other important term is the concept of **equilibrium** [59, Kyle, 1984]. A specific discussion is in Sec.5.5. Here we only make a brief introduction.

Definition 7 Equilibrium *in our genetic-thermodynamic algorithms is a word denoting a static condition, the absence of changes for the individuals in the population.*

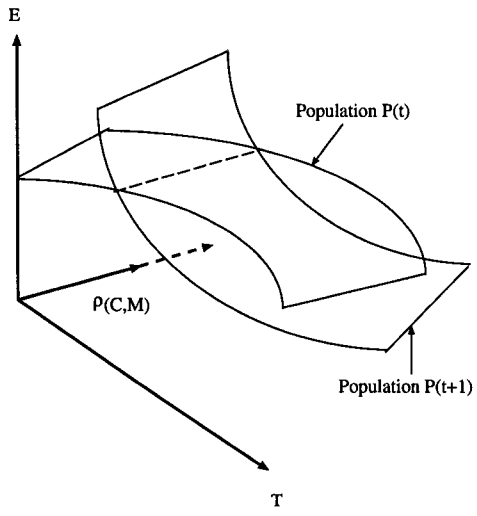


Figure 5.16: *An intuitive sketch of the relationship of populations*

In computation, it is taken to mean not only the absence of change but the absence of tendency toward change in an allowed criterion.

Thus a system at equilibrium is their individuals which exist under such condition that is no tendency for a change in the state to occur. Because any tendency toward change is caused by a driving force, the absence of tendency to change also indicates the absence of driving force. As to computation, it is difficult for us to classify what is a pure matter during search. Therefore, a population being in equilibrium is described as one in which all forces are in exact balance. However, all kinds of driving forces tend to bring about the sudden change of the individual's energy. For example, when two individuals are crossovered, their energy may have a little change. That is their genetic structures have been altered. The obvious observation will occur on their offspring. So the crossover operator can be a driving force.

The individuals make major changes on their structure. After this change, the locations of the parents will be different from the offsprings'. However, when the

parents face some driving forces and have made their structure's alteration, the location of the new population $\vec{P}(t+1)$, of course, will be different from the parents $\vec{P}(t)$. So the simulated phase transition has occurred at this moment. This change is illustrated in Fig. 5.16. T is the population temperature at time t and ρ is the genetic operator (see Sec.4.3). E is the energy, in fact, which is the objective value. One phase can be a hyperplane. Either genetic operation or temperature change always can determine the new locations of the populations. The observed measurement is the location change or energy change for that. Generally speaking, the transition is that an individual seeks the new surviving conditions to adapt to the artificial environment.

5.4.2 Quantitative analysis

The observed behavior of the individuals is by calculating their objective values (energy). To determine the criterion of transition, we can take **energy** into account. In the annealing process, the individuals follow the Boltzmann distribution P_T. According to Eq.4.9, the state probability is found by

$$P_T(E) = \frac{e^{\frac{-E_i}{T}}}{Z} \tag{5.18}$$

where $Z = \sum_{j\in\Omega} e^{-E_j/T}$, $T = T_t$ at time t, Ω is the search space defined as above.

The state transition matrix P_T also can be decomposed into two parts for convenience in the following. It consists of the next state generation mechanism, G_{ji}, which describes the probability of generating state i from j. and A_{ji} denotes the acceptance probability which describes the probability of accepting the generated state. Thus $P_T(j,i)$ can be found by

$$P_T(j,i) = \begin{cases} G_{ji}(T_t)A_{ji}(T_t) & i \neq j \\ 1 - \sum_{k=1}^{\lambda} G_{ki}(T_t)A_{ki}(T_t) & i = j \end{cases} \tag{5.19}$$

In this result, λ is the population size.

As for the generating probability, $\forall\ i, j\ \in \Omega$, the probability G_{ji} is written with the class of Gaussian-Markovian systems,

$$G_{ji} = G(\Delta x) = (2\pi T)^{-D/2} \cdot exp[-\Delta x^2/(2T)] \tag{5.20}$$

where $\Delta x\ =\ x_i - x_j, i = 1, 2, \cdots D$.

The acceptance probability is

$$A_{ji} = min\{1, exp[-\frac{E_i - E_j}{T_t}]\} \tag{5.21}$$

From our proposed modified Metropolis criterion (MMC) (see Sec.4.3.1), j is replaced with the minimum location (x_{min}) for individual $i\ \in\ \vec{P}(t)$ in Eq. (5.20) and Eq. (5.21), then G_i and A_j will become as follows.

$$G_i = G(\Delta x) = (2\pi T)^{-D/2} \cdot exp[-\Delta x^2/(2T)] \tag{5.22}$$

where $\Delta x\ =\ x_i - x_{min}, i = 1, 2, \cdots D$.

The acceptance probability A_i is

$$A_i = min\{1, exp[-\frac{E_i - E_{min}}{T_t}]\} \approx exp[-\frac{E_i - E_{min}}{T_t}] \tag{5.23}$$

Thus the schema $m(H, t+1)$ in $\vec{P}(t+1)$ after genetic-annealing operation of $\vec{P}(t)$ can be represented with the modification from Eq. (5.17) as

$$m(H, t+1) \geq m(H, t)\frac{\Phi(H, t)}{\bar{\Phi}(t)}[1 - p_c\frac{\delta(H)}{l-1}\overbrace{(1 - P_T^c)}^{\text{crossover state transition}}](1 - p_m \cdot o(H)\underbrace{(1 - P_T^m)}_{\text{mutation state transition}}) \tag{5.24}$$

P_T^c and P_T^m are the state transition probability determined by temperature T_t for crossover and mutation operations respectively. In fact, the arriving probability means that the individuals which can arrive beyond the MMC at equilibrium. Furthermore, it can be rearranged into

Theorem 2 (GASA; Original Contribution) *The schema* $m(H,t+1)$ *in* $\vec{P}(t+1)$ *after genetic and full annealing operation (GASA) of* $\vec{P}(t)$ *can be described as*

$$m(H,t+1) \geq m(H,t)\frac{\Phi(H,t)}{\bar{\Phi}(t)}[1-p_c\frac{\delta(H)}{l-1}+p_c\frac{\delta(H)}{l-1}P_T^c)](1-p_m\cdot o(H)+p_m\cdot o(H)P_T^m)) \tag{5.25}$$

The P_T^ρ can be described as

$$P_T^\rho = exp[-\frac{E_i - E_{min}}{T_t}] \tag{5.26}$$

,where $\rho = c\ (crossover)\ or\ m\ (mutation)$.

The mutants of the individuals come from their genetic alterations. So after the initial generation, if $G_i \approx 1$ and $exp[-\frac{E_i-E_{min}}{T_t}] < 1$, so Eq. (5.26). Substitute this result into Eq. (5.25), we have the direct observation of the schema $m(H,t+1)$ is related to the individual's energy. The increase of the schema is exponentially growing because of Eq. (5.26). The above derivation is from schema theory and simulated phase transition and used for full annealing. The result can be derived with other partial annealing mechanisms for GASA_C and GASA_M as follows. For GASA_C, without annealing on mutation, we have

Theorem 3 (GASA_C; Original Contribution) *The schema* $m(H,t+1)$ *in* $\vec{P}(t+1)$ *after genetic and partial annealing operation (GASA_C) of* $\vec{P}(t)$ *can be described as*

$$m(H,t+1) \geq m(H,t)\frac{\Phi(H,t)}{\bar{\Phi}(t)}[1-p_c\frac{\delta(H)}{l-1}(1-P_T^c)](1-p_m\cdot o(H))) \tag{5.27}$$

For GASA_M, without annealing on crossover, we have

Theorem 4 (GASA_M; Original Contribution) *The schema* $m(H,t+1)$ *in* $\vec{P}(t+1)$ *after genetic and partial annealing operation (GASA_M) of* $\vec{P}(t)$ *can be described as*

$$m(H,t+1) \geq m(H,t)\frac{\Phi(H,t)}{\bar{\Phi}(t)}[1-p_c\frac{\delta(H)}{l-1}](1-p_m\cdot o(H)(1-P_T^m)) \tag{5.28}$$

The above are the analytical characteristics of GASA algorithms.

5.5 Population Turbulence

A new concept of population turbulence is used in our mathematical analysis. It can be defined as below.

Definition 8 *The* **turbulence** *of the population in the genetic-thermodynamic operation is determined by the difference between the average energy of a population and the minimum energy of all individuals in this population.*

Definition 9 *The calculation of the turbulence for each population through generations can be represented as the mean value minus the minimum value of each population.*

When the turbulence is zero, this result is consistent with the static equilibrium for populations and can be plotted by an "equilibrium line" which is broadly used in the next discussions.

Lemma 5.5.1 *If* $\bar{E} = E_{min}$ *then* $E_i = E_{min}, \forall i$.

If $n(t)$ individuals in the population t, the proof is illustrated as below.

proof:

Because $\bar{E} = E_{min}$,

so we have $\frac{\sum_{i=1}^{n}}{n} E_i = E_{min}$.

Thus $\frac{E_1+E_2+\cdots+E_n}{n} = E_{min}$

But $E_{min} \leq E_i, i = 1, 2, \cdots, n$

Since $\sum_{i=1}^{n} E_i = n \cdot E_{min}$

$\Longrightarrow E_1 + E_2 + \cdots + E_n = \underbrace{E_{min} + E_{min} + \cdots + E_{min}}_{\text{n}}$

Therefore $E_{min} = E_i$

Thus, the standard deviation $\sigma(E(T_t))$ for the population $\vec{P}(t)$ is 0. That means all the individuals become the minimum one. So the equilibrium can be formulated as

$$< E(T_t) > \approx E_{min}(T_t) \tag{5.29}$$

We can substitute Eq. (5.29) into Eq. (5.6), then the difference term in Eq. (5.6) is set to 0. So the equilibrium temperature of each population at generation t is described as

$$\chi = -\frac{e^{-\chi^2}}{\sqrt{\pi}(1 + erf(\chi))} \tag{5.30}$$

and

$$\chi = -\frac{\sigma}{T} \tag{5.31}$$

where $T = T_t$ as above. So we can replace Eq. (5.30) with Eq. (5.31), a new population equilibrium is found by

$$\begin{aligned} \frac{\sigma}{T_t} &= \frac{exp(-\frac{\sigma^2}{T_t^2})}{\sqrt{\pi}[1 - erf(\frac{\sigma}{T_t})]} \\ &= \frac{exp(-\frac{\sigma^2}{T_t^2})}{\sqrt{\pi}[erfc(\frac{\sigma}{T_t})]} \end{aligned} \tag{5.32}$$

We can find the equilibrium, when $T_t = T_0$ from Eq. (5.32). From the above analysis, we can find if the $E_{min} =$ *the local minimum*, the equilibrium is called **local equilibrium**. This condition usually makes many traditional hill-climbing methods get stuck locally. But if $E_{min} =$ *the global minimum*, the system is at global equilibrium. When global equilibrium is reached, the global search will be completed

successfully. In general, we only can attain the conditional global equilibrium, that is, in an allowed error to the global minimum.

The average energy $< E >$ is evaluated from the average of the objective values at each generation as described in the following paragraphs. For example, from Sec.5.1, with the same initial population and same annealing schedule, we can calculate the initial equilibrium temperature $T_0 = 0.564$. In comparison with the conditions in Sec.5.1.1, we have Fig. 5.17~Fig. 5.24. From Fig. 5.17 and Fig. 5.18, the best result is found if we apply the equilibrium temperature $T_0 = 0.564$. It almost spends the least generations to get the best solution in this test. Fig. 5.18 indicates that a very high initial temperature (4000) will result in the stagnant effect from generation 8 to 36. Fig. 5.19 also can prove this. As to the temperature change, Fig. 5.20 and Fig. 5.21 show the average energy decrease fastest when $T_0 = 0.564$. Fig. 5.21 and Fig. 5.22 tell us that at $T_0 = 4000$, the annealing shows higher oscillation than the others. The convergence period of the too low temperature ($T_0 = 1e - 8$) is almost equal to the others in addition to the highest temperature curve. That means this temperature is still applicable to some problems, its drawback is the more spent generations compared to others. Fig. 5.23 shows that all the curves stay in the upper phase to the **equilibrium line**. This condition can illustrate the applied annealing schedule is a **never equilibrium** schedule for the populations. If the populations are all in equilibrium, the curves of Fig. 5.24 and Fig. 5.25 should become the horizontal line from the origin $(0, 0)$.

The other test is to compare the different annealing schedules with the same schedules applied in Sec.5.1.2. Fig. 5.26 shows that even with different annealing schedules, the curves with different annealing coefficients still stay in the upper phase to the **equilibrium**. In Fig. 5.27, the lower annealing coefficients make the curves

move leftwards except $c = 0.50$. Smaller annealing coefficients are not easy to reach equilibrium. When $c = 0.95$, we can see it almost follow the equilibrium line close and in parallel (see Fig. 5.26). The annealing with the largest annealing coefficient (0.95) shows the approximately convergence from the flat area between temperature 0 and 4. The convergence of $c = 0.95$ is better than others (see Fig. 5.28). Even with the slowest annealing speed, but we can have the most uniform annealing result compared to others.

Applying this equilibrium rule in Sec.5.2, we can compare characteristics on GASAs and GAs in Fig. 5.29. In Fig. 5.29, GAs curve deviates off the equilibrium line further than GASAs. In the beginning, the behavior is similar to GASA_M, then there is an obvious distance with GASA_M from $E_{min} = 0.39$ to $E_{min} = 0.1$ (during generation $1 \sim 3$, see Fig. 5.10). With the same annealing process, GASA is the closest curve to the equilibrium. This fact indicates that GASA addresses strongly the attainment of equilibrium compared with the other techniques.

5.6 Summary

This chapter applies mathematical analysis to explore the search mode of the GASA algorithms. With the genetic-thermodynamic combination, we can get a better performance than the traditional search algorithms. For the sake of advanced alteration of the algorithms, these analyses can help know us how to improve and use them. The major search theorems are focused on schema theory and the new simulated phase transition. The simulated phase transition and schema theory are combined with the concept of hyperplane exchange to explain the search mode of either genetic algorithms or GASA algorithms. The search behavior can be simulated to our artificial

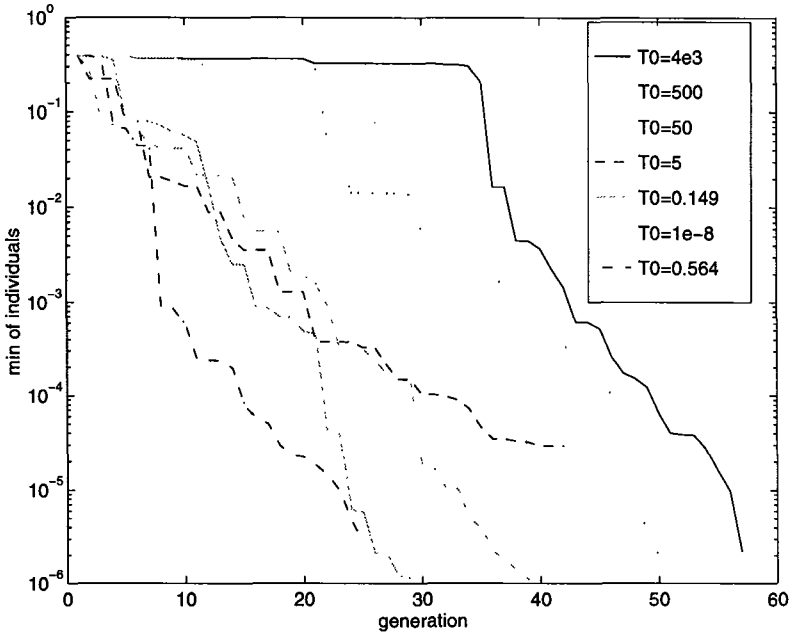


Figure 5.17: *The min of individuals vs. generation*

environment. In this system, the changes of Nature and of temperature are the basic factors that influence the search ability of these algorithms. The search ability can enhance the optimization technology or other related approaches.

With the above analysis, we also can find GASA algorithms have a two-step tuning capability. From genetic algorithms, the final candidate solutions are kept in a uniform oscillation. But for the GASA approaches, the oscillation will decrease because the annealing mechanism is involved. So the genetic operator can be deemed as the "macro-tuning mechanism", and the annealing would be the "micro-tuning mechanism" in this technique.

With the analysis of the equilibrium line, we can easily find the characteristic of these genetic approaches. Thus the new equilibrium theory provides a convenient basis to judge the population turbulence and the optimization path of these approaches.

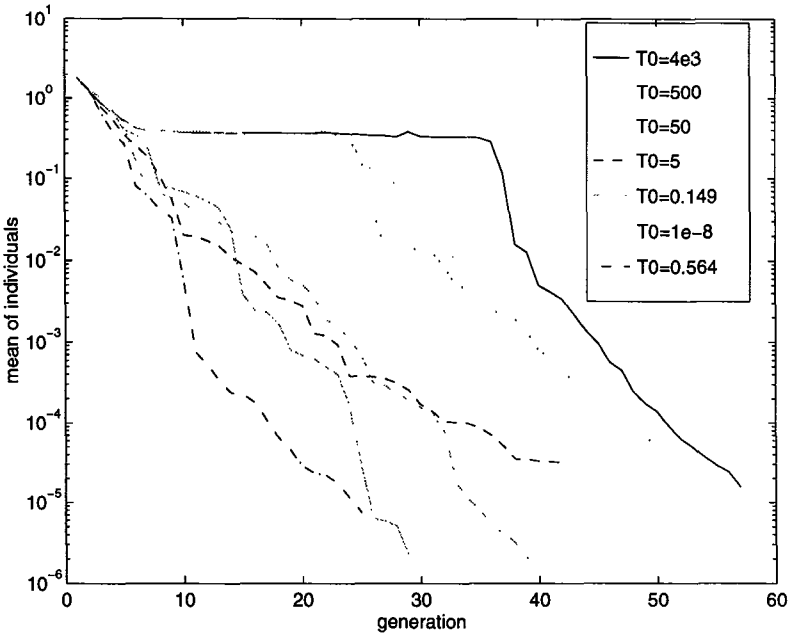


Figure 5.18: *The mean of individuals vs. generation*

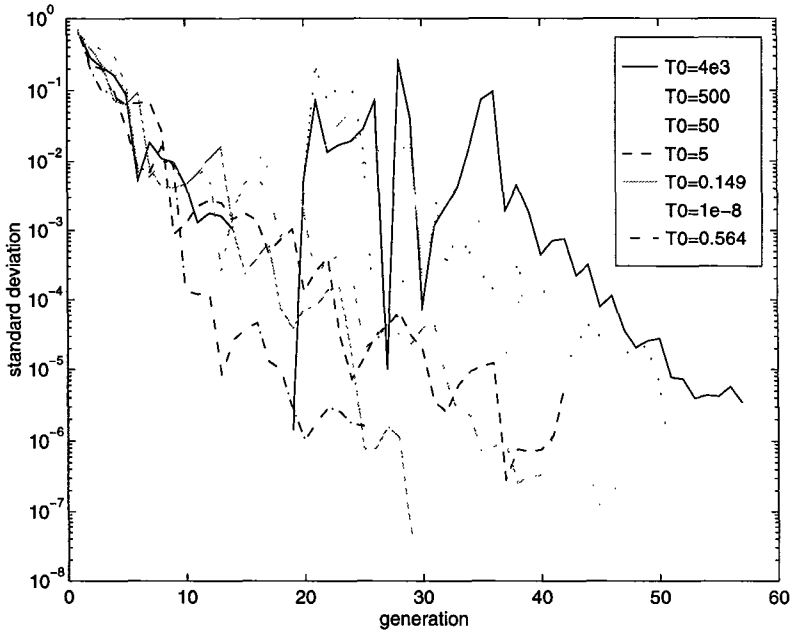


Figure 5.19: *The standard deviation vs. generation*

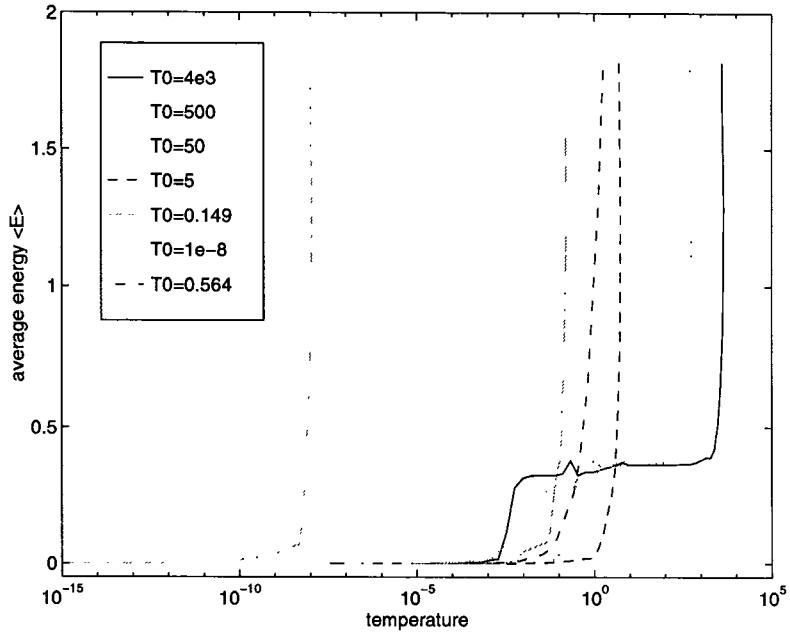


Figure 5.20: *The average energy <E> vs. temperature*

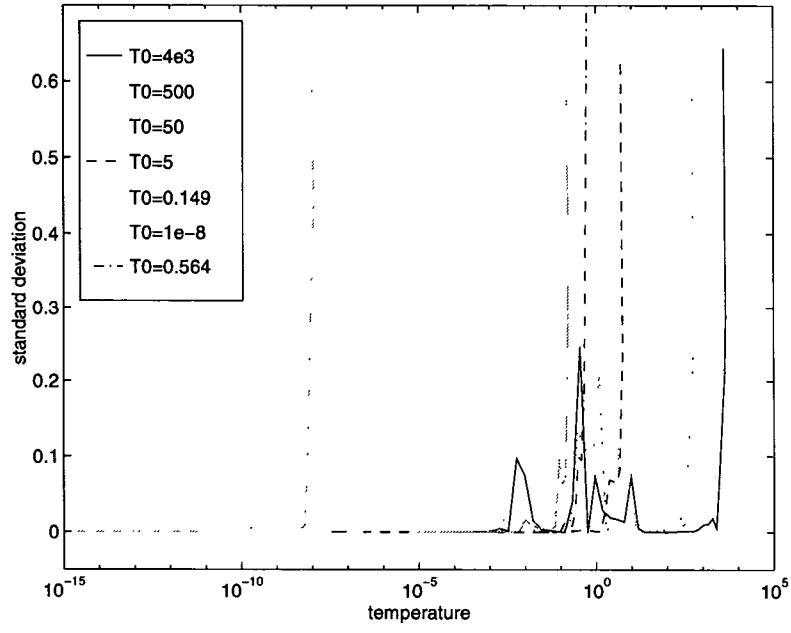


Figure 5.21: *The standard deviation vs. temperature (semilog)*

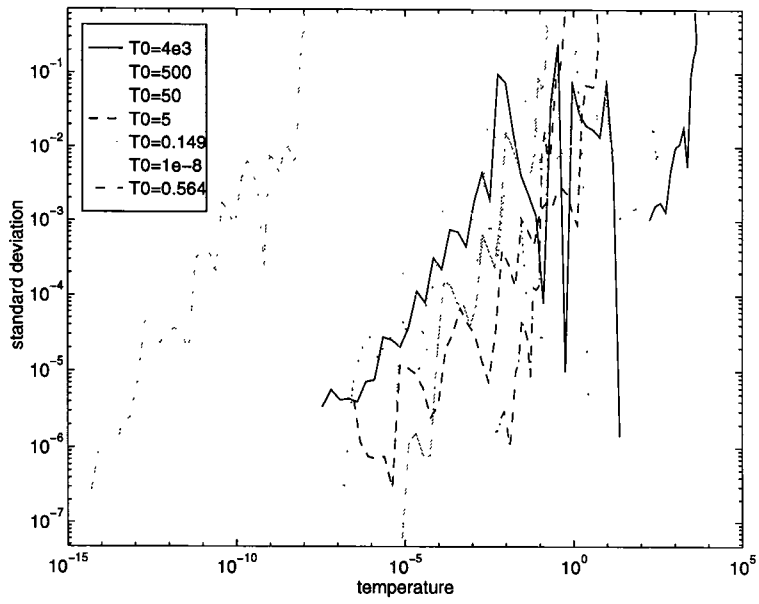


Figure 5.22: *The standard deviation of individuals vs. temperature (log-log)*

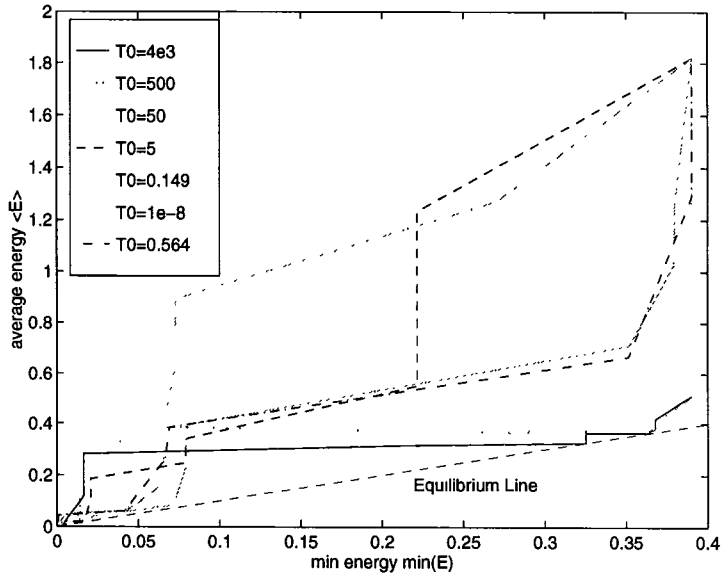


Figure 5.23: *Average energy <E> vs. min energy min(E)*

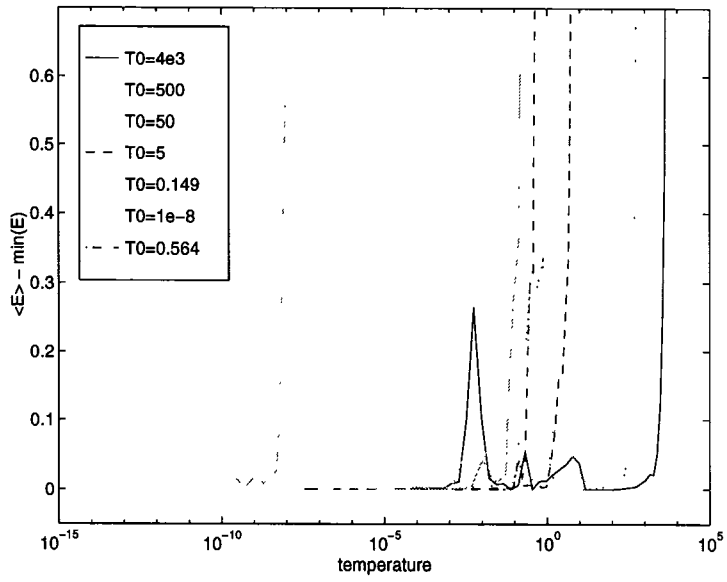


Figure 5.24: *Turbulence "average energy <E> - min energy min(E)" vs. temperature (semilog)*

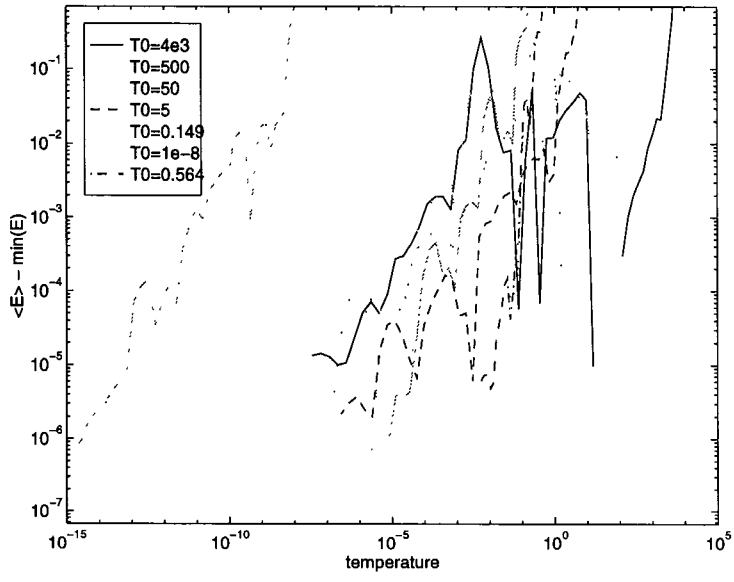


Figure 5.25: *Turbulence "average energy <E> - min energy min(E)" vs. temperature (log-log)*

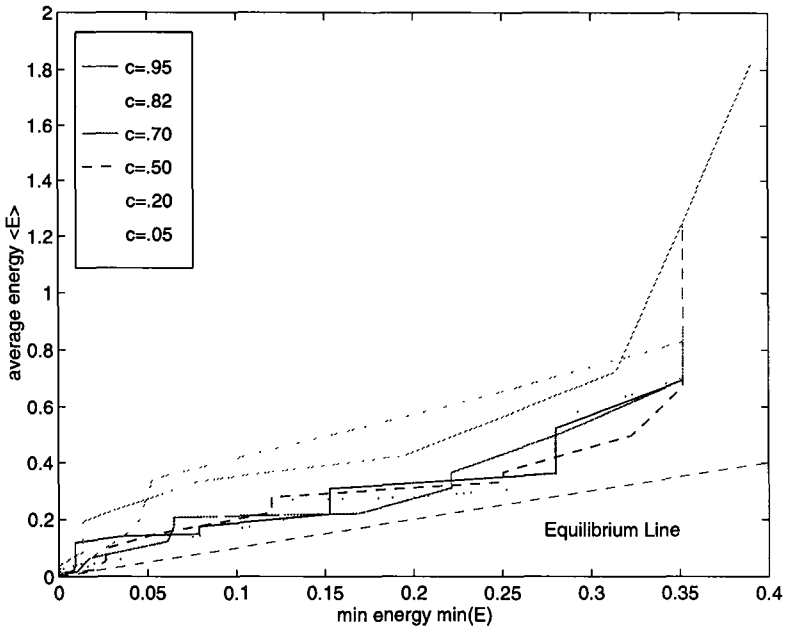


Figure 5.26: *Average energy <E> vs. min energy min(E)*

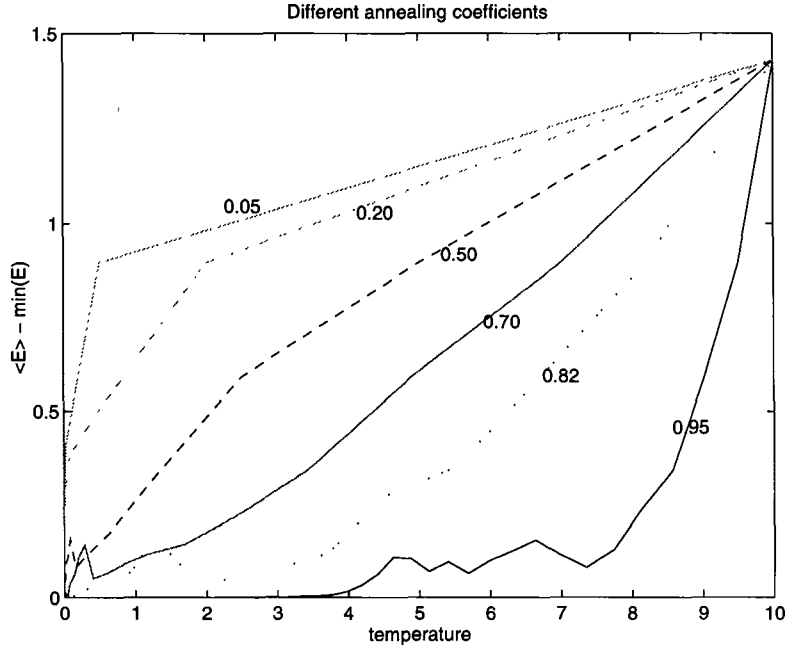


Figure 5.27: *Turbulence "average energy <E> - min energy min(E)" vs. temperature*

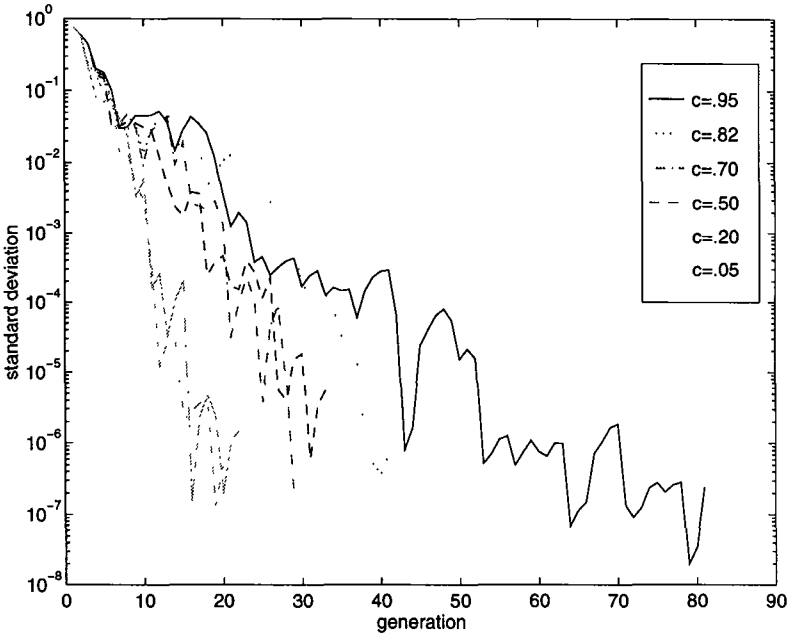


Figure 5.28: *Standard deviation vs. generation*

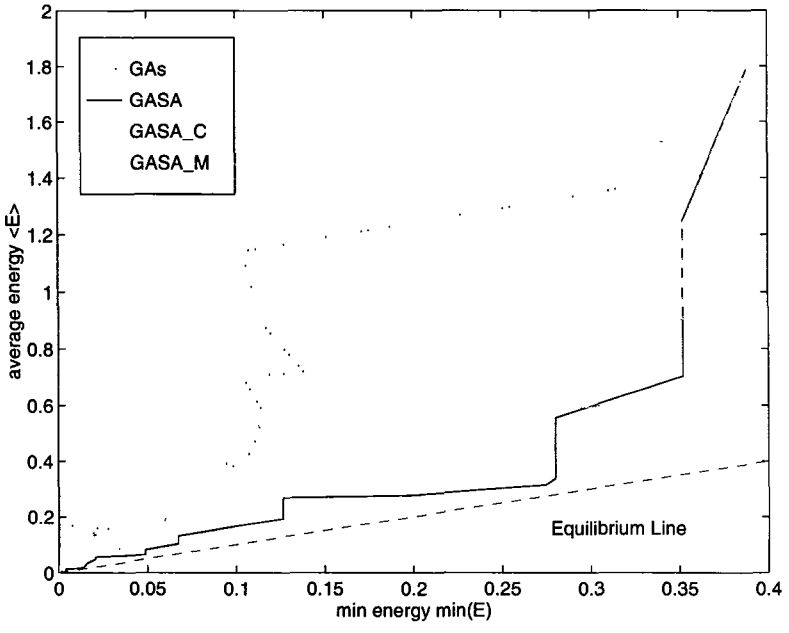


Figure 5.29: *Average energy <E> vs. min energy min(E)*

Chapter 6

Fed-Batch Bioreactor Optimization with the Genetic-Annealing Approach

Large-scale commercial processes such as cell mass and primary metabolite production processes as well as laboratory-scale operations for modeling microbial growth use fed-batch optimization. There is a strong incentive to develop efficient control schemes that would enable rapid start-up and stabilization of the stationary states in continuous bioreactors and the desired state trajectory in fed-batch bioreactors, given the slow dynamics usually associated with microbial growth and the risk of contamination that accompanies it. This study will concentrate on the dynamic control of the biochemical reactor. The fed-batch reactor often has the advantage of lower reactor volume over the two CSTR's in series and CSTR with cell recycle configurations [91, Sines et al.]. For example, this result can lower the cost of wastewater treatment. Also, fed-batch operation can avoid undesired effects which may appear with other types of practical operating modes (e.g., substrate inhibition in batch reactors). Their dynamics are extremely difficult to identify, their observed input/output relationships frequently exhibiting substantial stability punctuated by abrupt instabilities. From the

control engineer's viewpoint, the technical challenges to control fed-batch processes are time-varying parameters in process models, difficulty of measurement of the process variables and more. In fed-batch operation, the feed rate may be changed during the process but no product is removed until the end, and the optimal trajectory of the process variables will be time-varying [8, Åström, 1984].

In general, determining of the feed rate in fed-batch fermentation is a singular control problem [100, 70, Impe et al, 1993;Modak & Lim, 1989], because the control variable (the feed rate) appears linearly in the system of equations and/or in the performance index to be optimized. A number of techniques have been proposed to convert the singular control problem into a nonsingular one by taking variables other than the feed flow rate as the control variable. Therefore, these methods all suffer from problems with respect to practical implementation [99, Van Impe et al, 1992]. The approach most usually applied is the Pontryagin's maximum principle. This method only produces a necessary condition, instead of an "if and only if" condition, for solving the problem [76, 77, Pontryagin et al, 1962;Ramirez et al, 1987]. Thus, the solution derived from the maximum principle may not be the only solution to the optimal control problem. However, based on the direct consideration of state models, population-based algorithms can be used to solve these problems. Genetic algorithms (GAs), provide us a direct approach to apply the state variables themselves instead of their derivatives (from the maximum principle). So these methods can escape the trap of local minima. Moreover, a genetic-annealing approach is proposed to overcome the drawback of genetic algorithms — oscillations [97, Sun et al, 1994]. The test example in that paper illustrates that it is a useful global optimization technique. Here a proposed population-based computation algorithm [97, Sun et al, 1994] which combines genetic algorithms and simulated annealing is applied. It is a search algorithm and can be

used for parameter optimization and machine learning. Its optimization ability is used for optimal control to solve the fed-batch optimization problem.

6.1 Optimal Control and Generalized Genetic-Thermodynamic Algorithm (GASA)

Since the 1980s, genetic algorithms have been applied to solve optimal control problems [42, 57, Greffenstette, 1986;Krishnakumar & Goldberg, 1992]. The main differences between genetic algorithms (GAs) and other search algorithms involve the following aspects,

- GAs search with a population of points and proceed generation by generation instead of with a single point that proceeds point by point.
- GAs work with a binary coding of the parameters instead of the parameters themselves.
- GAs use probability transition rules instead of deterministic transition rules.

The annealing algorithm is also a random search algorithm [55, Kirkpatrick et al, 1983]. This algorithm has the following characteristics compared with other search approaches,

- it can process objective functions with arbitrary degrees of nonlinearities, discontinuities, and stochasticity.
- it can process quite arbitrary boundary conditions and constraints imposed on the objective function.

- it is implemented quite easily with the degree of coding quite minimal relative to other nonlinear optimization algorithms.

The combined genetic-thermodynamic GASA algorithm has both of the characteristics of genetic algorithms and simulated annealing algorithms and has been discussed in Chapter 4. The GASA algorithm is appealing because it allows the population to sustain the most fit individuals in most cases, but in a few cases, due to the probabilistic nature of simulated annealing, highly fit individuals are crossed over or mutated. The less fit individuals are more often subjected to the perturbations of crossover and mutation, and these are the individuals who could benefit the most from such changes.

Hereafter we formulate the optimal control problem and apply this combined algorithm to its solution.

6.1.1 Dynamic optimization

The objective of an optimal control problem is to determine the control policy that will optimize a specific performance criterion, subject to the constraints imposed on the system. For most optimal control problems, the major concern here is input optimization. The input function value can be represented by a decimal scalar of an individual of the population. Thus GAs usually code this variable with a binary set of the declaimed parameter set and obey the natural selection according to the fitness. The fitness of the input function is based on the value of a **performance index** (**function**) [78, 60, Ray,1981;Lewis, 1986]. The state of the dynamic system, $x(t)$, is a function of time t. In order to determine an optimal control policy, $u(t)$, we can formulate the problem as follows.

The system to be controlled is

$$\dot{x} = f(x, u, t), \;\; x(t_0) = x_0 \tag{6.1}$$

and is to follow a path $x(t)$, such that the performance function is

$$J(u) = \int_{t_0}^{t_f} F(x, u, t)dt \tag{6.2}$$

which will be extremized. The function $x(t)$ is continuously differentiable on the interval $[t_0, t_f]$, where t_0 and t_f are the initial and final time, respectively. Function $F(t)$ is also continuous in $x(t)$. The system equations become the constraints of the performance function.

6.2 Fed-Batch Bioreactor Optimization

The development of the general characteristics of the optimal rate profiles for a class of fed-batch fermentation processes was presented in fermentation by Lim et al [65, Lim et al, 1987]. The objective of fed-batch optimization is to determine the optimal feed rate profile which will maximize a given profit function. The usual form of a unstructured mathematical model for cell production is given by the following set of mass balance equations.

$$\frac{dS}{dt} = -\sigma X + s_F u \tag{6.3}$$

$$\frac{dX}{dt} = \mu X \tag{6.4}$$

$$\frac{dV}{dt} = u \tag{6.5}$$

where $X = x \cdot V$ and $S = s \cdot V$. X, S are the mass of the cell and substrate, u is the feed rate and V is the reactor volume, μ, σ are the specific rate of cell growth and the substrate consumption rate, respectively. $\mu = \frac{1}{X}\frac{dX}{dt}$ and $\sigma = \mu/Y$ and Y is the yield

coefficient. The initial conditions are assumed to be specified. Physical constraints must be imposed on the final fermentor volume and the feed rate. The constraint on volume is:

$$V(t_f) \leq V_f \tag{6.6}$$

, where t_f denotes the final reaction (fermentation) time. In practice, the pump feeding capacity $u(t)$ is bounded by

$$0 = u_{min} \leq u(t) \leq u_{max}, u_{max} \ \ given \tag{6.7}$$

The profit (performance) function to be maximized is the final outcome of the fermentation, so Eq. (6.2) can be represented by the functional

$$J = max(X(t_f)) = -min(x_f \cdot t_f) \tag{6.8}$$

where the final time is fixed. Thus, the problem is to determine the profile of the feed rate that maximizes the profit function Eq. (6.8).

6.3 Dynamic Optimization of Fed-Batch Bioreactor with GASA algorithm

The combined computational algorithm (GASA) of genetic algorithms and simulated annealing provides an efficient approach for optimization with high speed and precision [97, Sun et al, 1994]. The formulation of GASA computational algorithm can be represented by a nine-tuple (nine key operators) as in Eq. (5.15). The detail design procedure is in Chapter 5. The control variable in this problem is the feed rate $u(t)$.

6.3.1 GASA algorithm and fed-batch optimization

At the start, we can initialize the population with size λ with length l for each individual for feed rate u which is divided into several time intervals. For example, for individual i with length l in the population $P(t)$ can be $\nu_i(t_1), \nu_i(t_2), \cdots, \nu_i(t_n)$ ($\vec{\nu}$ is a binary set), n is the the number of time intervals and $t_n = t_f$. This parameter set of ν elements can be decoded into $u = decode(\nu)$ and then substituted for u into Eq. (6.5). The performance index J_i is given by Eq. (6.8). The performance index is in terms of the raw fitness and converted into the selection probability $p_i = J_i / \sum_{j=1}^{n} J_j$ for selection. Then we have to set up the temperature to anneal the individuals in the initial population $P(0)$. The best initial temperature T_0 is found by solving Eq. (5.32). The annealing coefficient c is usually set between $[0.7, 0.99]$. After selection and reproduction with the probability p_i, the existing individuals will undergo the genetic operations. We can determine which individuals are to be crossed-over or mutated according the probability $\pi_T = exp(-\triangle J/T)$ where $\triangle J = J - J_{min}$ and T is the current temperature. π_T is also called the "threshold probability". Individuals with $\pi_T > random[0, 1)$ are kept in the gene pool; otherwise, the unqualified individuals are crossed-over to a randomly selected site where their binary bits are exchanged. Then, the crossed-over individuals will be put into the gene pool. We can process the individuals in the gene pool with the other genetic operator — mutation. Mutation uses the same procedure as crossover, the only difference is that mutation occurs at the randomly selected site. The stop criterion is $|J_{new} - J_{old}| \leq \zeta$.

The procedures of the GASA feedback controller is illustrated in Fig. 6.1. The performance (J) evaluation for each individual (J_i) in the populations is calculated based on the equations in Eq. (6.5). Then the performance J is converted into

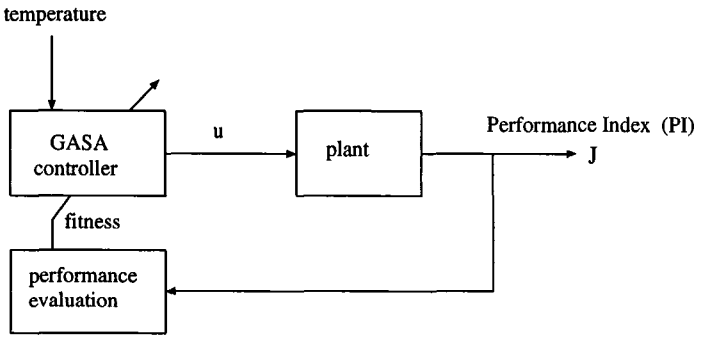


Figure 6.1: *An Illustration of GASA Feedback Controller*

the fitness. The genetic operation procedures, such as selection, reproduction are processed according to the fitness. The other parameter, the temperature T, is set up from the annealing schedule. Temperature identifies the individuals which need to be crossed-over or mutated. The final selected individuals of u are the feed rate to the bioreactor. Repeating these procedures, we can find the best cell mass generation by generation. The procedure does not stop until the criterion ζ is met.

6.4 Results and Discussion

A fed-batch culture for cell mass production is described by Eq. (6.3) to Eq. (6.5). The objective is to maximize the amount of cell mass represented in Eq. (6.8), where t_f is fixed. We consider a variable yield $\sigma = \mu(s)/Y(s)$ [70, Modak & Lim, 1989] as follows.

$$\mu(s) = \frac{0.504s(1 - 0.0204s)}{0.00849 + s + 0.0406s^2} \tag{6.9}$$

$$Y(s) = \frac{0.383(1 - 0.0204s)}{1 + 0.296s - 0.00501s^2} \tag{6.10}$$

where s is the substrate concentration. The specific growth rate is substrate inhibited, with the maximum rate of 0.482 h^{-1} (see Fig. 6.2) at $s = 0.37$. The yield Y is

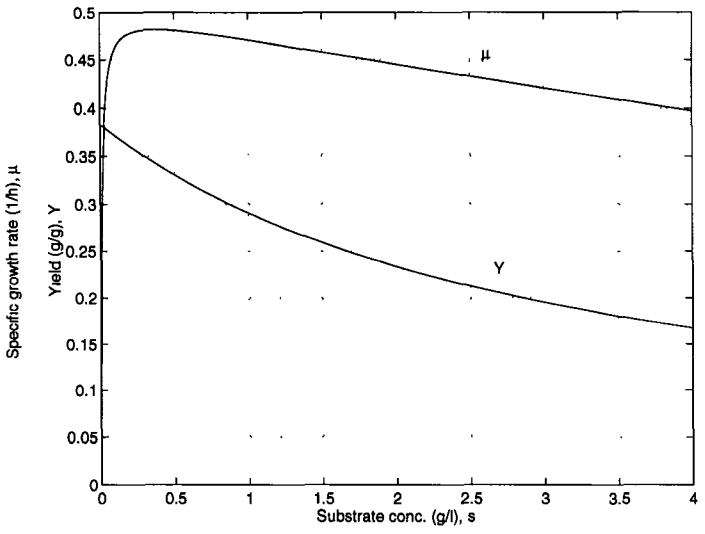


Figure 6.2: *Specific growth rate μ and yield Y vs. substrate concentration s*

monotonically decreasing as is shown in Fig. 6.2. The initial conditions applied here are $X_0 = 1g, S_0 = 1g, V_0 = 1l$. The feed concentration s_F is 10 g/l. The maximum volume (V_{max}) and maximum feed rate (u_{max}) are $5l$ and $4l/h$, respectively.

The feed rate profile derived from the maximum principle is not unique, because this analysis is only from a necessary condition. We call it **multiple-to-one** mathematically. The complexity of this problem can be found from the mathematical analysis. Since the first derivative of the Hamiltonian operator equals to zero, it only means a minimum or maximum exists. Even though the second derivative is zero, the global properties of the function are still unknown. We can use a simple simulation result to explain its non-uniqueness. Modak & Lim used a simple nonsingular control approach to optimize a fed-batch fermentation [70, Modak & Lim, 1989]. A ramp input function is applied as their initial guess. Here we apply 61 ramp input patterns to analyze the above problem. Fig. 6.3 illustrates the relationship between biomass, feed rate and time. The highest value occurs at the edge where $t = t_f$. The contour of

Fig. 6.3 is plotted in Fig. 6.4. Fig. 6.4 enhances the last point for no global extrema exist in the plateau of the picture. The feed rate profiles are shown in Fig. 6.5. The pump is turned off whenever the maximum volume is reached. All of the maximum volumes of these input patterns are less than or equal to V_{max} (see Fig. 6.6). The substrate concentration of the different patterns can be found from Fig. 6.7. When the feed rate is pretty low, the substrate concentration decreases fast. Fig. 6.8 shows some of the maximum biomass occur at the final time with different inputs. The theoretical maximum is 16.70 here. In practice, all biomass should be smaller than this theoretical value. The flow profile and volume of the cell in the bioreactor are described in Fig. 6.5 and Fig. 6.6, respectively.

Now we apply the classic genetic algorithms and the GASA algorithm to analyze the dynamic optimization of fed-batch bioprocess. The population size in this application is 50, and the length of the chromosome is 20. The time domain is divided into 50 intervals. For GASA, the initial temperature is 6 and the annealing coefficient is 0.95. The simulation is run with a SUN Sparc-10 workstation.

With GAs, Fig. 6.9(a) shows the individuals in the final generation (106). Six big gaps can be found in Fig. 6.9(b). This shows the significant oscillations which can occur in this singular control problem. This condition is also reflected in Fig. 6.10(b). Fig. 6.10(a) is the best fitness in generations. Fig. 6.10(c) shows the mean and min objective values versus the function calls.

Fig. 6.11 to Fig. 6.13 are the profiles for generation 106 with GAs from the best individuals in the population. The feed rate shows piecewise changes in Fig. 6.11. When time is close to 4.9, the pump is turned down because the bioreactor is full. The substrate concentration decreases when the feed is shut off and approaches zero at t_f. The cell mass is increasing and reaches its maximum at final time. Fig. 6.13 is the

Table 6.1: *The basic units in different algorithms*

Method	Cell Production
GAs	16.2900
GASA	16.2933
T.M.P	16.0800

product profile with the scale changes. The only product is the biomass. So we can find the best product of biomass is 16.2900 with GAs.

Fig.6.14~Fig. 6.18 shows the analysis with the GASA algorithm which optimizes the input profile. Fig. 6.14(a) shows the individuals in the final generation. Fig. 6.14(b) illustrates the best and mean objective values from the performance function $J = -X(t_f)$. The best value is 16.2933 for GASA in generation 133. A gap appears between mean and best. This situation also occurs around generation 45, 97 and 105, but it is not so serious as in GAs. Fig. 6.15(a) is the best fitness in generations. The standard deviation of individuals through generations are illustrated in Fig. 6.15(b). Fig. 6.15(c) shows the mean and min objective values versus the function calls.

The pictures, Fig. 6.16~Fig. 6.18, are the profiles for generation 133 with GASA. The feed rate shows piecewise changes in Fig. 6.16. When time is close to 5.6, the pump is turned off because the bioreactor is full. The substrate concentration decreases with the feed and approaches to zero at t_f. The cell mass is increasing when the feed is off and reaches its maximum at the final time. Fig. 6.18 is the product profile with the scale changed. The only one product is the biomass here. We find the best product of biomass is 16.2933 with GASA. The different cell mass values are compared in Table 6.1 with the maximum principle (T.M.P), GAs and GASA. Both of the population-based methods have a higher value for cell mass.

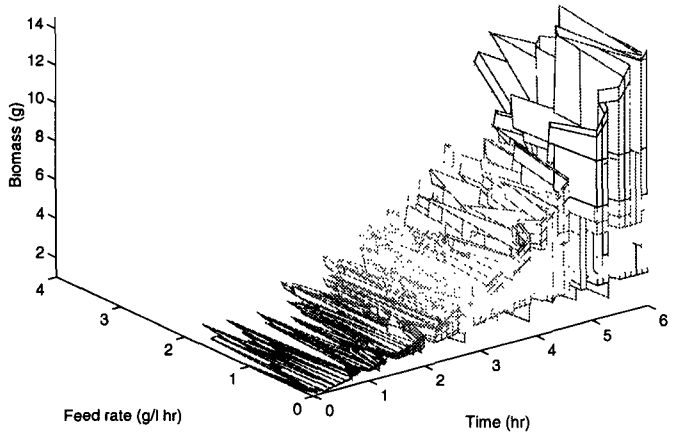


Figure 6.3: *The mesh plot for fed-batch optimization with different ramp input patterns*

In this result, GASA uses more generations (133) than that in GAs (106). It indicates that we still can improve GASA with several aspects, i.e. the annealing schedule. Thus, the population-based search approach provides us a good method to analyze dynamic nonlinear optimization as in fed-batch fermentation. Moreover, this method can be more complex by applying extra constraints or more bounds, and the results will be found faster than the traditional hill-climbing method.

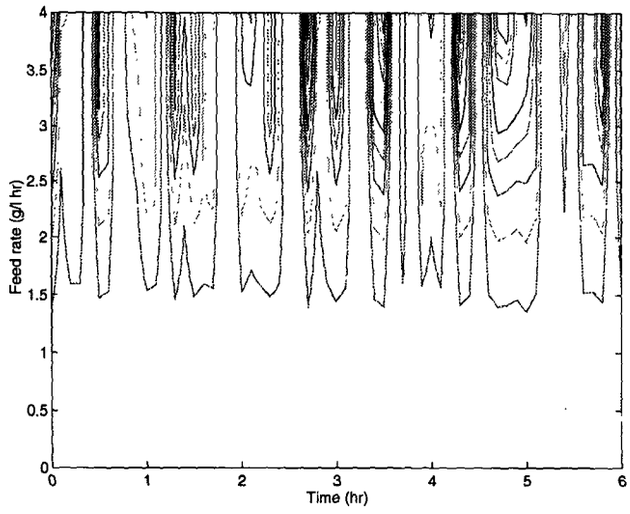


Figure 6.4: *The contour plot for fed-batch optimization with different ramp feed patterns*

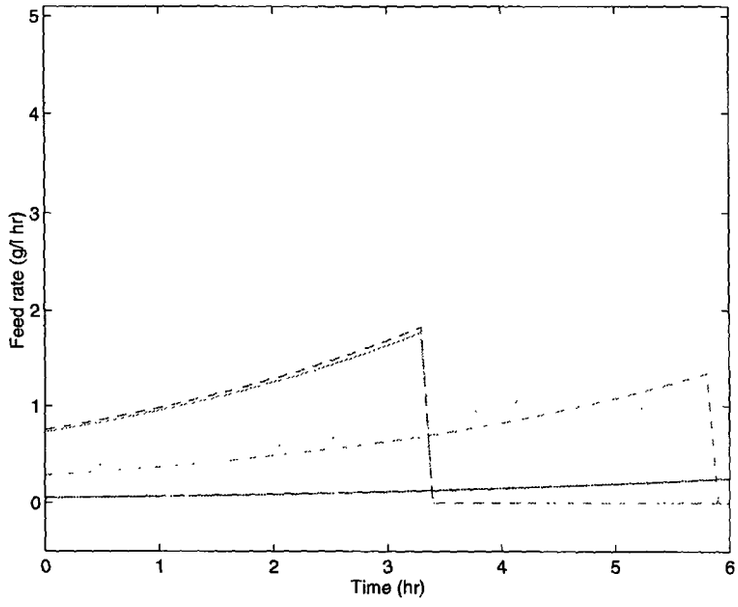


Figure 6.5: *Different feed patterns*

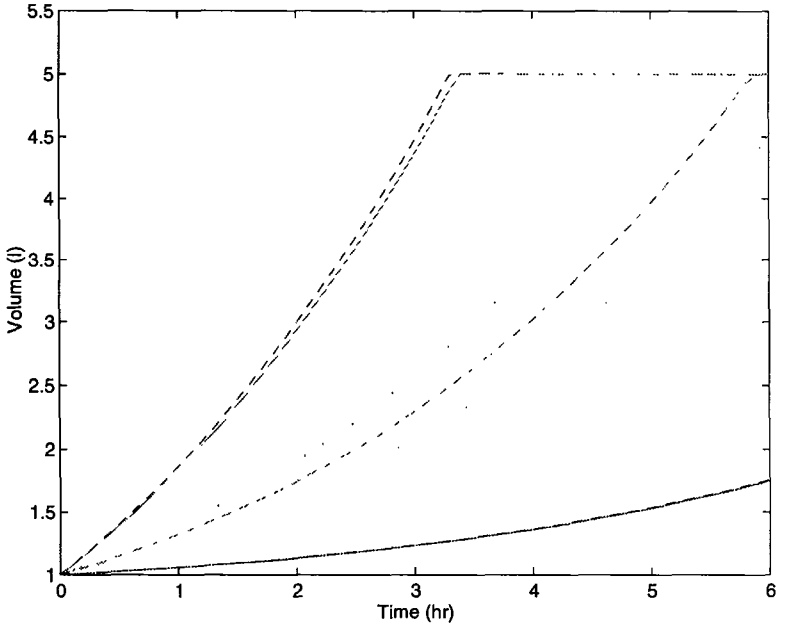


Figure 6.6: *The volume profile*

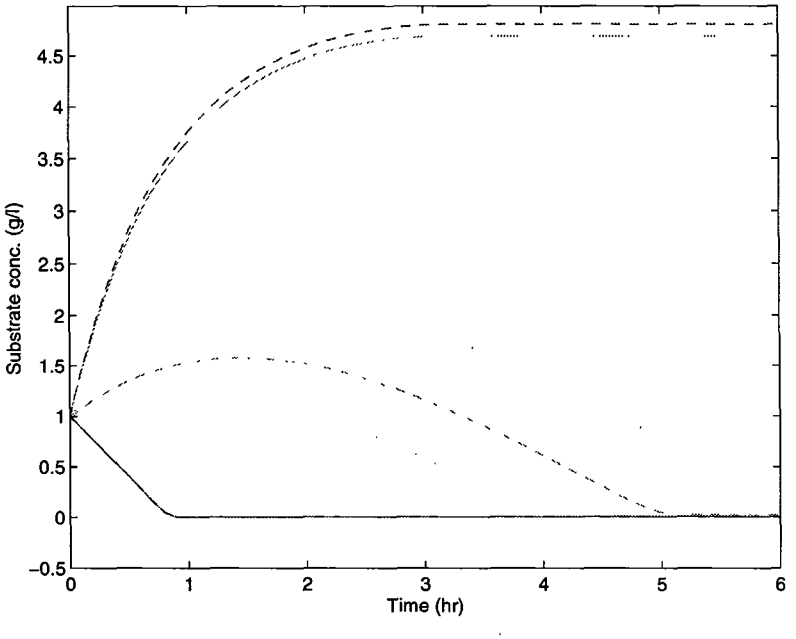


Figure 6.7: *The substrate conc. profile*

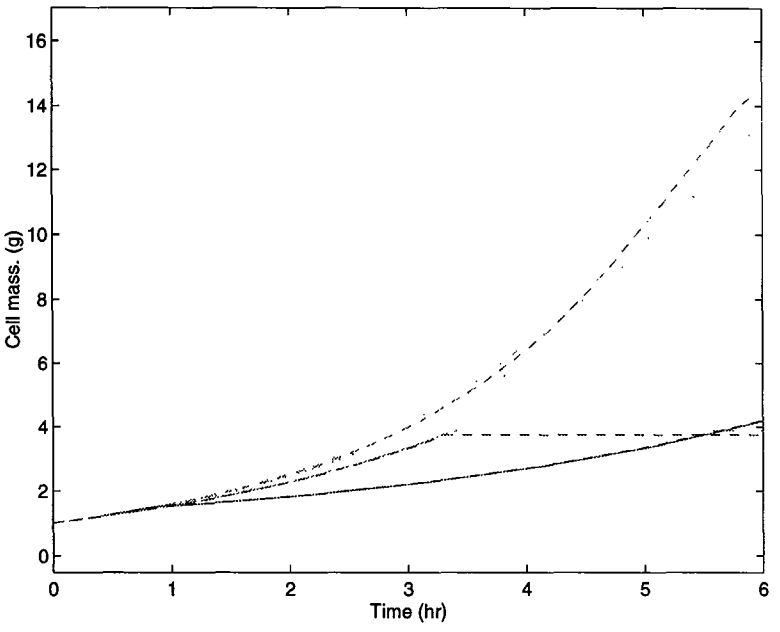
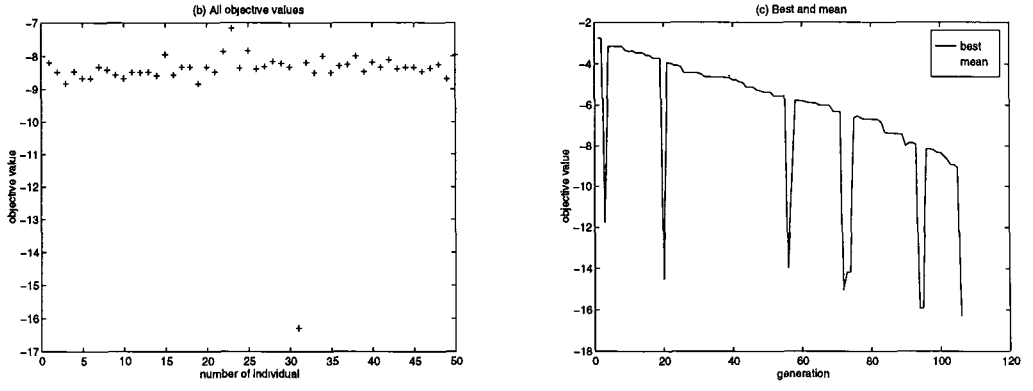


Figure 6.8: *The cell mass profile*



(a) (b)

Figure 6.9: *An Interpretation of GAs operation in generations: (a) The objective values of all individuals in generation 106. (b) Best and mean objective values of individuals in generations.*

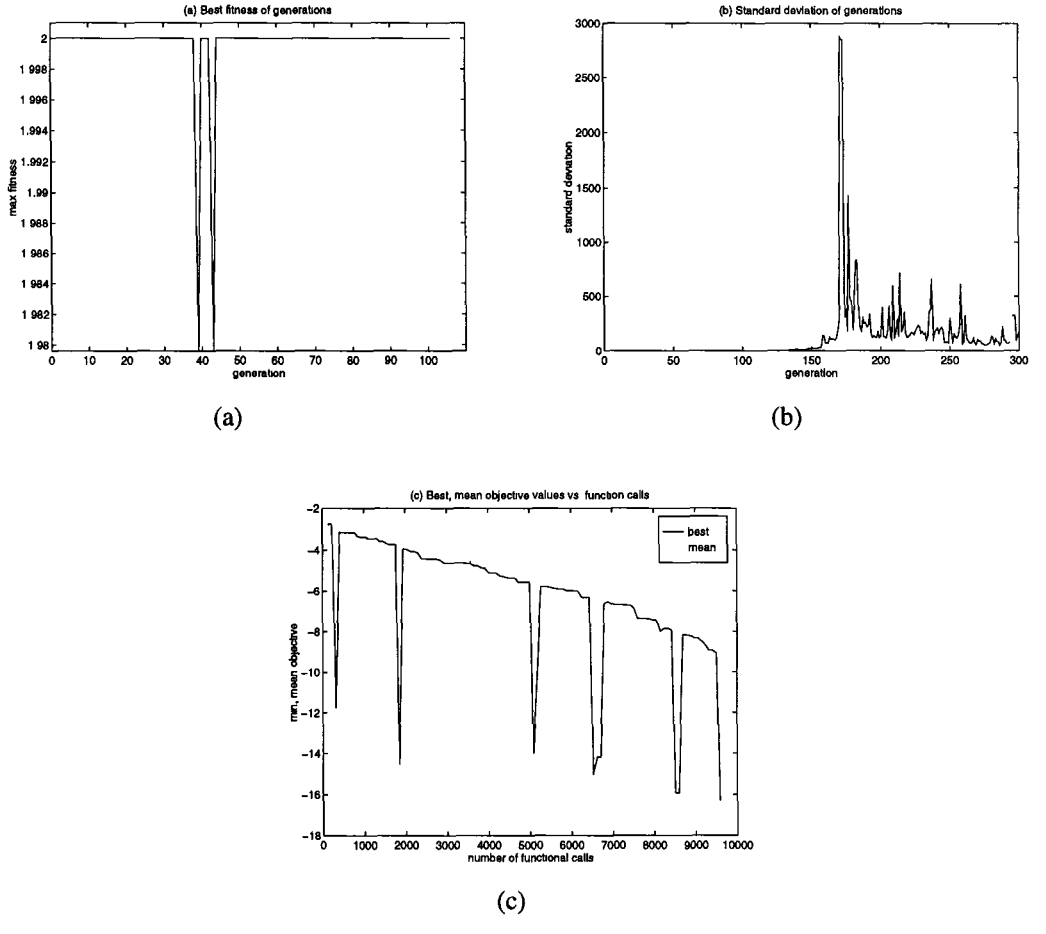


Figure 6.10: *Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.*

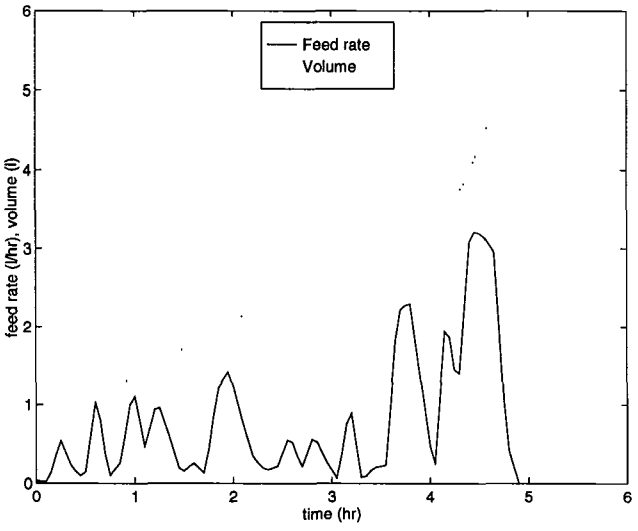


Figure 6.11: *The profiles of volume and feed rate with GAs*

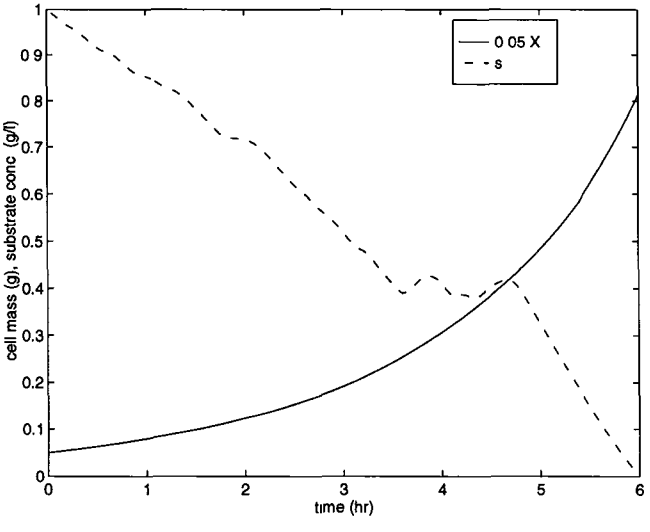


Figure 6.12: *The profiles of substrate conc. and cell mass with GAs*

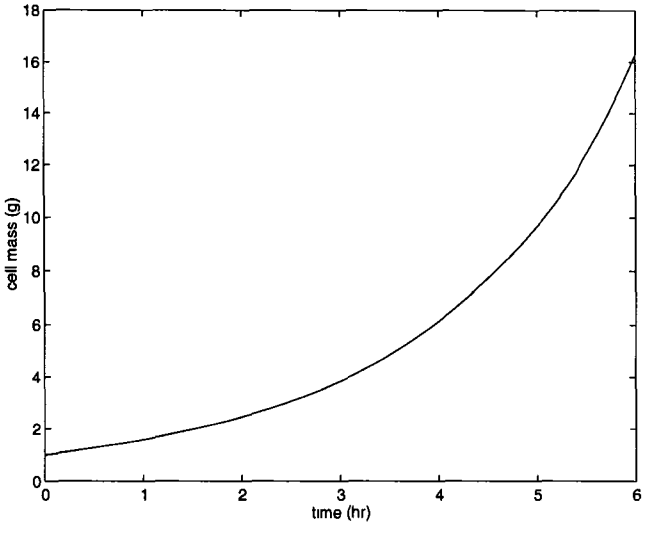
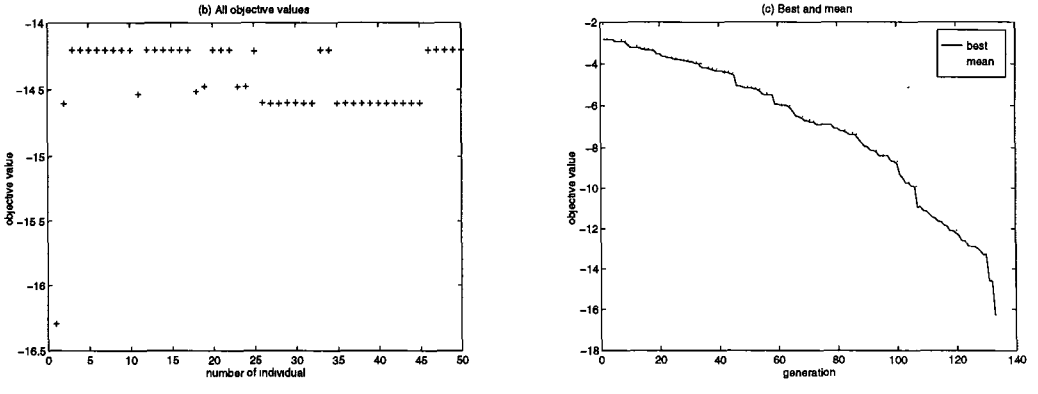
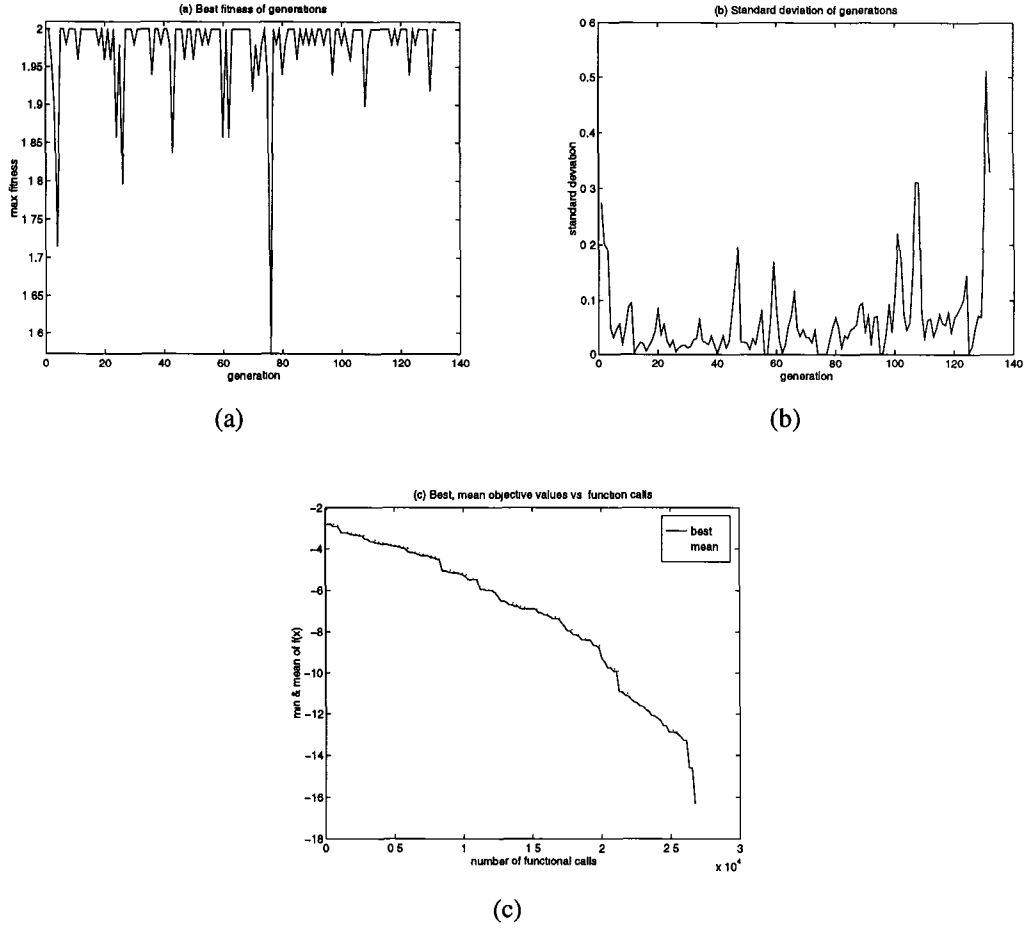


Figure 6.13: *The product profile with GAs*



(a) (b)

Figure 6.14: *An Interpretation of GASA operation in generations: (a) The objective values of all individuals in generation 133. (b) Best and mean objective values of individuals in generations.*



(a) (b)

(c)

Figure 6.15: *Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls.*

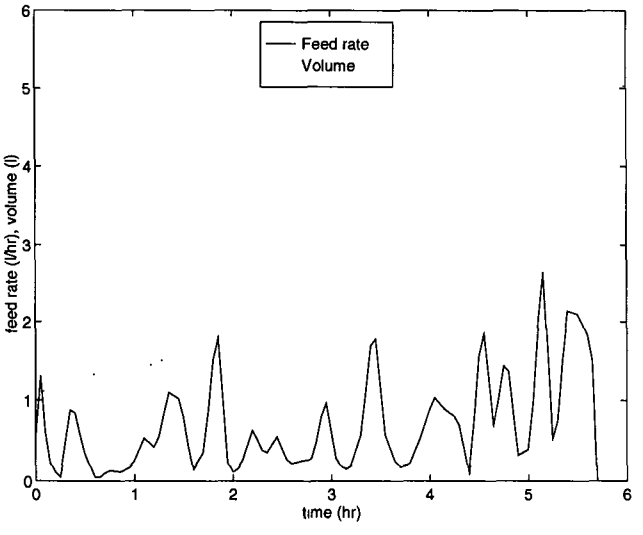


Figure 6.16: *The profiles of volume and feed rate with GASA*

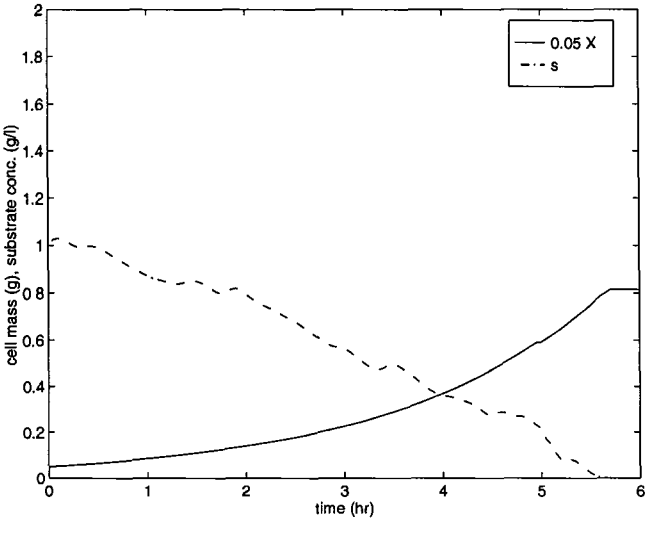


Figure 6.17: *The profiles of substrate conc. and cell mass with GASA*

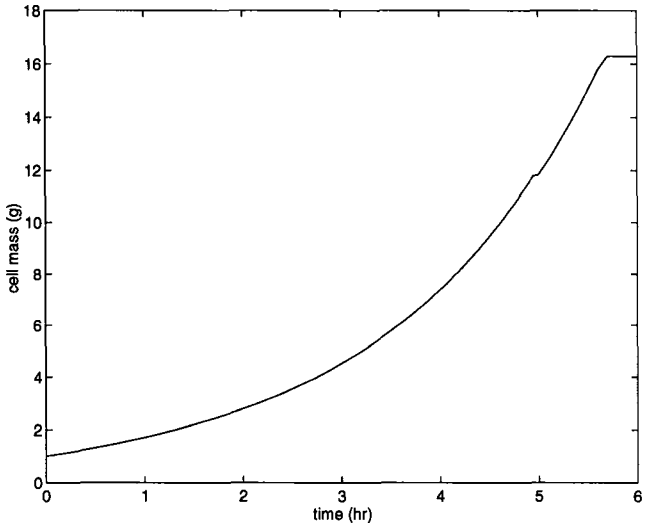


Figure 6.18: *The product profile with GASA*

Chapter 7

Control of a pH Plant Using Genetic-Annealing Approach

Population-based representations are becoming very popular due to the their parallel processing paradigm, their capabilities to represent nonlinear functions and to be adaptive [53, 93, 108, Irwin, 1992;Sistu & Bequette, 1991;Whitley & Starkweather, 1990]. Based on Darwin's survival-of-the-fittest strategy, genetic algorithms (GAs) have become a fast-growing field of research. GAs are self-adaptive search algorithms, based on the principles of genetics and natural selection, which, in control system engineering, can be used because of their optimization ability [28, Filho et al, 1994]. Because of the nonconvergence of GAs in some problems [83, Rudolph, 1994], the combined genetic-annealing approach (GASA) is proposed for nonlinear optimization problems [97, Sun et al, 1994] to improve the convergence of GAs using another stochastic optimization technique — simulated annealing [55, Kirkpatric et al, 1983]. The application used here is a pH plant control. As we know, control of pH is a difficult one because of the severe nonlinearity [90, 43, 15, Shinskey, 1988;Gustaffson, 1985;Buchholt & Kummel, 1979]. It is known that in case the plant involves a steady state pH change towards the neutrality point effected by feeding a chemical reagent, the output pH is highly sensitive to various of the mass flow of this controlling reagent. In

such cases, high demands are set on the static accuracy of the control components, and it is often hard or impossible to meet these demands with conventional schemes using feedback control [72, Orava & Niemi, 1974]. In process control, the Ziegler-Nichols (ZN) ultimate-cycle tuning [110, Ziegler & Nichols, 1942] is the most popular method to fine-tune the parameters of classical PID controllers. The ZN method is sometimes laborious, especially for processes with strong nonlinearity, because in fact the PID values obtained through the ZN method usually requires manual tuning. Thus, a new control tuning scheme for the PID controller is proposed here. This tuning ability of the population-based approaches (GAs, GASA) are derived from their optimization ability.

7.1 Genetic-Annealing Approach

The genetic-annealing approach is based on the combination of genetic algorithms and simulated annealing [97, Sun et al, 1994]. The behavior and characteristics of the combined genetic-annealing approach were explored in Chapter 5 with **schema theory** and thermodynamics. First, this approach can be described by discussing genetic algorithms and simulated annealing individually.

Genetic algorithms are stochastic global and population-based search algorithms that determine the locations and values of a set of points in the domain space. The criterion for which new points are generated or old points are discarded is a function of the existing population. A simple genetic algorithm (SGA) is described by Goldberg [38, Goldberg, 1989]. Individuals encode a set of decision variables by concatenating them in a bit string according to their fitness just like Nature works with chromosomes. The initial population is generated randomly and the population size is kept constant

throughout the process.

- Genetic Reproduction

  The fitness f_i is calculated from the objective function. Those states not selected are culled from the population. Because the average fitness of the population is defined as $\bar{f} = \sum_{j=1}^{n} f_j/n$, if there are $b_i(t)$ copies of an individual i at time t, the new population will have $b_i(t+1) = b_i(t) f_i/\bar{f}$ copies of f_i. The effect of this reproduction scheme is that above average performing individuals reproduce, replacing poorly performing individuals.

- Genetic Recombination

  The recombination operator used here is single-point crossover. Individuals are paired with a highly probability that crossover will take place. Thus, a crossover point is selected at random. For example, the two individuals {0 0|0 1 0} and {1 0 |1 0 1} are crossed-over at the location "|". After recombination, their offspring strings will be {0 0 1 0 1} and {1 0 0 1 0}. The mutation operator, occasionally flipping random bits in the population, causes individuals to sample points that span the space with precision defined by the encoding method.

The other technique to be used is **simulated annealing**. It is based on the Boltzmann distribution in statistical mechanics. The method was developed to be used with a nonconvex objective function. The approach of simulated annealing consists of three functional relationships.

1. Probability density of state-space of D parameters with $x = \{x_i, i = 1, 2, \cdots D\}$.

2. Probability density for acceptance of new cost-function given the just previous value.

3. Annealing schedule.

The combined approach (GASA) from the above two algorithms is based on the concept of "division" by energy. The new individuals are generated by genetic operators instead of item 1 of simulated annealing. In fact, item 2 enhances the division of the group of individuals according to their own energy (objective value). High energy individuals expense a greater number of genetic operations. On the contrary, the low-energy individuals are at the more stable states compared to the higher-energy ones. The algorithm design is illustrated in Chapter 4. In simple terms, the formulation of GASA computational algorithm can be represented by a nine-tuple set of operation as in Eq. (5.15). The optimized variables in this PID control problem are the control parameters (k_p, k_i, k_d).

7.2 Problem Formulation

Let us consider a system which is given by the differential equation:

$$\frac{dx}{dt} = G(x, u) \tag{7.1}$$

where $x(0)$ is given, x is an $(n \times 1)$ state vector and u is an $(m \times 1)$ control vector bounded by

$$LB \leq u(t) \leq UB \tag{7.2}$$

where LB and UB are the bounds for $u(t)$. It is derived to choose the control variable u such that the desired state is achieved in the operation time. The control variable u can be obtained by the control algorithm which is described later.

7.2.1 A pH plant model

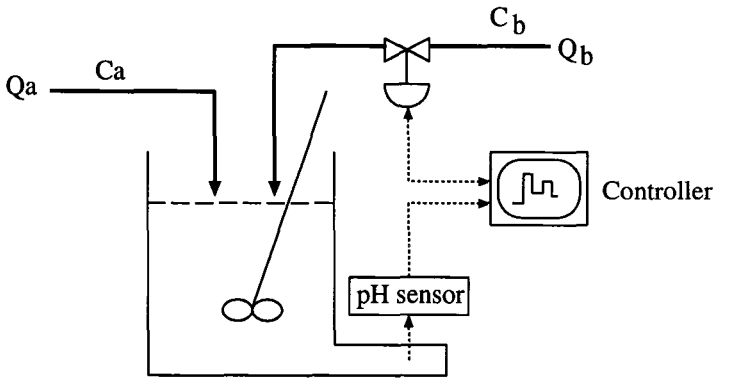


Figure 7.1: *A neutralization process*

The control problem is a neutralization process which uses a strong acid and strong base in a well stirred tank, and a pH measuring sensor which is illustrated Fig. 7.1. The pH control problem was solved by several different methods [15, 72, Buchholt & Kummel, 1979;Orava & Niemi, 1974]. The control objective is to maintain the pH value of the reactor at the point of neutralization. The manipulating variable is the flow of base. Mixing in the tank is perfect so the fluid phase is homogeneous. If the equilibrium of H^+ and OH^- is assumed to hold at any time, and a variable Q is used to denote the difference of concentration $[H^+]$ and $[OH^-]$. We find:

$$\begin{aligned} Q &= [H^+] - [OH^-] \\ &= [H^+] - \frac{K_a}{[H^+]} \end{aligned} \tag{7.3}$$

where $K_a = 10^{-14}(mol/l)$ is the equilibrium constant at 25^o. By solving the quadratic equation in Eq. (7.4) for $[H^+]$ and choosing the positive root, we have:

$$[H^+] = 0.5[Q + (Q^2 + 4 \times K_a)^{1/2}] \tag{7.4}$$

Under ideal conditions, the plant model can be described by the following quasi-linear equation [15, Buchholt & Kummel, 1979]:

$$\frac{dQ}{dt} = -(Q_a + Q_b)\frac{Q}{V} + (C_a Q_a - C_b Q_b)/V \tag{7.5}$$

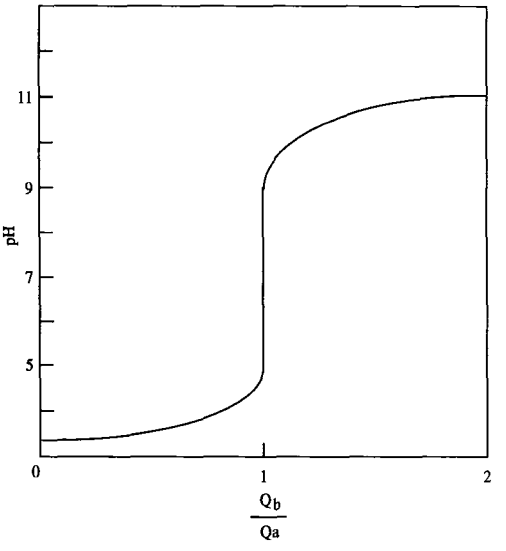


Figure 7.2: *Titration curve for neutralization of strong acid* ($10^{-3}mol/l$) *and strong base* ($10^{-3}mol/l$)

where C_a, Q_a and C_b, Q_b are concentration and flow of acid and base, respectively. V is the tank volume. The effect of dissociation of water is neglected because of $C_a >> 10^{-7}$ and $C_b >> 10^{-7}$. Q is zero at the neutrality point. Q cannot be measured directly, but can be converted from the measured pH signal. The pH value and $[H^+]$ are calculated by

$$pH = -log_{10}[H^+] \tag{7.6}$$

We use a strong acid and a strong base for the control input in Fig. 7.2. The model from Eq. (7.5) and Eq. (7.6) is highly nonlinear.

From an examination of Fig. 7.2, we see that a small fluctuation of Q_b/Q_a may cause a large deviation of pH around pH=5$\sim$9. This difficulty makes the traditional control strategies very sensitive, especially for PI/PID control. A very fine tuning for proportional gain is required for a satisfactory performance.

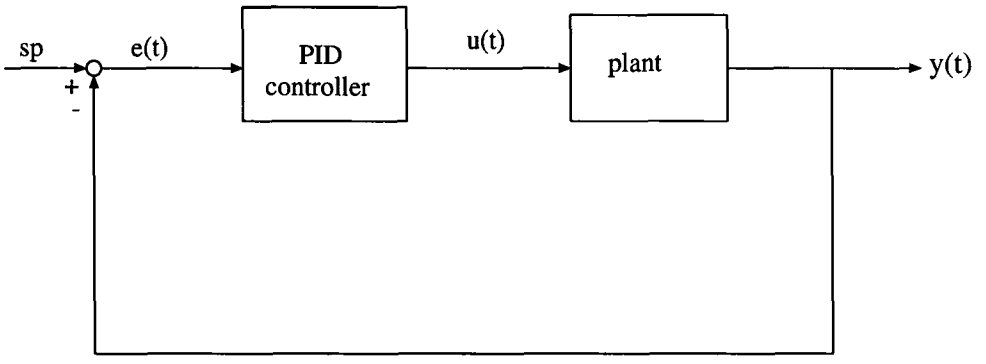


Figure 7.3: *A classical PID control system*

7.3 Control Algorithms

In this section, the development of control strategies for the neutralization process is described. These are a standard PID control, a dynamic GA PID controller, and our effective GASA PID controller. In each of the algorithms described the influent base stream flow is the manipulated variable.

7.3.1 PID controller

The continuous form of the PID control algorithm is used in this study. It can be described as follows.

$$Q_{b,t} = K_{PID}\left(e(t) + \frac{1}{\tau_I}\int^t e(s)ds + \tau_D\frac{de(t)}{dt}\right) \tag{7.7}$$

where K_{PID} is the gain of the PID controller, τ_I and τ_D are the integral and derivative constants, respectively. Fig. 7.3 shows the typical PID control system, where sp is the set point ($= pH_{sp}$), y is the system output (pH), and the error $e(t) = pH_t - pH_{sp}$. $u(t)$ is the control output and the input to the plant.

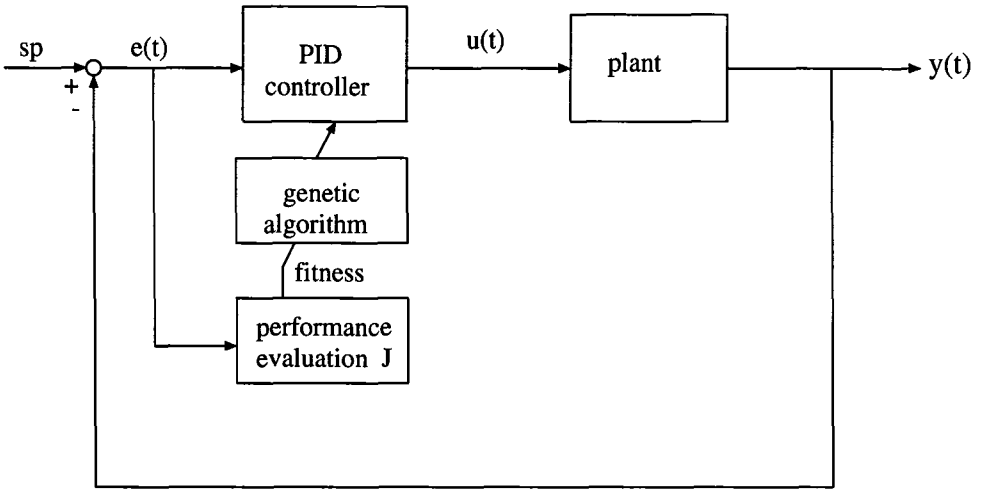


Figure 7.4: *A PID + GAs control system*

7.4 Control Tuning of the Plant

The Ziegler-Nichols (ZN) tuning settings, which are based on the ultimate-cycle , are given as below [103, Wang & Kwok, 1993].

$$K_P = 0.6K_u, \tau_I = \frac{\tau_u}{2}, \tau_D = \frac{\tau_u}{8} \tag{7.8}$$

where K_u and τ_u are the ultimate gain and period, respectively.

The performance function (PF), J, is used to find the optimal combination of PID parameters. A quadratic form of the error is suggested as our performance function and is written as follows.

$$J(t) = \min_{\delta u} ||e^{\top}(t) \cdot e(t)|| \tag{7.9}$$

The above performance function is the objective function which is evaluated from the system output and setpoint. The convergence speed of this PF is faster than that of "an error" ($e(t)$) only.

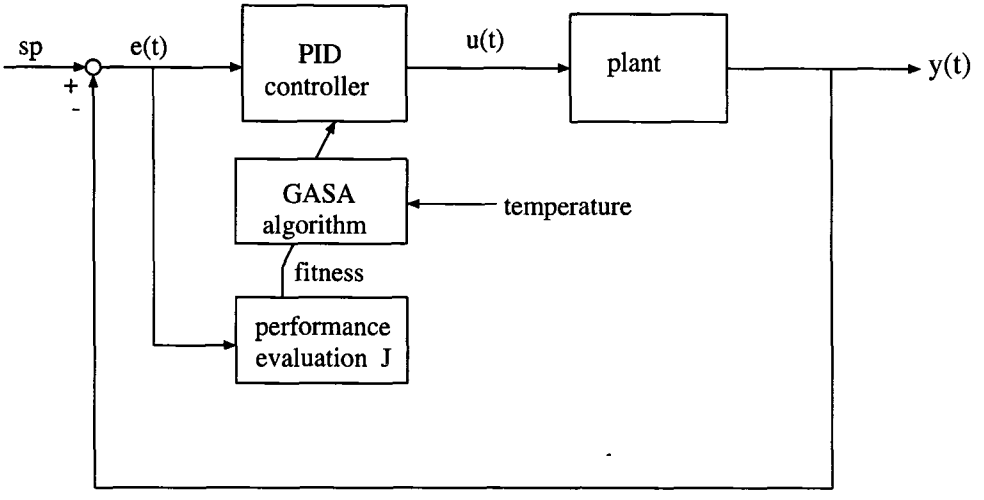


Figure 7.5: *A PID + GASA control system*

7.4.1 PID tuning by genetic-annealing approach

The parameters of PID controller are K_p, K_i, K_d. They can be represented in Eq. (7.7) as $K_p = K_{PID}, K_i = K_p/\tau_I, K_d = K_p\tau_D$. To use the genetic approach, a combination of these three parameters is formed as one **individual**. Each parameter can be generated with length l in binary. So an individual's length is $3l$. Suppose that K_p, K_i, K_d are bounded in $[0, K_{pb}]$,$[0, K_{ib}]$,$[0, K_{db}]$, where K_{pb}, K_{ib} and K_{db} are the upper bounds for K_p, K_i and K_d respectively.

Tuning by GAs

The individuals are initialized randomly with a population size λ. The decimal value is decoded from the binary strings of individuals. Fig. 7.4 indicates that GAs evaluate the fitness according to the performance function in Eq. (7.9). The high-fitness individuals can survive and be reproduced after selection. With the GA method, the reproduced individuals will be processed by the genetic operators (crossover and mutation) unconditionally. The new set for $\{K_p, K_i, K_d\}$ will be evaluated generation by generation. The PID control will be tested with the new parameter set until the

system is stable.

Tuning by GASA

The difference in tuning for GASA and GAs can be seen by comparing Fig. 7.5 and Fig. 7.4. Obviously, the additional procedure for GASA is the use of temperature and the calculated energy to select the "qualified" individual. Therefore, not all individuals will be processed by genetic operators. In this algorithm, only the "high energy" individuals will be processed. In practice, "high-energy" individuals mean those individuals with low survival probability. These individuals have to have their structural change by crossover and mutation to improve their chance of survival. The environment is the applied plant model. The thermodynamic selection criterion is dependent on $exp(-(E - E_{min})/T)$ which is so-called **Boltzmann distribution**.

E and E_{min} are the performance function value and its best performance in the population. This procedure is called the **Modified Metropolis Criterion** (MMC) [97, Sun et al, 1994] which is a modification of the Metropolis algorithm [69, Metropolis, 1953]. The qualified individuals are processed by genetic operation. However, the individuals which are processed by genetic operation are "conditional" instead of "unconditional" as in GAs. The PID control parameters $\{K_p, K_i, K_d\}$ still can be found generation by generation. The control action is operated with the applied control parameter set. When the $J < \zeta$, the overall operation is terminated, where ζ is a tiny positive number.

7.5 Experiment Result

The experiment apparatus is illustrated in Fig. 7.1. For the simulated pH plant, the process parameters are given as $Q_a = 0.11l/min$, $C_a = 10^{-3}mol/l$, $Q_b = 0 \sim$

$0.25l/min$, $C_b = 10^{-3}mol/l$ and the reactor volume V is $2.0l$. The pseudo steady state is at $dQ/dt = 0$, so $Q_b = Q_a C_a / C_b = 0.11l/min$. The operation time is 8 min.

The program is run on SUN Sparc 10. The integration method is Runge-Kutta method, the time step is 0.02 min to meet the requirement of the high sensitivity of pH neutralization operation. The upper bounds for the PID control parameters are chosen as $K_{pb} = 0.045$, $K_{ib} = 0.05$ and $K_{db} = 0.04$ which can be estimated from the step response to the process. The disturbance is considered as a white noise to perturb the process parameter, i.e. Q_a changes from $0.11l/min$ to $0.13l/min$. The process tends to deviate away from the neutrality point ($pH = 7$), but it can be corrected by the PID controller to maintain the overall stability.

The first technique applied for the close-loop operation is ZN. The ZN method is used to find the PID control parameters with continuous cycling. The ultimate gain and period are obtained as $K_u = 0.23$ and $\tau_u = 0.20$. So the $\{K_p, K_i, K_d\}$ can be calculated from Eq. (7.8) and shown in Table 7.1. Fig. 7.6 is the result of pH versus time with ZN method. Fig. 7.7 shows the error with the ZN method.

GAs and GASA were used to optimize the parameters of PID controller. The results are compared in Fig. 7.8~Fig. 7.11. The worst performance occurs around $t = 0.5$ min as seen in Fig. 7.10. After that, the base flow rate is almost $0.11l/min$ which is equal to the the acid flow rate. This indicates that the control point is around the neutrality point. The acidity (pH) of the tank is less than 7 after that time (see Fig. 7.11), that is, $[H^+] \approx [OH^-]$. Fig. 7.10 shows both GAs and GASA converge in this experiment. But GASA has less oscillation than GAs. From Fig. 7.8, we see that the genetic-annealing approach can maintain the pH reactor at the neutral value. The population size for GAs and GASA is 30 and the bits for each PID parameter were 10. Both techniques start from the same initial population. The starting temperature

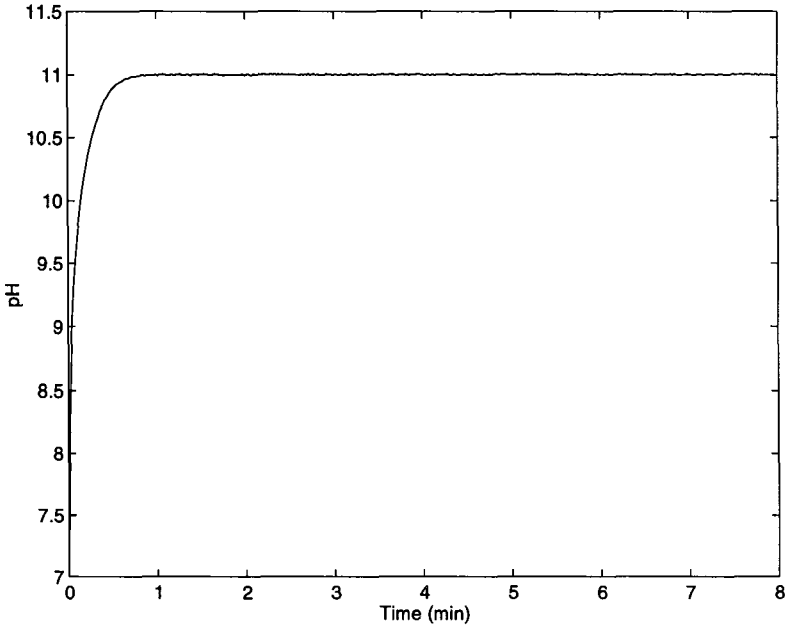


Figure 7.6: *pH vs. time for ZN tuning method*

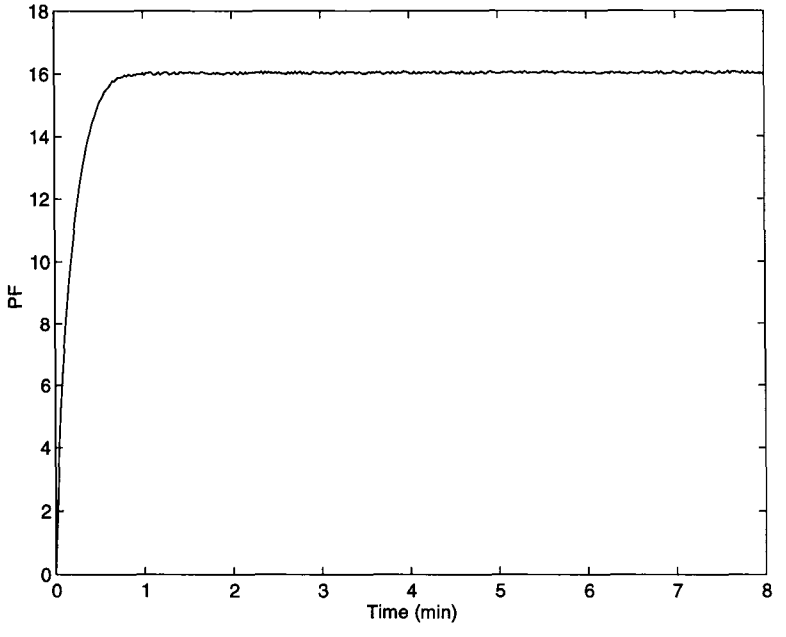


Figure 7.7: *Performance function (PF) vs. time for ZN tuning method*

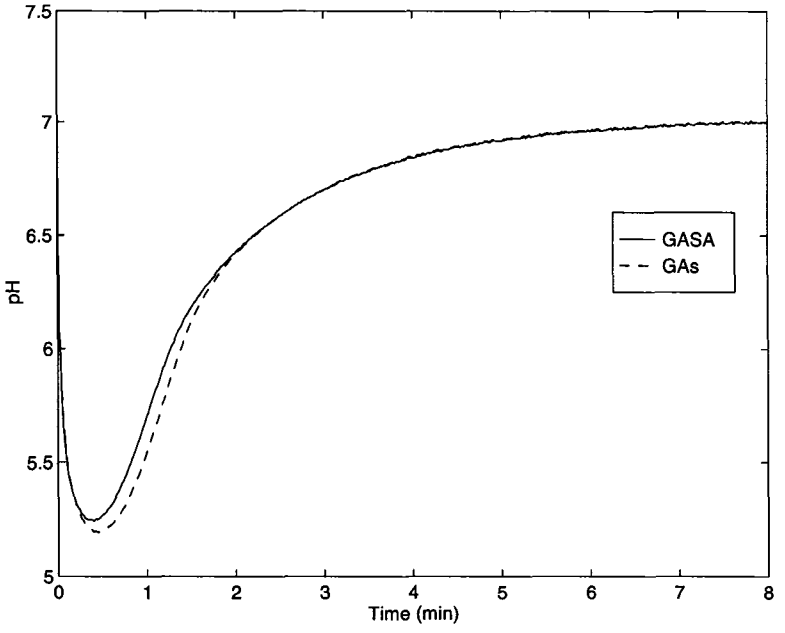


Figure 7.8: *pH vs. Time*

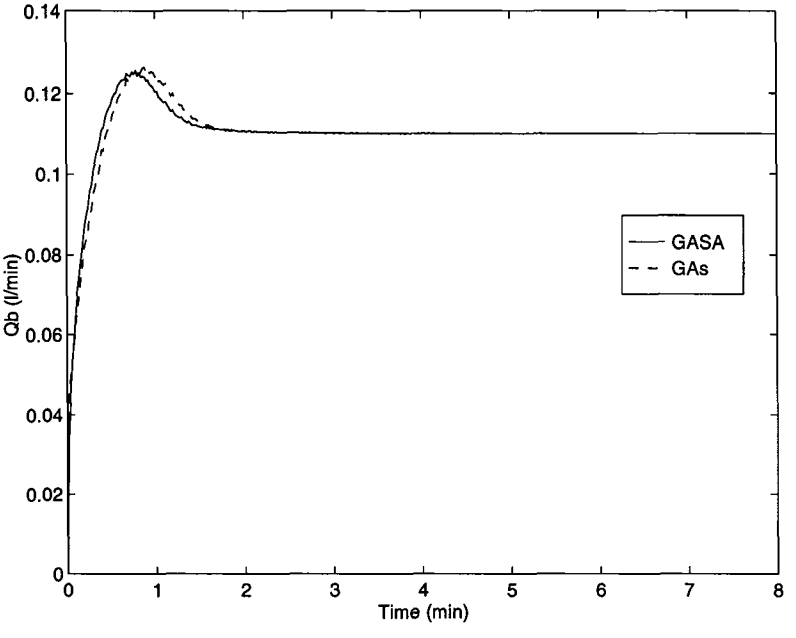


Figure 7.9: *The base flow rate profile*

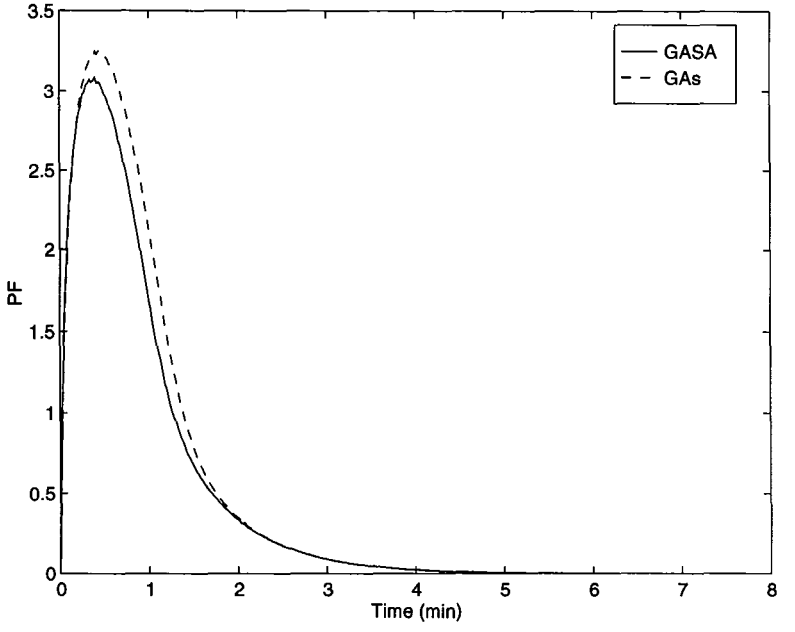


Figure 7.10: *PF vs. Time*

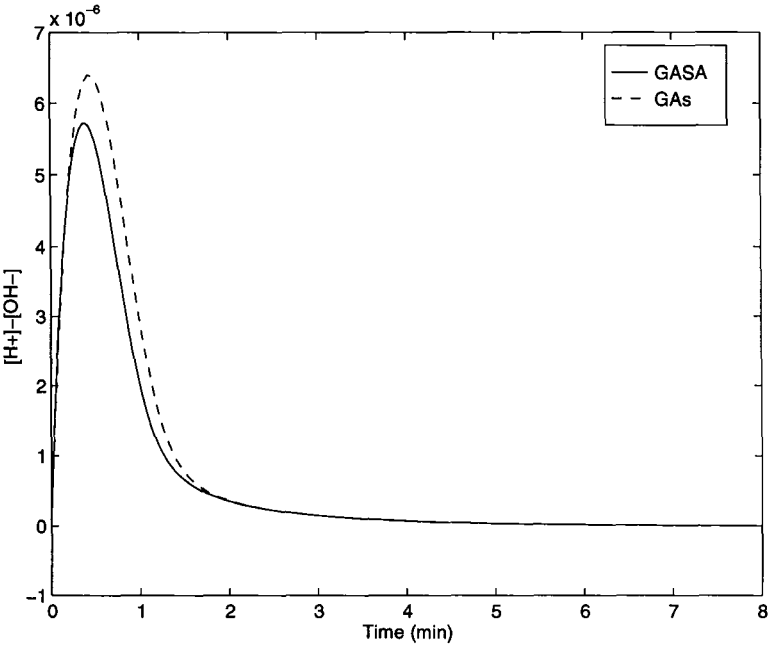


Figure 7.11: *The profile of* $[H^+] - [OH^-]$

Table 7.1: *The PID parameters and performance function (PF) for ZN, GAs and GASA*

	K_p	K_i	K_d	PF
ZN	0.13800	2.00000	0.00345	15.9986
GAs	0.02357	0.02796	0.00066	8.9×10^{-5}
GASA	0.02588	0.03094	0.00027	5.9×10^{-6}

for GASA is 6 and the annealing coefficient is 0.82.

The PID parameters and the central performance are listed in Table 7.1. The largest error happens when using the ZN method. On the other hand, it results in a pH process with too much base, and it is difficult to reach our control objective. The very sensitive nonlinearity causes the ZN method to fail , GAs took 201 generations to converge in this experiment, and GASA only took 21 generations.

7.6 Convergence Analysis

Models of industrial processes are only approximate representations of the actual systems; therefore, process control designs should accommodate parameter uncertainties and variations. The concept of robustness refers to the preservation of close-loop stability under allowable variations in system parameters. The stability of the operating point or region of the close-loop system needs to be asymptotically stable. However, feedback systems are inherently robust, that is, the relevant properties of the close-loop system tend to be preserved under parameter variations. For nonlinear systems, it is very difficult to use traditional methods to analyze the robustness of the control system. A simple method by observation of the trends is explained as below.

The stability of the control system design with our genetic-annealing technique can be found from Fig. 7.12(c) and Fig. 7.14(c) where the performance index decreases as the generations increase. Fig. 7.14(a) shows the magnitude of the variables — PID parameters are independent of one another. K_i has the largest magnitude for both the GAs and GASA. The locations of all individuals in the last generation can be found from Fig. 7.12(b) and Fig. 7.14(b). The latter picture of GASA indicates that the individuals move towards together very soon and the area is smaller than GAs'.

The fitness of GAs and GASA are illustrated in Fig. 7.13(a) and Fig. 7.15(a) respectively. GAs have very unfit individuals appear around generation 143. The unfit individual for GASA appears at generation 13. The scale window of fitness is 2 applied here (see Eq. (4.6)). The standard deviation for GAs (see Fig. 7.13(b)) is larger than GASA (see Fig. 7.15(b)) and shows strong oscillation compared to GASA. The output (pH) for GAs converges after generation 80, which is indicated in Fig. 7.13(d). GASA converges more quickly than GAs, it can be found in Fig. 7.15(d).

The optimization ability of the genetic-annealing method can be used to determine the optimal control parameter set. This method is more precise and faster than the traditional PID tuning technique and can have higher stability than the classical method (i.e. ZN method). Nonlinear design with population-based design can overcome the uncertainty of the system.

7.7 Summary

A nonlinear, constrained feedback controller design procedure for stationary lumped nonlinear systems is presented and analyzed. The global self-adaptive ability of the genetic-annealing method has been successfully applied for pH control. The

advantages of this method appears in the global search ability and its convergence characteristics. The GASA algorithm shows promise of being applicable to any control problem which lends itself to a minimization scheme. From the convergence analysis, we can understand more about the procedures of the execution of this algorithm. The assistance of temperature can weed out inferior points that occur, over generations. GA and GASA is compared with the classic Ziegler-Nichols tuning method. The PID control algorithm is reactivated with the novel population-based search algorithms and can have higher efficiency to the practical operation.

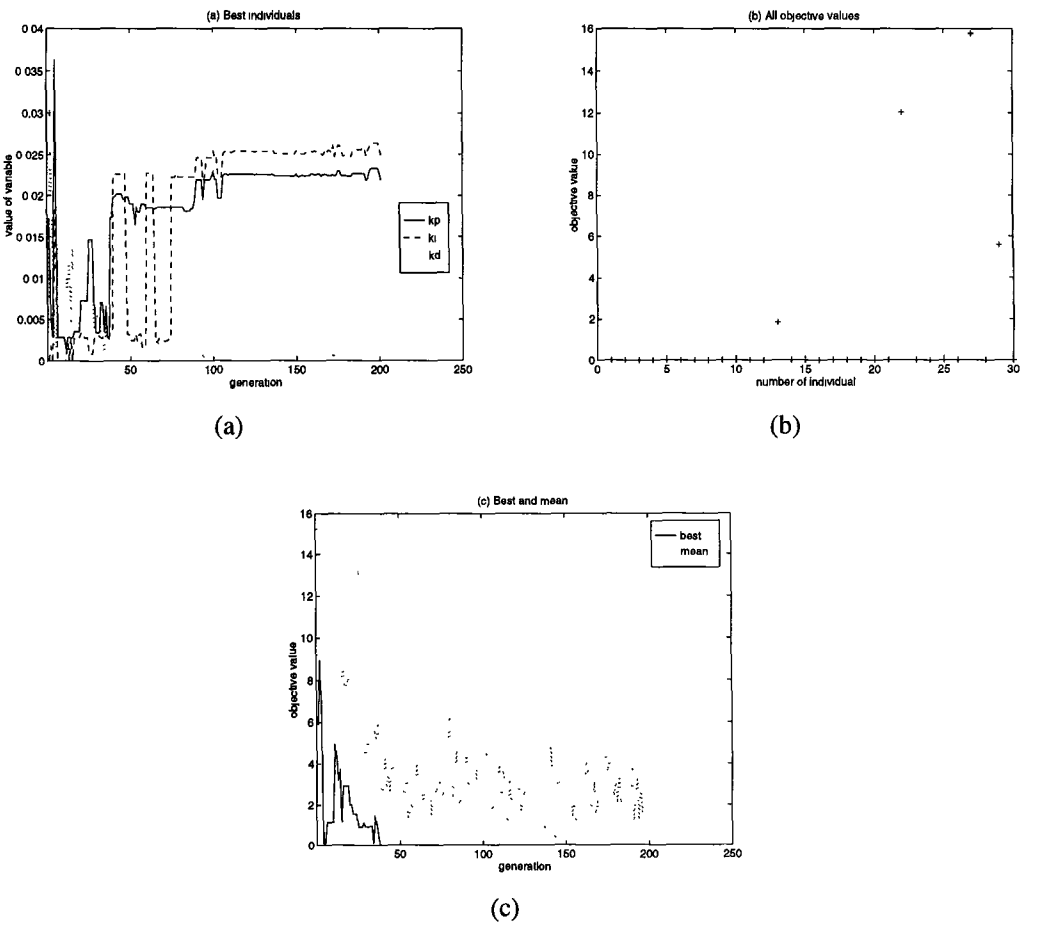


Figure 7.12: *An Interpretation of GAs operation in generations: (a) PID parameters of best individuals in generations. (b) The objective values of all individuals in generation 201. (c) Best and mean objective values of individuals in generations.*

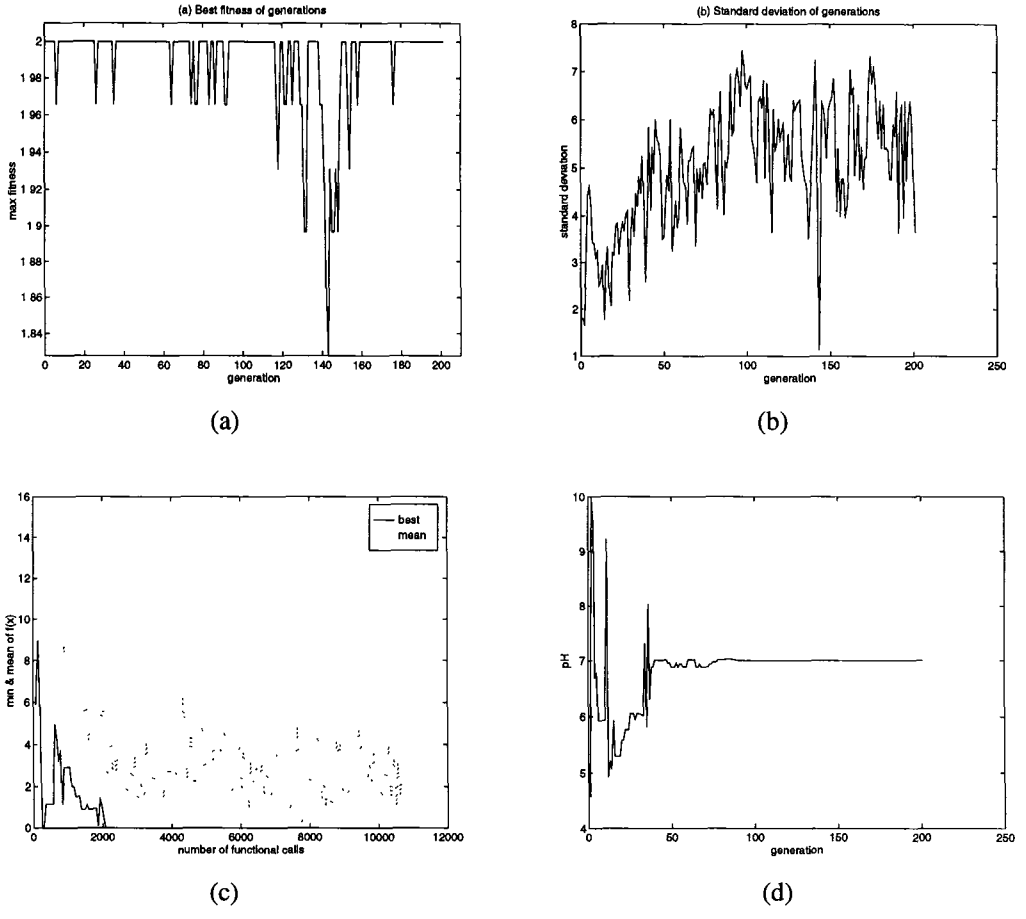
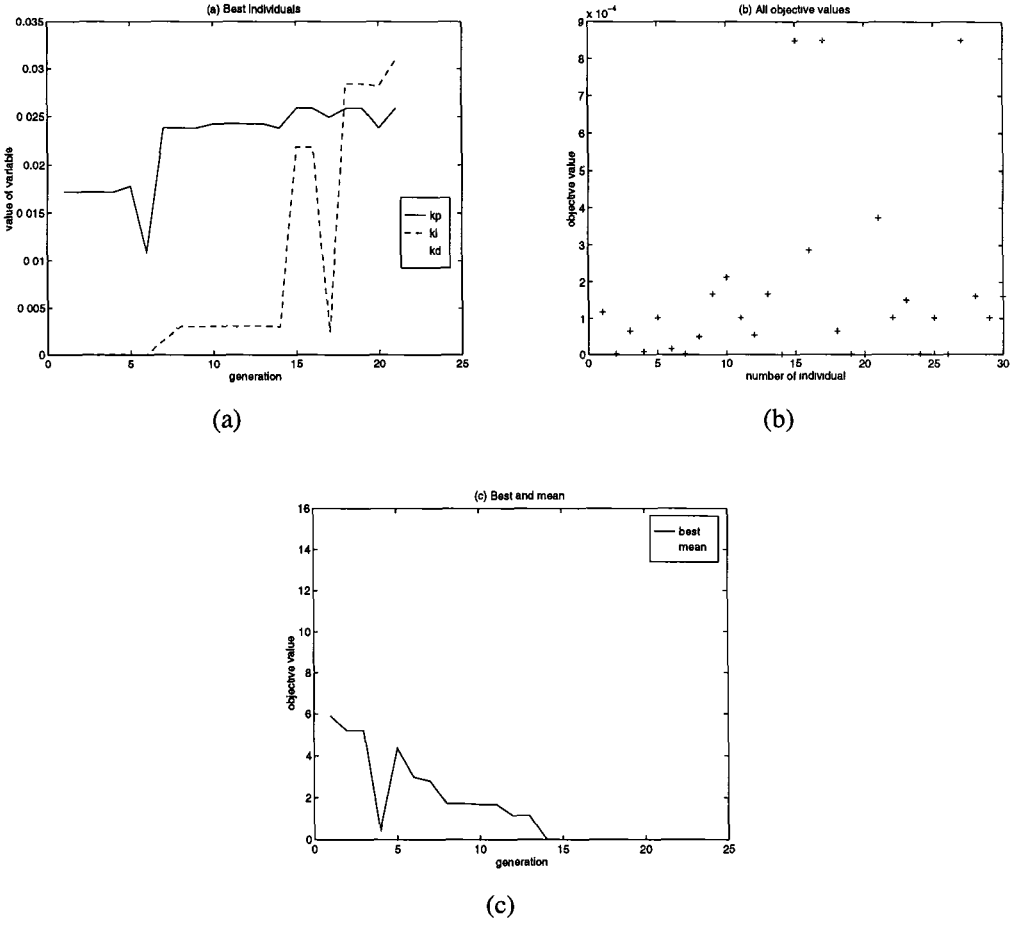


Figure 7.13: *Three comparison results of GAs: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. (d) Best pH vs. generation.*



(a) (b) (c)

Figure 7.14: *An Interpretation of GASA operation in generations: (a) PID parameters of best individuals in generations. (b) The objective values of all individuals in generation 21. (c) Best and mean objective values of individuals in generations.*

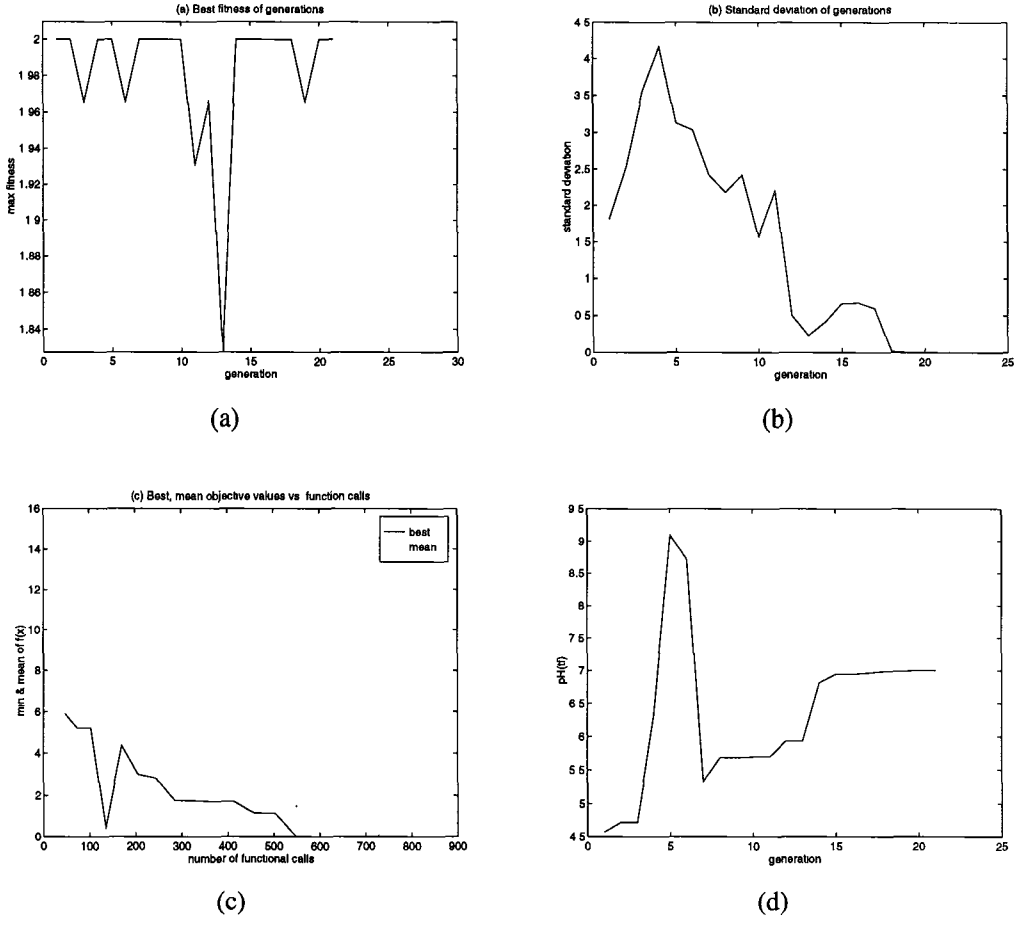


Figure 7.15: *Three comparison results of GASA: (a) Best fitness in generations. (b) Standard deviation of generation. (c) Best and mean objective values vs. function calls. (d) Best pH vs. generation.*

Chapter 8

Conclusions and Future Perspectives

As stated earlier in this thesis, the research undertaken was aimed at uncovering a combination of genetic algorithms and simulated annealing while addressing practical problems with this combination. The development of a combination of genetic algorithms and thermodynamic search algorithms is presented in this thesis. The contributions of this thesis have been marked in Chapter 2. The successful derivation and application of the combined approach (GASA algorithms) is explored. These goals have been fulfilled and detailed descriptions for each component of the thesis have been presented in each Chapter. This chapter issues several conclusions and important accomplishments of the whole thesis, and summarizes them in the first section of this chapter. Then some useful related issues that are worthy of examining in the future are raised in the second section.

8.1 Conclusions from the Dissertation

A very striking thing about genetic algorithms (GAs) and other parallel search algorithms is the richness of the form of computation. The recent theoretical developments and applications of genetic algorithms is illustrated in Chapter 3, which also reports an

effort to establish an analytical framework for a simple genetic algorithm. Population and selection effectively implement a fixed queue of states, and any time those states with the best heuristic value will tend to be "opened" in order to generate successors. One main difference from other conventional search algorithms is that the state is updated as a batch process through stochastic sampling. The general search can be initialized with many points together in the search domain. The natural selection is applied to the procedure of selection.

The main distinction for GAs is that it introduces a concept of "coding". This concept is quite different from the traditional search methods and other evolutionary algorithms. This action is also very helpful to the exploration and exploitation of the advance in the genetic and biochemical computation. This coding type can match the pairs of coding for genes in genetics. In practice, these procedures are programmed with binary Boolean algebra which declares strings by "0" and "1". The functions to implement GAs are assisted with the major genetic operators — crossover and mutation. They are the main mechanisms to create new structures for individuals.

The other combined computation algorithm is simulated annealing (SA), which is reviewed in Sec.3.3. The foundation is based upon the asymptotic behavior of a nonstationary simulated annealing algorithm model. SA originates from statistical mechanics and obeys some behaviors of thermodynamics. The main introduced concept from thermodynamics is "equilibrium". When particles follow the Boltzmann distribution, they could achieve pseudo thermodynamic equilibrium. The mathematical derivation of the decision rules is according to the Metropolis algorithm which determines the "qualified" ones in the search space. Other related search algorithms are structured in Chapter 3. Nonlinear programming and other typical search algorithms are also described in this chapter. These search algorithms are very useful for

nonlinear programming problems.

The inherently successful combination of genetic algorithms and simulated annealing is described in Chapter 4, which is abbreviated as **GASA**. The importance of this combination is considered from the format of the search behavior of the two combined algorithms (GAs and SA). GAs start with populations having many individuals and follow the survival-of-the-fittest rule in Nature. The parallelized search action begins with the individuals in populations. The effective division of the participating individuals are dependent on their energy states. The energy can be converted from the applied objective function. The algorithm design and some extracted characteristics are illustrated in this chapter. The modified Metropolis criterion is used to determine the qualification of an individual to proceed with advanced operations. This combined algorithm is investigated with a function optimization problem (see Sec.4.4) which has nonlinearities of objective functions and multiplicity of local extrema that have been stumbling blocks in the search for methods capable of locating global extrema. The GASA algorithm can overcome these difficulties generated in the multiple minima problem.

Furthermore, the search behavior of the GASA algorithm and other factors to influence the global search are advanced in research reported in Chapter 5. Several design procedures can improve the efficiency of the GASA algorithm. First of all, the **annealing schedule** (or **temperature scale** in annealing) produces a major effect on the consuming time of running and the efficiency of obtaining the result. The temperature plays a critical role in the search. The partial annealing mechanisms can produce the other GASA algorithms (GASA_C and GASA_M). The schema theorem is very important to explore the trajectory of GAs. We have extended the original schema theorem by Holland (1975) to obtain three new schema theorems on GASA,

GASA_C, and GASA_M. Along with the new concept **simulated phase transitions**, these theorems lend insight into explaining the search mode for GASA algorithms. The individuals occupy different states which are decided by their energy (objective value). The search by GASA is phase-by-phase instead of point-by-point. One phase can construct a search plane in the search space. The location of an individual is changed from one hyperplane to another. This behavior is illustrated with the aid of phase transition from thermodynamics. The three new theorems establish the robust theoretical fundamentals for the combined algorithms. The other invention is the submission of the new equilibrium rule for the population-based computation algorithms, whereas $< E >= \min(E)$ is applied to explain the convergence criterion for the population-based algorithms on optimization.

The industrial practical applications with the genetic-annealing based algorithm are explored in Chapter 6 and Chapter 7.

The dynamic optimization of the input profile for a fed-batch bioreactor is investigated in Chapter 6. The conventional optimization method derived from the maximum principle cannot define a sufficient condition when an optimized profile is found. These methods are also limited and time-consuming for the search procedure because of the classical hill-climbing methods which help us find the derivatives from the Hamiltonian operator. The results show the high cell productivity when the input is optimized by the GASA technique. In industry, a slight difference of even one gram can be scaled up to a great reduction of cost and operating time. The unstable nature of the system causes difficulties for dynamic optimization. However, the GASA algorithm provides an efficient approach for dynamic optimization.

The assistance with the genetic-annealing approach to the classical control strategy is presented in Chapter 7. In the chemical industry, PID control is a very popular control

scheme because of its convenience of operation. Controller operation is often based on an engineer's experience and ability to tune the control parameters. However, it can be inefficient and laborious. Under closed loop operation, the optimization ability of the genetic-annealing technique holds great benefits for plant control. The convergence analysis provides us a transparent overlook of the processes of this approach. This approach also provides an asymptotically optimal parameter setting technique for PID controllers incorporated in a continuous model-following control system. The self-regulation of the combined GAs and SA algorithms can quickly determine the optimal controller parameters when applied to the SISO pH plant.

8.2 Future Perspectives

The dramatic expansion in domain of application for our search afforded by the combined GASA algorithm has been mirrored by the diversity of applications beginning to appear in the literature. The GASA combination covers many diverse inspired concepts from biology, physics, genetics, mathematics and thermodynamics, and is complemented by the availability of parallel computing power which GASA can exploit. This rapid and broadly successful automation technique has given rise to the hope of full automation is a panacea for solving more complex problems.

Our technique allows the promise to approach objective optimization directly in an efficient way. To meet the increasing market competition and environmental constraints, this technique still can be explored from different aspects. They can be

- Nonlinear programming, dynamic optimization (e.g. fed-batch optimization), combinatorial optimization.
- System identification, fault detection (e.g. pipeline leak), decision of the un-

known and difficult parameters on modeling from experimental data.

- Prediction of molecular structures (e.g. protein structure), optimization of engineering designs (e.g. ship building).
- Improved traditional control strategies (e.g. PID, Kalman filter), modern control theory and intelligent control system.
- Pattern recognition, machine learning (e.g. classifier system), image processing (e.g. analysis of oil source from data acquisition through satellites).
- Combined with neural networks, fuzzy theory, expert system and other artificial intelligence approaches.
- Optimized chip design in semiconductor manufacturing and parallelization design of processors.
- Through machine learning, this technique still can be applied to model-based prediction and model predictive control.
- Provides an efficient optimization approach to econometrics with its strong search capability, and marketing prediction.
- The investigation of the related theoretical exploration. For example, the advanced computation methods originate from evolutionary computation methods, thermodynamics and other artificial intelligence techniques.

The above are the future perspectives for advanced development and application. Therefore, further research could well be directed to relate more close collaboration between scientists and engineers, and provides an advanced integration to the diversities between science and engineering with the crossed interdisciplines.

Appendix A

Nomenclature

All vectors are column vectors.

Roman Letters:

A_{ji} : State transition probability

$b(t)$: The number of individuals in generation t.

C : Crossover operator.

c : Annealing coefficient or temperature decrement parameter.

D : Dimension of function.

E : Energy or objective value.

$e(t)$: Error function for system output and set point, a vector.

f : Applied function.

G_{ji} : State generating probability.

$G(x, u)$: Applied model.

h : Generating probability.

J : Objective function or performance function (PF) or performance index (PI).

K_p, K_i, K_d : PID control parameters for proportional, integral and derivative respectively.

K_u : Ultimate gain for Ziegler-Nichols cycling tuning method.

LB : Lower bound for system input $u(t)$.

l : Population size.

M : Mutation operator.

$\vec{P}(t)$: Population vector at generation t.

p_c : Crossover rate.

p_m : Mutation rate.

p_i : Threshold probability.

S : Substrate mass (g).

s : Selection operator; substrate concentration (g/l).

T_t : Temperature at generation t.

t : generation for algorithms; time for applied systems.

t_f : Final operation time.

U : Binary set $(= \{0,1\}^l)$.

UB : Upper bound for system input $u(t)$.

$u(t)$: System input function.

V : Reactor or tank volume.

X : Cell mass (g), a scalar.

$\vec{X}$: A variable vector.

x : cell concentration (g/l), a scalar; state variable, a vector.

x_i : The ith variable.

x^* : Location of minimum.

Y : Yield coefficient (g/g).

y : System output.

$Z(T)$: Partition function.

Greek Letters:

δ : Scale function for fitness.

Γ : Decode function (binary $\rightarrow$ decimal).

μ : Specific growth rate ($1/h$).

ν : Binary bit (0 or 1).

$\underline{\underline{\nu}}(t)$: Binary matrix of $\nu(t)$.

$\mathbf{\Phi}_i$: Fitness of individual i.

ω : Scale window.

τ_u : Ultimate period for Ziegler-Nichols cycling tuning method.

Ξ : A whole gene pool.

Ξ_c : Gene pool for low-energy individuals and to be retained before crossover.

Ξ_{nc} : Gene pool for high-energy individuals and to be crossed-over.

Ξ_m : Gene pool for low-energy individuals and to be retained before mutation.

Ξ_{nm} : Gene pool for high-energy individuals and to be mutated.

ζ : Stop criterion for programming.

Other Symbols:

$*$: Optimal location; wild card ("don't care").

$\vec{}$: vector, on top.

$=$: matrix, at bottom.

Bibliography

[1] Aarts, E. and Korst, J., 1989. *Simulated Annealing and Boltzamnn Machine.* Wiley, Chichester, U.K.

[2] Aarts, E. and Laarhoven, P., 1985a. *Statistical cooling: a general approach to combinatorial optimization problems. Philips* Journal of Research, **40**, 193-226.

[3] Aarts, E. and Laarhoven, P. *A new polynominal time cooling schedule.* In *Proceedings of IEEE International Conference on Computer-Aided Design,* 206-208, Santa Clara, CA., 1985b.

[4] Ackley, D. H. *An empirical study of bit vector function optimization,* 170-204. In Davis [22], 1987.

[5] Ackley, D. H. and G. E. Hinton and T. J. Sejnowski, 1985. *A learning algorithm for Boltzman machines.* Cognitive Science, **9**, 147-169.

[6] Alander, J. T. *On finding the optimal genetic algorithms for robot control problems.* In *Proceedings IROS '91 IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. Intelligence for Mechanical Systems,* volume 3, 1313–1318, Osaka, Japan, 3-5 Nov 1991. IEEE, New York, NY.

[7] Arthur E. Bryson, Jr. and Yu-Chi Ho, 1975. *Applied Optimal Control: Optimization, Estimation, & Control.* John Wiley & Sons, New York, NY.

[8] Åström, K. L. and Hägglund, T., 1984. *Automatic Tuning of Simple Regulators with Specifications in Phase and Amplitude Margins.* Automatica, **20**:(5), 645-651.

[9] Axelrod, R. *The evolution of strategies in the iterated prisoner's dilemma,* 32-41. In Davis [22], 1987.

[10] Bailey, James and Ollis, D. F., 1986. *Biochemical Engineering Fundamentals.* 2nd ed. McGraw-Hill Inc., New York.

[11] Betts, J. T., 1977. *An accelerated multiplier method for nonlinear programming.* Journal of Optimization Theory & Application, **21**, 137-174.

[12] Biggs, M. C. *Constrained minization using recursive quadratic programming: some alternative subproblem formulations for parallel network.* In *Towards Global Optimization: proceedings of a workshop at the University of Cagliari, Italy.*, 1975.

[13] Bohachevsky, Ihor O., 1986. *Generalized simulated annealing for function optimization.* TECHNOMETRICS, **28**:(3), 209-217.

[14] Bremerman, H. J. *Optimization through evolution and recombination,* 93-106. Spartan Books, Washington, D. C., 1962.

[15] Buchholt, F and Kummel, M. *Self-tuning control of a pH neutralization process.*

[16] Charnes, A. and Wolfe, M., 1989. *Extended Pincus theorems and convergence of simulated annealing.* Int. J. Systems Sci., **20**:(8), 1521-1533.

[17] Cohoon, J. P. and Martin, W. N. and Richards, D. S. *Genetic algorithms and punctuated equilibria in VLSI*In , editors, *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, volume 496 of *Lecture Notes*

in Computer Science, 134–144, Dortmund, Germany, 1-3 Oct 1991. Springer-Verlag, Berlin, Germany.

[18] Curtis, A. R. D., October 1991. *An application of genetic algorithms to active vibration control.* Journal of Intelligent Material Systems and Structures, **2**:(4), 472–481.

[19] Cuthrell, J. E. and Biegler, L. T., 1989. *Simultaneous Optimization and Solution Methods for Bath Reactor Control Profiles.* Computers & Chemical Engineering, **13**:(1/2), 49.

[20] Thomas Dandekar and Patrick Argos, 1994. *Folding the Main Chains of Small Proteins with the Genetic Algorithm.* Journal of Molecular Biology, **236**, 844-861.

[21] Davis, L., 1991. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

[22] Davis, L. and Steentrup, M., 1987. *Genetic Algorithms and Simulated Annealing.* Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[23] Davis, Thomas Elder. *Toward an Exploration of the Simulated Annealing Convergence Theory onto the Simple Genetic Algorithm.* PhD thesis, University of Florida, Gainesville, FL., 1991.

[24] De Jong, K. *A 25 Year Perspective*, 125-134. In Zuradaet *al* [111], 1994.

[25] DeJong, K. A. *The Analysis of Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, MI., 1975.

[26] Dorigo, M. and Maniezzo, M. and Montanari, D. *Classifier-based robot control systems.* In *IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control*, 591–598, Delft, Netherlands, 1992.

[27] Etter, D. M. and Hicks, M. J. and Cho, K. H. *Recursive adaptive filter design using an adaptive genetic algorithm.* In *Proceedings of an International Conference on Accoustics, Speech, and Signal Processing*, volume 2, 635-638, 1982.

[28] Filho, J. and Alippi, C. and Treleaven, P. *Genetic Algorithm Programming Environments.*

[29] Finlayson, B.A., 1980. *Nonlinear Analysis in Chemical Engineering.* McGraw-Hill, New York.

[30] Floudas, C. A. and Pardalos, P. M., 1992. *Recent Advances in Global Optimization.* Princeton Series in Computer Science. Princeton University Press, Princeton, NJ.

[31] Fogel, D. B. *Evolutionary System Identification and Control.* In Weaver [104], 1271–1274.

[32] Fogel, D. B. *On the Philosophical Differences between Evolutionary Algorithms and Genetic Algorithms*In , editors, *Proceedings of the Second Annual Conference on Evolutionary Programming*, 23-29, La Jolla, California, 1993.

[33] Fogel, L. J. and Owens, A. J. and Walsh, M. J., 1966. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York.

[34] Fontain, E., Nov-Dec 1992. *Application of Genetic Algorithms in the Field of Constitutional Similarity.* Journal of Chemical Information and Computer Sciences, **32**:(6), 748–752.

[35] Frantz, D. R., 1e72. *Non-linearlities in genetic adaptive search.* Dissertation Abstracts International, **33**:(11), 5240B-5241B.

[36] Geman, S. and Geman, D., 1984. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, **PAMI-**:(6), 721-741.

[37] Gidas, B., 1985. *Non-stationary Markov Chains and Convergence of the Annealing Algorithm.* J. of Statistical Physics, **39**, 73-131.

[38] Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization & Maching Learning*. Addision-Wesley, Reading, MA.

[39] Goldberg, D. E. and Lingle, R. *Alleles, Loci, and the Traveling Salesman Problem.* In Grefenstette [41], 154-159. Canegie-Mellon Univ.

[40] Gorges-Schleuter, M. *ASPARAGOS: A Parallel Genetic Algorithm for Population Genetics*In , editors, *Parallelism, Learning, Evolution. Workshop on Evolutionary Models and Strategies - WOPPLOT 89*, 407-418, Berlin, 1991. Springer-Verlag.

[41] Grefenstette, J. J., 1985a. *Proceedings of an International Conference on Genetic Algorithms and Their Applications.* Lawerence Erlbaum Associates, Hilldale, NJ.

[42] Greffenstette, J. J., 1986. *Optimization of Control Parameters for Genetic Algorithms.* IEEE Transactions on Systems, Man, & Cybernetics, **16**, 122-128.

[43] Gustaffson, T. K., 1985. *An experimental study of a class of algorithms for adaptive pH control.* Chemical Engineering Science, **40**, 827-837.

[44] Hajeck, B. *Cooling schedules for optimal annealing.* In *Research Notes in Artificial Intelligence* [22], 1985.

[45] Hinton, G. E. and Sejnowski, T. J., 1983. *Analyzing cooperative computation.* Proceedings of the Fifth Annual Conference of the Cognitive Science Society.

[46] Holland, J., 1975. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press., Ann Arbor.

[47] Holland, J. H., July 1992. *Genetic Algorithms.* Scientific American, 66-72.

[48] Holland, John. *Hierarchibical descriptions of universal spaces and adaptive systems.* Ora projects 01252 and 08226, University of Michigan, Dept. of Computer and Communication Sciences, Ann Arbor, 1968.

[49] Huang, M. and Romeo, F. and Sangiovanni-Vincentelli, A. *An Efficient General Cooling Schedule for Simulated Annealing.* In *Proceedings of IEEE International Conference on Computer-Aided Desig n*, 381-384, Santa Clara, CA., 1986.

[50] Ingber, L., 1989. *Very fast simulated re-annealing.* Mathl. Comput. Modeling, **12**:(8), 967-973.

[51] Ingber, L., 1992. *Generic mesoscopic neural networks based on statistical mechanics of neocortical interactions.* Physical Reviews, **A 45**:(4), R2183-R2168.

[52] Ingber, L. and Rosen, B. *Very Fast Simulated Reannealing (VFSR).* University of Texas, San Antonio, TX, 1992. [ringer.cs.utsa.edu:/pub/rosen/vfsr.Z].

[53] Irwin, G. W. *Parallel algorithms for control.* In *Proc. 1992 IFAC Workshop on Algorithms and Architectures for Real-Time Control*, 15-22, Pergamon Press, 1992.

[54] Kirkpatrick, S., 1984. *Optimization by simulated annealing: Quantitative studies.* Journal of Statistical Physics, **34**:(5/6), 975-986.

[55] Kirkpatrick, S. and Gelatt Jr., C. D. and Vecchi, M. P., 1983. *Optimization by simulated annealing*. Science, **220**:(4958), 671-680.

[56] Kitano, H., August 1990. *Designing neural networks using genetic algorithms with graph generation system.* Complex Systems, **4**:(4), 461–476.

[57] Krishnakumar, K. and Goldberg, D. E., May-Jun 1992. *Control System Optimization Using Genetic Algorithms.* Journal of Guidance Control and Dynamics, **15**:(3), 735–740.

[58] K. Kristinsson and G. A. Dumont, Sep-Oct 1992. *System Identification and Control Using Genetic Algorithms.* IEEE Transactions on Systems Man and Cybernetics, **22**:(5), 1033–1046.

[59] Kyle, B. G., 1984. *Chemical and Process Thermodynamics.* Prentice-Hall Inc., Englewood Cliffs, N. J.

[60] Lewis, F. L., 1986. *Optimal Control.* Wiley, New York, N.Y.

[61] Li, W. C. and Biegler, L. T. and Economou, C. G. and Morari, M., 1990. *A Constrained Pseudo-Newton Control Strategy for Nonlinear Systems.* Computers & Chemical Engineering, **14**, 451-468.

[62] Li, W. C. and Biegler, L. W., 1988. *Process Control Strategies for Constrained Nonlinear Systems.* I&EC Research, **27**:(8), 1421-1433.

[63] Li, W. C. and Biegler, L. W., 1989. *Multi-Step Newton Type Control Strategies for Constrained Nonlinear Processes.* Chemical Engineering Research & Design, **67**, 562.

[64] G. E. Liepins and M. R. Hillard, November 1989. *Genetic algorithms: Foundations and applications.* Annals of Operations Research, **21**:(1-4), 31–57.

[65] Lim, H. C. and Tayeb, Y. J. and Modak, J. M. and Bonte, P., 1987. *Computational Algorithms for Optimal Feed Rates for a Class of Fed-Batch Fermentation: Numerical Results for Pencillin and Cell Mass Production.* Biotechnology & Bioengineering, **28**, 1408-1420.

[66] Lundy, M. and Mees, A., 1986. *Convergence of an Annealing Algorithm.* Math. Prog., 111-124.

[67] Margarida, F. and Salcedo, R. and de Azevdo, S., 1994. *Nonequilibrium Simulated Annealing: A Faster Approach to Combinatorial Minimization.* I&EC Research.

[68] Metropolis, N. and Rosenbluth, A. W. and Teller, A. H. and Teller, E., 1953. *Equation of state calculations by fast computing machines.* J. of Chem. Phys., **21**:(6), 1087-1092.

[69] Metropolis, N. and Rosenbluth, A. W. and Teller, A. H. and Teller, E., 1953. *Equations of State Calculations by Fast Computing Machines.* J. of Chem. Phys., **21**:(6), 1087-1092.

[70] Modak, J. M. and Lim, H. C., 1989. *Simple Nonsingular Control Approach to Fed-Batch Fermentation Optimization.* Biotechnology & Bioengineering, **33**, 11-15.

[71] Toshinosuke Muto and Yutaka Takagi, 1956. *The Theory of Order-Disorder Transitions in Alloys.* Academic Press, New York, NY.

[72] Orava, P. J. and Niemi, A. J., 1974. *State model and stability analysis of a pH control process.* International Journal of Control, **20**, 557-567.

[73] Otten, R. and Ginneken, L. *Floorplan design using simulated annealing.* In *Proceedings of IEEE International Conference on Computer-Aided Design*, 96-98, Santa Clara, CA., 1984.

[74] Park, D. and Kandel, A. and Langholz, G., 1994. *Genetic Based New Fuzzy Reasoning Models with Application to Fuzzy Control.* IEEE Transactions on Systems, Man, & Cybernetics, **24**:(1), 39-47.

[75] Pincus, M., 1970. *A Monte Carlo method for the approximate solution of certain types of constrained optimization problems.* Oper. Res., **18**, 1225-1228.

[76] Pontryagin, L. S. and Boltyanskii, R. V. and Gamkrelidze, R. V. and Mischenko, E. F., 1962. *The Mathematical Theory of Optimal Processes.* Wiley & Sons, New York, N.Y.

[77] Ramirez, W. F. and Beyer, K. P. and Abedi, H., 1987. *Disturbance Feedback in Model Predictive Control Systems.* Optimal Regulator Control for Systems with Time-Varying Measurable and Partially Measurable Disturbance, **8**, 159-174.

[78] Ray, W. H., 1981. *Advanced Process Control.* McGraw-Hill, New York, N.Y.

[79] Rechenberg, I., 1973. *Bionik, Evolution und Optimierung.* Naturwissenschaftliche Rundschau, **26**:(11), 465-472.

[80] Reed, J. and Toombs, R. and Barricelli, N. A., 1967. *Simulation of biological evolution on machine learning.* Journal of Theoretical Biology, **17**, 319-342.

[81] Reif, F., 1965. *Statistical and Thermal Physics.* McGraw-Hill, New York, N.Y.

[82] Rudolph, Günter. *Global Optimization by means of distributed evolution strategies.* In Schwefel and Männer [87], 209–213.

[83] Rudolph, Günter, 1994. *Convergence Analysis of Canonical Genetic Algorithms.* IEEE Trans. Neural Networks, Special Issue on Evolutionary Com putation, **5**:(1), 96-101.

[84] Rumelhart, D. E. and McClelland, J. L., 1986. *Parallel Distributed Processing*, volume I and II. MIT Press.

[85] Schmitendorf, W. E. and Shaw, O. and Benson, R. and Forrest, S. *Using Genetic algorithms for controller design: Simultaneous stabilization and eigenvalue placement in a region.* In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Hilton Head, SC, 1992.

[86] Schwefel, H. P., 1977. *Numerische Optimization von Computer-Modellen mittels der Evolutionsstrategie*. Interdisciplinary Systems Research, **26**.

[87] Schwefel, H. P. and Männer, R., 1-3 Oct 1991. *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1.* Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, Dortmund, Germany.

[88] Sechen, Carl, 1988. *VLSI placement and global routing using simulated annealing*. Kluwer Academic Publishers, Boston, MA.

[89] Shanno, D. F. *Recent advances in numerical techniques for large scale optimization*In , editors, *Neural Networks for Control.* MIT Press, Cambridge, MA, 1990.

[90] Shinskey, F. G., 1988. *Process Control Systems.* McGraw-Hill, New York.

[91] Sines, B. J. and Teather, E. W. and Harvey, S. P. and Weigand, W. A. *Investigation of Biological Reactor Design for Treatment of Methanol and Thiodiglycol Waste Stream.*

[92] Sirag, D. J. and Weisser, P. T., 1987. *Torward a unified thermodynamic operator.* Proc. of 2nd Intl. Conf. of Genetic Algorithms and Their Applications, 116-122.

[93] Sistu, P. B. and Bequette, B. W., 1991. *Nonlinear Predictive Control of Uncertain Processes: Application to a CSTR.* AIChE Journal, **37**:(11), 1171-1723.

[94] Spall, James C., 1992. *Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation.* IEEE Transactions on Automatic Control, **37**:(3), 332-341.

[95] Hongte Ted Su. *Dynamic Modeling and Model Predictive Control Using Generalized Perceptron Networks.* PhD thesis, University of Maryland, College Park, MD., 1992.

[96] Sun, R. L. *Evolving Natural Learning Algorithms on Dynamic Systems*, 1994. Ph.D. Proposal, University of Maryland at College Park. Department of Chemical Engineering, University of Maryland, College Park, MD.

[97] Sun, R. L. and Dayhoff, J. E. and Weigand, W. A., 1994. *A Population-Based Search Through Thermodynamic Operation.* Submitted to IEEE Transactions on Systems, Man, & Cybernetics.

[98] Unger, R. and Moult, J., May 1993. *Genetic Algorithms for Protein Folding Simulations.* Journal of Molecular Biology, **231**:(1), 75–81.

[99] Van Impe, J. F. and Bastin, G. and De Moor, B. and Van Breusegem, V. and Vandewalls, J., 1992. *Optimal Control of the Penicillin G Fed-Batch Fermentation : an Analysis of a Modified Unstructured Model.* Chemical Engineering Communication, **117**, 337-353.

[100] Van Impe, J. F. and De Moor, B. and Vandewalls, J., 1993. *Singular Optimal Control of Fed-Batch Fermentation Processes with Inequality Constraints.* IFAC 12th Triennial World Congress, 275-278.

[101] Varšek, Alen and Urbančič, Tanja and Filipič, Bodgan, September/October 1993. *Genetic Algorithms in Controller Design and Tuning.* IEEE Transactions on Systems, Man, & Cybernetics, **23**:(5), 1330-1339.

[102] Vecchi, M. P. and Kirkpatrick, 1983. *Global writing by simulated annealing.* IEEE Transactions on Computer-Aided Design, **CAD-2**:(2), 215-212.

[103] Wang, P. and Kwok, D. P., 1993. *Optimal design of PID process controllers based on genetic algorithms.* IFAC 12th Triennial World Congress, 193-197.

[104] Weaver, A. C., 1990. *IECON'90, 16th Annual Conference of the IEEE Industrial Electronics Society.* Pacific Grove, California.

[105] Wehrens, R. and Lucasius, C. and Buydens, L. and Kateman, G., Mar-Apr 1993. *Sequential Assignment of 2D-NMR Spectra of Proteins Using Genetic Algorithms.* Journal of Chemical Information and Computer Sciences, **33**:(2), 245–251.

[106] Weigand, W. A., 1981. *Maximum Cell Productivity by Repeated Fed-Batch Culture for Constant Yield Case.* Biotechnology & Bioengineering, **28**, 249-266.

[107] White, S. R. *Concepts of Scale in Simulated Annealing.* In *Proceedings, IEEE International Conference on Computer Design, VLSI in Computers*, 646–651, Port Chester, N.Y., October 1984.

[108] Whitley, D. *A Genetic Algorithm Tutorial.* Technical Report CS-93-103, Colorado State University, 1993.

[109] Whitley, D. and Starkweather, T., 1990. *Optimizing small neural networks using a distributed genetic algorithms.* Proceedings of International Conference of Neural Networks.

[110] Zafiriou, E. and Morari, M., 1985. *Digital Controllers for SISO Systems: A Review and a New Algorithms.* International Journal of Control, **42**:(4), 855-876.

[111] Ziegler, J. G. and Nichols, N. B. *Optimum settings for automatic controllers.*

[112] Zurada, J. and Marks, R. and Robinson, C., 1994. *Computational Intelligence: Imitiating Life*. IEEE Press.