A MULTI-OBJECTIVE GP-PSO HYBRID ALGORITHM FOR GENE REGULATORY NETWORK MODELING

by

XINYE CAI

B.ENG., Huazhong University of Science & Technology, 2004
M.S, University of York, 2006

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2009

Abstract

Stochastic algorithms are widely used in various modeling and optimization problems. Evolutionary algorithms are one class of population-based stochastic approaches that are inspired from Darwinian evolutionary theory. A population of candidate solutions is initialized at the first generation of the algorithm. Two variation operators, crossover and mutation, that mimic the real world evolutionary process, are applied on the population to produce new solutions from old ones. Selection based on the concept of survival of the fittest is used to preserve parent solutions for next generation. Examples of such algorithms include genetic algorithm (GA) and genetic programming (GP). Nevertheless, other stochastic algorithms may be inspired from animals' behavior such as particle swarm optimization (PSO), which imitates the cooperation of a flock of birds. In addition, stochastic algorithms are able to address multi-objective optimization problems by using the concept of dominance. Accordingly, a set of solutions that do not dominate each other will be obtained, instead of just one best solution.

This thesis proposes a multi-objective GP-PSO hybrid algorithm to recover gene regulatory network models that take environmental data as stimulus input. The algorithm infers a model based on both phenotypic and gene expression data. The proposed approach is able to simultaneously infer network structures and estimate their associated parameters, instead of doing one or the other iteratively as other algorithms need to. In addition, a non-dominated sorting approach and an adaptive histogram method based on the hypergrid strategy are adopted to address 'convergence' and 'diversity' issues in multi-objective optimization.

Gene network models obtained from the proposed algorithm are compared to a synthetic network, which mimics key features of *Arabidopsis* flowering control system, visually and numerically. Data predicted by the model are compared to synthetic data, to verify that they are able to closely approximate the available phenotypic and gene expression data. At the end of this thesis, a novel breeding strategy, termed network assisted selection, is proposed as an extension of our hybrid approach and application of obtained models for plant breeding. Breeding simulations based on network assisted selection are compared to one common breeding strategy,

marker assisted selection. The results show that NAS is better both in terms of breeding speed and final phenotypic level.

A MULTI-OBJECTIVE GP-PSO HYBRID ALGORITHM FOR GENE REGULATORY NETWORK MODELING

by

XINYE CAI

B.ENG., Huazhong University of Science & Technology, 2004
M.S, University of York, 2006

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2009

Approved by:

Major Professor
Dr Sanjoy Das

Abstract

Stochastic algorithms are widely used in various modeling and optimization problems. Evolutionary algorithms are one class of population-based stochastic approaches that are inspired from Darwinian evolutionary theory. A population of candidate solutions is initialized at the first generation of the algorithm. Two variation operators, crossover and mutation, that mimic the real world evolutionary process, are applied on the population to produce new solutions from old ones. Selection based on the concept of survival of the fittest is used to preserve parent solutions for next generation. Examples of such algorithms include genetic algorithm (GA) and genetic programming (GP). Nevertheless, other stochastic algorithms may be inspired from animals' behavior such as particle swarm optimization (PSO), which imitates the cooperation of a flock of birds. In addition, stochastic algorithms are able to address multi-objective optimization problems by using the concept of dominance. Accordingly, a set of solutions that do not dominate each other will be obtained, instead of just one best solution.

This thesis proposes a multi-objective GP-PSO hybrid algorithm to recover gene regulatory network models that take environmental data as stimulus input. The algorithm infers a model based on both phenotypic and gene expression data. The proposed approach is able to simultaneously infer network structures and estimate their associated parameters, instead of doing one or the other iteratively as other algorithms need to. In addition, a non-dominated sorting approach and an adaptive histogram method based on the hypergrid strategy are adopted to address 'convergence' and 'diversity' issues in multi-objective optimization.

Gene network models obtained from the proposed algorithm are compared to a synthetic network, which mimics key features of *Arabidopsis* flowering control system, visually and numerically. Data predicted by the model are compared to synthetic data, to verify that they are able to closely approximate the available phenotypic and gene expression data. At the end of this thesis, a novel breeding strategy, termed network assisted selection, is proposed as an extension of our hybrid approach and application of obtained models for plant breeding. Breeding simulations based on network assisted selection are compared to one common breeding strategy,

marker assisted selection. The results show that NAS is better both in terms of breeding speed and final phenotypic level.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I am very grateful to my advisor, Dr. Sanjoy Das, for his inspiring guidance, patient support and constant encouragement throughout my research. I am indebted to my co-advisor Dr. Stephen Welch, for his kindness and valuable advices. I would also like to express my gratitude to Dr. Praveen Koduru, who contributed his knowledge in programming and parallel computing during my research, and shared his experience to advice on how to adapt myself to PhD life. I also wish to thank all my committee members and outside chairman, Dr. Chris Lewis, Dr. Dwight Day, Dr. Doina Caragea and Dr. Alexander Beeser for their valuable suggestions on my work.

Next, I would like to give my thanks to Sharon Hartwich and Angie Pfizenmaier, for their specific and detailed help in filling out paperwork. I would also like to thank the staff of Electrical Engineering Department, Graduate School and library services at Kansas State University who provides me both their time and professional knowledge. Special gratitude to all my friends with whom I had great experience in KSU.

Last, but not least, I would like to thank my family, for their understanding and constant support, without which I would not be able to pursue my dream and complete this work.

Dedication

To my parents, grandparents, uncles and aunts,

For their constant support, encouragement and love.

CHAPTER 1 - Overview of Stochastic Multi-objective Optimization

Optimization is a popular research field with many important applications [1]. The aim of optimization is to find the best solutions to a given problem within a set of constraints. Classical optimization approaches mainly focus on single objective problems and aims to find the best possible solution, usually termed *global optimum*. Most real-world applications, however, involve the simultaneous optimization of more than one objective. Furthermore, it is not unusual to encounter situations where those objectives are in conflict with each other. For example, when designing a building, the architect always would like to minimize its cost while having its safety maximized. In these multi-objective problems, mathematical and algorithmic tools that are different from those for single-objective optimization are required. In fact, the notion of *optimality* changes when dealing with multi-objective optimization problems. We need to find the best tradeoff among the objectives.

Among several heuristics currently available, *stochastic algorithms* (SAs) such as evolutionary algorithms and particle swarm optimization approaches are among the most popular [2] [3]. These algorithms are a class of approaches that are inspired from natural metaphors. Stochastic algorithms have been popular in single-objective optimization and, more recently, also have applied to multi-objective problems. In this chapter, an overview of *stochastic multi-objective optimization* (SMO) is provided. It includes the basic concepts of multi-objective optimization, advantages of stochastic approaches and relevant most popular algorithms.

1.1 Basic Concepts

1.1.1 Multi-objective Optimization

There are three aspects that are important to note in the context of SMO. First, a *multi-objective optimization problem* (MOP) always has two or more objectives that are required to be optimized simultaneously. Second, there may be constraints imposed on the objectives. Third, objectives in MOP are usually in conflict with each other; otherwise, a single solution may exist which may be obtained by optimizing the objectives in sequential order. A MOP can be defined as:

Definition 1: Multi-objective optimization problem

Given a problem involving *N* variables $x_1, x_2, ... x_N$ in a *search space* $\boldsymbol{X} \subset \Re^N$, we assume, without loss of generality, *M* objectives $f_1(.),...,f_M(.)$ in *objective function space* $\boldsymbol{Y} \subset \Re^M$, are to be minimized.

Minimize $\boldsymbol{f}(\boldsymbol{x}) = (f_1(x_1, x_2,..., x_N),....f_M(x_1, x_2,..., x_N))$

The vector function is a mapping $\boldsymbol{f} : \boldsymbol{X} \rightarrow \boldsymbol{Y}$.

1.1.2 Pareto Optimality

In a MOP, because multiple objectives are involved, it is usually not possible to find a single solution which is optimal for all the objectives. Instead, many good solutions may exist. These solutions are always “trade-offs” or good compromises among the objectives. Since the conventional concept of optimality does not hold, a concept of Pareto optimality is adopted. Before formally defining Pareto optimality, we introduce the concept of dominance.

Definition 2: Dominance

Let $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}$ be two vector inputs. We say $\boldsymbol{x}$ dominates $\boldsymbol{y}$ (written as $\boldsymbol{x} \prec \boldsymbol{y}$) iff they satisfy the conditions:

$$\begin{array}{l} \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}, \forall\, i \in \{1,...,M\} \mid f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) \\ \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}, \exists\, j \in \{1,...,M\} \mid f_j(\boldsymbol{x}) < f_j(\boldsymbol{y}) \end{array}, \qquad \text{Eq. 1.1}$$

On the contrary, a solution $\boldsymbol{x}$ is considered to be a non-dominated solution iff there is no other solution that satisfies equation 1.1. The set of all non-dominated solutions form a *Pareto set.*

Definition 3: Pareto front

The projection of the Pareto set $\boldsymbol{P}$ in the *M* dimensional objective function space $\boldsymbol{Y}$ is called *Pareto front*, $\boldsymbol{F}$. [1]

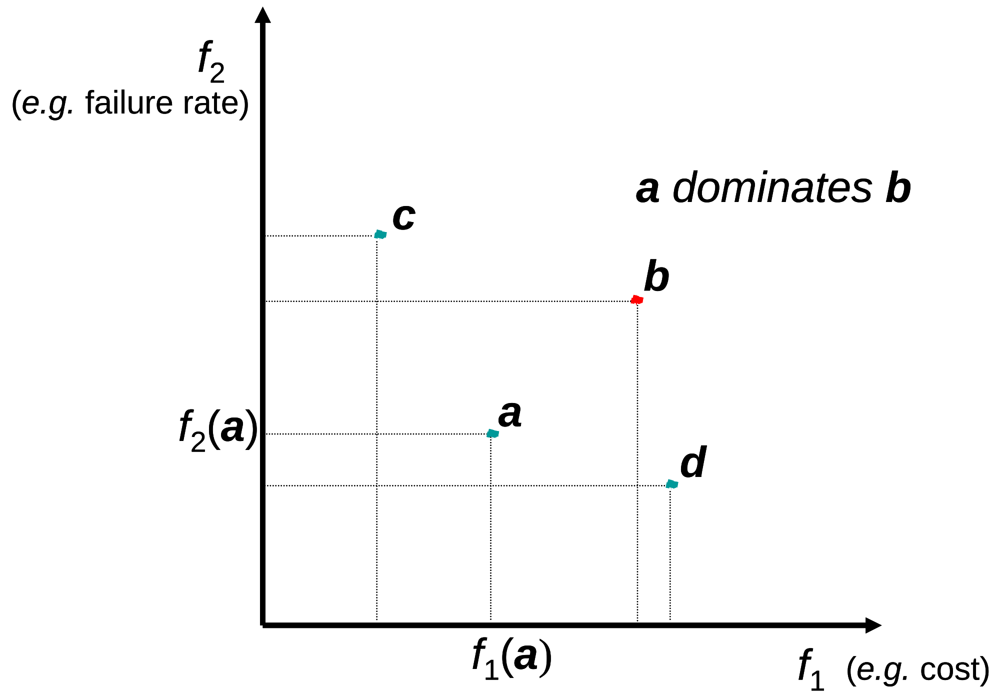
$$\boldsymbol{F} = \{(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}),...f_M(\boldsymbol{x})) | \boldsymbol{x} \in \boldsymbol{P}\}, \qquad \text{Eq. 1.2}$$

Figure 1.1 explains the concept of dominance. Fox example, when considering a building design project; architects need to minimize two objectives: cost and failure rate. Four possible schemes – ***a***, ***b***, ***c*** and ***d*** exist. According to definition 2, solution ***a*** dominates solution ***b***. However,

[1] To distinguish from a true Pareto front, we call non-dominated solutions that are not in the true Pareto front discovered so far by SAs a non-dominated front.

solutions ***a***, ***c*** and ***d*** do not dominate each other. Figure 1.2 shows a two-dimensional Pareto front according to definition 3.

Figure 1.1 Dominated and non-dominated solutions in two dimensions



1.1.3 Performance Measurements for Multi-objective Optimization

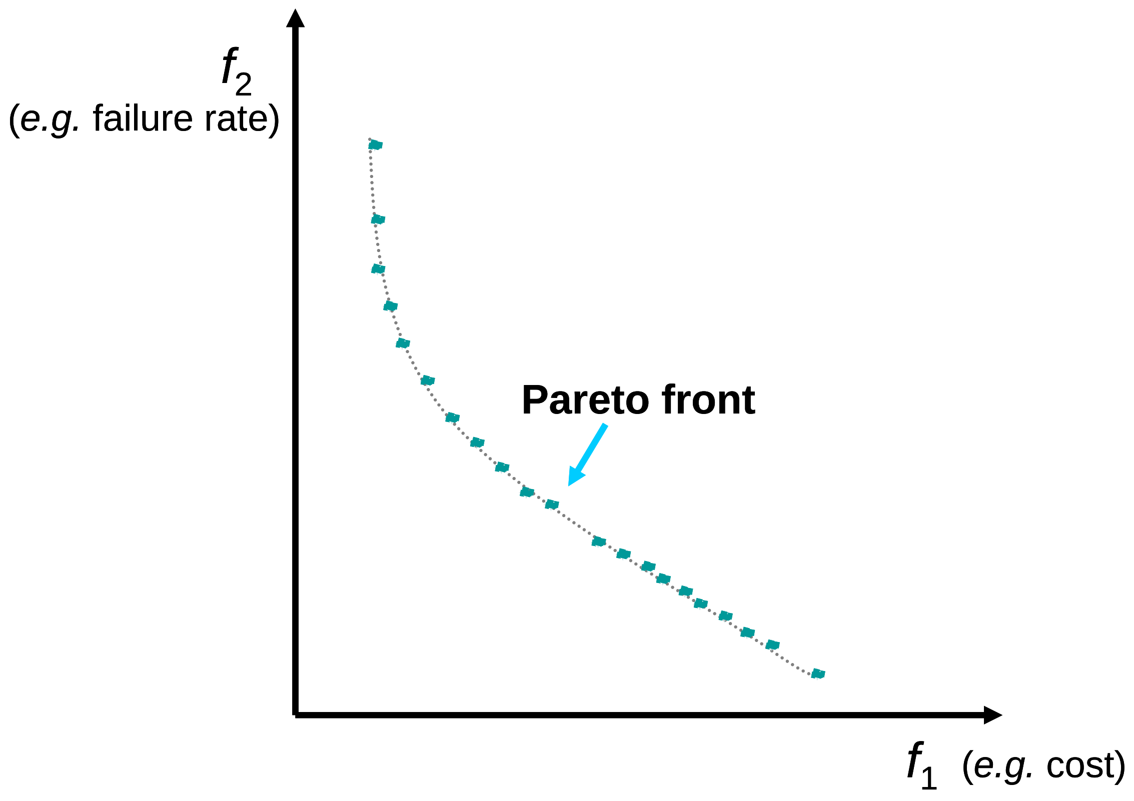
In order to test the performance of different algorithms in MOP, measures to allow a quantitative comparison of results are required. A variety of performance measures have been proposed [4][5][6]. Three goals have been summarized for a good SMO algorithm [5]:

1. The size of the non-dominated front should be maximized.
2. The non-dominated front found by the SMO algorithm should be as close as possible to the true front.
3. The solutions should be as uniformly distributed as possible.

The second and third goals are called *convergence* and *diversity* [7]. They are detailed in the following sections.

When considering MOP, the true Pareto front is usually not known beforehand in a real application. So test functions with known Pareto front are used to test the efficiency of the algorithms. ZTD functions are one class of the most popular test functions [8] [9] [10].

Figure 1.2 Two-dimension Pareto front



1.2 Stochastic Optimization

Since the 1950s, a variety of mathematical programming techniques have been developed to address MOP [11] [12]. Nevertheless, they all may have one or several of the following limitations, *(i)* prior knowledge of the true Pareto front is required; *(ii)* they may not work when the Pareto front is concave or disconnected -- they require differentiability of the objective functions and the constraints; *(iii)* they can only obtain one solution from each run.

Most stochastic algorithms are inspired from biology. For example, *evolutionary algorithms* (EAs) are inspired from biological evolution and *particle swarm optimization* (PSO) is inspired from the cooperative behavior of birds. In such algorithms, a solution candidate is sometimes called an *individual* and the set of solution candidates forms a *population.* There are four main features in SAs:

(i) an individual, *e.g*. decision vector, represents a solution to the given problem,

(ii) according to objective functions, each individual is evaluated and assigned a fitness to reflect the quality of the solution,

(iii) a selection process is performed on the population and

(iv) the population is updated and new solutions are generated in each generation

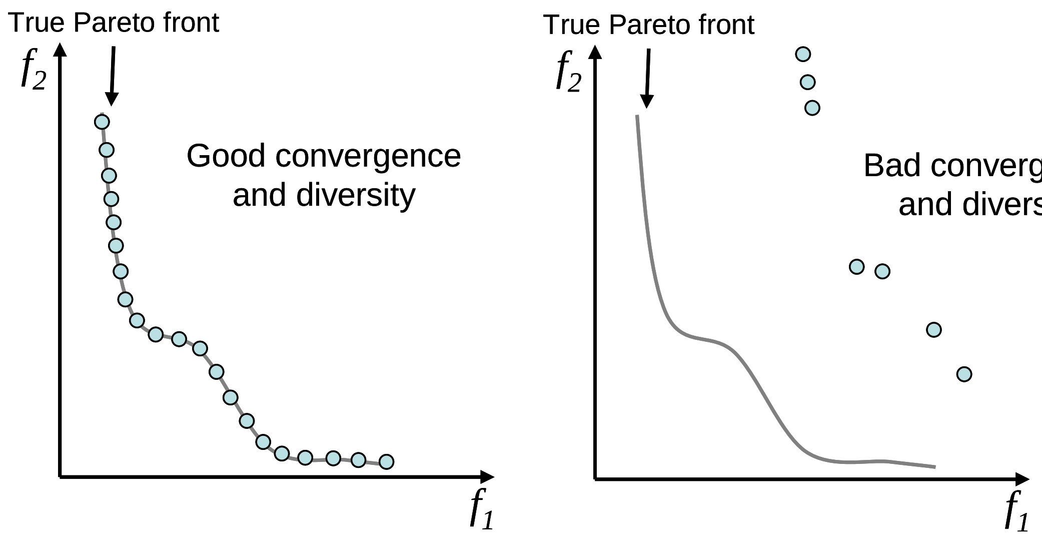
The above features enable SAs to address the limitations of classic multi-objective approaches. Firstly, SAs do not need any prior knowledge of the Pareto front. Secondly, being *population-based algorithms*, they are able to generate a non-dominated front in a single run.

1.3 Algorithm Design Issues

1.3.1 Convergence and Diversity

In an evolutionary algorithm, either convergence or diversity must be used as a criterion to discriminate between solutions and form a non-dominated front. Convergence is a term used to describe how close that the set of obtained non-dominated solutions in the population is to the true Pareto front. In addition to good convergence, another feature, termed diversity, used to describe the proper space intervals between solutions in the non-dominated front or Pareto front, is equivalently important. Usually, it is more desirable to have the distribution of the solutions spread out. In other words, an evenly populated Pareto front has good diversity. Figure 1.3 (left) shows a Pareto front with good convergence and diversity. In comparison, Figure 1.3 (right) illustrates bad convergence and diversity.

Figure 1.3 An illustration of good convergence and diversity (left) and bad convergence and diversity (right)



1.3.2 Techniques for Convergence

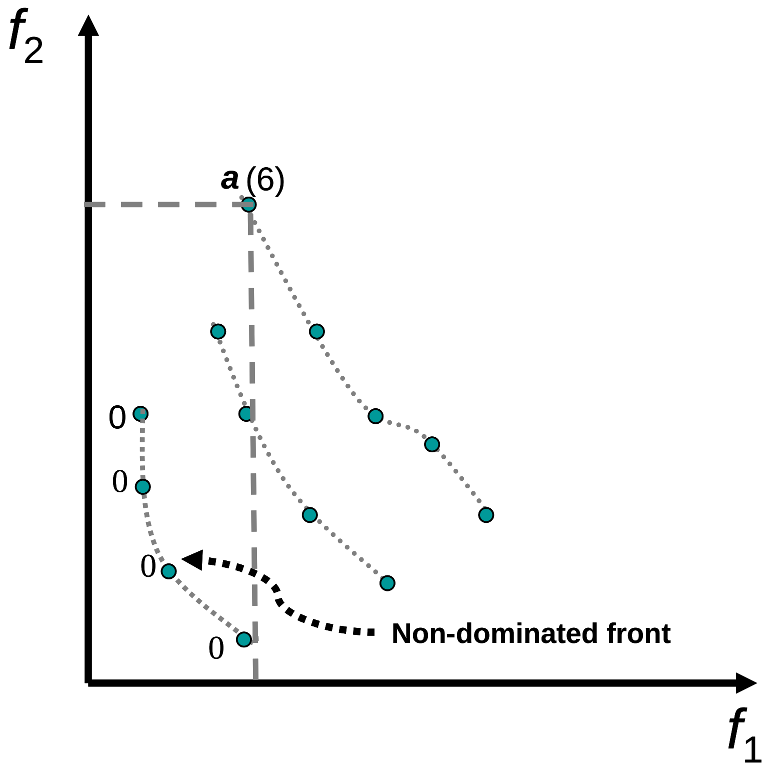
Techniques for convergence in SMO generally consist of *aggregation-based*, *criterion-based* and *Pareto-based* methods [13].

Aggregation-based methods turn a multi-objective function into a single parameterized objective function. The parameters of this function are varied to find a set of non-dominated solutions. Weighted-sum aggregation is a good example where the weight-parameter is changed during optimization process.

Criterion-based methods switch among objectives during the selection and variation process of SAs with a certain probability. Each time an individual is chosen for variation, the fitness value of the individual to a different objective will decide if it can be selected for variation.

The third method is based on the concept of Pareto dominance [3]. It can be divided into mainly two subcategories depending on how to rank population. The first ranking approach is usually referred to as *domination counting*. In this method, the rank of any solutions within a population of solutions is determined by the number of other solutions that dominates this solution. For example, in Figure 1.4, solution ***a*** is dominated by six other solutions, contained in dashed lines. Solutions with the counts of zero are assigned as non-dominated solutions and form the non-dominated front, as shown in Figure 1.4. Dominance counting is used in both SPEA[6] and SPEA2[18]. The other method is called *non-dominated sorting*. In this method, solutions of the same rank are the ones that do not dominate each other and the ones that have been dominated are assigned to lower rank. This method is used in NSGA II [14]. Dotted grey lines in Figure 1.4 show different ranks after applying non-dominated sorting. Among the three grey dotted lines, the one closest to origin forms the non-dominated front.

Figure 1.4 An illustration of domination counting and non-dominated sorting methods for convergence in 2-dimensional objective function space



1.3.3 Diversity Preservation

Diversity preservation is critical in a SMO design because in each generation, one tries to avoid identical or very similar solutions in the population. Solutions in the sparser region are favored to control the density of solutions in the objective function space. Different techniques that incorporate density information into selection process have been developed. There are mainly four diversity preservation strategies which are referred to as *fitness sharing, crowded comparison*, *histogram* and *nearest neighbor*, respectively.

The fitness sharing method uses the well known sharing function [16]:

$$f(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{share}} \right)^a, \text{if} \quad d_{ij} < \sigma_{share} \\ 0, \qquad\qquad \text{otherwise} \end{cases}, \qquad \text{Eq. 1.3}$$

where d_{ij} is the Euclidean distance between two neighboring solutions ***i*** and ***j*** and σ_{share} is a user defined niche radius parameter. Each solution ***i*** within others' niche radius will have its fitness degraded. The degradation function is as follows:

$$fitness_i = \frac{fitness_i}{c_i}, \qquad \text{Eq. 1.4}$$

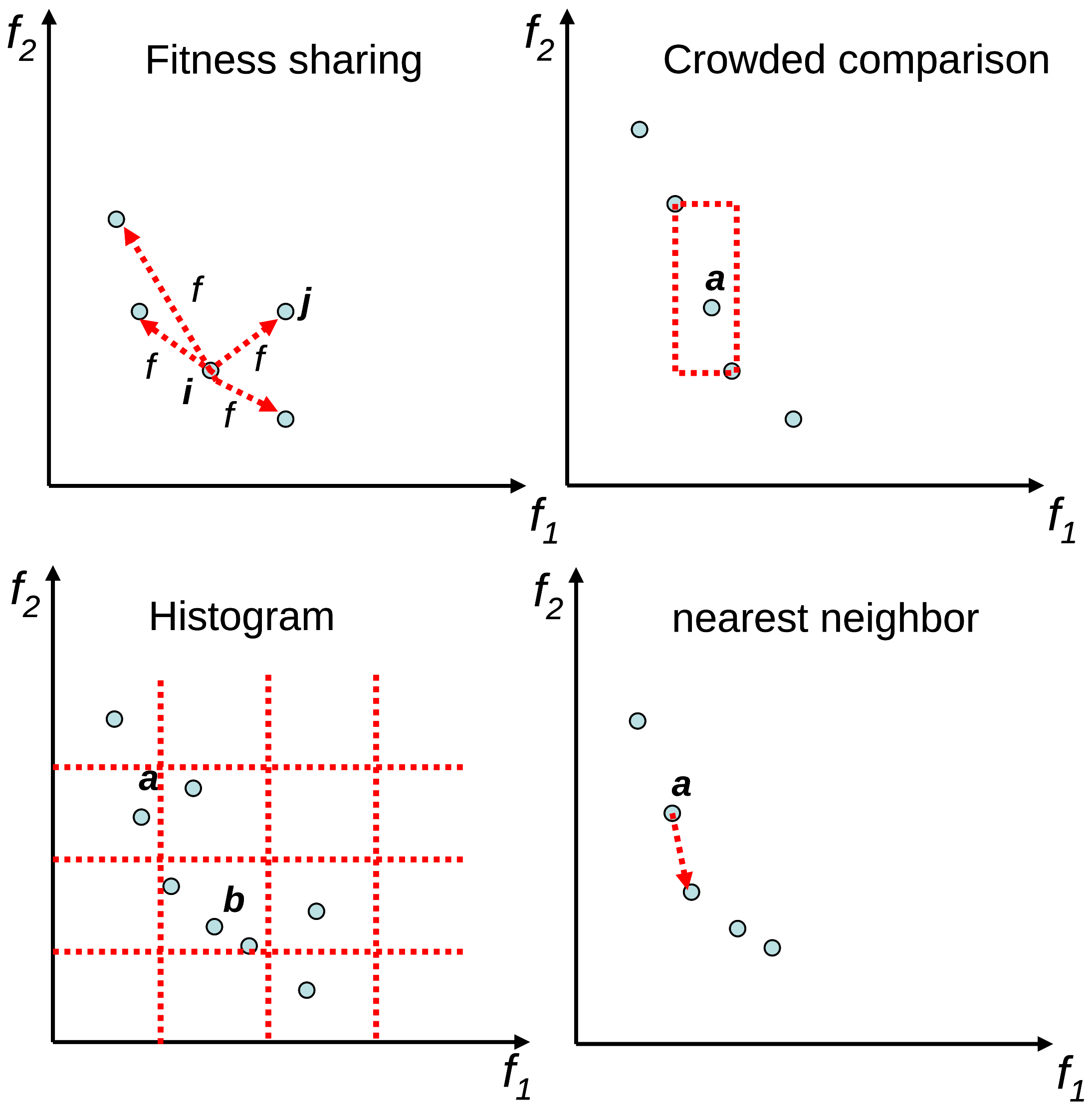
where c_i is called the niche count for solution ***i***:

$$c_i = \sum_{j=1}^{N} f(d_{ij}), \qquad \text{Eq. 1.5}$$

The fitness sharing method is demonstrated in Figure 1.5 (top, left). The solution niche count of solution ***i*** is the sum of the sharing functions to each of other solutions within the niche radius σ_{share}. The sharing function is actually one type of *kernel function* whose value represents the density estimate for the solution. In the above equations, niche count c_i (the sum of the sharing function for solution ***i***) represents density estimate for solution ***i***. Also, this density information is integrated in the selection process by the using fitness degradation function, where a solution with crowded neighbors will have its fitness degraded and is less likely to be selected over the optimization process. Fitness sharing has been widely used in SMO, such as MOGA[17], NSGA[20] and NPGA[19].

This diversity preservation approach, however, is criticized for the requirement of specifying the sharing parameter σ_{share} [14]. So a method based on crowded comparison for diversity preservation is proposed in NSGA II [14]. To estimate the density of solutions surrounding a particular solution in the population, the average distance of two points on either side of this point along each of the objectives is calculated. The value of this average distance contains density information, where larger value indicates the solution is located in a sparser region. It is worthy to note that the boundary solutions (solutions with either smallest or largest objective function values) are usually preferred in the optimization process; this can be accomplished by assigning an infinite distance value to them. Figure 1.5(top, right) illustrates how the distance value of solution ***a*** is calculated in the crowded comparison method.

Figure 1.5 An illustration of different diversity preservation approaches in MOP



The histogram method divides a *M* dimensional objective function space into small *M* dimensional hypergrids. Naturally, the number of solutions located in each hypergrid's cell determines the density of solutions in the regions. In Figure 1.5 (bottom, left), the grid where solution ***a*** is located is less dense with solutions than that of solution ***b***; so solution ***a*** is more likely to be favored. The hypergrid can be fixed, though usually it is adapted dynamically in each generation with the population of solutions. PAES [15] and MOPSO[CJM2004] have adopted this approach.

In the last method, nearest neighbor, each solution calculates its total distance from the nearest *K* neighboring solutions, as shown in Figure 1.4 (bottom right). A larger distance value suggests the solution is likely to locate in a sparsely populated region. A good example of its application is in SPEA2 [18] where the algorithm calculates for each solution the inverse of the distance to K^{th} nearest neighbor and adds it to the raw fitness value.

1.3.4 Elitism

Elitism is another important concept in stochastic multi-objective optimization. During the optimization process, sometimes good solutions are lost due to random effects. One possible way to cope with this problem is to use a deterministic selection operator on the combined population of parent and offspring, instead of replacing the parent population with the offspring. Another alternative is the use of archiving – a secondary population which maintains promising solutions. Most SMO uses the combination of both dominance and diversity information to select diverse non-dominated solutions to store into the archive.

1.4 Motivation of Future Research

Multi-objective optimization has been rapidly developing and expanding in recent years. Several interesting new approaches for optimization have been proposed in recent literature. A good example is the emerging of a class of artificial immune system (AIS) based multi-objective algorithms [22]. One of the urgent problems to be solved is to fit suitable approaches into different application domains. Innovation and hybridization of algorithms are also necessary when facing different applications.

CHAPTER 2 - Stochastic Approaches and Stochastic Multi-objective Optimization

2.1 Introduction

In Chapter 1, the concept of Pareto optimality for multi-objective optimization has been presented. In this chapter, we will introduce stochastic approaches, such as genetic algorithms (GAs), genetic programming (GP) and particle swarm optimization (PSO), all of which are widely used in both single objective and multi-objective optimization problems. At the end of this chapter, a survey of recent popular stochastic multi-objective algorithms is also provided.

2.2 Genetic Algorithms

Genetic algorithms are stochastic optimization approaches which mimic representation and variation mechanisms borrowed from biological evolution, such as *selection*, *crossover*, and *mutation* [23][24]. In this approach, a GA candidate solution is represented as a linear string analogous to a biological *chromosome*. The general scheme of GAs starts from a population of randomly generated candidate solutions (chromosomes). Each chromosome is then evaluated and given a value which corresponds to a *fitness* level in objective function space. In each generation, chromosomes are chosen based on their fitness to reproduce offspring. Chromosomes with a high level of fitness are more likely to be retained while the ones with low fitness tend to be discarded. This process is called *selection*. After selection, offspring chromosomes are constructed from parent chromosomes using operators that resemble crossover and mutation mechanisms in evolutionary biology. The crossover operator, sometimes called *recombination*, produces new offspring chromosomes that inherit information from both sides of parents by combining partial sets of elements from them. The mutation operator randomly changes elements of a chromosome with a low probability. Over multiple generations, chromosomes with higher fitness values are left based on the survival of the fittest.

2.3 Genetic Programming

2.3.1 Introduction

Genetic programming is a subclass of GAs, solutions of which are expressed as structures. GP has been successfully applied to system modeling and structure discovery. Within genetic programming, the process of problem solving is regarded as a search in the objective function space. Similar to GAs, basic elements in GP include representation, selection, crossover and mutation.

Representation is critical in GP because the search of GP is not performed directly in the objective function space, but rather in a representation space. With the same objective function space, the selection of a different representation may result in a different search space. Thus representation in GP plays an important role in the effectiveness of its algorithm. Many GPs use a graph to represent a topology directly. In addition, such graph can be encoded as a string of numbers which makes it easier for an algorithm to handle.

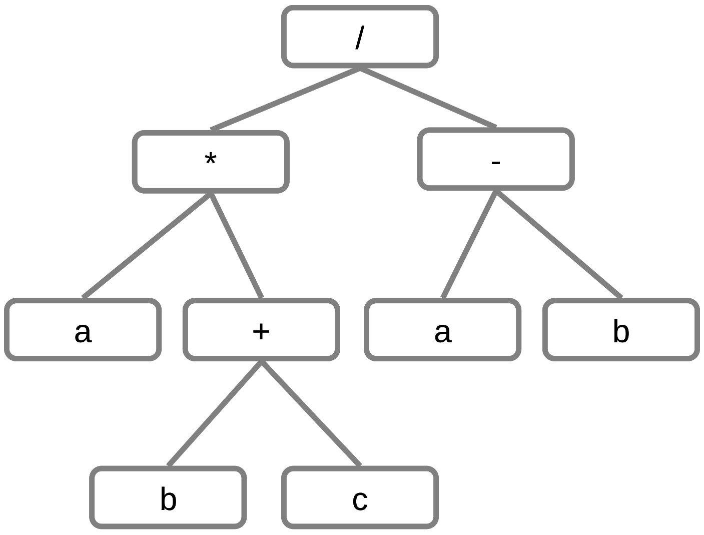
The selection operator is used to determine which individuals from parent chromosomes and their offspring will form the new generation. The whole search process actually implies a compromise between two contradictory requirements: *exploitation* of the best available solution and robust *exploration* of the search space. The selection operator is a critical means to maintain balance between exploitation and exploration.

Crossover and mutation operators are essential for the search process. The crossover operator is applied to two parent chromosomes and combine parts from each parent to create offspring chromosomes. The mutation operator is applied to one individual by changing parts of the chromosome at a very low rate.

2.3.2 Conventional Genetic Programming

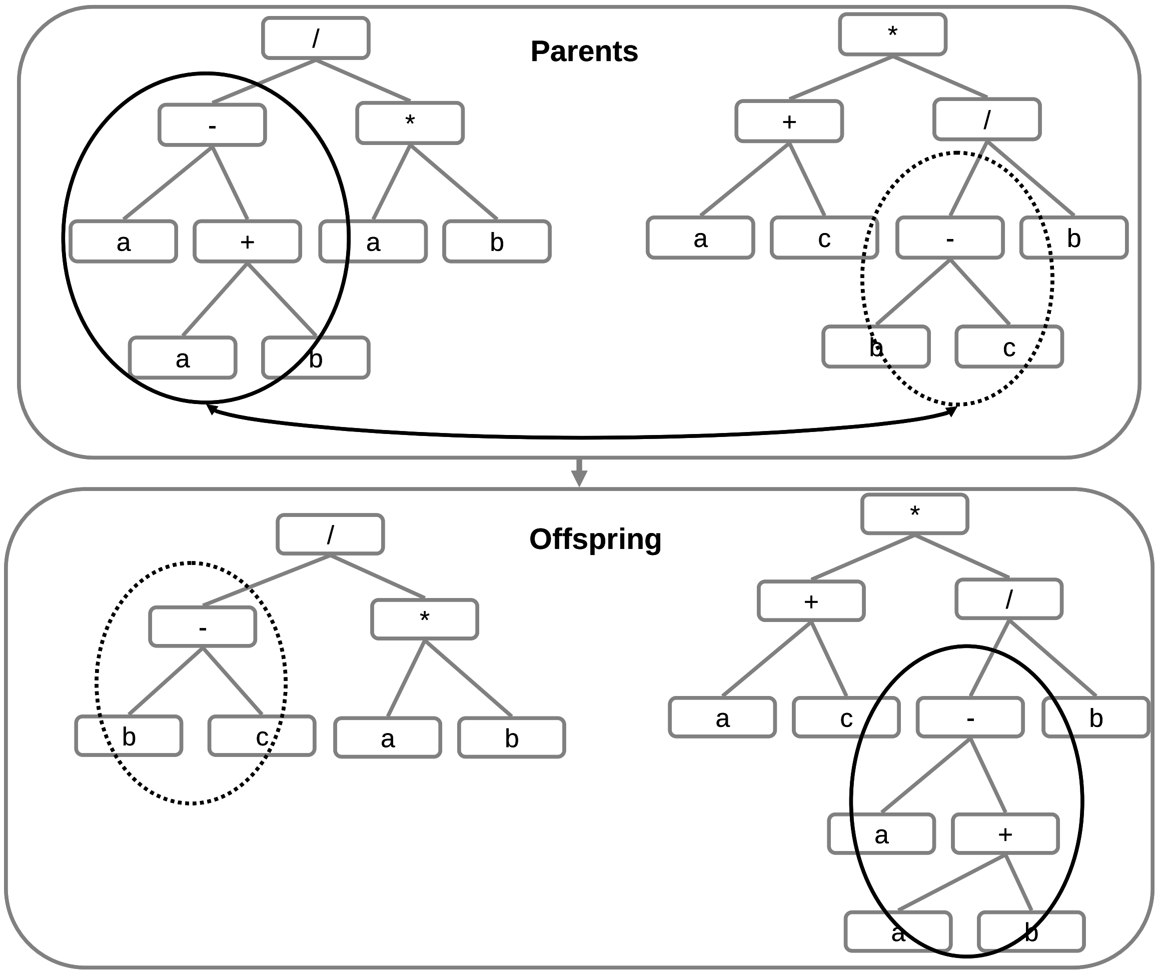
Genetic programming was originally devised by Koza[25]. In his original design, a solution consisting of *functions* and *terminals* appropriate to the problem space is represented as hierarchical tree. For example, a simple expression $\frac{a*(b+c)}{a-b}$ is represented as shown in Figure 2.1. In this example terminal sets $\boldsymbol{T} = \{a, b, c\}$and function sets $\boldsymbol{F} = \{'+', '-', '*', '/'\}$.The internal nodes of the tree structure are entries from the function set and leaf nodes are input data from the terminal set. Also a genetic programming tree and its corresponding expression can equivalently be represented in *prefix notation*, where functions always precede their terminals. In our example, expression $\frac{a*(b+c)}{a-b}$ is equivalent to $(/(*\,a(+\,bc))(-\,ab))$ in prefix notation.

Figure 2.1 Conventional hierarchical tree representation in GP



There is a probability that crossover is used for swapping the sub-trees in two separate chromosomes. This probability is called *crossover rate*. Also, a single point mutation operator is usually applied to chromosomes by randomly changing parts at a certain rate, called *mutation rate*. A conventional GP representation and its crossover process are shown in Figure 2.2.

Figure 2.2 Tree-based representation and crossover process in conventional GP



Conventional GP, however, has its shortcomings [26] [27] [25] [28] [29] [30] [31]. Firstly, tree structure based representation loses GP's generality to represent other different computational structures. Secondly, *bloat* problem can be observed in conventional GP. Bloat is used to describe the phenomenon that solutions have the tendency to become larger and exhaust computational resources. When bloat problem occurs, it is nearly impossible to find a small and efficient solution. A lot of effort has been made to improve conventional GP [43] [33] [34] [35]. An improved form of genetic programming is introduced in the following section.

2.3.3 Cartesian Genetic Programming

Cartesian genetic programming (CGP) [32] is an alternative graph-based form of genetic programming. A graph-based representation gives GP great generality so that it can represent neural networks, programs, circuits, networks and many other computational structures. Actually, we can consider tree structure as a special form of graph in which two nodes must have one and only one path between them (a path is a sequence of connected nodes). Also, graphs are more compact than the usual tree representation since subgraphs can be used more than once.

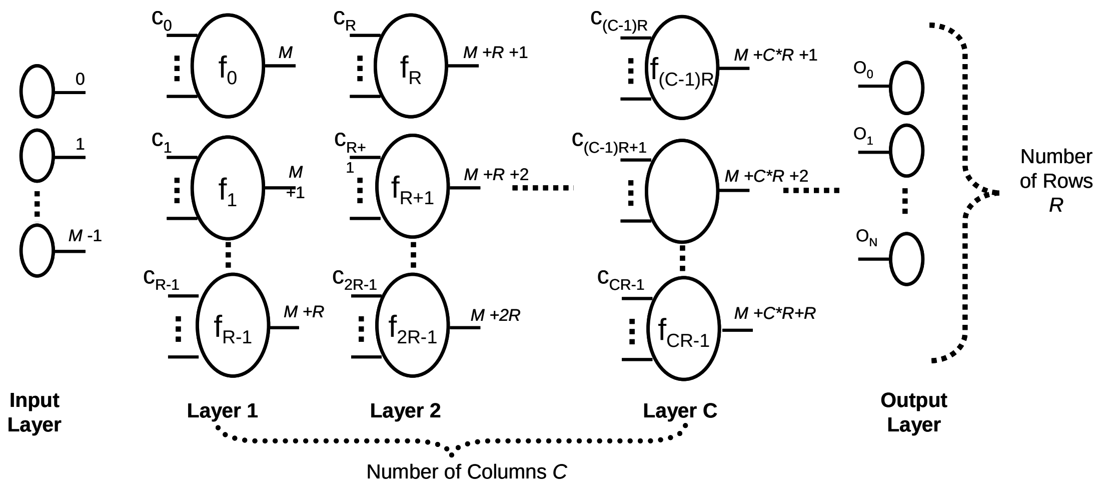
In CGP, chromosomes are encoded as a list of integers that represent the functions and connections between graph nodes and program inputs and outputs. CGP is loosely inspired from FPGAs (field programmable gate arrays) to evolve digital circuits. Its original form is represented as a group of Cartesian grids arranged in layers, mimicking the architecture of digital circuits. A representation of CGP can be seen in Figure 2.3. From the figure, it is clear to see that there are three types of layers: input layer, output layer and main layers. The input layer is on the far left of the figure with *M* input nodes, while the output layer is on the far right of the figure with *N* output nodes O_i. In between them are main layers, where each node is specified in order by a number of rows *R* and columns *C*. The nodes in the same column are not allowed to be connected, but they can connect to the nodes in the previous columns. A parameter called *levels-back* is used to define the number of columns back a node in a particular column can connect to.

The Cartesian representation can be encoded as a string of integers.

$\boldsymbol{c}_0, f_0; \boldsymbol{c}_1, f_1; ... \boldsymbol{c}_{CR-1}, f_{CR-1}; O_0, O_1 ..., O_N$

$\boldsymbol{c}_i$ denotes a vector of points in which the inputs of the node are connected. Each node also has a function; f_i represents a function which is listed in predefined function table. It also has *N* output genes O_i that denote the points where the *N* program outputs are taken from. Inputs of node $\boldsymbol{c}_i$ are restricted so nodes can only have their inputs connected to either program inputs or nodes from a previous (left) column. Function values are restricted to those available. In CGP, only the mutation operator is applied to the representation. During this process, a percentage of integers in the representation are changed to another randomly selected value. But restrictions described above must be strictly abided by.

Figure 2.3 An example of CGP representation and its encoded integer string form



Encoded representation Strings $c_0, f_0; c_1, f_1; ...c_{CR-1}, f_{CR-1}; O_0, O_1 ..., O_N$

In many implementations of CGP, the number of rows is set to one. Accordingly, the number of columns then becomes the maximum allowed number of nodes. Also, the levels-back parameter theoretically can be set to be any integer from one (in which case, nodes can only connect to the nodes in the column just prior to the current column this node is particularly located) to the maximum number of nodes (in which case a node can connect to any previous node). In order to give more flexibility to the CGP structure, in practice, the level-back is usually set to latter.

A simple model *(a + b) * c – c* represented in CGP with its encoded strings is given in Figure 2.4. The available functions include +, -, *. Although each node must have a function and a set of inputs for that function, the output of the node does not have to be used by other downstream nodes. Node 5 in Figure 2.4 is such a node. In other words, a node may not appear in the model, even though it exists in the representation. Such nodes are *redundant* or form *redundancy* in GP representation. These redundant nodes are inactive and have a neutral effect (termed *neutrality*) on fitness, because in later generations, they may be activated by the mutation operator. In Figure 2.5, after mutation changes a part of GP representation as shown in Figure 2.4, the whole model has been changed accordingly. Previously redundant node 5 has been activated. Similarly, formerly active nodes can be deactivated by mutation.

Figure 2.4 A simple model representing in CGP. The node 5 is a redundant node, since its output does not connect to any other downstream nodes. It also does not appear in the encoded strings or the model.

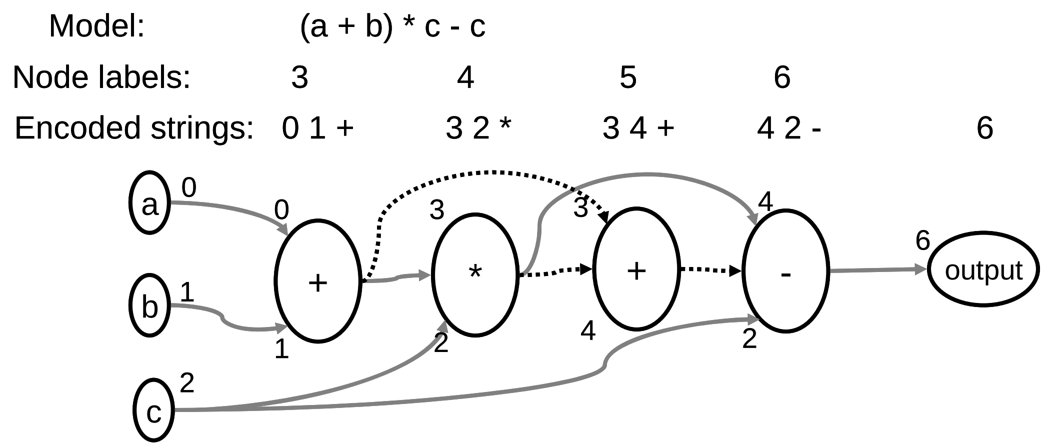
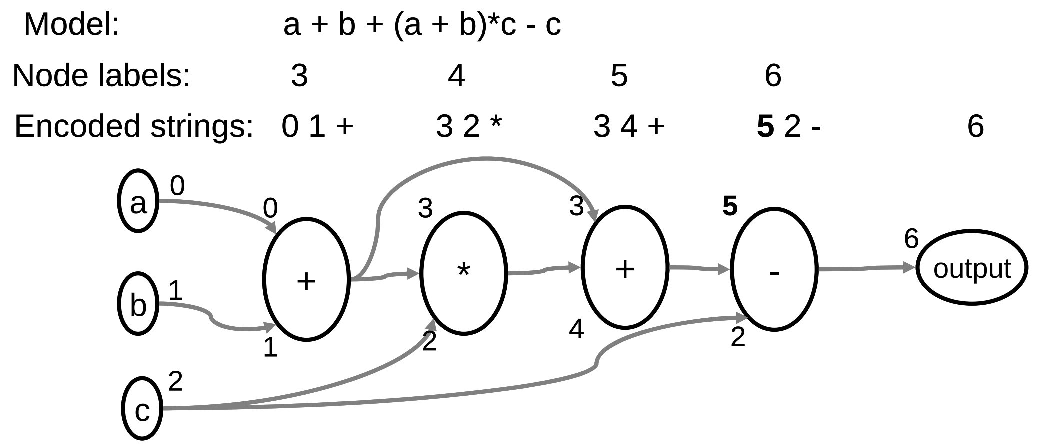


Figure 2.5 The CGP representation of new model is shown after mutation from the model in figure 2.3. Node 5 has been activated by mutation.


Model: a + b + (a + b)*c - c
Node labels: 3 4 5 6
Encoded strings: 0 1 + 3 2 * 3 4 + 5 2 - 6
a
b
c
0
1
2
+
*
+
-
output
0
1
3
2
3
4
5
2
6

2.3.4 Comparison of CGP with Conventional GP

As described earlier, bloat is one of the most serious drawbacks of genetic programming [26] [27]. Contrary to the tree structure based GP, CGP does not have a bloat problem. This is very likely owed to its pre-determined fixed number of nodes, and also the existence of the redundant nodes which could be activated or deactivated by mutation operators [36].

Redundancy is one distinctive feature of CGP. Redundancy indicates a large number of components which are not active in an individual in CGP. Nevertheless, they may become active during the evolutionary process. The high proportion of redundant nodes contributes to good performance for CGP. According to previous work with regard to CGP, it is most effective when the level of redundancy reaches 95% [37]. *Reuse* of subgraphs is another prominent feature of CGP. It makes graph-based CGP more compact than the usual tree representation. For example, node 3 in Figure 2.4 and Figure 2.5 is used by both node 4 and node 5.

CGP has been applied to a considerable number of fields: digital circuit design [38] [39], digital filter design [40], image processing [41], artificial life [42], bio-inspired developmental models [43] [44] [45] and molecular docking [46] [47]. CGP has also been adopted and hybridized within new evolutionary techniques such as cell-based optimization [48] and social programming [49].

2.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is an emerging stochastic, population based optimization approach [50] [51]. The techniques have evolved drastically since they were created and have been widely applied as a stochastic optimization approach to various fields.

In PSO, a solution is represented as a *particle*. Particles fly in the search space guided by their individual experience and the experience of the whole population. Each particle is actually a vector corresponding to a unique position (solution) in the search space. In addition, each particle is also associated with a velocity which is responsible for the motion of the particle. At the beginning of the algorithm, both the particles and their associated velocities are generated randomly. Over each generation, each particle’s position as well as velocity is to be updated until satisfactory solutions are found. A detailed algorithm as to how particles and velocities are updated will be given in Section 4.4.6.

2.5 A Survey of Stochastic Multi-objective Algorithms

2.5.1 Multi-objective GAs/GPs

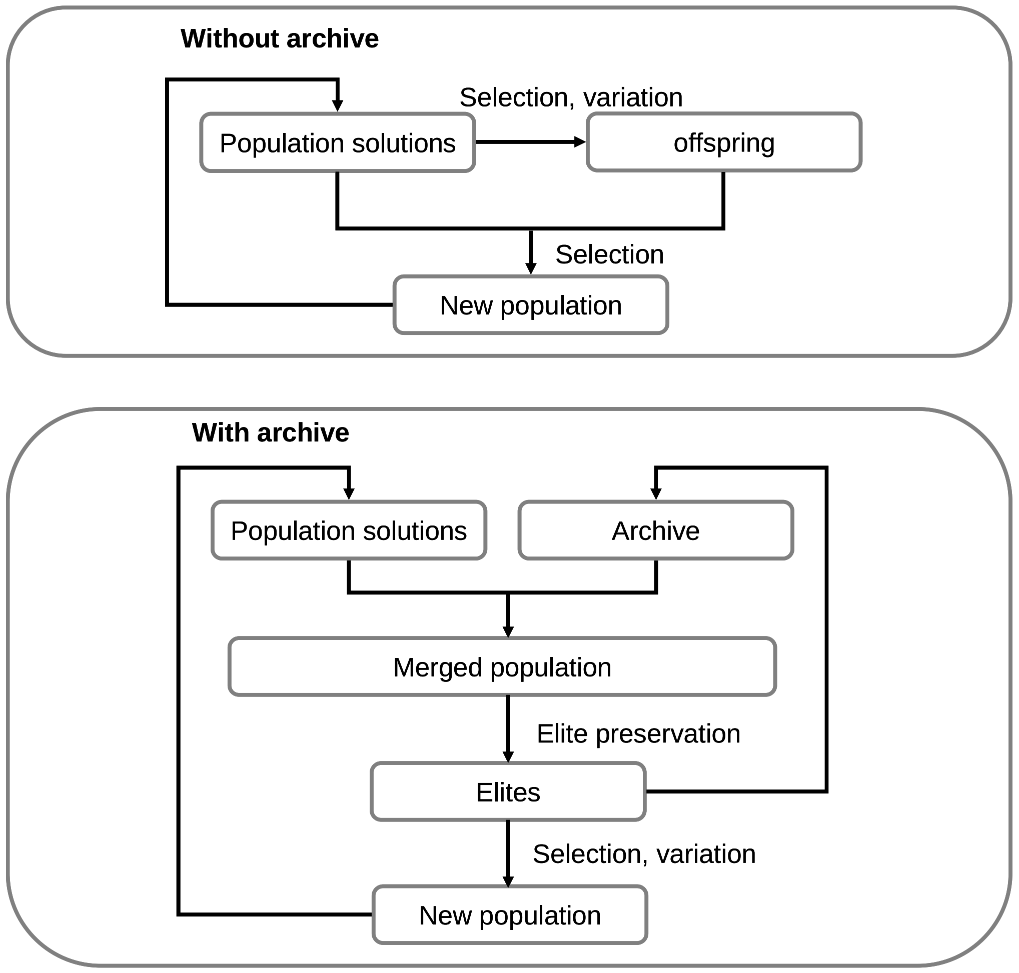
Multi-objective GAs are the most sophisticated among all sorts of stochastic multi-objective algorithms. The most representative multi-objective GAs include NPGA [19], NSGA [20], NSGA II [14] and PAES [15]. All of them have adopted the techniques for convergence and diversity described in Chapter 1.

Multi-objective GPs are relatively new. The inspiration for multi-objective GPs originates from multi-objective GAs. Another motivation of multi-objective GP is to overcome bloat in GPs (bloat has been detailed in previous sections), since the size of the model could be considered as another objective [53] [54]. The main techniques of multi-objective GPs with regard to achieving convergence and diversity are generally similar to those of multi-objective GAs. In fact, many structures of multi-objective GAs have been mapped onto GPs by introducing GPs representation with its associated genetic operators. However, the mainstream applications of multi-objective GPs are usually identifying models from huge output-input data.

Rodriguez-Vazquez *et al*. applied multi-objective GPs on identifying the structure of a nonlinear dynamic system [55]. The Pareto based method has been used as well as a tree-based GPs representation in his approach. Parrott *et al*. designed a multi-objective GP-based classifier [DXV2005]. Multi-objective techniques in his algorithm are motivated from Deb *et al*.'s NSGA-II [14]. In his approach, classification error and size of structure are considered to be the two objectives to be minimized. A similar idea of a multi-objective GP-based classifier has been proposed in [56]. In addition, multi-objective GPs have been used in financial predictive models [37] [57].

The use of an archive (or a repository) is becoming a trend in multi-objective GAs/GPs. The main motivation for this mechanism is the fact that a currently non-dominated solution may not necessarily be non-dominated among all the historical records. Figure 2.6 shows the flow charts of using an archive and without using an archive.

Figure 2.6 Flow charts of two methods in multi-objective GA/GP to implement elitism



2.5.2 Survey of Multi-objective GAs/GPs

2.5.2.1 NSGA-II

NSGA-II [14], proposed by Deb *et al*., is an improved version of NSGA [20]. The algorithm maintains both a population and an archive with the size of *N* respectively. Elite preservation is applied in each generation after the population is merged with the archive. The *N* best-ranked solutions are preserved in the archive. NSGA-II also proposes fast non-dominated sorting with complexity of O(MN^2) (where *M* is the number of objectives) to assign ranks to the solution in the merged population. When the number of non-dominated solutions exceeds the size of the archive, a crowded-comparison method for diversity is invoked to further discriminate among non-dominated solutions. Both fast non-dominated sorting and crowded comparison methods have been introduced in the first chapter. The total complexity of the algorithm is O(MN^2) as opposed to O(MN^3) in NSGA. The techniques used in NSGA-II have also been adopted in several multi-objective GP algorithms.

2.5.2.2 Rodriguez-Vazquez's MOGP

Multi-objective genetic programming (MOGP) by Rodriguez-Vazquez *et al.* is designed for a class of problems in engineering -- system identification [55]. In such a problem, a system model is required to be built satisfying a number of objectives, from input-output observation from the system.

In Rodriguez-Vazquez's MOGP, a non-dominated counting technique is used to obtain non-dominated solutions. The diversity issue is addressed by a fitness sharing method which encourages the reproduction of solutions located in sparser regions. The 'preference information' is introduced in the form of a goal vector, which specifies the favored region of search space over the optimization process. Accordingly, a 'preferability operator' takes responsibility of implementing objective preference based on preference information as well as keeping solutions within boundaries. These multi-objective optimization techniques are borrowed from MOGA [17]. With regard to GP, the hierarchical tree representation with both crossover and mutation operators is adopted. MOGP also takes GP's well-known bloat problem into consideration by considering the complexity of the model as one objective to be minimized.

2.5.3 Survey of Multi-objective PSO

Particle swarm optimization seems to be more suitable for real world MOO because of its high speed of convergence shown in single objective optimization [52]. In recent years, applying PSO to MOO has become increasingly popular. Moore and Chapman attempted to handle MOO by applying Pareto dominance into their approach, although it has been criticized for not adopting any scheme to maintain diversity [58]. The algorithm of Ray and Liew uses Pareto dominance for convergence and crowding comparison to maintain diversity as well as a multilevel sieve to handle constraints [59]. The algorithm proposed by Parsopoulos and Vrahaits focuses on addressing the difficulty of generating the concave portion of the Pareto front by using an aggregation function [60]. Hu and Eberhart [61] propose a dynamic neighborhood PSO which uses an approach similar to lexicographic ordering [62]. Fieldsend and Singh's approach adopts an unconstrained elite archive to store the non-dominated solutions during the optimization process [63]. A mutation operator is also applied on the velocities to avert premature convergence. Li [64] proposes an approach which applies the main techniques of NSGA II [14] to PSO algorithm. In Coello *et al.*'s version of multi-objective PSO (MOPSO) [21], he compares his results to three highly competitive SMO algorithms: NSAG II [14], PAES [15] and microGA[65]. Agrawal *et al*. [66] proposes an interactive particle swarm optimization algorithm (IPSO), which is similar to Coello's MOPOS. An incorporation of a decision maker gives this approach novelty and efficiency. Several attempts of adaptively optimizing the PSO parameters during the optimization process have also been made by researchers in the field [67] [68].

2.5.4 Coello's Multi-objective PSO

Coello's MOPSO [21] incorporates Pareto optimality with PSO to handle multi-objective optimization. It uses the adaptive hypergrid method (one of the histogram methods) to maintain diversity and an archive (or repository) for historical records of non-dominated solutions and memory of the individual best of a particle.

Besides particle population and velocities, the repository is also updated each iteration. The mechanism of the repository consists of two main parts: an archive controller for convergence and an adaptive hypergrid for diversity. All currently non-dominated solutions are to be inserted into the repository and dominated solutions are to be eliminated from the repository. Once the repository is full, a secondary criterion is applied to maintain diversity: particles located in less populated areas of objective space are reserved.

Coello argues that a mutation operator is able to improve performance because the nature of the high convergence speed of PSO may cause the Pareto front to fall on a local optimum. In his MOPSO an adaptive mutation operator is applied to particles as governed by a probability each iteration. However this probability is decreased in the process of optimization.

A simple scheme to handle constraints is also adopted. Whenever two individuals are compared, it follows these rules: if both are feasible in constraints, apply non-dominance directly; if one is feasible and the other is not, the feasible dominates; if both are infeasible, then the one with lowest amount of constraint violation wins.

CHAPTER 3 - Genetic Regulatory Network Modeling

3.1 Introduction

With the growing demand for food in the world, crop modeling has been a popular research area for many years [69]. Before the exploding development of genomic science, crop simulation modeling was a combination of physiology and empiricism [70][75]. The most recent trend is to predict plant phenotype of a plant by unraveling the network of interacting genes that actually control plant process at the expression level. Such a network in biology is called a *genetic regulatory network* (GRN).

Another reason for the popularity of gene modeling is the rapid advancement in the field of genome sequencing. As of August 2008, 843 organism genomes were completed with another 2951 in progress[1]. The genome of an organism consists of biologically coded information that plays an important role in control of cellular processes, its response to environmental stimuli and its development. Based on such circumstances, there is a major and growing gap between available genomic data and a functional knowledge of the networks whose operations the DNA encodes. This trend necessitates a great deal of work on the automated recovery of a gene network from the observed data, *e.g.* gene expression. This type of problem is called *genotype to phenotype mapping* in biology, and is considered to be a major issue facing applied biology today [71]. In the meantime, physiological methods have been used in crop models to predict phenotypes as responses to variable environmental inputs which may include time-varying temperature, solar radiation and soil water balance [70][74].

With the urgent demand of gene regulatory modeling based on input environmental and output phenotype data, there are a variety of challenges laid out in front of us [72]. Firstly, the number of variables to be considered in a gene network model, in many cases, is very high. Secondly, the number of gene expression profiles available may be much less than the number of variables. Thirdly, there is no standard model of the regulatory mechanisms for the genes, except for a generic cause-effect.

[1] http://www.genomesonline.org/gold.cgi

In addition, noise is another factor hindering model accuracy. Noise is inevitable in both environmental and phenotypic data. Slight noise could be magnified by intricate interactions within genes and make it difficult to recover a simple mathematical model.

In the rest of this chapter, Section 3.2 introduces basic concepts and terms of genetics. Section 3.3 and 3.4 present a survey of the methodology and models that are most commonly seen. The stochastic approach and its effort to address those challenges in gene regulatory modeling are explained in Section 3.5. The final section, 3.6, describes a flowering control model of a type of intensively studied plant *Arabidopsis thaliana*, which is used in the problem formulation of this dissertation in Chapter 4.

3.2 Basic Genetics

In biology, *genomes* contain an organism's entire hereditary information, which is encoded as *deoxyribonucleic acid* (*DNA*). The genome is made up of one or more extremely long molecules of DNA that are organized into *chromosomes*. DNA is a linear, double helical structure that looks like a molecular spiral staircase. The double helix is composed of two intertwined chains made up of building blocks called *nucleotides*. *Genes* are the region of chromosomal DNA that carry information specifying the chemical composition of *proteins*, which largely determine the structure and physiology of organisms. In the from-gene-to-protein process, genes specify the information on the timing as well as the amount of proteins to be synthesized. The primary structure of a protein is a linear chain of *amino acids*.

There are several steps leading from an active (expressed) gene to a protein. Two of them are *transcription* and *translation* [73]. Transcription is a process that copies the nucleotide sequence in one strand of gene into a complementary single-stranded molecule called *messenger ribonucleic acid* (*mRNA*). Subsequently translation produces a chain of amino acids based on the sequence of nucleotides in the mRNA. Those chains of amino acids will ultimately form proteins.

Genes are inactive when the DNA wraps around complexes of modified histone molecules (*nucleosomes*) making it inaccessible to transcription mechanisms. In order to become active, certain molecules must attach to the *promoter region*, an area of DNA upstream of the segment coding for the protein. These attaching molecules, often protein, are known as *transcription factors*.

Another important concept in genetics is genetic variation, which specifies the phenomenon that any particular gene may exist in different forms in different individuals. The different forms of the same gene are called *alleles*. This allelic variation is the basis for hereditary variation. However, because there are only one or two chromosome sets per cell in most organisms, there are only one or two alleles per gene. The classification of individuals by allelic combination is called *genotype*. In contrast, the characterization of organisms by their appearance is called *phenotype*. Even for the same genotype, environmental variation can cause distinctive phenotypes.

3.3 Gene Regulatory Network Modeling

Gene regulatory network modeling focuses on discovering the interaction between genes from genetic, environmental, and phenotypic observations. This kind of problem is considered as *reverse engineering* from a system engineer's standpoint, which tries to use the behavior of the system itself to directly infer the interactions of the system.

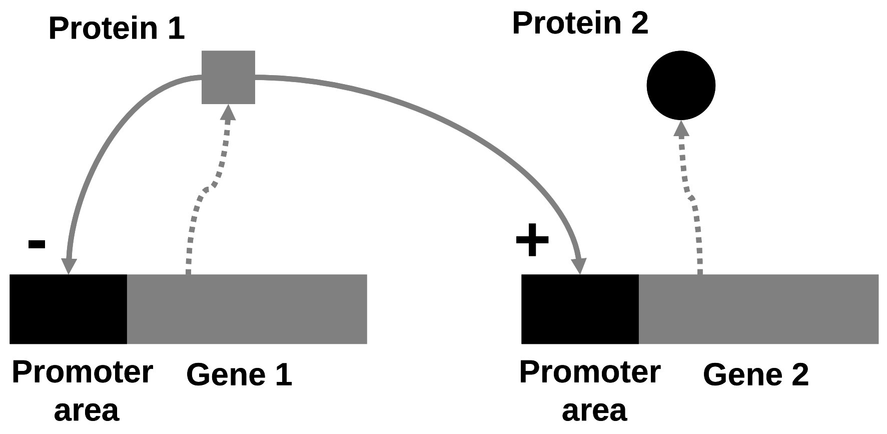
However, there is no single standard modeling approach to discover the structure and functionality of gene regulation from a large scale phenotypic and gene expression data. The choice of model is more of problem dependent. Moreover, a models often have parameters. Thus different techniques are required to estimate these parameters. In this section, most common gene regulatory network (GRN) modeling approaches as well as optimization techniques are introduced.

3.3.1 Graphical Models

In genetics, the transcription process begins with the transcription factor (attaching molecules) attaching to the promoter. The involvement of attaching molecules provides a means for gene regulation. If any needed molecules are unavailable, transcription can not begin and the gene is inactive. Transcription factors and mRNA degrade with time so their continuous production is required to sustain their action. In many cases, the transcription factors themselves are gene products that may be under gene regulation by others. In the opposite manner, some molecules (called repressor molecules) may occupy the attachment to the promoter and block transcription. Based on above understanding, it is straightforward and natural to use a graphical model to simulate the transcription process [76] [78] [79]. Graphical models generally consist of environmental inputs, phenotypic output(s), graphical nodes representing genes and arrows indicating the interaction between them (either promotion or repression factor). A simple example of genetic graphical model is in Figure 3.1. In this case, protein 1 has a repressing effect to its own gene (marked in '-'), but has a positive promotion on the transcription of gene 2 (marked in '+').

One criticism on graphical models is the lack of quantitative estimation on the outcomes based on environmental inputs and gene interactions. The graphical model is straightforward and easy to understand but qualitative and incapable of making arithmetic predictions.

Figure 3.1 An example of genetic graphical model



3.3.2 Boolean Network Models

In Boolean networks, models are presented as a directed graph where each node in the graph represents a gene. Different from graphical models, each node gives an output value of either "0" or "1", corresponding to the active ("on") or inactive ("off") status of a gene. Other nodes receive binary output values as their inputs. Inputs go through the node's internal Boolean function and calculate the current state of the node as output.

Conventional Boolean network models make use of a synchronous update scheme where each node in the model is updated at the same time controlled by a central clock. A sequence of states generated by Boolean functions is finite and may be repeated after certain number of updates. This is usually referred to as the state cycle or attractor. The structure of the Boolean network model can be validated or modified by comparing simulation results with time series observations.

Boolean network models have been used in genetic regulatory network modeling [80] [81] [82] and proved their analytical tractability and accuracy, although there are reports about the difficulties to simulate temporal dynamics of the real system [83].

3.3.3 Differential Equation Models

Differential equation models are used to simulate the cellular production rates of important proteins. These protein rates are often related to the concentrations of mRNA and levels of gene activation (transcription). In Baldi and Hatfield's model [85], for example, the gene expression level is presented dynamically associated with mRNA and protein levels. It can be written in the following equation:

$$\frac{dp}{dt} = R\,\boldsymbol{g} - \lambda p \,, \qquad \text{Eq. 3.1}$$

where p is the biochemical level; R and λp are the production and degradation rates of p per unit time and $\boldsymbol{g}$ is assumed to be a factor relating other gene products to production of p. To simulate genes' on/off behavior, $\boldsymbol{g}$ is usually considered to be a *transfer function* instead of a parameter constant. We obtain the updated model after substituting $\boldsymbol{g}$ with a linear form $\beta_0 p_0 + \beta_1 p_1 + ... + \beta_N p_N$, where p_i represents the levels of gene products affected by gene i and β_i is the effect strength of gene i (a negative number for promotion and positive number for repression). Under above circumstances, the new model can be written as:

$$\frac{dp}{dt} = R\,\boldsymbol{g}\left(\sum_{i=0} \beta_i p_i\right) - \lambda p \,. \qquad \text{Eq. 3.2}$$

In general, differential equation models are suitable for modeling complex dynamic system such as oscillations, cyclic patterns and switch-like behaviors [84]. However, a second step of estimating the parameters that associate with differential equation models is inevitable, which may increase the complexity of modeling process.

3.3.4 Linear Models

A gene regulatory network can be represented as a discrete time equation $\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t))$, where $\boldsymbol{x}(t) = (x_1(t),..., x_N(t))$ is a vector of element $x_i (1 \le i \le N)$

representing gene expression levels at time t and $\boldsymbol{f} = (f_1, ..., f_N)$ is a vector valued function from N dimensional space $\Re^N$ to $\Re^N$. When function $\boldsymbol{f}$ is linear, the equation becomes a linear model. In general, a biological system is nonlinear, but nonlinear models may cause more difficulties in estimating parameters from limited number of data samples. In addition, by using linear models for GRN, regulatory genes' interaction functions can be expressed as a *regulation matrix*. Then linear algebra methods such as linear regression, principal component analysis, singular value decomposition (SVD), Gaussian methods, *etc*., can be applied to solve linear models and estimate the strength of interactions. Examples of linear models can be found in [90] [91] [92].

3.3.5 Stochastic Models

The latest results in genetics demonstrate gene expression as a stochastic process [86] [87]. Many stochastic models are created on the basis of this new discovery, such as [88] [89]. One typical example of this class of model is the Langevin Equation [89], which is obtained by adding one more term to a differential equation as noise:

$$\frac{dx}{dt} = f_i(x_i) + v_i(t), \qquad \text{Eq. 3.3}$$

where $v_i(t)$ is the additive noise term.

In addition, since the stochastic models simulate a stochastic process, it uses *Monte-Carlo algorithms*, which is a class of computational algorithms that relies on repeated random sampling to approximate real results, to obtain solutions of the equation.

3.3.6 Neural Network Model

Neural networks were initially devised to model brain function to imitate cognitive feats such as the learning process and pattern recognition. A neural network model is composed of interlinked nodes (*neurons*), each of which always has a number of inputs and one output. Each interlink of a node is also associated with a weight value. Each node also contains one *transfer function* (often a sigmoidal function) that incorporates non-linearity into the model. To calculate the output of a neuron, nodal inputs' combinations with their associated weights are passed to the transfer function to obtain the output. In a word, the function of a neural network completely depends on four elements: *i*) the structure of its nodes and interlinks between them, *ii*) the method of combining nodal inputs for substituting into the transfer function, *iii*) the transfer function itself and, *iv*) weights value associated with each link. The first three elements are designed in advance while the last element weights are optimized to fit output data. This process is termed the *training process*. There have been a number of optimization techniques applied in neural network training, such as back propagation approach, genetic algorithms and particle swarm optimization.

In Welch *et al*.'s genetic neural network model [70], neural network is used to simulate ON and OFF behavior in gene regulation. Each node existing in the neural network model represents a gene. When this gene is turned ON, the weight applied on this interlink corresponds to the effect of regulation by this gene. However, mutation can result in deactivation of a gene so it does not function at all.

One of difficulties in neural network modeling is that modelers do not have preliminary knowledge on how large the network should be designed. Larger networks with more weights are presumed to retain knowledge of a more complicated nature and also cause the increasing complexity in modeling. A common method in neural network modeling is starting with smaller network and adding more nodes later when training efforts are not successful.

Some common approaches in GRN are reviewed in this section; however other methods such as Bayesian networks [93] are not discussed. The novel and emerging stochastic approach for GRN is revealed in the next section.

3.4 Stochastic approaches in Gene Regulatory Network (GRN)

Stochastic approaches such as genetic algorithms (GAs) and genetic programming (GPs) have been used in clustering of gene expression data [94] [95], inference of GRN structure [96] [97] and estimation of model associated parameters [70] [77].

In this literature, GAs are mostly used in model parameter estimation. Many GRN models introduced before are parametric models, *e.g.* differential equation and neural network models. Model parameter estimation can be considered as a non-linear optimization problem, where the objective function is the goodness-of-fit criterion. Commonly used goodness-of-fit criteria are least mean square (LMS).The optimizer may search the objective function space and converge to satisfied solution(s). One difficulty in such a problem is the estimation of solution may have premature convergence and land on local optima rather than the desired global optimum. This challenge arises from the complexity of the search landscape that commonly emerges in global optimization problems. In GA-based parameter estimation methods, a solution (chromosome) is actually the string of parameters to be optimized associated with GRN structure represented as a mathematical model. Selection, crossover and mutation operators are applied on the population of solutions until they converge to satisfactory results.

GP is an extension of GA, where candidate solutions are represented as certain structures rather than a string of numbers. This feature makes GP more suitable for estimating the structure or topology rather than the parameters of a network. For instance, Ando *et al*. [96] used GP to generate differential equation models, which represent genetic networks. Each GP solution is designed as a tree structure of mathematical operations (functions) and variables (terminals) and those variables represent each gene's mRNA concentration level. An example of how a differential equation model is encoded into a GP solution is shown in Figure 3.2. The two tree structures in the figure correspond to differential models as follows:

$$\frac{dx_1}{dt} = ax_1x_2^2 + b \qquad \text{Eq. 3.4}$$

$$\frac{dx_2}{dt} = cx_1x_2 + dx_2, \qquad \text{Eq. 3.5}$$

where a, b, c, d, are model parameters.

The fitness of each solution is defined as the sum of the squared error and the penalty for the degree of the equations:

$$fitness = \sum_{i=1}^{n}\sum_{k=0}^{T-1}(x_i^{'}(t_0 + k\Delta t) - x_i(t_0 + k\Delta t))^2 + a*m, \qquad \text{Eq. 3.6}$$

t_0 : the starting time

Δt : the step size

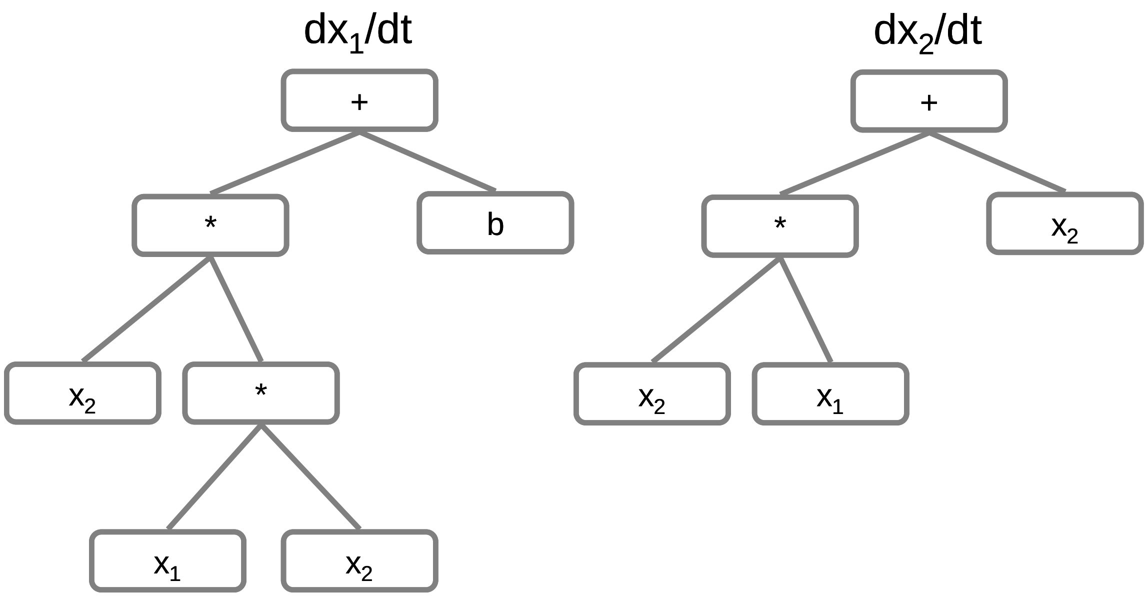
n : the number of the observable components

T : the number of the data points

where $x_i(t_0 + k\Delta t)$ is the given target time series (*k* = 0, 1, …, *T*-1); $x_i^{'}(t_0 + k\Delta t)$ is the time series acquired by calculating the a GP solution. *m* is the number of terms and *a* is the weight constant. This penalty term is generated to overcome the bloating problem in GP based on minimum description length (MDL) criterion, which has been often used in GP. In other words, the definition of fitness in [96] indicating a solution with a smaller number of terms and closer to the target time series has ae higher possibility to be selected in stochastic optimization process. Similar work using GP for GRN structure inference can be found in [97].

Besides GA and GP, particle swarm optimization is another popular and fast growing bio-inspired optimization algorithm. It has advantage of fast convergence and thus has been applied to various optimization problems in GRN [98] [99].

Figure 3.2 An example of a GP solution in [96]



3.5 Multi-objective Approaches in Gene Regulatory Network Modeling

To date, most algorithms developed to infer GRN are single-objective. However, previous work on single objective GRN showed that a network found by single objective algorithms can generate similar results to experimental data but they may not have structural or numerical resemblance to the real network [100] [101] [102]. This may occur because the optimization process is caught in local optima. Stochastic multi-objective approaches preserve the diversity of solutions in a population and present them as a Pareto front. Thus they are able to find multiple optima hopefully including the global optimum [103].

The multi-objective optimization approach is likely to be more suitable for genetic regulatory modeling and its associated parameter estimation based on following three reasons

[77]: *i*) Multiple data types (continuous, discrete, and/or categorical) are very problematic for the design of a single objective function; *ii*) Individual data sets usually are from different sources and may be inconsistent; *iii*) Tradeoffs between solutions may reveal the magnitude of discrepancies.

Based on above reasons, research on multi-objective algorithms in gene regulatory network modeling is relatively new but growing. Several attempts at applying multi-objective approaches to GRN have been made [103] [77].

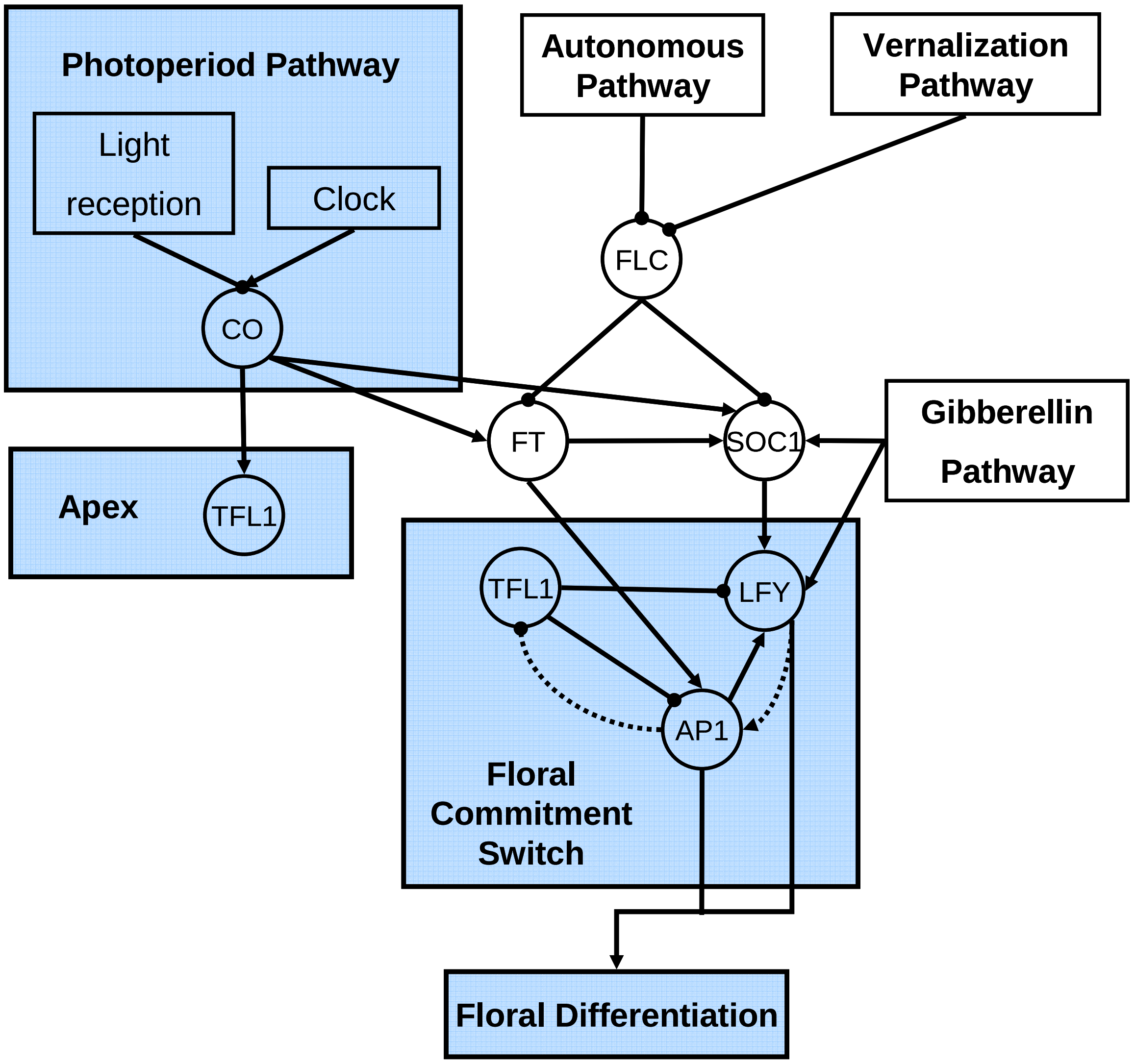
3.6 Flowering Control in *Arabidopsis thaliana*

Studies in flowering control are very critical in crop modeling to establish the growth and yield generating process within temporal limits. There has been extensive research on flowering control in Arabidopsis due to its small genome, short generation time, self-compatibility, amenability to stable transformation and the availability of numerous mutants [70]. Arabidopsis is a long-day plant. This means the stimulus of long days promote flowering in response. Under short days, flowering will be much later. Flowering in Arabidopsis consists of two stages; the first stage is to form an inflorescence (or bolting) and the second is to produce flowers. These two stages can be distinguished genetically [104]. The bolting stage is hugely influenced by environmental signals, such as day length and temperature; while the latter stage is less affected by additional environmental inputs. So in most related works on flowering control modeling of Arabidopsis, inflorescence (or bolting time) is considered as the criterion of flowering.

Flowering control in Arabidopsis at the genetic level has been gradually discovered and revealed with the advancement of genetic biology. The gene regulation of flowering control system, as shown in Figure 3.3, is well understood now. Input information includes the photoperiodic promotion pathway that senses day lengths; verbalization pathway that responds to an extended period of cold; the gibberellins pathway that responds growth hormone levels and an autonomous pathway. In the figure, *Flowering Locus C* (*FLC*) is a major repressive integrator gene which is downregulated by both the autonomous and vernalization pathways. The photoperiod pathway gene *Constans* (*CO*) functions to combine diurnal clock phase information with photoreceptor input to measure day length. Expression levels of key genes, including *FLC*, *SOC1* and *Flowering Locus T* (*FT*), are altered through the input information from all the pathways and are fed into a three-gene-switch including the inflorescence identity gene *Terminal*

Flower 1 (*TFL1*) and the floral meristem identity genes *LFY* and *Apetala1* (*AP1*). When this switch turns on, the plant is committed to flowering. The expression level of this three-gene-switch then feeds into floral differentiation and determine the growth of reproductive plant parts (flower, *etc*.).

Figure 3.3 Flowering Time Control in Arabidopsis



CHAPTER 4 - Problem Formulation and Multi-objective GP-PSO Hybrid Algorithm

4.1 Introduction

There are a small number of model organisms whose genetic networks have been studied in detail including Arabidopsis, bakers yeast, nematode, the sea urchin, and the fruit fly, Drosophila, among others. This research is directed at genetic models of Arabidopsis.

The general approach to gene network modeling involves developing mathematical models such as Boolean networks, Bayesian models, and linear differential equations and then utilizing available experimental data to estimate the parameters associated with these models. The goal in this research, however, is to infer a gene regulatory network structure and its parameters directly from large amounts of both gene expression data and phenotype data simultaneously. Identification of genomic regions that contain key genes, plus knowledge of their interactions may be sufficient for some applications [105] [106].

4.2 Data

Environmental data were collated as part of the activities of an international consortium investigating the evolutionary aspects of gene network pathway signal integration[1]. This project provided the context for a synthetic data set constructed for structure discovery. Eighteen sites were selected ranging from Coimbra, Portugal (40°13’N, 8°25’W) to Jokioinen, Finland (60°49’N, 23°30’W). For each day of the year from March 1 to June 30, daily average temperatures, (Tmax+Tmin)/2, were averaged for 25-30 years (most often 1971-1998), depending on the site. Daily photoperiods were obtained for these sites and dates from the United States Naval Observatory[2]. Due to plants’ sensitivity to light, we followed a common plant modeling practice of using Civil Twilight, which begins/ends with the sun six degrees below the horizon.

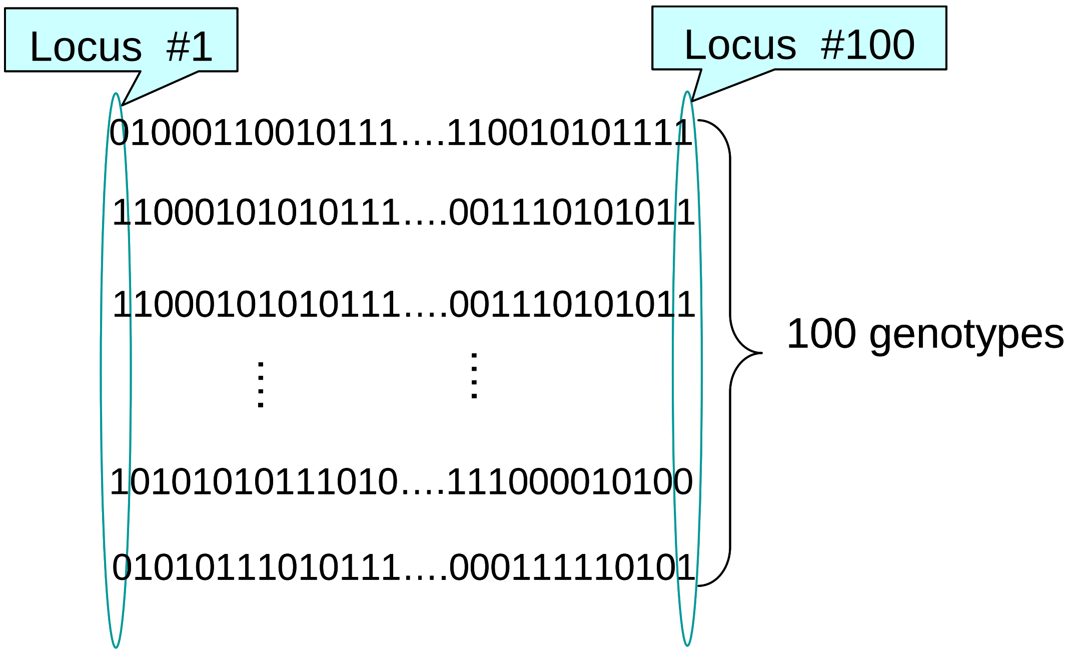
A synthesized and parameterized network, which mimics key features of the well known *Arabidopsis thaliana* flowering time control genetic network, was generated [107]. Functional

[1] http://www.egad.ksu.edu

[2] http://aa.usno.navy.mil/data/docs/RS_OneYear.html

characteristics of individual genes will be described in the next section; broader discussions of gene computational abilities are beyond the discussion of this dissertation, but can be read in [85] [108] [110]. Each gene had a single parameter that was assigned one of two different values, representing different mutant alleles. One hundred distinct genotypes were constructed representing different allelic combinations, as shown in Figure 4.1. Each genotype was described by 100 markers, equated for prototyping purposes to genes, among which the network genes were hidden. Each gene in the genome had two alleles, encoded as ‘0’ and ‘1’ accordingly, but only network genes influenced the phenotype. Each genotype was simulated at each site for each of three assumed planting dates spaced ca. one month apart. The synthetic data resulting from these simulations included: (*i*) the day of the year that the first inflorescence bud would become visible (bolting date, a commonly used proxy for floral initiation), and (*ii*) the gene expression time series for one gene in the actual network.

Figure 4.1 Illustration of synthetic genomes



4.3 Synthetic Network

The goal is to obtain a simplified genetic network that can simultaneously predict both the bolting dates and expression data as close to the synthetic data as possible. The performance measure was the Root Mean Squared (RMS) error *E* of the predictions of the generated models as compared to the synthetic data. That is,

$$E = \sqrt{\frac{\sum_{i}^{n}\left(D_i^{\text{data}} - D_i^{\text{mod}el}\right)^2}{n}} \qquad \text{Eq. 4.1}$$

where D_i^{data} and D_i^{model} are, respectively, the synthetic bolting dates (or gene expression) and those predicted by a particular model structure for the *i*th combination of genotype, geographic site, and planting date. The optimization routine should simultaneously minimize the RMS errors in prediction of both bolting dates and gene expression data, hence necessitating the use of multi-objective optimization algorithms.

The genes in the model genetic network are allowed to implement any of the following four functions: (*i*) gain: $o = c_g \cdot i_1$ (*ii*) summer: $o = c_s \cdot i_1 + i_2$ (*iii*) multiplier: $o = c_m \cdot i_1 \cdot i_2$ (*iv*) integrator: $o(t) = o(t-1) + c_i \cdot i_1(t)$. In each of the cases, i_1 and i_2 are the inputs and *o* the output. Each gene has a single parameter associated with it (*c* = c_g, c_s, c_m or c_i). As we allowed only two alleles per gene, each parameter is assigned two separate numerical values, one for each allele. Additionally, there are two inputs to each gene which can be either the outputs from other genes in the network, or an environmental input – either the photoperiod (*P*) or the temperature (*T*). These operators were chosen because (*i*) genes are, in fact, able to biochemically approximate them [108]; (*ii*) the first three ground quantitative genetic equations, currently the dominant formalism applied to the genotype to phenotype mapping problem and to the initial steps in gene discovery [111]; and (*iii*) all four are used to synthesize simple physiological process models that approximate plant behavior at a higher level of biological organization [108].

4.4 Multi-objective GP-PSO Hybrid Algorithm

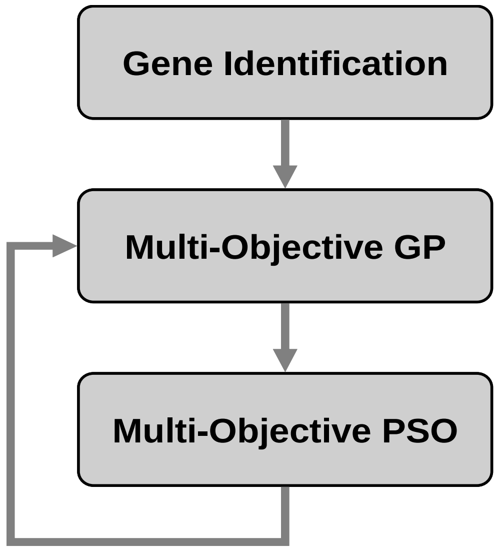
4.4.1 Overall Hybrid Algorithm

A multi-objective GP-PSO hybrid algorithm is proposed to address the defined problem. This approach can be roughly divided into three stages, as shown in Figure 4.2.

The first stage is for data pre-processing. It is well known that a biological system may contain a large number of genes and computational time grows exponentially along with the number of genes in the network model due to the *curse of dimensionality*. Therefore it is imperative to reduce total gene numbers in order to avoid huge computational overhead in modeling. The first stage, gene identification (GI), is applied to accomplish the above goal and identify a set of genes that are most likely to influence the flowering response.

After this, the multi-objective GP initializes a random population of N solutions, each of which is a network structure comprised of M identified genes. All the structures are evaluated and stored in a GP archive[1] as parent solutions. The use of this archive is for elite preservation which can be seen in a variety of stochastic optimization approaches. A mutation operator is used to generate new gene network structures.

Figure 4.2 A three-stage flow chart of overall hybrid algorithm



[1] GP archive is named to distinguish PSO archive for parameter estimation.

For each new network structure, multi-objective particle swarm optimization is applied to estimate parameters. Least RMS errors are adopted as the criterion in the predictions of both bolting date and gene expression. After that, the PSO non-dominated front, comprised of equally good estimated parameter vectors under the structure, is obtained and stored in the PSO archive. The solutions in the PSO archive, each of which consists of a network structure and its associated best estimated parameter vectors, and their corresponding RMS errors (fitnesses), are returned to multi-objective GP.

Multi-objective GP use returned solutions in conjunction with parent solutions to recalculate its new non-dominated front and then stored it in the GP archive. The solutions retained in the GP archive become the parent solutions for the next generation. This process is repeated for the maximum allowed number of generations. One critical component of algorithm design is that the method to form the non-dominated front in either multi-objective GP or PSO satisfies the definition of good dominance and diversity.

4.4.2 Gene Identification

Each genotype consists of 100 genes but only a subset of unknown cardinality is actually present in the flowering time control regulatory network. The basic goal of the gene identification step is to exclude genes from network membership if their alleles do not alter bolting time.

The detailed procedure is as follows. For each of the 100 loci, genotypes are divided into two groups based on their alleles. F-tests are applied to the corresponding bolting dates in these two groups. These tests reveal if the overall bolting date sets associated with different alleles are different with high statistical confidence. This is a simplified form of *quantitative trait locus mapping* (QTL) [111] [109], a standard mathematical method used as part of gene discovery. Average p-values of F-tests are shown in Appendix B. A smaller p-value indicates its corresponding gene is more likely to appear in the network.

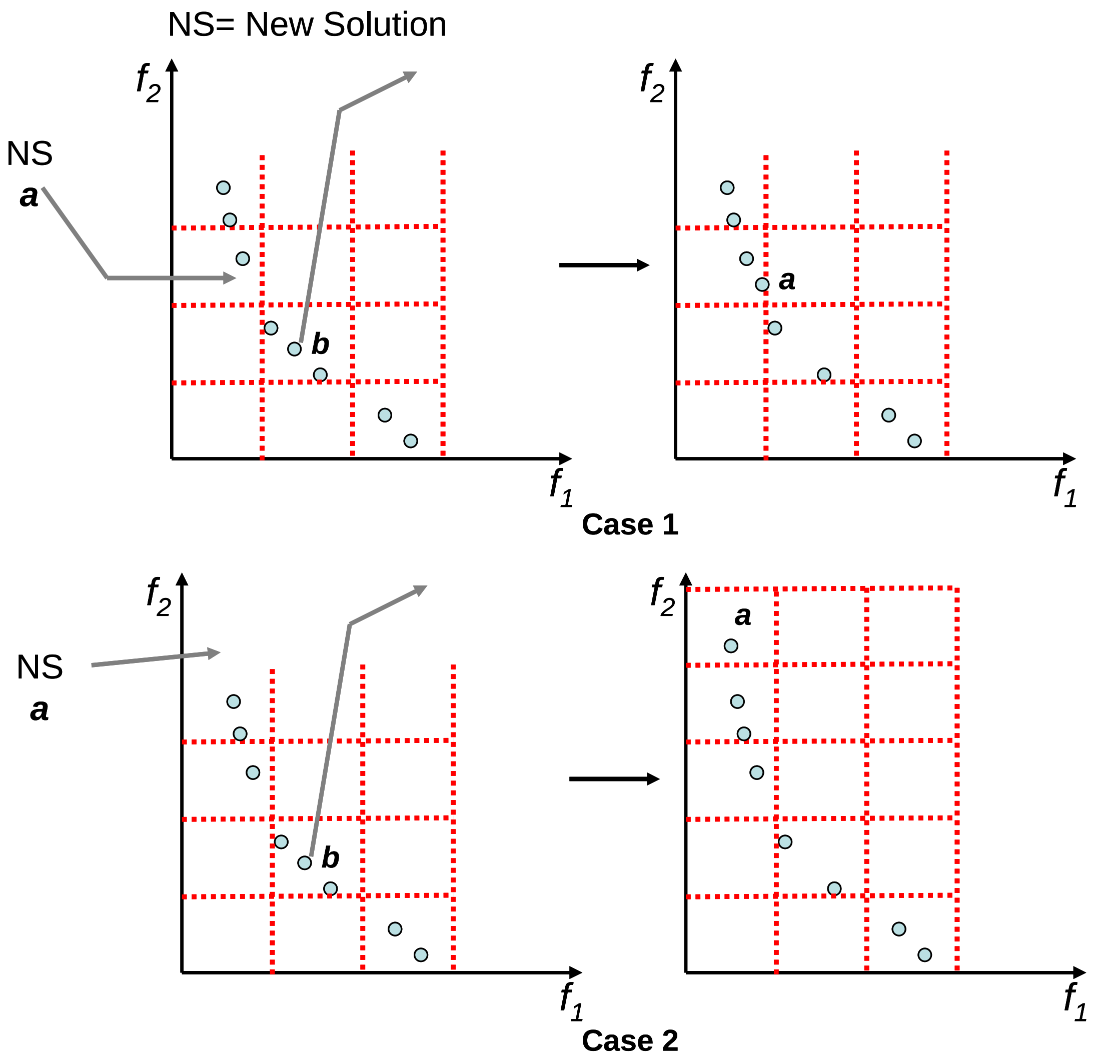
4.4.3 Multi-objective Optimization Issue and Archive Control

As explained in prior section, convergence and diversity are the two criteria [13] [7] for a multi-objective optimization algorithm: solutions should (*i*) rapidly converge to the Pareto front and (*ii*) be spread out on the front with proper intervals. To aid in rapid convergence, our algorithm implements non-dominated sorting and a histogram method is utilized for maintaining diversity in the resulting non-dominated front. Each of these two components is detailed as follows.

Non-dominated sorting approach [14]: In the first generation, a population of N solutions is ranked into different non-dominated fronts, each of which consists of solutions that do not dominate each other. Each solution can be compared with every other solution in the population to find if it is dominated. This requires $O(MN)$ comparisons for each solution, where M is the number of objectives. To find the non-dominated front with the highest rank (the first non-dominated level), the total complexity is $O(MN^2)$. The solutions of the first non-dominated front will then be saved as elites in an archive. In the following generations, each of offspring solutions is compared with every member in the archive, solutions that have been dominated in the archive are discarded and the ones that dominated them are inserted into the archive. Suppose the size of the archive is also N, archiving process also requires $O(MN^2)$. So the complexity of the non-dominated sorting algorithm in each generation is $O(MN^2)$.

Adaptive Histogram method (*Hypergrid*) [15] [21]: When the number of non-dominated solutions exceeds the archive size, the histogram method is activated for truncation and keeps the size of the front within that of the archive. The motivation behind this method is to produce a well-distributed non-dominated front. In the histogram method, the objective function space is divided into identically sized grid cells and more densely populated compartments are thinned. When the new solution is inserted into the archive, there may be two scenarios: *i*) If the new solution lies within the grid bounds, algorithm goes to archive flow control process directly (shown in Figure 4.3, case 1); *ii*) If the new solution lies outside the current grid bounds; the grids are restructured to include the new solution before following the archive flow control process (shown in Figure 4.3, case 2).

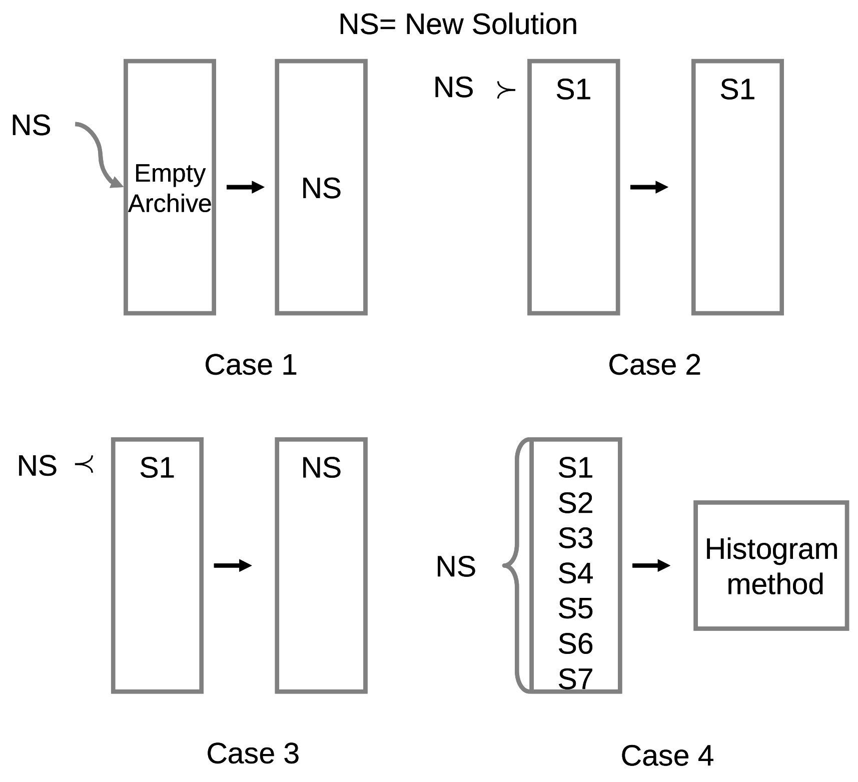
Figure 4.3 Graphical representation of the insertion of a new solution in the adaptive hypergrid when individual lies within/out current boundaries of hypergrid



In the *archive control* process, a fixed size archive is used to store the set of non-dominated solutions obtained at the end of each generation. The decision whether a new solution should be put into archive or not is based on different scenarios as follows: if the archive is empty, then new solution is accepted (case 1, Figure 4.4); if the new solution is dominated by any solution in the archive, it is discarded (case 2, Figure 4.4); otherwise it is accepted into the archive and also any solutions dominated by the newly added solutions are removed from the archive (case 3, Figure 4.4); lastly, if the maximum archive size is exceeded, the histogram method is invoked for truncation (case 4, Figure 4.4).

Both multi-objective GP and multi-objective PSO have their own archive but the same multi-objective optimization techniques and archive control strategy are applied to form a non-dominated front in GP and PSO archives respectively.

Figure 4.4 Archive flow control process



4.4.4 Representation Using Cartesian Genetic Programming

Cartesian genetic programming [32] is used to represent network models. Each solution is represented in the form of a string containing *C* fields, shown in Figure 4.5, where *C* is the number of putative network genes. Each field contains four entries, which designate the two gene inputs (either upstream network genes or environmental inputs), an entry representing gene function (*g*, gain; *s*, summer; *m*, multiplier; or *i*, integrator), and an index identifying the gene in the data that this particular field in the solution represents.

Figure 4.5 Representation of a solution as a string in CGP

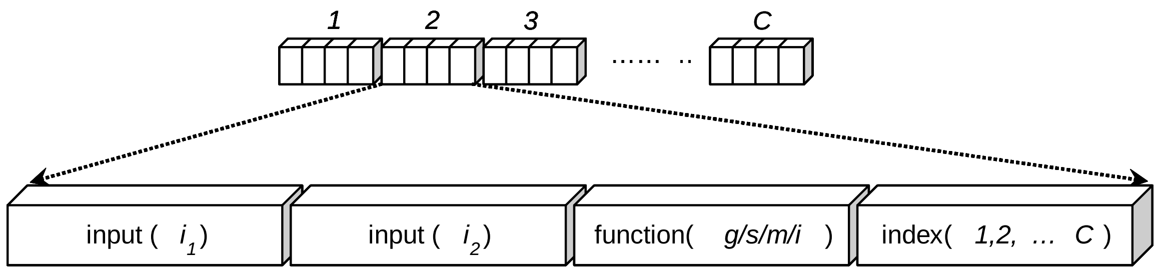
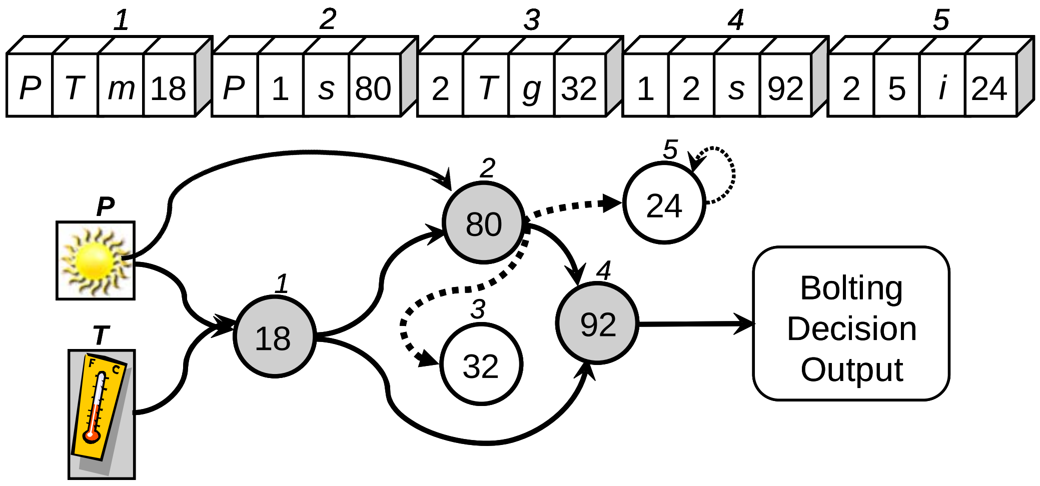


Figure 4.6 shows an example solution containing five genes that were identified during the gene identification process, along with the corresponding gene network. Bolting is predicted to occur at the earliest time when the output of gene 4 reaches or exceeds 1. In all the models there are two environmental input parameters: Photoperiod (*P*) and Temperature (*T*). Since gene 4 obtains its inputs from genes 1 and 2 whose inputs are environmental, the functional part of this network consists of only three genes. The nonfunctional portion is shown with dotted lines; genes 3 and 5 are effectively excluded. It is worthy to note that the number of effective inputs is determined by the function of the gene itself. Take gene 3 for instance, its function is *g*: gain, hence it has only one effective input from upstream genes. Its second input from Temperature (*T*) is not effective and not shown in the figure. In addition, it should be noted that this scheme only encodes network structure. The associated parameters c_g , c_s, c_m and c_i are stored separately and used in the multi-objective PSO section.

Figure 4.6 Representations of a sample solution and its corresponding network topology



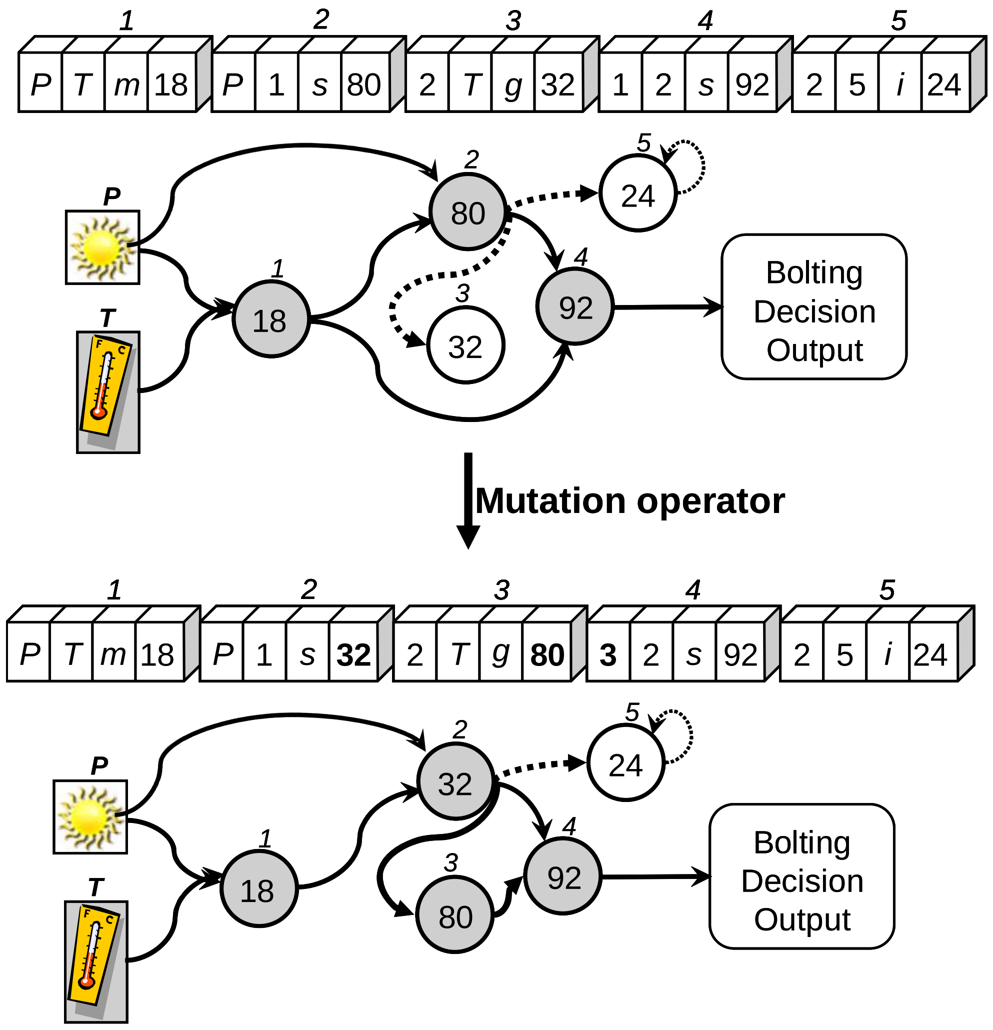
4.4.5 Mutation

The main objective of the mutation operator is to provide sufficient exploration of the search space. Network description elements mutate according to predefined probabilities and within ranges of field-specific feasible values. Additionally, mutations are subject to the following constraints: (*i*) feedback loops are not allowed, and (*ii*) gene indexes are unique. From prior knowledge, it is known that several well worked out developmental genetic networks have largely feed forward topologies. So loops were excluded from candidate structures in order to reduce mathematical computations in this prototype. In addition, gene index entries (field 4) must be unique within any one solution string, since markers (genes) are distinct entities.

Figure 4.7 is an example of applying the mutation operator on encoding strings and the corresponding changes on the network structure. Bold elements in the strings show the entries that were mutated. Index numbers in gene 2 and gene 3 switch and one input entry of gene 4 flips from 1 to 3, which causes a drastic change in network structure: gene 3 is included into the network and becomes functional. This process is called *activation of redundancy* as illustrated in Section 2.3. In the same manner, a previously functional gene may be inactivated and become a dysfunctional gene.

Crossover is another commonly used variation operator in GP. According to [32], empirically, crossover does not show statistically significant improvement on the performance of CGP representation. Nevertheless, the lack of a crossover operator necessitates a high mutation rate. A mutation operator with a 12% mutation rate is used in our hybrid algorithm.

Figure 4.7 Network topology change after applying mutation operator



4.4.6 Multi-objective PSO Based Parameter Estimation

Multi-objective PSO as implemented here has adopted several techniques from Coello's MOPSO [21]. For each regulatory network structure generated by the multi-objective GP algorithm, PSO is used to obtain the parameters associated with each gene. The parameters c_1 through c_M are treated as a vector **c** and the swarm is populated initially by a random vector $\boldsymbol{c}(j)$, $j = 1, \ldots, P$, where P is swarm size. Each vector $\boldsymbol{c}(j)$ corresponds to a position of the j^{th} particle in the swarm. There is also a PSO archive that stores non-dominated parent solutions.

Letting subscripts t and $(t+1)$ denote iteration numbers, positions are incremented from the instantaneous velocity, $\mathbf{v}_t(i)$, as follows,

$$\mathbf{c}_{t+1}(i) = \mathbf{c}_t(i) + \mathbf{v}_t(i). \qquad \text{Eq. 4.2}$$

The velocity is updated using the particle's own recorded previous best position, as well as the current location of the other particles. The update rule is

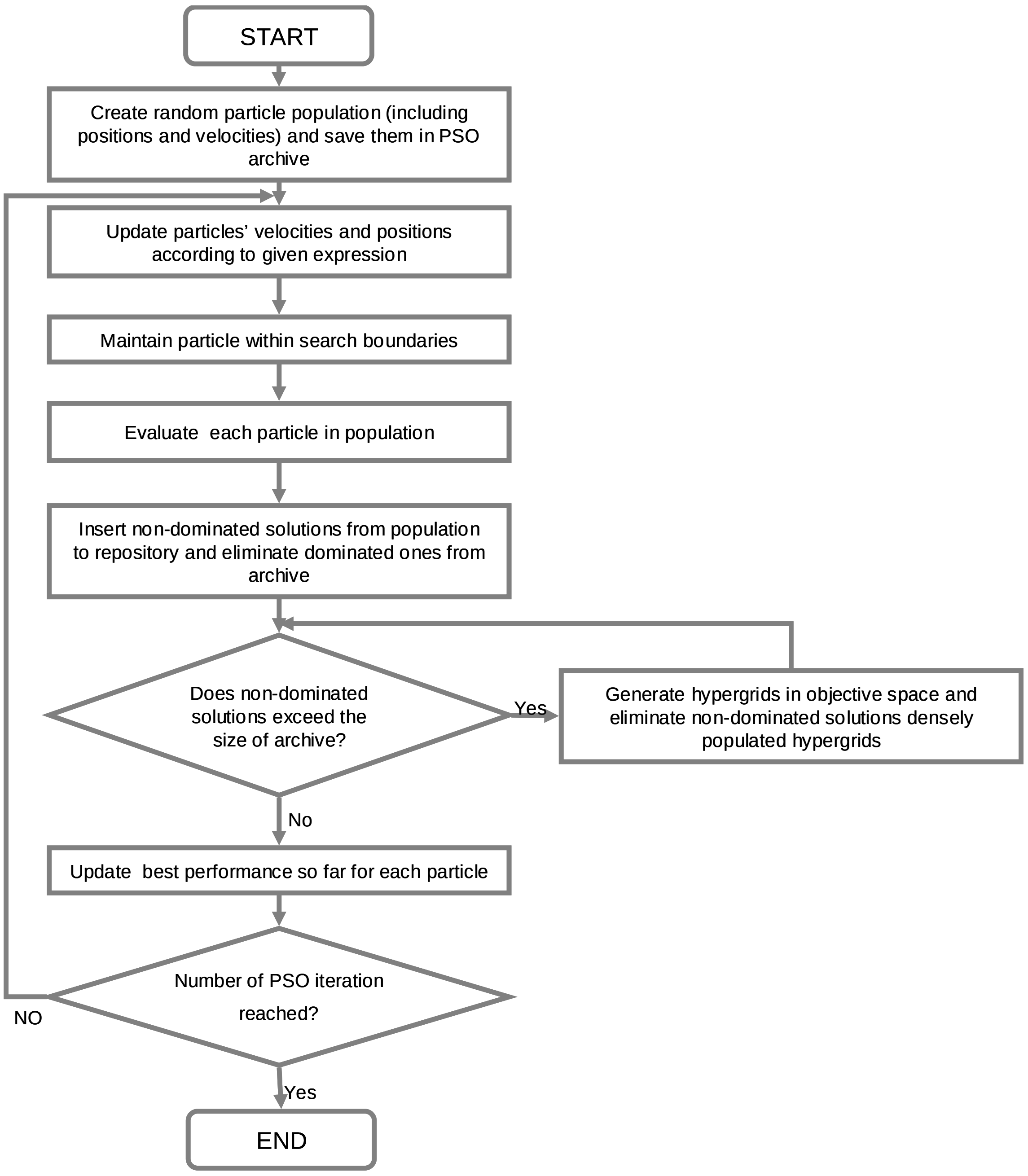
$$\boldsymbol{v}_{t+1}(j) = \chi\times\boldsymbol{v}_t(j) + C_1\times U[0,1]\times(\boldsymbol{c}_{ib}(j) - \boldsymbol{c}_t(j)) + C_2\times U[0,1]\times(\boldsymbol{c}_{gb,t}(h) - \boldsymbol{c}_t(j)). \qquad \text{Eq. 4.3}$$

In the above equation, C_1 and C_2 are, respectively, the *cognitive* and *social constants*, and χ is a *constriction coefficient*, which helps in maintaining stability. $U[0,1]$ is a uniformly distributed random number in the range of [0, 1]. The quantity $\mathbf{c}_{\mathrm{ib}}$ is the individual best recorded position of the ith particle so far. $\boldsymbol{c}_{gb,t}$(h) is a value that is taken from the PSO archive; index h is selected in the following way: the hypergrid which contains least particles is chosen and $\boldsymbol{c}_{gb,t}(h)$is one particle randomly picked in this hypercube. $\boldsymbol{c}_{gb,t}$ is considered as the global best position, in terms of diversity, of any particle in the current iteration t in our approach.

Velocity and position corrections are applied to restrict particles to the predefined search space. When a particle moves beyond the specified region, it is returned to the boundary it has passed beyond. Additionally, its velocity is multiplied by (-1) so that the particle bounces back to search in the opposite direction.

The evaluation of a solution is based on the minimum RMS errors of the objectives evaluated using the equation in Section 4.31. To compute the bolting date goodness-of-fit, the gene with the least RMS error is taken to be the network output gene and is used to score the trial solution. The same method is applied to evaluate gene expression error. The flow chart of our multi-objective PSO is shown in Figure 4.8.

Figure 4.8 Flow chart of multi-objective PSO for network parameters estimation


START
Create random particle population (including positions and velocities) and save them in PSO archive
Update particles' velocities and positions according to given expression
Maintain particle within search boundaries
Evaluate each particle in population
Insert non-dominated solutions from population to repository and eliminate dominated ones from archive
Does non-dominated solutions exceed the size of archive?
Yes
Generate hypergrids in objective space and eliminate non-dominated solutions densely populated hypergrids
No
Update best performance so far for each particle
Number of PSO iteration reached?
NO
Yes
END

CHAPTER 5 - Results and Discussion

5.1 Introduction

This chapter explains the details of the results after extensive experiments with simulations of the GP-PSO hybrid algorithm. The structure of the rest of this chapter is as follows: Section 5.2 briefly describes the simulation setup of the hybrid approach. Section 5.3 focuses on presenting and analyzing results obtained from the simulations.

Root mean square error (RMS) between predicted and target data is considered as the ‘fitness’ of the hybrid approach. Our algorithm is two-objective: one objective is to minimize RMS between predicted and target bolting time and the other is to minimize RMS between predicted and target gene expression data. After simulation, a non-dominated front consisted of multiple solutions (gene networks) will be shown. One or two gene networks will be extracted from the non-dominated front and illustrated graphically. Parameters and functions associated with the gene network will also be demonstrated in tables. The synthetic network, which was referred to in Section 4.3, will be revealed. Predicted output from obtained networks (both bolting dates and gene expression) will be compared with that of the synthetic network (explained in Section 4.2). In addition, both obtained networks and synthetic networks will be compared and analyzed in mathematical equations. The last section of this chapter, Section 5.4, will address specific issues related to the obtained results.

5.2 Simulation Setup

For the multi-objective GP, a population of N = 50 solutions was used with a fixed archive size of 50. The maximum number of generations was set to 60 with a mutation rate of 12%. A population size of P = 50 was used for the PSO algorithm, and the total number of iterations per generation was set to 100. The PSO archive size was fixed at 50. The cognitive and social constants C_1, C_2 were set to 2.0 and 2.1, respectively. The constriction coefficient χ was set to 0.4.

From the gene identification stage, we were able to evaluate each gene's confidence level how that gene is likely to affect the final phenotype. But we did not have any prior knowledge of the number of genes that existed in the gene network. Thus multiple runs with different numbers of identified genes M = 6, 8, 10, 12, 14, 17, respectively, were carried out. Noisy phenotype data (STD = 2 days) were used in the genes number M = 17 run.

Based on the simulation setups, the number of function evaluations for each objective is $50*60*50*100 = 1.5\times10^7$.

5.3 Results

The synthetic network that was used to generate the problem data is shown in Figure 5.1. Its associated parameters and functions for each of the nodes are shown in Table 5.1. In the rest of this thesis, the parameters and functions table for gene networks will be presented in the following manner: the first column shows the gene index numbers in the network; the second column shows the corresponding functions; the third column indicates parameter symbols, the subscript parts of which imply the initial letter of the function and corresponding gene index number; the fourth column presents parameter values for allele marker #0; and the fifth column presents parameter values for allele marker #1.

Figure 5.1 The synthetic gene regulatory network

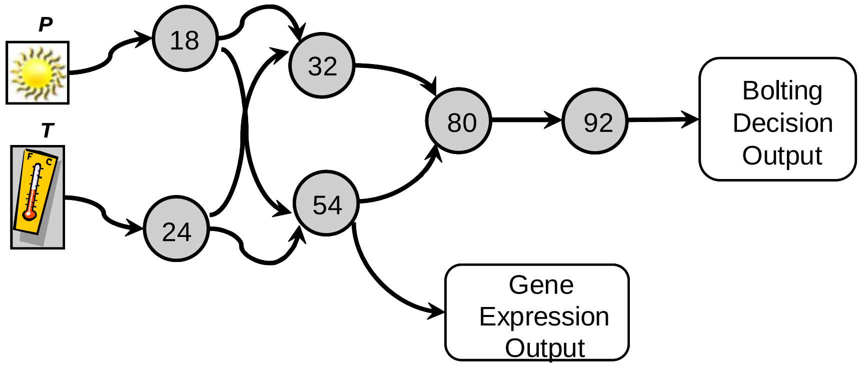


Table 5.1 Parameters and functions associated with synthetic gene network shown in figure 5.1

Gene #	**function**	**Parameter Symbol**	**Parameter Value for Allele #**	
			0	**1**
18	*gain*	$C_{g,18}$	*1.2*	*0.8*
24	*gain*	$C_{g,24}$	*0.8*	*1.2*
32	*summer*	$C_{s,32}$	*0.9*	*1.1*
54	*multiplier*	$C_{m,54}$	*8.1e-5*	*8.5e-5*
80	*summer*	$C_{s,80}$	*6.2e-4*	*6.3e-4*
92	*integrator*	$C_{i,92}$	*0.8*	*1.2*

In order to assess the performance of the proposed algorithm, solutions in the initial population as well as those obtained at the end of the algorithm's execution are compared. Figure 5.2 shows the non-dominated front obtained in one of the sample runs for $M = 6$. The points with the (+) sign represent the non-dominated solutions in the initial randomly generated population and points with the (*) sign represent the non-dominated solutions obtained at the end of the run. It can be seen from the figure that the algorithm has good convergence on both objective and at the same time achieved an evenly distributed final non-dominated front.

Figure 5.3 and Figure 5.4 show the network structures of two sample solutions selected from the final non-dominated solutions front for the run with $M = 6$, respectively. The predicted RMS errors of the sample #1 solution of the 6-gene run are 2.3 days (bolting date) and 0.0003 (gene expression level). It has all the genes present in the synthetic network, except for gene #54. Similarly, the RMS errors of its counterpart non-dominated sample #2 solution in our 6-gene run are 1.8 days and 0.0009, respectively.

Table 5.2 and Table 5.3 show the associated parameters and functions for each node illustrated in Figure 5.3 and 5.4 respectively.

Figure 5.2 Non-dominated solutions obtained from simulating *M* = 6 gene run

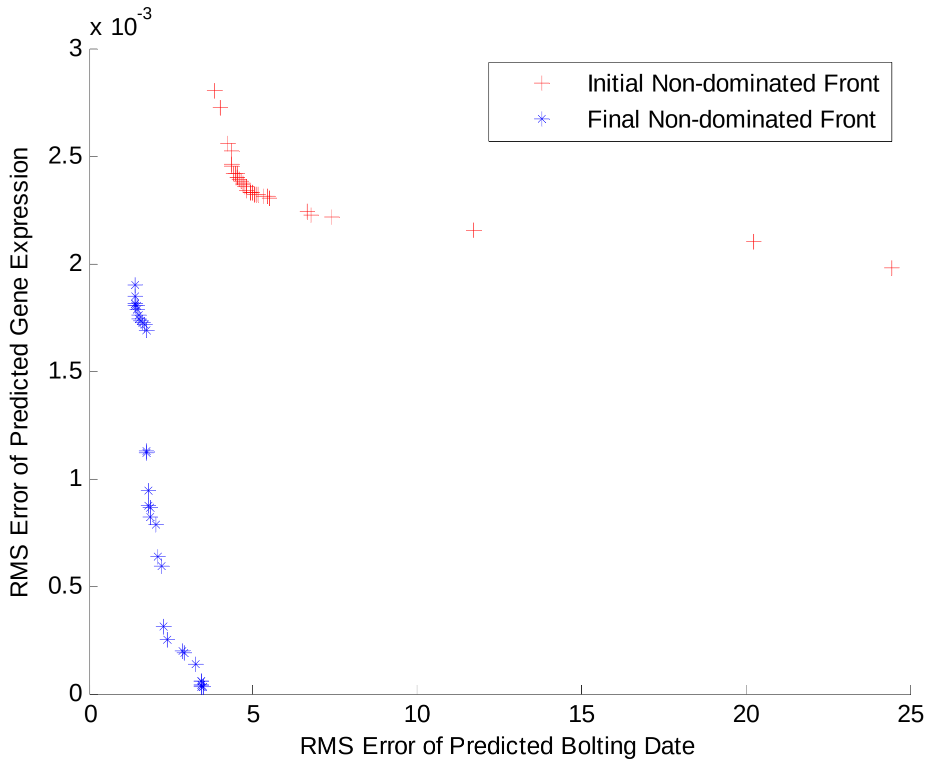


Figure 5.3 Sample network #1 of a solution obtained in *M* = 6 gene run

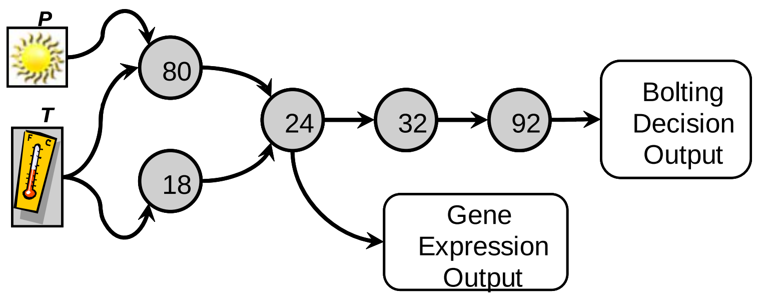


Figure 5.4 Sample network #2 of a solution obtained in *M* = 6 gene run

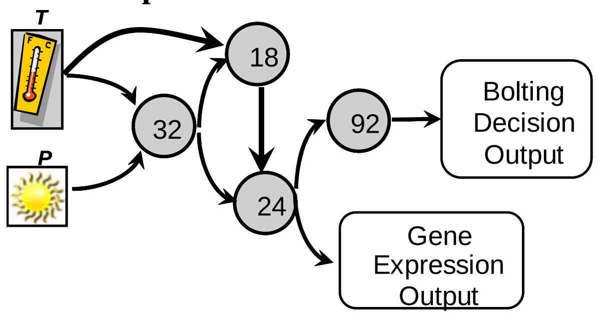


Table 5.2 Parameters and functions associated with the gene network shown in Figure 5.3

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
18	*gain*	$C_{g,18}$	*0.00064*	*0.000441*
24	*multiplier*	$C_{m,24}$	*0.12*	*0.18442*
32	*integrator*	$C_{i,32}$	*0.17884*	*0.20027*
54	-	C_{54}	-	-
80	*summer*	$C_{s,80}$	*0.058532*	*0.03856*
92	*integrator*	$C_{i,92}$	*0.49112*	*0.9257*

Table 5.3 Parameters and functions associated with the gene network shown in Figure 5.4

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
18	*summer*	$C_{s,18}$	*0.97461*	*0.01078*
24	*multiplier*	$C_{m,24}$	*1.6e-5*	*2.3e-5*
32	*summer*	$C_{s,32}$	*0.79734*	*0.810017*
54	-	C_{54}	-	-
80	-	C_{80}	-	-
92	*integrator*	$C_{i,92}$	*1.7585*	*2.8781*

Figure 5.5 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #1 of a solution obtained in *M* = 6 gene run)

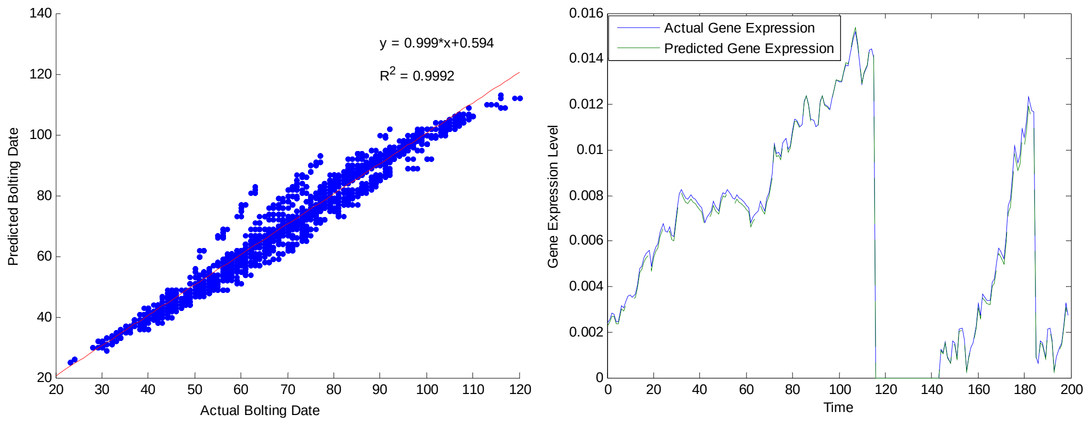
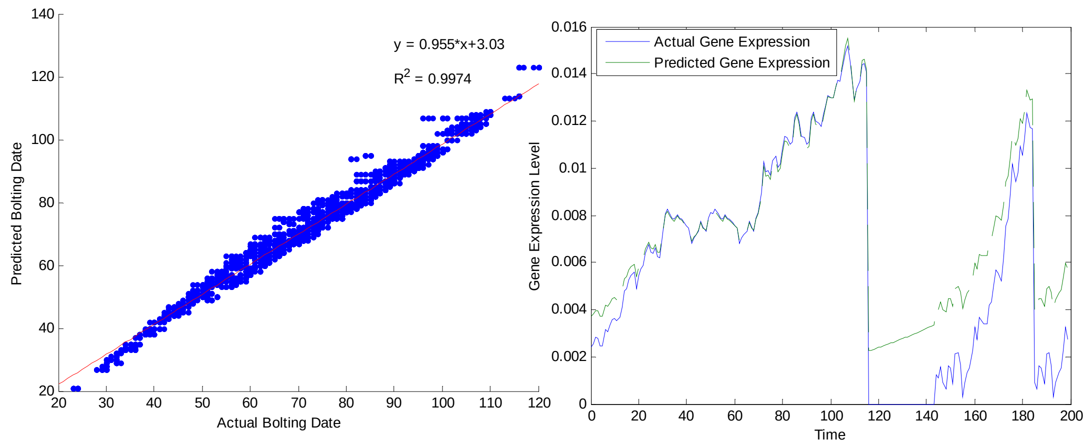


Figure 5.6 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #2 of a solution obtained in *M* = 6 gene run)



In order to give more visual and straightforward comparisons between synthetic networks and networks obtained by the proposed algorithm, comparisons of the predicted and actual bolting dates or gene expression data are shown in Figure 5.5 and Figure 5.6. In Figure 5.5/5.6 (left), a linear regression to bolting date comparison is given in right top corner and a coefficient of determination R^2 of linear regression is also presented.

Figure 5.7 Non-dominated solutions obtained from simulating *M* = 8 gene run

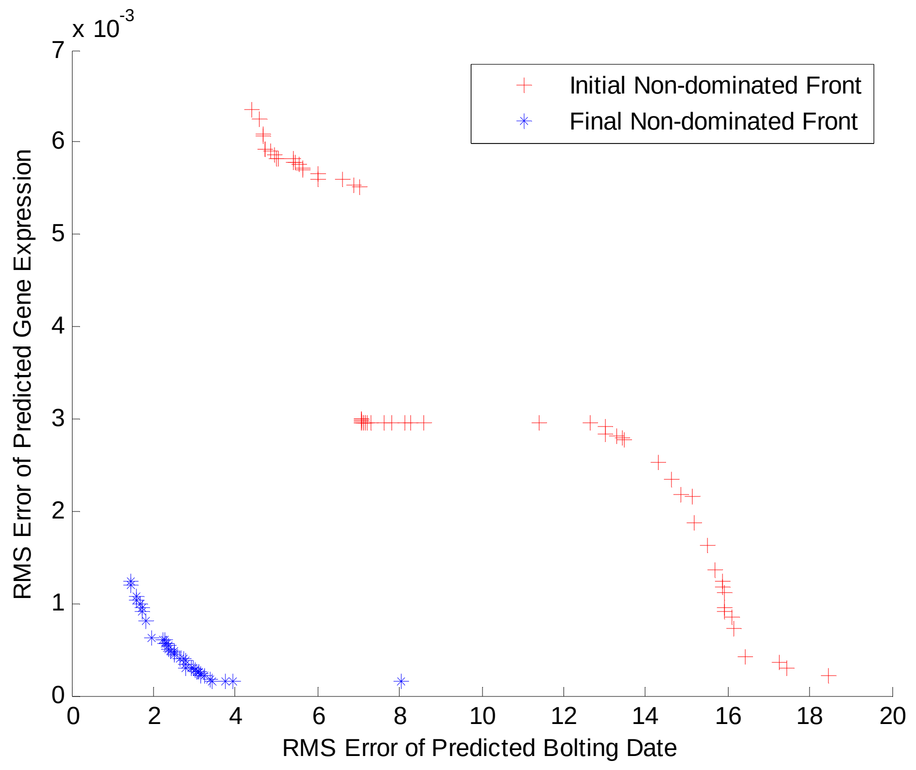


Figure 5.8 Sample network #1 of a solution obtained in *M* = 8 gene run

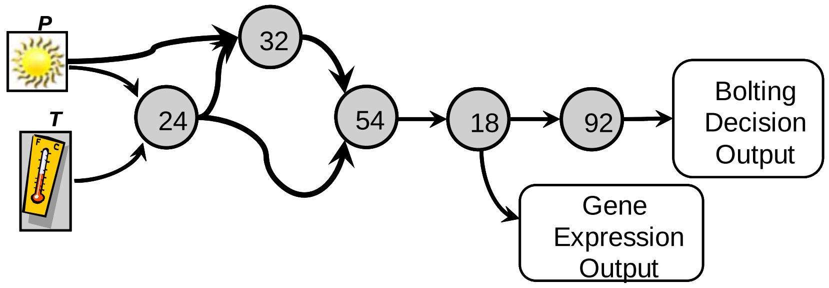


Figure 5.9 Sample network #2 of a solution obtained in *M* = 8 gene run

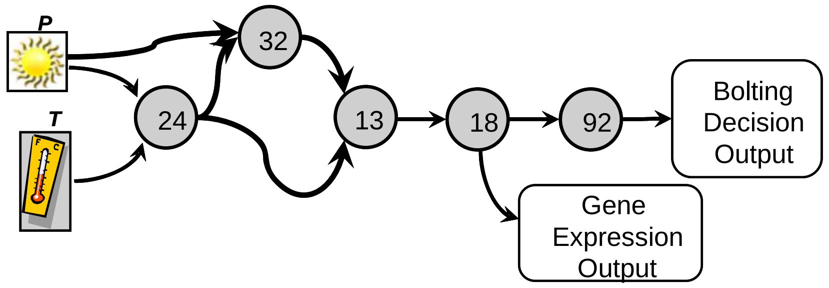


Table 5.4 Parameters and functions associated with the gene network shown in Figure 5.8

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
18	*gain*	$C_{g,18}$	*1.3286*	*0.88411*
24	*multiplier*	$C_{m,24}$	*5.6e-5*	*8.6e-5*
32	*summer*	$C_{s,32}$	*0.54132*	*0.54518*
54	*summer*	$C_{s,54}$	*2.6e-5*	*1.16e-4*
80	-	C_{80}	-	-
92	*integrator*	$C_{i,92}$	*1.8867*	*2.9725*

Table 5.5 Parameters and functions associated with the gene network shown in Figure 5.9

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
13	*summer*	$C_{s,13}$	*1.7747*	*1.8221*
18	*gain*	$C_{g,18}$	*2.24e-4*	*1.57e-4*
24	*multiplier*	$C_{m,24}$	*0.33378*	*0.49127*
32	*summer*	$C_{s,32}$	*0.00806*	*0.010978*
54	-	C_{54}	-	-
80	-	C_{80}	-	-
92	*integrator*	$C_{i,92}$	*1.7741*	*2.8085*

Figure 5.10 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #1 of a solution obtained in *M* = 8 gene run)

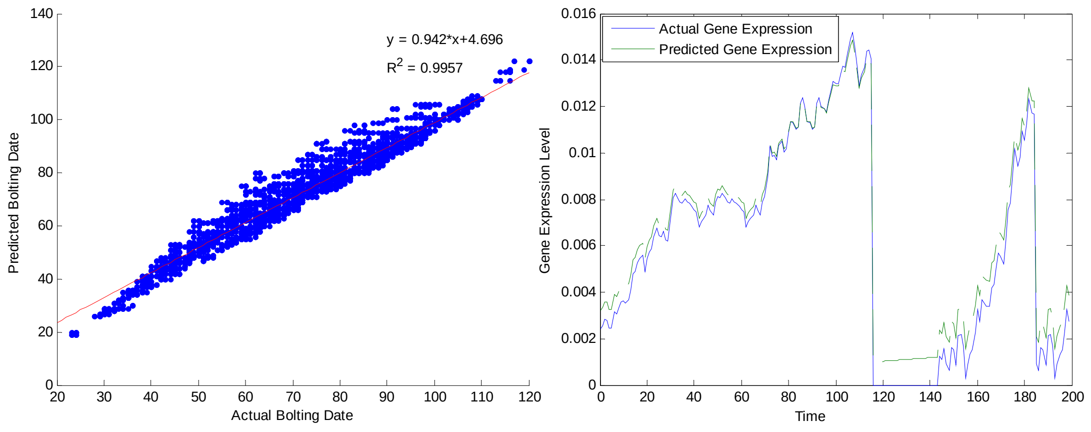
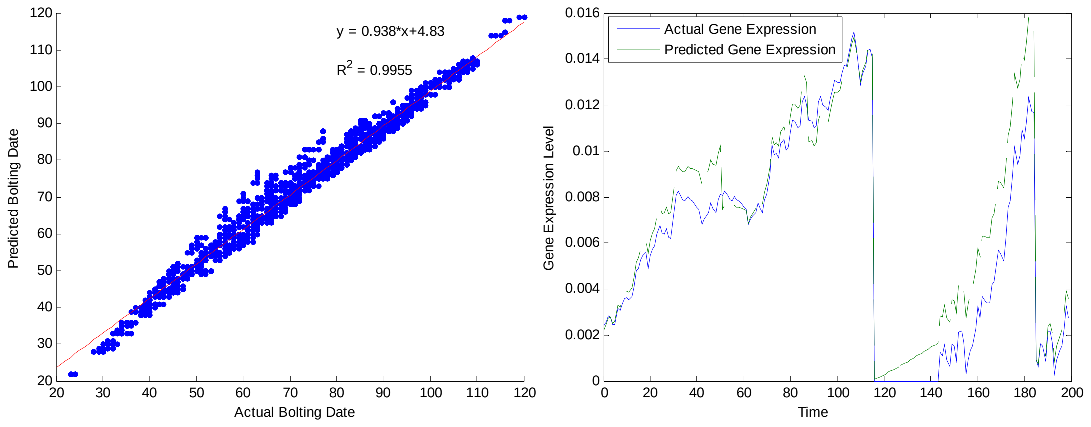


Figure 5.11 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #2 of a solution obtained in *M* = 8 gene run)



For *M* = 8 gene run, Figure 5.7 demonstrates the initial ('+') and final ('*') non-dominated front after applying our algorithm. Figures 5.8 and 5.9 present two sample networks of solutions from the 8 gene run, with RMS error of bolting dates 3.1 days and gene expression level 0.00025 in solution #1; 1.9 days and 0.0006 in solution #2. Tables 5.4 and 5.5 show the parameters and functions associated with the network in Figures 5.8 and 5.9 respectively. Figure 5.10 illustrates the comparison of actual versus predicted bolting dates and gene expression level generated by sample solution #1 while Figure 5.11 illustrates that of solution #2 in this 8 gene run.

Figure 5.12 Non-dominated solutions obtained from simulating *M* = 10 gene run

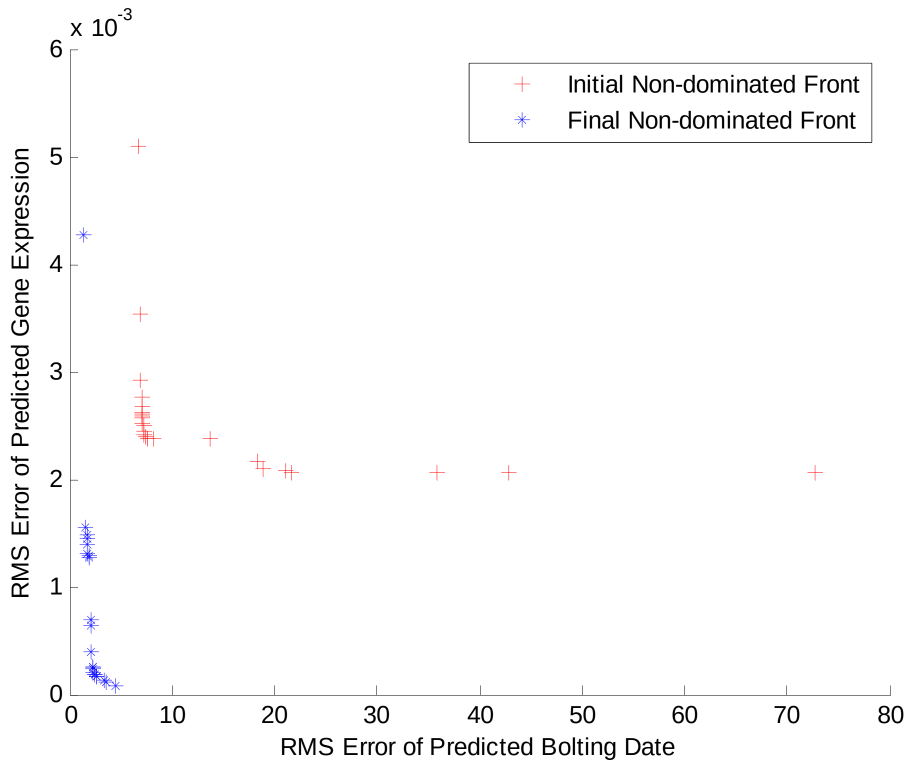


Figure 5.13 Sample network #1 of a solution obtained in *M* = 10 gene run

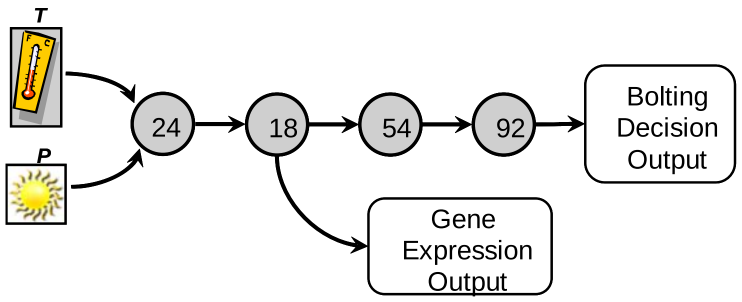


Figure 5.14 Sample network #2 of a solution obtained in *M* = 10 gene run

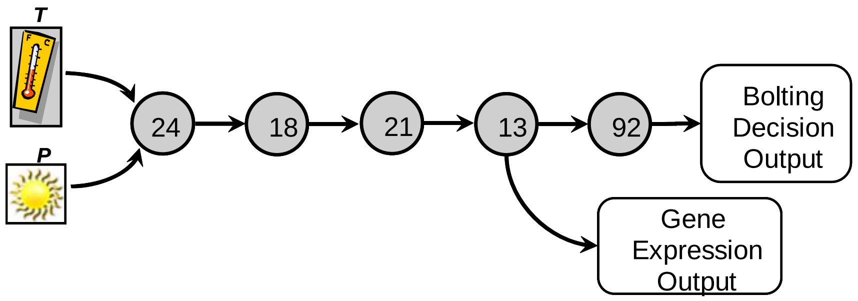


Table 5.6 Parameters and functions associated with the gene network shown in Figure 5.13

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
18	*gain*	$C_{g,18}$	*0.090229*	*0.059627*
24	*multiplier*	$C_{m,24}$	*0.000909*	*0.001338*
32	-	C_{32}	-	-
54	*integrator*	$C_{i,54}$	*0.84275*	*0.74101*
80	-	C_{80}	-	-
92	*integrator*	$C_{i,92}$	*0.11721*	*0.24078*

Table 5.7 Parameters and functions associated with gene network shown in Figure 5.14

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
13	*gain*	$C_{g,13}$	*0.009624*	*0.009305*
18	*gain*	$C_{g,18}$	*0.559969*	*0.37422*
21	*gain*	$C_{g,21}$	*0.17868*	*0.1715*
24	*multiplier*	$C_{m,24}$	*0.091111*	*0.13009*
32	-	C_{32}	-	-
54	-	C_{54}	-	-
80	-	C_{80}	-	-
92	*integrator*	$C_{i,92}$	*1.8221*	*2.9829*

Figure 5.15 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #1 of a solution obtained in *M* = 10 gene run)

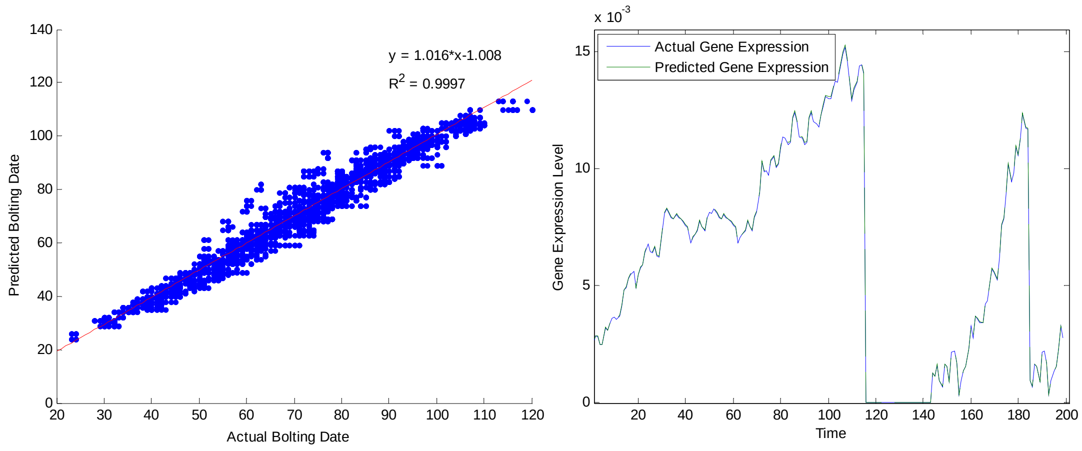
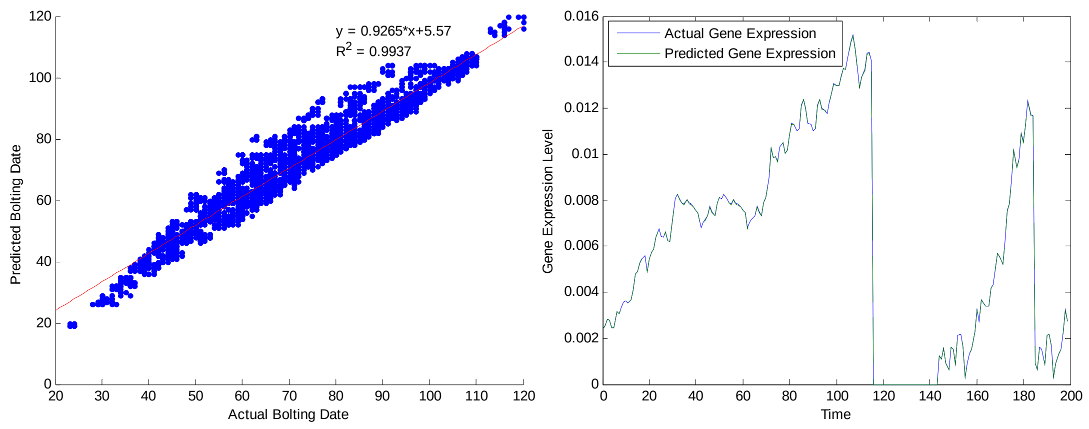


Figure 5.16 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #2 of a solution obtained in *M* = 10 gene run)



For the *M* = 10 gene run, convergence and diversity of the final non-dominated front are shown and compared with that of the initial non-dominated front in Figure 5.12. Two networks of solutions selected from the final non-dominated front are given in Figures 5.13 and 5.14 respectively. The former solution has RMS error 2.4 days in bolting dates and 0.0002 in gene expression level; the latter solution has RMS error 3.4 days in bolting dates and 0.00013 in gene expression level. Their associated parameters and functions are demonstrated in Tables 5.6 and 5.7 correspondingly. Comparison of actual versus predicted bolting dates and gene expression of these two solutions are shown in Figures 5.15 and 5.16 respectively.

Figure 5.17 Non-dominated solutions obtained from simulating *M* = 12 gene run

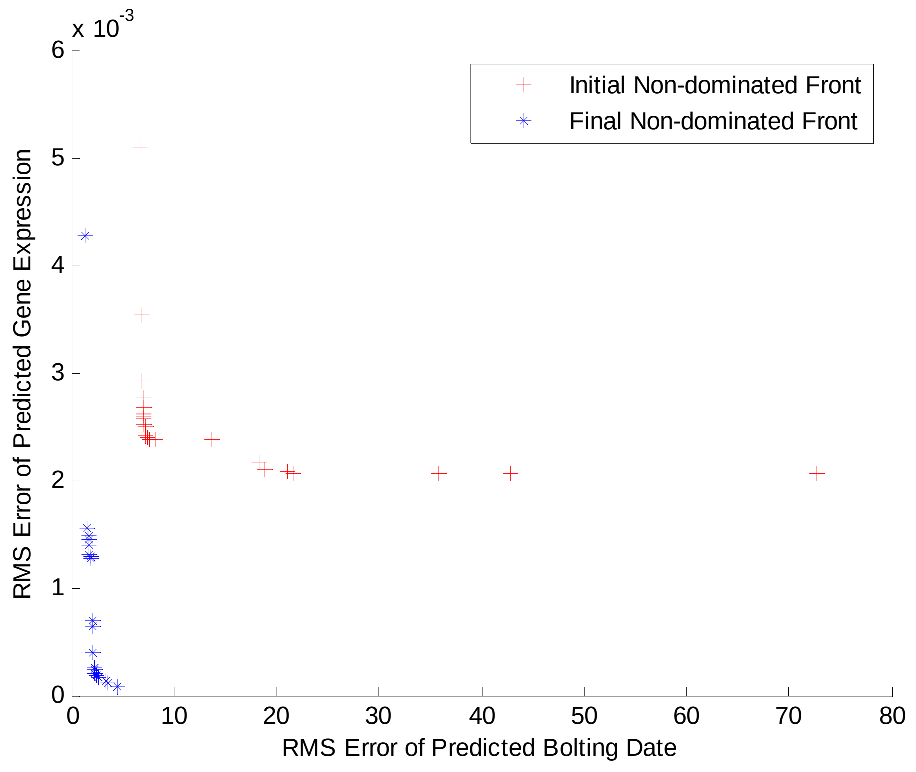


Figure 5.18 Sample network of a solution obtained in *M* = 12 gene run

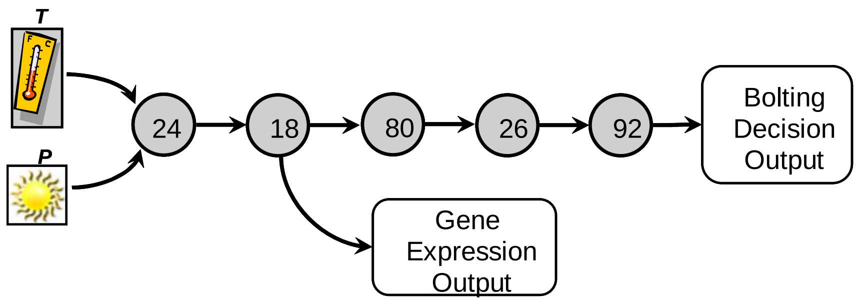


Table 5.8 Parameters and functions associated with the gene network shown in Figure 5.18

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	1
18	*gain*	$C_{g,18}$	*0.00472*	*0.003153*
24	*Multiplier*	$C_{m,24}$	*0.017231*	*0.02622*
26	*Integrator*	$C_{i,26}$	*1.2395*	*1.2737*
32	-	C_{32}	-	-
54	-	C_{54}	-	-
80	*Integrator*	$C_{i,80}$	*0.5056*	*0.48973*
92	*Gain*	$C_{g,92}$	*0.14633*	*0.28778*

Figure 5.19 Comparison of actual vs. predicted bolting dates and gene expression (Sample network of a solution obtained in *M* = 12 gene run)

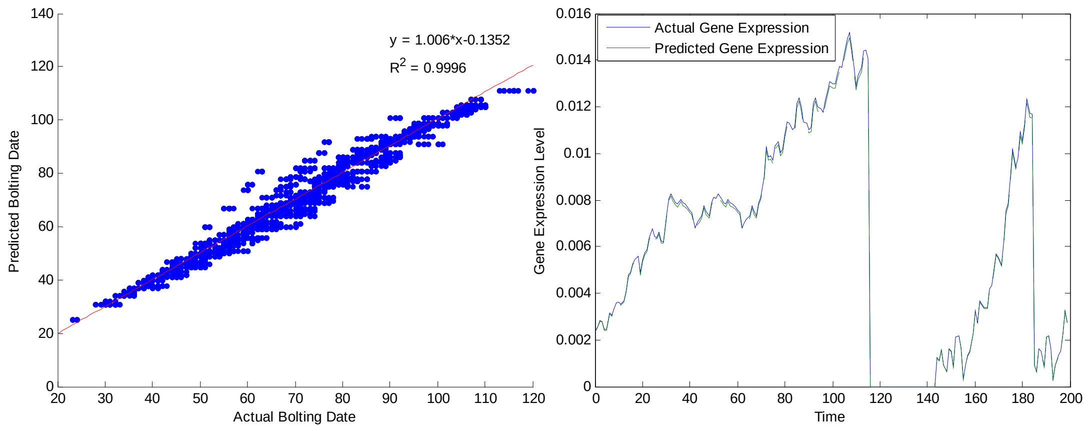


Figure 5.17 illustrates the non-dominated front obtained from the 12 gene run. One solution withdrawn from the front is demonstrated in Figure 5.18. Its associated parameters and functions are listed in Table 5.8. Figure 5.19 shows the comparison of actual versus predicted data for this solution. The RMS error in bolting dates and gene expression level for this solution are 2.3 days and 0.00025.

Figure 5.20 Non-dominated solutions obtained from simulating *M* = 14 gene run

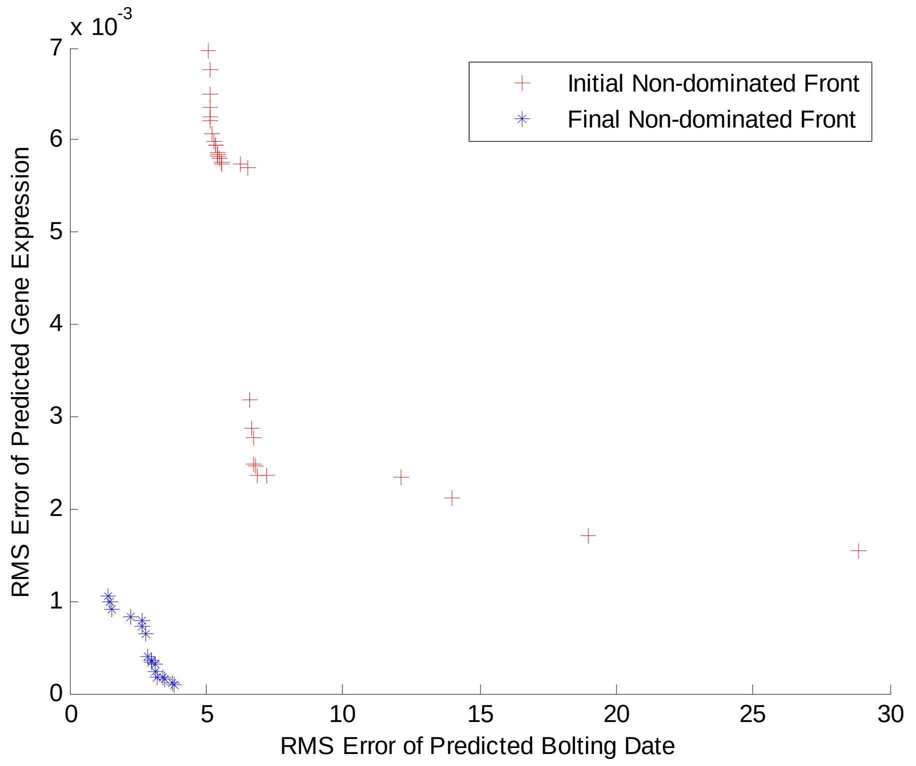


Figure 5.21 Sample network #1 of a solution obtained in *M* = 14 gene run

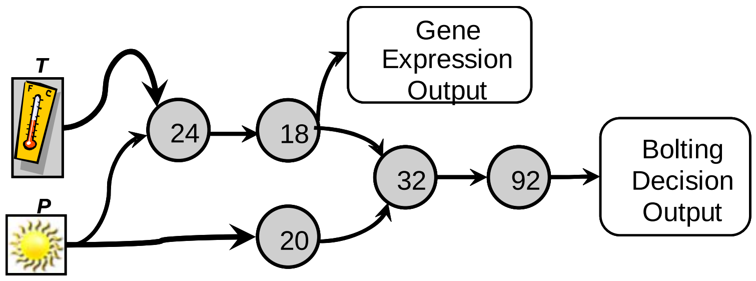


Figure 5.22 Sample network #2 of a solution obtained in *M* = 14 gene run

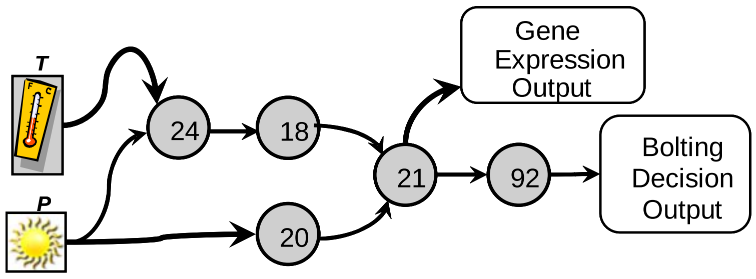


Table 5.9 Parameters and functions associated with the gene network shown in Figure 5.21

Gene #	**Function**	**Parameter Symbol**	**Parameter Value for Allele #**	
			0	**1**
18	*gain*	$C_{g,18}$	*0.00032*	*0.000212*
20	*integrator*	$C_{i,20}$	*0.73234*	*0.71198*
24	*multiplier*	$C_{m,24}$	*0.26749*	*0.38329*
32	*multiplier*	$C_{m,32}$	*0.038691*	*0.040016*
54	-	C_{54}	-	-
80	-	C_{80}	-	-
92	*gain*	$C_{i,92}$	*0.14811*	*0.36625*

Table 5.10 Parameters and functions associated with the gene network shown in Figure 5.22

Gene #	**function**	**Parameter Symbol**	**Parameter Value for Allele #**	
			0	**1**
18	*gain*	$C_{g,18}$	*0.75968*	*0.498452*
20	*integrator*	$C_{i,20}$	*4e-6*	*6e-6*
21	*summer*	$C_{s,21}$	*8.0e-5*	*7.4e-5*
24	*multiplier*	$C_{m,24}$	*1.3153*	*1.8698*
32	-	C_{32}	-	-
54	-	C_{54}	-	-
80	-	C_{80}	-	-
92	*gain*	$C_{g,92}$	*1.6466*	*2.8733*

Figure 5.23 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #1 of a solution obtained in *M* = 14 gene run)

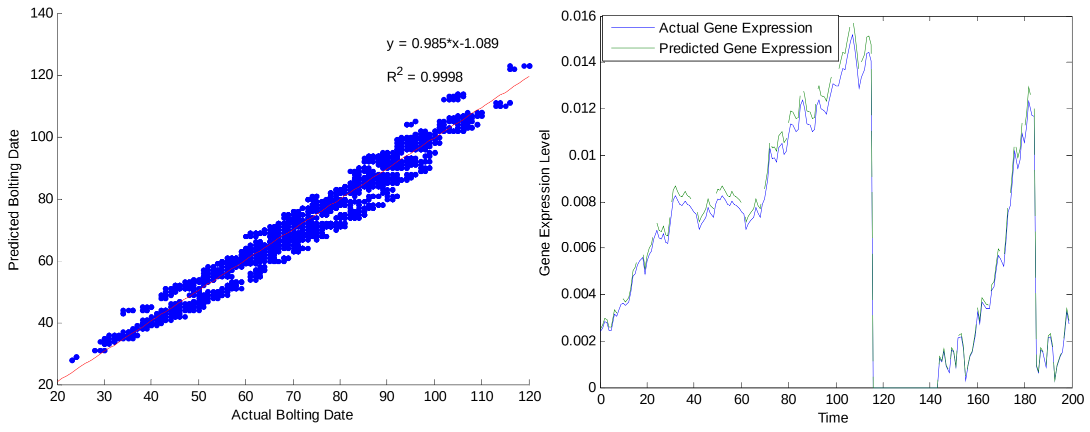
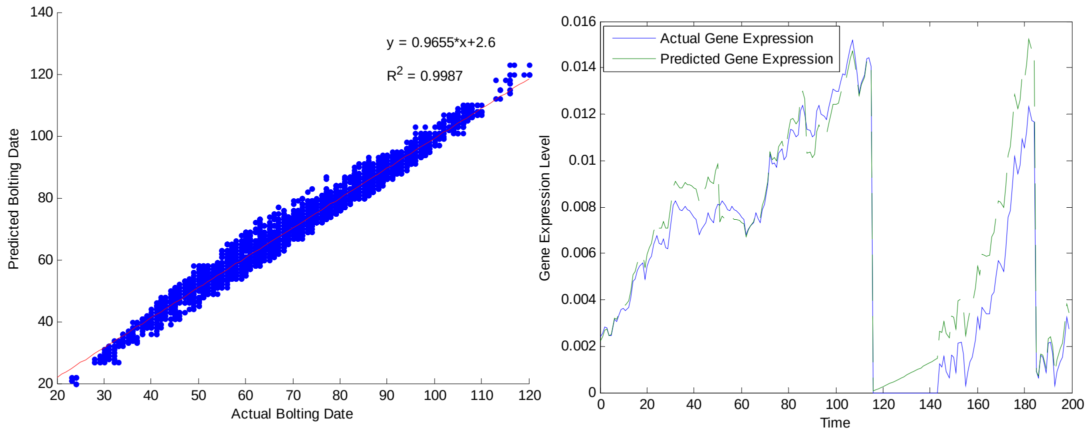


Figure 5.24 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #2 of a solution obtained in *M* = 14 gene run)



For the *M* = 14 gene run, Figure 5.20 shows the final non-dominated front after applying the proposed approach. Two networks of the solutions selected from the final non-dominated front are given in Figures 5.21 and 5.22 respectively. The former solution has RMS error 2.9 days in bolting dates and 0.0004 in gene expression level; the latter solution has RMS error 1.5 days in bolting dates and 0.0009 in gene expression level. Their associated parameters and functions are demonstrated in Tables 5.9 and 5.10 correspondingly. Comparison of actual versus predicted bolting dates and gene expression of these two solutions are shown in Figures 5.23 and 5.24 respectively.

Figure 5.25 Non-dominated solutions obtained from simulating *M* = 17 gene run

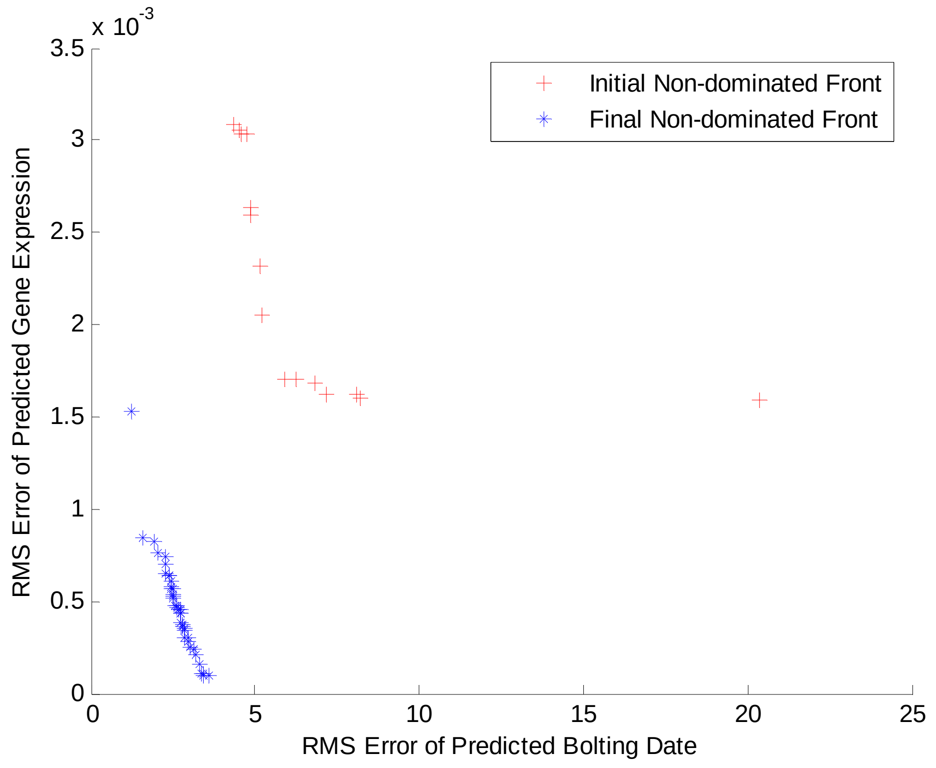


Figure 5.26 Sample network #1 of a solution obtained in *M* = 17 gene run

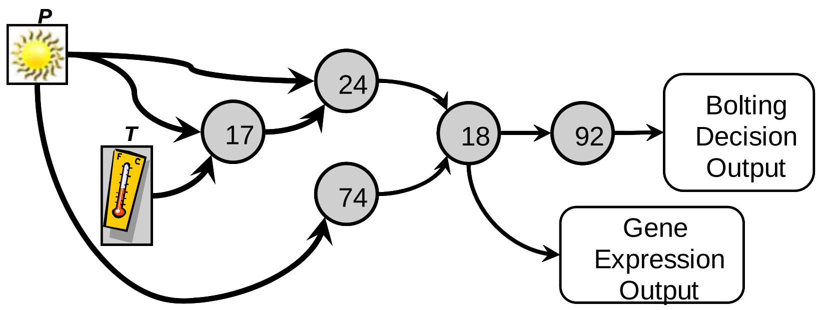


Figure 5.27 Sample network #2 of a solution obtained in *M* = 17 gene run

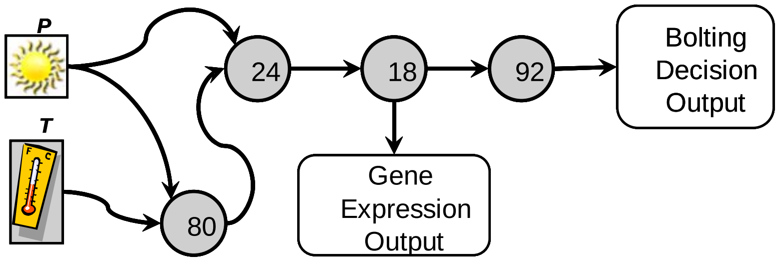


Table 5.11 Parameters and functions associated with the gene network shown in Figure 5.26

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	**1**
17	*summer*	$C_{s,17}$	*0.01*	*0.0235*
18	*summer*	$C_{s,18}$	*1.2*	*0.80961*
24	*multiplier*	$C_{m,24}$	*5.2e-005*	*8.3e-005*
32	-	C_{32}	-	-
54	-	C_{54}	-	-
74	*gain*	$C_{g,74}$	*1.69e-004*	*1.65e-004*
80	-	C_{80}	-	-
92	*integrator*	$C_{i,92}$	*1.7146*	*2.7925*

Table 5.12 Parameters and functions associated with the gene network shown in Figure 5.27

Gene #	function	Parameter Symbol	Parameter Value for Allele #	
			0	**1**
18	*gain*	$C_{g,18}$	*0.002053*	*0.001403*
24	*multiplier*	$C_{m,24}$	*0.036626*	*0.054389*
32	-	C_{32}	-	-
54	-	C_{54}	-	-
80	*summer*	$C_{s,80}$	*0.05222*	*0.038843*
92	*integrator*	$C_{i,92}$	*1.7453*	*3.3096*

Figure 5.28 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #1 of a solution obtained in *M* = 17 gene run)

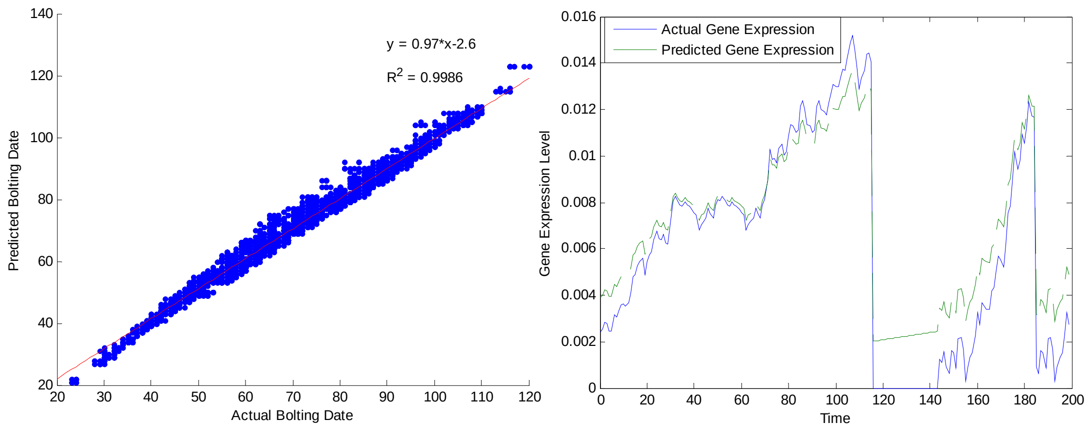
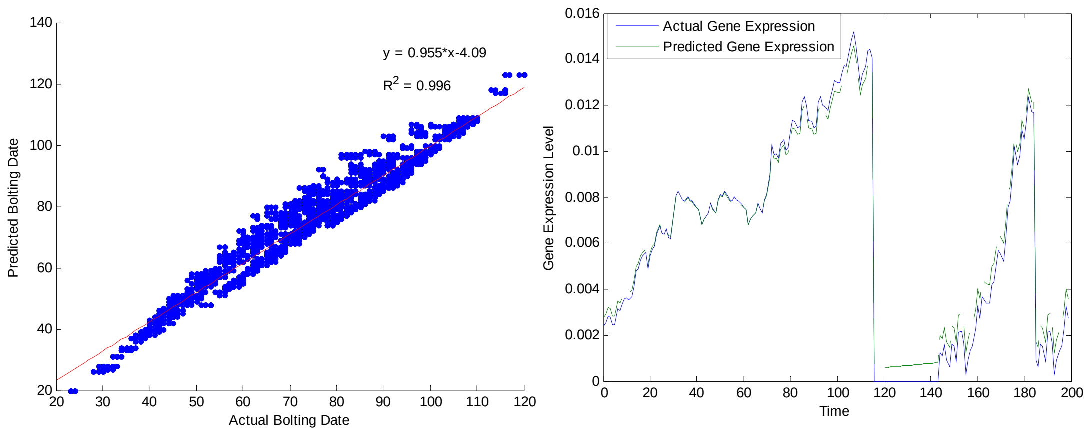


Figure 5.29 Comparison of actual vs. predicted bolting dates and gene expression (Sample network #2 of a solution obtained in *M* = 17 gene run)



The *M* = 17 gene run is the run with the most potential candidate genes to form a solution. Figure 5.25 shows the initial non-dominated front in '+' and final non-dominated front in '*'. The number of non-dominated solutions increases significantly and both good convergence and diversity are obtained at the end of proposed algorithm compared to initial non-dominated solutions. Figures 5.26 and 5.27 show one sample solution from the final non-dominated solutions and Tables 5.11 and 5.12 are the parameters and functions associated with them respectively. The former solution has RMS error of 3.4 days in bolting dates and 0.0001 in gene expression level; the latter solution has RMS error of 1.6 days in bolting dates and 0.00085 in gene expression level. Both solution #1 and #2 are able to recover 4 correct genes out of 17 candidate genes compared to the 6 gene synthetic network. Comparison of actual versus predicted bolting dates and gene expression are shown in Figures 5.28 and 5.29 respectively for these two solutions. All of these figures indicate that the generated network is able to accurately predict the response of the synthetic gene network.

A more discerning numerical comparison of results for all of the runs is given in Table 5.13 by substituting the gene functions at each of the nodes, their associated parameters and evaluating the mathematical equivalent expressions for each network. The raw mathematical expressions do not appear similar. However, after parameter substitution (allele 0 shown) similarities emerge in terms ‘$T * P$’ and ‘$O_{92}(t\text{-}1)$’, which are vital components in estimating the bolting date.

Table 5.13 Numerical formulas before and after parameter substitution

Network	Bolting date prediction gene Output	Parameters substituted Bolting date prediction gene Output
Synthetic Gene Network	$O(t) = C_{18} * C_{32} * C_{80} * C_{92} * T$ $+ C_{24} * C_{80} * C_{92} * P$ $+ C_{18} * C_{24} * C_{54} * C_{92} * T * P$ $+ O_{92}(t-1)$	$O(t) = 5.35 \times 10^{-04} * T$ $+ 4 \times 10^{-04} * P$ $+ 6.2 \times 10^{-05} * T * P$ $+ O_{92}(t-1)$
Sample Gene Network#1 from *M*=6 run	$O(t) = C_{18} * C_{24} * C_{32} * C_{80} * C_{92} * T^2$ $+ C_{18} * C_{24} * C_{32} * C_{92} * T * P$ $+ C_{92} * O_{32}(t-1)$ $+ O_{92}(t-1)$	$O(t) = 3.94 \times 10^{-07} * T^2$ $+ 6.7 \times 10^{-06} * T * P$ $+ 0.49 * O_{32}(t-1)$ $+ O_{92}(t-1)$
Sample Gene Network#2 from *M*=6 run	$O(t) = C_{18} * C_{24} * C_{32} * C_{92} * T^2$ $+ (C_{18} * C_{24} * C_{92} + C_{24} * C_{32} * C_{92})$ $* T * P$ $+ C_{24} * C_{92} * P^2$ $+ O_{92}(t-1)$	$O(t) = 2.1 \times 10^{-05} * T^2$ $+ 4.97 \times 10^{-05} * T * P$ $+ 2.8 \times 10^{-05} * P^2$ $+ O_{92}(t-1)$

Sample Gene Network#1 from *M*=8 run	$O(t) = C_{18} * C_{32} * C_{54} * C_{92} * P + (C_{18} * C_{24} * C_{54} * C_{92} + C_{18} * C_{24} * C_{92}) * T * P + O_{92}(t-1)$	$O(t) = 3.5 \times 10^{-05} * P + 1.4 \times 10^{-04} * T * P + O_{92}(t-1)$
Sample Gene Network#2 from *M*=8 run	$O(t) = C_{13} * C_{18} * C_{32} * C_{92} * P + C_{18} * C_{24} * C_{92} * T * P + C_{13} * C_{18} * C_{92} * O_{32}(t-1) + O_{92}(t-1)$	$O(t) = 5.6 \times 10^{-06} * P + 1.3 \times 10^{-04} * T * P + 7 \times 10^{-04} * O_{32}(t-1) + O_{92}(t-1)$
Sample Gene Network#1 from *M*=10 run	$O(t) = C_{18} * C_{24} * C_{54} * C_{92} * T * P + C_{92} * O_{54}(t-1) + O_{92}(t-1)$	$O(t) = 8 \times 10^{-06} * T * P + 0.84 * O_{54}(t-1) + O_{92}(t-1)$
Sample Gene Network#2 from *M*=10 run	$O(t) = C_{13} * C_{18} * C_{21} * C_{24} * C_{92} * T * P + O_{92}(t-1)$	$O(t) = 1.45 \times 10^{-04} * T * P + O_{92}(t-1)$
Sample Gene Network#1 from *M*=12 run	$O(t) = C_{18} * C_{24} * C_{80} * C_{92} * T * P + C_{26} * C_{92} * O_{80}(t-1) + C_{92} * O_{26}(t-1)$	$O(t) = 7.26 \times 10^{-06} * T * P + 0.09 * O_{80}(t-1) + 0.15 * O_{26}(t-1)$
Sample Gene Network#1 from *M*=14 run	$O(t) = C_{18} * C_{20} * C_{24} * C_{32} * C_{92} * T * P^2 + C_{18} * C_{24} * C_{32} * C_{92} * T * P * O_{20}(t-1) + O_{92}(t-1)$	$O(t) = 3.6 \times 10^{-07} * T * P^2 + 4.5 \times 10^{-07} * T * P * O_{20}(t-1) + O_{92}(t-1)$
Sample Gene Network#2 from *M*=14 run	$O(t) = C_{18} * C_{21} * C_{24} * C_{92} * T * P + C_{20} * C_{92} * P + C_{92} * O_{20}(t-1) + O_{92}(t-1)$	$O(t) = 1.3 \times 10^{-05} * T * P + 6.6 \times 10^{-06} * P + 1.65 * O_{20}(t-1) + O_{92}(t-1)$

Sample Gene Network#1 from *M*=17 run	$O(t) = C_{18} * C_{24} * C_{92} * T * P + C_{17} * C_{18} * C_{24} * C_{92} * P^2 + C_{18} * C_{24} * C_{74} * C_{92} * P + O_{92}(t-1)$	$O(t) = 1.1\times10^{-04} * T * P + 1.1\times10^{-06} * P^2 + 1.8\times10^{-08} * P + O_{92}(t-1)$
Sample Gene Network #2 from *M*=17 run	$O(t) = C_{18} * C_{24} * C_{80} * C_{92} * P^2 + C_{18} * C_{24} * C_{92} * T * P + O_{92}(t-1)$	$O(t) = 6.67\times10^{-06} * P^2 + 1.3\times10^{-04} * T * P + O_{92}(t-1)$

Table 5.14 Sensitivity analysis to numerical formulas in Table 5.13

Term Relative Sensitivity	Numerical Formula for Bolting Dates Prediction	*T*	*P*	**T*P*	*T*^{2}	*P*^{2}	*T* P*^{2}
Synthetic Gene Network	$O(t) = 5.35\times10^{-04} * T + 4\times10^{-04} * P + 6.2\times10^{-05} * T * P + O_{92}(t-1)$	1.97	7.16	8.6	-	-	-
Sample Gene Network#1 from *M*=6 run	$O(t) = 3.94\times10^{-07} * T^2 + 6.7\times10^{-06} * T * P + 0.49 * O_{32}(t-1) + O_{92}(t-1)$	-	-	9.72	5.52	-	-
Sample Gene Network#2 from *M*=6 run	$O(t) = 2.1\times10^{-05} * T^2 + 4.97\times10^{-05} * T * P + 2.8\times10^{-05} * P^2 + O_{92}(t-1)$	-	-	7.24	7.15	88	-
Sample Gene Network#1 from *M*=8 run	$O(t) = 3.5\times10^{-05} * P + 1.4\times10^{-04} * T * P + O_{92}(t-1)$	-	7.35	6.34	-	-	-

Sample Gene Network#2 from *M*=8 run	$O(t)=5.6\times10^{-06}*P$ $+1.3\times10^{-04}*T*P$ $+7\times10^{-04}*O_{32}(t-1)$ $+O_{92}(t-1)$	-	1.425	1.44	-	-	-
Sample Gene Network#1 from *M*=10 run	$O(t)=8\times10^{-06}*T*P$ $+0.84*O_{54}(t-1)$ $+O_{92}(t-1)$	-	-	7.44	-	-	-
Sample Gene Network#2 from *M*=10 run	$O(t)=1.45\times10^{-04}*T*P$ $+O_{92}(t-1)$	-	-	5.99	-	-	-
Sample Gene Network#1 from *M*=12 run	$O(t)=7.26\times10^{-06}*T*P$ $+0.09*O_{80}(t-1)$ $+0.15*O_{26}(t-1)$	-	-	1.7438	-	-	-
Sample Gene Network#1 from *M*=14 run	$O(t)=3.6\times10^{-07}*T*P^{2}$ $+4.5\times10^{-07}*T*P*O_{20}(t-1)$ $+O_{92}(t-1)$	-	-	-	-	-	7.942
Sample Gene Network#2 from *M*=14 run	$O(t)=1.3\times10^{-05}*T*P$ $+6.6\times10^{-06}*P$ $+1.65*O_{20}(t-1)$ $+O_{92}(t-1)$	-	6.88	5.93	-	-	-
Sample Gene Network#1 from *M*=17 run	$O(t)=1.1\times10^{-04}*T*P$ $+1.1\times10^{-06}*P^{2}$ $+1.8\times10^{-08}*P$ $+O_{92}(t-1)$	-	8.514	10.264	-	77.2	-

Sample Gene Network #2 from *M*=17 run	$O(t)=6.67\times10^{-06}*P^2$ $+1.3\times10^{-04}*T*P$ $+O_{92}(t-1)$	-	-	8.704	-	66.38	-

We carried out a sensitivity analysis to see how the variation in the bolting date outputs predicted by mathematical models can be apportioned, both qualitatively and quantitatively, to the variation in the parameters of each term in Table 5.13. The results of sensitivity analysis are shown in Table 5.14. The *sensitivity* of formulas is considered as ratio of the relative change in the output *BD* to the relative change in each term's weight σ, $\left|\frac{(BD_2-BD_1)/BD_2}{(\sigma_2-\sigma_1)/\sigma_2}\right|$, where $\sigma_1=10^{-8}$, $\sigma_2=10^{-2}$. *BD* denotes average bolting dates simulated under 18 different planting environments.

From Table 5.14, it is clear that the most significant terms of the numerical equation for a synthetic network are '*P*' and '*T * P*' with condition number 7.16 and 8.6 respectively. Contrarily, the '*T*' term has a condition number of 1.97 which is significantly small and can be neglected. Based on such simplification, the sample gene network #1 from the *M* = 8 run is the best network. Its structure and condition number of both '*P*' and '*T * P*' terms are very close to those of the synthetic network.

5.4 Further Result Discussion

The synthetic network consists of 6 genes with indexes #18, 24, 32, 54, 80 and 92, out of genomes made up of 100 gene markers. Putting aside the factor of noise, this indicates that the variation of phenotype data (either bolting dates or expression data) obtained at two different mutant genotypes with the identical environmental inputs would be observed only when the alleles on one or multiple loci of these 6 genes vary within the genotype pair. In order to discover the degree of each allele switch that causes phenotype variation, an analysis based on phenotype data obtained from the synthetic network at different mutant genotypes is performed.

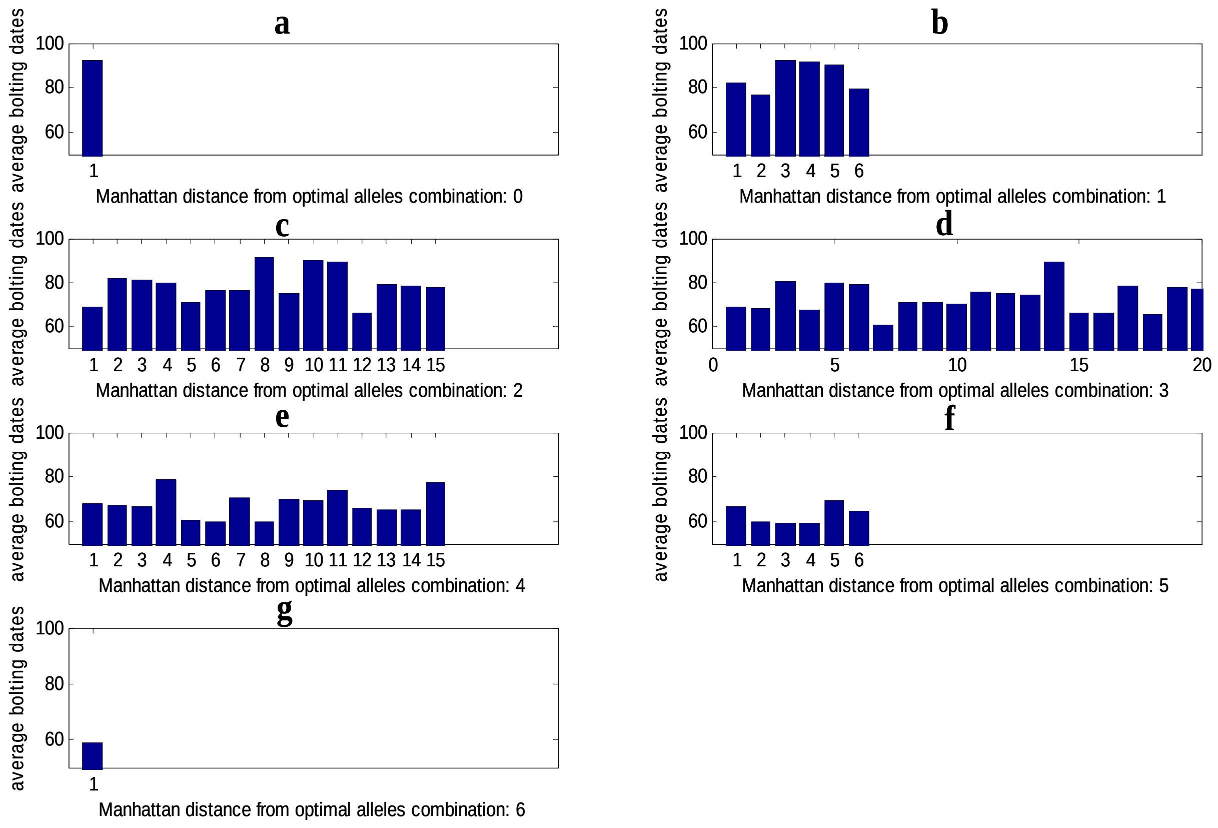
Recall from Section 4.2, each gene has two mutant alleles (represented by marker '0' or '1'). A genotype is constructed as a string of markers, representing different allelic combination.

The size of the string is the number of genes in the network. After simulating all the possible combinations, the genotype with allelic string ‘100001’ is found to have the latest bolting dates, as shown in Figure 5.30(a). Figure 5.30(b) shows the bolting dates for allelic strings with one bit different from that in Figure 5.30(a) (the Manhattan distance is 1). Figure 5.30(c), (d), (e), (f) and (g) illustrate the bolting dates for allelic strings with two, three, four, five and six bits different from that in Figure 5.30(a), respectively.

From Figure 5.30, it can be seen that genes #18, 24 and 92 have a significant effect and gene #80 has some impact on the phenotype prediction. On the other hand, genes #32 and 54 have very little impact on phenotype. The gene networks obtained from the proposed algorithms are consistent with our discovery from phenotype analysis.

Although different genes are contained in gene networks obtained in different runs, the important genes #18, 24 and 92 which have significant impact on phenotype are found in all of the obtained networks, indicating that our algorithm is capable of capturing the important genes that are significant enough to be not affected by incidental factors. All of the inferred network structures are able to predict phenotype data very close to real data. In addition, all the network structures acquired by the proposed approach are small networks, indicating that CGP has been able to reduce the known problem of bloating, which is often seen in GP.

Figure 5.30 Impact of 6 genes on phenotype of bolting dates



CHAPTER 6 - Network Assisted Selection for Breeding

6.1 Introduction

In the previous chapter, we showed how small plausible gene networks can be derived from phenotypic data, such as bolting dates and gene expression, using multi-objective stochastic optimization techniques. This thesis also includes a proposal for providing breeding strategies in plants based on computer simulation.

Plant breeding is a process of using deliberate crosses of related individuals to produce desirable lines. Breeding relies on new combination of chromosomes or recombination within chromosomes to generate new lines and a selection strategy to keep lines with desired characteristics. In the commonly used selection strategies, *marker assisted selection* (MAS) is based on the allele marker(s) linked to a trait (phenotype) of an individual [112]. The technique has accelerated breeding and has improved the accuracy of crosses compared to selection of phenotypes alone and allowed breeders to produce new lines with combined traits that were impossible before [113].

Marker assisted breeding does not account for phenotypic behavior that arises from interaction between genes for which no single gene is individually responsible. This phenomenon is called epistasis. Epistasis has been modeled by Kauffman using the *NK* fitness landscape [114] [115].

In this chapter, we make use of the *NK* fitness landscape for a theoretical study on marker assisted breeding and we also propose an approach that considers epistasis. The remainder of this chapter is organized as follows. The *NK* fitness landscape model will be introduced first. We will also illustrate the concept of applying this method in NK fitness landscape models to show that the proposed selection strategy may potentially produce faster improvements. Next, the network obtained by our GP-PSO hybrid algorithm will be applied in breeding experiments. Comparison of different breeding strategies by computer simulation will also be shown.

6.2 *NK* Fitness Landscape

Stuart Kauffman devised the '*NK* fitness landscape' model to explore the way epistasis control the ruggedness of an adaptive landscape, where *N* indicates the number of genes in the model and epistasis *K* the interaction between genes [114][115].

The NK model can be considered as a stochastic method for generating a fitness function $\boldsymbol{F} : \{0,1\}^N \to \Re^+$, on a binary string, $\boldsymbol{x} \in \{0,1\}^N$, where the genotype $\boldsymbol{x}$ consists of *N* loci, with two possible alleles at each locus x_i. Kauffman conceives of each gene as contributing a fitness component. So the fitness function can be further generalized as the average of *f* fitness components F_i contributed by each locus *i*. Each fitness component F_i is determined by its own allele x_i, and also the alleles at *K* other epistatic loci that affects it. Thus the fitness function can be written in the following form:

$$F(\boldsymbol{x}) = \frac{1}{f}\sum_{i=1}^{N} F_i\left(x_{j_1(i)}, x_{j_2(i)}, .. x_{j_{K+1}(i)}\right), \qquad \text{Eq. 6.1}$$

where $\{j_1(i), j_2(i), ..., j_{K+1}(i)\} \subset \{1, ..., N\}$. The index sets $\{j_1(i), j_2(i), ..., j_{K+1}(i)\}$ comprise a gene-fitness map that can be represented as a $f \times N$ matrix $\boldsymbol{M} = \lfloor m_{ij} \rfloor, i = 1...f, j = 1...N$, where $m_{ij} \in \{0,1\}$ and m_{ij} indicates whether locus *j* contributes to fitness component *i* with '1' representing yes and '0' representing no. It is assumed that that each fitness component is affected by one gene, and vice versa.

Letting $\boldsymbol{r}_i$ be the rows of ***M***, where $\boldsymbol{r}_i = \lfloor m_{ij} \rfloor, j = 1...N$. The fitness component F_i can be obtained by using a single uniform pseudo-random function *U*:

$$F_i(\boldsymbol{x}) = U(\boldsymbol{x}.\bullet \boldsymbol{r}_i, \boldsymbol{r}_i, i) \sim \text{uniform on}[0,1), \qquad \text{Eq. 6.2}$$

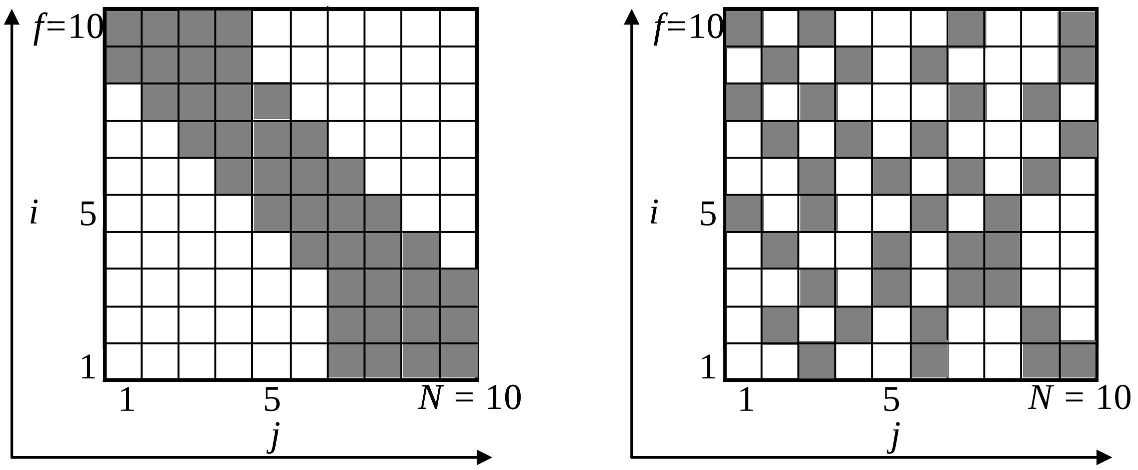
where $U : \{0,1\}^N \times \{0,1\}^N \times \{1, ..., N\} \to [0,1)$ and $\bullet$ is the Hadamard product which can be expressed as:

$$\boldsymbol{x}.\bullet \boldsymbol{r}_i = \begin{bmatrix} x_1 m_{i1} \\ x_2 m_{i2} \\ \cdots \\ x_N m_{iN} \end{bmatrix}. \qquad \text{Eq. 6.3}$$

In Eq.6.2, any change in one of the three arguments $\boldsymbol{x}.\bullet \boldsymbol{r}_i$, $\boldsymbol{r}_i$ and i will result in a randomly generated new value for $U(\boldsymbol{x}.\bullet \boldsymbol{r}_i, \boldsymbol{r}_i, i)$. If we store the values for all the possible allelic combinations (genotypes) ***x***, it requires $2^{(K+1)}$ spaces. In addition, based on the fact that one fitness component F_i is determined by one gene with its K epistasis, we can know the number of components f equals the number of loci in the genotypes N. Thus it requires storage of $2^{(K+1)}N$ in total to implement this function.

There are two methods of how to generate a gene fitness map: *adjacent neighborhoods* and *random neighborhoods*. The gene fitness map using either method requires the main diagonal be filled. When using the adjacent neighborhood method, the main diagonal would be surrounded by K filled adjacent diagonals in the gene fitness map. The random neighborhoods method, however, fills each row with K randomly selected units besides the diagonal unit that's been filled. These two methods and their corresponding gene fitness maps are illustrated in Figure 6.1(left) and Figure 6.1(right) respectively.

Figure 6.1 Two gene fitness maps: adjacent neighborhoods (left) and random neighborhoods (right) when N = 10, f = 10 and K = 3



6.2.1 Breeding Simulation in NK Fitness Landscape

The objective of a breeding simulation is to find the largest output of each of a variety of *NK* fitness landscape models, as shown in Eq. 6.1. The ruggedness of the fitness function is controlled by parameter *K*. Two different selection strategies, MAS and our proposed approach, are used in breeding simulation. Both simulations start with a population of 45 lines that are randomly generated, each of which is a combination of N = 20 alleles. By using different selection strategies for the crossover, a population of 45 new lines will be generated. New lines from crossover become the parent population for the next generation. The cycle above will repeat over multiple generations until convergence.

In MAS, we run a basic genetic algorithm to obtain an elite genotype which has the largest output value through the *NK* fitness landscape model. The top 10 lines are selected based on closeness (Manhattan distance) to the elite genotype. The crossover on all the combinations of pairs in the top 10 lines will yield 10*9/2 = 45 new lines as the new population in next generation.

In the proposed approach, genotypes are selected for crossover based on average predictions after crossover. For each pair of combinations from the 45 lines, simulate all possible genotypes after crossover. The average of these predicted output values in *NK* model is considered to be the expected 'fitness' of the corresponding pair after crossover.

We apply breeding simulations on two scenarios of *NK* landscape models: the adjacent neighborhoods method and the random neighborhoods method, depending on how the gene fitness interaction map was generated. In each scenario, computer simulations were conducted by varying the *NK* fitness landscape parameter *K*. Apparently, there is no epistasis (gene interaction) when K = 0. The NK models become increasingly complex with the increase of *K*. Both MAS and our proposed approach used one point crossover. Mutation, however, was not applied. Due to the *Monte Carlo* feature of the simulation, 20 independent runs were performed for each model under the landscape parameters, N = 20, while *K* varies at 0, 1, 3, 5, 7, and 9. The results were averaged over 20 runs.

Figure 6.2 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 0. The *NK* model is generated by using the adjacent neighborhood method.

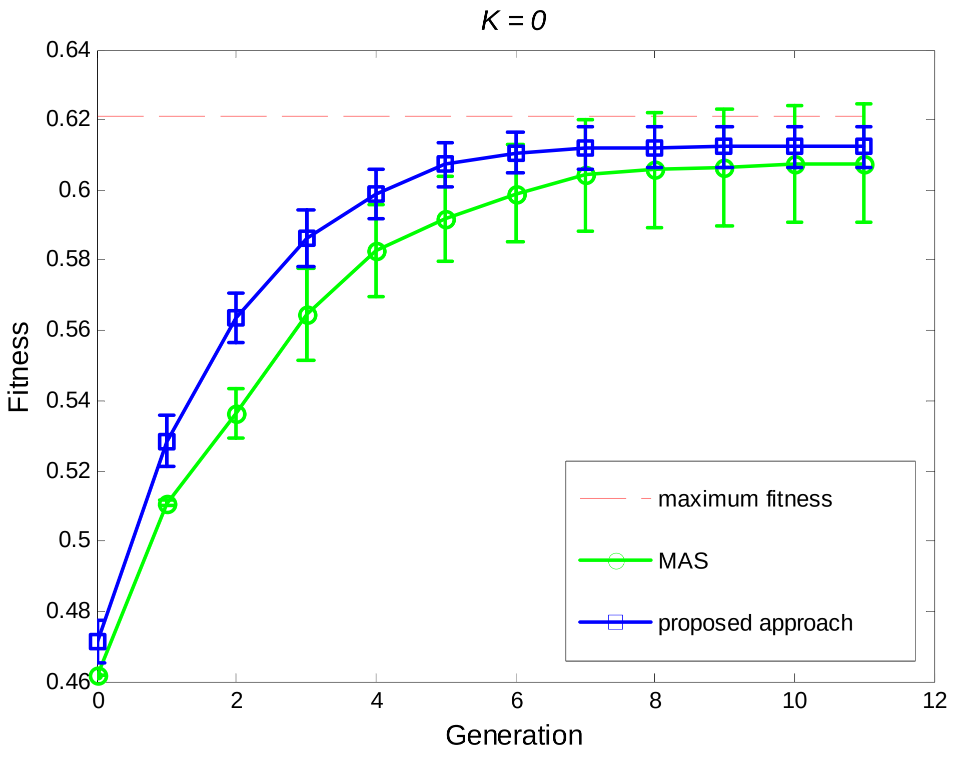


Figure 6.3 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 1. The *NK* model is generated by using the adjacent neighborhood method.

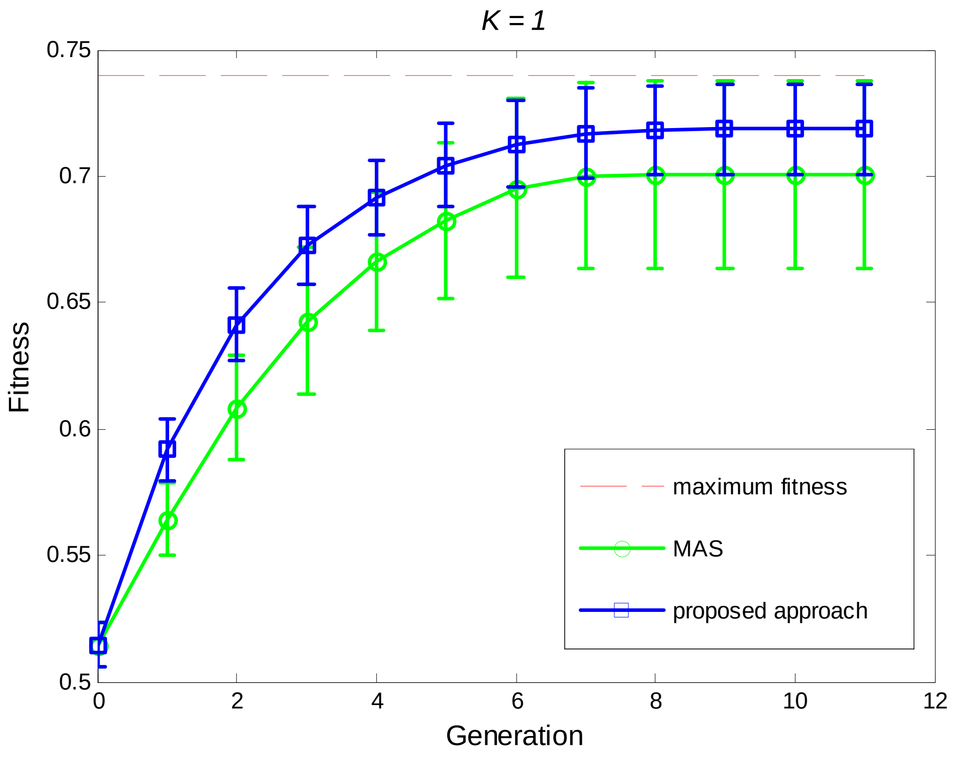


Figure 6.4 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 3. The *NK* model is generated by using the adjacent neighborhood method.

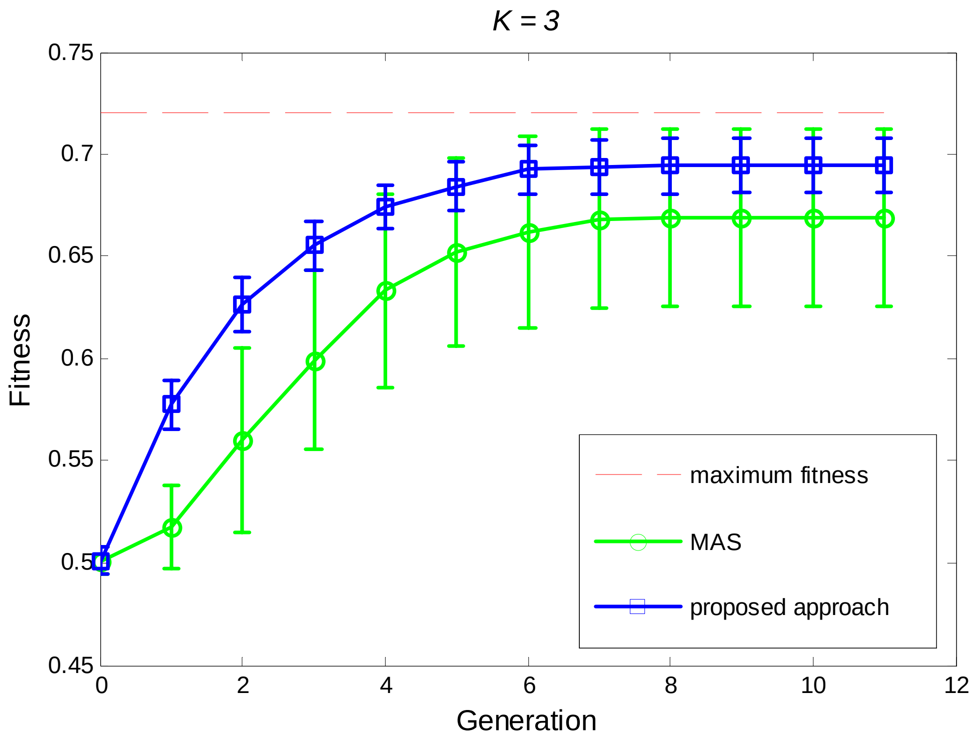


Figure 6.5 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 5. The *NK* model is generated by using the adjacent neighborhood method.

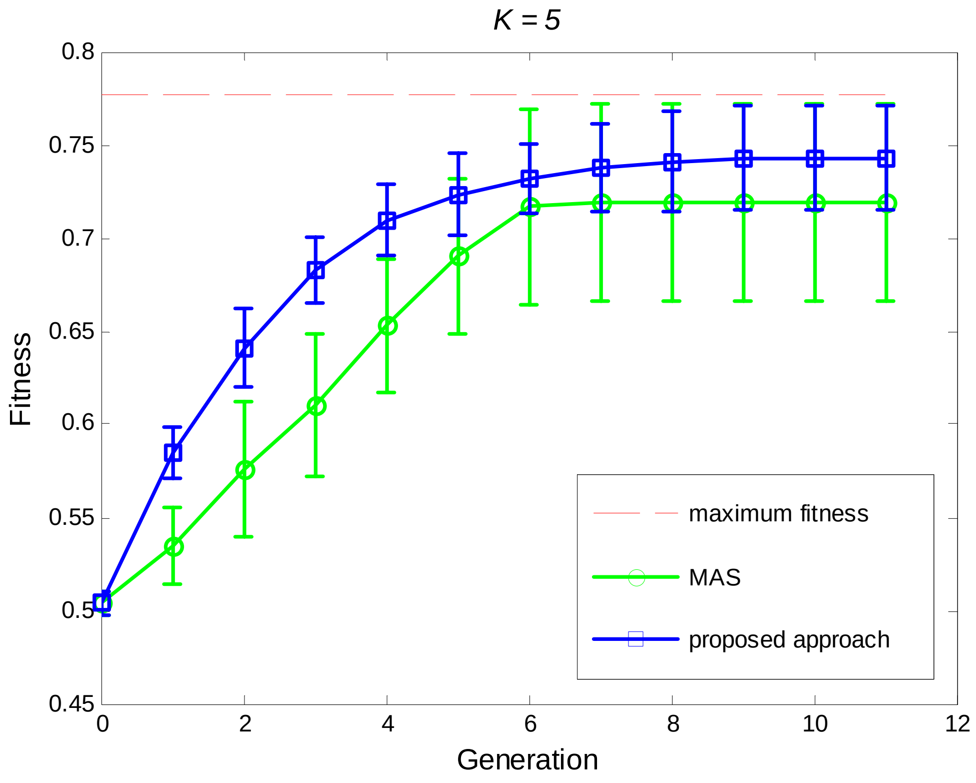


Figure 6.6 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 7. The *NK* model is generated by using the adjacent neighborhood method.

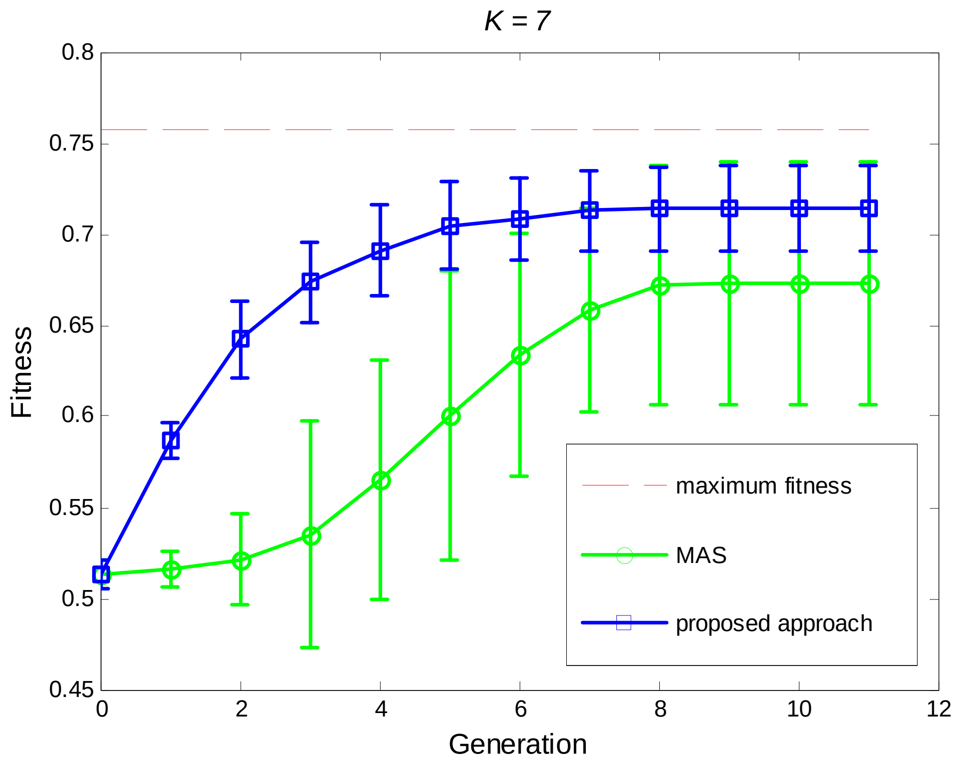


Figure 6.7 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 9. The *NK* model is generated by using the adjacent neighborhood method.

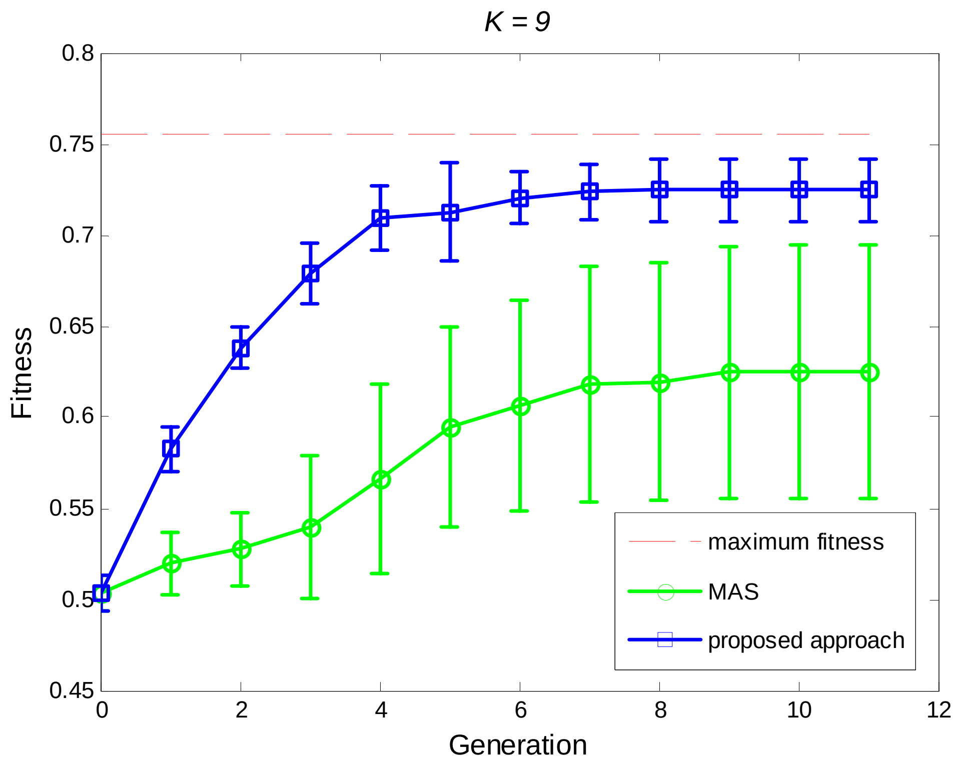


Table 6.1 Comparison of MAS and our proposed approach after convergence with multiple NK fitness landscape models generated by the adjacent neighborhoods method.

epistasis *K*		0	1	3	5	7	9
elite fitness		0.7181	0.7401	0.720	0.7767	0.7578	0.7550
MAS	mean	0.6075	0.7005	0.6687	0.7194	0.6732	0.6250
	std	0.0169	0.0373	0.0433	0.0529	0.0668	0.0696
Proposed	mean	0.6123	0.7185	0.6948	0.7429	0.7143	0.7245
	std	0.0060	0.0176	0.0133	0.0280	0.0233	0.0171

Comparisons of MAS and our proposed approach simulated on *NK* fitness models (the adjacent neighborhoods method) are shown in Figure 6.2 to 6.7. The straight dash line demonstrates the fitness value of the elite line obtained from the genetic algorithm for MAS. To demonstrate the statistical confidence over multiple runs, error bars, whose vertical distances denote $2*\sigma$ (standard deviation), are also shown in the figures. The final converged mean fitnesses as well as the standard deviations over multiple runs can be found in Table 6.1. Simulations on *NK* fitness landscape models (the random neighborhoods method) are shown in Figure 6.8 to 6.13, respectively. The corresponding final mean fitnesses and the standard deviations over multiple runs are demonstrated in Table 6.2.

As we can see in the figures and tables, our proposed approach achieves faster convergence, and is also able to obtain a higher mean fitness value at the end of each run. Another phenomenon we are particularly interested in is the smaller variations in simulations that applied our proposed approach, which potentially indicate better accuracy than MAS-based breeding. Along with the increase of the *NK* model complexity (increase of gene epistasis *K*), our proposed approach shows increasingly significant outperformance over MAS, as shown in Table 6.1 and 6.2, which indicates our proposed approach is able to capture the additional gene interactions in the models.

Figure 6.8 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 0. The *NK* model is generated by using the random neighborhoods method.

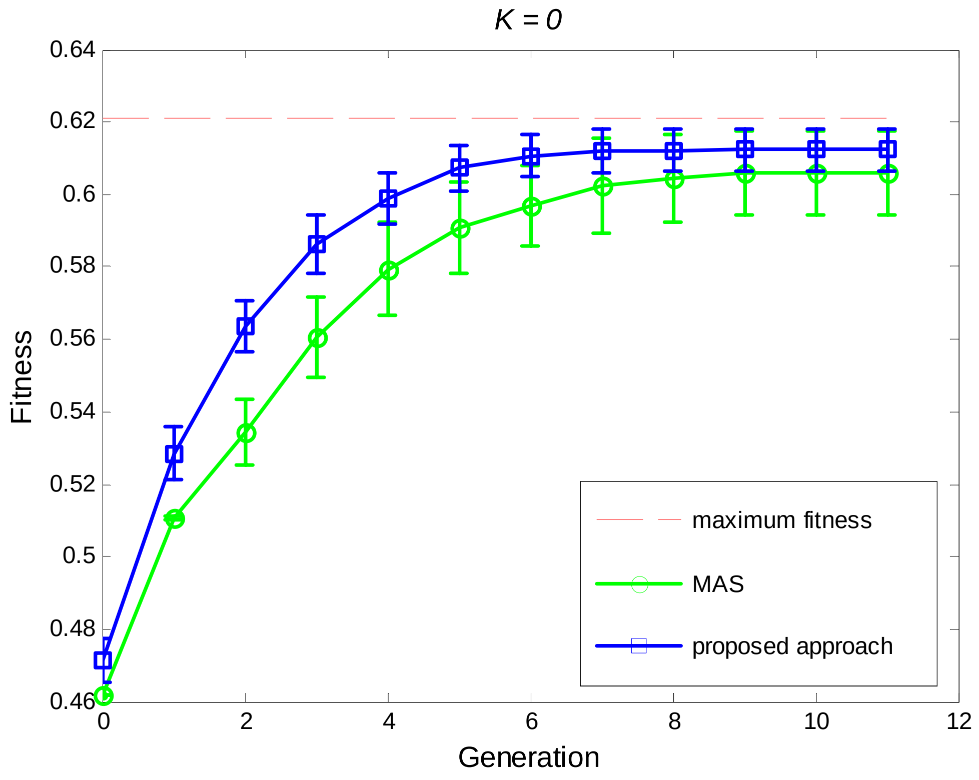


Figure 6.9 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 1. The *NK* model is generated by using the random neighborhoods method.

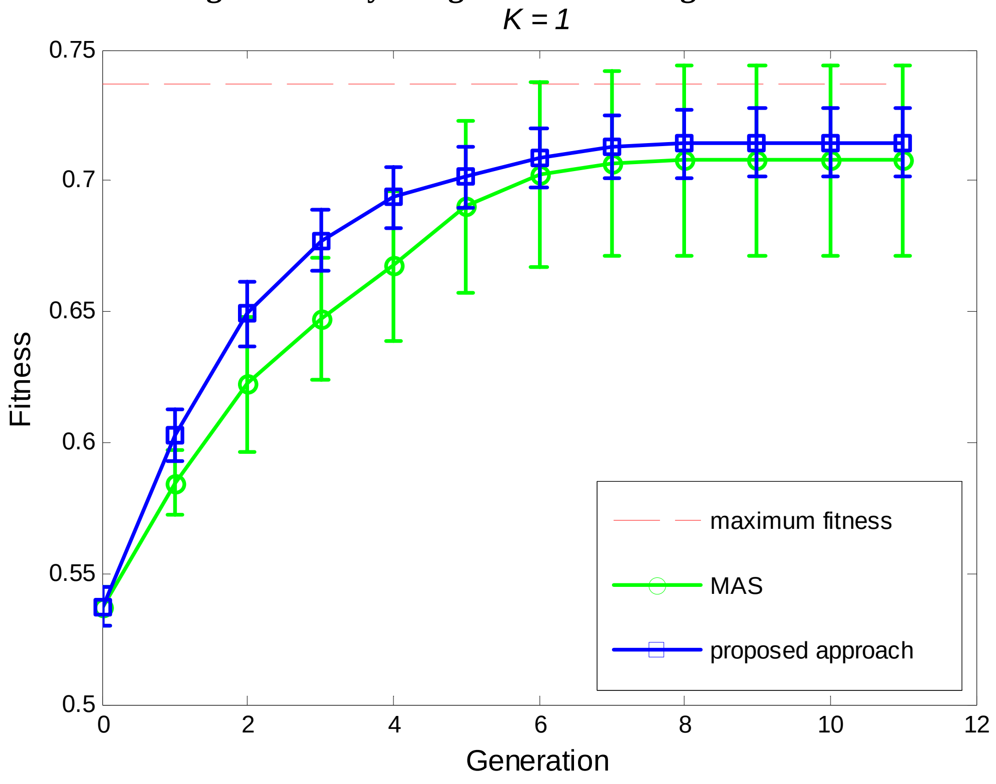


Figure 6.10 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 3. The *NK* model is generated by using the random neighborhoods method.

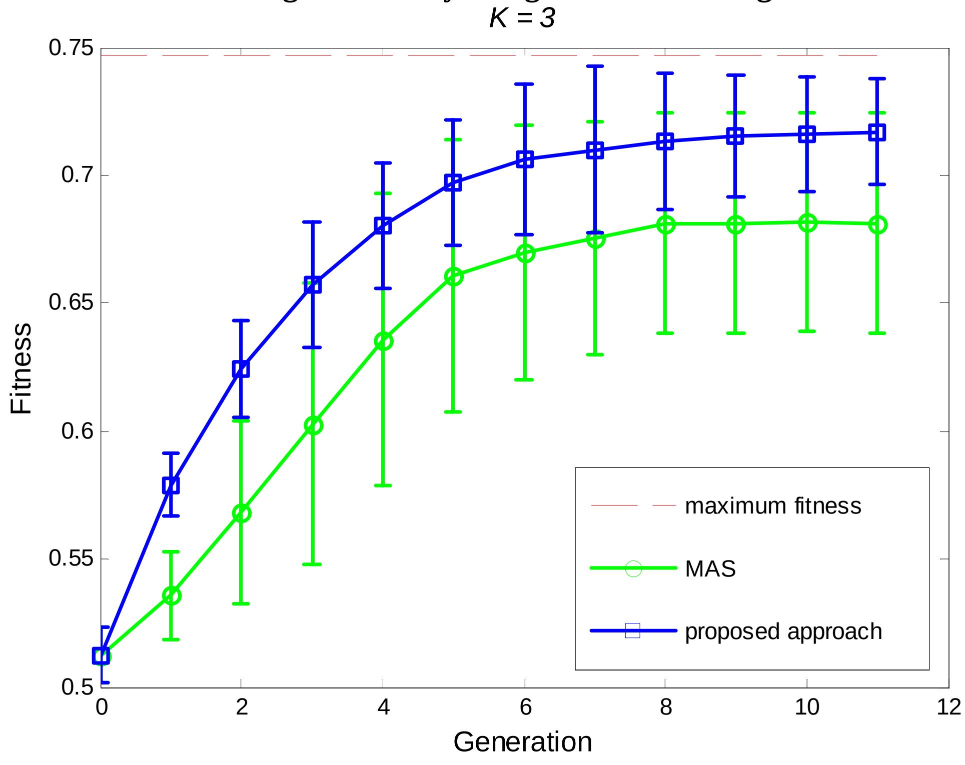


Figure 6.11 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 5. The *NK* model is generated by using the random neighborhoods method.

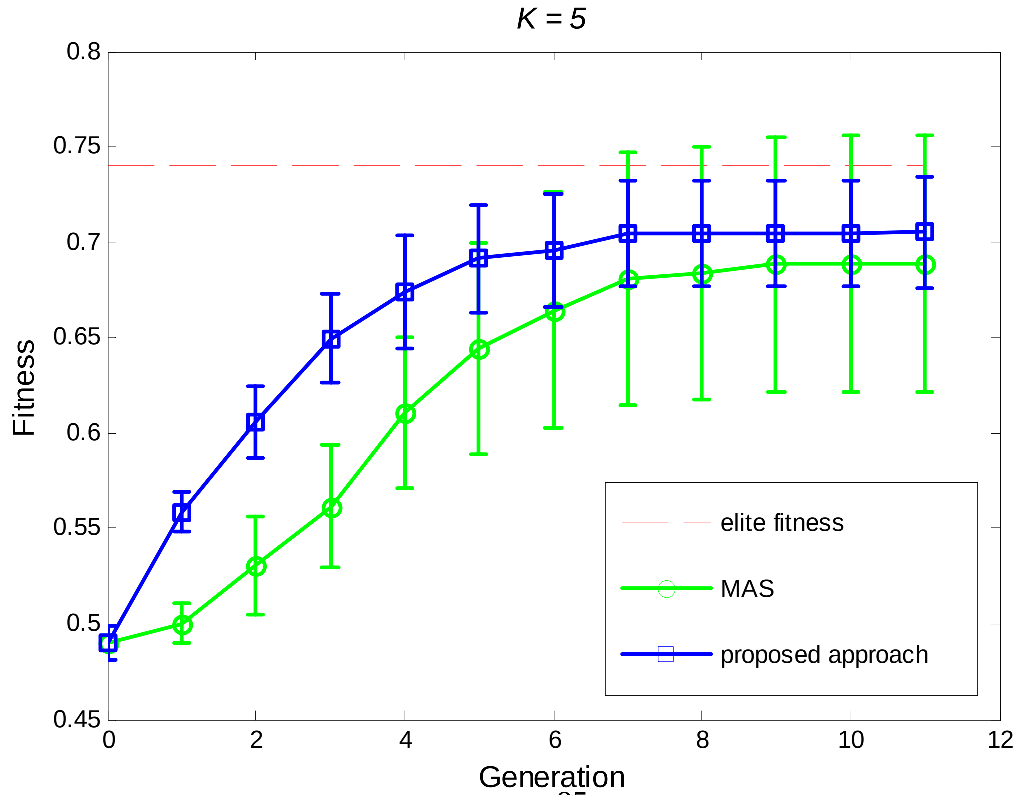


Figure 6.12 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 7. The *NK* model is generated by using the random neighborhoods method.

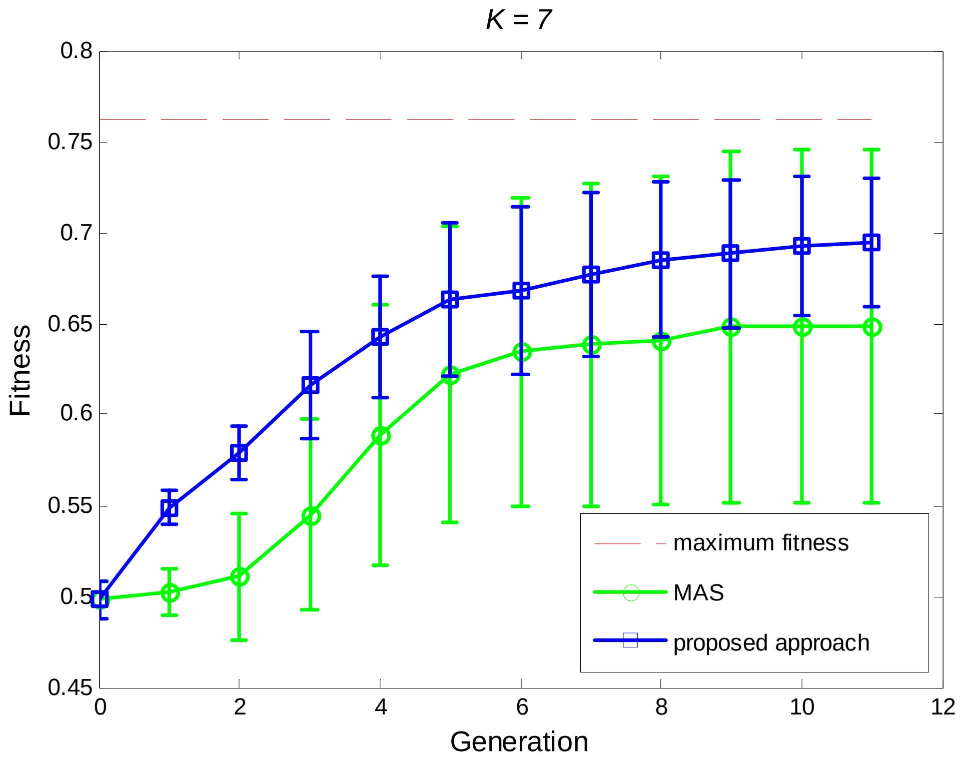


Figure 6.13 MAS vs. our proposed approach on *NK* model based breeding, where *N* = 20 and *K* = 9. The *NK* model is generated by using the random neighborhoods method.

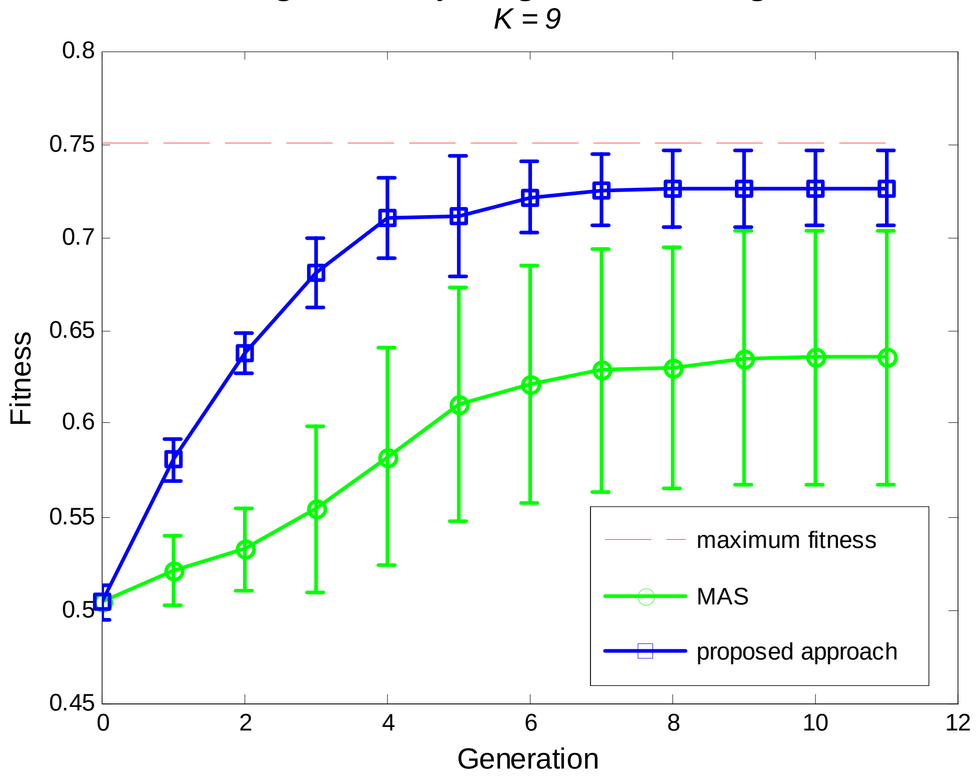


Table 6.2 Comparison of MAS and NAS after convergence in the random neighborhoods based on *NK* models

epistasis *K*		0	1	3	5	7	9
elite fitness		0.6212	0.7372	0.7471	0.7400	0.7618	0.7502
MAS	mean	0.6032	0.7077	0.6813	0.6867	0.6485	0.6356
	std	0.0195	0.0362	0.0433	0.0669	0.0676	0.0696
Proposed	mean	0.6123	0.7147	0.7170	0.7052	0.6948	0.7263
	std	0.0060	0.0133	0.0209	0.0288	0.0355	0.0203

6.3 Plant Breeding Simulations

In the last section, we proposed an approach that used prediction to guide the breeding process. The simulation results on *NK* fitness landscape models show that our proposed approach outperforms well-known MAS because of its ability to capture the gene interactions. Based on the concept of the proposed approach in the last section, we propose a *network assisted selection* (NAS) that makes use of networks obtained from GP-PSO hybrid algorithms to guide the breeding process for plant breeding simulations.

The simulation was set up as follows. The objective of the plant breeding simulations was to find the lines (genotypes) with the latest bolting dates. In each breeding simulation, lines were evaluated against 54 environments (3 planting sites * 18 planting dates). The average bolting dates after evaluation of lines through the synthetic network mimicked those in real world planting. Network obtained by our hybrid algorithm is used to make close prediction.

To maintain the consistency of our proof-of-concept simulation for NAS-based breeding on *NK* fitness models, the simulation was designed as close to the breeding simulation on *NK* models as possible. The plant breeding process was conducted as follows. Forty-five lines were randomly generated as the initial breeding population, each being a combination of 100 alleles. By using different selection strategies, a population of 45 new lines was generated after applying the crossover. The new lines generated by the crossover became the parent population for the next generation. The cycle above was repeated over multiple generations until the population reached convergence. Because the simulations were stochastic in nature, a total of 20 breeding runs were conducted for each selection strategy.

A randomly selected network obtained by the GP-PSO hybrid algorithm is used. The NAS was implemented as follows.

NAS: Genotypes are selected for crossover based on network assisted predictions. For each pair of combinations from 45 lines in the parent population, simulate all possible genotypes after crossover. Evaluate them through networks obtained from our hybrid algorithm. The average of these predicted bolting dates is considered to be the expected 'fitness' of the corresponding pair after crossover. Forty-five pairs with the best expected fitness are selected for crossover to generate new lines.

Two different selection strategies were selected to compare with NAS separately: *marker assisted elite selection* (MAES) and *marker assisted tournament selection* (MATS). They were conducted respectively as follows.

MAES: The top 10 lines are selected based on closeness (Manhattan distance) to the elite genotype. The elite genotype is obtained by a simple genetic algorithm. Crossover all the combinations of pairs, which yields 10*9/2 = 45 new lines.

MATS: Repeat the tournament selection using closeness to elite genotype as criteria to select 10 lines. Crossover all the combination of pairs, which yields 10*9/2 = 45 new lines.

Figure 6.14 shows the comparison of average bolting dates over ten generation for MAES and NAS. It can be seen in the figure that NAS converges faster than MAES, and NAS and MAES reach almost the same bolting dates after the convergence. Figure 6.15 shows the comparison of average bolting dates for MATS and NAS and we can draw a similar conclusion.

The gene regulatory network, as shown in Figure 5.1, is only controlled by 6 genes. Simulations on such a small network explain the fast convergence. We believe that the benefit of NAS will be greater if we apply it to a bigger model.

Figure 6.14 Comparison of MAES and NAS in Plant Breeding

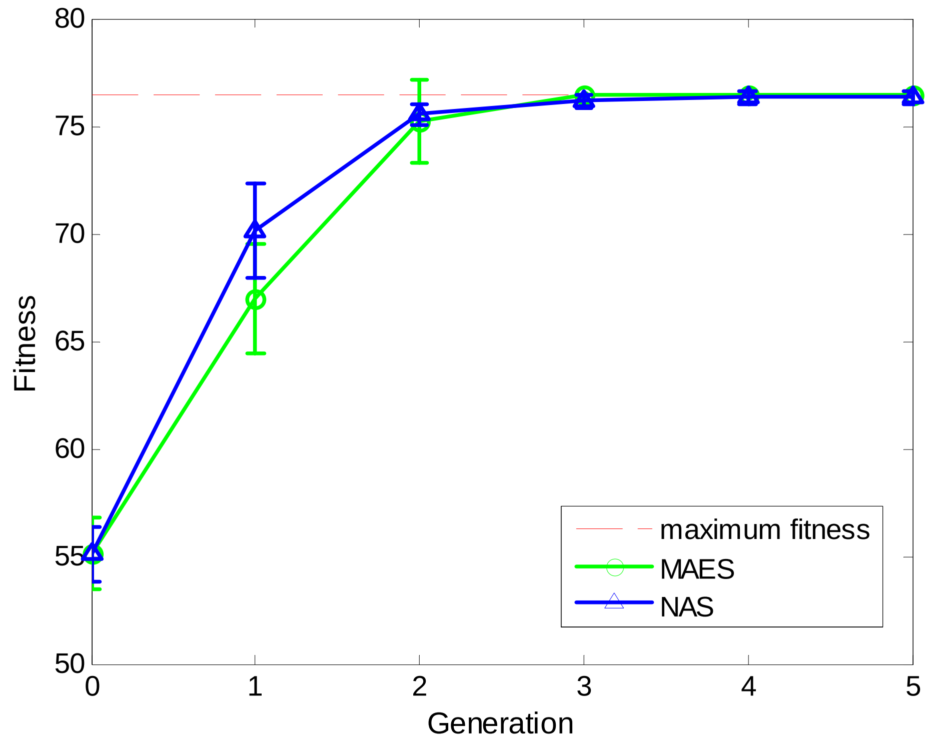
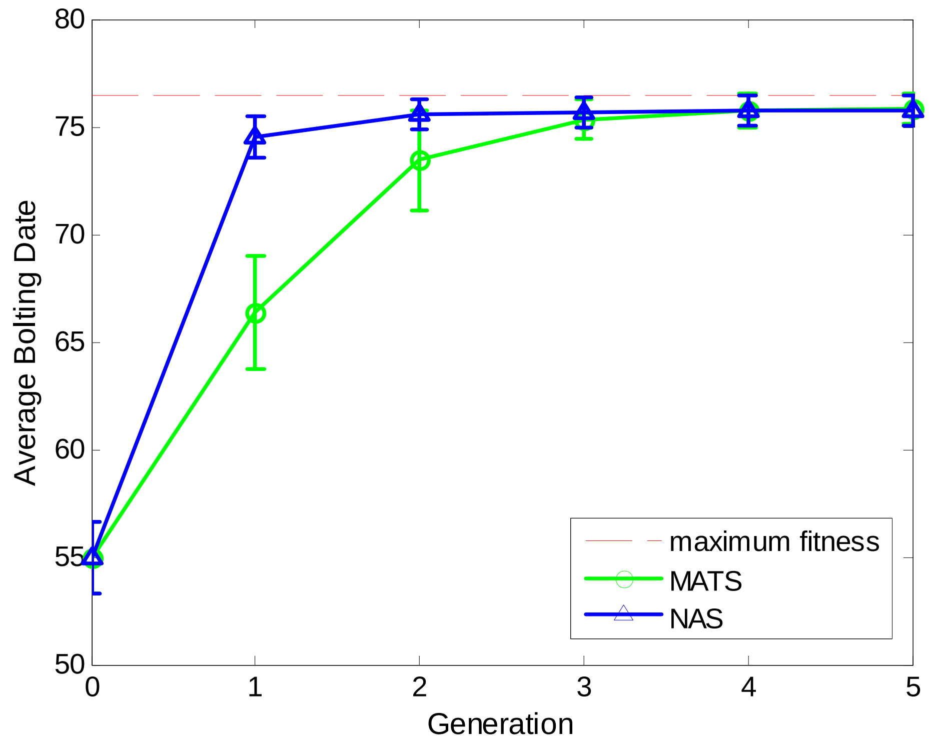


Figure 6.15 Comparison of MATS and NAS in Plant Breeding



CHAPTER 7 - Conclusion and Future Work

The huge amount of experimental data in molecular biology requires us to find an effective approach for gene regulatory modeling. A wide range of models have been used in gene modeling. However, almost all of them require building a model structure first and then optimizing associated parameters based on goodness-of-fit.

The proposed GP-PSO hybrid algorithm is an effective and novel approach that is able to infer network structure and optimize associated parameters simultaneously. CGP, a special form of GP, is used to recover the structures and PSO is selected for parameter estimations due to its well-known fast convergence. One of the greatest advantages of CGP over conventional GP is its lack of bloat. With CGP, solutions are not likely to contain huge but numerically meaningless components that exhaust computer resources.

Moreover, the multi-objective design of our algorithm enables us to take advantage of different types of data, in our case, both phenotype data and gene expression. The concept of dominance relationships is adopted in a multi-objective optimization that provides a selection pressure toward a Pareto front. Convergence and diversity are two critical criteria in multi-objective optimization. We address the former issue with non-dominated sorting and the latter with a histogram approach based on the concept of hypergrids.

The gene identification similar to QTL mapping was adopted as a preprocessing step to reduce the number of genes that are likely to exist in the synthetic gene network and consequent computational time. Based on the number of genes left after the gene identification, multiple runs were conducted. The non-dominated fronts at the end of our algorithm in results over multiple runs showed good convergence and diversity compared to initial population. Multiple networks and associated parameters in final non-dominated fronts over different runs were presented. The predicted bolting dates and gene expression level by these obtained solutions (networks and their associated parameters) were very close to the real data. Those solutions have also been converted into numerical equations and compared with the equation of the synthetic network. Our discovery was that although they look different at the first glance, the numerically significant parts of equations remain similar.

One application for the obtained networks is plant breeding. Thus we proposed network assisted selection which utilized the network predictions to guide the breeding process. First we applied this concept on different NK fitness landscape models where it proved to be effective from simulation. Simulation of NAS breeding shows it outperforms another advanced breeding strategy – MAS, in terms of both convergence rate and desired phenotype.

Further work may include using the real phenotypic data instead of data generated by the synthetic network. The scope of the current thesis is restricted to find small gene regulatory networks that could be as good as a synthetic network for some applications. The links between the number of objectives and solutions are still unknown. There has been very little research in analysis of obtained gene network structures and estimated parameters, in both parameter space and fitness space.

References

1. D. Corne, M. Dorigo and F. Glover, *New Ideas in Optimization*. McGrawHill, London, UK, 1999.

2. L. J. Fogel, *Artificial Intelligence through Simulated Evolution: Forty Years of Evolutionary Programming*, John Wiley & Sons, Inc., New York, 1999.

3. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

4. C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," In Hans-Michael Voigt *et al.*, editor, Parallel Prolem Solving from Nature – PPSN IV, *Lecture Notes in Computer Science*, pp. 584-593, Berlin, Germany, Springer-Verlag, Sep.1996.

5. E. Zitzler, K. Deb, and L. Thiele, "Comparison of multi-objective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, 2000.

6. E. Zitzler and L. Thiele, Multi-objective evolutionary algorithms: "A comparative case stuy and the strength Pareto approach," *Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, Nov. 1999.

7. S. Das, B. Panigrahi, "Multi-objective evolutionary algorithms," *Encyclopedia of Artificial Intelligence*, J. R. Rabual, J. Dorado and A. Pazos Eds, Idea Group Publishing, to be published.

8. K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evolutionary Computation*, vol. 7, no. 3, pp. 205-230, 1999.

9. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable multi-objective optimization test problems,” *Evolutionary Computation*, vol. 1, pp.825-830, Piscataway, New Jersey, 2002

10. K. Deb, L. Thiele, M. Laumanns and E. Zitzler, “Scalable test problems for evolutionary multi-objective optimization,” *Evolutionary Multi-objective Optimization: Theoretical Advances and Applications*, pp. 105-145, Springer, USA, 2005.

11. M. Ehrgott, *Multi-criteria Optimization*, Springer, second edition, 2005.

12. K. M. Miettinen, *Nonlinear Multi-objective Optimization*, Kluwer Academic Publisher, Boston, Massachusetts, 1999.

13. E. Zitzler, M. Laumanns, S. Bleuler, “A tutorial on evolutionary mulit-objective optimization,” *workshop on Multiple Objective Metaheuristics* (*MOMH2002*), Springer-Verlag, Berlin, Germany, 2004.

14. K. Deb, A. Pratab, S. Agrawal and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, Apr. 2000.

15. J. D. Knowles and D. W. Corne, “Approximating the nondominated front using the pareto archived evolution strategy”, *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 149-172, 2000.

16. B. W. Silverman, *Density estimation for statistics and data analysis*, Chapman and Hall, London, 1986.

17. C. M. Fonseca and P. J. Fleming, “Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization,” In S. Forrest, editor, *proceedings of the*

Fifth International Conference on Genetic Algorithms, pp. 416-423, San Mateo, California, 1993.

18. E. Zitzler, M. Laumanns and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algogrithm for multi-objective optimization,” In K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimization, and Control*, pp. 19-26, Barcelona, Spain, 2002.

19. J. Horn, N. Nafpliotis, and D. E. Goldberg, “A niched pareto genetic algorithm for mulitobjective optimization,” In *proceedings of the First IEEE Conference on Evolutionary Computation*, vol.1, pp. 82-87, Piscataway, NJ, 1996.

20. N.Srinivas and K. Deb, “Multi-objective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, no. 3, pp. 221-248, 1994.

21. C. A. Coello, G. T. Pulido and M. S. Lechuga, “Handing multi-objectives with particle swarm optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256-279, Jun. 2004.

22. C. A. Coello and N. C. Cortes, “Solving multi-objective optimization problems using an artificial immune system,” *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163-190, 2005.

23. http:// en.wikipedia.org/wiki/Genetic_algorithm

24. M. Mitchell, *An Introduction to Genetic Algorithm*. MIT Press, 1998.

25. J. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

26. L. Altenberg, “Emergent phenomena in genetic programming,” In A. V. Sebald and L. J. Fogel (eds.), *Evolutionary Programming: Proceeding s of the Third Annual Conference*, pp. 233-241, World Scientific Publishing, 1994.

27. P. J. Angeline, “Subtree crossover causes bloat,” In J. R. Koza, *et al*.(eds.), Genetic Programming 1998: *Proceedings of the Third Annual Conference*, 745-752, Morgan Kaufmann, 1998.

28. W. B. Langdon, and R. Poli, “Fitness causes bloat: mutation,” In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty(eds), *EuroGP98: First European workshop on Genetic Programming*, pp. 37-28, Springer-verlag, 1998.

29. P. Nordin, and W. Banzhaf, “Complexity compression and evolution,” In L. Eshelman(ed.), *Genetic Algorithms: Proceedings of the Sixth International Conference*, pp. 310-317, Morgan Kaufmann, 1995.

30. T. Soule, J. A. Foster, and J. Dickinson, “Code growth in genetic programming,” In J. R. Koza, *et al*. (eds.*), Genetic Programming 1996: Proceedings of the First Annual Conference*, pp.215-223, MIT Press, 1996.

31. T. Soule, *Code Growth in Genetic Programming*, PhD thesis, University of Idaho, 1998.

32. J. F. Miller and P. Thomson, “Cartesian genetic programming,” in *Proceedings of 3rd European Conference on Genetic Programmin*g, vol. 1802, R. Poli *et al*., Eds., New York, 1999, pp.17-30, *Lecture Notes in Computer Science*.

33. M. A. Lones, “Enzyme genetic programming,” PhD dissertation, Dept. Electronic Engineering, University of York, North Yorkshire, UK, 2003.

34. Xinye Cai, Stephen L. Smith, and Andy M. Tyrrell, "Benefits of employing an implicit context representation on hardware geometry of CGP”, in *Proceeding of 6th*

International Conference, Evolvable Systems : From Biology to Hardware, , Sitges, Barcelona, Spain , September 12-14, 2005, pp 143-154.

35. Xinye Cai, Stephen L. Smith, and Andy M. Tyrrell, "Positional independence and recombination in cartesian genetic programming", in *Proceedings of the 9th European Genetic Programming Conference*, *EuroGP 2006*, Budapest, Hungar, April 10-12, 2006, pp 351-360.

36. J. Miller, "What bloat? Cartesian genetic programming on Boolean problems," in *Proceeding of Genetic Evolutionary Computation Conference*: Late Breaking Papers, E. D. Goodman, Ed., 2001, pp. 295-302.

37. J. Miller and S. L. Smith, "Redundancy and computational efficiency in Cartesian genetic programming", *IEEE Transaction on Evolutionary Computation*, April, 2006, vol. 2, no. 2, pp. 167 -174.

38. J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the evolutionary design of digital circuits—Part I," *Genetic Programming Evolvable Machines*, vol. 1, pp. 8-35, 2000.

39. J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the evolutionary design of digital circuits—Part II," *Genetic Programming Evolvable Machines*, vol. 1, pp. 259-288, 2000.

40. J. F. Miller, "Evolution of digital filters using a gate array model," in Proceeding of *1st European Workshops Evolutionary Image Analysis, Signal Processing Telecommunication*, vol. 1596, R. Poli *et al*., Eds., New York, pp.17-30, 1999.

41. L. Sekanina, *Evolvable Components: From Theory to Hardware Implementation*. New York, Springer Verlag, 2004.

42. J. A. Rothermich and J. F. Miller, "Studying the emergence of multicellularity withcartesian genetic programming in artificial life," in *Proceedings of Genetic*

Evolutionary Computation Conference: Late Breaking Papers, E. Cantu-Paz, Ed., pp.397-403, 2002.

43. J. F. Miller and P. Thomson, "A developmental method for growing graphs and circuits," in Evolvable Systems: From Biology to Hardware, in *Proceedings of 5th International Conference*, *ICES 2003*, vol. 2606, A. M. Tyrrell *et al*., Eds., Proceedings published as Lecture Notes in Computer Science, New York, pp. 93-104, 2003.

44. J. F. Miller and W. Banzhaf, "Evolving the program for a cell: From French flags to Boolean circuits," in *On Growth, Form and Computers,* S. J. Kumar and P. J. Bentley, Eds. pp. 278-301, Academic, New York, 2003.

45. J. F. Miller, Evolving Developmental Programs for Adaptation, Morphogenesis, and Self-repair, *7th European Conference on Artificial Life,* Dortmund, September 14-17, 2003, Proceedings published as Lecture Notes in *Artificial Life*, vol. 2801, pp. 256-265, 2003.

46. A. B. Garmendia-Doval, D. S. Morley, and S. Juhos, "Post docking filtering using Cartesian genetic programming," in Proceedings of 6th International Conference on *Artificial Evolution,* P. Liardet, Ed., pp. 435-446, 2003.

47. A. B. Garmendia-Doval and J. F. Miller, "Cartesian genetic programming and the post docking filtering problem," in *Genetic Programming Theory Practice II,* U. M. O'Reilly, Ed., 2004.

48. J. Rothermich, F. Wang, an J. F. Miller, "Adaptivity in cell based optimization for information ecosystems," in Proceedings Congress *Evolutionary Computation*, Piscataway, NJ, pp. 490-497,2003.

49. M. S. Voss, "Social programming using functional swarm optimization," in Proceedings of *IEEE Swarm Intelligence Symposium*, 2003.

50. M. Clerc, *Particle Swarm Optimization*. ISTE Press, UK, 2005.

51. J. Kennedy and R.C. Eberhart, R. C., “Particle swarm optimization,” *Proceeding of IEEE International Conference on Neural Networks,* IV, 1942--1948. Piscataway, NJ: IEEE Service Center.

52. J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

53. S. Bleuler, M. Brack, L. Thiele, and E. Zitzler, “Multi-objective genetic programming: Reducing bloat by using SPEA2,” *in Proc. Congress Evolutionary Computation*, 2001, pp. 536-543.

54. E. deJong, R. A. Watson, and J. B. Pollack, “Reducing bloat and promoting diversity using multi-objective methods,” *in Proc. of GECCO’01 2001*, pp.11-18.

55. K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming, “Identifying the structure of nonlinear dynamic systems using multi-objective genetic programming,” *IEEE Transactions on Systems, Man and Cybernetics: Part A. Systems and Humans*, vol. 34, no. 4, pp. 531-545, 2004.

56. R. Curry and M. Heywood, “One-class learning with multi-objective genetic programming,” *IEEE International Conference on System, Man and Cybernetics*, Oct. 2007, pp. 1938-1945.

57. M. G. Schuster, “A multi-objective genetic programming approach for pricing and hedging derivative securities,” *IEEE International Conference on Computational Intelligence for Financial Engineering*, March, 2003, pp. 77-84.

58. J. Moore and R. Chapman, *Application of Particle Swarm to Multiobjective Optimization*: Department of Computer Science and Software Engineering, Auburn University, 1999.

59. T. Ray, K. M. Liew, “A swarm metaphor for multiobjective design optimization,” *Engineering optimization*, vol. 34, no. 2, pp. 141-153, Mar.2002.

60. K. E. Parsopoulos and M. N. Vrahatis, “ Particle Swarm optimization method in multiobjective problems,” in *proc. 2002 ACM Symp. Applied Computing* (*SAC’2002*), Mardrid, Spain, 2002, pp.603-607.

61. X. Hu and R. C. Eberhart, “Multi-objective optimization using dynamic neighborhood particle swarm optimization,” in *Proc. Congr. Evolutionary Computation* (*CEC’2002*), vol. 2, Honolulu, HI, May 2002, pp.1677-1681.

62. C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutioanry Algorithms for Solving Multi-objective Problems*. Norwell, MA: Kluwer, 2002, ISBN 0-3064-6762-3.

63. J. E. Fieldsend and S. Singh, “A multi-objective algorithm based upon particle swarm optimization, and efficient data structure and turbulence,” *Proc. 2002 U.K Workshop on Computational Intelligence*, Birmingham, U.K., Sept. 2002, pp.37-44.

64. X. Li, “An nondomonated sorting particle swarm optimizer for mltiobjective optimization,” in *Lecture Notes in Computer Science*, vol. 2723, Proc. *Genetic and Evolutionary Computation – GECCO 2003 – Part I*, E. Cantu-Paz *et al*., Eds. Berlin, Germany, July 2003, pp.37-48.

65. C. A. Coello Coello and G. T. Pulido, “Multi-objective optimization using a micro-genetic algorithm,” in *Proc. Genetic and Evolutionary Computation Conf.* (*GECCO2001*), L. Spector, E. D. Goodman, A. Wu, W. B. Landon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. PEZESHK, M. H. Garzon, and E. Burke, Eds., San Francisco, CA, 2001, pp.274-282.

66. Shubham Agrawal, Yogeesh Dashora, Manoj Kumar Tiwari, and Young-Jun Son,, “Interactive particle swarm: a Pareto-adaptive Metaheuristic to multi-objective optimization,” in *IEEE Transaction on System, Man and Cybernetics – Part A: System and Human*, vol. 38, no. 2, pp. 258 – 277. March, 2008.

67. P. K. Tripathi, Sanghamitra Bandyopadhyay and S. K. Pal, “Adaptive multi-objective particle swarm optimization algorithm,” *2007 IEEE Congress on Evolutionary Computation (CEC2007’)*, Singapore, pp. 2281 – 2288, 2007.

68. Karin Zielinski and Rainer Laur, “Adaptive parameter setting for a multi-objective particle swarm optimization algorithm,” *2007 IEEE Congress on Evolutionary Computation (CEC2007’)*, Singapore, pp. 3019 -3026, 2007.

69. J. Hanks and J. T. Ritchie, “Modeling plant and soil systems”, Agronomy Monograph no. 31, pp.545, ASA, CSSA, and SSSA, Madison, WI,1991.

70. S. M. Welch, J. L. Roe and Z. Dong, “A genetic neural network model of flowering time control in Arabidopsis thaliana,” *Agronomy Journal*, vol. 95, 71-81, 2003.

71. M. Cooper, S. C. Chapman, D. W. Podlich and G. L. Hammer, “The GP problem: quantifying gene-to-phenotype relationships,” Silico Biology, vol. 2, 151-164, 2002.

72. N. Soranzo, G. Bianconi and C. Altafini, “Comparing association network algorithms for reverse engineering of large scale gene regulatory networks: synthetic vs real data”, *Bioinformatics*, vol. 00, no. 00, pp. 1-7, 2007.

73. Anthony J. F. Griffiths, Richard C. Lewontin, Willian M. Gelbart and Jeffrey H. Miller, *Modern Genetic Analysis: Integrating Genes and Genomes*, W. H. Freeman, February, 2002.

74. S. M. Welch, J. L. Roe, S. Das, Z. Dong, R. He and M. B. Kirkham, “Merging genomic control networks and Soil-Plant-Atmosphere-Contiuum (SPAC) models,” *Agricultural Systems*, vol. 86, 243-274, 2004.

75. S. M. Welch, Z. Dong, J. L. Roe and S. Das, “Flowering time control: gene network modeling and the link to quantitative genetics,” *Australian Journal of Agricultural Research*, vol. 56, 919-936, 2005.

76. M. A. Blazquez, *Flower development pathway*, J. Cell Sci. vol.113, pp. 3547-3548.

77. P. Koduru, Z. Dong, S. Das, S. M. Wlech, J. Roe and E. Charibit “Multi-Objective Evolutionary-Simplex Hybrid Approach for the Optimization of Differential Equation Models of Gene Networks", *IEEE Transactions on Evolutionary Computation,* accepted.

78. M. Koornneef, C. Alonso-Blanco, A. J. M. Peeters, W. Scoppe, “*Genetic control of flowering time in Arabidopsis*”, *Ann. Rev. Plant Physil. Plant Mol. Biol*. 49, 345-370.

79. J. M. Martinez-Zapater, G. Coupland, C. Dean and M. Koornneef, “The transition to flowering in Arabidopsis,” In *Arabidopsis*, Cold Spring Harbor Lab Press, Cold Spring Harbor, New York, pp.403-433.

80. S. A. Kauffman, *The origins of order: self-organization and selection in evolution,* Oxford University Press, 1993.

81. S. Huang, “Gene expression profiling, genetic networks and cellular states: an integrating concept for tumorigenesis and drug discovery,” *Journal of Molecular Medicine*, vol. 77, pp. 469-480, 1999.

82. Y. Maki, D. Tominaga, M. Okamoto, S. Watanabe, Y. Eguchi, "Development of a system for inference of large scale genetic networks," *Proceedings of the Pacific Symposium on Biocomputing*, World Publishing Co., vol. 6, pp. 446-485, 2001.

83. M. Sugita, "Functional analysis of chemical system in vivo using a logical circuit equivalent II: The idea of a molecular automaton," *Journal of Theoretical Biology*, vol. 4, pp. 179, 1963.

84. L. A. Segel, Modeling dynamic phenomena in molecular and cellular biology. Cambridge University Press, 1984.

85. P. Baldi and G. W. Hatfield, *DNA microarrays and gene expression*, Cambridge University Press, 2002.

86. M. B. Elowitz, A.J. Levine, E. D. Siggia and P. S. Swain, "Stochastic gene expression in a single cell", *Science*, 297: 1183-1186.

87. W. J. Blake, M. Kaern and J. J. Collins, "Noise in eukaryotic gene expression," *Nature*, 422:633-637, 2003.

88. A. S. Ribeiro, R. zhu and S. A. Kauffman, "A general modeling strategy for gene regulatory networks with stochastic dynamics," *Journal of Computational Biology*, vol. 13, no. 9, pp. 274-284, 2006.

89. W. Coffey, Y. P. Kalmykov and J. T. Waldron, *the Langevin Equation*, World Scientific Publishing Company, 2004.

90. P. D'Haeseleer, X. Wen, S. Fuhrman and R. Somogyi, "Linear modeling of mRNA expression levels during CNS development and injury," *Pacific Symposium Biocomputing*, vol. 4, pp. 41-52, 1999.

91. T. S. Gardner, D. di Bernardo, D. Lorenz and J. J. Collins, “Inferring genetic networks and identifying compound mode of action via expression profiling,” *Science*, vol. 301, pp. 102-105, 2003.

92. R. Guthke, U. Moller, M. Hoffmann, F. Thies and S. Topfer, “Dynamic network reconstruction from gene expression data applied to immune response during bacterial infection,” *Bioinformatics*, vol. 21, pp. 1626-1634, 2005.

93. N. Friedman, M. Linial, I. Nachman and D. Peer, “Using Bayesian network to analyze expression data,” *Computational Biology*, vol. 7, pp. 601-620, 2000.

94. J. Huang, H. Shimizu and S. Shioya, “Clustering gene expression pattern and extracting relationship in gene network based on artificial neural networks,” *J. Biosci. Bioeng.*, vol. 96, pp. 421-428, 2003

95. Z. H. Chan, N. Karabov and L. J. Collins, “a two stage methodology for gene regulatory network extraction from time-course gene expression data,” *Expert System Application*, vol. 30, pp. 59-63, 2006.

96. S. Ando, E. Sakamoto and H. Iba, “Evolutionary modeling and inference of gene network,” *Information Science*, vol.145, pp. 237-259, 2002.

97. E. Sakamoto and H. Iba, “Identifying gene regulatory network as differential equation by genetic programming,” in *Genome Informatics*, vol. 11, pp 281-283, 2000.

98. R. Xui, X. Hu and D. C. Wunsh II, “Inference of genetic regulatory networks with recurrent neural network models,” in *Proceeding s of 26th Annual International Conference of the IEEE EMBS*, San Francisco, CA, U.S, September, 2004.

99. H. W. Ressom, Y. Zhang, J. Xuan, Y. Wang and R. Clarke, “Inference of gene regulatory networks from time course gene expression data using neural networks and swarm

intelligence," *Computational Intelligence and Bioinformatics and Computational Biology, CIBCB'06*, pp. 1-8, 2006.

100. J. Herz, "Statistical issues in reverse engineering of genetic networks," in *Proceedings of the Pacific Symposium on Biocomputing*, 1998.

101. C. Pridgeon and D. Corne, "Genetic network reverse engineering and network size: can we identify large GRNs?" In *Proceedings of the Computational Intelligence in Bioinformatics and Computaitonal Biology (CIBCB'2004)*, pp. 32-36, 2004.

102. I. Ono, R. Yoshiaki Seike, N. Ono and M. Matsui, "An evolutionary algorithm taking account of mutual interactions among substances for inference of genetic networks," In *proceedings of the IEEE Congress on Evolutionary Computation (CEC' 2004)*, pp. 2060-2067, 2004.

103. C. Spieth, F. Streichert, N. Speer and A. Zell, "Multi-objective model optimization for inferring gene regulatory networks," *Lecture Notes in Evolutionary Multi-Criterion Optimization*, vol. 3410, pp. 607-620, 2005.

104. S. H. Howell, *Molecular genetics of plant development*, Cambridge University Press, Cambridge, UK, 1998.

105. G. Hammer, M. Cooper, F. Tardieu, S. Welch, B. Walsh, F. van Eeuwijk, S. Chapman, and D. Podlich, "Models for navigating biological complexity in breeding improved crop plants", *Trends in Plant Science*, vol. 11, pp. 587-593, 2006.

106. C. Weinig, M. T. Brock, J. A. Dechaine and S. M. Welch, 'Resolving the genetic basis of invasiveness and predicting invasions', *Genetica*, vol. 129, no. 2, pp. 205-216, 2007.

107. I. Ausın, *et al*., "Environmental regulation of flowering", *International Journal of Developmental Biology*, vol. 49, no. 5/6, pp.689–705, 2005.

108. S. M. Welch, J. L. Roe, S. Das, Z. Dong, R. He and M. B. Kirkham, “Merging genomic control networks and Soil-Plant-Atmosphere-Contiuum (SPAC) models”, *Agricultural Systems*, vol. 86, pp.243-274, 2005.

109. G. Seaton, C. S. Haley, S. A. Knott, M. Kearsey and P. M. Visscher, “QTL Express: mapping quantitative trait loci in simple and complex pedigrees”, *Bioinformatics* vol. 18, pp. 339-340, 2002.

110. S. M. Welch, Z. Dong, J. L. Roe and D. Das, “Flowering time control: gene network modeling and the link to quantitative genetics”, *Australian Journal of Agricultural Research*, vol. 56, pp. 919-936, 2005.

111. C. S. Haley and S. A. Knott, “A simple regression method for mapping quantitative trait loci in line crosses using flanking markers”, *Heredity*, vol. 69, pp. 315-324, 1992.

112. J. M. Ribaut and D. A. Hoisington, Marker assisted selection: new tools and strategies. *Trends Plant Science*, vol. 3, pp 236-239, 1998.

113. H. S. Chawla, *Introduction to plant biotechnology*, Science Publisher Inc, Enfield, U.S.A.

114. S. A. Kauffman and S. Levin, “Towards a general theory of adaptive walks on rugged landscape,” *Journal of Theoretical Biology,* vol.128, pp. 11-45, 1987.

115. S. A. Kauffman, “Adaptation on rugged fitness landscapes,” In D. Stein, editor, Lectures in *the Sciences of Complexity,* pp. 527-618. Addison-Wesley, Redwood City. SFI Studies in the Sciences of Complexity, Lecture Volume I.

Appendix A - Terms and Definitions

Allele: One of the different forms of a gene that can exist at a single locus.

Amino acids: The basic building block of proteins.

Arabidopsis thaliana: A small flowering plant with a relatively short life cycle, which makes it popular as a model organism in plant biology and genetics.

Aggregation-based, criterion-based and *Pareto-based*: Three major techniques used in multi-objective optimization to achieve good convergence when forming the non-dominated front.

Bloat: The phenomenon that solutions have the tendency to become larger and exhaust computational resources in genetic programming that uses a tree structure as representation.

Chromosome: Originally indicating an organized structure of DNA and protein that is found in cells, its borrowed by evolutionary algorithms to indicate a representative solution.

Convergence and diversity: Two important metrics in a stochastic multi-objective algorithm design.

Criterion-based: See *Aggregation-based*.

Crossover rate: The probability that crossover operator is applied on a chromosome in evolutionary algorithms.

Crowded comparison: See *Fitness Sharing*.

Curse of dimensionality: The problem caused by the exponential increase in volume associated with adding extra dimensions to a (mathematical) space.

Domination counting and *non-dominated sorting*: Two Pareto-based ranking techniques to achieve good convergence when forming the non-dominated front.

Deoxyribonucleic acid (*DNA*): A double chain of linked nucleotides; the fundamental substance of which genes are composed.

*Elitism***:** A strategy in evolutionary algorithms where the best one or more solutions, called the elites, in each generation, are inserted into the next, without undergoing any change. This strategy usually speeds up the convergence of the algorithm. In a multi-objective framework, any non-dominated solution can be considered to be elite.

Evolutionary algorithms: A type of stochastic algorithms that's inspired from Darwin's evolutionary theory. These techniques include genetic algorithms, genetic programming, evolutionary programming and evolutionary strategy.

Exploration and *exploitation*: Terms used in a search algorithm to indicate in-breadth search and in-depth search respectively.

Fitness: A measure that is used to determine the goodness of a solution for an optimization problem.

Fitness landscape: A representation of the search space of an optimization problem that brings out the differences in the fitness of the solutions, such that those with good fitness are "higher". Optimal solutions are the maxima of the fitness landscape.

Fitness sharing, crowded comparison, histogram and *nearest neighbor*: Four techniques used in multi-objective optimization to achieve good diversity, see Section 1.3.3

Functions and terminals: Two basic elements in genetic programming that uses tree structure for representation. Terminals indicate the terminal nodes and functions indicate non-terminal nodes in such a tree.

Gene: The fundamental physical and functional unit of heredity, which carries information from one generation to the next; a segment of DNA composed of a transcribed region and a regulatory sequence that makes transcription possible.

Generation: A term used in evolutionary algorithms that roughly corresponds to each iteration of the outermost loop.

Genome: The entire complement of genetic material in a chromosome set.

Genotype: The specific allelic composition of a cell.

Genotype to phenotype mapping: A term used in biology indicating the problem of mapping an organism's allelic combination (genotype) to its physical traits (phenotype).

Genetic regulatory network: A network consisted of interacting genes that actually control certain processes in molecular biology.

Global optimum: The best solution in a single objective optimization problem.

Level-back: The number of columns back a node in a particular column can connect to in Cartesian genetic programming.

Marker assisted selection (MAS): A selection strategy in plant breeding based on genotypes (combinations of allele markers).

Monte-Carlo algorithms: A class of computational algorithms that rely on repeated random sampling to compute their results

Mutation rate: The probability that a mutation operator will be applied on a chromosome in multi-objective algorithms.

Nearest neighbor: See *Fitness Sharing*.

Network assisted selection (NAS): A proposed selection strategy based on mathematic networks for plant breeding in this thesis.

Neuron: A basic element in neural network model.

Neutrality: Used in Cartesian genetic programming. Refers to inactive components which may be activated in the future.

Non-dominated sorting: See *Domination Counting*.

Non-dominated front: The set of non-dominated solutions found at certain time by a given algorithm.

Nucleosome: The basic unit of eukaryotic chromosome structure; a ball of eight histone molecules wrapped about by two coils of DNA.

Nucleotide: A molecule composed of a nitrogen base, a sugar, and a phosphate group; the basic building block of nucleic acids.

*Objective function***:** The function to be optimized. In a minimization problem, the fitness varies inversely as the objective function.

Objective function space: The corresponding values of an objective function in a search space.

Optimality: Equivalent to optimization; the study of problems in which one seeks optimal solutions.

Pareto-based: See *aggregation-based*.

Pareto set: The set of optimal non-dominated solutions.

Pareto front: The projection of Pareto set in objective function space.

Particle: A basic component in particle swarm optimizations.

Particle swarm optimization: A type of stochastic algorithms inspired from cooperative behavior of a flock of birds.

Phenotype: The form taken by some character in a specific individual.

Population-based algorithm: An algorithm that maintains an entire set of candidate solutions, each solution corresponding to a unique point in the search space of the problem.

Promoter region: A regulatory region a short distance from the end of a gene that acts as the binding site for RNA polymerase.

Quantitative trait locus mapping (QTL): The statistical study of the alleles that occur in a locus and the phenotypes (physical forms or traits) that they produce.

Redundant: A term used in genetic programming to indicate components that have not appeared in a solution at a certain point in time.

Reuse: A term used in Caretisan genetic programming to indicate that subgraphs can be used simultaneously by others.

Reverse engineering: A process of inferring the interactions of a system from its behaviors.

Search space: Set of all possible solutions for any given optimization problem. Almost always, a neighborhood around each solution can also be defined in the search space.

Selection, *crossover* (*recombination*) and *mutation*: Three well-known mechanisms in biological evolution, as three important steps (operators) borrowed in evolutionary algorithms.

Stochastic algorithms: Methods which incorporate probabilistic (random) elements.

Stochastic multi-objective optimization: Stochastic algorithms that are able to deal with multi-objective optimization problems.

Training process: A process to obtain weights from data based on goodness-of-fit in a neural network model.

Transcription: The synthesis of RNA from a DNA template.

Transcription factor: A protein that binds to a cis-regulatory element (for example, an enhancer) and thereby, directly or indirectly, affects the initiation of transcription.

Appendix B - P-value of Gene Identification

Gene #	Average p-value
92	0.091075
18	0.091158
24	0.13037
80	0.14767
32	0.15
54	0.16448
17	0.16469
74	0.16642
13	0.16668
20	0.16721
21	0.16787
93	0.16798
95	0.16835
39	0.16839
50	0.16913
89	0.16925
6	0.16933
90	0.16939
71	0.16953
28	0.16957
25	0.16982
29	0.16993
36	0.17016
26	0.17029
48	0.17032
30	0.17069
77	0.17072
11	0.17078

8	0.17085
15	0.1715
61	0.17163
76	0.1717
23	0.17179
66	0.17188
12	0.17204
79	0.17207
85	0.1725
91	0.17251
46	0.17266
56	0.17272
27	0.17275
35	0.1729
1	0.17304
19	0.17313
57	0.17324
53	0.17326
40	0.17327
59	0.17331
84	0.17335
96	0.17338
55	0.17352
49	0.17354
52	0.17357
44	0.17399
2	0.17405
97	0.17416
9	0.17421
63	0.17433
67	0.17437

94	0.17451
69	0.17454
87	0.17467
45	0.17473
78	0.17478
37	0.17482
72	0.17482
64	0.17488
86	0.17492
38	0.17493
100	0.17513
82	0.1753
65	0.17535
60	0.17543
83	0.1755
88	0.17557
34	0.1758
47	0.17599
22	0.17607
33	0.17627
42	0.17628
62	0.1763
73	0.17631
99	0.17652
51	0.17653
41	0.1766
70	0.17661
43	0.17705
4	0.1775
31	0.17762
3	0.17764

14	0.17764
7	0.17784
5	0.17786
81	0.17788
58	0.17799
98	0.17862
75	0.17866
16	0.17913
68	0.17922
10	0.17943