유전 프로그래밍을 위한 적응 연산자 메커니즘

Adaptive Operator Mechanism for Genetic Programming

MinHyeok Kim

Ph.D. Thesis

School of Computer Science and Engineering
Seoul National University

August, 2013

Supervisor: Robert Ian McKay

Abstract

Genetic programming (GP) is an effective evolutionary algorithm for many problems, especially suited to model learning. GP has many parameters, usually defined by the user according to the problem. The performance of GP is sensitive to their values. Parameter setting has been a major focus of study in evolutionary computation. However there is still no general guideline for choosing efficient settings. The usual method for parameter setting is trial and error.

The method used in this thesis, adaptive operator mechanism, replaces the user's action in setting rates of application of genetic operators. adaptive operator mechanism autonomously controls the genetic operators during a run. This thesis extends adaptive operator mechanism to genetic programming, applying existing adaptive operator algorithms and developing them for TAG3P, a grammar-guided GP which supports a wide variety of useful genetic operators. Existing adaptive operator selection algorithms are successfully applied to TAG3P; their performances are competitive with systems without an adaptive operator mechanism. However they showed some drawbacks, which we discuss. To overcome them, we suggest three variants on operator selection, which performed somewhat better.

We have investigated evaluation of operator impact in adaptive operator mechanism, which measures the impact of operator applications on improvement of solution. Hence the impact guides operator rates, evaluation of operator impact is very important in adaptive operator mechanism. There are two issues in evaluation of operator impact: the resource and the method. Basically all history information of run are able to be used as resources for the operator impact, but fitness value which is directly related with the improvement of solution, is usually used as a resource. By using a variety of problems, we used two kinds of resources: accuracy and structure

in this thesis. On the other hand, although we used same resources, the evaluated impacts are different by methods. We suggested several methods of the evaluation of operator impact. Although they require only small change, they have a large effect on performance.

Finally, we verified adaptive operator mechanism by applying it to a real-world application; a modeling of algal blooms in the Nakdong River. The objective of this application is a model that describes and predicts the ecosystem of the Nakdong River. We verified it with two researches: fitting the parameters of an expert-derived model for the Nakdong River with a GA, and modeling by extending the expert-derived model with TAG3P.

Keywords: **Adaptive Operator Mechanism, Adaptive Operator Selection, Genetic Programming, Evolutionary Algorithms, Parameter Control, Parameter Setting**

Student Number: 2005-23499

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Genetic Programming (GP, Koza (1992, 1994)) and Genetic Algorithm (GA, Holland (1975); Goldberg (1989)) are popular evolutionary algorithms and they are effective for many problems; while GA is commonly used for optimization, GP is usually used for learning. Both systems have many common features. They have a population and make it be evolved for searching a solution. Comparing to GA, the main feature of GP is that its representation is a tree-based structure. Thus chromosomes of GP can be easily extended under one's extension rules and they can express structurally complex solutions. Based on the feature, many various types of GP, such as grammar-guided genetic programming (GGGP), in which formal grammars are used to build solutions, are exist. In these GP systems, formal grammars can be used to set a declarative bias on the search process of GP.

In addition to the above, diverse and known as useful genetic operators for GP and GGGP are investigated, and more operators are being considered. Because GP systems usually operate on an infinite solution space, it is difficult to analyze the

effects of genetic operators exactly. Thus, controlling the effects of genetic operators appropriately become an important issue in evolutionary computation, since unsuitable settings can waste computational effort, as the number of genetic operators increase. In the simplest approach, parameter settings may be based on folklore or a priori knowledge or on preliminary experiments. But prior knowledge may be wrong, while preliminary experiments are both expensive, and potentially misleading. More sophisticated methods use the preceding performance of operators as a guide to their likely future performance. The first important step came with Schwefel's one-fifth success rule (Schwefel (1981)) for continuous optimization in evolution strategies (ES), which adapted mutation step size. However it has limited relevance to operators with discrete effects as appear in GP or GA. As the solution of this problem, we introduce adaptive operator mechanism, which is an adaptive mechanism on genetic operator, for on-line controlling the application of the variation operators.

1.2 Our Approach and Its Contributions

Extension of Adaptive Operator mechanism to GP

While many studies of adaptive operator mechanism have investigated GA and Differential Evolution (DE) domain (Qin et al. (2009); Gong et al. (2010)), relatively little research has been applied to GP (Niehaus and Banzhaf (2001)). By directly applying several adaptive operator mechanisms to GP systems, we extend domain area of adaptive operator mechanism to GP systems. Moreover the result suggested a guideline on new adaptive operator mechanism for GP.

New Adaptive Operator Mechanism for GP

Most adaptive operator mechanisms were originally developed for general systems, not for a specific GP such as TAG3P. Thus even though adaptive operator mechanism works well for GP, it has restrictions, in particular when there are many operators. We investigated new adaptive operator mechanisms for GP systems: three operator selection variants and a number of useful methods of the evaluation of operator impact. The variants, which we suggested, are designed for many operators in different ways. We compared them to a typical GP and existing adaptive operator mechanisms. We approached the evaluation of operator impact in two ways: resources and methods. Two resources, accuracy and structure information, are used to evaluate an operator impact on diverse problems and four methods, which require only small change, large affected to performance.

Analysis of Genetic Operators

Through the empirical analysis of experiments, which are explained at section 1.2, we generated a deeper understanding of genetic operators of TAG3P: subtree-crossover, subtree-mutation, reproduction, insertion, deletion, duplication, truncation, point replacement and relocation. The analysis showed how these operators worked at different situations: different problems and the progress of a run. It suggested a combination of operators, which is appropriate for a given problem.

Real-world Application

We applied adaptive operator mechanisms to EAs working on a real-world problem: Nakdong River Modeling. This enabled us to verify the value of the usefulness of adaptive operator mechanism in a real-world application. The objective of the Nakdong River modeling problem is to build a prediction model for zooplankton in

the Nakdong River. Details of the problem will be described later. This work consist of two parts. One is parameter fitting with GA, and the other is modeling with TAG3P.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 explains background knowledge related to each chapter, including explanation of tree adjoining grammar based genetic programming. First, we describe genetic algorithm and genetic programming. Then, tree adjoining grammar based genetic programming is explained. Chapter 3 introduces adaptive operator mechanisms, mainly adaptive operator selection. We summarize adaptive mechanisms and adaptive operator selection, and introduce three algorithms. Then, we apply the algorithms to tree adjoining grammar based genetic programming, which has nine diverse genetic operators, in chapter 4. Chapter 5 proposes three new operator selection algorithms for the GP system. They are variants of algorithms in chapter 4 and show better performance than previous ones. In addition, the empirical analysis provides deep understanding of genetic operators and their features. Chapter 6 suggests several methods for the evaluation of operator impact, which guide operator rates by evaluating the operator impact based on the whole run history. Chapter 7 verifies the usefulness of adaptive operator mechanism with a real-world application: Nakdong River modeling. Not only genetic programming but also genetic algorithm were used for this experiment. We finish in chapter 8 with a summary of the conclusion and future works.

Chapter 2

Related Works

Evolutionary Algorithms (EAs) are generic population-based metaheuristic optimization algorithms, which mimic biological evolution in nature. To solve a give problem in EAs, individuals in a population affect to each other and evolve during a run. This chapter explains two EAs: genetic algorithm (GA) and genetic programming (GP). Both algorithms are used as a base algorithm which adaptive mechanism is applied to. In particular, We introduce a specific GP, tree adjoining grammar guided GP (TAG3P). It is a grammar-guided GP which has a tree adjoining grammar (TAG) based representation.

2.1 Evolutionary Algorithms

2.1.1 Genetic Algorithm

From the Genetic Algorithm (GA, Holland (1975); Goldberg (1989)) is proposed, it is a currently popular evolutionary algorithm which shows its usefulness for solving diverse real-world problems, in particular optimizations (Mitchell (1996)).

GA uses linear and fixed length of strings as the representation, which is called

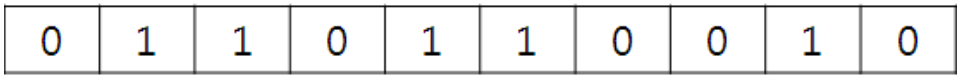

Figure 2.1: Representation of Genetic Algorithm

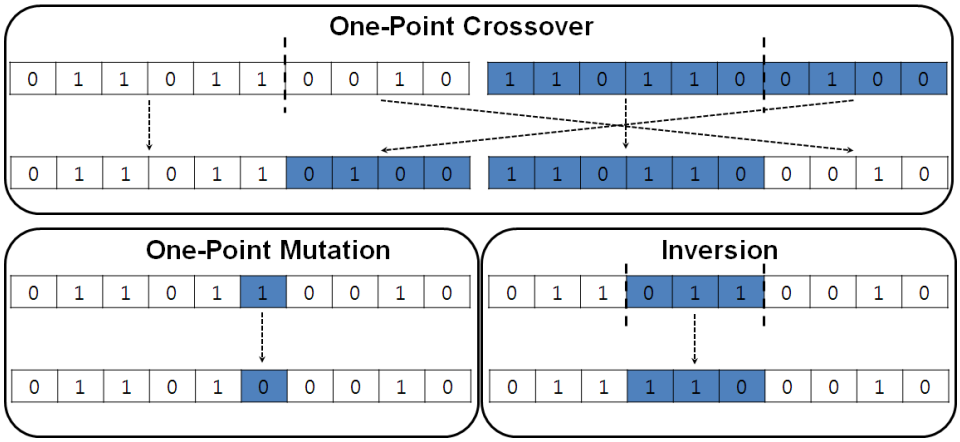

Figure 2.2: Genetic Operators in GA

chromosome (Fig. 2.1). Each element of chromosome, called as gene, has a value of a variety of types; binary value, integer, real number, character and so on. Each gene represents the value of some aspect of the solution, so a type of gene value is dependent on a given problem to solve. There have been diverse GA systems using different representation such as real coding, gray coding, messy coding, variant length chromosomes, and so on (Mitchell (1996)).

Genetic operator changes individuals to find a solution. Crossover and mutation are two main genetic operators. Crossover is the main genetic operator and it resembles genetic recombination of genome in biological evolution. Mutation is the secondary operator which is used to keep a degree of genetic diversity in the population. Holland (1975) suggested three operator on binary-coded GA; one-point crossover,

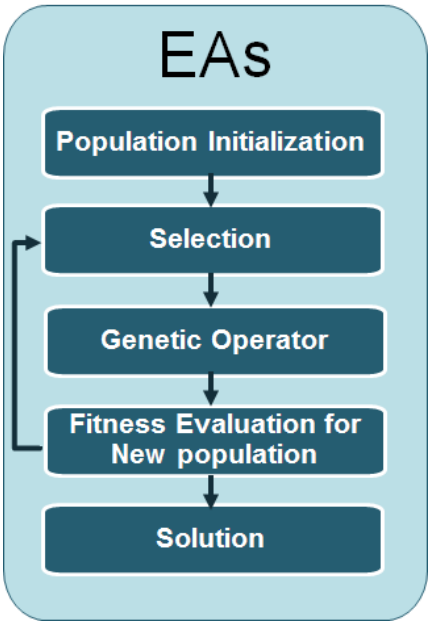

Figure 2.3: Scheme of Evolutionary Algorithms

one-point mutation and inversion (Fig. 2.2). One-point crossover exchanges segments of two chromosomes. One-point mutation flips a random gene. Inversion changes a segment of chromosome in reverse order. From three operators in the beginning, there have been a lot of new and bio-inspired genetic operators (Mitchell (1996); Bäck et al. (2000a,b)). For example two-point crossover, uniform crossover, uniform mutation and Gaussian mutation are existing. Indeed, it is not difficult to design operators which make limited changes in GA chromosome.

The basic process of GA is shown in figure 2.3. Figure 2.3 describes the basic scheme of all EAs, including GA. After GA randomly makes an initial population, it repeats to apply genetic operators to population until the end criteria is satisfied.

2.1.2 Genetic Programming

Genetic programming (GP, Koza (1992, 1994)) has been defined as a machine learning method to evolve computer programs (Banzhaf et al. (1997)). GP is inspired by GA so GP has many common features with GA. The main difference between GA and GP is the representation of chromosome. While GA uses a fixed-length of string-based chromosome, GP uses a tree-based chromosome with variable size and shape. A tree-based representation makes GP is flexible. Therefore, GA is used for the task of optimizing parameters for solutions when their structure is known, while GP is more used to learn and discover both content and structure of solutions (Banzhaf et al. (1997)).

The main genetic operators in GP are also crossover and mutation. They change subtree in GP chromosome. For example, subtree crossover exchanges subtrees of two chromosomes if they can be attached to the opposite tree. Genetic operators in GP change not only values in tree but also structure of tree, therefore, comparing to GA, many operators which more diversely affect individuals exist in GP. Many research on GP operators and their effects have long been investigated. Recent research on GP operators has focused on effects by restrictions on subtree crossover; by some restrictions, subtree crossover affects on GP bloat (Angeline (1998); Langdon (2000); Terrio and Heywood (2002)), or it causes specific changes (McPhee et al. (2008); Beadle and Johnson (2008); Nguyen et al. (2009)).

Original GP (Koza (1992)) represents solutions as expression trees, and in principle searches the space of all expressions that can be built out of a specific set of function and atom symbols.[1] This generality is often useful, but it also can lead to

[1]We here avoid Koza's terminology of terminals and nonterminals for these symbols, because it causes confusion in the context of grammar-based systems. We reserve those terms for their original – grammatical – meaning.

problems. We frequently have prior knowledge that can restrict our search to specific forms. We may know that other forms are meaningless (e.g semantically inconsistent: $Y = \text{TRUE} \times 3$), or almost certain not to be correct solutions (e.g. incompatible with physical consistency laws: $Y = \text{mass} \times \text{volume} + \text{time}$), or unlikely to be useful even if they are correct (e.g. requiring information that is unlikely to be available: $Y(t) = X(t+1) \times Z(t+1)$).

Grammar guided GP (GGGP) addresses these issues by restricting the search space to the language defined by a grammar. The underlying assumption is that our background knowledge can generally be represented by the restrictions implicit in these grammars. Various kinds of grammars have been used, those from the Chomsky's hierarchy, and particularly context free grammars (CFGs) being the most common (Wong and Leung (1997); Whigham (1994, 1995); Ryan et al. (1998)) . With the exception of grammatical evolution (GE), they resemble expression-tree GP in evolving tree structures, with restricted forms of subtree crossover and mutation designed to maintain consistency with the grammar.

2.1.3 Tree Adjoining Grammar based Genetic Programming

Tree Adjoining Grammars

Tree adjoining grammars (TAGs) are tree-generating and analysis system for Natural Language Processing (Joshi et al. (1975); Joshi and Schabes (1997)). The objective of TAGs is to more directly represent the structure of natural languages than Chomsky languages. Chomsky hierarchy grammars were originally designed to highlight the reusability aspects of natural language – the relationship between "The cat sat on the mat" and "The cat sat on the dog", "The cat sat next to the mat" etc. What Joshi et al. (1975) was the first to recognize is that they don't do a particularly good job of explaining the relationship between the first sentence and "The big, black cat sat

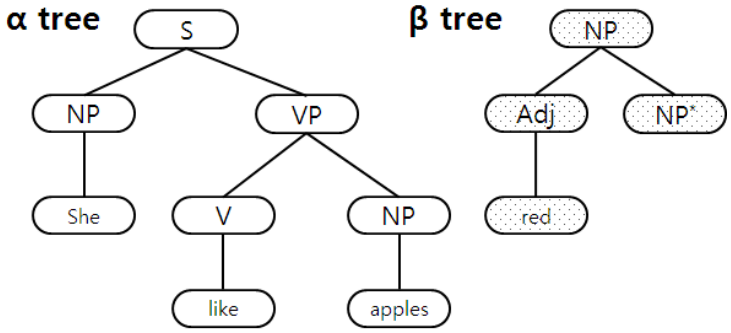

Figure 2.4: Simple Elementary Trees

possessively on the shaggy gray mat which it had commandeered". Joshi pioneered the idea of forming a grammar using insertable elements (auxiliary or β trees), which can always be inserted into certain contexts. Thus a β tree representing an adjective such as "black" can always be inserted (adjoined) at the start of a noun phrase. In this view, all sentences can be built up from a basic stock of simple sentences (the α trees) by the operation of adjunction; Joshi demonstrated that these tree adjunct grammars subsume CFGs and are mildly context sensitive. They can also be more succinct than equivalent CFGs. For example, representing the subject/predicate number agreement of English in a CFG would require complete copies of the rest of the grammar, one for singular sentences and the other for plural. Tree adjunct grammars can directly and economically represent this agreement. Figure 2.4 shows a example of elementary trees: α tree and β tree. While α tree is a general tree in which terminal or non-terminal symbols are labeled on nodes, β tree has a foot node on which * is marked. Foot node is used when β tree is inserted (adjoined) to other tree (Fig. 2.5).

It was subsequently recognized that incorporating the characteristic operation of

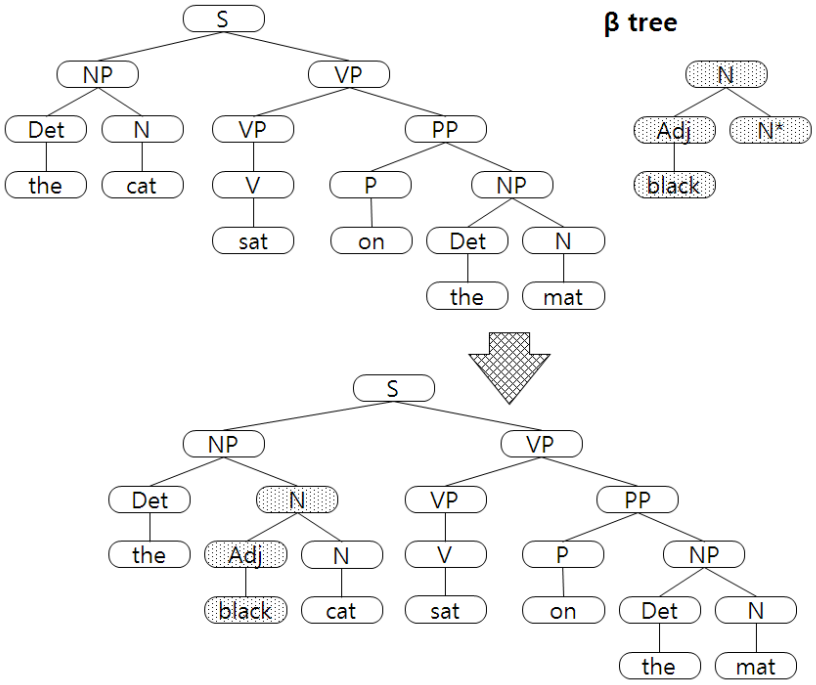

Figure 2.5: Adjunction

CFGs – substitution – into tree adjoining grammars was useful. It does not extend the set of representable languages, but it substantially reduces the complexity of representation (Joshi et al. (1975); Joshi and Schabes (1997)). For example, there are many adverbs in English – 'very', 'darkly', etc. Without substitution, we would need a separate β tree for each adverb. With substitution, we can have a single nonterminal – 'ADVERB' – and substitute with adverbs as required. This might seem a trivial difference (replacing many β trees with a corresponding number of lexical elements). The true economy is revealed when we consider that adverbs may be used in different contexts (modifying adjectives, verbs, whole sentences etc.). Without substitution, we would have to repeat β trees for the whole lexicon for each

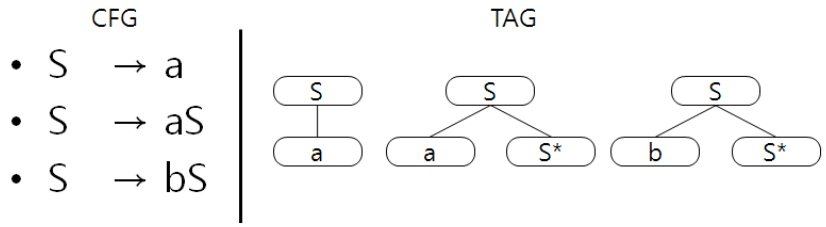

Figure 2.6: Simple example of CFG and TAG

use. With substitution, we only need one β tree (incorporating 'ADVERB') for each use, together with a single set of rules providing for substitution of 'ADVERB' by its lexicon. Thus today, the acronym TAG normally means a tree-adjoining grammar (i.e. one which incorporates both substitution and adjunction as operators).

Tree Adjoining Grammar Guided Genetic Programming

TAGs have been used as the basis for a number of EA systems including GP (Hoai et al. (2002, 2006); Murphy et al. (2010)). Tree Adjoining Grammar Guided Genetic Programming (TAG3P, Hoai et al. (2002); Hoai (2004); Hoai et al. (2006)) is a grammar guided genetic programming which uses TAGs format as the tree representation.

TAG3P is based on tree adjoining grammars; specifically on a subset of these grammars in which substitution is only permitted to introduce terminals. Using TAG3P is very similar to using a CFG-guided GP system. TAG3P specifies a grammar to define the search space, but unlike with CFGs, it does so by defining sets of α and β trees. A simple example, with an equivalent CFG grammar is shown in fig. 2.6.

TAG3P uses TAG derivation trees over this grammar as its genotype representa-

tion. Thus a basic TAG3P implementation merely needs to supply an initialization mechanism for TAG derivation trees, and crossover and mutation operators. In the simplest form, it can simply use subtree mutation and crossover as in GGGP, with a restricted for of subtree mutation (restricted in this case to start with an α tree) as its initialization algorithm. The most complex part of the implementation lies in the first half of the genotype-phenotype mapping, in which a TAG derivation tree is transformed to the corresponding Chomsky-grammar derived tree. But even in this case, although the coding is complex, the idea is relatively simple – it simply consists of starting with the α tree, and then starting from the root of the derivation tree, applying the adjunction (or substitution) specified in a particular node to the current derived tree at the specified position. Once the derived tree has been generated, it is a normal Chomsky-style derivation tree as in any Chomsky-based GGGP system, and may be transformed to the final expression tree and evaluated, just as in GGGP.

TAG3P has a lot of useful genetic operators (Hoai (2004); Hoai et al. (2006)). In this thesis, we introduce and use nine different TAG3P operators; subtree crossover, subtree mutation, reproduction, insertion, deletion, duplication, truncation, relocation and replacement. Among them, insertion and deletion, and duplication and truncation are in a dual relation with each other to avoid size biases, they are used as a dual pair, working together, and being applied at an equal rate. In summary, then, we used 5 single operators and 2 dual operators. They are briefly summarized in table 2.1, and explained in more detail below.

Subtree Crossover (X) is the traditionally most-used GP operator, well known for its effectiveness in exploitation of good solution components. A random point is selected in the first parent; a compatible location is then chosen in the second parent, and the subtrees below those points in both parents are exchanged. In expression-

Table 2.1: Genetic Operators of TAG3P

X	Subtree Crossover	Exchanges random subtrees of two individuals
M	Subtree Mutation	Replaces a random subtree with a newly generated one
Rd	Reproduction	Reproduction retains the individual unchanged
I	Insertion	Inserts exactly one adjunction instruction on the frontier
D	Deletion	Deletes exactly one adjunction instruction from the frontier
I/D		Insertion/Deletion dual operator
D	Duplication	Copies a random subtree, adding it to another location in the individual
T	Truncation	Removes a random subtree from the individual.
D/T		Duplication/Truncation dual operator
Rep	Point Replacement	Replaces a frontier node with another
Rel	Relocation	Moves a randomly chosen subtree to another location in the individual

tree GP, all locations are compatible, but in GGGP and TAG3P, they are required to be grammar-compatible (i.e. have the same non-terminal label). Of course, this exchange may breach size limits. This may be handled in various ways, but in TAG3P, the whole operation is repeated a fixed number of times; if none of these tries are successful, the operation is aborted and a new one chosen.

Subtree Mutation (M) is also a traditional GP operator, particularly favored for its exploratory capabilities. The subtree at a randomly-chosen location in the parent is deleted, then replaced with one newly generated using the (random) initialization algorithm.

Reproduction (Rd) is the final traditional operator from GP: it simply replicates individuals from one generation to the next (and thus shares some properties with elitism, in that, in combination with selection, it increases the probability of retaining fit individuals).

(Point) Insertion & Deletion (I/D) while insertion adds to the frontier an instruction to adjoin one β tree to a random open location, deletion removes one. Thus the expected size change (plus or minus) is one. Thus used in dual mode (i.e. with equal probability of application), the expected change in average individual size resulting from these operators is zero. Because they cause the minimum possible change, they are useful for fine-tuning the size or structure of individuals.

(Subtree) Duplication & Truncation (D/T) are also used as dual operators, again to avoid size bias. They are useful for coarse adjustment. Duplication (sometimes known as replication) copies a randomly-chosen subtree from the individual, adding it to a randomly-chosen compatible point in the same individual. Truncation is similar to deletion, but removes the whole subtree below a randomly chosen point.

Point Replacement (Rep) is a small-scale operator, randomly choosing a frontier node and replacing it with another (it is thus equivalent to a deletion/insertion sequence, and has no effect on size).

Relocation (Rel) makes larger-scale structural changes in an individual. It disconnects a random subtree from an individual, and re-connects it in a randomly-chosen compatible location in the same individual. It has no effect on size, and may be viewed as a form of (size-fair) self-crossover.

Chapter 3

Adaptive Mechanism and Adaptive Operator Selection

This chapter describes adaptive mechanism, in particular, adaptive operator selection (AOS). It introduces three existing AOS methods, probability matching (PM), adaptive pursuit (AP) and multi-armed bandits (MABs), which have showed good performances in genetic algorithm (Goldberg (1990); Thierens (2007); DaCosta et al. (2008)).

3.1 Adaptive Mechanism

Evolutionary algorithms have many parameters. Hence the issue of parameter setting of an evolutionary algorithm is critical for good performance, finding the well-suited setup for an evolutionary algorithm have been a long grand challenge of the field (Eiben et al. (2007)). Eiben distinguished two major forms of parameter setting: parameter tuning and parameter control (Fig. 3.1). Parameter tuning is a typical approach. It finds good parameter values based on preliminary experiments and

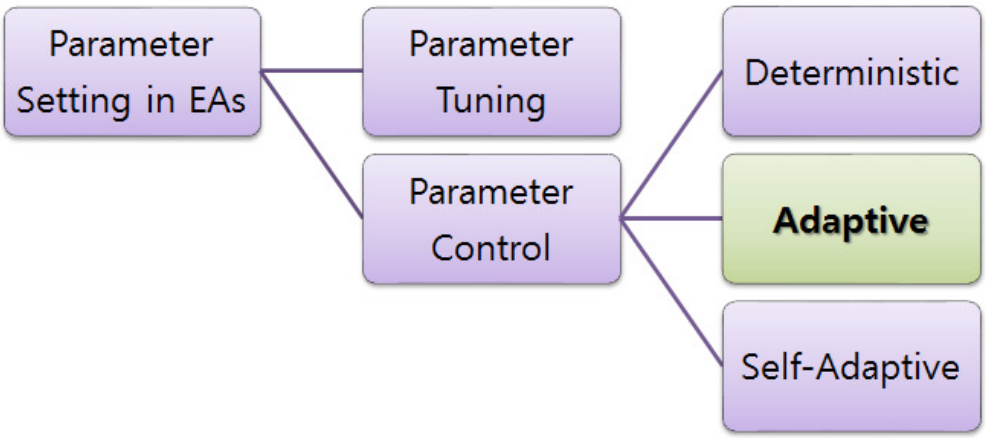

Figure 3.1: Parameter Setting in EAs

rules of thumb and sets them before the run (De Jong (2007)). Meanwhile parameter control changes parameter values during the run. Parameter control can be further distinguished deterministic, self-adaptive, and adaptive. Deterministic predefine parameter values as functions of time, usually linearly. In most cases however, trials and errors are still essential in deterministic. In self-adaptive, parameters are in the individual and they are optimized together. It is acknowledged one of the most effective approaches, but it often increases the complexity of the problem. Adaptive, which is mentioned as adaptive mechanism in this thesis, predefines parameter values as functions of all history information of the run. During the run, adaptive receives feedback and modifies the parameter values.

3.2 Adaptive Operator Selection

The parameters that control evolutionary operators have long been an issue in EAs. While some parameters such as the population size and maximum number of generations are relatively easy to be set, the operator application rates are more difficult to be set. Moreover many operators in GP make it be more difficult. Two kinds of parameters are involved: parameters that control the rates of application of specific operators, and parameters that control the scale of the operators. The former were of particular concern in the genetic algorithms literature, and the latter in Evolution Strategies. AOS is an approach of adaptive mechanism which controls only parameters of genetic operators. The objective of AOS is to define an on-line strategy for selecting the most appropriate variation operators. Using the preceding performance of operators as a guide to their likely future performance, AOS modifies parameter values and suggests well-suited operators during the run.

As shown on Fig. 3.2, AOS interactively works with EAs. For every end of generations, AOS receives history information from EAs. Based on the information, AOS updates its internal status and suggests new operator application rates for the next generation to EAs. This process is repeated until the end of the run. In general, AOS is consists of two parts; operator selection and evaluation of operator impact, and it includes an internal status vector of which each element is corresponded to each genetic operator. The internal status indicates the usefulness of operator based on accumulated impacts.

3.2.1 Operator Selection

Operator selection is a core part of AOS which aims to suggest the most appropriate operator based on the internal status. From the internal status, operator selection proposes a new operator application rates to EAs. After values of internal status are

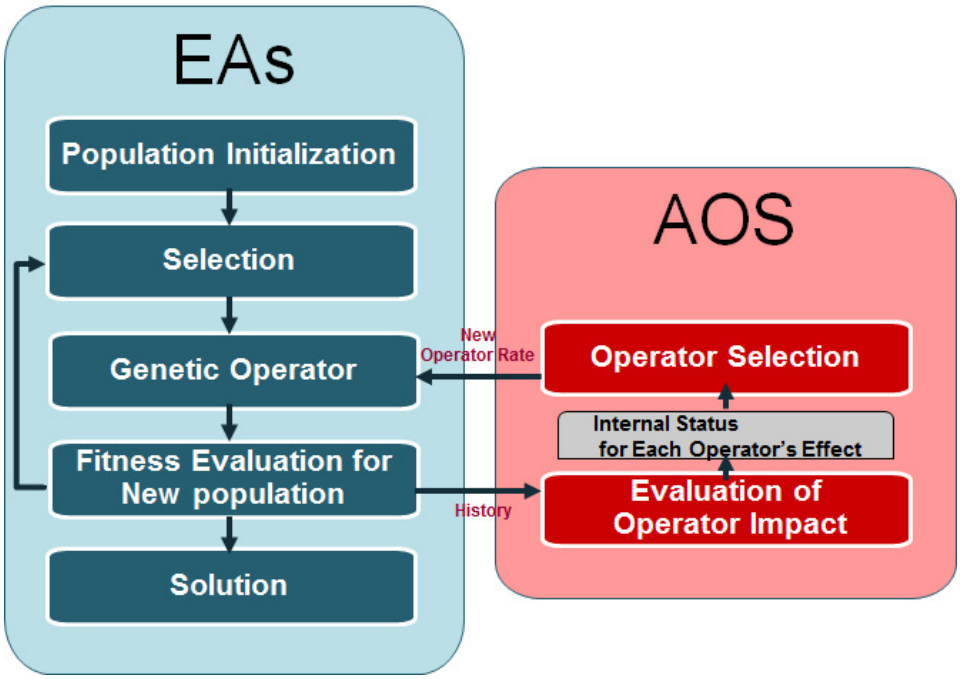

Figure 3.2: Scheme of Adaptive Operator Selection

updated by evaluation of operator impact, operator selection start to recommend the most effective operator. With the corresponding algorithm, operator selection generally changes the operator application rates as a new one.[1] With the suggested rates for operator, EAs are able to choose the effective genetic operator.

3.2.2 Evaluation of Operator Impact

Impact is a measure of the effect of operators for the current generation, and it also works as a guideline on change of operator rates. For example, an operator

[1]MABs suggests only one operator, however we consider that MABs suggests an operator application rates in which one operator has 1 but all other have zero.

which has a good impact value, easily has high rate, then it has more chance to be used in and affect to a run. In other word, impact is the main criteria of the operator usage. Therefore, defining the impact of operator is an important factor on AOS, and what evaluation of operator impact does are to define the impact and to update the internal status with the impact. As resources for evaluating operator impact, all running history information can be used. In most cases of GAs, fitness is widely used, however more information, such as a depth of individuals, can be used in GPs. Although the same operator selection methods are used, results are differed by evaluation of operator impact. We will discuss it in more detail at chap 6.

Adaptive operator selection is comprised of a triple (I, Op, Ip), where I is an internal status, Op is an operator application rates and Ip is a current impact value. All elements are vectors of which size is same to the number of genetic operators. Ip is calculated from newly generated population. Then Evaluation of operator impact updates I with Ip. While Ip is a measure of immediate effect of genetic operators, I is an accumulated effect of operators. I is updated with Ip and other history information of a run, such as the number of operator is selected. Finally, based on I, operator selection makes a new Op vector for the next generation. With this suggested Op, EAs choose the effective genetic operator.

3.3 Algorithms of Adaptive Operator Selection

As parameter setting has long been issued, many studies for AOS have been proposed. The first important step came with Schwefel's one-fifth success rule (Schwefel (1981)) for continuous optimization in evolution strategies (ES), which adapted mutation step size. B.A. Julstrom investigated a mechanism of adapting operator probabilities in a steady-state GA. A probability of each operator in the mechanism

is proportional to the corresponding recent contribution (Julstrom (1995)). A. Tuson and P. Ross proposed performance based operator probabilities which are adjusted during a run of GA (Tuson and Ross (1996)). CW Ho showed a well-performed GA system with adaptive probabilities (Ho et al. (1999)). They adapt the mutation and the crossover rates during a GA run. More recently, Barbosa proposed adaptive operator probabilities in real coded steady-state GA (Barbosa and Sá (2000)). Thierens suggested a method, called adaptive pursuit, which has rapid converged adaptive operator (Thierens (2005)). DaCosta and Fialho proposed multi-armed bandits which choose an operator by balancing between exploration and exploitation (DaCosta et al. (2008); Fialho and Schoenauer (2009)). In this thesis, we introduce the last three algorithms; probability matching, adaptive pursuit and multi-armed bandits.

3.3.1 Probability Matching

Probability matching (PM, Goldberg (1990); Barbosa and Sá (2000)) is a simple and eidetic algorithm which chooses one of operators iteratively, for applying it to the system. With the recent operator impact which is measured from new generated population, PM updates the internal status. Then PM modifies operator application rates for the next operator selection, and this process is repeated until the end of a run. In whole process, PM aims to match the operator application rates (probabilities) to their corresponding impact.

Formally, let's assume that there is a set of K genetic operators $\{OP_1, OP_2, ..., OP_K\}$. An operator OP_i has its corresponding probability (application rate) $P_i(t)$, corresponding impact $I_i(t)$, and corresponding quality (internal status) $Q_i(t)$, for time t. Probability $P_i(t)$ is used for choosing an operator OP_i ($0 \leq P_i(t) \leq 1 \forall t, i$ and $\sum_{i=1}^{K} P_i(t) = 1$). Evaluation of operator impact works follows. At first, an impact $I_i(t)$ is returned from the system, when an operator OP_i is executed at time t, and it

is a measure for the usefulness of corresponding operator. A quality $Q_i(t+1)$ means how good the operator OP_i is, and its value is updated with the former quality $Q_i(t)$, the former impact $I_i(t)$ and the adaptation rate α ($0 \leq \alpha \leq 1$) (Eq. 3.1).

$$Q(t+1) \quad = \quad Q(t) + \alpha(I(t) - Q(t)) \tag{3.1}$$

$Q_i(t+1)$ is used to set the value of the next probability $P_i(t+1)$. Basically, $P_i(t+1)$ is set as the proportion of $Q_i(t+1)$ to the sum of all qualities $\sum_{j=1}^{K} Q_j(t)$. However, for no operator gets 0 probability value, it uses the minimum probability P_{min}: $0 \leq P_{min} \leq 1$ (Eq. 3.2). It means, the maximum probability which an operator can get, is restricted to $1-(K-1)P_{min}$.

$$P_i(t+1) \quad = \quad P_{min} + (1 - K \cdot P_{min}) \frac{Q_i(t)}{\sum_{j=1}^{K} Q_j(t)} \tag{3.2}$$

The detail algorithm is given in table 3.1.[2]

In conclusion, PM sets the operator application rate via the performance of operator directly; The fairness and simpleness are main merits of PM.

3.3.2 Adaptive Pursuit

Adaptive pursuit (AP, Thierens (2005, 2007)) algorithm is based on Pursuit (Thathachar and Sastry (1985)) which is a rapidly converging algorithms for learning automata. The main difference between PM and AP, is a method to update operator application rates. While PM sets operator rates to the same portion of corresponding impacts, AP emphasize only the most effective operator.

[2] As presented in Thierens (2005). This is applied to all details of algorithms.

Table 3.1: Algorithm for Probability Matching

ProbabilityMatching(P,Q,I,K,P_{min},α)
for $i \leftarrow 1$ **to** K **do**
  $P_{init}(i) \leftarrow \frac{1}{K}$
  $Q_{init}(i) \leftarrow 1.0$
end for
while NotTerminated?() **do**
  OperatorSelectionBy$P(t)$
  $I_i(t) \leftarrow$ GetImpact
  $Q_i(t+1) \leftarrow Q_i(t) + \alpha(I_i(t) - Q_i(t))$
  for $i \leftarrow 1$ **to** K **do**
    $P_i(t+1) \leftarrow P_{min} + (1 - K.P_{min}) \frac{Q_i(t)}{\sum_{j=1}^{K} Q_j(t)}$
  end for
end while

The basic process of AP is same to PM. Under the same assumption of a set of K operators $\{OP_1, OP_2, ..., OP_K\}$, operators are selected with corresponding probability $P_i(t)$. After operators are executed, the impact $R_i(t)$ is returned and quality $Q_i(t+1)$ is updated. At this time, evaluation of operator impact is exactly same to PM. In other words, quality is updated with previous quality value, returned impact and the adaptation rate α (Eq. 3.1).

However a detail of operator selection is different. To set values of $P_i(t+1)$, AP firstly finds one operator OP_{i^*} which has the largest value of quality $Q_{i^*}(t+1)$, then AP divides operators two groups; the most effective operator OP_{i^*} and others. AP increases $P_{i^*}(t+1)$ with the maximum probability P_{max} and learning rate β, but it decreases others with P_{min} and β (Eq. 3.3). In conclusion, AP grows up only the

most effective operator and reduces others instead.

$$\begin{aligned} i^* &= argmax\{Q_i(t+1), i = 1...K\} \\ P_{i^*}(t+1) &= P_{i^*}(t) + \beta(P_{max} - P_{i^*}(t)) \\ P_i(t+1) &= P_i(t) + \beta(P_{min} - P_i(t)) \qquad \text{for} \quad i \neq i^* \end{aligned} \tag{3.3}$$

The detail algorithm is given in table 3.2

Table 3.2: Algorithm for Adaptive Pursuit

AdaptivePursuit(P,Q,I,K,P_{min},P_{max},α,β)
$P_{max} \leftarrow 1 - (K-1)P_{min}$
for $i \leftarrow 1$ **to** K **do**
  $P(i) \leftarrow \frac{1}{K}$
  $Q(i) \leftarrow 1.0$
end for
while NotTerminated?() **do**
  OperatorSelectionBy$P(t)$
  $I_i(t) \leftarrow$ GetImpact
  $Q_i(t+1) \leftarrow Q_i(t) + \alpha(I_i(t) - Q_i(t))$
  $i^* \leftarrow$ argmax$(Q_{i^s}(t+1))$
  $P_{i^*}(t+1) \leftarrow P_{i^*}(t) + \beta(P_{max} - P_{i^*}(t))$
  for $i \leftarrow 1$ **to** K **do**
    if $i \neq i^*$ **then**
      $P_i(t+1) \leftarrow P_i(t) + \beta(P_{min} - P_i(t))$
    end if
  end for
end while

The advantage of AP is quick fluctuation. Once an operator is chosen as the most effective, AP gives a huge advantage to the operator. The operator is easily

re-chosen with the advantage. However, for only one operator, AP ignores the rest operators. In addition difference among the rests doesn't have any meaning; AP decreases application rates of the rests with equal proportion. For instance, when there are three operators; OP_1, OP_2 and OP_3, and their corresponding impacts are 9, 10 and 1 by order. Then, OP_2 is chosen as the most effective one and there is no difference between OP_1 and OP_3.

3.3.3 Multi-Armed Bandits

Comparing to PM and AP, Multi-armed bandit (MAB, DaCosta et al. (2008); Fialho and Schoenauer (2009)) has a different style. MAB is based on Upper Confidence Bound (UCB, Auer et al. (2002)) algorithm. The main feature of MAB is that it chooses the optimem operator based on a balance between exploration and exploitation (Eq. 3.4).

$$\begin{aligned} \hat{I_i}(t) &= \frac{1}{t}\sum_{t'=1}^{t} I_i(t') \\ \text{UCB}_i(t) &= \hat{I_i}(t) + C\sqrt{\frac{\log\sum_{o'=1}^{K} n_{o'}(t)}{n_i(t)}} \\ i^* &= \text{argmax}\{UCB_i, i = 1...K\} \end{aligned} \tag{3.4}$$

The exploitation term calculates the average impact of operator $\hat{I_{i,t}}$, up to time t, while the exploration term $n_{i,t}$ measures how often the operator is selected. A scaling factor C is needed to balance the two terms, because the impact range is unknown a priori.

The basic MAB, which is called static MAB (S-MAB), computes the average impact $\hat{I_{i,t}}$ over the whole period of evolution. This average impact $\hat{I_{i,t}}$ represents the performance of corresponding operator, as the exploitation term. Otherwise, the exploration term measures the number which the operators is selected, then S-MAB

gives more chance to less-selected operator. Because S-MAB uses the average impact over whole run, it is stable and smoothly changed, however S-MAB is sometimes weak in dramatically changing situation (DaCosta et al. (2008)).

Another style, dynamic MAB (D-MAB), uses the Page-Hinckley test (PH test, Hinkley (1970)), with parameters δ and λ (Eq. 3.5). PH test is used to determine when to reset the MAB log; the average operator impact, the number of selected and internal value of PH test are set to 0. The reset function by PH test makes D-MAB be more suitable to the dynamically changing situation than S-MAB.

$$\begin{aligned} \hat{I_i}(t) &= \frac{1}{t}\sum_{t'=1}^{t} I_i(t') \\ m_t &= \sum_{t'=1}^{t}(I_i(t') - \hat{I_i}(t) + \delta) \\ M_t &= max\{m_{t'}, t' = 1...t\} \\ PH_t &= M_t - m_t \\ Return & \quad (PH_t > \lambda) \end{aligned} \tag{3.5}$$

However, D-MAB still has a problem in its parameters; a scaling factor C and parameters for PH test δ and λ. When MAB uses the direct value of the fitness, scaling factor C has two different role; for the scale of the fitness and for the balance between exploitation and exploration. Thus, C becomes very sensitive parameter. In addition δ and λ are also sensitive and dependent on the impact value Fialho et al. (2010). Two variants; Sum of Ranked-Bandit (SR-B) and AUC-Bandit (AUC-B) (Fialho et al. (2010)) use the comparison-based impact for overcome the parameter sensitivity issue. Two algorithms do not use the direct impact value, they use the rank of impacts. SR-B, as its name says, uses the sum of the ranks of the impacts, which is normalized by the sum of all ranks (Eq. 3.6).

$$SR_{i,t} = \frac{\sum_{OP_r=i} D^r(W-r)}{\sum_{r=1}^{W} D^r(W-r)} \tag{3.6}$$

On the other hands, AUC-B uses the rank with a different way based on the Area Under the ROC Curve (AUC, Bradley (1997)) algorithm. The Receiver Operating Characteristic (ROC) curve, is originally used in signal detection theory, illustrates the performance of a binary classifier system. This algorithm draws the ROC curve for each operator, with the rank-based sorted list. Then, it uses the size of area under the ROC curve, as the operator impact.

Chapter 4

Preliminary Experiment for Adaptive Operator Mechanism

Probability matching, adaptive pursuit and multi-armed bandits are good AOS methods in genetic algorithm. In this chapter, as the first step of the research, we tried to direct-apply these AOSs to GP. We used TAG3P for this experiment, hence TAG3P has many and various genetic operators which have a variety of effects to individuals (Hoai et al. (2006)).

4.1 Test Problems

The 14 problems are used for these experiments. They fall into two categories: 10 symbolic regression problems and 4 target-structure problems.

The 10 symbolic regression problems were the regular symbolic regression problems ($F_n : n \in \{4, \ldots, 9\}$), Quintic ($Q$) and Sextic ($S$) problem, Trigonometric (T) problem and Two Boxes ($2B$) problem defined in table 4.1. They are all from well-known problem families proposed by Koza (Koza (1992, 1994)). Following Koza, we

Table 4.1: Problem Definitions – Symbolic Regression Problems

Problem	$F_n : n = 4, ..., 9$	Quintic	Sextic
Objective	Minimize MAE of cases		
Cases	20 Random Points from[-1,+1]	50 Random Points from[-1,+1]	
Target	$F_n = \Sigma_{i=1}^{n} x^i$	$x^5 - 2x^3 + x$	$x^6 - 2x^4 + x$
Fitness	Sum of absolute errors of fitness cases		
Atoms	X		
Success Predicate	Error $< \epsilon$ on all fitness cases		
Error Bound (ϵ)	0.1		
Functions	+, -, $\times$, $\div$, sin, cos, exp, log		

Problem	Trigonometirc	2-Box
Objective	Minimize MAE of cases	
Cases	20 Random Points from[0,2π]	10 Random Integer Points from$[1, 10]^6$
Target	$\cos 2x$	$WHL - whl$
Fitness	Sum of absolute errors of fitness cases	
Atoms	X, 1	W, H, L, w, h, l
Success Predicate	Error $< \epsilon$ on all fitness cases	
Error Bound (ϵ)	0.1	
Functions	+, -, $\times$, $\div$, sin	+, -, $\times$, $\div$

use the mean absolute error (MAE) as the fitness function. A solution is declared a success when its absolute error is less than ϵ for all fitness cases.

Target-structure problems differ from symbolic regression problems in that the fitness function, and the required solution is defined more directly in terms of properties of the solution expression tree. We used the Majority and Order (Table 4.2,

Table 4.2: Problem Definitions – Target-Structure Problems

Problem	Majority	Order
Objective	$\|P_i\| > \|N_i\|$ for all $i \leq n$	Some P_i is earlier than any N_i in preorder traversal, for all $i \leq n$
Targets	$n = 25, 30$	
Fitness	Number of i which satisfy the Objective	
Atoms	$P_1, P_2, ..., P_n; N_1, ..., N_n$	
Functions	-	
Success Predicate	n fitness number	

Goldberg and O'Reilly (1998); O'Reilly and Goldberg (1998)).[1]

The target property of the Majority Problem of size n (M_n) is that for each $i \leq n$, the number of nodes containing P_i is larger than the number containing N_i. For Order of size n (O_n), in a preorder traversal, for each $i \leq n$, at least one node P_i is encountered before any node N_i. In this paper, we use problems M_{25}, M_{30}, O_{25} and O_{30}.

4.2 Experimental Design

In this experiment, we used GP systems. With comparing to GA, GP has more complex chromosome structure and GP has more various genetic operators than GA. Thus it seems that AOS is more useful on GP system. Actually, previous AOS researches on GA treated only many variants of crossover and mutation. For example, five genetic operators; four crossovers and one mutation are used in Fialho

[1]These problems are more usually expressed in terms of an operator 'JOIN'; for economy of expression, we have re-named this operator '-'.

et al. (2009). By the way, the main genetic operators of GP are also crossover and mutation. Thus it is not much useful to apply Adaptive Mechanism to the standard GP. For this reason, we paid attention to TAG3P, which has many and various genetic operators. To apply Adaptive Mechanism to GP, we used two kinds of TAG3P systems. One is Linear-TAG3P (LTAG3P) which is a simple version of TAG3P, and the other is normal TAG3P.

The former use a linear form of elementary trees of which All nodes couldn't have more than one child node. Instead of LTAG3P do not permit to have multiple children node, LTAG3P has an encapsulated node which implies information for the fitness evaluation, and it makes LTAG3P can describe the same solution space to TAG3P's. However, cause of it has to interpret encapsulated nodes, it takes more time for the fitness evaluation. Moreover, even it is able to cover the same solution space, it has a huge bias when it is extent to the larger tree. For this limitation of LTAG3P, LTAG3P used not all test problems, it ran only for F_6, F_9, Q and S for this experiment. And we applied PM and AP to LTAG3P for comparing a normal LTAG3P. Moreover, LTAG3P lose a tree characteristic, so it can't use all TAG3P operators. In this experiment, LTAG3P used only 3 genetic operators; crossover, mutation and reproduction.

On the other hands, the latter is a standard TAG3P which has many and various genetic operators, as described in 2.1.3. While LTAG3P was restricted at genetic operators and test problems, TAG3P ran with all 7 operators for 14 problems.

4.2.1 Search Space

Figure 4.1 shows the elementary trees that we used for F_n, Q and S problems. The upper is for LTAG3P and the lower is for TAG3P. For the T and $2B$ problems, trees β_9-β_{12} were omitted, and the symbol 'X' was replaced by a lexicon – for T consisting

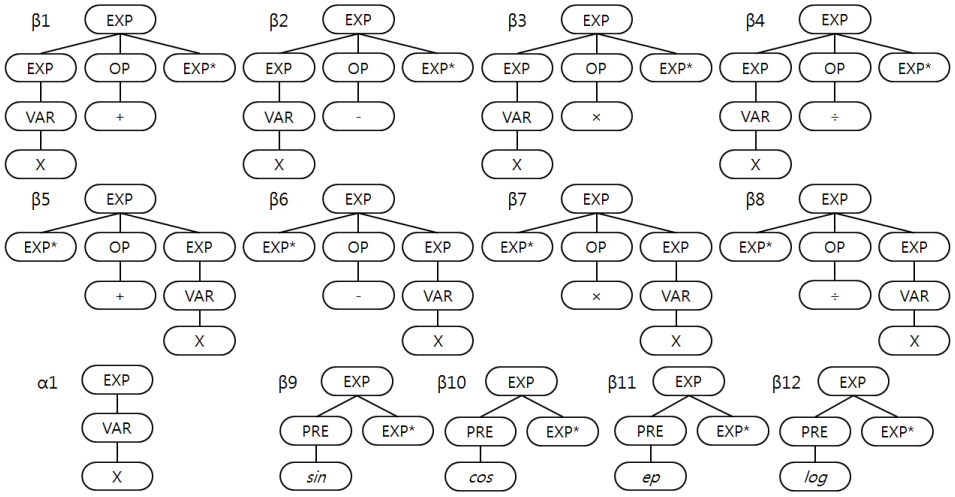

Figure 4.1: Elementary Trees for LTAG3P and TAG3P

of $\{1, X\}$, and for $2B$ of $\{W, D, L, w, d, l\}$. For the target-structure problems, the only β trees used were β_2 and β_6; for the M_n and O_n problems, the symbol 'X' was replaced by the lexicon $\{P_1, \ldots, P_n, N_1, \ldots, N_n\}$. In all cases, these are equivalent to CFGs, and we used the corresponding CFG grammars for GGGP. Again, in all cases these are equivalent to the full expression set over the functions and atoms of the specific language. Thus all systems were exploring essentially the same search spaces ('essentially' because the effects of depth/size limits may differ slightly depending on the system).

Other figures for the grammar (elementary tree) for each problem are in appendix.

4.2.2 General Parameter Settings

The evolutionary settings are as in table 4.3.

Of course, we need to additionally specify parameters for AOS, as in table 4.4.

Table 4.3: Setting for General Evolutionary Parameters (Preliminary Experiment)

Parameter	Value	Parameter	Value
Runs	100	Elite	None
Population	500	Tournament	Size 3
Generation	50		
Individual Size Range			
Symbolic Regression	2 . . . 40	Target-Structure	2 . . . 1000

Table 4.4: Setting for Adaptive Mechanism Parameters
(K denotes the number of operators)

	Parameter	Value
PM	Initial Rate P_{init}	$1/K$
&	Min. Rate P_{min}	$1/4K$
AP	α	0.8
AP	Max. Rate P_{max}	$1-(K-1)\cdot P_{\text{min}}$
	β	0.8
	δ	0.15
MAB	λ	0.5
	Scalar Factor C	0.5

Finally, we used ratio between fitness values of child and corresponding parent as the impact of each operator. For focusing the elite individuals and avoiding extremely large fitness value, we used 30% elite individuals.

Table 4.5: Success Proportion for Symbolic Regression Problems
on Preliminary Experiment (LTAG3P/TAG3P)

	LTAG3P			TAG3P			
	w/o AOS	PM	AP	w/o AOS	PM	AP	MAB
F_4	-%	-%	-%	96%	95%	94%	99%
F_5	-%	-%	-%	83%	92%	91%	82%
F_6	37%	45%	49%	52%	50%	57%	47%
F_7	-%	-%	-%	48%	41%	40%	33%
F_8	-%	-%	-%	16%	19%	18%	18%
F_9	13%	13%	19%	17%	17%	19%	10%
Q	45%	62%	63%	71%	72%	81%	64%
S	30%	29%	62%	95%	94%	96%	96%
T	-%	-%	-%	76%	79%	66%	67%
$2B$	-%	-%	-%	28%	29%	28%	21%

4.3 Results and Discussion

Table 4.5 shows the proportion of success on 4 and 10 symbolic regression problems for LTAG3P and TAG3P systems. In most cases, we can see AOS mechanisms worked well in both GP systems, except MAB. As the reason why MAB didn't work well, we could consider the sensitiveness of MAB parameters. This sensitiveness is already mentioned at 3.3.3. For this sensitiveness, it is not easy to find good parameter values in MAB algorithm. Otherwise, other two algorithms showed better performance than Normals, in particular, in LTAG3P. In LTAG3P, AOS algorithms are better than LTAG3P without AOS mechanism for all run-problems. Specially, AP performed quite better for S problem. For F_5 and Q, AP performed about 10% more than

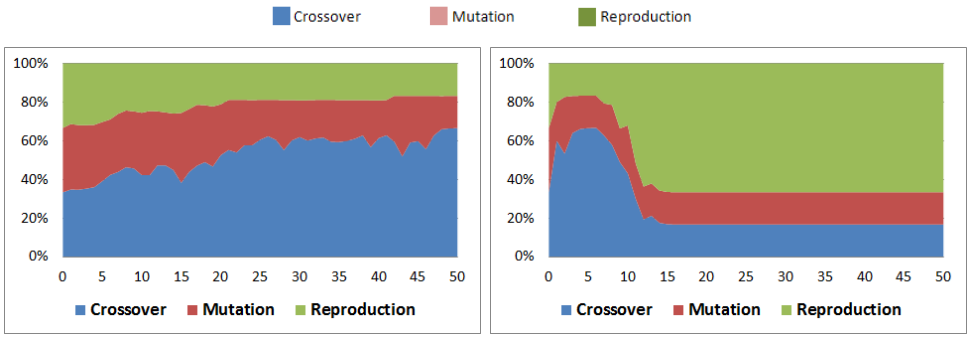

Figure 4.2: Change in Operator Application Rates for F_9, LTAG3P.
Left: PM, Right: AP

TAG3P without AOS. PM was also good. Comparing to AP, however, the detail performance of PM is similar or little worse than. By the way, in TAG3P, even AOS mechanism better performed over most problems, TAG3P without AOS is sometimes better. For example, for F_4 and F_7, no AOS algorithm couldn't show better performance than TAG3P without AOS.

Table 4.6: Success Proportion for Target-Structure Problems
on Preliminary Experiment (TAG3P)

	TAG3P			
	Normal	PM	AP	MAB
M_{25}	19%	19%	35%	10%
M_{30}	5%	3%	17%	3%
O_{25}	67%	76%	77%	59%
O_{30}	47%	51%	58%	30%

Following the same overall layout, table 4.6 shows the success rate on the target-structure problems. As MAB still didn't performed well, AP showed definitely better

performance than others, and PM and Normal followed AP by order.

In addition to observing the performance of the adaptation algorithms, it is important to see their overall effect; what operator rates do they actually select? Figure 4.2 shows the evolution of these rates of LTAG3P for the F_{18} problem. Space precludes showing such plots for all problems, but inspection of them shows that all problems and treatments may be divided into three regions: up to generation 5, in which the crossover rate either rises or stays steady and the mutation rate either stays steady or falls; generations 5 to 20, during which the crossover rate may remain steady for some time or drop (sometimes precipitously) and the mutation rate generally stays fairly steady; and generations 20 to 50, during which the crossover rate either stabilizes or continues to fall, and the mutation rate generally stays fairly steady. Thus we may characterize the behavior by observing the ratios at generations 5, 20 and 50, which we show for all problems.

Figure 4.4 shows the evolution of these rates of TAG3P for the a problem. We could check the change of operator rates, too. Crossover still shows overwhelming portion for all problems, the second and belows are different for problems. Mutation and duplication rates have large portions in symbolic regression problems, mutation and insertion are good for majority problems, and duplication/truncation is good for order problems. Relocation didn't show any effect for majority, as it can't affect to the fitness evaluation for Majority problem.

More figures for the rates change are in appendix.

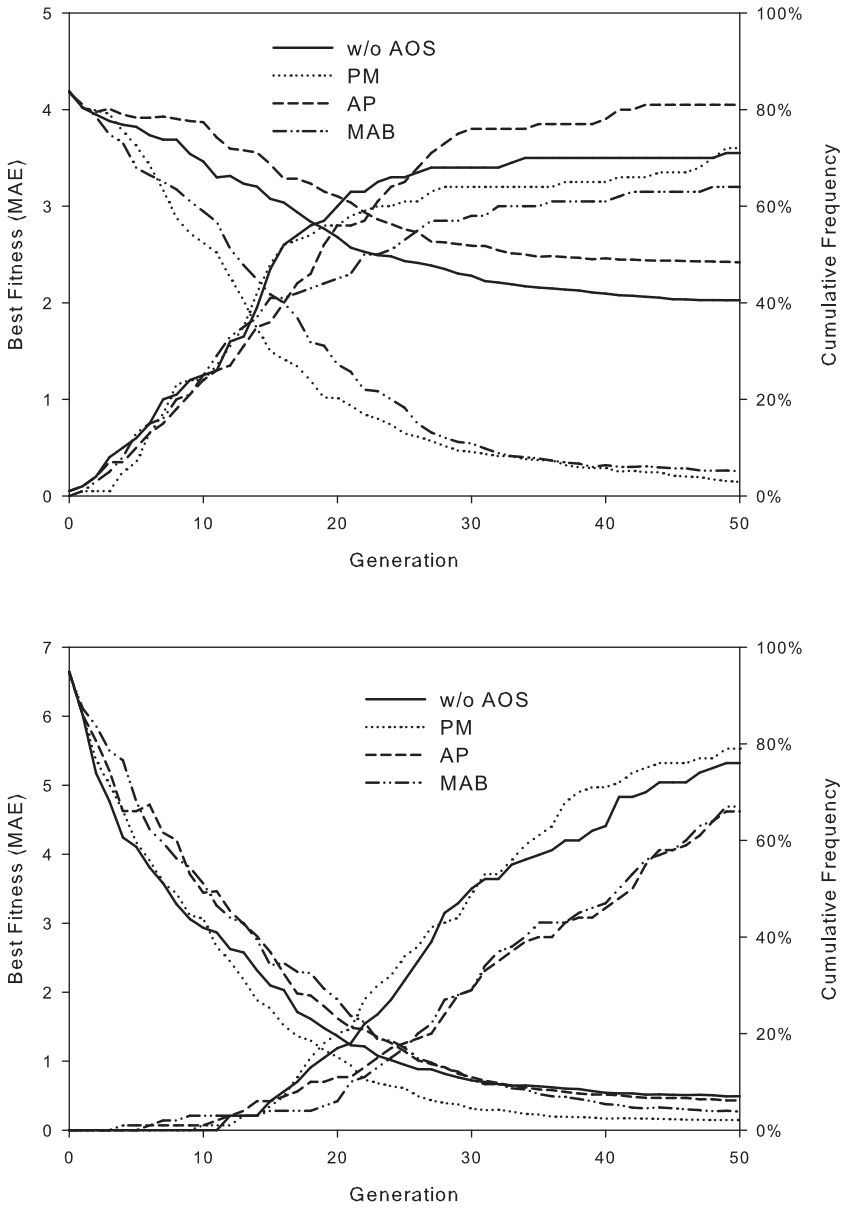

Figure 4.3: Mean of Best FitnessTop: Quintic, Bottom: Trigonometric

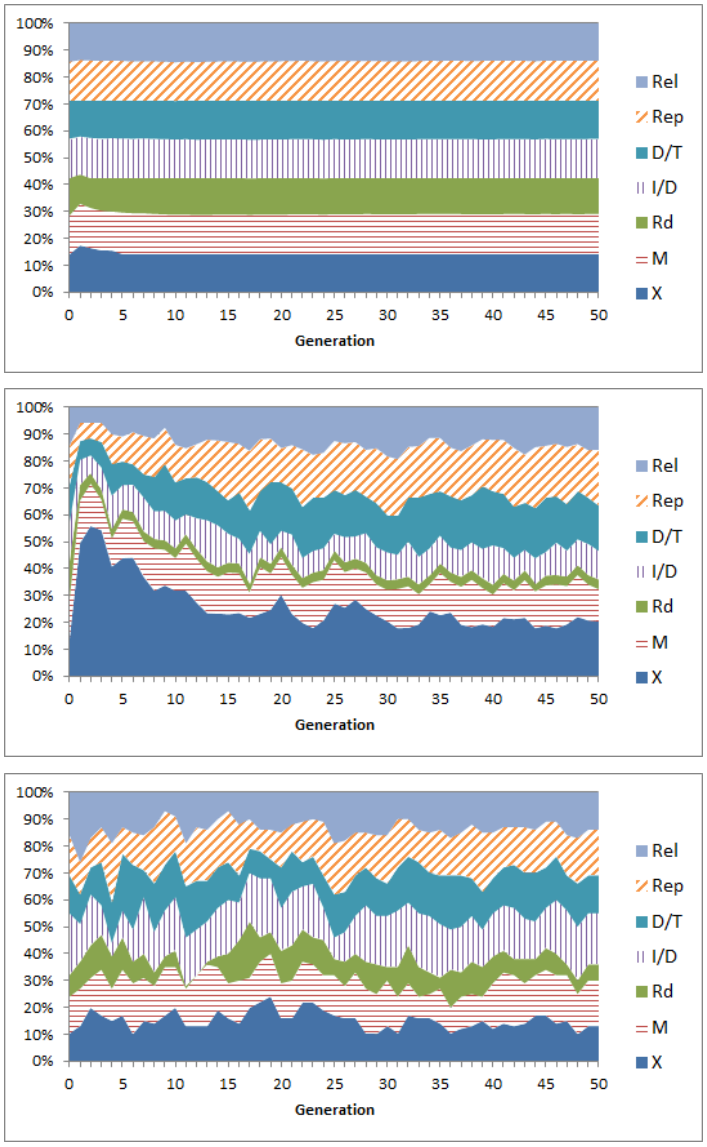

Figure 4.4: Change in Operator Application Rates for Trigonometric
Top: PM, Middle: AP, Bottom: MAB

Chapter 5

Operator Selection

The operator selection is one of the main parts of AOS mechanism. With given impacts of operators and the corresponding internal status, it suggests new operator rates. In chapter 4, we successfully applied adaptive operator selection to genetic programming, however it showed some restrictions for multiple and highly diverse operators. Chapter 5 introduces three operator selection methods; powered probability matching, adaptive probability matching and recursive adaptive pursuit. They are variants of probability matching and adaptive pursuit, which are designed to make operators to be more distinguished. They have the same form of evaluation of operator impact but use a different method of operator selection.

5.1 Operator Selection Algorithms for GP

5.1.1 Powered Probability Matching

Powered Probability Matching (PPM, Kim et al. (2012a)) is a variant of PM which more widely spreads operator probabilities. When PM sets operator probabilities, PM uses the quality rates. It mights help to directly reflect the trend of operators,

however, PM may not work well when operator impacts are very similar, as often occurs when there are many operators. For avoiding this problem, PPM amplifies the differences through exponentiation (Eq. 5.1).

$$P_i(t+1) \quad = \quad P_{min} + (1 - K \cdot P_{min}) \frac{Q_i(t)^K}{\sum_{j=1}^{K} Q_j(t)^K} \tag{5.1}$$

The basic process of PPM is exactly same to PM; how the operators are selected and how the quality values are updated through the impacts. The only difference is in operator selection part; how the algorithm sets the operator probabilities. As equation 5.1, PPM uses the quality rates, with applying K exponentiation to the value, for extending the difference without loss of trend of operators. Moreover, by using K as exponent number, the difference will be increased as the number of operators is increased. Table 5.1 is the detail algorithm of PPM.

Table 5.1: Algorithm for Powered Probability Matching

ProbabilityMatching(P,Q,I,K,P_{min},α)
for $i \leftarrow 1$ **to** K **do**
  $P_{init}(i) \leftarrow \frac{1}{K}$
  $Q_{init}(i) \leftarrow 1.0$
end for
while NotTerminated?() **do**
  OperatorSelectionBy$P(t)$
  $I_i(t) \leftarrow$ GetImpact
  $Q_i(t+1) \leftarrow Q_i(t) + \alpha(I_i(t) - Q_i(t))$
  for $i \leftarrow 1$ **to** K **do**
    $P_i(t+1) \leftarrow P_{min} + (1 - K.P_{min}) \frac{Q_i(t)^K}{\sum_{j=1}^{K} Q_j(t)^K}$
  end for
end while

5.1.2 Adaptive Probability Matching

Adaptive Probability Matching (APM, Kim et al. (2012a)) is a algorithm which combines two algorithms; AP and PM. AP divides operators into two groups. One is the most effective operator, and the other is a group of rest operators. AP increases the former's rate, but it decreases all the others' rates equally. In the other words, AP concentrates the only one, and ignores the relative impacts of the other operators. However, we found the result that the performance is changed by the second influential operator even the first influential operator is same to Hoai (2004). APM is made to distinguish the other operators, with applying PM algorithm partially.

APM follows AP in increasing the rate of the most effective operator as in AP. However, it then divides the remaining operator rate amongst the other operators according to their relative quality value, as in PM (Eq. 5.2).

$$\begin{aligned} i^* &= argmax\{Q_i, i = 1...K\} \\ P_{i^*}(t+1) &= P_{i^*}(t) + \beta(P_{max} - P_{i^*}(t)) \\ P_i(t+1) &= P_{min} + (1 - P_{a^*}(t+1) - (K-1) \cdot P_{min}) \frac{Q_i(t)}{\sum_{j=1, j \neq a}^{K} Q_j(t)} \\ &\qquad \text{for} \quad i \neq i^* \end{aligned} \tag{5.2}$$

Table 5.2 is the detail algorithm of APM.

5.1.3 Recursive Adaptive Pursuit

Recursive Adaptive Pursuit (r-AP, Kim et al. (2012c)) is another style of AP variants for giving differences among non-best effective operators. While APM partially applies PM algorithm, r-AP uses AP iteratively to distinguish the rests. r-AP also follows the AP process format. So its process is same until the update of quality vector. AP finds just one the most effective operator however r-AP otherwise sorts all

Table 5.2: Algorithm for Adaptive Probability Matching

AdaptivePursuit(P,Q,I,K,P_{min},P_{max},α,β)
$P_{max} \leftarrow 1 - (K-1)P_{min}$
for $i \leftarrow 1$ **to** K **do**
  $P(i) \leftarrow \frac{1}{K}$
  $Q(i) \leftarrow 1.0$
end for
while NotTerminated?() **do**
  OperatorSelectionBy$P(t)$
  $I_i(t) \leftarrow$ GetImapct
  $Q_i(t+1) \leftarrow Q_i(t) + \alpha(I_i(t) - Q_i(t))$
  $i^* \leftarrow \text{argmax}(Q_i(t+1))$
  $P_{i^*}(t+1) \leftarrow P_{i^*}(t) + \beta(P_{max} - P_{i^*}(t))$
  for $i \leftarrow 1$ **to** K **do**
    if $i \neq i^*$ **then**
      $P_i(t+1) \leftarrow P_{min} + (1 - p_{a^*}(t+1) - (K-1).P_{min}) \frac{Q_i(t)^K}{\sum_{j=1}^{K} Q_j(t)^K}$
    end if
  end for
end while

operators with the order of the effectiveness. Then, r-AP applies pursuit algorithm in order, with considering the minimum probability P_{min}. Table 5.3 is the detail algorithm of r-AP.

Table 5.3: Algorithm for Recursive Adaptive Pursuit

AdaptivePursuit(P,Q,I,K,P_{min},P_{max},α,β)
$P_{max} \leftarrow 1 - (K-1)P_{min}$
for $i \leftarrow 1$ **to** K **do**
  $P(i) \leftarrow \frac{1}{K}$
  $Q(i) \leftarrow 1.0$
end for
while NotTerminated?() **do**
  OperatorSelectionBy$P(t)$
  $I_i(t) \leftarrow$ GetImpact
  $Q_i(t+1) \leftarrow Q_i(t) + \alpha(I_i(t) - Q_i(t))$
  for $n \leftarrow 1$ **to** K **do**
    if $n == K$ **then**
      $P_{i_n}(t+1) \leftarrow (1 - \sum_{x=1}^{n-1} P_{i_x})$
    else
      $i_n \leftarrow \text{argmax}(Q_i(t+1))$ for $i \neq i_1, ..., i_{n-1}$
      $P_{i_n}(t+1) \leftarrow (1 - \sum_{x=1}^{n-1} P_{i_x})(P_{i_n}(t) + \beta(P_{max} - P_{i_n}(t)))$
    end if
  end for
end while

5.2 Experiments and Results

5.2.1 Test Problems

To investigate adaptive operator mechanism in GP system, we used the 16 problems (Kim et al. (submitted)). 14 problems come from 4.1, and we added two target-structure problems. The 16 problems are used for these experiments fall into two

categories: 10 symbolic regression problems and 6 target-structure problems.

The outline of 10 symbolic regression problems and 4 target-structure problems are same to 4.1. Thus we describe only two additional target-structure problems.

Table 5.4: Problem Definitions – Target-Structure Problems (Daida's)

Problem	Daida (Narrow)	Daida (Wide)
Objective	Tree has target depth and number of atoms	
Targets	$D_t = 20$, $T_t = 20$	$D_t = 10$, $T_t = 320$
Fitness	$w_d \times (1 - \frac{\|d - D_t\|}{D_t}) + w_t \times (1 - \frac{\|t - T_t\|}{T_t})$	
Atoms	X	
Functions	-	
Success Predicate	100 Fitness Value	

Two additional target-structure problems are Daida problems (table 5.4, Daida et al. (2003)). The character of the Daida problem is rather different. There is only one kind of atom, X. The objective is to find a tree with the specified depth and number of atoms. In Daida et al. (2003), Daida demonstrated that the difficulty of the problem (for GP) varied in different regions of these values, being hardest for narrow or wide values, and easier for intermediate values. We chose two difficult (i.e. region III) settings: narrow (D_N) and wide (D_W).

5.2.2 Experimental Design

Five systems were used for comparison; three baseline and three AOS algorithms. Three baselines (called TAG3PC09 and TAG3PEQ) are used. TAG3PC09 uses only two most popular genetic operators; crossover and mutation. They reflect a traditional scenario for GP manner; high crossover rate and low mutation rate. We set two rates as 0.9 and 0.1. TAG3PEQ consists of TAG3P with all operators from

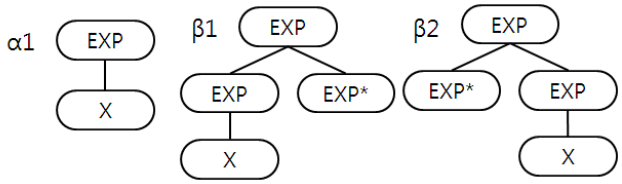

Figure 5.1: Elementary Trees

section 2.1.3, and all having the same operator rates (i.e. 14.29%). It assumes zero-knowledge status for TAG3P with multiple operators to a problem. Conversely, three AOS algorithms use PPM, APM and r-AP for contrast to non-adaptive treatments. The initial operator rates of all three algorithms are set the same as in TAG3PEQ, i.e. to 14.29%.

Search Space

Figure 5.1 shows the elementary trees (TAG grammar) that we used for D_n and D_w. Actually two nodes are tied with an operator JOIN, but we omitted the symbol 'JOIN' for easy calculation of size and depth.

General Parameters

The basic setting is same to 4.2, but the setting for Daida problem is different. The evolutionary settings are as in table 5.5.

PPM, APM and r-AP are variants of PM or AP, so we set their parameter settings as same to 4.2. And we used same impact evaluation to 4.2.

Table 5.5: Setting for General Evolutionary Parameters

Parameter	Value	Parameter	Value
Runs	100	Elite	None
Population	500	Tournament	Size 3
Generations			
Except D_x	50	D_x	1000
Individual Size Range			
Symbolic Regression	2 . . . 40	Target-Structure	2 . . . 1000

5.2.3 Results and Discussion

There is some debate in the GP literature about what is the best performance metric for distinguishing between different GP systems. However real-world GP applications may have different purposes (e.g. finding a good fit to data vs recovering an exact model when it is known that the data must be generated by a simple process, and that there is little noise). Thus different performance metrics are only to be expected. Rather than address this issue here, we provide a number of performance metrics (within the available space) to illuminate performance issues as well as we can.

Symbolic Regression Problems

Table 5.6 shows the proportion of success on ten symbolic regression problems, while figure 5.2 shows the mean best fitness and cumulative frequency of success curves for F_7 and $2B$ problems.

Even the 'zero knowledge' approach of just using all operators performed reasonably well, comparably to the performance of simply using subtree mutation throughout. This baseline performance gives some encouragement that operator rate adap-

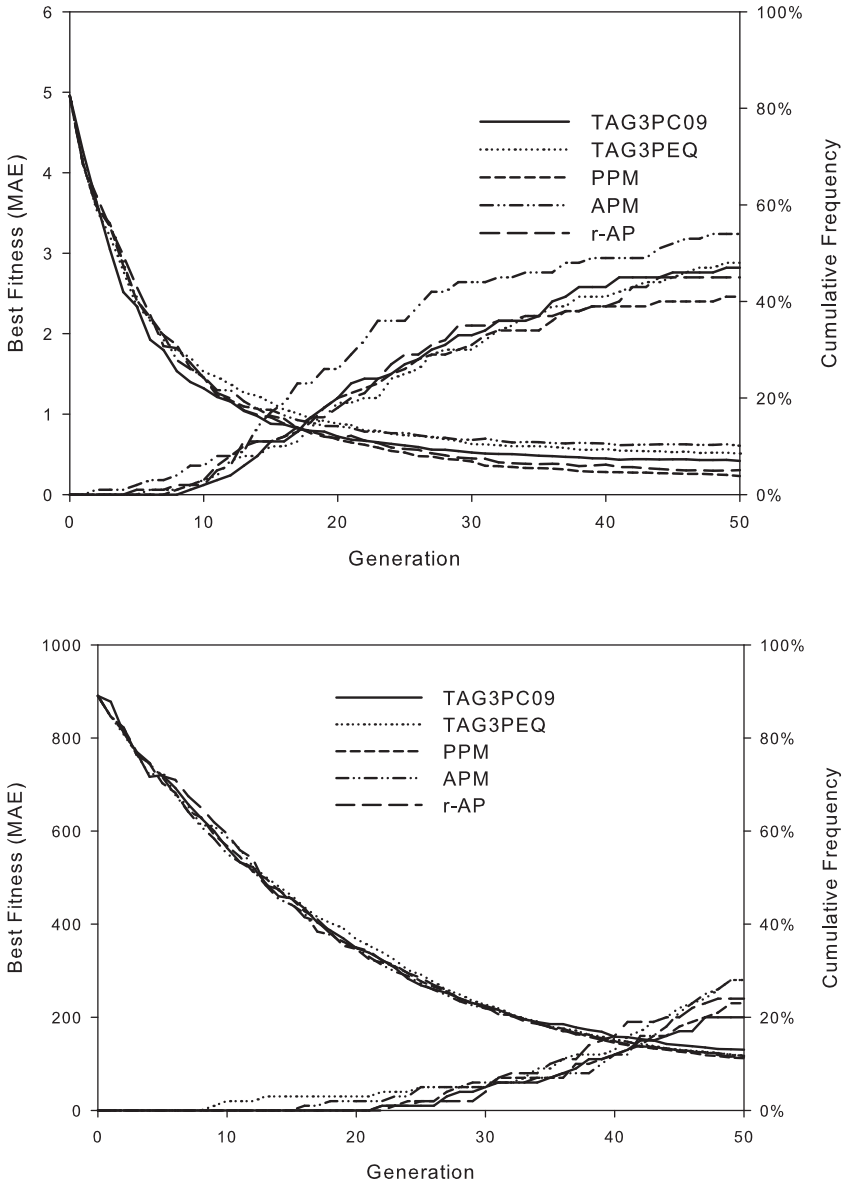

Figure 5.2: Mean of Best Fitness and Cumulative Frequencies for Symbolic Regression, Top:F_7, Bottom:$2B$

Table 5.6: Success Proportion for Symbolic Regression Problems:
(2 w/o AOSs and 3 AOSs)

	TAG3PC09	TAG3PEQ	PPM	APM	r-AP
F_4	94%	96%	95%	95%	94%
F_5	93%	83%	90%	96%	89%
F_6	54%	52%	56%	57%	60%
F_7	47%	48%	41%	54%	45%
F_8	19%	16%	18%	24%	25%
F_9	18%	17%	17%	16%	19%
Q	88%	71%	73%	84%	77%
S	96%	95%	97%	98%	99%
T	70%	76%	74%	72%	73%
$2B$	20%	28%	23%	28%	24%

tation might be able to improve performance – and in fact, this is what we see. The adaptive performance, particularly with APM, comes close to that of simply choosing the optimal mutation operator. Foreshadowing some of our results from Kim et al. (2012a), the benefit of choosing the optimal operator is, as we might expect, greatest in the cases where the choice of operator is clear, as with the Q problem; where the choice is less clear, as with the $2B$ problem, the benefits are correspondingly equivocal.

Target-Structure Problems

Following the same overall layout, table 5.7 shows the success rate on the target-structure problems, and figure 5.3 the mean best fitness cumulative success frequency curves for D_W problem. Overall, the value of adaptive mechanisms is clearer in these

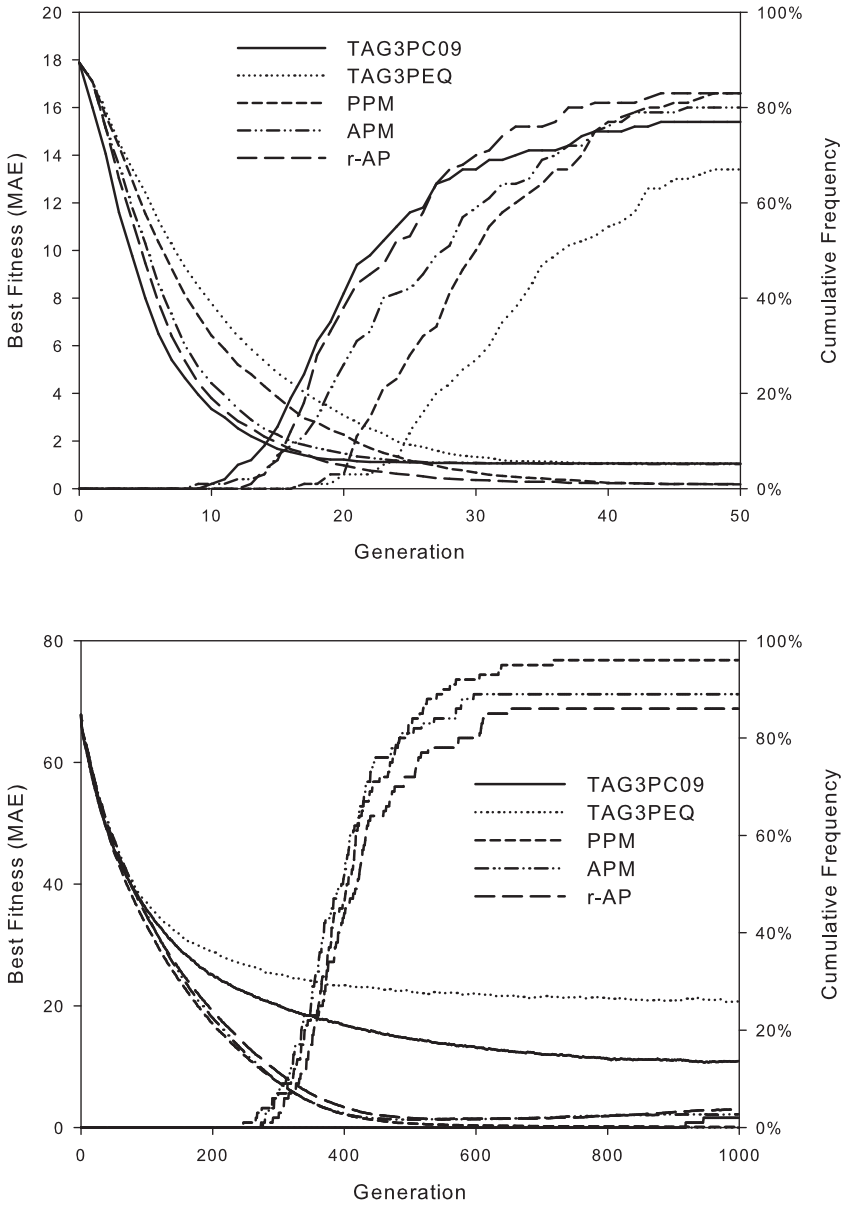

Figure 5.3: Mean of Best Fitness and Cumulative Frequencies for Target Structure Problems, Top:O_{25}, Bottom:D_W

Table 5.7: Success Proportion for Target-Structure Problems:
(2 w/o AOSs and 3 AOSs)

	TAG3PC09	TAG3PEQ	PPM	APM	r-AP
M_{25}	33%	19%	15%	32%	33%
M_{30}	12%	5%	6%	15%	14%
O_{25}	77%	67%	83%	80%	83%
O_{30}	59%	47%	51%	59%	55%
D_N	24%	88%	100%	98%	97%
D_W	2%	0%	96%	89%	86%

cases, with AP now performing better than APM. As a matter of fact, AP gives near-comparable performance to that of the best mutation operator – and in the case of the D_W problem, substantially better, suggesting that in this case, synergy between operators may be a key issue.

Interestingly, increasing the rate of I/D in the D_W problem (and correspondingly decreasing the crossover rate) appeared to have substantial benefits. This could be either because D/T are especially beneficial in this problem, so that increasing their rate of application is desirable – or because subtree crossover is particularly destructive in this problem, so that decreasing its rate is beneficial. The analysis of operator rates casts further light on this situation.

Overall, we see that combining multiple operators, with different suitability for different problems, with an operator rate adaptation mechanism can be an effective strategy for solving problems where it is unknown exactly which operator is most suitable. By examining the actual rates that adaptive mechanism chose at different stages of evolution, we can hope to gain further insight into the reasons for this performance.

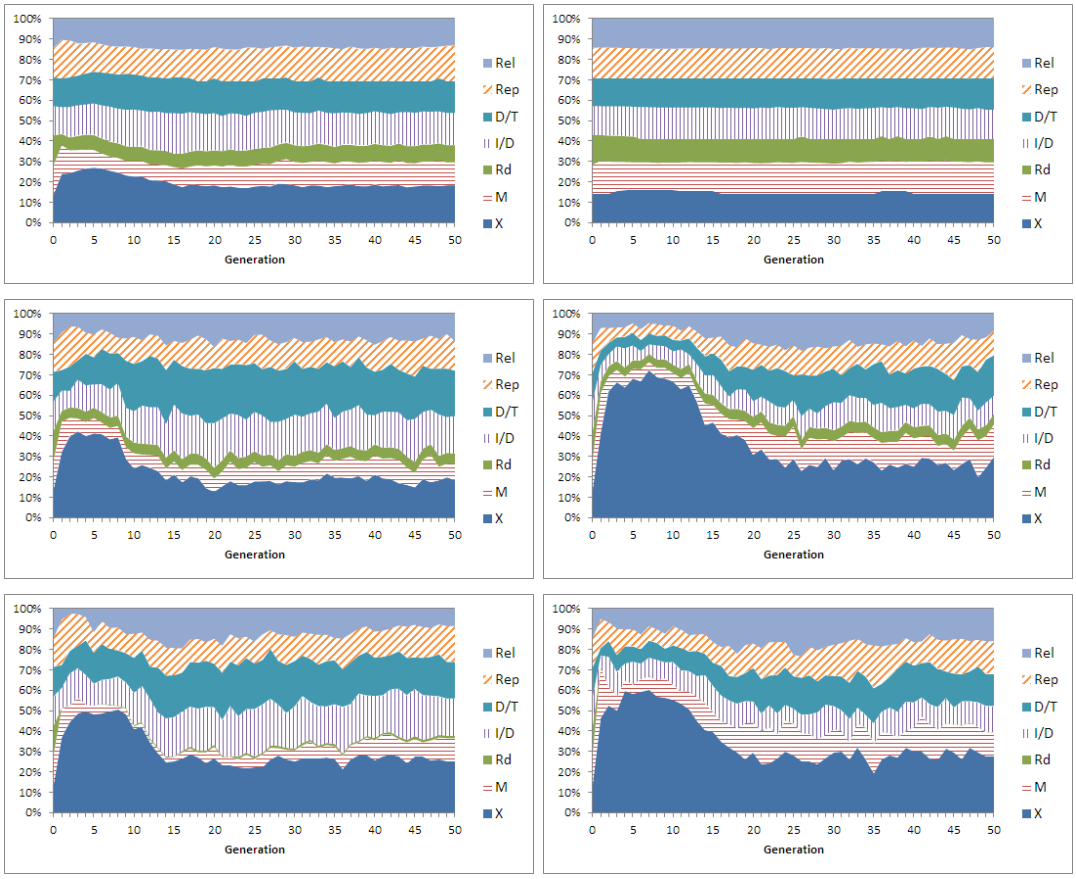

Figure 5.4: Change in Operator Application Rates

Left:F_7, Right:2-Box

Top:PPM, Middle:APM, Bottom:r-AP

Operator Adaptation: Overall Operator Rates

Figure 5.4 and 5.5 shows the changes in operator rates for some typical combinations of adaptive mechanism and problem instance. Leaving aside for one moment the highly anomalous case of Daida's problem, much of the overall behavior is what

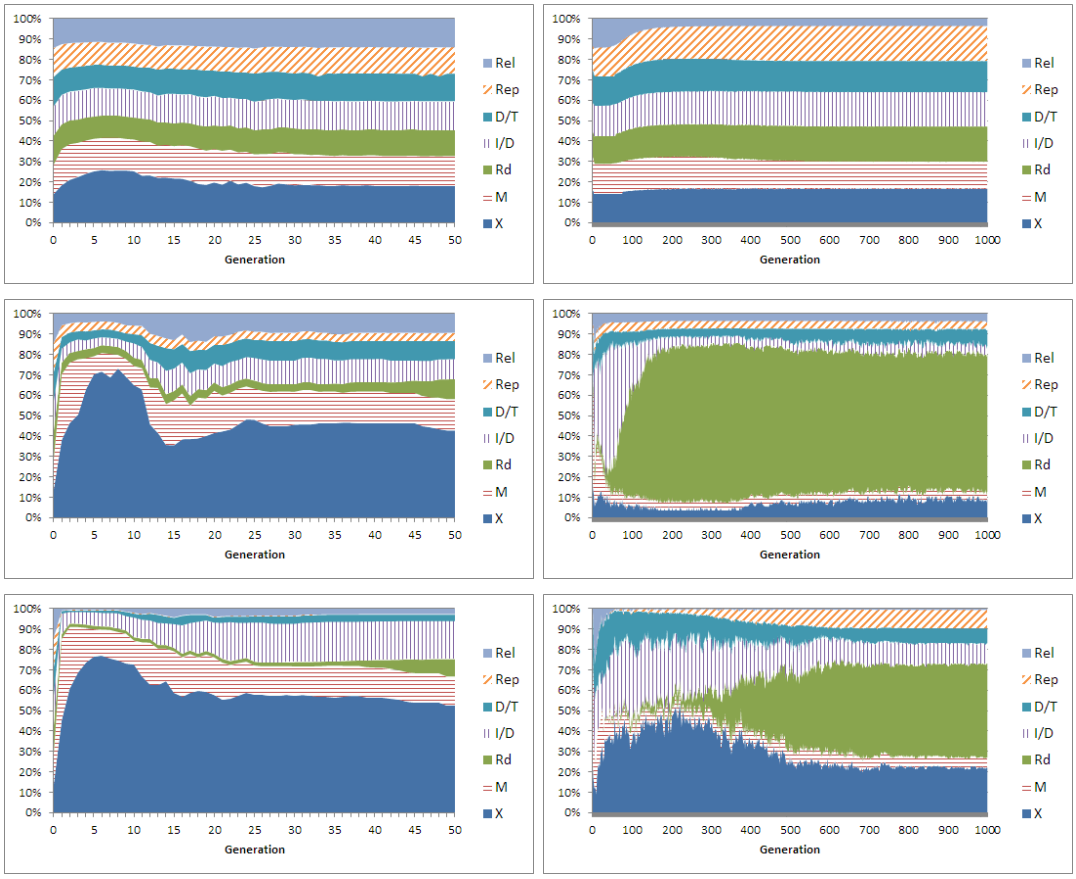

Figure 5.5: Change in Operator Application Rates
Left:O_{25}, Right:D_W
Top:PPM, Middle:APM, Bottom:r-AP

most GP practitioners would predict based on experience. Notably, in almost all experiments, subtree crossover is highly productive earlier in a run, so its rate rises, but peaks somewhere around 10 generations (the precise point depending on problem and adaptation mechanism), and falls away thereafter. This subsequent drop is consistent with typical discussion of the destructive effects of crossover. What is

perhaps more interesting is the behavior of the other operators, particularly after the crossover peak. Once crossover falls away, there is room for the other operators to expand. But which one depends heavily on the problem. What one can say in general is that reproduction is rarely competitive (implying that the other operators are at least somewhat productive throughout the runs). The results confirm that on some problems, one or other mutation operator dominates – duplication/truncation on the quintic problem, subtree mutation on the O_{30} problem, and to a lesser extent on M_{30} – but on others, such as the $2B$ problem, no one mutation operator dominates, and the important thing seems to be to keep the crossover rate relatively low, and the reproduction rate very low.

The D_W problem is completely different. In the early stages of the run, insertion/deletion is effective (an unsurprising result, given previous publications on this problem, Hoai et al. (2006)), but so is subtree mutation. On the other hand, crossover performs very poorly early on. It is not completely surprising that the insertion/deletion takes over from subtree mutation subsequently, given that finer scale tuning is likely to be required. Nor is it surprising, with the difficulty of this problem, that eventually the best thing to do is nothing (most changes damage the current solutions) so that reproduction comes to dominate. Perhaps a little more surprising is the gradual increase in crossover and duplication/truncation later in the run.

Operator Adaptation: Operator Rates in a Single Run

While figure 5.4 and 5.5 are operator rates over 100 runs, figure 5.6 and 5.7 shows the changes in operator rates over a single run. Figure 5.6 is a change of operator rates in F_{07} problem and figure 5.7 is in M_{30} problem. Overall, a change of tendencies of operator rates in both figures is matched to a change of fitness. For example, a sharp

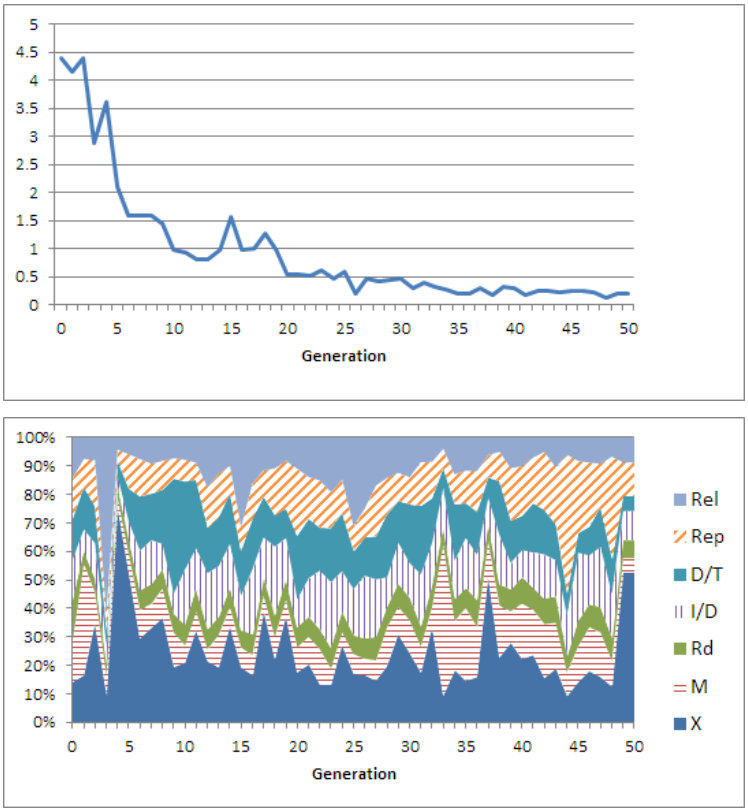

Figure 5.6: Change in Operator Application Rates in a Single Run for F_7 on APM

decrease of fitness value and increase of X are simultaneously happened at generation 5 in figure 5.6. From generation 20, when fitness curve starts to converged, M, I/D and *Rep* often have a large portion. Figure 5.7 also shows an similar interaction between fitness and operator rates. Before the fitness convergence (approx. generation 15), X has the largest rates, but others, in particular M, has more portion after the fitness convergence. Whenever fitness curves causes any change from the current status, we could usually find corresponding changes in operator application rates.

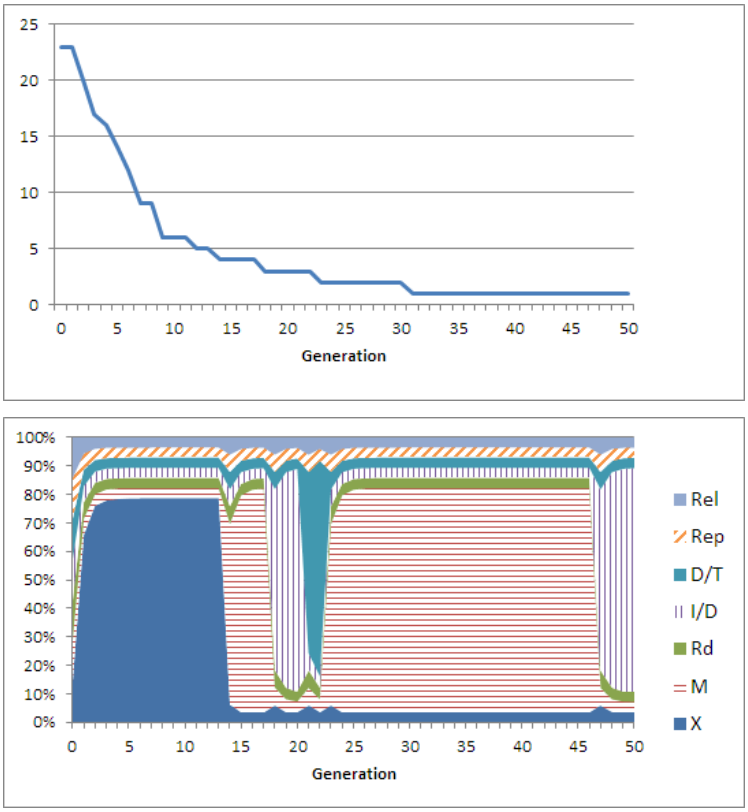

Figure 5.7: Change in Operator Application Rates in a Single Run for M_{30} on APM

Through the empirical analysis of the change in operator rates over whole run, we could generate a proper set of genetic operators for a given problem, whereas the analysis of single runs shows how genetic operators work in diverse situations of problems or the progress of a run. Even though runs of the same problem, the changes in operator rates in a single run showed different tendencies. However we could see the interaction between fitness and operator rates from an empirical analysis of single runs in adaptive operator mechanisms.

Chapter 6

Evaluation of Operator Impact

The evaluation of operator impact is also one of the main parts of AOS. The evaluation of operator impact defines the operator impact and updates the internal status with the impact, so that it provides base resources for the operator selection. Because the impact defined by the evaluation of operator impact becomes a measure of how much an operator affect a run, the evaluation of operator impact is a key issue for AOS.

For the evaluation of operator impact, several different methods have been proposed. The main issues on the evaluation of operator impact are summarized two things; resources for the impact of operator and methods for the impact be assigned to. As resources and methods, the most common method is the fitness improvement, which is brought by the newly generated child individual, when compared to the best individual (Davis (1989)), to the current median (Julstrom (1995)), or its parent (Thierens (2005); Fialho et al. (2009); Kim et al. (2012a)) in the population. Meanwhile, a relative fitness improvement is used in Gong et al. (2010), taking into account the difference of the fitness of the offspring with that of its parent, and normalizing it by the ratio between its fitness and the best one in the current

population. Davis (1989) and Julstrom (1995) have proposed to assign impact to the strategies that were used to generate the ancestors of the current individual, by means of a bucket brigade scheme. Two more recent approaches, targeted toward highly multi-modal problems, considered both fitness improvement and the variation of some diversity measure to design the impact of operators: aggregating them in a mechanism termed Compass (Maturana et al. (2009)), or treating the issue as a 2-objective problem, and using as an impact the Pareto Dominance score (Maturana et al. (2010)).

We tried to use two impact resources by using a variety of problems at chapter 5. In this chapter, we introduce several methods for the evaluation of operator impact.

6.1 Rates for the Amount of Individual Usage

6.1.1 Definition of Rates for the Amount of Individual Usage

We used 30% of elite individuals at the previous chapter 4 and 5. The reason why we used restricted amount of individuals is for avoiding the extremely large or small fitness value which may cause an overflow. Moreover, as EAs aim to find the optimal solution, to focus the elite individuals is useful to measure the impact of an operator for the optimal solution. We chose 30% from empirical experiments, but proper rate value may be different for problems. Meanwhile, as the progress of run goes to the end, more individuals in population are converged. Even the same rate is applied, variances of the elite individuals are different by the progress. For that reason we also considered a linearly changing percentage for choosing the elite individuals.

Table 6.1 describes five methods of rates for the amount of individual usage. Three methods use fixed rates and two methods have changing rates. We tried two change methods; $LC1$ and $LC2$. $LC1$ changes the rate from 10% to 50%, via the

Table 6.1: Definition of Five Rate Policies for the Amount of Individual Usage

Policy	Detail
Percentage 10 ($P10$)	Fitness ratio over 10% elite individuals
Percentage 30 ($P30$)	Fitness ratio over 30% elite individuals
Percentage 50 ($P50$)	Fitness ratio over 50% elite individuals
Linear Change #1 ($LC1$)	Linearly changed, from 10% to 50%
Linear Change #2 ($LC2$)	Linearly changed, from 50% to 10%

time (generation), and $LC2$ does reversely. $P10$, $P30$ and $P50$ are the base methods, and they used the fixed percentage values. We compared these five methods with the same problem sets to 4.1. And we also used the same settings to 4.2.

6.1.2 Results and Discussion

This experiment compares the range of elite individuals.

Table 6.2 is success proportion for all problems. There is no best strategy for all problems, however $P10$ and $LC1$ generally have not good performances and $P30$ and $LC2$ relatively show good performance.

Considering the feature of individual set of each method, $P10$ has the largest impact value and $P50$ has the smallest. However, the number of individual pairs, which $P10$ has, is the smallest, and $P50$ has the most pairs. In fact, bigger method includes smaller ones; $P50$ use individuals over 0 to 50%, and $P10$ use over 0 to 10%. Meanwhile, the value of individual pairs is decreased during a run. At the beginning of a run, it has many chances to improve much, so it is easy to have a big value. However, after some generations, only few individual pair gets a value more than 1. In other words, as run is processed, the impact value is decreased and the number

Table 6.2: Success Proportion: Individual Usage Rates

	AP P10	AP P30	AP P50	AP LC1	AP LC2
F_4	98%	94%	93%	96%	97%
F_5	87%	91%	90%	85%	89%
F_6	51%	57%	64%	54%	58%
F_7	39%	40%	49%	43%	53%
F_8	14%	18%	19%	11%	20%
F_9	13%	19%	15%	18%	13%
Q	74%	81%	76%	80%	83%
S	96%	96%	93%	96%	96%
T	72%	66%	76%	72%	67%
$2B$	28%	28%	26%	23%	22%
M_{25}	28%	35%	31%	35%	30%
M_{30}	11%	17%	14%	11%	17%
O_{25}	74%	77%	83%	78%	80%
O_{30}	46%	58%	69%	56%	69%
	APM P10	APM P30	APM P50	APM LC1	APM LC2
F_4	98%	94%	97%	94%	97%
F_5	93%	95%	87%	88%	90%
F_6	52%	52%	59%	54%	60%
F_7	42%	56%	44%	33%	51%
F_8	18%	23%	16%	16%	20%
F_9	11%	13%	18%	15%	12%
Q	85%	78%	72%	78%	80%
S	96%	96%	91%	95%	97%
T	75%	77%	74%	74%	85%
$2B$	28%	22%	23%	25%	27%
M_{25}	20%	34%	33%	26%	31%
M_{30}	13%	13%	14%	14%	9%
O_{25}	76%	82%	76%	80%	85%
O_{30}	45%	68%	71%	59%	67%

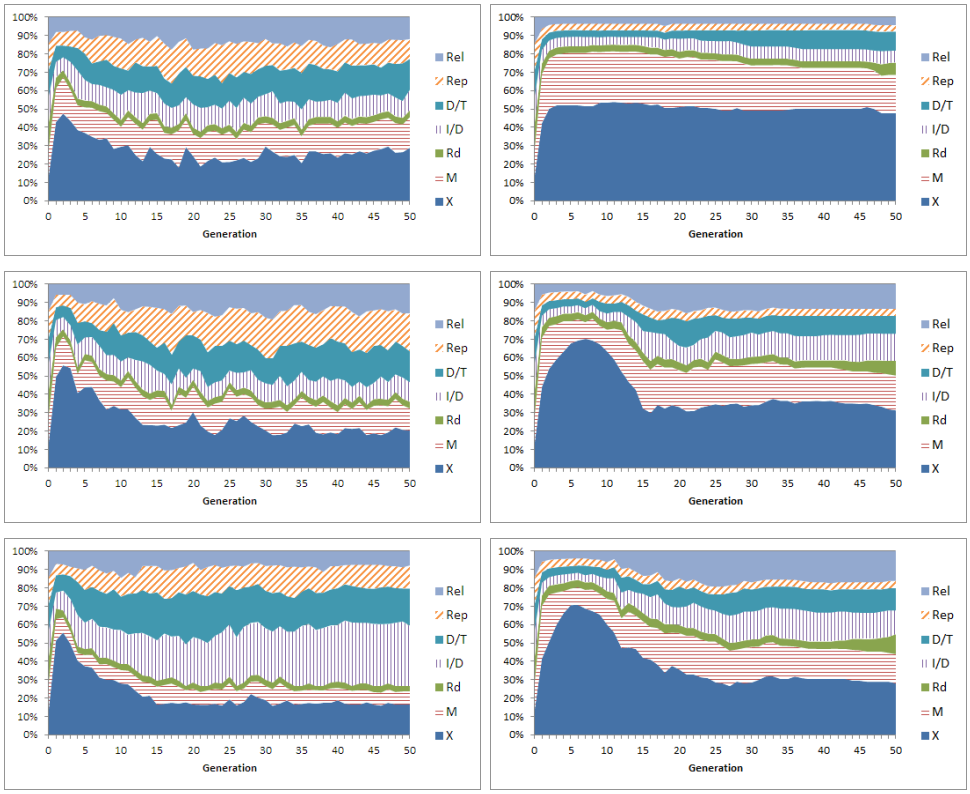

Figure 6.1: Change in Operator Application Rates
Left: Trigonometric on AP, Right: O_{30} on APM
from Top: $P10$, $P30$ and $P50$

of individual pairs which have more than 1 value, is also decreased.

These features are shown at figure 6.1, 6.2, 6.3 and 6.4. Figure 6.1 and 6.2 show the operator rate and figure 6.3 and 6.4 are its corresponding internal status. In these figures, $P10$, $P30$, $P50$, $LC1$ and $LC2$ show different features. From $P10$ to $P50$, we could find a sequential change of rates of some operators and values of internal status. $LC1$ and $LC2$ show features as mixtures; the beginning parts of $LC1$ and

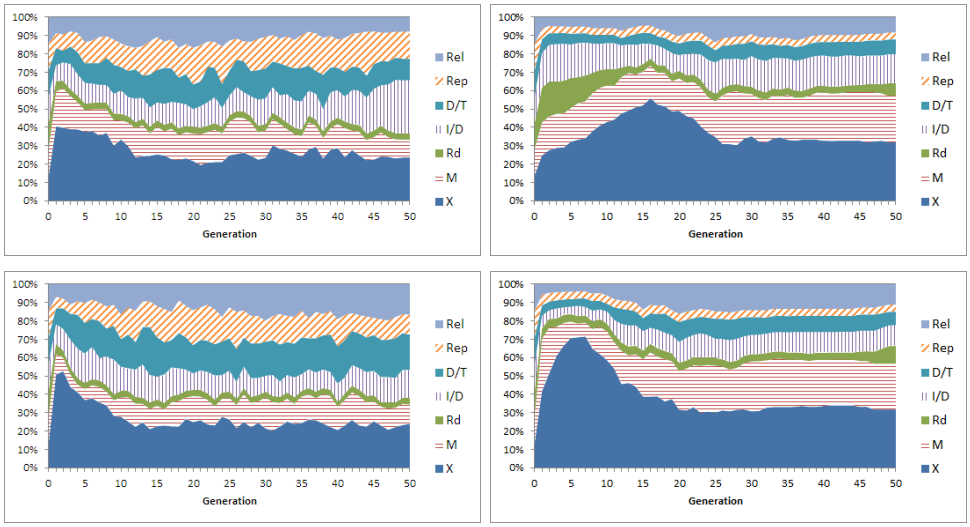

Figure 6.2: Change in Operator Application Rates
Left: Trigonometric on AP, Right: O_{30} on APM
from Top: $LC1$ and $LC2$

$LC2$ are similar to $P10$ and $P50$, but the last parts of both methods are similar to with reverse order.

In the figure 6.1 and 6.2, the notable point is the proportion of mutation (M) operator. The proportion of M is big in $P10$, but its scale is small in $P50$. Otherwise, the proportion of insertion/deletion (I/D) grows over $P10$ to $P50$. That is, only few individuals are improved and fitness values of the rest get worse by mutation. Meanwhile, insertion/deletion makes many individuals get better fitness values, but they improve individuals only slightly. Therefore, the proportion of I/D is small in $P10$ method. Crossover (X) has high rate value for all methods. In particular, it is better in $P30$ and $P50$ than $P10$. By this, we can guess X make many individuals be quite-better.

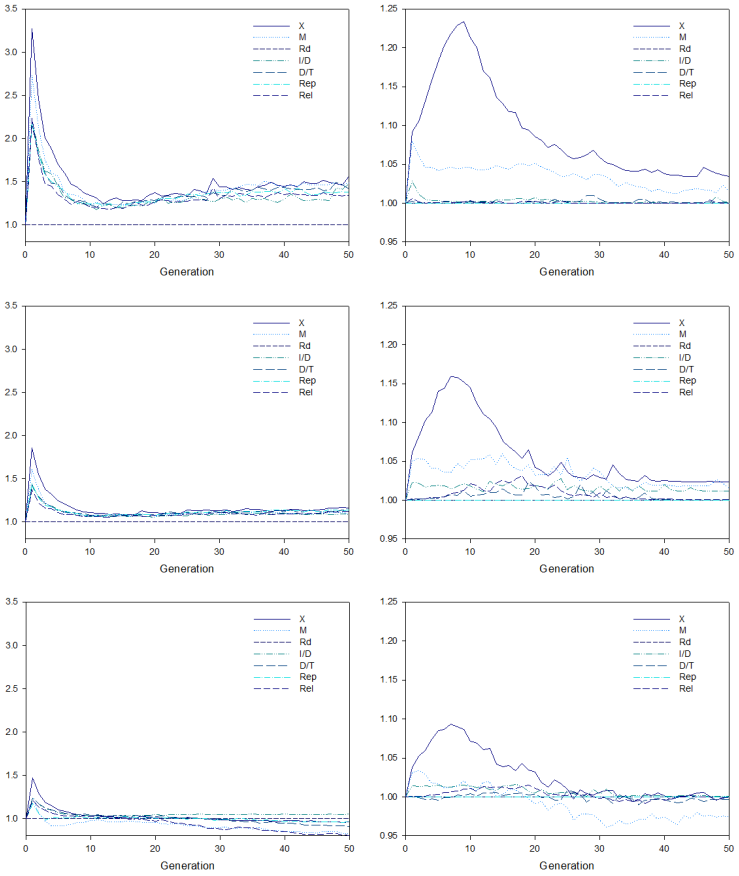

Figure 6.3: Change in Internal Status
Left: Trigonometric on AP, Right: O_{30} on APM
from Top: $P10$, $P30$ and $P50$

The tendency of figure 6.3 and 6.4 is same to our guess. $P10$ has the highest value and $P50$ has the lowest. Over whole run, all values in $P10$ are more than one. But some values in $P50$ are less than one and its proportion is grown over the run. So that reason, it is hard to get a good impact value for I/D in $P10$, and it is difficult to M in $P50$.

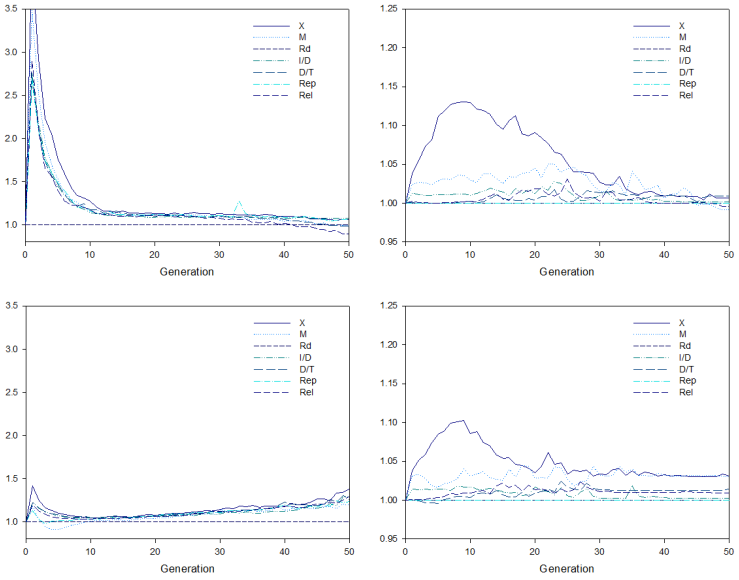

Figure 6.4: Change in Internal Status

Left: Trigonometric on AP, Right: O_{30} on APM

from Top: $LC1$ and $LC2$

6.2 Ratio for the Improvement of Fitness

Fitness improvement is a usual way to evaluate an operator impact. The basic concept of fitness improvement is to compare a newly generated child individual to the present population (e.g. corresponding parent individuals or the best individual). This section describes some methods of fitness improvement which is based on ratio of fitness of child and its corresponding parent individuals. A basic definition of fitness ratio is in equation 6.1.

$$Ratio = \frac{\bar{F}_p}{F_c} \tag{6.1}$$

where F_c is fitness of child individual and $\bar{F}_p$ is mean fitness of its corresponding parents. The number of parents is usually one, but we used a mean value of parents in case of child have more than two parents.[1] If there is no improvement from parents to child, ratio value become 1. When an improvement occurs, ratio value is bigger than 1, otherwise it is less than 1 in a environment of less fitness is better.

6.2.1 Pairs and Group

When the evaluation of operator impact evaluates an impact value, it usually uses more than one individuals like as section 6.1. That is, some child and its correspond parent pairs are used together for evaluating of operator impact. We suggest two methods which uses these pairs (Table 6.3).

Table 6.3: Definition of Two Fitness Improvement Methods

Method	Detail
Individual Pairs ($I.Pairs$)	Mean of $\frac{\bar{F}_p}{F_c}$ for each operator
Individual Groups ($I.Groups$)	(Mean of $\bar{F}_p$)/(Mean of F_c) for each operator

The unit of $I.Pairs$ is a pair of child and its corresponding parents, however $I.Groups$ divides them into two groups: children group and its corresponding parents group. While $I.Pairs$ measure the mean of ratio values of pairs, $I.Groups$ calculate the ratio of two mean values from parents group and children group. In other words, $I.Pairs$ checks average improvement over individuals, but $I.Groups$ estimates the improvement from the overall parents to the overall children.

[1] Crossover uses two parents

6.2.2 Ratio and Children Fitness

For avoiding an overflow and focusing the optimal solution, we used partial elite individuals at section 6.1. For every end of generations, we sorted individuals and chose elite individuals for evaluation. A key issue at this point is a sort key which is a criterion of the sort. Hence the fitness ratio is used for the evaluation, fitness ratio itself is a general sort key. However, only fitness of child individual is also a reasonable choice as a sort key, because the ultimate objective is to find the optimal solution which has a good fitness value. Although a fitness improvement is very large, bad child fitness valued individual is not necessary as the optimal. A newly suggested method is, to sort with child fitness order and to evaluate the impact with the fitness ratio (Table 6.4).

Table 6.4: Definition of Fitness Improvement with Two Sort Key

Method	Detail
Ratio Sort ($R.Sort$)	Mean of $\frac{\bar{F}_p}{F_c}$ in $\frac{\bar{F}_p}{F_c}$ order
Child Sort ($C.Sort$)	Mean of $\frac{\bar{F}_p}{F_c}$ in F_c order

6.2.3 Experimental Design

Two comparison experiments are set for the ratio for improvement of fitness: $I.Pairs$ vs. $I.Groups$ and $R.Sort$ vs. $C.Sort$. Two operator selections, AP and APM, are used on both experiments. By adding initial character of the evaluation of operator impact to the method of the operator selection, we denote each combined method. For example, AP/G means AP with $I.Groups$ and APM/P_C means APM with $I.Pairs$ in $C.Sort$ order. For experiments, we used the same problem sets and parameter setting to section 4.1.

6.2.4 Result and Discussion

Table 6.5: Success Proportion: $I.Pairs$ and $I.Groups$

	AP/P	AP/G	APM/P	APM/G
F_4	94%	94%	98%	94%
F_5	92%	91%	89%	95%
F_6	51%	57%	56%	52%
F_7	46%	40%	47%	56%
F_8	18%	18%	20%	23%
F_9	14%	19%	21%	13%
Q	84%	81%	76%	78%
S	96%	96%	95%	96%
T	67%	66%	72%	77%
$2B$	28%	28%	26%	22%
M_{25}	37%	35%	32%	34%
M_{30}	15%	17%	17%	13%
O_{25}	81%	77%	82%	82%
O_{30}	61%	58%	71%	68%

Table 6.5 is success proportions for all problems. It seems that there is no big difference between $I.Pairs$ and $I.Groups$ in the performance. Much $I.Pairs$ is better in AP, and much $I.Groups$ is better in APM even $I.Pairs$ seems to be slightly better at more complex problems in APM. However, nothing is clearly superior to the other.

The difference between two methods can be caught when we look the change during a run. Figure 6.5 and 6.7 show average changes of operators during a run. Briefly said, these figures look similar. In particular, the rate value of each operator and its variation in figure 6.5 and 6.7 resemble. The main difference of theses pairs, is that $I.Groups$ looks slightly more wild, while $I.Pairs$ is more smooth. On a closer

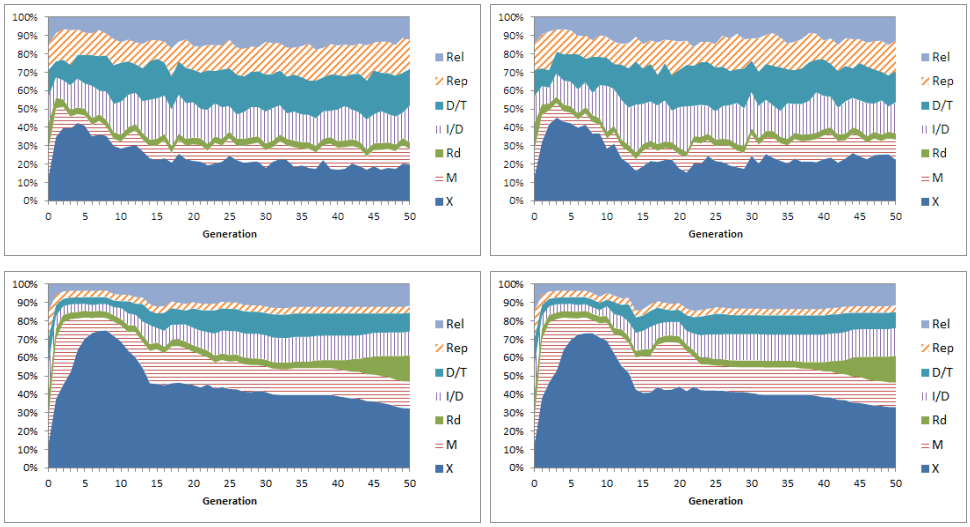

Figure 6.5: Change in Operator Application Rates on AP

Left: $I.Pairs$, Right: $I.Groups$

Top: F_7, Bottom: O_{25}

view, more peaks are observed in $I.Groups$ figures. That is, $I.Groups$ looks it is more changed in shorter time.

Figure 6.6 and 6.8 are their corresponding mean Quality Vector, and they can provide an explanation for rates figures. Figure 6.6 and 6.8 also look similar, however $I.Groups$ has smaller values than $I.Pairs$. Some values in $I.Groups$ are dropped under the one which means it didn't improve on average. At the last, $I.Groups$ shows more peaks like as in rates figures.

Comparing two methods, $I.Pairs$ has larger variance than $I.Groups$. For example, let's assume there are 3 parent-child pairs, and their fitness are (5, 2), (7, 6) and (6, 7). Then, the value of $I.Pairs$ is 1.508 and 1.2 by $I.Groups$. Meanwhile, in case of (2, 5), (7, 6) and (6, 7) pairs, $I.Pairs$ has value 0.808 and $I.Groups$ has value

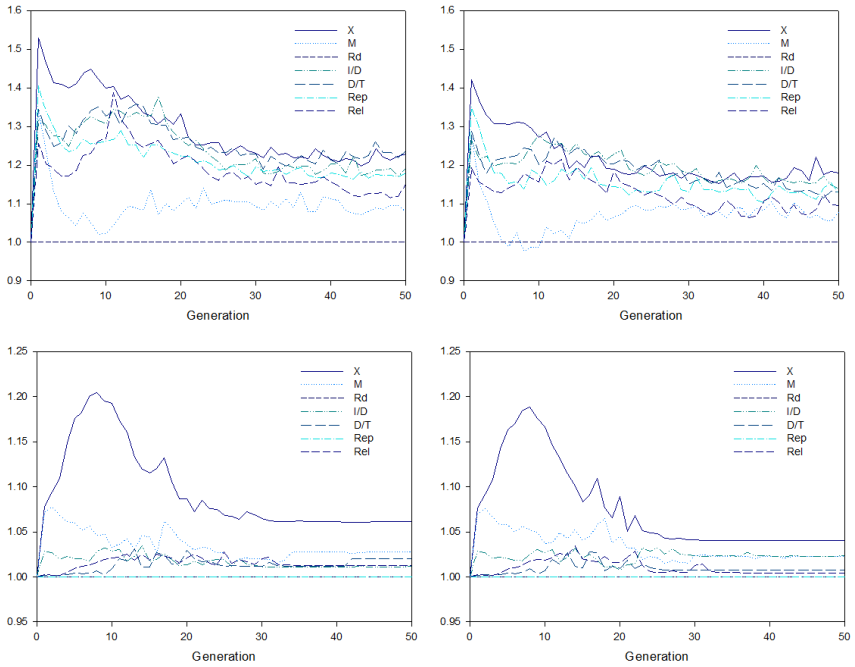

Figure 6.6: Change in Internal Status on AP
Left: $I.Pairs$, Right: $I.Groups$
Top: F_7, Bottom: O_{25}

0.833. Like these simple examples, $I.Pairs$ is more sensitive than $I.Groups$. If there is one big or one small pair in the individual set for an operator, $I.Pairs$ cause a bigger change than $I.Groups$. On the other hand, it also means that quality values of operator in $I.Groups$ are more dense than $I.Pairs$. So the largest quality valued operator can be more easily changed in $I.Groups$ and it causes more peaks in rates figures.

At last, this experiment investigated two different definitions about the good individual. While $R.Sort$ sees the ratio value, $C.Sort$ focuses the fitness of child

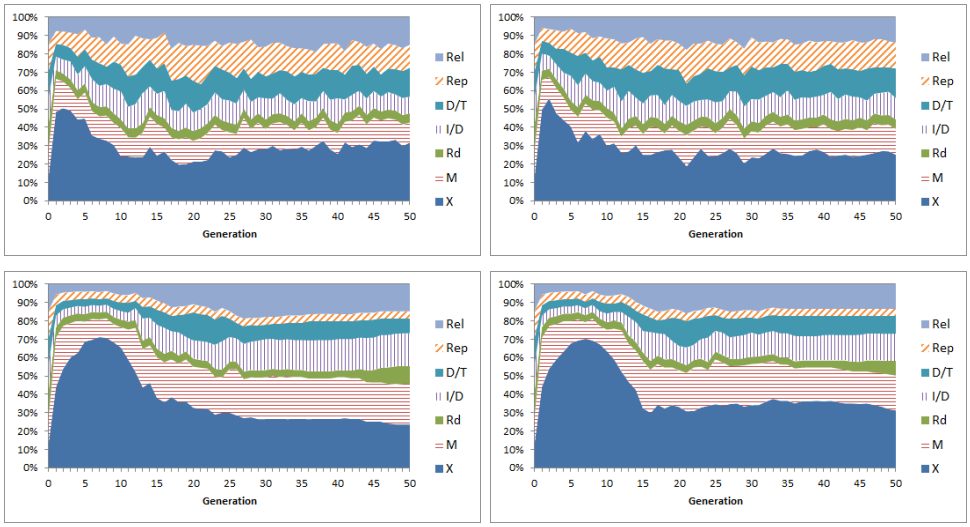

Figure 6.7: Change in Operator Application Rates on APM
Left: $I.Pairs$, Right: $I.Groups$
Top: Trigonometric, Bottom: O_{30}

individual only. Let's assume there are two parent-child pairs $R.Sort$ and q, and their fitness are $I.Pairs$:(5, 3) and q:(1, 2), each. $R.Sort$ chooses $R.Sort$ as the good individual, but $C.Sort$ chooses q, even q didn't improve. With these criteria, two methods sort individuals and calculate the operator impact value.

Table 6.6 is success proportions for all problems. Comparing two methods, they are not big different to each other and there is also no superior one. However, it seems $I.Pairs$ is good in APM method. At more problems, $R.Sort$ shows better performances.

In figure 6.9, rate figures of two methods are similar, but it seems that more peaks are in $I.Pairs$. The proportion of I/D is bigger at $C.Sort$, but Rel's one is bigger at $R.Sort$.

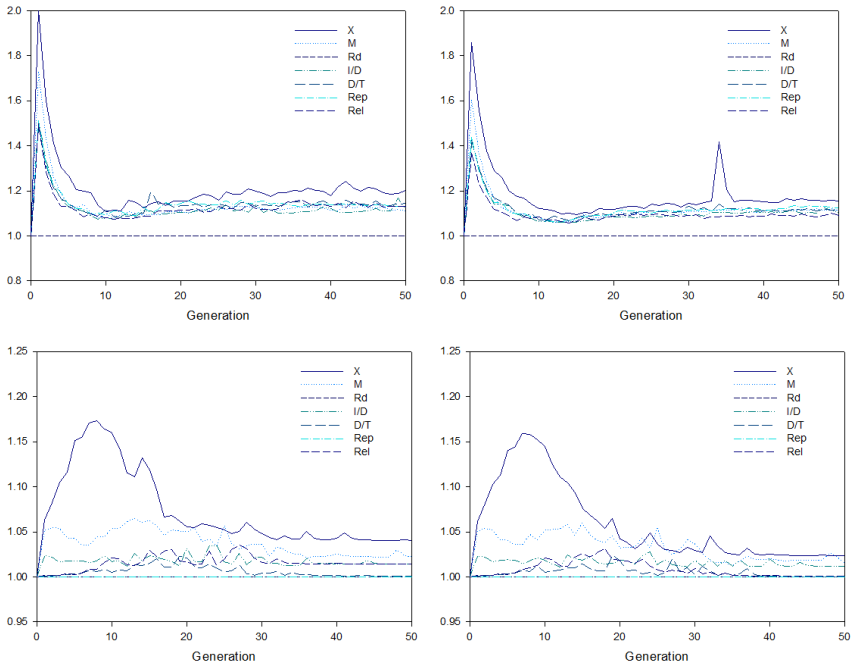

Figure 6.8: Change in Internal Status on APM
Left: $I.Pairs$, Right: $I.Groups$
Top: Trigonometric, Bottom: O_{30}

Figure 6.10 is its corresponding quality vector. What $C.Sort$ choose as the good individuals are the individual has a good child fitness value. Therefore, it doesn't guarantee a good ratio value, then its impact value is less than $R.Sort$'s. However, the difference between $R.Sort$ and $C.Sort$ is differ to the difference of $I.Pairs$ and $I.Groups$. On the case of $I.Pairs$ and $I.Groups$, the smaller value's one has more peaks. But $C.Sort$ has smaller quality values, but it shows less peaks.

it is easy to guess that $C.Sort$ has smaller impact value than $R.Sort$, cause of what $C.Sort$ takes is not a good ratio valued one. In place of $C.Sort$ investigates

Table 6.6: Success Proportion: *R.Sort* and *C.Sort*

	AP/P$_R$	AP/P$_C$	APM/P$_R$	APM/P$_C$
F_4	94%	95%	94%	93%
F_5	91%	91%	95%	90%
F_6	57%	54%	52%	65%
F_7	40%	52%	56%	45%
F_8	18%	19%	23%	20%
F_9	19%	13%	13%	13%
Q	81%	79%	78%	76%
S	96%	96%	96%	98%
T	66%	77%	77%	71%
$2B$	28%	23%	22%	29%
M_{25}	35%	29%	34%	36%
M_{30}	17%	14%	13%	15%
O_{25}	77%	80%	82%	75%
O_{30}	58%	63%	68%	56%

the amount of improvement itself, it focuses the improvement of good results. So, if the result is not good, even an operator occur a big improvement, it is useless in *C.Sort*. In other words, good fitness value is the necessary condition before *C.Sort* consider the operator impact. Contrastively, *R.Sort* only check improvement. Even the fitness value of an individual is good, it can be useless unless it is caused with a big improvement. But, the fitness values that *R.Sort* chooses, are not so bad. Basically, all individuals are selected by the selection mechanism before operators is applied to. Therefore, *R.Sort* chooses the good improvement ones on the baseline by the selection mechanism.

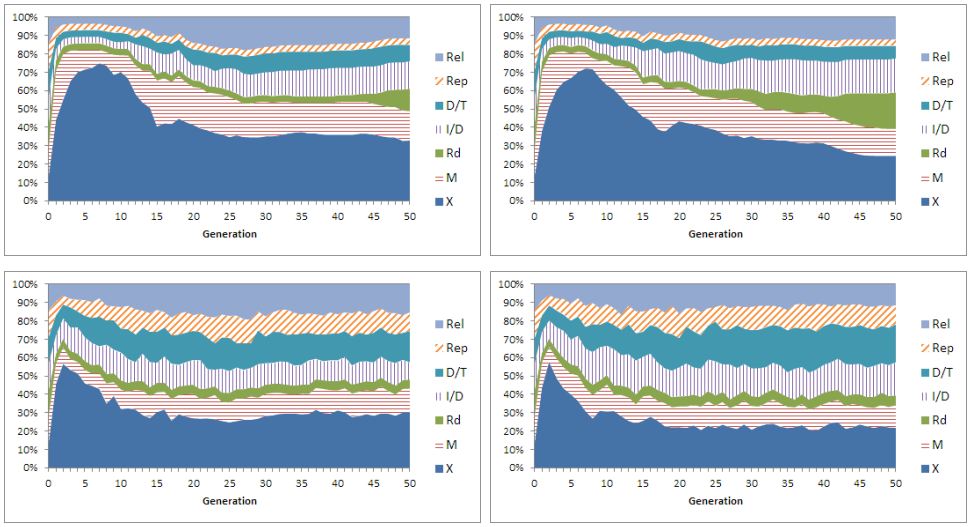

Figure 6.9: Change in Operator Application Rates

Left: *R.Sort*, Right: *C.Sort*

Top: O_{30} at AP, Bottom: Q at APM

From these features of *R.Sort* and *C.Sort*, I/D operator are used with different ways. I/D small-changes an individual, so it is good for the fine tuning. So, the scale of the fitness-improvement by I/D is restricted. By this reason, I/D can't receive any attention in *R.Sort*, before other operators' result get worse. However, *C.Sort* can focus I/D which is slow but be better steadily. In particular, it may be useful after middle of the learning. Meanwhile, *Rel* has more proportion in *R.Sort*, but it seems the reaction by the change of I/D.

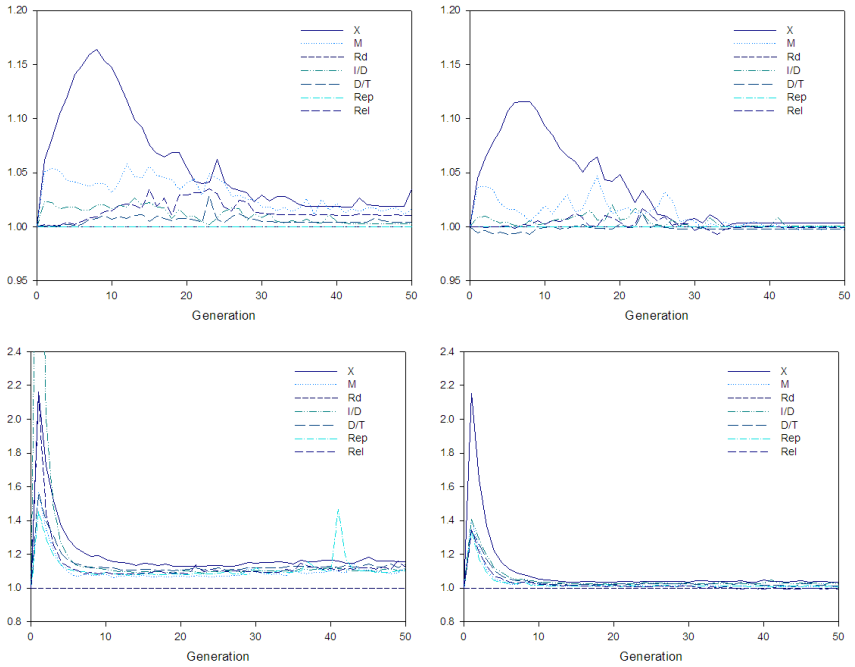

Figure 6.10: Change in Internal Status

Left: *R.Sort*, Right: *C.Sort*

Top: O_{30} at AP, Bottom: Q at APM

6.3 Ranking Point

6.3.1 Definition of Ranking Point

Section 6.2 used raw fitness value, however raw fitness value has not only merits but also demerits. In most EAs, fitness values of the initial population are very poor. Thus the biggest improvement of fitness is usually happened after the first generation. In addition, a scale of fitness improvement is decreasing as EAs run, because only small changes are occurred after some generations in EAs. Therefore,

a huge raw fitness value is able to more affect to the next generations than later values. For avoiding this problem, several methods can be used such as normalization. Ranking point, we introduce at this section, is also one of those methods.

$$Point_i = \frac{R_H - R_L + 1 - R_i}{(R_H + R_L)\frac{R_H - R_L + 1}{2}} \tag{6.2}$$

Ranking point uses fitness based ranking information for evaluating the operator impact. At the end of every generation, all individuals in population are sorted in fitness order. Then each individual gets a point from a rank which an individual received. Equation 6.2 shows the relation between a point and a rank, where R_i is a rank of an individual i, R_L is the lowest (best) rank and R_H is the highest (worst) rank. Hence all individuals are not used for the evaluation, we set R_L and R_H.

6.3.2 Experimental Design

Three operator selection methods, PPM, APM and r-AP, are used for the comparison between raw fitness and ranking point. Both evaluation methods, we used 30% of elite individuals for the operator impact. We used same problem sets to section 5.2.1 and same parameter setting to section 5.2.2.

6.3.3 Result and Discussion

Table 6.7 is the success proportion for all problems. The performances of two evaluation methods show no big difference, but there is an interesting thing between PPM and APs (APM and r-AP). PPM with ranking point more-performed than raw fitness, on the contrary APs with ranking point less-performed. In particular, result from D_W shows a big advancement. And except O_n, PPM with ranking point shows better performance than improvement ratio. However, in almost problems, the performances of APs with ranking point are worse than with improvement ratio.

Table 6.7: Success Proportion: Ranking Point

	Raw Fitness			Ranking Point		
	PPM	APM	r-AP	PPM	APM	r-AP
F_4	95%	95%	94%	97%	92%	93%
F_5	90%	96%	89%	93%	85%	85%
F_6	56%	57%	60%	60%	54%	44%
F_7	41%	54%	45%	45%	37%	37%
F_8	18%	24%	25%	17%	22%	22%
F_9	17%	16%	19%	15%	18%	17%
Q	73%	84%	77%	77%	88%	81%
S	97%	98%	99%	94%	91%	87%
T	74%	72%	73%	85%	67%	70%
$2B$	23%	28%	24%	31%	23%	17%
M_{25}	15%	32%	33%	20%	28%	34%
M_{30}	6%	15%	14%	3%	7%	7%
O_{25}	83%	80%	83%	76%	78%	72%
O_{30}	51%	59%	55%	44%	39%	48%
D_N	100%	98%	97%	100%	8%	9%
D_W	51%	96%	89%	95%	1%	0%

As the weak point, PMs (PM and PPM) pointed out that they couldn't well distinguish operators which has similar impacts. As 4.1 and 5.2.3, all operator rates of PMs were similar to each other. That is, PMs' probability distribution methods are not suitable for GP. Even PPM is made to overcome this weak point, it failed to completely overcome the weak point. However, hence ranking point gives point by the rank, definite differences are exist between ranks, even their differences are very small. So, the ranking point could amplify the difference, and it makes PM's work

better.

Conversely, ranking point shows less performance at APs. In particular, APs with ranking point quite less performed on the target-structure problems. In the same context to previous, the ranking point might provide a distinguishable impact. But APs have already amplified the difference by focusing only the most effective one. With the proportion result, we can assume ranking point and APs have any confliction.

6.4 Pre-Search Structure

6.4.1 Definition of Pre-Search Structure

AOS provides effective operators based on their previous performances. However even though AOS uses the most recent operator impact, it is evaluated on the past environment. Formally, the operator rate at generation t, P_t, is described as a function $F(I_1, ..., I_{t-1})$ where I_n is an operator impact at generation n. Although the operator impact is evaluated by the most recent population, it sometimes occurs a wrong guidence, in particular at dynamically changing situation. In addition, the operator impact has a bias because all operators have different rates. The impact of the operator which has high rate value is evaluated from many individuals which the operator is applied to, however only few individuals are resources for the impact of operator of low rate. The basic concept of Pre-Search structure are more direct and fair impact by sampling process.

Figure 6.11 is the scheme of pre-search structure. Before the main process of evolutionary algorithm, a sampling process is inserted. Sampling process makes some samples which are made by equal portion of genetic operators. That is, each genetic operator has the equal number of samples that the corresponding operator is applied

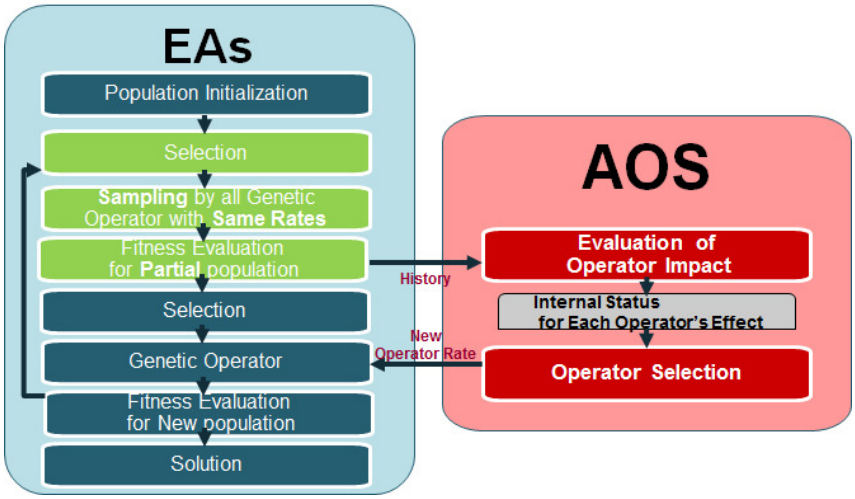

Figure 6.11: Scheme of Pre-Search Structure

to. These samples are a partial population for the next generation. AOS evaluates the operator impact and it suggests a new operator application rates for making the rest population for the next generation. We made the amount of samples is defend to the minimum probability P_{min} (Eq. 6.3)

$$\#ofSamples \quad = \quad P_{min} \times |Population| \tag{6.3}$$

As PPM, APM and r-AP have P_{min}, Pre-search structure makes samples as much as P_{min} is applied for each genetic operator. But when operator selection is applied at pre-search structure, it uses operator selection methods with zero P_{min} cause the expected amount of individuals by P_{min} are already applied as samples.

6.4.2 Preliminary Experiment for Sampling

One of big issues of pre-search structure is summarized this question, is it possible that finite number of samples provide enough information for operator impact? Kim et al. (2012b) answers the question, yes. The finite number of sampling can catch the feature of genetic operators.

This work aims to characterize the effect on fitness, size or depth of the various evolutionary operators. The change depends on the state of the system, hence we wanted to see how that change itself varied over the course of an evolutionary run. We did this by conducting typical GP runs. At each generation, in addition to the normally-created children which were actually used in the evolutionary run, we generated extra children simply to evaluate the effects of the different operators, but not otherwise used in the run.

In each generation, we took 200 additional samples for each operator (in addition to those used for evolution) – of the same order as the number of real trials of each operator in a generation. We selected the parents for these trials using the selection mechanism. Thus we were examining the children actually reachable after selection.

We conducted detailed analyses on all experiments, but can only show F_9 and O_{30} due to space. F_9 is intermediate in difficulty and typical of both extremes, while O and M problems behaved similarly to each other. F_9 and O_{30} are sufficient to summarize the general trends, though we will mention some more detailed observations when appropriate. For brevity, we denote a plot for function X_m calculated from the fittest $n\%$ of children as $X_m^{n\%}$, with $n \in \{10, 30, 50, 70, 90\}$ and $X \in \{F, M, O\}$. The figures show how the genetic operators change the properties of individuals in each learning stage. The horizontal bars indicate means over 100 runs, while the vertical lines show their standard deviations.

All plots show how each operator changes the specific property for individuals

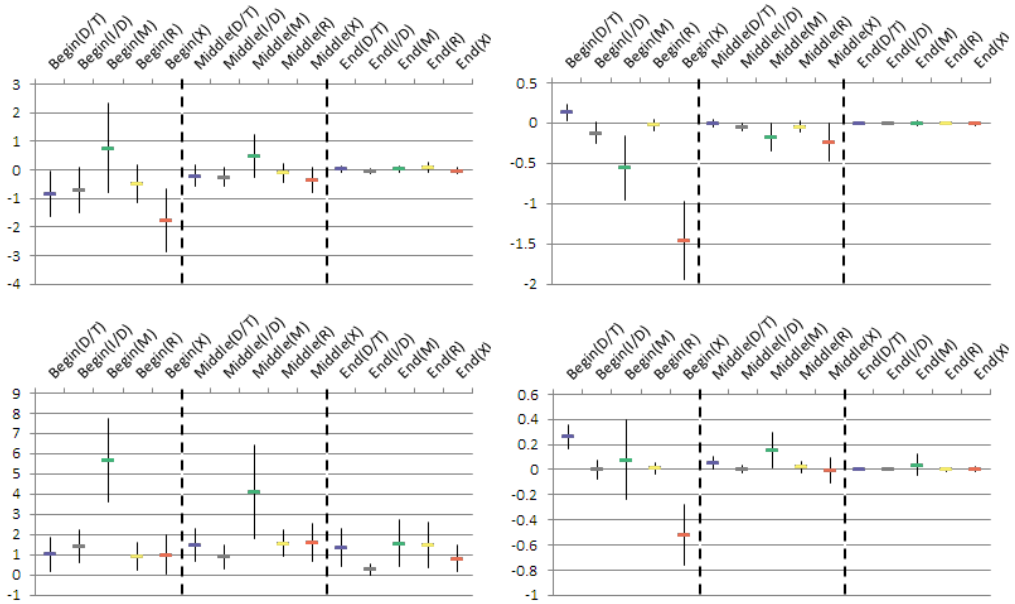

Figure 6.12: Fitness Change for Selected Parents.
Top: 30% Elite; Bottom: 70% Elite;
Left: F_9; Right: O_{30}

(the difference between child and parent values – for fitness, negative values indicate improvement). Replication is omitted because it deterministically has no effect.

Fitness Analysis

The results in Fig. 6.12 overall reflect our understanding of evolutionary behavior: the operators have a larger range of effect in early search (they are more exploratory), whereas later on, elite children resemble their parents much more.

The most notable differential effect in Fig. 6.12 is the much larger range of effect of the traditional M and X operators: the new TAG3P operators have a much

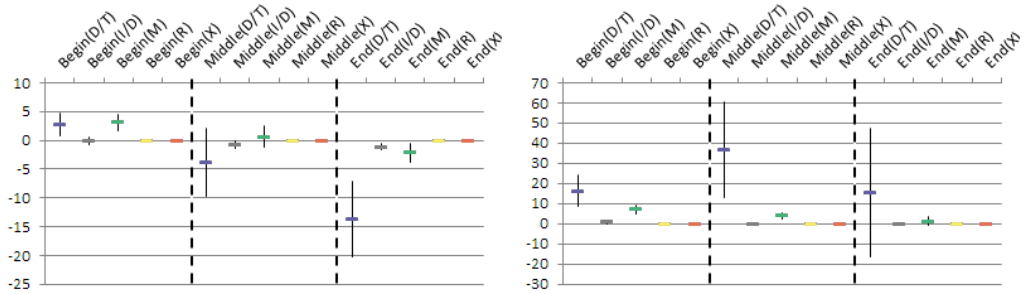

Figure 6.13: Size Change, Left: $F_9^{50\%}$; Right: $O_{30}^{50\%}$

smaller overall range of effect, suggesting that they are much less exploratory. In the early stages, X is on average much more beneficial than mutation – for F_9, most of the 30% elite children are an improvement on their parents, while much fewer M children are; any benefit from M comes from rarer positive mutations. While M is overall constructive for problem O_{30}, it is still substantially less so than X. However the effect of X rapidly diminishes, especially for O_{30}; M remains effective longer.

I/D are generally beneficial in early stages (the 30% elite see some worthwhile improvement on their parents. I/D retains small but very slightly beneficial effect until the end stages, befitting its proposed role as a fine-tuning operator.

D/T behave similarly to I/D on F_9, though any beneficial effect disappears by the end stages. Their effect on O_{30} is rather different, being slightly damaging in the early stages of search, very slightly beneficial in the mid stages, and losing all effect at the end.

R throughout has a relatively small effect, disappearing almost entirely by the end stages (deterministically, it had no effect in the majority problem, since it cannot change the fitness).

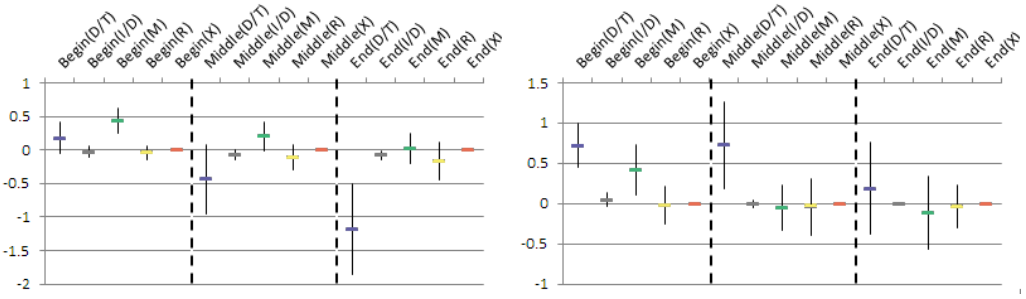

Figure 6.14: Depth Change, Left: $F_9^{50\%}$; Right: $O_{30}^{50\%}$

Size Analysis

While we saw different trends between 30% and 70% elite children in the fitness plots, there was no such difference for size – size effects were independent of child fitness; we display the results for the 50% elite. R and X do not change size at all, so we omit them from discussion.

D/T generally causes a size change over the run (Fig. 6.13), with the scale increasing gradually. However the effect is reversed between the problems: D/T decreases size for F_9 but increases it for O_{30} (similar, but less pronounced, effects were seen with other operators). The difference may be because most individuals were near the size bound in F_9, so that many larger duplications would fail, while most truncations would succeed, introducing a bias.

M began by slightly increasing the size of individuals, but the scale decreased to zero for O_{30}, and M eventually became reducing for F_9. I/D (by design) made only very small size changes throughout.

Depth Analysis

We omit analysis of X because, as with size, most operator applications result in no change in depth, so there is little to see.

The general trends are similar to size (Fig. 6.14), but on a reduced scale (because of the logarithmic relationship between depth and size). The shapes of the plots are generally very similar. The only exception is with operator R, which shows a slight bias toward depth reduction, increasing in scale over time.

We investigated the roles genetic operators play and what they are useful for. We confirmed that crossover is an effective operator in the early stages of GP, but it is not effective throughout a run. Subtree mutation, another well known operator, causes large changes in fitness, even in the middle of a run, but the changes are generally negative. Insertion/deletion may be a useful alternative, leading to smoother fitness search – It is effective for fine-tuning, but at the risk of getting stuck in local optima. Duplication/truncation and relocation may be useful when structural change is needed, but can also have negative effects on poorly-matched problems.

More generally, we may conclude that there is value in having a diverse range of operators: they really do perform different tasks, either in different problems, or at different times in the evolution of solutions for the same problem. Since we will not, in general, have a priori knowledge of which operator is most suitable a any specific time, this motivates and justifies research into operator adaptation in evolutionary algorithms in general, and in GP in particular.

6.4.3 Experimental Design

Problem sets and parameter setting are same to section 5.2. PPM, APM and r-AP are used for this experiment. And two pre-search structures are used. The first one use only one best sample (One Elite), and the second use 30% elite individuals from

samples (30%).

6.4.4 Result and Discussion

Table 6.8: Success Proportion: Pre-Search Structure

	One Elite			30%		
	PPM	APM	r-AP	PPM	APM	r-AP
F_4	64%	76%	62%	93%	95%	95%
F_5	57%	65%	56%	90%	85%	87%
F_6	35%	31%	24%	53%	49%	54%
F_7	17%	26%	25%	44%	41%	43%
F_8	9%	15%	9%	16%	17%	18%
F_9	8%	10%	7%	21%	10%	19%
Q	59%	85%	80%	63%	78%	69%
S	26%	56%	52%	98%	91%	94%
T	33%	59%	55%	73%	70%	66%
$2B$	15%	0%	0%	15%	17%	18%
M_{25}	7%	0%	0%	10%	27%	28%
M_{30}	0%	0%	0%	6%	13%	10%
O_{25}	19%	0%	0%	81%	78%	83%
O_{30}	0%	0%	0%	48%	58%	55%
D_N	100%	100%	100%	100%	98%	97%
D_W	0%	0%	0%	51%	96%	89%

Table 6.8 shows the success proportion of AOS with pre-search structures. Except a few cases, pre-search structure with one elite didn't show good performances. However pre-search structure with fitness ratio over 30% elite individuals shows as same performance as the normal structured AOS. As the reason why pre-search structure with one elite sample couldn't work as well as we expected, we guess a lack of infor-

mation makes the performance be worse. Fialho et al. (2008) showed that AOS in GA environment could well-perform with only one extreme fitness individual. However, it is not sure that it is same in GP environment. With the same view, one sample seems to be a main reason of bad performance, but it is not certain. Meanwhile, 30% well-performed for all problems; its performances are similar to table 5.6 and 4.6. From the result, we can conclude that pre-search structure can work as much as the performance of a normal structured AOS. However, it is still uncertain whether it can over-perform a normal structured AOS mechanism. We guess the key point of pre-search structure is on the number of samples. To more investigate its usefulness, more varied numbers of samples are needed to be researched.

Chapter 7

Application: Nakdong River Modeling

The Nakdong, which has more than 500 km of length, is the longest river in South Korea. Approximately 10 million people live in and use water from the Nakdong River basin and it causes conflicting requirements of water usage. Therefore, the management system for Nakdong river is essential. In this chapter, we introduce a prediction model of algal bloom for water quality of Nakdong river and we apply AOS mechanism to the model.

7.1 Problem Description

7.1.1 Outline

The Nakdong River system is one of the major regulated river systems of North East Asia. As Korea has been developed, upstream dams and an estuarine barrage are built in the river, so Nakdong river regularly shows, by turns, characteristics of a reservoir and a river. Regional climatic conditions govern the hydrological regime: the

annual rainfall is nearly 1,200 mm per year, over 60% concentrated in the Summer period (from June to September) (Jeong et al. (2007)). Hence approximately 10 million people live in and use water from the Nakdong River basin, high demand and intensive use of water resources lead to conflicting requirements, a key issue being the occurrence of algal blooms, fueled by the nutrients injected upstream, which periodically blight the river in the vicinity of Busan ($\approx$ 5 million people) in the lower part of the river. The lower Nakdong River experiences recurrent algal blooms of Summer cyanobacteria and Winter diatoms (Ha et al. (1999, 2003)). Mitigating these algal blooms is a key economic and social issue. Widely various limnological research in terms of water quality (Kim et al. (1998, 2007)) and plankton dynamics (Ha et al. (1999); Kim and Joo (2000)) have been conducted. So the important is the management of the river that the Korean government invested in the vicinity of USD 19 billion (for four major rivers, of which this is the largest) in a scheme to improve its water management, and an intensive monitoring programme known as the National Long-Term Ecological Research (LTER) has been carried out over the past decade (Kim and Kim (2011)).

7.1.2 Data Description

In building the models presented in this thesis, we used geographical, hydrological, meteorological, physicochemical and biological datasets. We have data from nine measuring stations throughout the catchment (Fig. 7.1). They were originally selected based on the availability of data and geographical importance. Six stations ($S1, S2, ..., S6$) are located in the main channel of the river, and the other three stations ($T1, ..., T3$) are situated in major tributaries. Of those stations, algal concentration in the lowest ($S1$, Mulgeum) is the most important hence the high population ($\approx$ 5 million people) of Busan draws its water nearby. Data (e.g. water temperature,

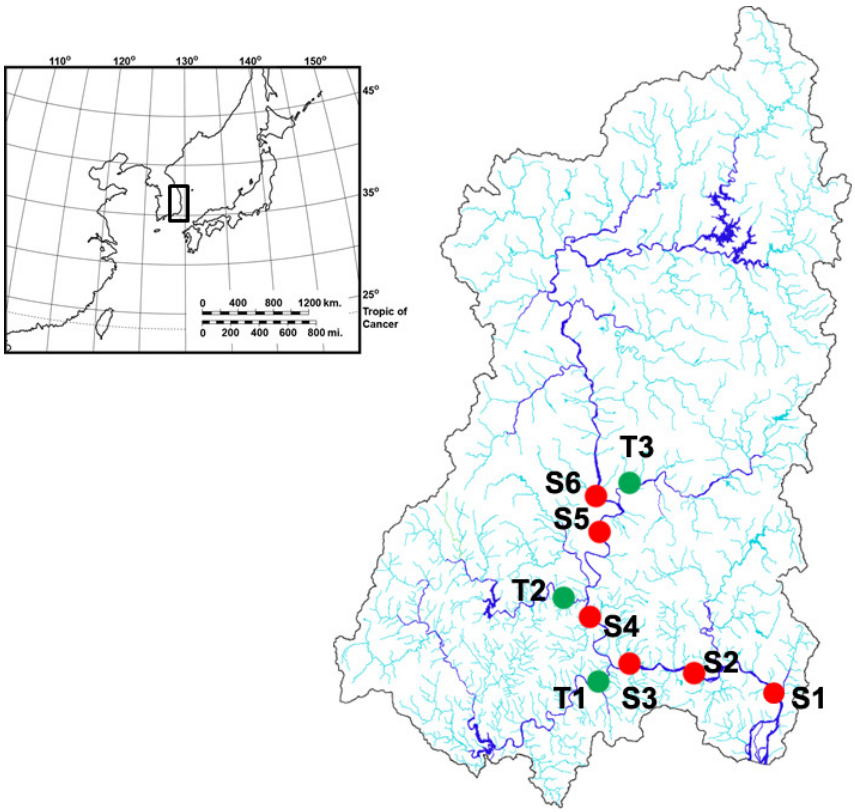

Figure 7.1: Nakdong River Basin

solar radiation, precipitation, flow rates, nutrient concentrations, and chlorophyll *a*) are collected during thirteen years (1996 - 2008), and most were daily-collected. Hydrological (e.g. flow rate) and meteorological (e.g. irradiance and precipitation) data were provided by the Korean Water Management Information System (Korean Ministry of Land, Infrastructure and Transport (WAMIS)) and the Korea Meteorological Administration (Korea Meteorological Administration (KMA)). Others were provided by Limnology Laboratory of Pusan National University.

7.1.3 Model Description

The process model for Nakdong river in this thesis is based on the model which was introduced in Kim et al. (2010). it is consist of two contemporaneous processes describing the hydrological (flow of bodies of water) and biological (dynamics of plankton) mechanisms. The big differences from the previous model are two things; the nutrient and temperature equations and zooplankton. The nutrient and temperature equations in this thesis more reflect the commonly-used forms (Cole and Buchak (1995); Arhonditsis and Brett (2005)) by incorporating maximum and minimum values of parameters. And new variables of zooplankton, which effects the growth of phytoplankton, are added (Kim and Joo (2000)).

Hydrological Process

The flow model maintains a mass balance of water in river, from upstream to downstream. The flow rates are based on the data from WAMIS, who uses a model of the form given in equation 7.1.

$$Flow \quad = \quad \alpha \times (H + \beta)^{\gamma} \tag{7.1}$$

where H is the water level (height) of the river, and α, β and γ are site- and time-specific parameters varying with the riverbed contour. α, β and γ are recalibrated at infrequent intervals through direct height and flow measurements. However, these three parameters are only changed prospectively on recalibration, not retrospectively for the forward prediction for the flood mitigation, so that they are optimized to give the most accurate flow rates at high flows. However, we need the most accurate estimates in low flow period cause algal blooms occur primarily in low flow regimes. In addition, these parameters are affected by the change of riverbed contour. The

change of riverbed contour is generally gradual over time, but punctuated at irregular intervals by the silt carried in the extreme flows from episodic events and more rarely by dredging. The parameter re-estimation by WAMIS does not coincide with the events affecting the river contour. Therefore, fitting parameters were sometimes used for extended periods in the data as supplied by WAMIS.

To overcome these problems, we have re-estimated the historical α, β and γ parameters, using the known occurrences of extreme flows from meteorological events and inferring occurrences of dredging from sudden changes in mass balance, to segment the data over time. Then we used the actual calibrations by WAMIS to infer these values both prospectively and retrospectively, and focusing on minimizing the flow error in low flow regimes. With the new parameter values, we re-estimated the river flow over the study period to better fit our purpose.

The flow model uses a simple flow mass balance between stations, and it provides flow-at-time information to the biological process model. Equation 7.2 shows the basic flow model, which has three parts; inflow from upstream A, flow retention downstream B, and run-off R by precipitation.

$$Flow_{B,t+d} = (1 - r_A) \cdot F_{A,t} + r_B \cdot F_{B,t} + R_{B,t+d} \tag{7.2}$$

where $Flow_{X,t}$ denotes the flow at station X at time t, d is the time it takes water to flow from station A to station B, and r_X is the fraction of the water that is retained at station X. Thus, $(1 - r_A) \cdot F_{A,t}$ is the outflow from station A, $r_B \cdot F_{B,t}$ is the proportion of flow retained at station B due to non-laminar flow, and $R_{B,t+d}$ indicates the inflow arising from run-off of precipitation occurring in the catchment of station B at time $t + d$. A simple additive model is used for the confluence of two streams.

Biological Process

The biological process bidirectionally interacts with the hydrological process, and it decides the temporal dynamics of the phytoplankton biomass (BP), a proxy for the trophic state of water body. The biological process model mediates the change of phytoplankton in a flowing water body over time-specifically, the transit time between stations. The transit time is determined by the distance and corresponding velocity of water between two neighbor stations. The velocity data were provided by the Nakdong River Environment Research Center of the National Institute of Environmental Research. We downscaled the velocities based on site-specific regression using the corresponding flow rates. All the coefficients of determination (r^2) were greater than 0.99.

$$
\begin{aligned}
\frac{dBP}{dt} &= BP \cdot (Growth_A - Breath_A) - BZ \cdot Grazing \qquad (7.3)\\
Growth_A &= (\sqrt[24]{1 + C_{UA}} - 1) \cdot f(V_{lgt}) \cdot g(V_n, V_p, V_{si}) \cdot h(V_{tmp})\\
Breath_A &= (1 - \sqrt[24]{1 - C_{BRA}}) \cdot e^{Q_{10a}(V_{tmp} - 20)}\\
Grazing &= (1 - \sqrt[24]{1 - C_{MFR}}) \cdot \frac{BP - C_{Fmin}}{K_{FS} + Chl_A - C_{Fmin}} \cdot e^{-C_{ZT}(V_{tmp} - 20)^2}\\
f(V_{lgt}) &= \frac{V_{lgt}}{C_{bl}} \cdot e^{1 - \frac{V_{lgt}}{C_{bl}}}\\
g(V_n, V_p, V_{si}) &= MIN(\frac{V_n}{K_n + V_n}, \frac{V_p}{K_p + V_p}, \frac{V_{si}}{K_{si} + V_{si}})\\
h(V_{tmp}) &= MAX(e^{-C_{PT}(V_{tmp} - C_{btp1})^2}, e^{-C_{PT}(V_{tmp} - C_{btp2})^2})
\end{aligned}
$$

$$
\begin{aligned}
\frac{dBZ}{dt} &= BZ \cdot (Growth_Z - Breath_Z - Death_Z) \qquad (7.4)\\
Growth_Z &= (\sqrt[24]{1 + C_{UZ}} - 1) \cdot \frac{BP - C_{Fmin}}{K_{FS} + Chl_A - C_{Fmin}} \cdot e^{-C_{ZT}(V_{tmp}-20)^2}\\
Breath_Z &= (1 - \sqrt[24]{1 - C_{BRZ}}) \cdot e^{Q_{10b}(V_{tmp}-20)} + C_{BMT} \cdot Grazing\\
Death_Z &= (1 - \sqrt[24]{1 - C_{DZ}}) \cdot 0.9^{(V_{tmp}-20)}
\end{aligned}
$$

Table 7.1: Model Variables

Variable	Description	Unit
V_{wd}	wind	day^{-1}
V_{ph}	pH	N/A
V_{cd}	electric conductivity	μS cm^{-1}
V_{do}	dissolved oxygen	mg L^{-1}
V_{tmp}	water temperature	°C
V_{tb}	turbidity	NTU
V_{fp}	fish predation	N/A
V_{sd}	Secchi depth	cm
V_{bc}	bacteria density	N/A
V_{lgt}	light intensity	MJ m^{-2} d^{-1}
V_n	nitrogen	mg L^{-1}
V_p	phosphorus	μg L^{-1}
V_{si}	silicon	mg L^{-1}

The main equations for algal biomass were a simplified form incorporating photosynthetic production ($Growth_A$), metabolic degradation ($Breath_A$) and herbivorous zooplankton grazing activity ($Grazing$) (Eq. 7.3). Algal growth was subject to multiplicative influences from solar radiation (V_{lgt}), water temperature (V_{tmp}) and nutrient concentrations (nitrate V_n, phosphate V_p and silica V_{si}) (Table 7.1). These limiting functions were partially adapted from the studies of Cho and Shin

Table 7.2: Model Parameters and their Exploration Bounds

Parameter	Description	Unit	Reference Value	Bounds
C_{UA}	maximum growth rate of phytoplankton	day^{-1}	1.89	0.1∼4.0
C_{UZ}	maximum growth rate of zooplankton	day^{-1}	0.15	0.0∼0.3
C_{BRA}	breath rate of phytoplankton	day^{-1}	0.021	0.0∼0.17
C_{BRZ}	breath rate of zooplankto	day^{-1}	0.05	0.0∼0.20
Q_{10A}	Q_{10} coefficient (for BA)	$°C^{-1}$	0.069	0.01∼0.13
Q_{10Z}	Q_{10} coefficient (for BZ)	$°C^{-1}$	0.05	0.01∼0.09
C_R	choosing coefficient for feeding	N/A	0.88	0.2∼1.0
K_{FS}	half-saturation constant of food	$\mu g\ L^{-1}$	5.0	4.0∼6.0
C_{Fmin}	minimum food concentration	$\mu g\ L^{-1}$	1.0	0.1∼1.9
C_{btp1}	blue-green optimal temperature 1	°C	27.0	20.0∼34.0
C_{btp2}	diatom optimal temperature 2	°C	5.0	1.0∼20.0
C_{MFR}	maximum feeding rate	day^{-1}	0.19	0.01∼0.8
C_{bl}	best light for phytoplankton	MJ m^{-2} d^{-1}	26.78	24.0∼30.0
K_n	half-saturation constant of nitrogen	mg L^{-1}	0.0351	0.02∼0.05
K_p	half-saturation constant of phosphorus	mg L^{-1}	0.00167	0.001∼0.020
K_{si}	half-saturation constant of silica	mg L^{-1}	0.00467	0.001∼0.2
C_{DZ}	death rate of zooplankton	day^{-1}	0.04	0.01∼0.10
C_{BMT}	breath multiplier on grazing	N/A	0.04	0.01∼0.07
C_{PT}	temp coefficient for phytoplankton growth	$°C^{-2}$	0.005	0.003∼0.2
C_{ZT}	temp coefficient for zooplankton growth	$°C^{-2}$	0.005	0.003∼0.2

(1998), Hongping and Jianyi (2002), and Arhonditsis and Brett (2005). We modified them to use two optimal temperatures for phytoplankton growth, since this river has been dominated by Summer cyanobacteria (Ha et al. (1999)) and Winter diatom (Ha et al. (2003)) blooms. The optimal values were determined based on Cho and Shin's experiment (Cho and Shin (1998); Reynolds (2006)). Zooplankton abundance (BZ) plays a key role in limiting phytoplankton biomass due to grazing pressure (Equation 7.4). The governing equations of zooplankton metabolism and grazing activity stemmed from Hongping and Jianyi (2002). Specifically we added

a temperature-dependent factor for phyto- and zoo-plankton respiration rates, and both grazing and mortality of zooplankton.

For the process models, we explored the ranges of model parameters from the previous related studies (Cho and Shin (1998); Everbecq et al. (2001); Hongping and Jianyi (2002); Arhonditsis and Brett (2005); Reynolds (2006)). We used our best estimates of these parameter values (Table 7.2) from the studies as a baseline for comparison. We also estimated boundary values from these studies. These boundary values are intended to represent ecological knowledge in the sense that we have high certainty that the parameter values lie within these regions. Presented with a well-fitting model whose parameter values lay outside the region, we would reject the model in preference to accepting the parameter values. This is important because there is little point in any parameter estimation method searching outside this region. The objective of this study is to investigate the quality of models that may be generated within these constraints for the Nakdong River ecosystem.

The system consists of two top-level models. The river flow model manages the interaction between stations, while the algal growth model calculates the change of status at each station. All measured data from the four highest stations, tributary stations T_1, T_2, T_3 and main channel station S_6, were used as sources. The model uses their data to compute values for downstream stations, which may be compared with the measured values. At a confluence where tributaries join the main channel, flow rates and water column parameters are combined, then propagated to the next reach.

7.1.4 Methods

We applied AOS to two experiments. One is a parameter optimization with GA (Kim et al. (in revision)), and the other is a modeling with TAG3P.

Table 7.3: Genetic Operators for Parameter Optimization by GA

	Genetic Operator	Description
1X	1-point Crossover	Values beyond a randomly chosen gene are exchanged
2X	2-point Crossover	Values between two randomly chosen genes are exchanged
UniformX	Uniform Crossover	For each gene, the value is chosen from either parent with probability 0.5
ArithX	Arithmetic Crossover	For each gene, uses the mean of the two parent values
RandX	Random Crossover	For each gene, uses a random value between the two parent values
ReproM	Reproduction Mutation	Values between two randomly chosen genes are re-initialised
UniformM	Uniform Mutation	For each gene, the value is re-initialised with probability 0.5
RangeM	Range Mutation	For each gene, value is randomly changed within 5% of range

Parameter Optimization

Parameter Optimization with GA aims to optimize the model parameters using a canonical GA. We used 20 real-valued genes as the chromosome of GA. They are corresponded to the 20 model parameters in the model (Table 7.2). We used 8 various genetic operators; 5-crossover and 3-mutation (Table 7.3). All operators have different features and they affect to individuals in a variety of ways. For this experiment, three different systems are used; w/o AOS, w AOS and Opt. w/o AOS is

a GA system without AOS mechanism, and its role in this experiment is the baseline for comparison. w AOS has the same GA system but AOS mechanism is applied to the system. At last, Opt is a GA system with two specific operator pairs; 1-point crossover and reproduction mutation. They are selected by analyzing the result of w AOS. We used 13 years ecological data described at 7.1.2, from 1996 to 2008. The data was divided into two parts; data from 1996 to 2005 was used for training and that from 2006 to 2008 was used for testing. Table 7.4 shows the evolutionary parameter setting in detail.

Table 7.4: Problem Definitions and Evolutionary Parameters for Parameter Optimization with GA

GA Type	real coded	Number of Operators K	8
Fitness	RMSE at station $S1$	Initial Operator Rate P_{init}	0.125
Runs	500	Minimum Rate P_{min}	$\frac{1}{10K}$
Maximum Generations	100	Maximum Rate P_{max}	$1-(K-1)\cdot P_{min}$
Population Size	100	Adaptation Rate α	0.8
Elite Size	1	Learning Rate β	0.8
Tournament Size	4	AOS	APM / Fitness Ratio

Modeling

Modeling with TAG3P is an extension of GA works at the previous section. The objective of this research is to generate more exact prediction models of the water quality of the river, by adding extensions to the biological process. Therefore, this research works not only to optimize 20 parameter values but also to find more correct river equation. Table 7.5 describes operators, variables and position of extensions. We used TAG3P with 7 genetic operator sets (Table 2.1 in section 2.1.3) and 185 elementary trees (1 α tree for biological process and 184 β trees for extensions) for this experiment. For this experiment, two systems, w/o AOS and w AOS, are used.

Table 7.5: Definition of Extensions

Ext	Operator	Variable	Position
Ext_1	$+, -, \times, \div, log, exp$	V_{ph}, V_{cd}	$\frac{dBP}{dt}$
Ext_2	$+, -, \times, \div, log, exp$	V_{sd}	$\frac{dBZ}{dt}$
Ext_3	$+, -, \times, \div, log, exp$	V_{alk}	$Growth_A$
Ext_4	$+, -, \times, \div, log, exp$	V_{tb}	$f(V_{lgt})$
Ext_5	$+, -, \times, \div, log, exp$	V_{do}, V_{ph}	$Breath_A$
Ext_6	$+, -, \times, \div, log, exp$	V_{tmp}	$Grazing$
Ext_7	$+, -, \times, \div, log, exp$	V_{tmp}, V_{alk}	$Growth_Z$
Ext_8	$+, -, \times, \div, log, exp$	V_{tmp}, V_{do}, V_{ph}	$Breath_Z$

Table 7.6: Problem Definitions and Evolutionary Parameters for Modeling with TAG3P

GA Type	real coded	Number of Operators K	7
Fitness	RMSE at station $S1$	Initial Operator Rate P_{init}	0.143
Runs	50	Minimum Rate P_{min}	$\frac{1}{10K}$
Maximum Generations	50	Maximum Rate P_{max}	$1 - (K-1) \cdot P_{min}$
Population Size	100	Adaptation Rate α	0.8
Elite Size	1	Learning Rate β	0.8
Tournament Size	4	AOS	APM / Fitness Ratio

Data usage and most settings are same to the parameter optimization with GA. Table 7.6 shows the evolutionary parameter setting in detail.

Table 7.7: Performance for Parameter Optimization with GA

	Training				Test			
RMSE	avg.	med.	stdev.	best.	avg.	med.	stdev.	best.
w/o AOS	24.8972	24.8741	0.09612	24.6853	22.558	22.5382	0.09938	22.3837
AOS	24.7489	24.7185	0.11386	24.6179	22.4725	22.446	0.12095	22.3009
Opt	24.7256	24.7106	0.09143	24.6306	22.3984	22.3896	0.05304	22.3073

7.2 Results

7.2.1 Parameter Optimization

Table 7.7 shows the average, median, standard deviation and best value of RMSE for the training and test. We omitted the performance of the basis model, however, Kim et al. (2010) already showed, by applying GA model to the basis model, it is possible to get the more correct river model. It seems there is no difference between two river modeling methods. GA parameter optimization with AOS mechanism shows little better performance, but not a big. However, this is a meaningful result. When we tried to compare with Mann-Whitney test, the performance with AOS was better than that at the 1% significance level. Parameter optimization with AOS mechanism better performed for all comparing measures; average, median and the best. However, the difference between two methods is small in the best, while average and median's differences are similar.

Figure 7.2 is two comparisons of actual value vs. predicted value from GA optimization system w/o and with AOS mechanism. As the absolute difference between two systems are small as shown at table 7.7, two figures show very similar trend. We could check both methods were able to well predict the actual data.

Figure 7.3 shows the change of operator rates over 500 runs. With considering

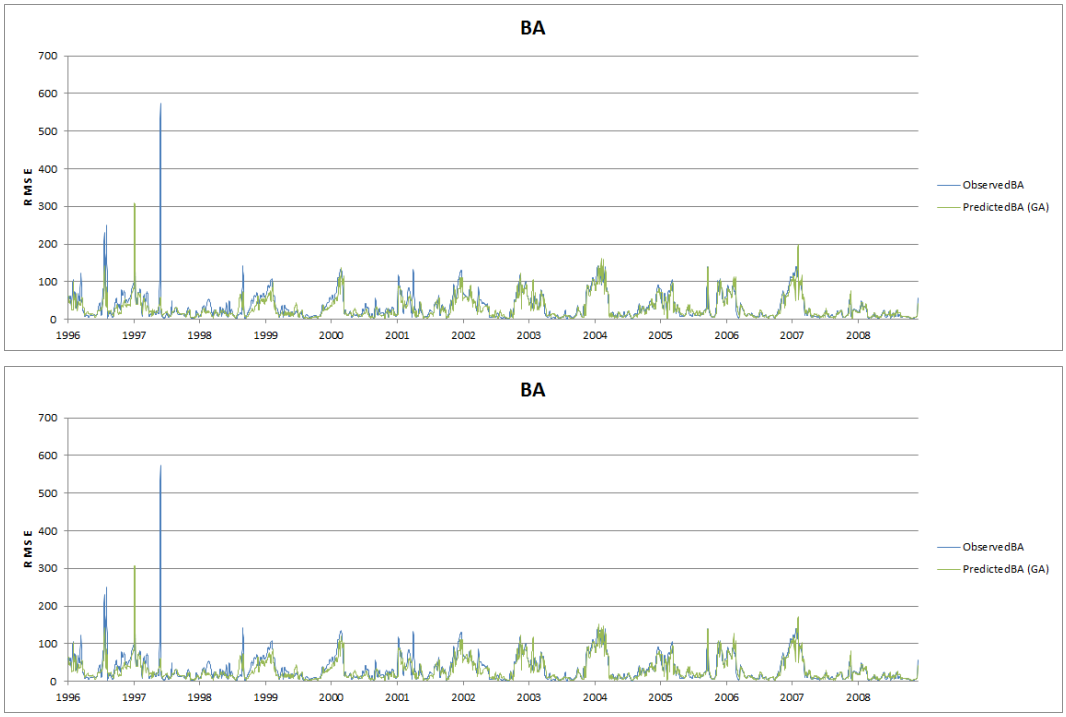

Figure 7.2: Chlorophyll a, Actual vs Predicted (with GA Parameter Fitting)
Top: without AOS, Bottom: AOS (r-AP)

fitness change, the time of fitness convergence and one-point crossover's grow up are matched. From 20 to 30 generation, it makes sense that the tendency of a run is changed; almost individuals are converged and small changes become to be more useful. 1-point crossover, reproduction mutation and range mutation, they shows more portions at figure 7.3, are operators which can cause more change even in late generations. From these facts, we can conclude AOS well-works in this problem and it helps to more-perform.

In addition, we set an additional experiment Opt from figure 7.3. We chose two operators, 1-point crossover and reproduction mutation for a run of Opt. As the

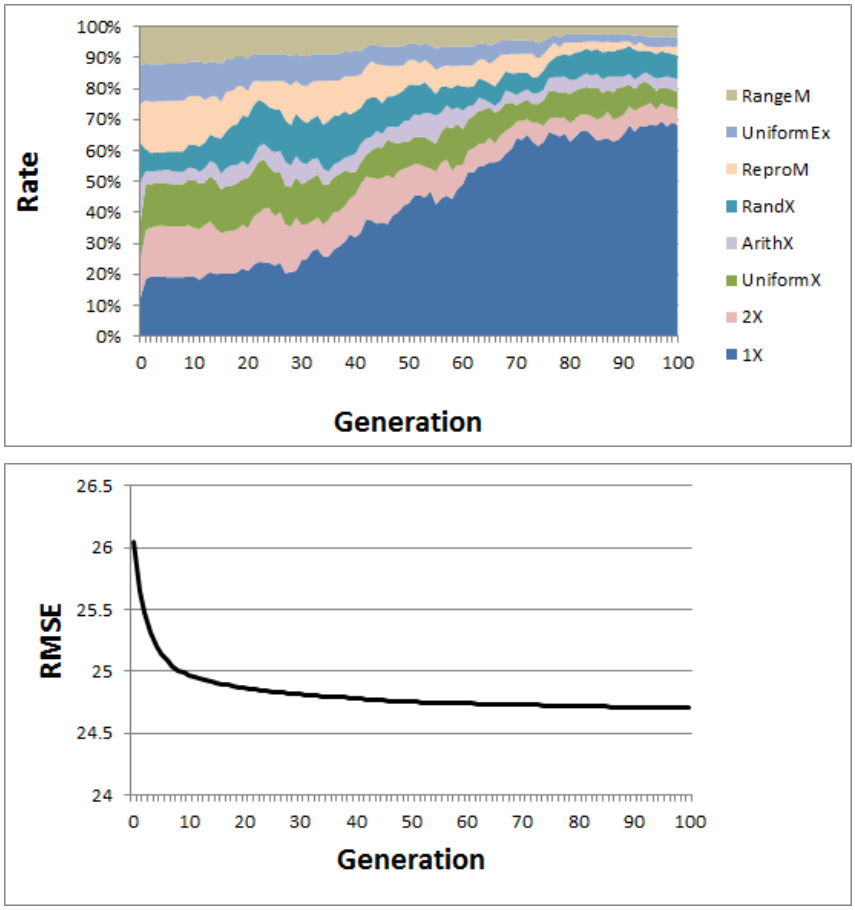

Figure 7.3: Change in Operator Application Rates
and Mean of Best Fitness for Parameter Optimization with GA

result, its overall performance is better than others (Table 7.7) in both training and test, at the 1% significance level of Mann-Whitney. However, AOS shows slightly better the best value.

Figure 7.4 shows the probability distribution of each parameter obtained from the 500 evolved process models (i.e. candidate solutions). It may be able to guide

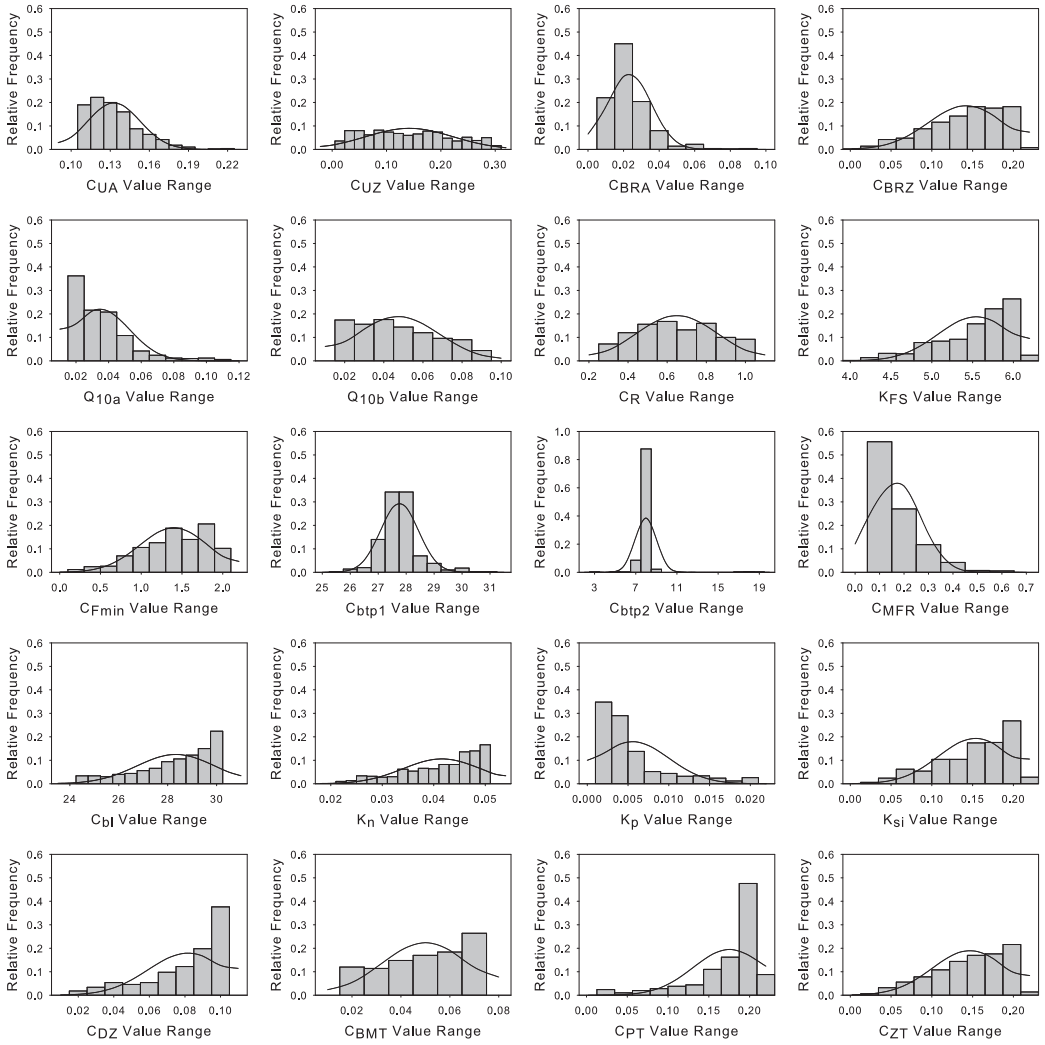

Figure 7.4: Probabilistic Distributions for Parameter Values from Best Evolved Process Model

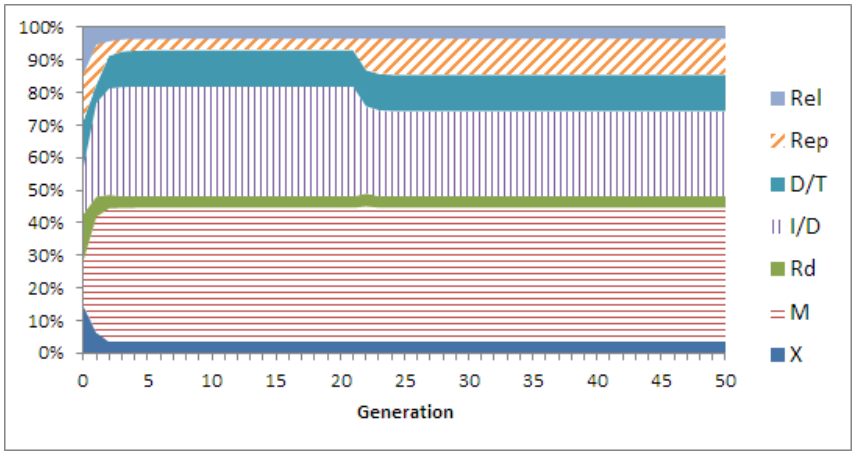

Figure 7.5: Change in Operator Application Ratesfor Modeling with TAG3P

management decisions for the river system.

7.2.2 Modeling

Table 7.8: Performance for Modeling with TAG3P

	Training			
RMSE	avg.	med.	stdev.	best.
w/o AOS	1.98E+16	1.20E+07	4.84E+16	25.5368
AOS	1099152	25.52535	2035420	24.9577

Table 7.8 shows the average, median, standard deviation and best value of RMSE for the training data[1]. Comparing to the result of parameter optimization, this work overally had quite large RMSE values. In addition, while RMSEs of paramter optimization are well regulated, it had pretty large standard deviation. Nevertheless AOS overally better-perform than w/o AOS, for all items. In particular, median value shows that more than half runs could find solutions with AOS but only less

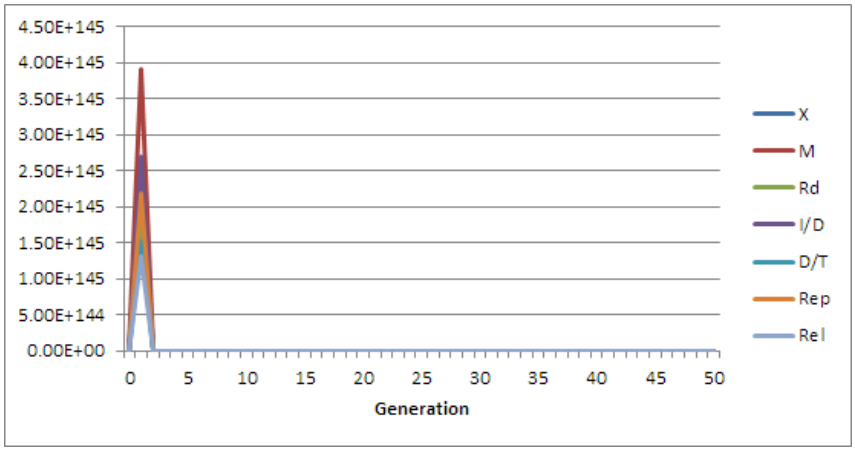

Figure 7.6: Change in the Operator Impactfor Modeling with TAG3P

than half of w/o AOS succeed.

Figure 7.5 shows the average change of operator rates. Comparing to previous experiments on TAG3P at chapter 4, 5 and 6, it rarely changed. Except the first few generations, there are little changes over whole generations. We could find a reason from figure 7.6.

Figure 7.6 describes the operator impacts which are evaluated for each generation. All operators received extremely large impact at the first generation. In parameter optimization, all parameter values to be optimized are in specific ranges, so individuals of an initial population of GA exist in a range. However, an initial population of GP has more freedom to be built; many elementary trees (extensions) can be added at extension points freely, so there is practically no boundary of fitness value. As a result, initial individuals had poor and wide-ranged fitenss values, and an extremely huge improvement on fitness, which affects to whole generations, is occurred at the first generation. Therefore, operators which received an extremely large impact at

[1]Because AOS is not compeltely applied to this work, we skipped RMSE for the test data

the first generation keep their rates highly over whole run. To overcome this problem, other approaches, such as ranking point or a normalized fitness, are required.

7.3 Summary

Section 7.2.1 and 7.2.2 shows that adaptive operator mechanism successfully is applied to real-world application. In both problems, methods with AOS statistically better-performed than methods without AOS. In addition, we could find a good operator combination from the result of AOS. However, AOS mechanism in GP includes a problem of extremely huge first impact which causes a wrong guideline to operators. For solving this problem, other approaches are required.

Chapter 8

Conclusion

8.1 Summary

This thesis has presented a number of issues about adaptive operator mechanism for genetic programming. Genetic programming has been already shown to well perform in many diverse problem domains. It has many parameters which affect to its performance. They enable the user to adapt the algorithm to the problem at hand. However, although they provide flexibility, it is difficult and complex to use them simultaneously, the necessity of AOS comes to the fore.

As the first step of this research, we successfully applied the AOS mechanism to GP systems in chapter 4. We used existing AOS algorithms; probability matching, adaptive pursuit and multi-armed bandit, in two kinds of GP systems: Linear tree adjoining grammar-guided genetic programming and the standard tree adjoining grammar-guided genetic programming. Compared to the standard GP system, these two systems have some good points. In particular, they have more useful genetic operators, while the standard GP is limited to subtree crossover and mutation as genetic operator. The TAG3P operators we used were subtree crossover, subtree

mutation, reproduction, insertion/deletion, duplication/truncation, point replacement and relocation. All operators had already shown their usefulness. However, it is difficult to determine suitable application rates for the operators without any prior knowledge when we have many operators. That is one reason we applied the AOS mechanism to TAG3P, rather than the standard GP. However, each AOS algorithms show some limitations. PM didn't distinguish genetic operators, because the operator impacts are too similar to each other. AP ignored the difference among operators, except the most effective one, so it is not suitable for many operators. MAB failed to apply for its sensitive parameter. Moreover, it chooses only one most effective operator at one time, so it is easily biased.

We suggested three new operator selection approaches in chapter 5; Powered Probability Matching, Adaptive Probability Matching, and recursive Adaptive Pursuit. All are variants of PM and AP, are designed to overcome previously mentioned drawbacks. PPM amplified the difference of impact between operators through exponentiation. It succeeded in increasing the difference, however the performance of PPM was similar to PM. On the other hand, APM and r-AP worked successfully. APM could distinguish not only the most effective operator but also rest operators. It suggested the most effective operator and several second effective operators. It showed good performance for many problems.r-AP uses the method of AP iteratively. So it easily emphasizes effective operators, but conversely it pressures the least effective operators to have minimum probability, P_{min}. So r-AP is effective in a smoothly changing environment, but its response is too slow for a rapidly changing environment. On the other hand, through empirical analysis of this experiment, we generated a deeper understanding of genetic operators. All operators have different effects, however we couldn't understand all effects through only theoretical analysis. Moreover it is difficult to understand the effects because GP usually operate on an

infinite solution space. The analysis showed the interaction between operators and individuals, and it showed the operator effects on diverse situation.

The evaluation of operator impact measures the impact of operators which is used to determine the operator rate. Many researchers have long addressed this issue and many different approaches have been developed. In chapter 6, we investigated four methods: changing rate for the amount of individual usage, ratio for the improvement of fitness, ranking point and pre-search structure. They require only small change, they have a large effect on performance. Different individual usage rates are preferred by different problems; higher rate tends to be better at more complex problem. However changing rate suggested a good solution for problems of both sides. The second one introduced various evaluation methods for fitness improvement. Fitness improvement is a usual measurement for evaluation of operator impact. We suggested three methods which use ratio value between two fitness values of child and the corresponding parents in various ways. Pairs-based ratio value usually has larger variance of the operator impact than groups-based ratio value, and latter showed small difference between the operator impacts. Therefore an operator which AOS online suggested, was easily changed over generation when we used groups-based ratio value. When we used child individual's fitness as a sort key, fine-tuning operators such as insertion/deletion, got a better impact value. On the other hand, ranking point is a refined fitness which is designed for avoiding an overflow which is caused by extremely large/small raw fitness value. The basic concept of ranking point is to change raw fitness values to rank-based points; we linearly changed in this thesis. Consequently ranking point improved the performance of PPM, however it made APM and r-AP be worse. Finally, we presented pre-search structure. The evaluation of operator impact gets the operator impact from the newly generated population, however because the population is generated by genetic operators with different

rates, there is a bias. Moreover the operator impact is evaluated by the preceding performance on the past generations, so it sometimes makes errors at dynamically changing environment. Pre-search structure is designed to overcome these problems by sampling, and it showed performance as good as a typical AOS.

Chapter 7 is applies the AOS mechanisms to a real-world application: the Nakdong River modeling. This work enabled us to verify the value of the usefulness of AOS mechanism in a real-world problem. It is consist of two experiments: parameter optimization with GA and modeling with TAG3P. Parameter optimization is a simple GA application, which finds the model parameters using GA. Each gene in chromosome is correspond to a parameter of the basic equation of the Nakdong River Model. On the other hand modeling is an extension work of parameter optimization. Including parameter optimization, it builds a good prediction model of algal bloom. In the result, methods which the AOS mechanism is applied, shows statistically better performance in both experiments. Moreover, a combination of genetic operators, which AOS empirically suggested, showed a good performance. However, while AOS was well applied to parameter optimization, modeling with GP showed a drawback. Individuals of the modeling work could have mostly no boundary fitness value, so extremely large improvement was occurred at the first generation and the operator impact at the first generation affected to whole generations. At most a half of runs failed to apply AOS mechanism in result, and other approaches, such as ranking point, are required for the more successful application of the AOS mechanisms.

8.2 Future Works

Pre-Search Structure and the Number of Samples

In chapter 6, we proposed pre-search structure and showed its usefulness. We tried using only one sample and 30% elite individuals in samples. The former was definitely bad, but the latter showed performance as good as normal AOS. However we expected pre-search structure has better performance than normal AOS, because the operator impact is evaluated on current population in pre-search structure. Therefore, even though pre-search structure is as good as normal AOS, it is little different in our purpose. However there still exists various ways to extend pre-search structure. The number of samples is one example of the ways. We expect an investigation into the relationship between the impact of operators and the number of samples improves performance of pre-search structure. Furthermore this research may be helpful in understanding the relationship between the impact of operators and population size.

Synergy Effect of Genetic Operators

Adaptive operator mechanisms in this thesis applied one genetic operator to generate one individual. Moreover, operator impact is evaluated for only one operator, however, multiple operators are usually applied to one individual at the same generation in a typical GP; both crossover and mutation are applied to one individual in one generation. When multiple operators are applied together to one individual, a synergy effect may be observed (Spears (1995); Hong et al. (1995); Yoon and Moon (2002)). Although one operator on its own may not have a good impact, it is possible to observe a synergy effect. Moreover, a similar synergy effect can sometimes be observed between generations. A simple and brutal approach for synergy effect is to consider all combinations of all genetic operators, however it requires an expensive

cost. More sophisticated method is to use restricted combinations: combinations with one primary operator or combinations of two categorized operators . In our future work, we hope to investigate synergy effects between genetic operators.

Multiple Resources for the Evaluation of Operator Impact and Multi-Objective Optimization

We used two kinds of resource for the evaluation: accuracy and structure information. They were both based on the fitness, so the objective of problem and resources were matched. However many GP problems incorporate preferences unrelated to fitness, such as size or depth of solution; smaller sized solution is better if it has the same fitness. We expect that a method of the evaluation of operator impact which considers fitness and preferences together, may be useful to find more sophisticated optimal solutions. For this research, we will use multi-objective optimization (MOO, Coello Coello (2006)). For example, accuracy-based fitness and complexity-based preference could be set as the objective function of MOO, which is a key part of the evaluation of operator impact.

Adaptive Mechanism for Whole Parameters in GP

We investigated adaptive mechanism for GP, in particular adaptive operator selection, however GP has many other parameters which are not related to genetic operator. Population size, a maximum number of generations and selection pressure are those parameters and they also have a large effect on performance. In our future work, we hope to extend this research to the parameters. The research of presearch structure and the number of samples may be helpful to investigate adaptive population size.

Appendix A

More Information on Grammars

A.1 Trigonometric Problem

Table A.1: Context Free Grammar for the Trigonometric Problem

EXP	$\rightarrow$	EXP OP EXP | PREOP EXP | VAR
OP	$\rightarrow$	+ | - | $\times$ | $\div$
PREOP	$\rightarrow$	*sin*
VAR	$\rightarrow$	X | 1

Table A.1 and figure A.1 depict the productions of the context free grammar, and the corresponding elementary trees for the tree adjoining grammar, for the trigonometric problem (in the figure, T denotes a lexicon that can be substituted by a member of the set $\{X, 1\}$).

A.2 2-Box Problem

Table A.2 and figure A.2 depict the productions of the context free grammar, and the corresponding elementary trees for the tree adjoining grammar, for the 2-Box

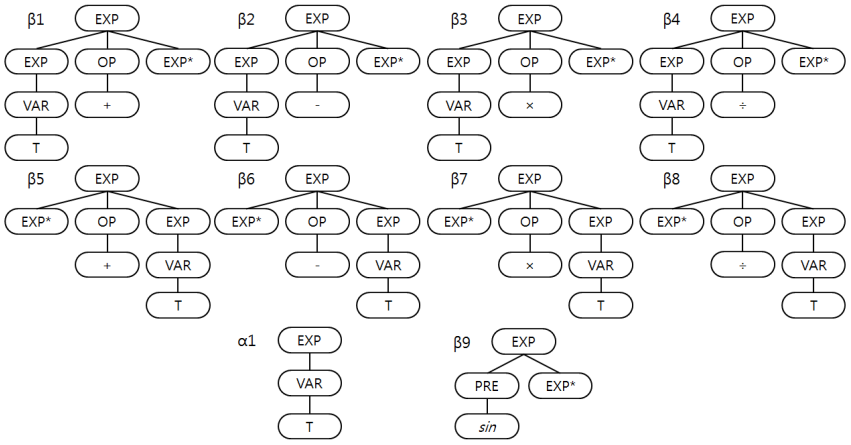

Figure A.1: Elementary Trees for the Trigonometric Problem

Table A.2: Context Free Grammar for the 2-Box Problem

EXP	$\rightarrow$	EXP OP EXP | VAR
OP	$\rightarrow$	$+$ | - | $\times$ | $\div$
VAR	$\rightarrow$	W | H | L | w | h | l

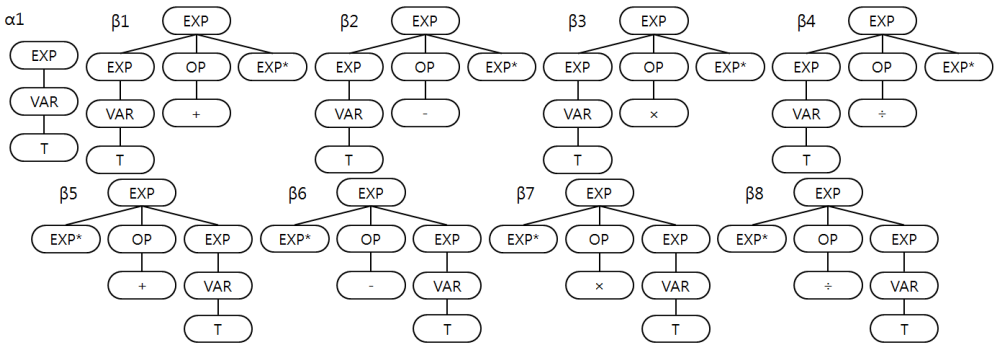

Figure A.2: Elementary Trees for the 2-Box Problem

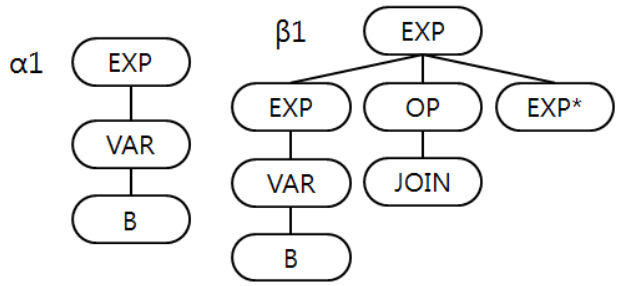

Figure A.3: Elementary Trees for the the Majority and Order Problems

problem (in the figure, T denotes a lexicon that can be substituted by a member of the set $\{W, H, L, w, h, l\}$).

A.3 Majority and Order Problems

Table A.3: Context Free Grammar for the Majority and Order Problems

EXP	$\rightarrow$	EXP OP EXP | VAR
OP	$\rightarrow$	$JOIN$
VAR	$\rightarrow$	$P_1 \mid P_2 \mid \ldots \mid P_n \mid N_1 \mid \ldots \mid N_n$

Table A.3 and figure A.3 depict the productions of the context free grammar, and the corresponding elementary trees for the tree adjoining grammar, for the majority and order problems (in the figure, T denotes a lexicon that can be substituted by a member of the set $\{P_1, P_2, ..., P_n, N_1, ..., N_n\}$).

Table A.4: Context Free Grammar for the Daida Problem

EXP	$\rightarrow$	EXP OP EXP | VAR
OP	$\rightarrow$	*JOIN*
VAR	$\rightarrow$	*X*

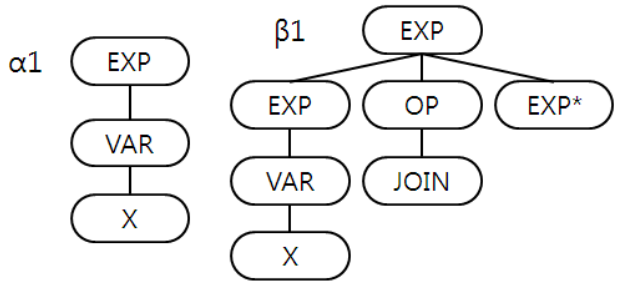

Figure A.4: Elementary Trees for the DAIDA Problem

A.4 DAIDA problem

Table A.4 and figure A.4 depict the productions of the context free grammar, and the corresponding elementary trees for the tree adjoining grammar, for the Daida problem.

Appendix B

Supplementary Figures

B.1 Change in Operator Application Rates on Preliminary Experiment (LTAG3P)

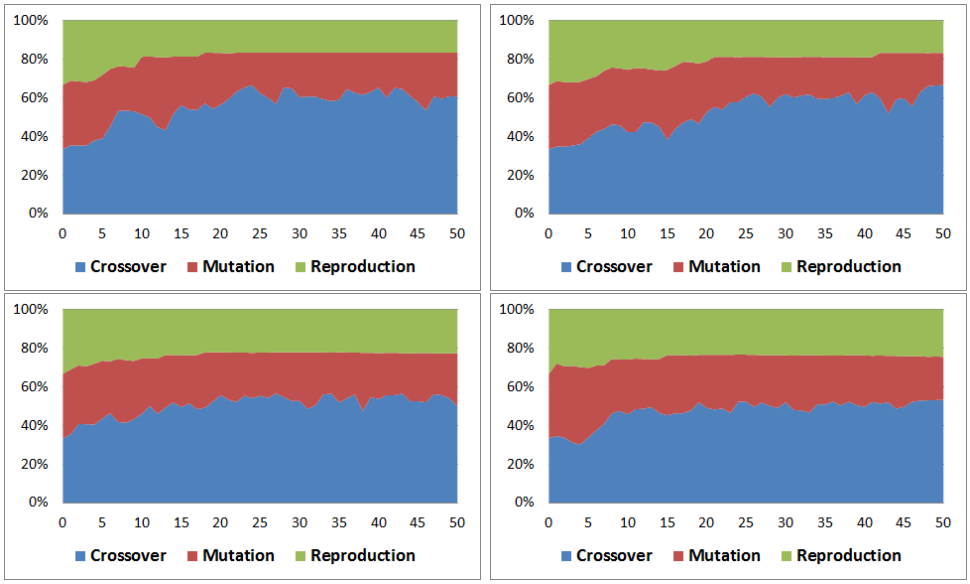

Figure B.1: Change in Operator Application Rates on PM (LTAG3P)

From Top Left, F_6 and F_9, Quintic and Sextic

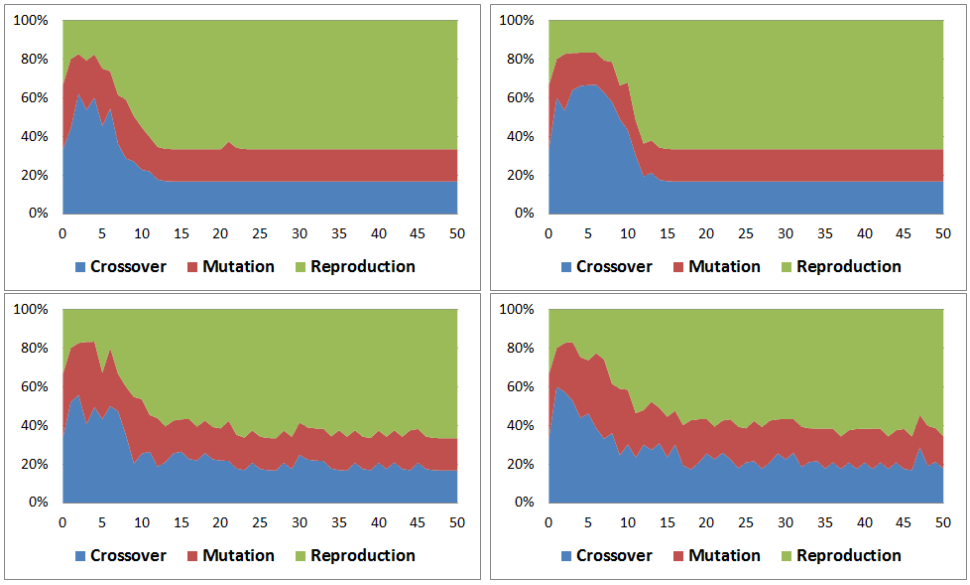

Figure B.2: Change in Operator Application Rates on AP (LTAG3P)

From Top Left, F_6 and F_9, Quintic and Sextic

B.2 Mean of Best Fitness on Preliminary Experiment (LTAG3P)

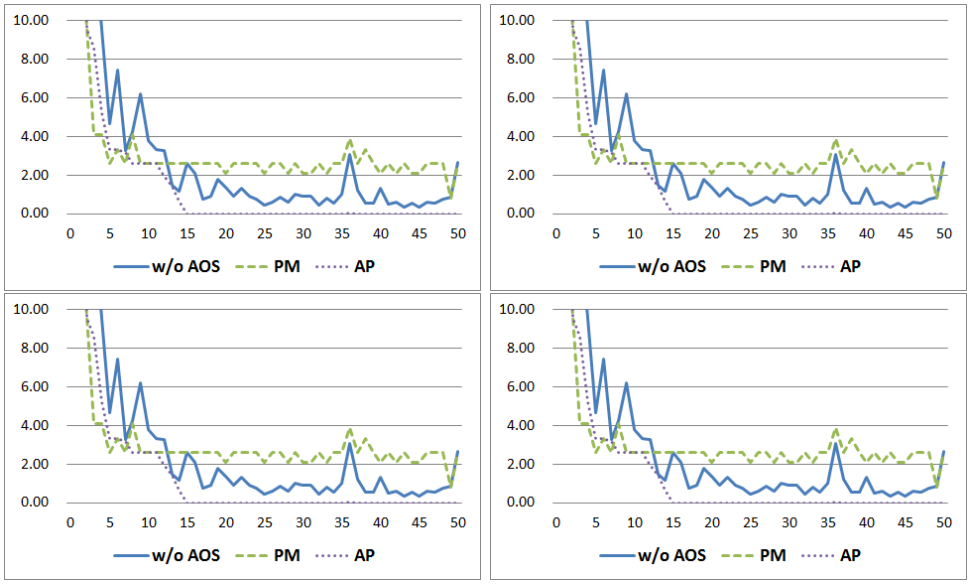

Figure B.3: Mean of Best Fitness (LTAG3P)

From Top-Left, F_6 and F_9, Quintic and Sextic

B.3 Change in Operator Application Rates on Preliminary Experiment (TAG3P)

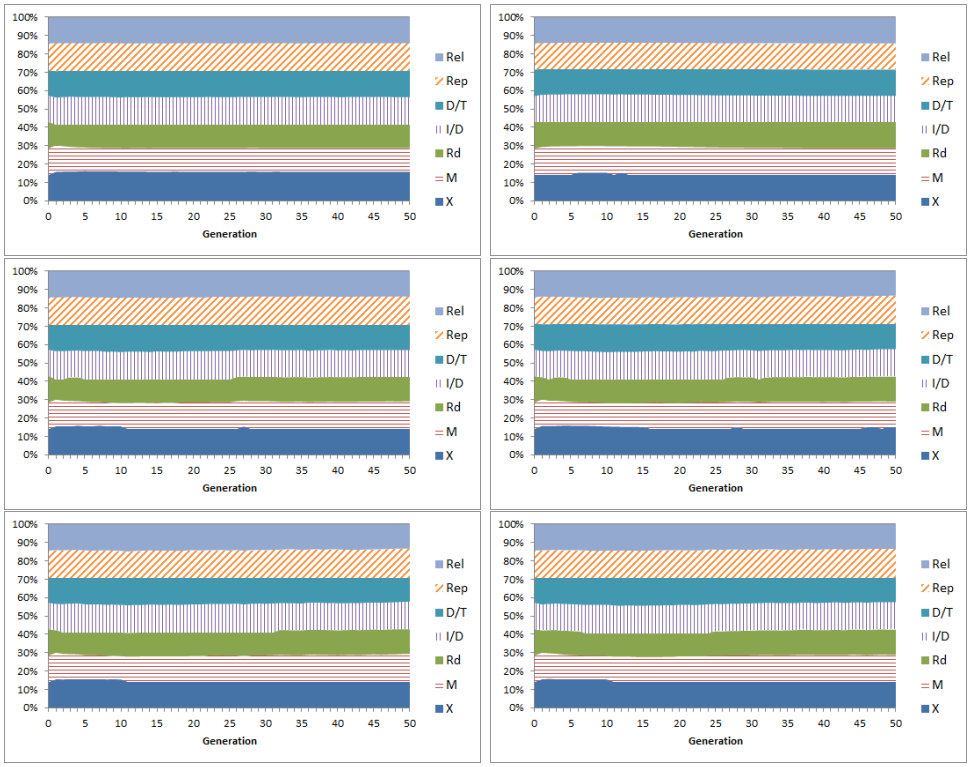

Figure B.4: Change in Operator Application Rates on PM #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9

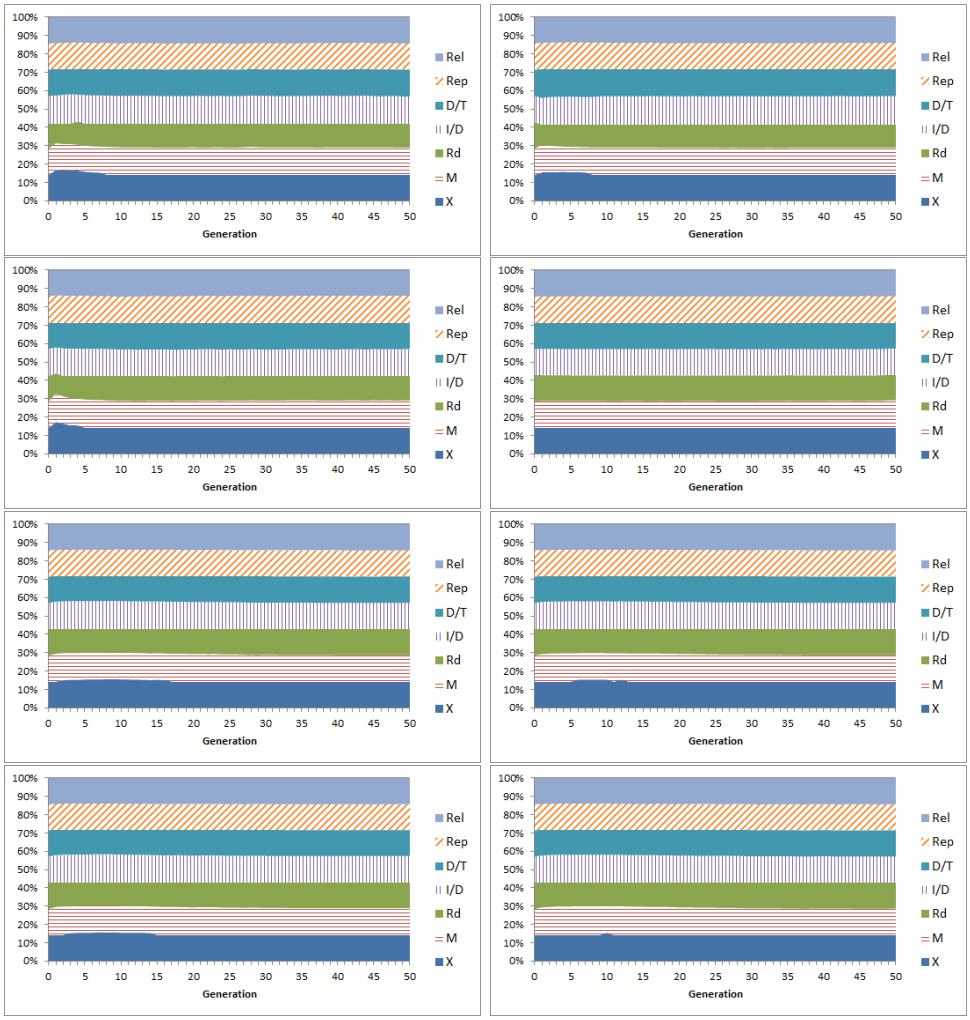

Figure B.5: Change in Operator Application Rates on PM #2
From Top-Left, Quintic and Sextic, Trigonometric and 2-Box,
M_{25} and M_{30}, O_{25} and O_{30}

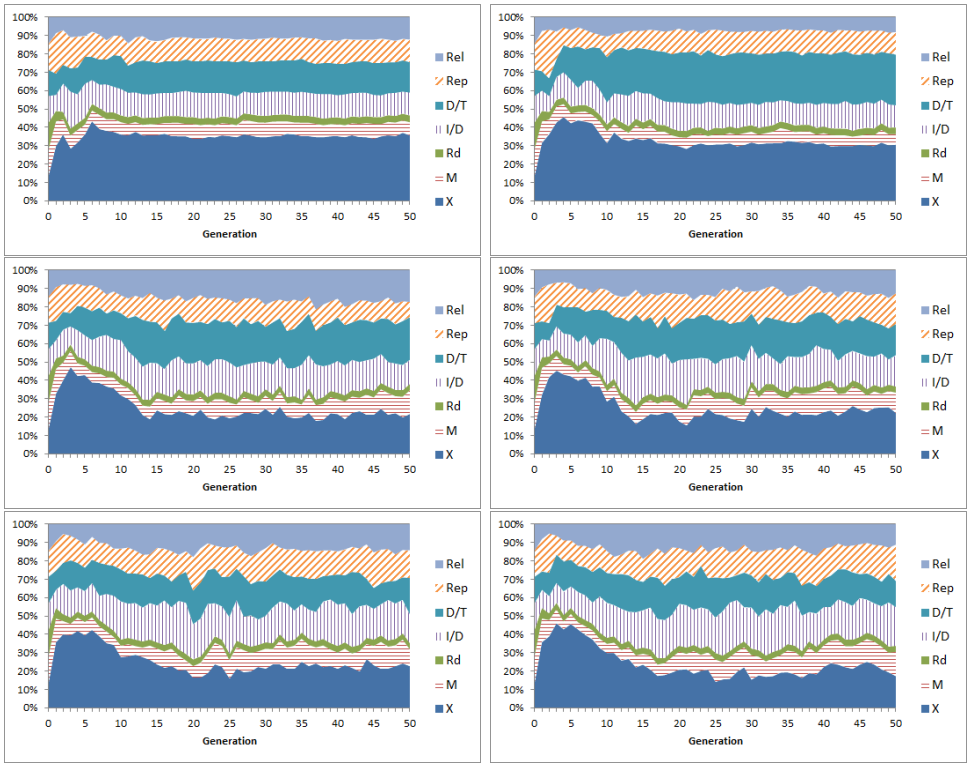

Figure B.6: Change in Operator Application Rates on AP #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9

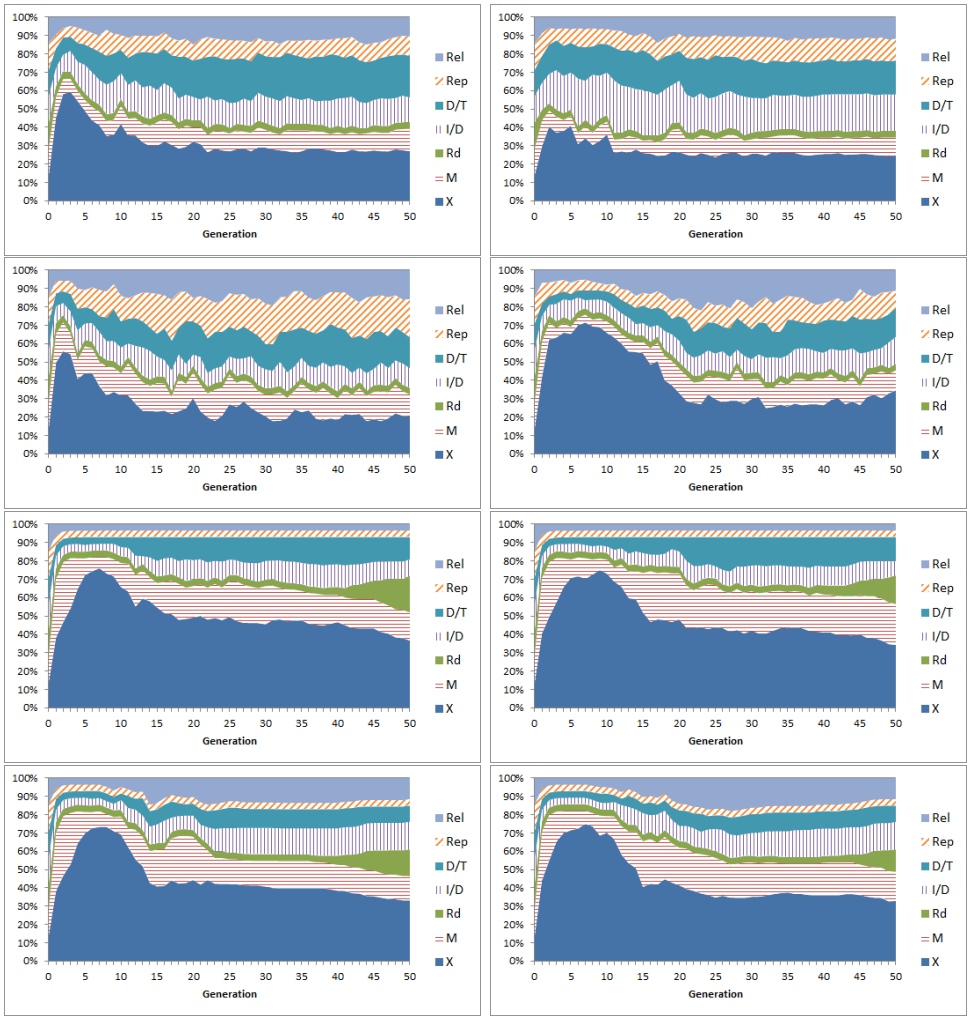

Figure B.7: Change in Operator Application Rates on AP #2
From Top-Left, Quintic and Sextic, Trigonometric and 2-Box,
M_{25} and M_{30}, O_{25} and O_{30}

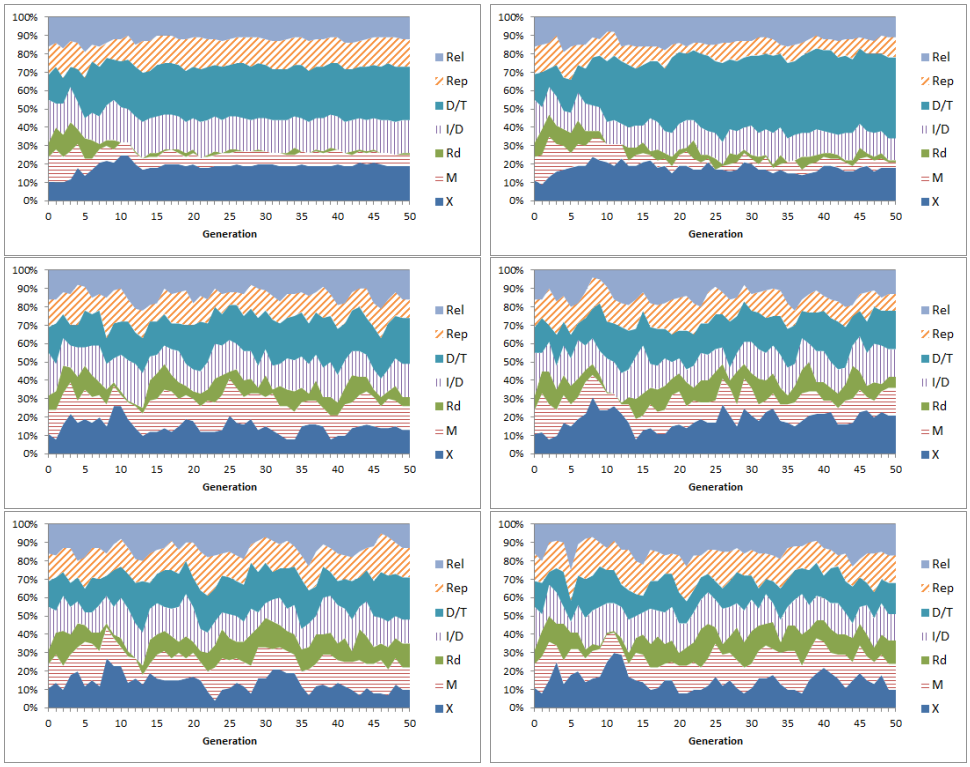

Figure B.8: Change in Operator Application Rates on MAB #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9

Figure B.9: Change in Operator Application Rates on MAB #2
From Top-Left, Quintic and Sextic, Trigonometric and 2-Box,
M_{25} and M_{30}, O_{25} and O_{30}

B.4 Mean of Best Fitness on Preliminary Experiment (TAG3P)

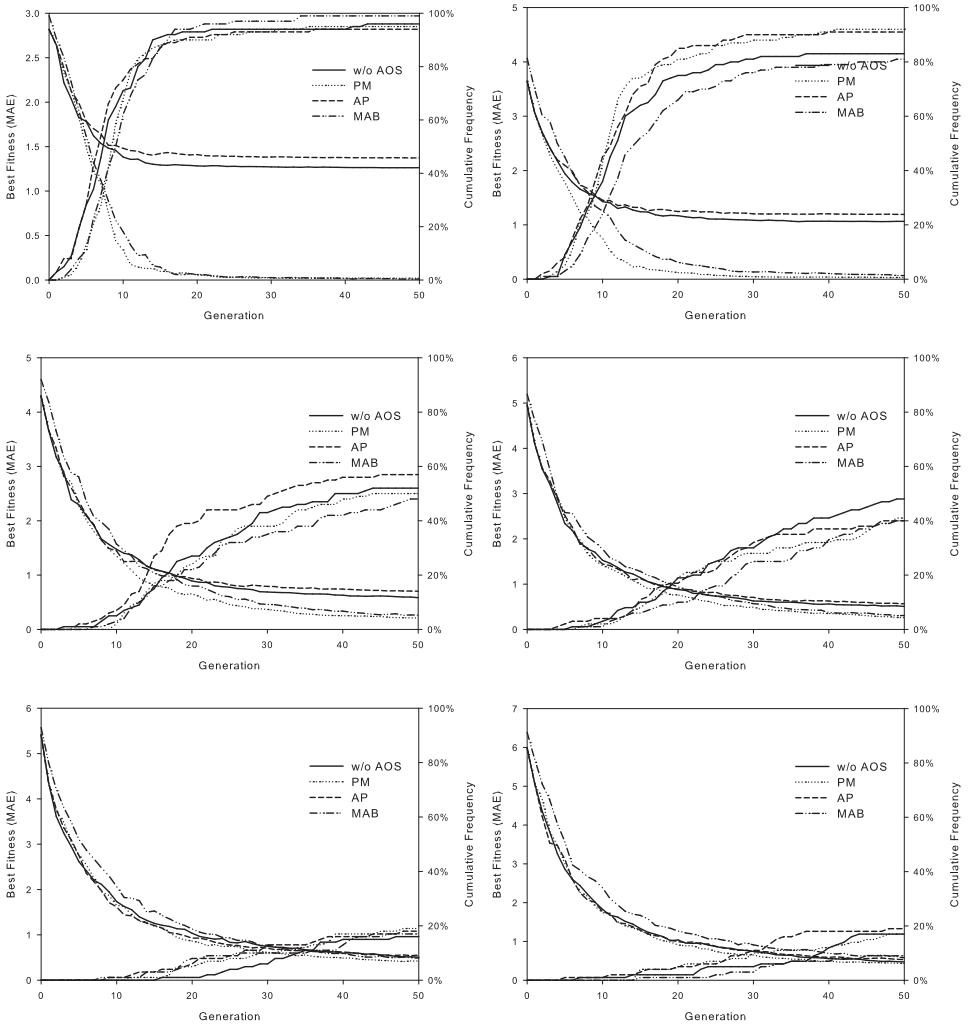

Figure B.10: Mean of Best Fitness (TAG3P) #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9

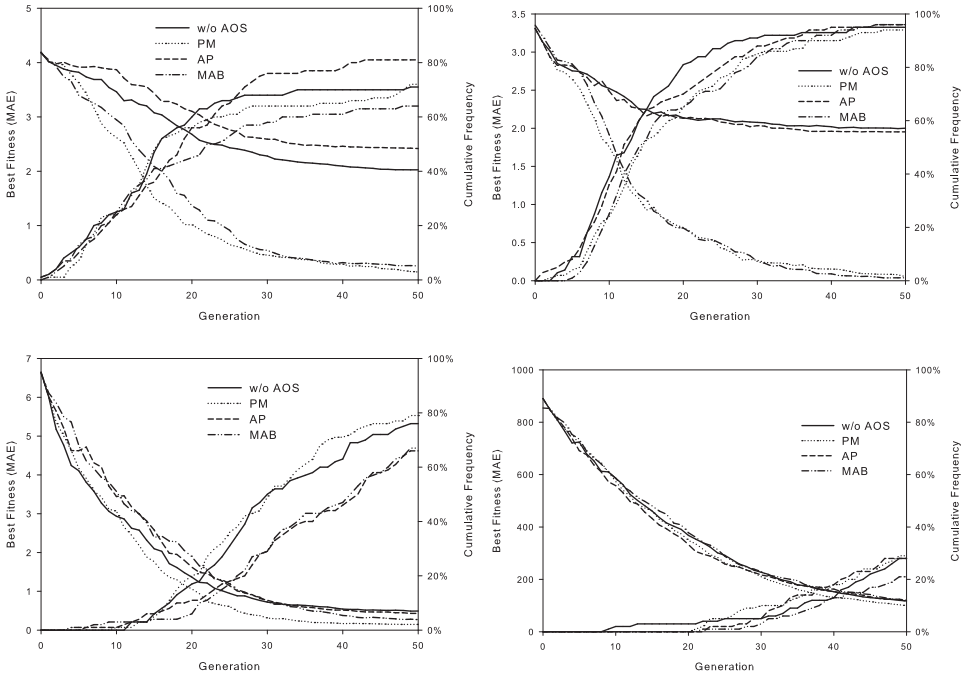

Figure B.11: Mean of Best Fitness (TAG3P) #2

From Top-Left, Quintic and Sextic, Trigonometric and 2-Box

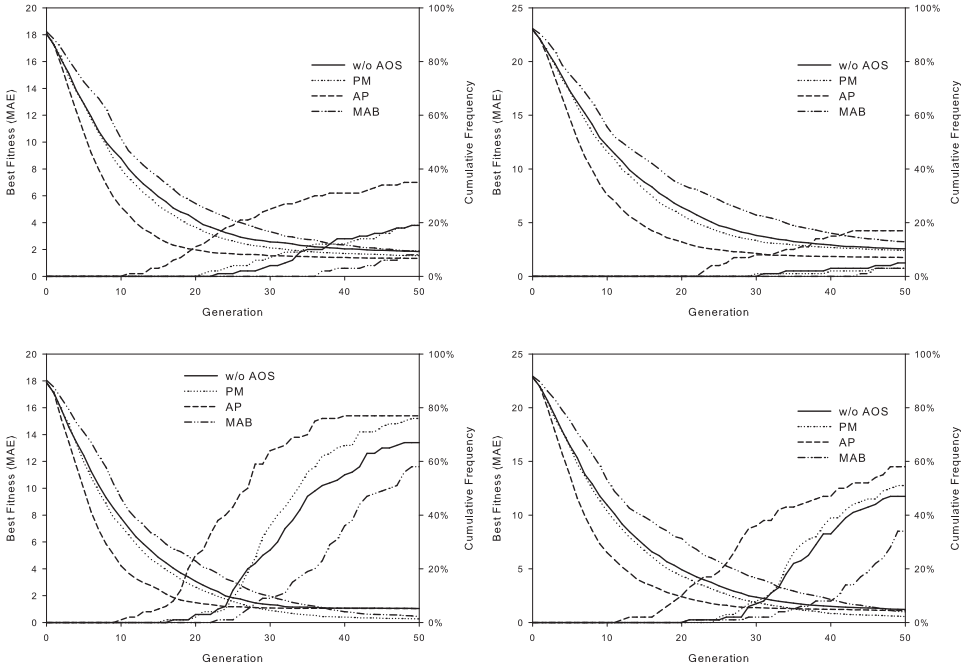

Figure B.12: Mean of Best Fitness (TAG3P) #3

From Top-Left, M_{25} and M_{30}, O_{25} and O_{30}

B.5 Change in Operator Application Rates on Operator Selection

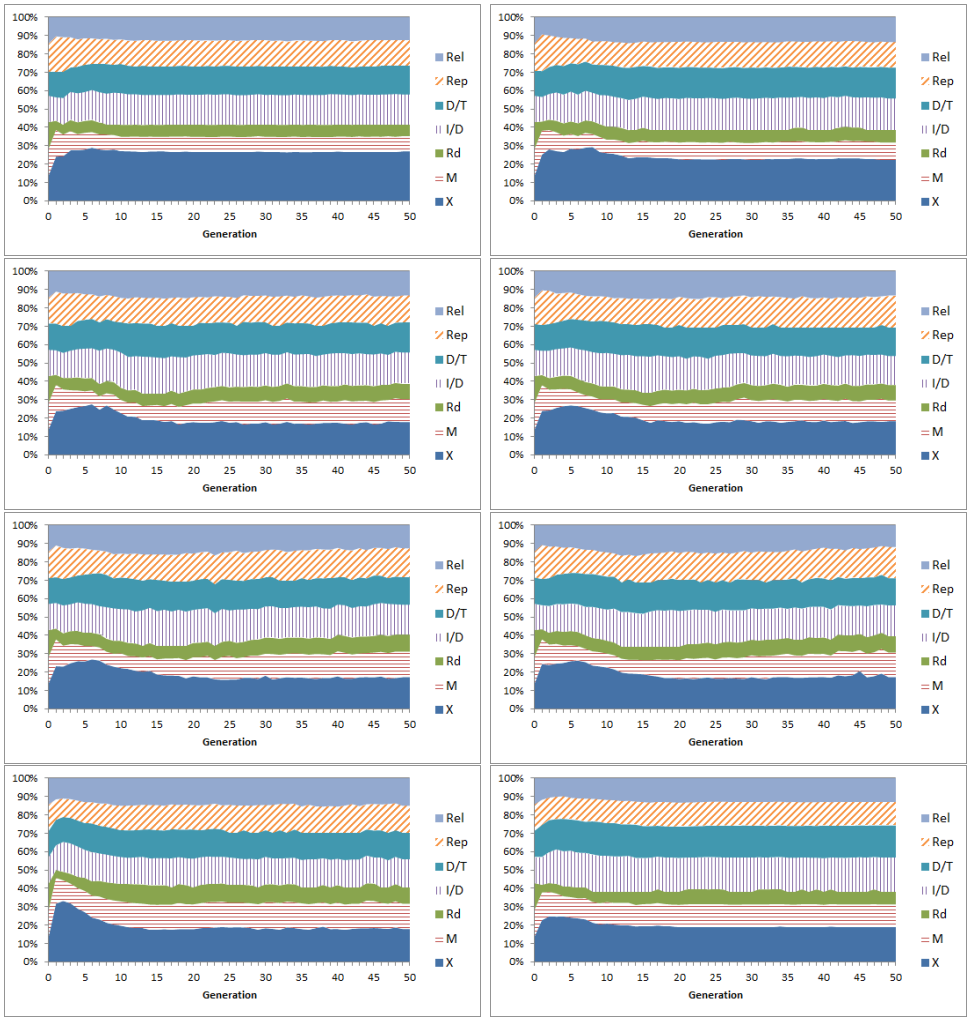

Figure B.13: Change in Operator Application Rates on PPM #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9, Quintic and Sextic

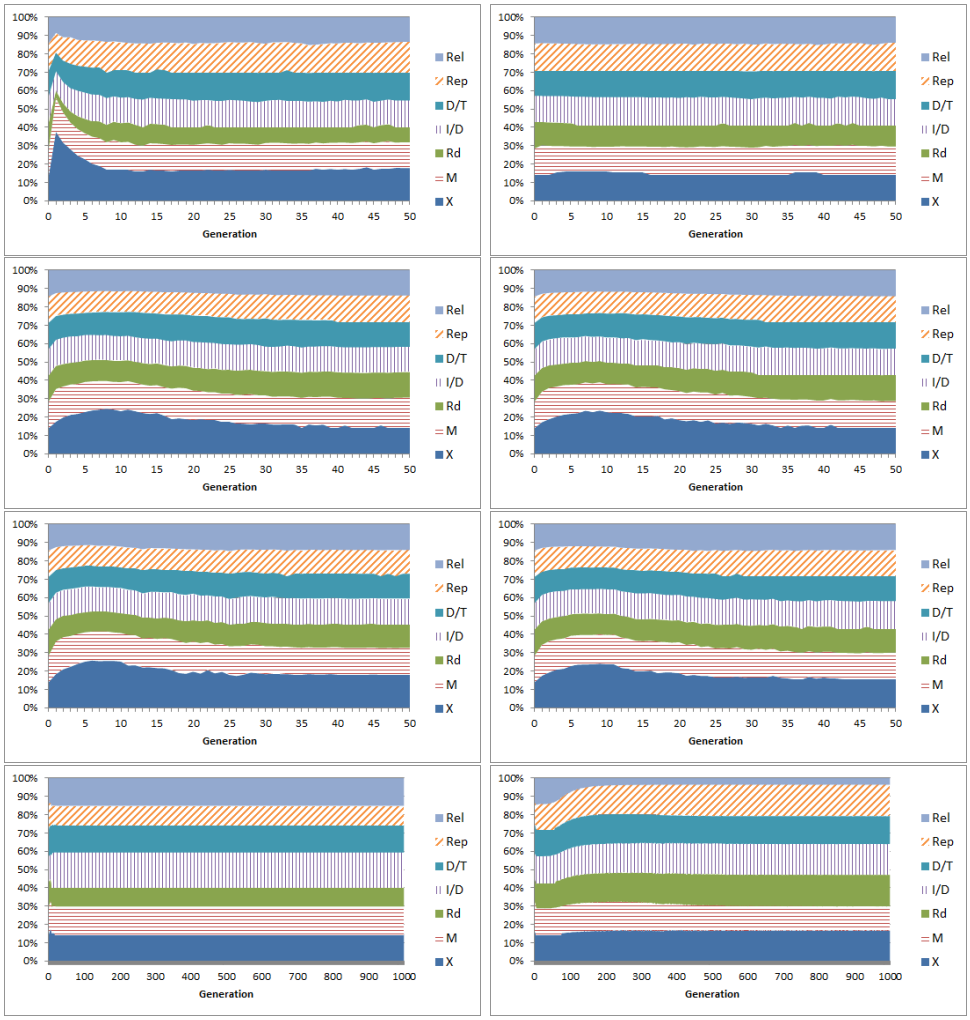

Figure B.14: Change in Operator Application Rates on PPM #2

From Top-Left, Trigonometric and 2-Box, M_{25} and M_{30}, O_{25} and O_{30}, D_N and D_W

Figure B.15: Change in Operator Application Rates on APM #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9, Quintic and Sextic

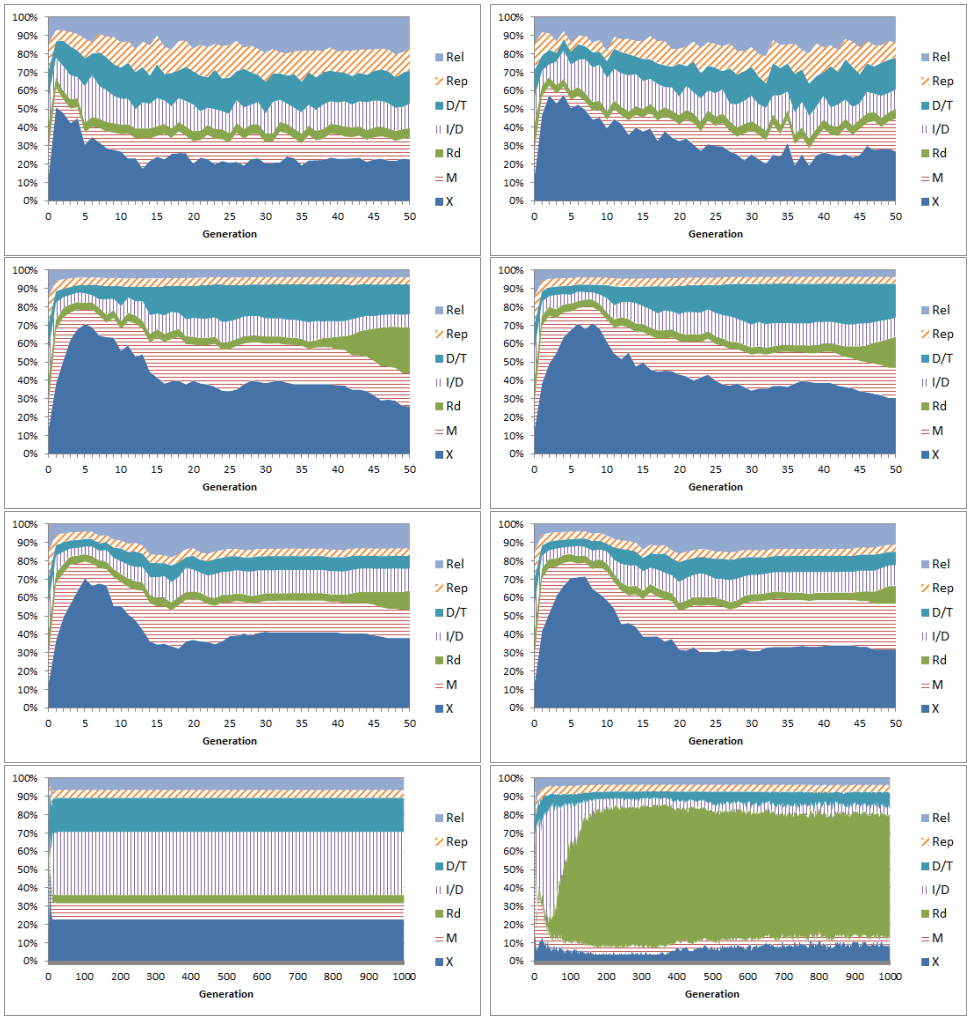

Figure B.16: Change in Operator Application Rates on APM #2

From Top-Left, Trigonometric and 2-Box, M_{25} and M_{30}, O_{25} and O_{30}, D_N and D_W

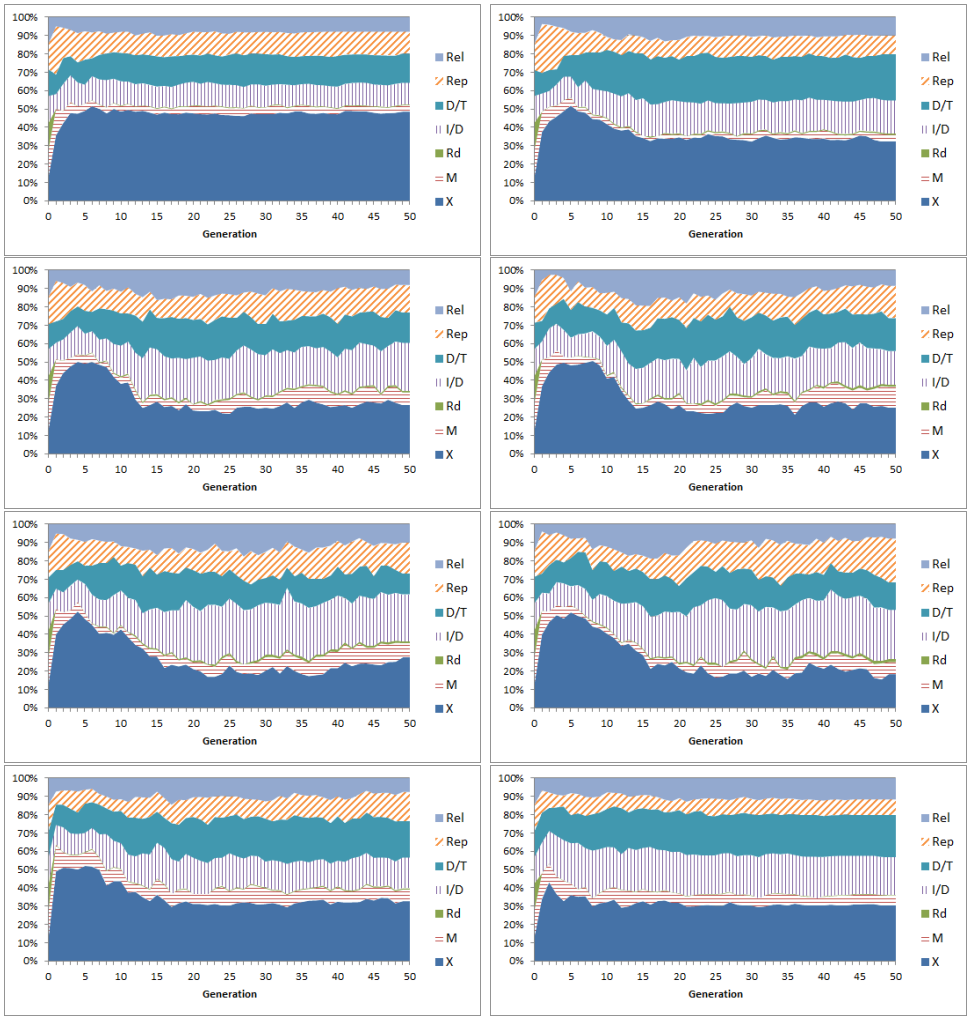

Figure B.17: Change in Operator Application Rates on rAP #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9, Quintic and Sextic

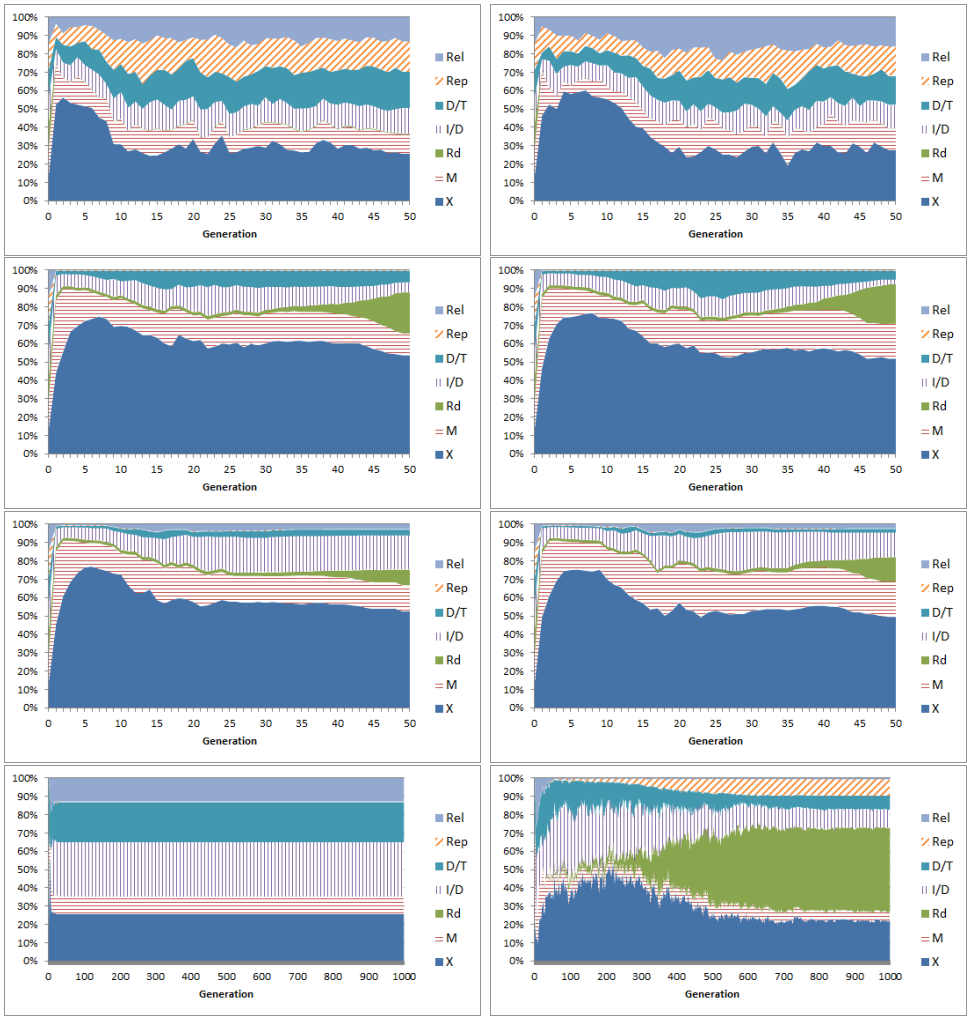

Figure B.18: Change in Operator Application Rates on rAP #2

From Top-Left, Trigonometric and 2-Box, M_{25} and M_{30}, O_{25} and O_{30}, D_N and D_W

B.6 Mean of Best Fitness on Operator Selection

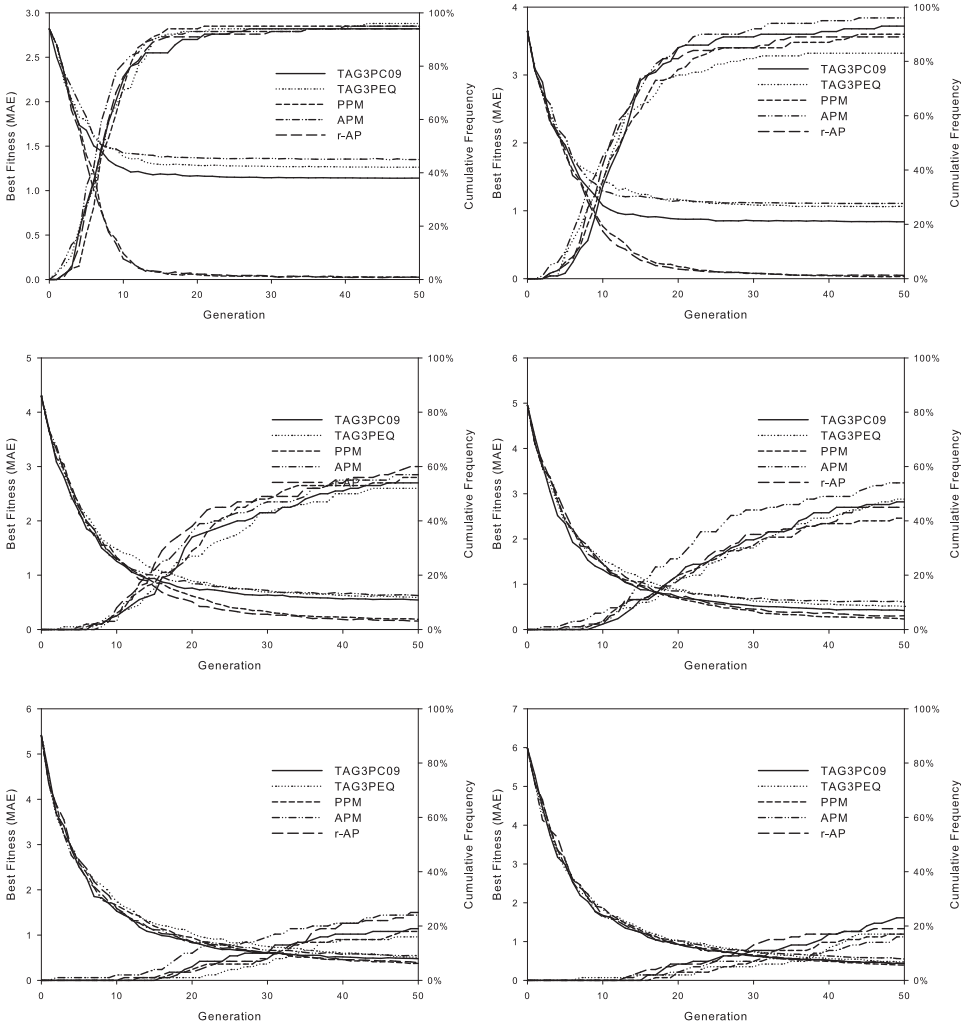

Figure B.19: Mean of Best Fitness #1

From Top-Left, F_4 and F_5, F_6 and F_7, F_8 and F_9

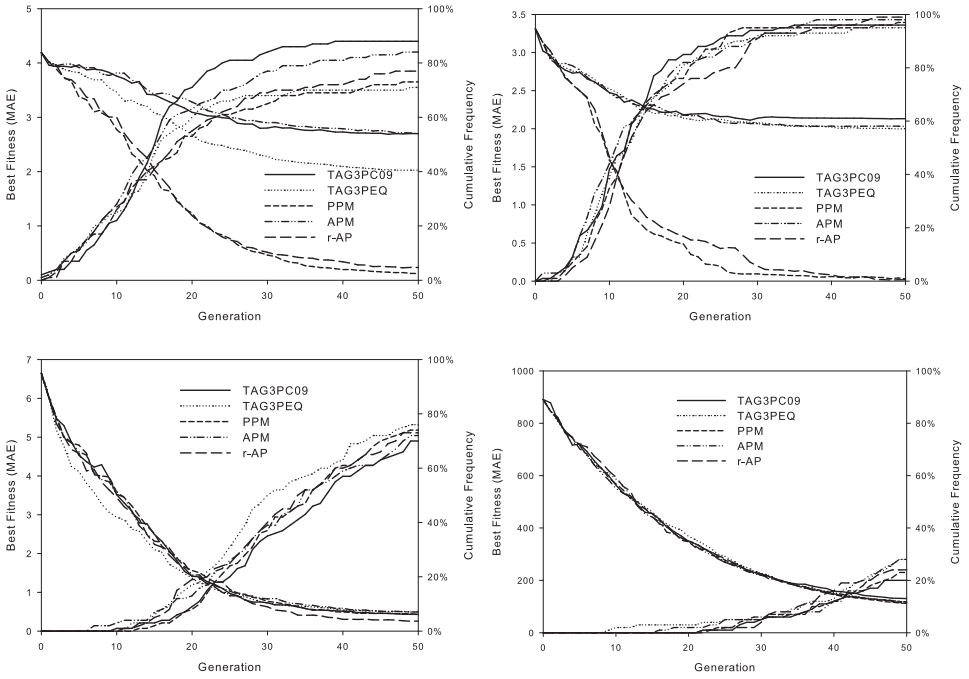

Figure B.20: Mean of Best Fitness #2

From Top-Left, Quintic and Sextic, Trigonometric and 2-Box

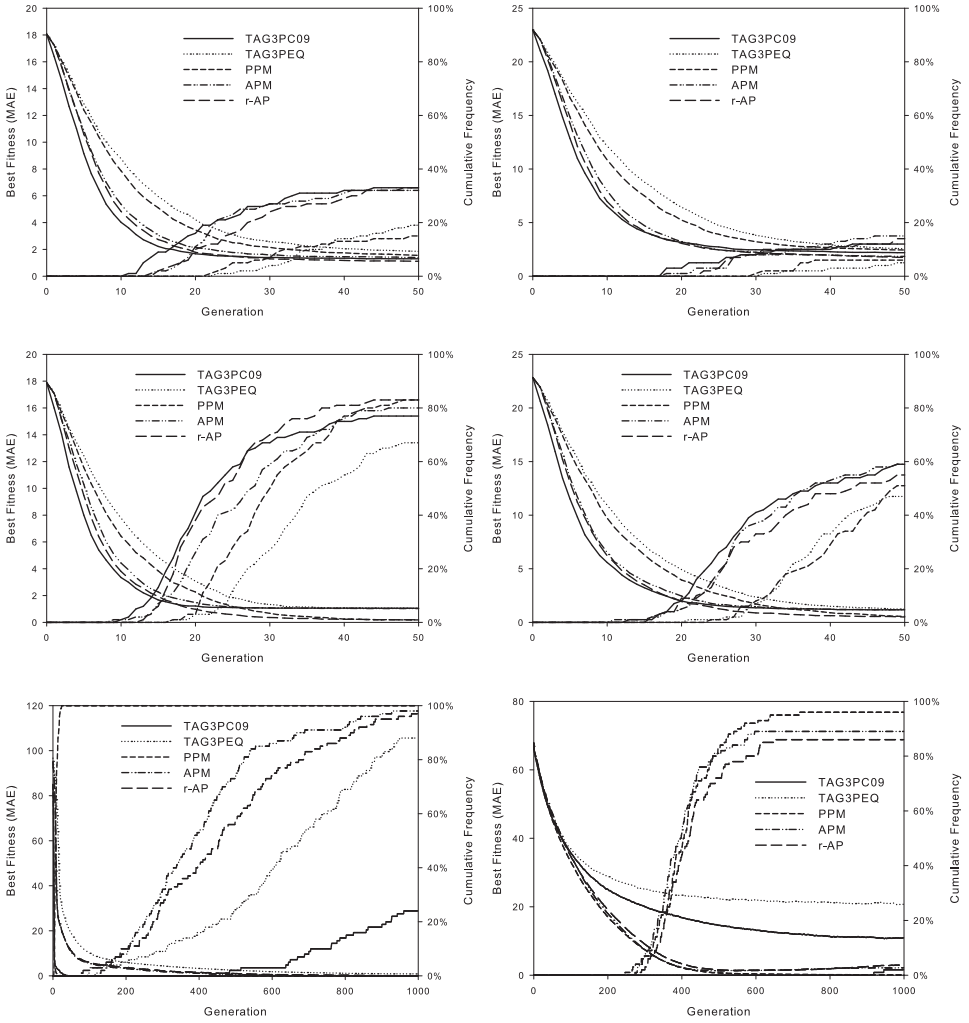

Figure B.21: Mean of Best Fitness #3

From Top-Left, M_{25} and M_{30}, O_{25} and O_{30}, D_N and D_W

Bibliography

P.J. Angeline. Subtree crossover causes bloat. In *Proceeding of 3rd Annual Conference on Genetic Programming 1998 (GP98)*, pages 745–752. Morgan Kaufmann, 1998.

G.B. Arhonditsis and M.T. Brett. Eutrophication model for Lake Washington (USA) Part I. model description and sensitivity analysis. *Ecological Modelling*, 187(2-3): 140–178, 2005.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

T. Bäck, D. Fogel, and Z. Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*, volume 1. Taylor & Francis, 2000a.

T. Bäck, D. Fogel, and Z. Michalewicz. *Evolutionary computation 2: advanced algorithms and operations*, volume 2. Taylor & Francis, 2000b.

W. Banzhaf, P. Nordin, R.E. Keller, and F.D. Francone. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, 1997.

H.J.C. Barbosa and AM Sá. On adaptive operator probabilities in real coded genetic algorithms. In *Workshop on Advances and Trends in Artificial Intelligence for Problem Solving (SCCC'00)*, 2000.

L. Beadle and C.G. Johnson. Semantically driven crossover in genetic programming. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 111–116. IEEE, 2008.

A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

K.-J. Cho and J.-K. Shin. Growth and nutrient kinetics of some algal species isolated from the Naktong River. *ALGAE*, 13(2):235–240, 1998.

C.A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, 2006.

T.M. Cole and E.M. Buchak. CE-QUAL-W2: A two-dimensional, laterally averaged, hydrodynamic and water quality model, version 2.0. user manual. instruction report EL-95-1. Technical report, DTIC Document, 1995.

L. DaCosta, Á. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic Multi-Armed bandits. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 2008)*, pages 913–920. ACM, 2008.

J. Daida, H. Li, R. Tang, and A. Hilss. What makes a problem GP-hard? validating a hypothesis of structural causes. In *Proceedings of the 5th Annual Conference on Genetic and Evolutionary Computation (GECCO 2003)*, pages 1665–1677. ACM, 2003.

L. Davis. Adapting operator probabilities in genetic algorithms. In *Proceeding of the 3rd International Conference on Genetic Algorithms*, pages 61–69, 1989.

K. De Jong. Parameter setting in EAs: a 30 year perspective. *Parameter Setting in Evolutionary Algorithms*, pages 1–18, 2007.

A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J.E. Smith. Parameter control in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pages 19–46. Springer, 2007.

E. Everbecq, V. Gosselain, L. Viroux, and J.-P Descy. POTAMON: a dynamic model for predicting phytoplankton composition and biomass in lowland rivers. *Water research*, 35(4):901–912, 2001.

Á. Fialho and M. Schoenauer. Adaptive operator selection in EAs with Extreme-Dynamic Multi-Armed bandits. In *SLS-DS 2009: Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, pages 11–15, 2009.

Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. *Proceedings of the 10th international conference on Parallel Problem Solving from Nature-PPSN X*, pages 175–184, 2008.

Á. Fialho, M. Schoenauer, and M. Sebag. Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009)*, pages 779–786. ACM, 2009.

Á. Fialho, M. Schoenauer, and M. Sebag. Toward comparison-based adaptive operator selection. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010)*, pages 767–774. ACM, 2010.

D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley, 1989.

D.E. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*, 5(4):407–425, 1990.

D.E. Goldberg and U.M. O'Reilly. Where does the good stuff go, and why? how contextual semantics influences program structure in simple genetic programming. *Genetic Programming*, pages 16–36, 1998.

W. Gong, Á. Fialho, and Z. Cai. Adaptive strategy selection in differential evolution. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010)*, pages 409–416. ACM, 2010.

K. Ha, E.-A. Cho, H.-W. Kim, and G.-J. Joo. *Microcystis* bloom formation in the lower Nakdong River, South Korea: importance of hydrodynamics and nutrient loading. *Marine and Freshwater Research*, 50:89–94, 1999.

K. Ha, M.-H. Jang, and G.-J. Joo. Winter *Stephanodiscus* bloom development in the Nakdong River regulated by an estuary dam and tributaries. *Hydrobiologia*, 506:221–227, 2003.

D.V. Hinkley. Inference about the change-point in a sequence of random variables. *Biometrika*, 57(1):1–17, 1970.

C.W. Ho, K.H. Lee, and K.S. Leung. A genetic algorithm based on mutation and crossover with adaptive probabilities. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, volume 1, pages 768–775. IEEE, 1999.

N.X. Hoai. *A Flexible Representation for Genetic Programming: Lessons from Natural Language Processing*. PhD thesis, University of New South Wales, Australian Defence Force Academy, 2004.

N.X. Hoai, R.I. McKay, and D. Essam. Some experimental results with tree adjunct grammar guided genetic programming. In *Proceedings of 5th European Conference on Genetic Programming (EuroGP 2002)*, pages 104–121, 2002.

N.X. Hoai, R.I. McKay, and D. Essam. Representation and structural difficulty in genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2): 157–166, 2006.

J.H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

I. Hong, A.B. Kahng, and B.-R. Moon. Exploiting synergies of multiple crossovers: initial studies. In *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 1995)*, pages 245–250. IEEE, 1995.

P. Hongping and M. Jianyi. Study on the algal dynamic model for West Lake, Hangzhou. *Ecological Modelling*, 148(1):67–77, 2002.

K.S. Jeong, D.K. Kim, and G.J. Joo. Delayed influence of dam storage and discharge on the determination of seasonal proliferations of microcystis aeruginosa and stephanodiscus hantzschii in a regulated river system of the lower nakdong river (south korea). *Water research*, 41(6):1269–1279, 2007.

A. K Joshi, L. S Levy, and M. Takahashi. Tree adjunct grammars*. *Journal of computer and system sciences*, 10(1):136–163, 1975.

A.K. Joshi and Y. Schabes. Tree-adjoining grammars. *Handbook of formal languages*, 3:69–124, 1997.

B.A. Julstrom. What have you done for me lately? adapting operator probabilities

in a steady-state genetic algorithm. In *Proceeding of 6th International Conference on on Genetic Algorithms (ICGA 1995)*, pages 81–87. Morgan Kaufmann, 1995.

D.-K. Kim, H. Cao, K.-S. Jeong, F. Recknagel, and G.-J. Joo. Predictive function and rules for population dynamics of *Microcystis aeruginosa* in the regulated Nakdong River (South Korea), discovered by evolutionary algorithms. *Ecological Modelling*, 203(1-2):147–156, 2007.

D.-K. Kim, R.I. McKay, H. Shin, Y.-G. Lee, and N.X. Hoai. Ecological application of evolutionary computation: Improving water quality forecasts for the nakdong river, korea. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2010)*, pages 2005–2012. IEEE, 2010.

E.-S. Kim and Y.-S. Kim. Current status of Korea long-term ecological research (KLTER) network activities compared with the framework activities of the long-term ecological research (LTER) networks of the United States and China. *Journal of Ecology and Field Biology*, 34(1):19–29, 2011.

H.-W. Kim and G.-J. Joo. The longitudinal distribution and community dynamics of zooplankton in a regulated large river: a case study of the Nakdong River (Korea). *Hydrobiologia*, 438(1):171–184, 2000.

H.-W. Kim, K. Ha, and G.-J. Joo. Eutrophication of the lower Nakdong River after the construction of an estuarine dam in 1987. *International Review of Hydrobiology*, 83:65–72, 1998.

M.H. Kim, R.I. McKay, D.-K. Kim, and N.X. Hoai. Evolutionary operator self-adaptation with diverse operators. In *Proceedings of 15th European Conference on Genetic Programming (EuroGP 2012)*, pages 230–241, 2012a.

M.H. Kim, R.I. McKay, Kangil Kim, and N.X. Hoai. Analysing the effects of diverse operators in a genetic programming system. In *Proceedings of the 12th international conference on Parallel Problem Solving from Nature-PPSN XII*, pages 387–396, 2012b.

M.H. Kim, N. Park, and R.I. McKay. Analysis of adaptive pursuit methods on various situations. In *Proceeding of Korean Institute of Information Scientists and Engineers (KIISE) 2012 Fall Conference*, pages 265–267. KIISE, 2012c.

M.H. Kim, N. Park, R.I. McKay, H. Shin, Y.-G. Lee, K.-S. Jeong, and D.-K. Kim. Evolutionary parameter exploration in complex and refractory models: A test case in riverine water quality modelling. *PLOS ONE*, pages 1–21, in revision.

M.H. Kim, R.I. McKay, and N.X. Hoai. Mutation operators in tree-based genetic programming: Importance, tailorability, adaptability. *Computational Intelligence*, pages 1–65, submitted.

Korea Meteorological Administration. Korea Meteorological Administration, KMA. URL `"http://www.kma.go.kr"`.

Korean Ministry of Land, Infrastructure and Transport. Korean Water Management Information System, WAMIS. URL `"http://www.wamis.go.kr"`.

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

J.R. Koza. *Genetic Programming II Automatic Discovery of Reusable Programs*. MIT Press, 1994.

W.B. Langdon. Quadratic bloat in genetic programming. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO 2000)*, pages 451–458. ACM, 2000.

J. Maturana, Á. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 2009)*, pages 365–372. IEEE, 2009.

J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *Journal of Heuristics*, 16(6):881–909, 2010.

N. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. In *Proceedings of 11th European Conference on Genetic Programming (EuroGP 2008)*, pages 134–145. Springer, 2008.

M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1996.

E. Murphy, M. O'Neill, E. Galván-López, and A. Brabazon. Tree-adjunct grammatical evolution. In *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE, 2010.

Q. Nguyen, N.X. Hoai, and M. O'Neill. Semantic aware crossover for genetic programming: the case for real-valued function regression. *Proceedings of 12th European Conference on Genetic Programming (EuroGP 2009)*, pages 292–302, 2009.

J. Niehaus and W. Banzhaf. Adaption of operator probabilities in genetic programming. *Proceedings of 4th European Conference on Genetic Programming (EuroGP 2001)*, pages 325–336, 2001.

U.M. O'Reilly and D.E. Goldberg. How fitness structure affects subsolution acquisition in genetic programming. *Proceedings of 1st European Conference on Genetic Programming (EuroGP 1998)*, pages 269–277, 1998.

A.K. Qin, V.L. Huang, and P.N. Suganthan. Differential evolution algorithm with

strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009.

C.S. Reynolds. *The ecology of phytoplankton*. Cambridge University Press, 2006.

C. Ryan, J.J. Collins, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the First European Conference on Genetic Programming (EuroGP 1998)*, pages 83–95, 1998.

H. P Schwefel. *Numerical optimization of computer models*. John Wiley & Sons, Inc., 1981.

W.M. Spears. Adapting crossover in evolutionary algorithms. In *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 367–384, 1995.

M.D. Terrio and M.I. Heywood. Directing crossover for reduction of bloat in gp. In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2002)*, volume 2, pages 1111–1115. IEEE, 2002.

M.A.L. Thathachar and P.S. Sastry. A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(1):168–175, 1985.

D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO 2005)*, pages 1539—1546. ACM, 2005.

D. Thierens. Adaptive strategies for operator allocation. In *Parameter Setting in Evolutionary Algorithms*, pages 77—90. Springer, 2007.

A. Tuson and P. Ross. Cost based operator rate adaptation: An investigation. In

Proceedings of the 4th international conference on Parallel Problem Solving from Nature-PPSN IV, pages 461–469, 1996.

P.A. Whigham. Genetic programming and spatial information. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence (AI'94)*, pages 124–131, 1994.

P.A. Whigham. Grammatically-based genetic programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, 1995.

M.L. Wong and K.S. Leung. Evolutionary program induction directed by logic grammars. *Evolutionary Computation*, 5(2):143–180, 1997.

H.-S. Yoon and B.-R. Moon. An empirical study on the synergy of multiple crossover operators. *IEEE Transactions on Evolutionary Computation*, 6(2):212–223, 2002.

초록

유전 프로그래밍은 모델 학습에 효과적인 진화 연산 알고리즘이다. 유전 프로그래밍은 다양한 파라미터를 가지고 있는데, 이들 파라미터의 값은 대체로 주어진 문제에 맞춰 사용자가 직접 조정한다. 유전 프로그래밍의 성능은 파라미터의 값에 따라 크게 좌우되기 때문에 파라미터 설정에 대한 연구는 진화 연산에서 많은 주목을 받고 있다. 하지만 아직까지 효과적으로 파라미터를 설정하는 방법에 대한 보편적인 지침이 없으며, 많은 실험을 통한 시행착오를 거치면서 적절한 파라미터 값을 찾는 방법이 일반적으로 쓰이고 있다.

본 논문에서 제시하는 적응 연산자 메커니즘은 여러 파라미터 중 유전 연산자의 적용률을 설정해 주는 방법으로, 학습 중간중간의 상황에 맞춰 연산자 적용률을 자동적으로 조정한다. 본 논문에서는, 기존의 적응 연산자 방법을 다양한 유전 연산자를 가진 문법 기반의 유전 프로그래밍인 TAG3P에 적용하고 새로운 적응 연산자 방법을 개발함으로써, 적응 연산자 메커니즘의 적용 범위를 유전 프로그래밍 영역까지 확장하였다. 기존의 적응 연산자 알고리즘을 TAG3P에 적용시키는 연구는 성공적으로 이루어졌으나 몇 가지 문제점을 드러내었다. 이 문제점은 본문에서 후술한다. 이 문제점을 해결하기 위해 유전자 선택에 대한 새로운 변형 알고리즘을 제시하였고, 이는 기존 알고리즘과 비교하여 더 좋은 성능을 보여주었다.

한편으로 유전 연산자가 해의 향상에 미치는 영향을 측정하는 연산자 영향력 평가에 대한 연구도 진행하였다. 적응 연산자 메커니즘에서는 측정된 영향력을 바탕으로 연산자의 적용률을 변화시키기 때문에 영향력 평가는 적응 연산자 메커니즘에서 매우 중요하다. 이 연구에서는 어떤 정보를 이용하여 영향력을 측정할 것인지, 그리고 어떤 방법을 이용하여 영향력을 측정할 것인지의 두 가지 주요 쟁점을 다룬다. 연산자 영향력 평가에는 학습 과정의 모든 정보가 사용될 수 있으며, 대체로 해의 향상과 직접적인 관련이 있는 적합도를 이용한다. 본 논문에서는 다양한 문제를 이용하여 정확도와 구조에 관련된 두 지표를 영향력 평가에 이용해보았다.

한편으로 같은 정보를 이용하더라도 그것을 활용하는 방법에 따라 측정되는 영향력이 달라지는데, 본 논문에서는 작은 변화를 통해서도 큰 성능 변화를 야기시킬 수 있는 영향력 평가 방법을 몇가지 소개한다.

마지막으로 적응 연산자 메커니즘을 실제 문제에 적용함으로써 유용성을 확인하였다. 이를 위해 사용된 실제 문제는 낙동강의 녹조 현상에 대한 예측으로, 낙동강의 생태 시스템을 묘사하고 예측하는 모델을 개발하는 것을 목적으로 한다. 2가지 연구를 통해 유용성을 확인하였다. 우선 전문가에 의해 만들어진 기본 모델을 바탕으로, 유전 알고리즘을 이용하여 모델의 파라미터를 최적화 하였고, 그리고 TAG3P를 이용하여 기본 모델의 확장하고 이를 통해 새로운 모델을 만들어 보았다.

Keywords: **적응 연산자 메카니즘, 적응 연산자 선택, 유전 프로그래밍, 진화 알고리즘, 파라미터 조절, 파라미터 세팅**

학번: 2005-23499

감사의 글

지난 대학원 생활을 하나의 논문으로 마무리하면서, 지나온 나날에 대한 소회와 앞으로 펼쳐질 연구자로서의 길에 대한 떨림이 섞여 기묘한 느낌을 줍니다. 긴 시간 동안 하나의 연구 주제에 대해 연구하면서 겪은 경험은 앞으로의 삶을 걸어가는데 있어 소중한 바탕이 되었고, 그 과정에서 학문 및 다양한 부분에서 저를 도와주신 많은 분들께 이 자리를 빌어 감사의 말씀을 드립니다.

우선 본 논문을 작성하는데 큰 도움을 주신 심사위원들께 감사드립니다. 제 박사과정 지도교수님이신 Robert Ian McKay 교수님께 크나 큰 감사의 마음을 전합니다. 창의적이고 진취적인 연구를 할 수 있게 끔 항상 독려해주시고 조언을 해주시는 한편, 교수님 스스로 보여주신 연구를 향한 열정적인 모습은 저로 하여금 스스로를 돌아보고 또 채찍질을 할 수 있는 계기를 마련해 주었습니다. 석사과정 지도교수님이신 장병탁 교수님께 또한 크게 감사드립니다. 교수님께서는 인공지능 및 기계학습 전반에 걸친 다양한 분야를 두루 섭렵할 수 있는 기회를 열어주셨으며, 이론적 연구뿐 아니라 응용연구의 중요성을 가르쳐주셨습니다. 심사위원장을 맡으신 문병로 교수님께도 감사드립니다. 심사 과정에서 교수님께서 해주신 논평은 본 논문의 질을 향상시키는데 도움을 주었을 뿐만 아니라, 앞으로의 연구를 진행하는데도 큰 도움이 되었습니다. 심사위원이셨던 성균관대학교 컴퓨터공학과의 안창욱 교수님께도 감사드립니다. 교수님께서는 심사과정 틈틈이 연구 내적인 부분 뿐 아니라 연구 외적인 면에서도 많은 조언을 해주셨습니다. 마지막으로 심사위원이자 선배이신 신수용 교수님께 감사드립니다. 교수님께서는 선배연구자이자 연구실 선배로서 본 논문의 연구에 국한하지 않고 연구활동 전반에 대해 도움을 주셨습니다.

또한 본 논문의 연구주제를 정하는데 도움을 주신 Xuan Hoai Nguyen 교수님, 일본 국립 정보학 연구소에서의 연구 기간 동안 지도해 주신 Atsuhiro Takasu 교수님과 Masada Tomonari 교수님, 그리고 하버드 아동 병원에서의 연구 기간 동안 많은 도움을 주신 공석원 박사님, 황규백 교수님, 박진호 교수님, 이인희 박사님께도

감사드립니다.

8년 간 몸담았던 서울대학교 컴퓨터공학부 구조복잡도 연구실 및 바이오지능 연구실의 동기 및 많은 선후배들께 감사 말씀 드립니다. 오랜 기간 같은 연구실에서 여러 대소사를 함께 겪으며 서로의 연구를 복돋아 준 강일이, 묵묵히 본인의 연구를 진행하는 모습을 보여준 Dharani, 생태 정보학이라는 다소 생소했던 분야를 접하게 해주고 많은 경험을 전해 준 동균이형, 세심하고 꼼꼼한 모습으로 연구실의 많은 일을 도와준 남용이, 새로이 연구실에 들어와 열정적인 모습을 보여준 항준, 준혁, 창완이를 비롯한 은경, 재희, 해수, 윤근, 원욱, 준석, 인수형, 진호형, 중석이형 등의 구조복잡도 식구들에게 감사드립니다. 또한 선배이자 형으로서 대학원 생활에 많은 도움을 준 민오형, 넓고 다양한 식견을 보여준 정우형, 학사졸업논문부터 많은 도움을 주신 병희형을 비롯하여 영균이형, 선이형, 동연이형, 하영이형, 호식이형, 제근이형, 권일이형, 주경이형, 기영이형, 수진이누나 등의 바이오지능 식구들에게도 감사드립니다.

대학원 생활 동안 많은 지지를 보내준 친구 및 지인들에게도 감사인사를 드립니다. 고등학교 동기이자 동네 친구로 오랜 기간 함께 지내온 승훈이, 다양한 연구 주제를 바탕으로 열정적인 모습을 보여주신 광석이형, 학부시절부터 함께 했던 자민, 두회를 비롯한 여러 02학번 동기들과 동문 선후배들, 바쁜 연구활동 중에 틈틈히 작은 여유를 느낄 수 있게 해준 반포지하상가 및 활력소 친구들 및 그 외 여러 지인 분들께 감사드립니다.

기나긴 대학원 생활을 무사히 마칠 수 있었던 것은 저를 믿고 사랑해주는 분들이 제 주위에 계셨기 때문입니다. 긴 세월 동안 묵묵히 저를 믿어주시고 지원해주신 아버지, 어머니, 그리고 형에게 무한한 사랑과 감사를 드립니다.