2009

A Framework for Estimating the Applicability of GAs for Real-World Optimization Problems

Hsin-yi Jiang
Iowa State University, hsinyij@yahoo.com

A Framework for Estimating the Applicability of GAs for Real-World Optimization Problems

Abstract
This paper introduces a methodology for estimating the applicability of a particular Genetic Algorithm (GA) configuration for an arbitrary optimization problem based on run-time data. GAs are increasingly employed to solve complex real-world optimization problems featuring ill-behaved search spaces (e.g., non-continuous, non-convex, non-differentiable) for which traditional algorithms fail. The quality of the optimal solution (i.e., the fitness value of the global optimum) is typically unknown in a real-world problem, making it hard to assess the absolute performance of an algorithm which is being applied to that problem. In other words, with a solution provided by a GA run, there generally lacks a method or a theory to measure how good the solution is. Although many researchers applying GAs have provided experimental results showing their successful applications, those are merely averaged-out, \emph{ad hoc} results. The results cannot represent nor guarantee the usability of the best solutions obtained from a single GA run since the solutions can be very different for each run. Therefore, it is desirable to provide a formalized measurement to estimate the applicability of GAs to real-world problems. This work extends our earlier work on the convergence rate, and proposes an evaluation metric to quantify the applicability of GAs. Through this metric, a degree of convergence can be obtained after each GA run so that researchers and practitioners are able to obtain certain information about the relation between the best solution and all of the feasible solutions. To support the proposed evaluation metric, a series of theorems are formulated from the theory of matrices. Moreover, several experiments are conducted to validate the metric.

Disciplines
Artificial Intelligence and Robotics

A Framework for Estimating the Applicability of GAs for Real-World Optimization Problems

Technical Report #09-19

Hsin-yi Jiang

Department of Computer Science
Iowa State University

CHAPTER 1. INTRODUCTION

1.1 Overview of this Work

This dissertation develops a framework for estimating the applicability of a particular Genetic Algorithm (GA) configuration for an arbitrary optimization problem based on run-time data. GAs are increasingly employed to solve complex real-world optimization problems featuring ill-behaved search spaces (e.g., non-continuous, non-convex, non-differentiable) for which traditional algorithms fail. The quality of the optimal solution (i.e., the fitness value of the global optimum) is typically unknown in a real-world problem, making it hard to assess the absolute performance of an algorithm that is being applied to that problem. In other words, with a solution provided by a GA run, a method or theory to measure the quality of the solution is generally lacking. Although many researchers applying GAs have provided experimental results showing their successful applications, those are merely averaged-out, *ad hoc* results. They cannot represent nor guarantee the usability of the best solutions obtained from a single GA run, since the solutions can be very different for each run. Therefore, it is desirable to provide a formalized measurement to estimate the applicability of GAs to real-world problems.

1.2 Problem Statement and Motivation of this Work

In many real-world optimization problem domains, researchers have gradually found that GAs possess the properties suitable for certain classes of applications. Many researchers have reported success in applying GAs to real-world problems, but have failed to provide a theoretical foundation to explore or explain why GAs were appropriate for the specific problems. As GAs are increasingly utilized, a foundational study on how well GAs can perform for each

of the various problem domains becomes crucial. Note that many theoretical studies have investigated the behaviors of GAs using Markov chains [Eiben, A. E. and Aarts, E. H. L. and Hee, K. M. V. (1991); Nix, A. E. and Vose, M. D. (1992); Ding, L. and Yu, J. (2005); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007); Fogel, D. B. (1995); Rudolph, G. (1994); Suzuki, J. (1995, 1998); Vose, M. D. and Liepins, G. E. (1991); Rudolph, G. (1996); Davis, T. E. and Principe, J. C. (1993)]. Yet, no existing theoretical studies are built upon the linkage between the theory and application of GAs. Through existing analyses, the convergence of Canonical Genetic Algorithms (CGAs) with best solutions maintained has been proven [Rudolph, G. (1994)], the expected value of the first hitting time of GAs has been calculated [Ding, L. and Yu, J. (2005)], and the convergence rate of GAs has been predicted by the second largest eigenvalue of the transition matrix [Suzuki, J. (1995); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. Other methods, such as the analysis on Walsh transformation [Bethke, A. D. (1980); Forrest, S. and Mitchell, M. (1993)] or Fourier transformation of fitness functions of GAs [Kosters, W. A. and Kok, J. N. and Leiden, P. F. (1999)], have also been adopted by some researchers. Although many papers have proposed various approaches to analyzing the behaviors of GAs, most existing GA theories lack concern for practicality [Schoenauer, M. et al. (2007); Jiang, H. and Chang, C. K. (2008); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. For instance, the convergence of GAs studied by Rudolph is only meant as a theoretical study, since time is assumed to go to infinity [Rudolph, G. (1994)]. In reality, it is impossible for any application to wait for an infinite amount of time to obtain the optimal solution. Moreover, the computation time for deriving transition matrices of GAs with respect to real-world problems is much larger than calculating fitness values of the entire search space (i.e., the state space). It is not practical because researchers will not waste more time to obtain weaker solutions. In general, "theoretical studies of GAs are criticized for rarely being applicable to the real-world [Schoenauer, M. et al. (2007)]," and applications of GAs to real-world problems are frequently studied without foundational support [Jiang, H. and Chang, C. K. (2008); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. In application domains of GAs, a more practical and

functionally equivalent approach to evaluate the applicability of GAs is desired in view of the current state of the art.

1.3 The Goal of this Work

This work aims to develop a practical support for researchers and practioners to evaluate the applicability of GAs to their problem domains. We have proven that the convergence rate of Markov transition matrices with respect to encodings of optimization problems is related to the second largest eigenvalue of the transition matrices in absolute value with its physical meaning explained. The second largest eigenvalue can bound the expected value of the first hitting time of the optimal solutions corresponding to those optimization problems [Ding, L. and Yu, J. (2005); Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]; however, the computation time of constructing the transition matrix with respect to an optimization problem takes more than the computation time of calculating all the feasible solutions [Jiang, H. and Chang, C. K. (2008)]. Relying on the second largest eigenvalue of the transition matrix with respect to a problem to acquire convergence degrees of GAs is impractical due to the long computation time. With a concern for practicality, we propose a novel method, an important approach for real-world applications, to derive the applicability of a GA to a problem based on the approximate sum of eigenvalues. Mathematically, the sum of eigenvalues of a matrix is equal to the sum of the diagonal elements (which is called "trace") of the matrix. According to that property, our methodology is developed to estimate the trace of the corresponding transition matrix. Through this method, a degree of convergence can be determined for each GA run. Being aware of the degree of convergence, researchers and practioners will be able to obtain certain information about the applicability of GAs and know how good the solutions generated by GAs are, so that correct decisions can be made. Moreover, a possible approach for estimating the number of generations needed for global convergence is also proposed. The general methodology is illustrated in Figure 1.1. In summary, this dissertation makes the following contributions to theory and application of GAs:

- a problem statement for the current state of the art;

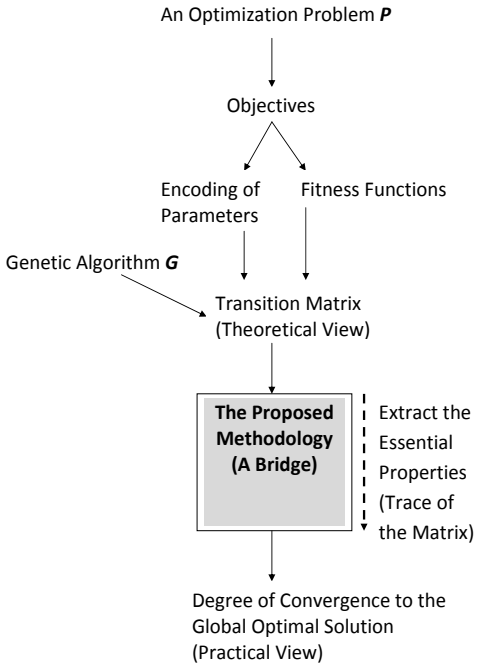


Figure 1.1 General Methodology for Estimating the Applicability of GA **G** for Optimization Problem **P**

- a novel and possible approach to build a linkage between theory and application of GAs;
- the verification of the proposed approach.

The rest of this dissertation is organized as follows. Chapter 2 provides the literature review of this work. Chapter 3 reviews the basic operators of GAs, Markov chains, and how to model the operators of GAs using Markov chains. Research assumptions and definitions are also presented in this chapter. Chapter 4 proves that the convergence rate of a CGA with best solution maintained over time depends on the second largest eigenvalue of the corresponding transition matrix, and explains the relationship between the second largest eigenvalue of the transition matrix, and the first hitting time (i.e., expected waiting time) of the optimal solution of a GA. Chapter 5 explores an evaluation matrix to evaluate the applicability of GAs to real world optimization problems. In Chapter 6, a case study about an evolutionary algorithm (EA) used in software testing (also called evolutionary testing) is provided. Chapter 7 proposes a possible approach for estimating the number of generations needed for the global convergence.

Chapter 8 proves that the proposed metric can be generalized to support certain classes of EAs. Chapter 9 concludes the dissertation and outlines future research work.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Real-World Optimization Problems

Generally, if a problem has more than one feasible solution, the problem can be considered as an optimization problem. Theoretically, the optimization problem is defined as the problem which can be solved by more than one feasible solution and has at least one criterion to evaluate solutions, with the goal being the search for the best solution within the domain of all the feasible solutions.

In the real-world, there are many optimization problems in which GAs are applicable. Among many fields of study, such as combinatorial optimization problem domains, machine learning, information retrieval, and data mining, project management and software engineering are two heavily studied fields that provide ample opportunities to apply GAs for performance improvement.

I have research interests in both Project Management (PM) and Software Engineering (SE). Some problems in those two domains are listed below.

2.1.1 Problem Domain 1 - Project Management Problems

In PM, optimal scheduling is one of the typical optimization problems [Chang, C. K. and Christensen, M. J. (1999); Chang, C. K. and Christensen, M. J. and Zhang, T. (2001); Chao, C. (1995); Ge, Y. (2004)]. One branch in PM research explores ways to intelligently match employees to tasks with respect to the factors gathered during early project development. Typical factors include salaries of employees (costs), capabilities of employees, learning curves of employees, and potential hazard levels of assignments [Xu, R. and Qian, L. and Jing, X. (2003)]. Researchers may expand or narrow this list of factors depending upon their research

goals.

2.1.2 Problem Domain 2 - Software Engineering Problems

Software Testing Problems - In SE, the traditional waterfall model regards testing as a key component of verification and validation (V&V) activities [Pressman, R. S. (2005)]. Verification is to inspect whether specific functions are built correctly in the software, and validation is to examine whether the software meets customer requirements. To conduct testing on a piece of software, various strategies have been proposed. Some strategies are formulated as optimization problems [Berndt, D. and Fisher, J. and Johnson, L. and Pinglikar, J. and Watkins, A. (2003); Berndt, D. J. and Watkins, A. (2004); Briand, L. C. and Labiche, Y. and Shousha, M. (2004, 2005); Clark, J. and Dolado, J. J. and Harman, M. and Hierons, R. M. and Jones, B. and Lumkin, M. and Mitchell, B. and Mancoridis, S. and Rees, K. and Roper, M. and Shepperd, M. (2003); Vieira, F. E. and Menezes, R. and Braga, M. (2006)]. For instance, with regard to branch testing in a structural testing problem [Clark, J. and Dolado, J. J. and Harman, M. and Hierons, R. M. and Jones, B. and Lumkin, M. and Mitchell, B. and Mancoridis, S. and Rees, K. and Roper, M. and Shepperd, M. (2003)], the condition statements of the test-aim branch are formulated into a fitness function, which guides the search of input data to satisfy all the conditions of the test-aim branch. That is, researchers measure the "distance" between the test-aim branch and the branch caused by a set of input data. GAs minimize the distance so that the input data (called test cases) of a test aim can be generated.

Software Module Clustering Problems - Owing to the rapid development of computer technologies, industrial applications are increasingly equipped with highly complex software systems, which often consist of a large number of components. To streamline the design phase, a large-scale system can be designed hierarchically. Instead of directly integrating all of the components into a system, a set of congruent components are first grouped into a subsystem. How to cluster the original set of components into several subsystems has attracted the attention of researchers for many years [Chang, C. K. and Cleland-Haung, J. and Hua, S.

and Kuntzmann-Combelles, A. (2001)]. Oftentimes, they try to maximize cohesion inside a component and minimize coupling among components. This kind of problem also represents a class of optimization problems in SE [Mitchell, B. S. and Mancoridis, S. (2002)].

2.2 Analyzing the Behaviors of GAs

2.2.1 Markov Chain Based Approaches

A variety of methods have been employed to analyze the behaviors of GAs [Rudolph, G. (1994); DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995); Suzuki, J. (1995); He, J. and Kang, L. (1999); He, J. and Yao, X. (2001); Ding, L. and Yu, J. (2005); Coley, D. A. (1999); Mitchell, M. (1996); Goldberg, D. E. (1989); Grefenstette, J. J. (1992); Bethke, A. D. (1980); Forrest, S. and Mitchell, M. (1993); Bridges, C. L. and Goldberg, D. E. (1991); Naudts, B. and Kallel, L. (2000)]. Among them, different approaches are derived and asserted. While each method has its own merits, Markov chain analysis can be most successfully applied to capture the essential spirit of GAs due to the following reasons:

- The initial population of a GA run is based on a probability distribution. Usually, it is a uniform distribution.
- From one generation to another, the GA selects the individuals based on the proportions of the fitness values of the individuals in the current population. In other words, the selection operator selects the individuals from (and only from) the current state where each individual has some probability of being preserved in the next generation. Note that the fitness function of the GA run is considered in the selection operation.
- Regardless of the type of recombination employed to solve the optimization problem, the recombination and mutation operators are both related to probability issues.

My research adopts Markov chain analysis to investigate the behaviors of GAs. Previous work by Rudolph [Rudolph, G. (1994)], De Jong [DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995)], Suzuki [Suzuki, J. (1995)], He [He, J. and Kang, L. (1999); He, J.

and Yao, X. (2001)], and Ding [Ding, L. and Yu, J. (2005)], including their methodologies and results, are particularly relevant to my approach.

Rudolph's main contribution on the behavior of GAs is to prove that the CGA with best solution maintained converges to its global optimal solution [Rudolph, G. (1994)]. Additionally, he mentions that in an ergodic Markov chain, the expected value of the transient time in which an arbitrary state i goes to any other state j is finite. This claim is a well established result in Markov chain theory. The detailed proof can be seen in [Iosifescu, M. (1980) (p. 133)]. This implies that the expected value of the transient time in which the initial state moves to the optimal state is finite, since both the initial state and the optimal state are the states in Markov chain.

Later, De Jong et al. proposed a method to capture the "hardness" of a GA (i.e., the level of difficulty to apply it) by computing the expected waiting time (i.e., the first hitting time) through the use of transition matrices [DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995)]. To my knowledge, this was the first attempt to use the first hitting time to predict the applicability of GAs. In 2005, the expected first hitting time of the optimal state was investigated and calculated again by Ding et al. [Ding, L. and Yu, J. (2005)]. They proposed an approach to reformulate the transition matrice so that the formula to derive the expected value of the first hitting time became simpler than before. Besides the computation of the expected waiting time, De Jong et al. introduced the concept of predicting the behaviors of GAs within a fixed number of generations based on the derived transition matrices [DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995)]. This concept provides an insight for the waiting time in practical use, which is similar to Assumption 3 in Chapter 3 of this dissertation.

In Suzuki's work, the convergence rate was first linked to the second largest eigenvalue of the transition matrix [Suzuki, J. (1995)]. This assertion matches my earlier result, which is derived from a different approach in Markov chain theory [Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)]. In addition to the eigenvalue-based approach, He and Kang proposed another perspective to bound the convergence rate through the "minorization condition" in Markov chain theory [He, J. and Kang, L. (1999)].

He and Yao devoted their work to calculating the computational time complexity of evolutionary algorithms (EAs) [He, J. and Yao, X. (2001)]. Their work generalized Droste's work, which provided a rigorous complexity analysis of the $(1 + 1)$ EA (i.e., EA with a population size of 1 and only with mutations), for a class of fitness functions [Droste, S. and Jansen, T. and Wegener, I. (1998)]. With He and Yao's approach, Droste's work was extended to general EAs through drift analysis. Several drift conditions were studied for deriving the computational time. Their work began with modeling the evolution of an EA population as a random sequence, e.g., a Markov chain. The general case was considered with three operators (selection, crossover, mutation) and a population with multiple individuals. Then, they analyzed the relationship between the drift of the sequence and the optimal solution of the problem. Various bounds on the first hitting time were derived under different drift conditions. They also asserted that some drift conditions caused the random sequence to drift away from the optimal solution, while others enabled the sequence to drift towards the optimal solution. The conditions used to determine the time complexity of an EA to solve a problem were also investigated and proposed [He, J. and Yao, X. (2001)].

2.2.2 Other Approaches

Starting from the early 1970s, a series of approaches different from Markov chain analysis were proposed to analyze the behaviors of GAs. In 1975, Holland introduced the notion of schemas to formalize the informal notion of "building blocks" [Coley, D. A. (1999) Mitchell, M. (1996)]. His building block hypothesis stated that GAs attempt to find highly fit solutions to a problem through short, low-order, and above-average schemata. However, the schema theory merely demonstrated a rough idea in high level that better performing schemata will receive an increasing number of trials in the next generation. It does not give us much information about the detailed analysis of the behaviors of GAs. In 1987, the term "deception problem" was coined by Goldberg, who said that a problem is deceptive if certain hyperplanes guide the search toward some solutions or genetic building blocks that are not globally competitive [Goldberg, D. E. (1989)]. Comparing Goldberg's statement with Holland's building block hypothesis,

the deception problem appears to suggest a contradiction. Hence, deception problems are considered "hard" problems for GAs. Nevertheless, the deception is neither a sufficient nor necessary condition to characterize problems that are hard for GAs [Grefenstette, J. J. (1992)].

Goldberg isolated the deception problems from Bethke's work [Bethke, A. D. (1980); Forrest, S. and Mitchell, M. (1993)]. Bethke used discrete Walsh functions to analyze the fitness functions of GAs. He developed the Walsh-Schema transform to calculate the average fitness of schema efficiently and used it to characterize functions as easy or hard for GAs to optimize. This method helps to produce ideas for solving a problem; however, it can be difficult to convert functions to Walsh polynomials. This method, a static analysis, examines only a flat population, where every possible string is assumed to be represented in equal proportion [Bridges, C. L. and Goldberg, D. E. (1991)]. Bethke's method fails to capture the more dynamic aspects found in GAs. To address this issue, Bridges and Goldberg proposed another approach called the Nonuniform Walsh Transform [Bridges, C. L. and Goldberg, D. E. (1991)]. Unfortunately, as in the case of Walsh transformations, it is difficult to convert fitness functions into such forms.

In 2000, Naudts and Kallel studied two widely-known predictive measures of problem difficulty in GAs (with both the GA-easy and GA-hard functions): epistasis variance and fitness distance correlation [Naudts, B. and Kallel, L. (2000)]. They found that the values of the measures can be completely unreliable and entirely uncorrelated to the convergence quality and speed of GAs.

In general, the GA theory developed thus far shows that it is difficult to fully capture the behaviors of GAs, especially in finite time with different types of fitness functions. The missing link is the bridge between the theory and its applicability to practical problems [Jiang, H. and Chang, C. K. (2008)]. This observation coincides well with the recent claims stated in the front pages of the journal [Schoenauer, M. et al. (2007)] concerning the difficulty in directly linking GA theory to real-world applications. In this study, the objective is to develop the necessary support theory to effectively bridge the gap between evolutionary computation and real-world applications.

CHAPTER 3. BACKGROUND

This chapter reviews and presents the background information for GAs, Markov chains, how to model GAs using Markov chains, existing theorems, and research assumptions of my work.

3.1 Optimization Problems and Fitness Functions of GAs

As mentioned before, an optimization problem is defined as the problem which can be solved by more than one feasible solution, has at least one criterion to evaluate solutions, with the goal of the problem being the search for the best solution within the domain of all the feasible solutions. The (feasible) solutions of GAs are encoded into strings, usually called chromosomes or individuals. With a GA being chosen as the optimization method, the criteria to evaluate solutions are formulated as a fitness function for the GA.

3.2 The Canonical Genetic Algorithm and Its Operators

The CGA (also called a simple GA) can be sketched as follows [Rudolph, G. (1994)].

Choose an initial population (i.e., a list of a fixed number of individuals)

Repeat

Compute the fitness of each individual

Perform selection

Perform crossover

Perform mutation

Until stopping criterion is satisfied

In other words, it is composed of three operators:

- Selection (also called Reproduction)
- Crossover
- Mutation

Selection is a process in which individuals are copied according to their fitness values. Usually, the individuals with higher fitness values have higher probabilities to be selected into the next generation. Therefore, a typical fitness function for a selection operator should be the function to be maximized. This operator is actually an artificial version of natural selection, the Darwinian theory of "survival of the fittest" [Goldberg, D. E. (1989)]. Various selection methods, such as roulette wheel selection (proportional selection), tournament selection, and (μ, λ) selection, etc., are proposed. Among them, roulette wheel selection is commonly adopted in the literature. It selects individuals based on their proportions of the fitness values among the individuals in the current population (generation). For demonstration purposes, this dissertation mainly discusses roulette wheel selection. In Chapter 8, the selection operator is generalized to any type of selection methods.

Crossover, including one point crossover, two point crossover, and uniform crossover, etc., mimics the mating of creatures. It swaps some bits of two chosen individuals. The resulting individuals are passed into the next generation.

The mutation operator simulates biological mutation, maintaining genetic diversity from one generation to another. A simple and common way to implement it is to sweep each individual bit of strings within a population once, and each bit has a fixed probability to be flipped to another number.

3.3 An Overview of Markov Chain

Markov chains are named after Prof. Andrei A. Markov (1856-1922) who first published his result in 1906 [Ching, W. K. and Ng, M. K. (2006)]. His research work on Markov chains launched the study of stochastic processes, which was followed by a large variety of applications. In this work, the discrete homogeneous finite state Markov chain is applied.

Specifically, a Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property [Iosifescu, M. (1980)]. The Markov property is the property that:

$$Pr(X_{n+1} = x|X_n = x_n, \ldots, X_1 = x_1) = Pr(X_{n+1} = x|X_n = x_n), \forall n \in N. \quad (3.1)$$

The future state is only dependent on the current state and independent of the past states.

A Markov chain is called *time-homogeneous* if:

$$Pr(X_{n+1} = x|X_n = y) = Pr(X_n = x|X_{n-1} = y), \forall n \in N. \quad (3.2)$$

To form a Markov chain, three basic components should be considered: a state space, an initial distribution, and a transition matrix.

Consider a *random walker* in a small town. Within the town, there are a finite number of places to go. Suppose that at time t, $t \in N$, the random walker stands in a place in the town. At time $t+1$, he walks to any place in the town with a certain probability, dependent only upon the place he was in at time t (i.e., time is a non-factor). If each place in the town is assigned a distinct number (as an index), say, $1, 2, \ldots, k$ ($k \in N$), and at each time t, X_t is a random variable denoting the index of the random walker's location, $(X_0, X_1, \ldots)$ is a random process taking values in $\{1, 2, \ldots, k\}$. Such a random process can be treated as a discrete homogeneous finite state Markov chain. Figure 3.1 illustrates the transition matrix of a Markov chain for the random walker in the small town. In Figure 3.1, places are considered as states of the Markov chain. If the current time is t, '$*$' represents the probability of transitioning from state i to state j ($i, j \in \{1, \ldots, k\}$) at time $t + 1$.

3.3.1 Definitions and Theorems in GA, Markov Chain Theory, and Linear Algebra

Definition 1. A nonnegative square matrix A is said to be stochastic if and only if the sum of the entries in any row of A is 1.

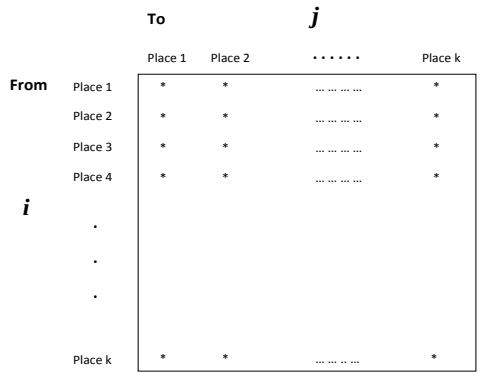


Figure 3.1 The Transition Matrix of the Random Walker in the Small Town

Definition 2. A square matrix $A_{r\times r}$ is called positive (i.e., $A > 0$) if and only if $a_{ij} > 0$ $\forall i, j \in \{1, 2, \ldots, r\}$.

Definition 3. A stochastic matrix A is said to be regular if and only if there exists a natural number r such that A^r is positive (i.e., $A^r > 0$).

Note that the product of stochastic matrices is a stochastic matrix.

Definition 4. A state in a Markov transition matrix is called transient if there is a non-zero probability that once the chain leaves that state, it will never return.

Definition 5. A state in a Markov transition matrix is called absorbing if once the chain enters that state, it never leaves.

Definition 6. The trace of a square matrix $A_{r\times r}$ is defined to be the sum of the elements on the main diagonal of A, i.e.,

$$Trace(A) = a_{11} + a_{22} + \cdots + a_{rr}.$$

Definition 7. *[Burden, R. L. and Faires, J. D. (2005)] [Convergence Rate]* Suppose $\{\beta_n\}_{n=1}^{\infty}$

is a sequence known to converge to zero, and $\{\alpha_n\}_{n=1}^{\infty}$ converges to a number α. Then it is called that $\{\alpha_n\}_{n=1}^{\infty}$ converges to α with convergence rate (or rate of convergence) $O(\beta_n)$ if there exist a constant K and a number $M' > 0$ such that

$$|\alpha_n - \alpha| \leq K|\beta_n|, \text{ for all } n \geq M'. \tag{3.3}$$

In addition, Inequality (3.3) indicates that $\alpha_n = \alpha + O(\beta_n)$ for large n.

Definition 8. *[Empirical Convergence]* Let P be a matrix. P^n converges to P^* as $n \to \infty$. Then P^n is called to empirically converge to P^* if for an arbitrarily small $\epsilon > 0$, $\exists$ a constant $n_1(\epsilon) > 0$ such that

$$|p_{ij}^{(n)} - p_{ij}^*| < \epsilon, \forall n \geq n_1(\epsilon),$$

where $p_{ij}^{(n)}$ is the element in the i^{th} row and j^{th} column of P^n, and p_{ij}^* is the element in the i^{th} row and j^{th} column of P^*.

Theorem 1 *[Rudolph, G. (1994)] The transition matrix of the CGA with mutation probability $p_m \in (0, 1)$, crossover probability $p_c \in [0, 1]$ and fitness proportional survivor selection is regular.*

Theorem 1, formulated by Rudolph, is used to prove the convergence of CGA, with best solution maintained, to its global optimal solution [Rudolph, G. (1994)].

Theorem 2 *[Iosifescu, M. (1980)] If P is a $r \times r$ regular stochastic matrix, then P^n converges as $n \to \infty$ to a positive stable stochastic matrix $\prod = ev^T$, where $e = (1, 1, \cdots, 1)^T$ is a $r \times 1$ column vector in which all elements are of the value 1, and $v^T = (v_1, v_2, \cdots, v_r)$ is a $1 \times r$ probability row vector with non-null entries. Moreover, there exists a constant $a > 0$ such that*

$$|p_{ij}^{(n)} - v_j| \leq an^{m_2 - 1}|\lambda_{2\_P}|^n, \tag{3.4}$$

where $p_{ij}^{(n)}$ is the i^{th} row and j^{th} column of P^n, λ_{2_P} is the second largest eigenvalue of P in absolute value, and m_2 is the (algebraic) multiplicity of λ_{2_P}.

Theorem 3 *[Iosifescu, M. (1980)] Let the transition matrix P be*

$$P = \begin{pmatrix} P_1 & 0 \\ R & A \end{pmatrix},$$

where P_1 is regular, $R \neq 0$. Then

$$P^n = \begin{pmatrix} P_1^n & 0 \\ \sum_{i=0}^{n-1} A^{n-1-i} R P_1^i & A^n \end{pmatrix}.$$

As $n \to \infty$, P^n converges to

$$\lim_{n\to\infty} P^n = \begin{pmatrix} e_1 v^T & 0 \\ e_2 v^T & 0 \end{pmatrix}, \tag{3.5}$$

where $v^T = (v_1, v_2, \cdots, v_{r_1})$ is a $1 \times r_1$ probability row vector with non-null entries, $e_1 = (1, 1, \cdots, 1)^T$ is a $r_1 \times 1$ column vector if r_1 is the number of rows in P_1, and $e_2 = (1, 1, \cdots, 1)^T$ is a $r_2 \times 1$ column vector if r_2 is the number of rows in R.

Note that A^n converges to 0 as $n \to \infty$. Moreover, $\lim_{n\to\infty} P^n$ has the property that each of the columns has the same entry value. This is insightful for the proof that the initial state will not impact the final state if time goes to infinity.

Theorem 4 *[Iosifescu, M. (1980)][Perron's Formula] If A is a square matrix of order r, $\lambda_1, \lambda_2, \ldots, \lambda_q$, $q \leq r$, are the eigenvalues of A, and $m_1, m_2, \ldots, m_q$ are the (algebraic) multiplicities of the eigenvalues, respectively, $m_1 + m_2 + \cdots + m_q = r$, then*

$$a_{ij}^{(n)} = \sum_{k=1}^{q} \frac{1}{(m_k - 1)!} \left(\frac{d^{m_k - 1}}{d\lambda^{m_k - 1}} \left(\frac{\lambda^n A_{ij}(\lambda)}{\prod_{i \neq k} (\lambda - \lambda_i)^{m_i}} \right) \right)_{\lambda = \lambda_k},$$

where $a_{ij}^{(n)}$ is the element in the i^{th} row and j^{th} column of A^n, and $A_{ij}(\lambda)$ is the element in the i^{th} row and j^{th} column of the adjoint of the matrix $(\lambda I_r - A)$; that is, $A_{ij}(\lambda)$ is equal to the product of $(-1)^{i+j}$ and determinant of the minor of $(\lambda I_r - A)_{ji}$. Note that the minor of $(\lambda I_r - A)_{ji}$ is derived by deleting the j^{th} row and i^{th} column from the matrix $(\lambda I_r - A)$, and I_r is the identity matrix of order r.

Theorem 5 *[Iosifescu, M. (1980)] If A is a regular matrix, then there exists a real eigenvalue $\lambda_1 > 0$ which is simple (i.e., of algebraic multiplicity 1) and which exceeds the absolute values of all other eigenvalues of A.*

Lemma 1 *[Iosifescu, M. (1980)] If A is a stochastic matrix (i.e., A is nonnegative and the sum of the elements in any row of A is 1), then the eigenvalues of A are in absolute value at most equal to 1. Moreover, 1 is an eigenvalue of A.*

Theorem 6 *[Horn, R. A. and Johnson, C. R. (1985)] [Schur] If A is a square matrix of order r with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$ in any prescribed order, there exists a unitary matrix S (square matrix) of order r such that*

$$S^* A S = T = [t_{ij}]$$

is an upper triangular, with diagonal entries $t_{ii} = \lambda_i$, $i = 1, 2, \ldots, r$. That is, every square matrix A is unitarily equivalent to a triangular matrix whose diagonal entries are the eigenvalues of A in a prescribed order.

Theorem 2, 3, 4, 5, 6, and Lemma 1 are directly from Markov chain theory and linear algebra. Theorem 4, 5, 6, and Lemma 1 are utilized to deduce my preliminary result [Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007)].

3.3.2 Research Assumptions

The first two assumptions adopted in this research are based on Rudolph's work [Rudolph, G. (1994)].

Assumption 1 *[Problem Definition] The problems $max\{f(b)|b \in IB^l\}$, where $0 < f(b) < \infty$ for all $b \in IB^l = \{0,1\}^l$, and l is the length of the binary strings which represent feasible solutions, are the subjects for discussion.*

Assumption 2 *[Choice of Solution Method] The CGA, which only has selection, crossover, and mutation operators, with the best solution maintained, is the algorithm to be analyzed. (More specifically, roulette wheel selection, any crossover operation, and bit mutation are considered before Chapter 8 in this dissertation.)*

The last assumption pertains to the concerns for practicality and applicability.

Assumption 3 *The number of generations of GAs is a reasonably large number, and it is fixed.*

3.3.3 Markov Chain Analysis for GAs

3.3.3.1 State Representations

Two types of state representations for the finite state homogeneous Markov transition matrices are commonly adopted in Markov chain analysis. Michael D. Vose, Joe Suzuki, et al. applied the transition matrix with the states representing the occurrences of the individuals [Suzuki, J. (1995, 1998); Vose, M. D. and Liepins, G. E. (1991)]. The cardinality of different populations (i.e., the dimension of the state space), becomes [Suzuki, J. (1995)]

$$|S| = \left(\begin{array}{c} m + 2^l - 1 \\ m \end{array} \right),$$

where S is the state set, m is the size of a population, and l is the length of the binary strings.

Günter Rudolph and David B. Fogel [Fogel, D. B. (1995); Rudolph, G. (1994)] proposed the transition matrix in which the states are defined by every possible configuration of an entire population of bit strings. Therefore, there are 2^{ml} states, where m is the size of a population, and l is the length of the binary strings.

Although the approaches of both Vose and Rudolph possess similar concepts and functionalities, and they can be converted to each other (see Theorem 10), the representations are different. Each approach has its advantages. While fewer states and the distributions of the individuals can be obtained in Vose's approach, the analysis of this approach is not as intuitive as Rudolph's approach. For instance, if there are four individuals (3-digit binary strings) within a generation, say "101, 111, 011, 111", this generation is represented by the state "101111011111" in Rudolph's approach. For Vose's approach, one has to list the sorted state space "000, 001, 010, 011, 100, 101, 110, 111" first, and obtain the state "11101101100". Since my research goal is to seek an evaluation metric to estimate the applicability of GAs for real-world applications and Markov chain analysis is not so practical (discussed in Section VI), either approach can be chosen as the transient analyzing method. Due to the fact

that Rudolph's approach is more intuitive than Vose's, my work uses Rudolph's approach to transform the encodings of optimization problems to Markov chains [Rudolph, G. (1994)]. Rudolph's work studies the convergence of CGAs with the best solution maintained. The transition matrices of his Markov chains are described below.

3.3.3.2 Transition Matrices

A CGA is a *canonical* GA which consists of an m-tuple of binary strings $b_i, i \in \{1, 2, \ldots, m\}$, of length l. The bits of each string are considered to be the genes of an individual chromosome. The m-tuple of individual chromosomes is said to be a population of a generation. From one generation to another, CGA applies three operators on the population. The operators include selection, crossover, and mutation operators.

The selection operator, in which the roulette wheel selection (proportional selection) is assumed, forms a transition matrix S. As mentioned before, there are m (an even number) individuals for each generation. By Assumption 1, S is a $2^{ml} \times 2^{ml}$ matrix with the element

$$s_{ij} = \frac{\prod_{k=1}^{m} O_k \cdot f(\pi_k(j))}{(\Sigma_{k=1}^{m} f(\pi_k(i)))^m}$$

if $\{\pi_1(j), \pi_2(j), \ldots, \pi_m(j)\} \subseteq \{\pi_1(i), \pi_2(i), \ldots, \pi_m(i)\}$, where $\pi_k(i), k \in \{1, \ldots, m\}$ are the k^{th} segment of length l from the state i, O_k is the number of occurrences of $\pi_k(j)$ in state i, and $f(\cdot)$ is the fitness function. Otherwise, $s_{ij} = 0$.

By the same token, the crossover transition matrix C is also a $2^{ml} \times 2^{ml}$ matrix. After the crossover method is determined, I_i is defined to be the index set in which each element is a binary string with length ml representing a possible mating method for the individuals of state i. For each $r \in I_i$, let p_r be the probability of r being selected as a mating method. If p_c is the crossover probability, then

$$c_{ij} \;=\; (1 - p_c)\delta_{ij} + p_c \cdot (\sum_{r \in I_i} p_r \cdot \prod_{k=1}^{\frac{m}{2}} P\{Cr(\pi_{2k-1}(r), \pi_{2k}(r)) \;=\; (\pi_{2k-1}(j), \pi_{2k}(j))\}), \quad (3.6)$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases},$$

$Cr(\cdot,\cdot)$ is the result of the crossover operation, and $P\{\cdot\}$ is the probability of the event. In (3.6),

$$P\{Cr(\pi_{2k-1}(r),\pi_{2k}(r)) = (\pi_{2k-1}(j),\pi_{2k}(j))\}$$

depends on the crossover method applied.

Bit mutation is designed to serve as a background operator to ensure that all possible alleles can occur in the population [Fogel, D. B. (1995)]. Once the population of chromosomes reaches a configuration such that crossover no longer produces offspring outperforming their parents, it is the only operator which leads the population to leap out of the homogeneous populations. The mutation transition matrix is denoted as M, which is a $2^{ml} \times 2^{ml}$ matrix. Let $p_m \in (0,1)$ be the probability of flipping each individual bit, then

$$m_{ij} = \prod_{k=1}^{m} p_m^{H(\pi_k(i),\pi_k(j))}(1-p_m)^{l-H(\pi_k(i),\pi_k(j))},$$

where $H(\cdot,\cdot)$ is the Hamming distance of the strings (chromosomes). The matrix CMS, the product of C, M, and S, forms a transition matrix for CGA.

In order to show that the CGA with the best solution maintained converges to its global optimum, the state space is extended from 2^{ml} to $2^{(m+1)l}$. That is, for each state (m individuals), there is a referenced individual (assumed to be the leftmost individual) with it. The referenced individual is for the calculation with respect to the presence of the best solution. Since there are 2^l distinct referenced individuals, the new state space is $2^{ml} \cdot 2^l = 2^{(m+1)l}$. The new transition matrix P is

$$P = \begin{pmatrix} CMS & & & \\ & CMS & & \\ & & \ddots & \\ & & & CMS \end{pmatrix}, \tag{3.7}$$

where each of the diagonal squares $CMS's$ is corresponding to a referenced individual. The referenced individuals are sorted by fitness values in descending order. That is, the first diagonal square represents the highest fitness value, and the second square represents the second highest fitness value, and so on.

The upgrade matrix U is also for maintaining the best solution. If a state in the i^{th} diagonal square has a best fitness value higher than the fitness value of the referenced individual corresponding to the i^{th} diagonal square, this state is upgraded to the j^{th}, $j < i$, diagonal square in which the fitness value is equal to the best fitness value of the state. The structure of U is

$$U = \begin{pmatrix} U_{11} & & & \\ U_{21} & U_{22} & & \\ \vdots & \vdots & \ddots & \\ U_{2^l,1} & U_{2^l,2} & \cdots & U_{2^l,2^l} \end{pmatrix}, \tag{3.8}$$

which is a lower triangular matrix. Moreover, since it is assumed that there is a unique optimal solution for the optimization problem in Rudolph's work [Rudolph, G. (1994)], U_{11} is the only $2^{ml} \times 2^{ml}$ identity matrix. That is, none of the states in the first diagonal square need to be upgraded.

Figure 3.2 shows an example of the structure of PU with the dashes representing nonzero entries of PU, where $m = 1$ and $l = 3$. In fact, the figure can be generalized to any value of m and l.

3.3.3.3 The Proof of Convergence of CGA

Rudolph's proof on the convergence of CGA, with best solution maintained to its global optimal solution, is quite unique [Rudolph, G. (1994)]. The concept and method will be briefly reviewed in this subsection.

From Equation (3.7) and (3.8) in the previous subsection, we have

$$PU = \begin{pmatrix} CMSU_{11} & 0 \\ R & A \end{pmatrix}, \tag{3.9}$$

where R and A are corresponding sub-matrices. Since U_{11} is a $2^{ml} \times 2^{ml}$ identity matrix, the block matrix $CMSU_{11}$ in Equation (3.9) can be simplified as CMS.

According to Theorem 1, CMS is regular. Theorem 3 shows that the matrix PU converges (i.e., CGA with best solution maintained converges). In addition, based on Equation (3.5), the first 2^{ml} states are absorbing states. That is, regardless of the initial population (state),

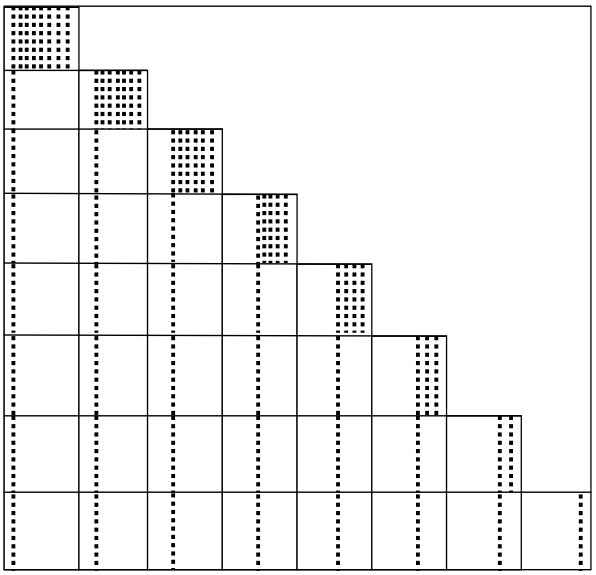


Figure 3.2 Structure of the Extended Transition Matrix PU

the populations (states) of CGA with best solution maintained will eventually be restricted within a subset of all the populations (states). According to the design of upgrade matrix U in the previous subsection, all the populations in the subset are under the condition that the global optimal solution is found. Their referenced individuals have the highest fitness value. Hence, we can conclude that CGA with best solution maintained converges to its global optimal solution.

CHAPTER 4. CONVERGENCE RATE AND THE FIRST HITTING TIME

Rudolph's proof shows that a CGA with the best solution maintained over time will eventually converge to its global optimal solution; however, how long it takes to converge is not specified. To address this problem, a study was conducted to investigate and analyze the convergence rate of a CGA with the best solution maintained over time.

4.1 Convergence Rate

From Theorem 4 [Perron's Formula], without loss of generality, we can bound $a_{ij}^{(n)}$ by estimating the term,

$$\left(\frac{d^{m_k-1}}{d\lambda^{m_k-1}}\left(\frac{\lambda^n A_{ij}(\lambda)}{\prod_{i\neq k}(\lambda-\lambda_i)^{m_i}}\right)\right)_{\lambda=\lambda_k}, \tag{4.1}$$

where $k \in \{1, 2, \ldots, q\}$, as follows. Since $A_{ij}(\lambda)$ is equal to the product of $(-1)^{i+j}$ and determinant of the minor of $(\lambda I_r - A)_{ji}$, it is a polynomial of order $(r-1)$. Let

$$f_{ij}(\lambda) = \frac{\lambda^n A_{ij}(\lambda)}{\prod_{i\neq k}(\lambda-\lambda_i)^{m_i}}, \tag{4.2}$$

then we get the order of the function $f_{ij}(\lambda)$ is $(n+(r-1))-(r-m_k) = n+(m_k-1)$. Therefore, the order of $(m_k-1)^{th}$ derivative of $f_{ij}(\lambda)$ in equation (4.1) is n. Moreover, a multiplier of $O(n)$ is applied everytime during the process in which the derivative is obtained. Hence, it is concluded that if A is a square matrix of order r, $\lambda_1, \lambda_2, \ldots, \lambda_q$, $q \leq r$, are the eigenvalues of A, and $m_1, m_2, \ldots, m_q$ are the (algebraic) multiplicities of the eigenvalues, respectively, $m_1 + m_2 + \cdots + m_q = r$, then there exists a positive number K such that

$$|a_{ij}^{(n)}| \leq K n^{m_{max}-1} |\lambda_{max}|^n, \tag{4.3}$$

where $a_{ij}^{(n)}$ is the element of A^n in the i^{th} row and j^{th} column, $m_{max} = max\{m_1, m_2, \ldots, m_q\}$, and $|\lambda_{max}| = max\{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_q|\}$. This inequality will be exploited to show that the convergence rate is related to one of the eigenvalues of the transition matrix later.

From section 2, the transition matrix of the CGA with the best solution maintained is

$$PU = \begin{pmatrix} CMSU_{11} & & & \\ CMSU_{21} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{2^l,1} & CMSU_{2^l,2} & \cdots & CMSU_{2^l,2^l} \end{pmatrix}. \tag{4.4}$$

Since it is a lower triangular matrix, we get that the eigenvalues of PU are the eigenvalues of the diagonal blocks. Moreover, in the matrix $(PU)^n$, where n is the fixed number of generations, the elements of the first 2^{ml} columns are related to the probabilities of convergence to the global optimum. With the initial distribution p_0 (a row vector) being known beforehand, to compute the probability of convergence to the global optimum, we have to sum up the first 2^{ml} elements of $p_0(PU)^n$. Without loss of generality, we can assume that $p_0 = (p_{0,1}, p_{0,2}, \ldots, p_{0,2^{(m+1)l}})$. As an alternative, we compute the summation of the elements other than the first 2^{ml} ones of $p_0(PU)^n$.

In equation (4.4), the matrix PU can be represented as

$$PU = \begin{pmatrix} CMSU_{11} & 0 \\ R & A \end{pmatrix}, \tag{4.5}$$

where

$$R = \begin{pmatrix} CMSU_{21} \\ \vdots \\ CMSU_{2^l,1} \end{pmatrix}, \tag{4.6}$$

and

$$A = \begin{pmatrix} CMSU_{22} & & & \\ CMSU_{32} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{2^l,2} & CMSU_{2^l,3} & \cdots & CMSU_{2^l,2^l} \end{pmatrix}. \tag{4.7}$$

Then,

$$(PU)^n = \begin{pmatrix} (CMSU_{11})^n & 0 \\ \sum_{i=0}^{n-1} A^i R (CMSU_{11})^{(n-1)-i} & A^n \end{pmatrix}. \tag{4.8}$$

Let $v(n) = p_0(PU)^n$. Since A is a square matrix, for the element $a_{ij}^{(n)}$ of A^n, the inequality (4.3) holds. Hence, we have

$$\sum_{i=2^{ml}+1}^{2^{(m+1)l}} v_i(n) = \sum_{i=1}^{2^{(m+1)l}-2^{ml}} p_{0,2^{ml}+i} \sum_{j=1}^{2^{(m+1)l}-2^{ml}} a_{ij}^{(n)} \leq (2^{(m+1)l} - 2^{ml}) K n^{m_{max}-1} |\lambda_{max}|^n, \tag{4.9}$$

The following shows $|\lambda_{max}| < 1$. Since the upgrade matrix upgrades some columns of A to the first 2^{ml} columns of PU, the sum of each row of A is less than 1. Therefore, if λ is an eigenvalue of $A = (a_{ij}^{(1)})$ and $u = (u(i))$ is its left eigenvector, we have

$$\lambda u(j) = \sum_{i=1}^{2^{(m+1)l}-2^{ml}} u(i) a_{ij}^{(1)}, \ j = 1, 2, \ldots, 2^{(m+1)l} - 2^{ml}. \tag{4.10}$$

That is,

$$\sum_{j=1}^{2^{(m+1)l}-2^{ml}} |\lambda u(j)| \leq \sum_{i=1}^{2^{(m+1)l}-2^{ml}} |u(i)| \sum_{j=1}^{2^{(m+1)l}-2^{ml}} a_{ij}^{(1)} < \sum_{i=1}^{2^{(m+1)l}-2^{ml}} |u(i)|. \tag{4.11}$$

Hence, $|\lambda| < 1$. Since the eigenvalues of PU are the eigenvalues of the diagonal blocks, we get that the set of eigenvalues of A is contained in the set of eigenvalues of PU. Moreover, since $CMSU_{11}$ is a regular (stochastic) matrix (Theorem 3 in [Rudolph, G. (1994)]), from Lemma 1 and Theorem 5, we obtain that $\lambda_1 = 1$ is an eigenvalue of $CMSU_{11}$, i.e., $\lambda_1 = 1$ is an eigenvalue of PU, and λ_1 is *simple*. If we agree that the eigenvalues, $\lambda_1, \ldots, \lambda_k$, $k \leq 2^{(m+1)l}$, of PU are in descending order $1 = \lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_k|$, with $m_1, m_2, \ldots, m_k$ being the (algebraic) multiplicities, respectively, the following inequality can be derived.

$$\sum_{i=2^{ml}+1}^{2^{(m+1)l}} v_i(n) \leq K' n^{m'_{max}-1} |\lambda_2|^n, \tag{4.12}$$

where $K' > 0$ and m'_{max} is the maximal (algebraic) multiplicity of the eigenvalues of PU. That is, regardless the initial distribution of the populations, the probability of convergence to the global optimum is greater than or equal to $1 - K' n^{m'_{max}-1} |\lambda_2|^n$.

If $K' n^{m'_{max}-1} |\lambda_2|^n > 1$, that means either the fixed n is not large enough, or λ is not small enough, this equation is meaningless. For any optimization problem, we first need to

compute its transition matrix CMS, extend it to PU, find the eigenvalue λ_2, which is related to the convergence rate of the problem, of PU, and then apply λ_2 to $K'n^{m'_{max}-1}|\lambda_2|^n$. With n satisfying $K'n^{m'_{max}-1}|\lambda_2|^n \leq 1$, we can obtain the GA's hardness of the problem (i.e., the applicability of GA to the problem). Nevertheless, it is concluded that the computations described above are time consuming and impractical for the current state of the art. By the following subsection, we get that traces of the transition matrices $(PU)^n$ is highly related to $|\lambda_2|$. In Chapter 5, the relationship is modified and applied so that an alternative approach to substituting the estimation of $|\lambda_2|$ is proposed to estimate the applicability of GAs to optimization problems. Chapter 7 uses the proposed method in Chapter 5 and provides a possible approximation method to estimate the number of generations needed for the global convergence.

4.1.1 Eigenvalues of Transition Matrices and Fix Points of Genetic Algorithm

The convergence rate is shown to be related to $|\lambda_2|$, the second largest eigenvalue of the transition matrix in absolute value. This subsection provides the relationship between the second largest eigenvalue of the transition matrix and the fix points of GA.

Theorem 6 demonstrates that every square matrix is unitarily equivalent to a triangular matrix whose diagonal entries are the eigenvalues of the matrix. Hence, the transition matrix $(PU)^n$, representing transition probabilities for every n generations of a GA, can be written as

$$(PU)^n = (STS^*)^n = ST^nS^*, \tag{4.13}$$

where S is a unitary matrix and T is upper triangular with diagonal entries are the eigenvalues of PU. Let the eigenvalues of PU be denoted as $\lambda_1 = 1, \lambda_2, \ldots, \lambda_{2^{(m+1)l}}$ (in absolute descending order). Then T^n is an upper triangular matrix with diagonal entries are $\lambda_1^n, \lambda_2^n, \ldots, \lambda_{2^{(m+1)l}}^n$. Moreover, the trace of $(PU)^n$ is

$$Trace((PU)^n) = Trace((ST^n)S^*) = Trace(S^*ST^n) = Trace(T^n) = \sum_{i=1}^{2^{(m+1)l}} \lambda_i^n. \tag{4.14}$$

By the inequality

$$|\sum_{i=1}^{2^{(m+1)l}} \lambda_i^n| \leq \sum_{i=1}^{2^{(m+1)l}} |\lambda_i|^n \leq 1 + (2^{(m+1)l} - 1)|\lambda_2|^n, \tag{4.15}$$

it is obtained that

$$Trace((PU)^n) \leq 1 + (2^{(m+1)l} - 1)|\lambda_2|^n. \tag{4.16}$$

That is, $|\lambda_2|^n$ is bounded (from the left) by the trace of the transition matrix $(PU)^n$. Since $|\lambda_i| < 1$ for $i \in \{2, \ldots, 2^{(m+1)l}\}$, there exists a large number $M_1 > 0$ such that

$$Trace((PU)^n) = 1 + \lambda_2^n + \cdots + \lambda_{2^{(m+1)l}}^n \geq C_1, \ \forall n \geq M_1,$$

where $C_1 \in (0, 1]$ is a constant. That implies that there exists a constant K" such that

$$1 + (2^{(m+1)l} - 1)|\lambda_2|^n \leq 1 + (2^{(m+1)l} - 1) \leq K"Trace((PU)^n), \ \forall n \geq M_1. \tag{4.17}$$

Equation (4.17) shows that if n is large enough, $Trace((PU)^n)$ with a constant multiple can also form an upper bound for $|\lambda_2|^n$. Combined with Inequality (4.16), it can be known that computing λ_2 is equivalent to computing the trace of the transition matrix. Additionally, the diagonal element of the transition matrix corresponding to each state shows probability of a fix point over the solutions in the GA search space. The product of any distribution of transient states and the diagonal elements of the transition sub-matrix corresponding to the transient states represents the total probability of fix points, which are outside the set of optimal populations (absorbing states) of GA. Note that a population ξ^* is called an optimal population if there exists i, $1 \leq i \leq m + 1$, such that $\pi_i(\xi^*)$ is an optimal solution. For searching purpose, we do not want GA to stay within a transient state too long since the state is already visited. The revisiting of a state will cost some time without any improvement of solutions. In other words, if the total probability of fixed points of GA can be reduced, the probability of GA to search other candidate states will be increased for the number of the states is finite. In general, when n is large enough (e.g., $n \geq 10$), the smaller the trace, the better the performance of GA.

4.2 Second Largest Eigenvalue versus Expected First Hitting Time

The research on computation time of GAs used to solve optimization problems is important for the foundation and theory of evolutionary algorithms. Both convergence rate and expected first hitting time express the information on time complexity.

From Ding's work [Ding, L. and Yu, J. (2005)], the *first hitting time* is defined as

$$\tau(\xi^*) = min\{k \geq 0 | \xi_k = \xi^*\},$$

where ξ^* is an optimal population, and $\{\xi_k | k \in N\}$ is a discrete homogeneous Markov chain. Moreover, the expected first hitting time is calculated as

$$E[\tau(\xi^*)] = \sum_{k \geq 0} k \times P\{\tau(\xi^*) = k\}. \tag{4.18}$$

As mentioned in Chapter 2, a Markov theorem tells us that the expected value of the transient time in which an arbitrary state i goes to any other state j is finite. That is, there exists $0 < M_1 < \infty$ such that

$$E[\tau(\xi^*)] < M_1,$$

since the optimal state is also a state in the state space of the Markov chain. With respect to the same problem, if the second largest eigenvalue in absolute value of PU: $\lambda_2 \neq 0$ (i.e., CMS does not have identical rows), we can get that there exists $0 < C_1 < \infty$ such that

$$E[\tau(\xi^*)] < C_1 |\lambda_2|. \tag{4.19}$$

For an arbitrary fitness function, an inequality similar as Inequality (4.19) can be derived. In fact, if a problem has the corresponding $\lambda_2 \neq 0$, it is possible to use λ_2 to estimate the upper bound for the expected first hitting time. Suppose $\lambda_2 \neq 0$ is given while m and l are also given. Based on the arrangement of eigenvalues in Section 4.1.1, the inequality

$$1 + |\lambda_2|^n + |\lambda_3|^n + \cdots + |\lambda_{2^{(m+1)l}}|^n \leq 1 + (2^{(m+1)l} - 1)|\lambda_2|^n$$

always holds. For a small $\epsilon > 0$, the solution of $\epsilon = (2^{(m+1)l} - 1)|\lambda_2|^n$ on n can be an estimation of the upper bound for the expected first hitting time.

CHAPTER 5. THE EVALUATION METRICS

5.1 The Bridge between Theory and Practice

Although the expected first hitting time and the convergence rate can be mathematically computed, there is still a huge gap between the theoretical prediction and the estimation on the applicability of GAs to problems in real world applications. As I know, the calculations on the expected first hitting time or convergence rate include the computation on the corresponding transition matrix of problems. To obtain the transition matrix with respect to a problem, the matrices C, M, and S should be considered. Since S is derived from selection operator, in which the proportional selection is applied, the construction time is much longer than the total computation time on the fitness values of the entire search space. Hence, the theory is only an ideal view. The real world demands a practical approach for the estimation on applicability of GAs.

In order to derive a more practical approach, a direction related to Markov chain analysis is suggested. It is introduced as follows.

5.1.1 Equivalent Forms in terms of Convergence

The extended transition matrix PU is used to prove that the CGA with best solution maintained converges to the global optimal solution. Because the computation time of the transition matrix is not practical, CMS and PU cannot be computed directly. PU includes even more states than CMS. Our goal is to find a way which can extract only the essential properties of the transition matrix so that an evaluation metric can be formulated to evaluate the applicability of GAs to real-world problems.

Instead of PU, the transition matrix CMS is used to develop the metric in the next section.

According to Theorem 3, the convergence of $(PU)^n$ as $n \to \infty$ is proved; however, it does not show much information about the relationship between $(PU)^n$ and $(CMS)^n$, especially when n is finite. In order to further investigate the convergence behavior among $(PU)^n$, $(CMS)^n$, and A^n (the sub-matrix of $(PU)^n$ in Equation (3.9)) within finite steps (i.e., $n \in N$ and $n < \infty$), Theorem 7 is formulated. Note that the term *empirical convergence* is defined in Definition 8.

Theorem 7 *For $n \in N$, the empirical convergence of $(PU)^n$ is equivalent to the empirical convergence of both $(CMS)^n$ and A^n, where A is the sub-matrix of PU in Equation (3.9).*

Proof. By Definition 8, the empirical convergence of $(PU)^n$ directly implies the empirical convergence of both $(CMS)^n$ and A^n.

Now suppose that both $(CMS)^n$ and A^n converge empirically, respectively. To prove the empirical convergence of $(PU)^n$, one has to first verify with respect to each row of blocks of $(PU)^n$, if all of the blocks are summed, the result is equal to $(CMS)^n$, for all $n \in N$. The statement is proved by applying Mathematical Induction on n (the steps).

First, suppose that $n = 2$. Based on the structure of PU (e.g., Figure 3.2), the property can be obtained that each row of blocks has exactly 2^{ml} (the order of CMS) nonzero columns. Let all of the 2^{ml} columns be named from left to right as $column_1$, $column_2$, ..., $column_{2^{ml}}$. It can be shown that their positions within any row of blocks are distinct and sorted. That is, within a row of blocks, no matter which block it is in, $column_1$ is always the first column in the block, $column_2$ is always the second column in the block, and so on. To compute $PU \cdot PU$, for any row i, $1 \leq i \leq 2^{(m+1)l}$, of the left PU, the first nonzero entry, which is in the first column, is multiplied with the first row of the right PU, the second nonzero entry, which is in the second column, is multiplied with the second row in the corresponding row of blocks of the right PU, and so on. (Figure 5.1 shows an illustration of $PU \cdot PU$, where $m = 1$ and $l = 3$.) Since the 2^{ml} nonzero entries in the i^{th} row of the left PU are from a row of CMS, and rows with 2^{ml} nonzero entries of the right PU are from rows of CMS, it can be derived that

$$Row_j((CMS)^2) = \sum_{g=1}^{2^l} Row_j((PU)^2_{hg}), \tag{5.1}$$

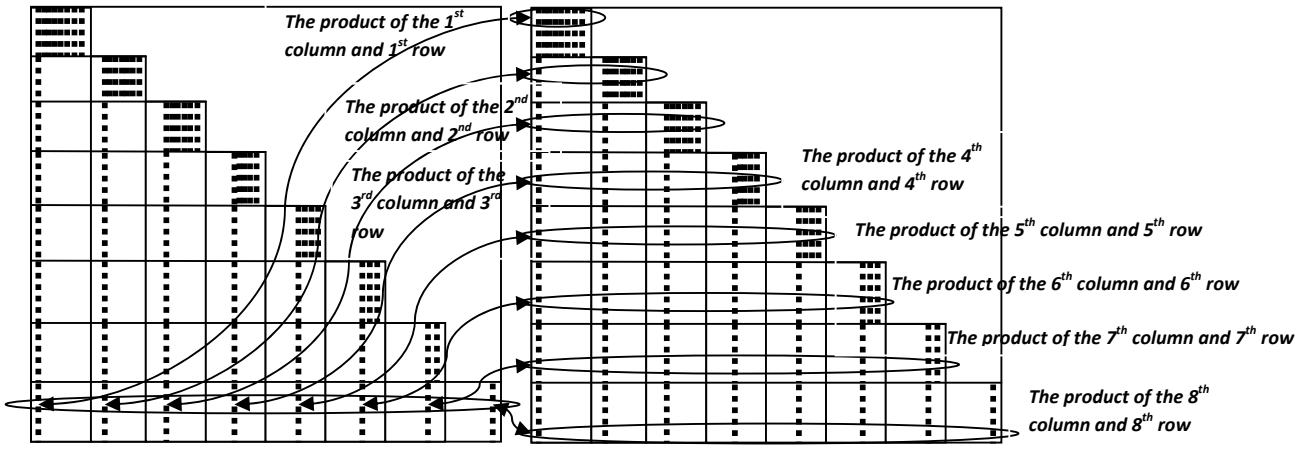


Figure 5.1 An illustration of the Product $PU \cdot PU$

where $j = (i \ mod \ 2^{ml})$, $Row_j(\cdot)$ indicates the j^{th} row in the block, $(PU)^2_{uv}$ is the block in the u^{th} row and v^{th} column of $(PU)^2$, and $h = \lceil \frac{i}{2^{ml}} \rceil$. From Equation (5.1), it can be obtained that with respect to each row of blocks of $(PU)^2$, if all of the blocks are summed, the result is equal to $(CMS)^2$. In other words, the statement holds for $n = 2$.

Suppose for $n = k$, the statement holds. That is, with respect to each row of blocks of $(PU)^k$, if all of the blocks are summed, the result is equal to $(CMS)^k$.

For $n = k + 1$, the product $PU \cdot (PU)^k$ needs to be computed. Let $(PU)^k_{uv}$ be the block in the u^{th} row and v^{th} column of $(PU)^k$. For any row i, $1 \leq i \leq 2^{(m+1)l}$, of PU, assume C_1, C_2, ..., $C_{2^{ml}}$ are the aforementioned nonzero entries, $j = (i \ mod \ 2^{ml})$, and $h = \lceil \frac{i}{2^{ml}} \rceil$. Then, based on the rule for the product of two matrices, it is derived that

$$Row_j((PU)^{k+1}_{hw}) = \sum_{z=0}^{h-w-1} C_{w+z} Row_{w+z}((PU)^k_{(w+z)w}) + \sum_{x=h}^{2^{ml}} C_x Row_x((PU)^k_{hw}),$$

if $w \leq h$, and

$$Row_j((PU)^{k+1}_{hw}) = (0, \ldots, 0)_{2^{ml}},$$

if $w > h$. Hence, it is obtained that

$$\sum_{w=1}^{2^l} Row_j((PU)^{k+1}_{hw}) = \sum_{w=1}^{h} Row_j((PU)^{k+1}_{hw}) \tag{5.2}$$

$$\begin{aligned}
=&C_1 Row_1((PU)^k_{11}) + C_2 Row_2((PU)^k_{21} + (PU)^k_{22}) + \\
&C_3 Row_3((PU)^k_{31} + (PU)^k_{32} + (PU)^k_{33}) + \cdots + \\
&C_{h-1} Row_{h-1}((PU)^k_{(h-1)1} + \cdots + (PU)^k_{(h-1)(h-1)}) + \\
&C_h Row_h((PU)^k_{h1} + \cdots + (PU)^k_{hh}) + \\
&C_{h+1} Row_{h+1}((PU)^k_{h1} + \cdots + (PU)^k_{hh}) + \\
&\cdots + C_{2^{ml}} Row_{2^{ml}}((PU)^k_{h1} + \cdots + (PU)^k_{hh}).
\end{aligned}$$

Since it is already known that with respect to each row of blocks of $(PU)^k$, if all of the blocks are summed, the result is equal to $(CMS)^k$ (i.e., the statement holds for $n = k$), it can be derived that the right hand side (RHS) of Equation (5.2) is equal to $Row_j((CMS)^{k+1})$. Hence, the statement holds for $n = k + 1$. That is, with respect to each row of blocks of $(PU)^{k+1}$, if all of the blocks are summed, the result is equal to $(CMS)^{k+1}$.

Secondly, the empirical convergence of $(PU)^n$ needs to be proved. Let $\epsilon > 0$ be a small number, and $u^{(n)}_{ij}$ be the element in the i^{th} row and j^{th} column of $(PU)^n$.

By Definition 8 and the fact that both $(CMS)^n$ and A^n converge empirically, respectively, Inequality (5.3) and Inequality (5.4) are derived. For $\frac{\epsilon}{2} > 0$, $\exists$ a constant $n_1 > 0$ such that

$$|v^{(n)}_{ij} - v^*_j| < \frac{\epsilon}{2}, \forall n \geq n_1, \tag{5.3}$$

where $v^{(n)}_{ij}$ is the element in the i^{th} row and j^{th} column of $(CMS)^n$, and $e(v^*)^T$ is the limit of $(CMS)^n$ (see Theorem 2). In addition, for $\frac{\epsilon}{2} \cdot \frac{1}{2^l - 1} > 0$, $\exists$ a constant $n_2 > 0$ such that

$$|a^{(n)}_{ij} - 0| < \frac{\epsilon}{2(2^l - 1)}, \forall n \geq n_2, \tag{5.4}$$

where $a^{(n)}_{ij}$ is the element in the i^{th} row and j^{th} column of A^n, and 0 is the limit of A^n. Then

$\exists n_3 = \max\{n_1, n_2\}$ such that

$$\begin{aligned}
|u_{ij}^{(n)} - v_j^*| &= |u_{ij}^{(n)} + \sum_{k=1}^{2^l-1} a_{i,k\cdot 2^{ml}+j}^{(n)} - v_j^* - \sum_{k=1}^{2^l-1} a_{i,k\cdot 2^{ml}+j}^{(n)}| \\
&= |v_{ij}^{(n)} - v_j^* - \sum_{k=1}^{2^l-1} a_{i,k\cdot 2^{ml}+j}^{(n)}| \\
&\leq |v_{ij}^{(n)} - v_j^*| + \sum_{k=1}^{2^l-1} |a_{i,k\cdot 2^{ml}+j}^{(n)}| \\
&< \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon, \forall n \geq n_3,
\end{aligned}$$

where $2^{ml} + 1 \leq i \leq 2^{(m+1)l}$ and $1 \leq j \leq 2^{ml}$. For other i and j, the proof is trivial. It is derived directly from Inequality (5.3) and Inequality (5.4).

That is, from the statement of Mathematical Induction and the fact that both $(CMS)^n$ and A^n converge empirically, respectively, the empirical convergence of PU is proved. □

Based on Inequality (3.4) and Inequality (4.12), it is implied that the empirical convergence of $(CMS)^n$ and A^n depend on $|\lambda_{2\_CMS}|$ (the second largest eigenvalue of CMS in absolute value) and $|\lambda_2|$, respectively. Since CMS is one of the diagonal blocks of PU, $|\lambda_{2\_CMS}| \leq |\lambda_2|$. If $0 < \lambda_{2\_CMS} < 1$, sufficiently small $\lambda_{2\_CMS}^n$ implies that n is large enough. Hence, $|\lambda_2|^n$ is small enough for the empirical convergence of A^n. Suppose after $(n-1)$ generations, the elements in A^{n-1} satisfy

$$|a_{ij}^{(n-1)} - 0| < \frac{\epsilon}{2^{(m+1)l} - 2^{ml}}, \text{ for some } \epsilon.$$

Then regardless of the initial distribution,

$$P\{\text{The } n^{th} \text{ Generation is an Optimal State}\} > 1 - \epsilon.$$

5.2 The Proposed Evaluation Metric

5.2.1 Overview of the Methodology

This section proposes an evaluation metric, which derives the applicability of a GA to a problem from the estimated trace of the corresponding transition matrix. Through this

method, a degree of convergence can be determined for each GA run. According to that, researchers and engineers will be able to obtain a comprehensive view of the applicability of GAs and know how good the solutions generated by GAs are, so that more correct decisions can be made. The general methodology diagram is illustrated in Figure 1.1.

5.2.2 Evaluating the Applicability of GAs from Run Time Data

The major concerns for the applicability of GAs to real-world problems are the computation time and the quality of the obtained solutions. Actually, there are tradeoffs between them. For instance, typically the longer the computation time, the better the solution is. Since they are tightly related, without loss of generality, we can fix one and utilize another to estimate the applicability of GAs to problems. Assumption 3 forces us to fix the computation time, whereas the quality of the obtained solution will be used to measure the problem applicability of GAs.

Due to the lack of information on the landscapes of fitness functions and encodings, it is difficult to analyze the quality of the obtained solution itself; however, the quality of the obtained solution can be measured indirectly through the degree of convergence under certain evaluation metrics from run time data. Theorem 8 functions as a support for the evaluation metric on degrees of convergence. The reason for employing the trace of $(CMS)^{n'}$, where n' is fixed, to estimate degrees of convergence is because it quantifies degrees of convergence. With it, a number on the degree of convergence of GA will be given at the end of a GA run. Based on that, one can easily determine whether or not the GA run converges within n' generations.

Theorem 8 *The trace of* $(CMS)^n$ *converges to* 1 *as* $n \to \infty$.

Proof. By Theorem 1, 5, and Lemma 1, it is obtained that CMS has only one eigenvalue which is equal to 1 and others are less than 1.

Moreover, by Theorem 6, every square matrix is unitarily equivalent to a triangular matrix whose diagonal entries are the eigenvalues of the matrix. Hence, $(CMS)^n$ can be written as

$$(CMS)^n = (STS^*)^n = ST^nS^*,$$

where S is a unitary matrix and T is upper triangular. The diagonal entries of T are the eigenvalues of CMS. Let the eigenvalues of CMS be denoted as $\lambda_1, \ldots, \lambda_{2^{ml}}$ with $1 = \lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_{2^{ml}}|$. Then

$$Trace((CMS)^n) = Trace((ST^n)S^*) = Trace(S^*ST^n) = Trace(T^n) = \sum_{i=1}^{2^{ml}} \lambda_i^n. \quad (5.5)$$

From Equation (5.5), the result, the trace of $(CMS)^n$ converges to 1 as $n \to \infty$, is derived. □

In fact, Theorem 8 coincides with a result from Theorem 2. In Theorem 2, since CMS is regular, it converges to a positive stable stochastic matrix $\prod = ev^T$ as $n \to \infty$, where $e = (1, 1, \cdots, 1)^T$ is a $2^{ml} \times 1$ column vector in which all elements are of the value 1, and $v^T = (v_1, v_2, \cdots, v_r)$ is a 1×2^{ml} probability row vector with non-null entries. Based on that, the result, $\lim_{n\to\infty} Trace((CMS)^n) = 1$, can be obtained.

5.2.3 Theoretical Framework for Approximating the Applicability of GAs

This subsection mainly discusses the methodology adopted to evaluate the applicability of GAs based on Theorem 8.

The overview of the idea includes:

1. Applying a Monte-Carlo-like simulation (Empirical Probability) [Leemis, L. H. and Park, S. K. (2005)] to estimate $Trace((CMS)^{n'})$, $n' \in N$, from states visited by a GA run. Note that n' should be chosen carefully so that $n - n'$ can be a large number, where n is the number of generations.

2. Comparing the obtained $Trace((CMS)^{n'})$ with the "expected" value 1, and then deriving a value for the degree of convergence with respect to that GA run.

In Assumption 3, n, the number of generations, is a reasonably large number and it is fixed. After a GA run, there will be n generations in total. Each generation corresponds to a state from the state space of the Markov transition matrix CMS. During a GA run, a state

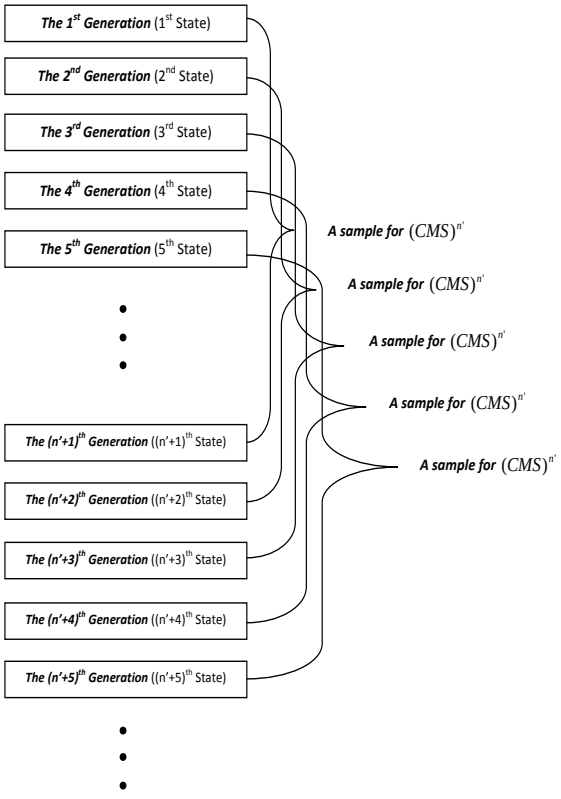


Figure 5.2 The Estimation on the Trace of $(CMS)^{n'}$

representing the next generation will be generated after every mutation operation. If all of the n generations (states) are recorded, a list of n states sorted chronologically will be generated at the end of each GA run. With that list, a number $n' \in N$, where $n' < n$ and $n - n'$ is a large number, is chosen. The starting states and the ending states are determined accordingly to estimate the trace of $(CMS)^{n'}$. Figure 5.2 illustrates this concept; it shows that for each generation (state) i, $1 \leq i \leq (n - n')$, given as a starting state, the $(i + n')^{th}$ generation (state) is the corresponding ending state. After the matching, there are $(n - n')$ pairs of (Starting State, Ending State). Based on the results, one can compute the frequency, or probability, of any starting state which goes back to itself. If all the obtained frequencies are summed up, the obtained value is the approximation of $Trace((CMS)^{n'})$. Note that an ending state of a pair can also be a starting state of another pair.

According to the my empirical investigation, the transition matrix CMS constructed by CGA has the property that its eigenvalues, which have large absolute values, are all real and positive if p_m is small (e.g. $p_m < 0.45$). Only a few eigenvalues of CMS, whose absolute values are close to 0, are complex numbers. Intuitively, the transition matrix M constructed by a small mutation probability p_m is symmetric and strictly diagonally dominant (i.e., M is a positive definite matrix). It has eigenvalues which are all real and positive. The sparse matrices C and S do not significantly impact the eigenvalues of M. Therefore, the eigenvalues of CMS primarily depend on the eigenvalues of M. This fact implies that $Trace((CMS)^{n'})$ decreases to 1 as n' increases.

Based on the discussion above and Theorem 8, to determine the degree of convergence one simply needs to compare the obtained value with 1. If the value (or criterion) |Obtained Value$-$ 1| is close to 0, the degree of convergence is high. In other words, the degree of convergence with respect to a GA run depends on the estimation of $Trace((CMS)^{n'})$ from run time data. With n' being fixed, one can compare different encodings of the same problem, or obtain the values on how good the solutions are for several GA runs with respect to different problems.

The merit of this approach is that for every GA run, a degree of convergence can be derived. Researchers can evaluate the obtained solution based on the degree of convergence. Since it is already shown that the convergence of $(PU)^n$ is equivalent to the convergence of both $(CMS)^n$ and A^n in Equation (3.9) in Theorem 7, Inequality (4.12) tells us that if the best solution is recorded, it has low probability that a premature convergence occurs in the n'^{th} generation when the run time estimation on |Obtained Value $-$ 1| is close to 0 (i.e., high degree of convergence). The inverse statement, with the maintenance of the best solution, a premature convergence has low probability to acquire the estimation on |Obtained Value $-$ 1| close to 0 from run time data. Figure 5.3 is an illustrative example of the relationships among the fitness function, the extended transition matrix PU, and the transition matrix CMS with the best solution maintained under a premature condition.

In Figure 5.3, the dashed lines show the encoded points for the fitness function, and the numbers represent the order of fitness values sorted from highest to lowest: The highest fitness

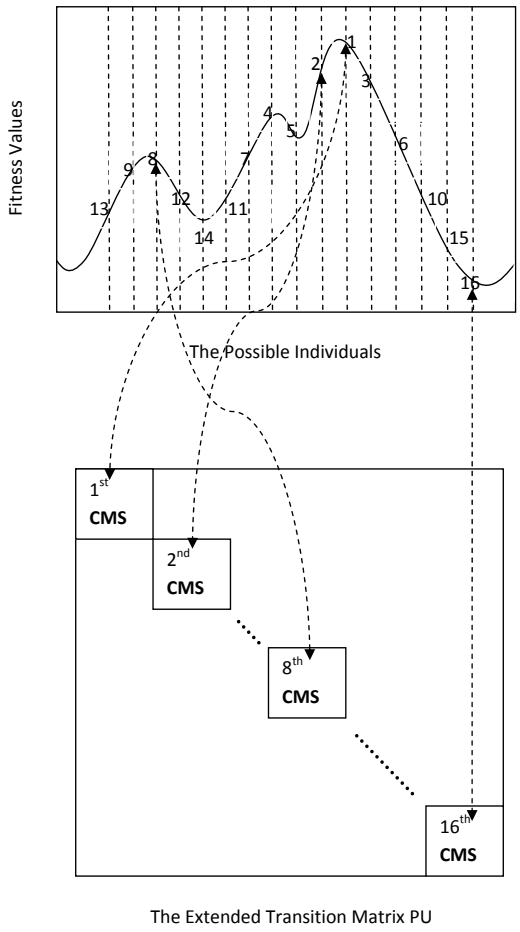


Figure 5.3 An example for fitness function, PU, and CMS

value, number 1, corresponds to the first diagonal block of CMS in PU; the second highest fitness value, number 2, corresponds to the second diagonal block of CMS in PU, and so on. Now, suppose that in a GA run, a premature convergence occurs because the population stays within the local optimal solution, number 8. One can show that the probabilities of some states corresponding to the 8^{th} diagonal block of CMS, going to some states corresponding to the 8^{th} diagonal block of CMS, are not small. Therefore, the probability for the estimation on |Obtained Value $-$ 1| being not close to 0 is high.

5.2.4 Foundation of the Methodology

According to Theorem 8 and the discussion in previous subsection, it can be derived that the trace of $(CMS)^n$ determines the degree of convergence. In other words, to obtain the degree of convergence, one can calculate the trace of $(CMS)^n$ first. A traditional way to compute it is first to compute the transition matrix CMS; however, as mentioned, the construction time of CMS takes more than the total computation time of all feasible solutions. Therefore, the traditional way is considered to be impractical due to the large computation time.

My methodology to approximate the trace of $(CMS)^{n'}$ is based on the following. In Figure 5.2, the first generation (state) is chosen at random from all feasible solutions. After the first generation is determined, the second generation can be generated by the crossover, mutation, and selection operators of a GA accordingly. In the perspective of Markov chains, the distribution of the second generation is generated based on the product of the distribution of the first generation and the transition matrix CMS. Similarly, the distribution of the third generation is generated based on the product of the distribution of the first generation and the transition matrix $(CMS)^2$, and so on. In general, for $i \in N$, if the distribution of the i^{th} generation is determined, the distribution of the $(i+n')^{th}$ generation can be generated based on the product of the distribution of the i^{th} generation and the transition matrix $(CMS)^{n'}$. To estimate the trace of $(CMS)^{n'}$, the i^{th} generation and the $(i+n')^{th}$ generation need to be coupled $(i, n' \in N)$. Now consider the physical meaning of the trace of $(CMS)^{n'}$. The diagonal elements of $(CMS)^{n'}$ represent the probabilities of the fix points (states) from the i^{th}

generation to the $(i+n')^{th}$ generation for a GA. Therefore, the summation of the frequencies (probabilities) of any starting state going back to itself approximates the trace of $(CMS)^{n'}$.

A question about the distribution of samples may be raised. Besides the first generation, the i^{th} generation ($i \geq 2$) is dependent on the $(i-1)^{th}$ generation. That is, after the first generation, the starting state is not chosen uniformly. According to my investigation, this dependency will not cause any problem. On the other hand, it helps improve the accuracy of the approximation. Suppose the transition matrix

$$(CMS)^{n'} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1,2^{ml}} \\ t_{21} & t_{22} & \vdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ t_{2^{ml},1} & \cdots & \cdots & t_{2^{ml},2^{ml}} \end{pmatrix}. \tag{5.6}$$

For the purpose of illustration, we may assume that t_{11} has a large value, say 0.99, and $t_{2^{ml},2^{ml}}$ has a small value, say 0.01. If the first state (i.e., the state corresponds to the element t_{11} in Equation (5.6)) is generated as the i^{th} generation ($i \leq n$), in the $(i+n')^{th}$ generation, it has a high probability (0.99) to be chosen again. Other states have lower probability to become the $(i+n')^{th}$ generation. Similarly, if the last state is generated as the i^{th} generation, in the $(i+n')^{th}$ generation, it has merely 0.01 probability to be chosen again. Other states have a large probability to become the $(i+n')^{th}$ generation. Accordingly, a state with a large diagonal value has a high probability to become a starting state, and more samples will be collected for estimating it. A state with a small diagonal value may be ignored from the estimation. Compared with the samples collected uniformly, it increases the accuracy of the approximation since the number of samples is fixed (by n and n') and the large diagonal values dominate the trace. If more samples with the starting states being the states with small diagonal values are gathered, fewer samples with starting states being the states with large diagonal values will be collected. In that case, the estimations of large diagonal values as well as the approximation of the trace may not be so accurate.

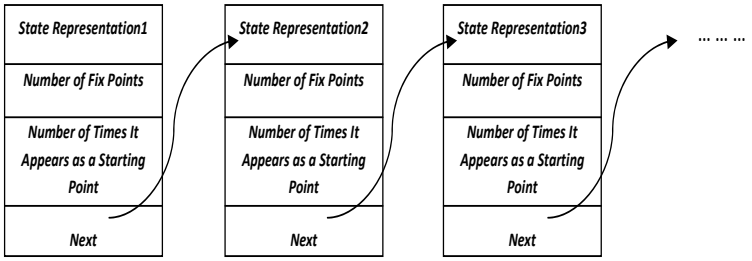


Figure 5.4 The Data Structure of the Proposed Methodology

5.2.5 The Implementation and Data Structure of the Proposed Methodology

In this subsection, the implementation and a solution for the data structure of the proposed methodology are described. After introducing the data structure, the implementation is discussed accordingly.

Since the only concern for the estimation is the approximated trace of $(CMS)^{n'}$, the data structure can be designed as in Figure 5.4 where it can be seen that the linked list is adopted for the data structure. In each node, there are four fields. The first field represents the starting state, which is a binary string. The next two fields are the numbers related to the starting state in the first field. That is, the second field represents the number of ending states which are the same as the starting state in the first field, and the third field represents the total occurrence of the starting state in the first field. The fourth field represents the address of the next node.

The following shows how the proposed methodology is implemented:

1. Set the list to be NULL.
2. Record the first n' states as starting states using an array.
3. Start from the $n' + 1$ state, do 4) to 7) until the stop condition is satisfied.
4. Search for the corresponding starting state in the list. If it is not found, insert the corresponding starting state in the list.

5. Compare the current state with the corresponding starting state. If it is the same as the starting state, add 1 to the second field.

6. Add 1 to the third field.

7. Replace the corresponding starting state with the current state in the array.

Since for each generation the mutation operator sweeps the entire state once, all of the states can be recorded at the same time as the mutation. The recording will not increase the complexity of the computation time. In the next subsection, detailed information on the space and time complexity is calculated.

5.2.6 Complexity Analysis

5.2.6.1 Space Complexity

Based on the implementation in the previous section, an array and a list must be maintained during a GA run. Each element in the array needs to record a binary string with length ml. Since there are n' states preserved in the array, the space complexity of the array is $O(n'ml)$. Moreover, there are $(n - n')$ pairs of (Starting State, Ending State). Therefore, the list has to maintain at most $(n - n')$ nodes. Similar to the array, each node in the list needs to record a binary string with length ml, so the space complexity of the list is $O((n - n')ml)$. Hence, the total space complexity is $O(nml)$.

5.2.6.2 Time Complexity

The extra time needed for the estimation includes the time to record the states in the array (Step 2 and 7 in the previous section), the time to search the corresponding starting states in the list (Step 4), and the comparison time for the current states and the corresponding starting states (Step 5).

Since there are in total n states, the time to record the states in the array is $O(nml)$. For each generation, the search time depends on the length of the list. As mentioned previously,

the list is initialized as NULL. Therefore, the search time is at most

$$ml(1 + 2 + 3 + \cdots + (n - n')) = \frac{ml(n-n')(n-n'+1)}{2}. \quad (5.7)$$

The search time is determined to be $O((n-n')^2 ml)$. Moreover, due to the fact that there are $(n-n')$ pairs of (Starting State, Ending State), the comparison time for the current states and the corresponding states is $O((n-n')ml)$. Therefore, the total time complexity is $O(n^2 ml)$.

5.3 Experimental Validation of the Proposed Evaluation Metric

My goal for the experiments is to investigate the usability of the proposed evaluation metric. As this metric is not sensitive to the GA parameter settings, we fix the value of some of these parameters to facilitate the experimental comparisons in the following sections. The fixed parameters are:

- the crossover probability p_c is set to 0.5
- the mutation probability p_m is set to 0.01 (a small value)

5.3.1 One Dimensional Fitness Functions

Eight fitness functions with different degrees of difficulty for GAs are selected and investigated. The eight fitness functions consist of: a linear function, a quadratic function, a periodic function, a fraction function, and four functions considered as (different degrees of) hard fitness functions for GAs [Horn, J. and Goldberg, D. E. (1995); Horn, J. (1995); Sareni, B. and Krahenbuhl, L. (1998); Petrowski, A. (1996)] (the one with the isolation feature without any guidance from fitness values is called *needle-in-a-haystack (NIAH)* [Horn, J. and Goldberg, D. E. (1995); Horn, J. (1995)]). The fitness functions and their optimal fitness values are presented in Table 5.1 (the default value for the domains is $[0, 600]$).

f_7 and f_8 are regarded as difficult fitness functions for CGA with best solution maintained over time because they have multiple modes [Sareni, B. and Krahenbuhl, L. (1998); Petrowski, A. (1996)]. f_7 consists of unequal spaced peaks of uniform heights, and f_8 consists of unequal spaced peaks of nonuniform heights.

Table 5.1 The Selected Fitness Functions

Fitness Function	Optimal Value
$f_1(x) = x$	600
$f_2(x) = 360000 - x^2$	360000
$f_3(x) = 1 + sin(x)$	2
$f_4(x) = \frac{1}{\sqrt{x+1}}$	1
$f_5(x) = \begin{cases} -x^2 + 90000, x < 300 \\ 100000, x = 300 \\ -(x-600)^2 + 90000, x > 300 \end{cases}$	100000
$f_6(x) = \begin{cases} 100000, x = 0 \\ -(x-300)^2 + 90000, x \neq 0 \end{cases}$	100000
$f_7(x) = sin^6(5\pi[x^{3/4} - 0.05])$	1
$f_8(x) = e^{-2(ln\ 2)(\frac{x-0.08}{0.854})^2} sin^6(5\pi[x^{3/4} - 0.05])$	1

5.3.2 Discussions on the Sizes of Samples

One may argue how we can be sure that the size of the samples is enough for deriving a proper approximation. To answer this question, the experiments on the sizes of samples with respect to the sizes of encodings for a problem are conducted.

For the following observations, I apply the theoretical truth, if the power n' is fixed, the value $Trace((CMS)^{n'})$ is fixed. With the setting on both the number of individuals in a generation $m = 2$ and the length of feasible solutions $l = 2$, CMS has a total of $2^4 = 16$ states. Table 5.2 lists the average results for fitness function f_2 from 20 trials of GA runs for the number of generations from 5000 to 10000 with the actual traces compared. Note that similar results are also derived for other fitness functions.

Table 5.2 The Estimated $Trace((CMS)^{n'})$ w.r.t. the Number of Generations and Power n'

Power n' / No. of Gens.	10	12	14	16	18	20
5000	2.7331	2.5614	2.4094	2.2540	2.1660	2.0517
6000	2.7343	2.5584	2.4156	2.2564	2.1737	2.0527
7000	2.7379	2.5620	2.4104	2.2631	2.1715	2.0532
8000	2.7333	2.5692	2.4143	2.2613	2.1742	2.0565
9000	2.7397	2.5672	2.4070	2.2584	2.1716	2.0539
10000	2.7351	2.5558	2.4105	2.2708	2.1671	2.0548
Actual Traces	3.3043	3.1863	3.0771	2.9737	2.8763	2.7843

As we can see, with the same n', no matter how many generations exist, the results for

estimated $Trace((CMS)^{n'})$ are almost the same for each one, and are close to actual traces. Therefore, it is concluded that those numbers of generations are large enough for $l = 2$ and $m = 2$.

Now let the length l be set to 16. Table 5.3 shows the average results for fitness functions f_1 and f_2 from 20 trials of GA runs for the number of generations from 5000 to 10000, respectively. Note that similar results are also derived for other fitness functions.

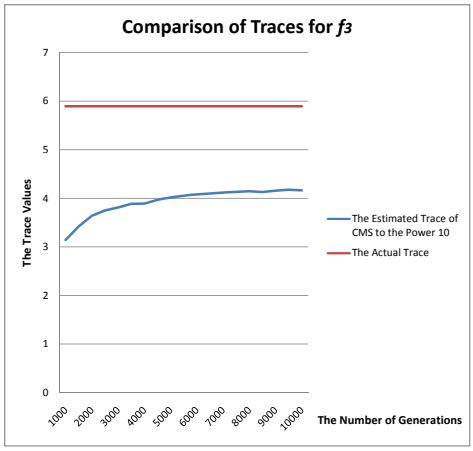
Table 5.3 The Estimated $Trace((CMS)^{n'})$ With Respect To the Number of Generations and Power n'

	Fitness Function f_1				Fitness Function f_2			
Power n' No. of Gens.	10	15	20	25	10	15	20	25
5000	7.0202	1.9653	1.2016	1.0667	6.5991	1.9553	1.3250	1.2053
6000	8.1490	2.1476	1.3326	1.0867	7.6623	1.9990	1.3516	1.1179
7000	9.7629	2.4942	1.4066	1.1031	9.2037	2.4565	1.3858	1.2133
8000	10.8016	2.7857	1.3546	1.1576	10.3304	2.6599	1.4224	1.2306
9000	12.5045	3.2925	1.4917	1.0618	11.8056	3.0480	1.5589	1.0657
10000	13.8309	3.6587	1.5685	1.2171	13.2582	3.3984	1.4045	1.1884

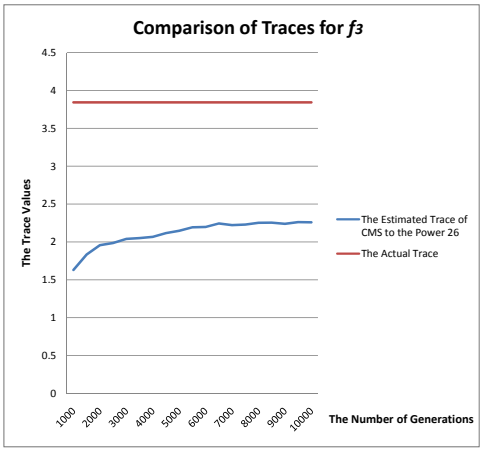
Table 5.3 shows that the results are different from previous results since as the number of generations increases, the estimated $Trace((CMS)^{n'})$ increases. This occurs because the numbers of samples for the 2^{32} states are relatively small. Some diagonal elements of $(CMS)^{n'}$ will not be calculated by the small numbers of samples; however, based on the data from Table 5.3, we get that both of the $Trace((CMS)^{n'})$ converge to 1 as n' increases. A conclusion can be drawn that although the number of samples sometimes may not be large enough for a large l, for some n', the estimated $Trace((CMS)^{n'})$ is still referenceable. The setting of the n' depends on the setting of l and experiences.

As mentioned before, it is believed that large diagonal elements are easily estimated by the proposed metric. To verify, I have run several experiments. The result of f_2 is presented in Figure 5.5(a) and 5.5(b). Similar results can be derived for other fitness functions. As shown, the estimated $Trace((CMS)^{n'})$ increases rapidly before a large enough number of generations (e.g., $n = 5000$) and after, it increases asymptotically to its actual value. If we choose the fixed number of generations close to the turning point ($n = 5000$), a certain amount (percentage) of $Trace((CMS)^{n'})$ can be derived. The detailed data from Figure 5.5(a) and 5.5(b) is shown in

Table 5.4.



(a) Power $n' = 10$



(b) Power $n' = 26$

Figure 5.5 The relationship between estimated trace values, actual trace values, and numbers of generations with $m = 3$, $l = 3$

Table 5.4 The Data of Figure 5.5(a) and 5.5(b)

	Power $n' = 10$			Power $n' = 26$		
No. of Gens	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
1000	3.1428	5.8952	53.31	1.6304	3.8432	42.42
1500	3.4209	5.8952	58.03	1.8348	3.8432	47.74
2000	3.6386	5.8952	61.72	1.9571	3.8432	50.92
2500	3.7500	5.8952	63.61	1.9859	3.8432	51.67
3000	3.8105	5.8952	64.64	2.0407	3.8432	53.10
3500	3.8867	5.8952	65.93	2.0509	3.8432	53.36
4000	3.8913	5.8952	66.01	2.0675	3.8432	53.80
4500	3.9689	5.8952	67.32	2.1166	3.8432	55.07
5000	4.0181	5.8952	68.15	2.1479	3.8432	55.89
5500	4.0554	5.8952	68.79	2.1935	3.8432	57.08
6000	4.0788	5.8952	69.19	2.1977	3.8432	57.18
6500	4.1034	5.8952	69.61	2.2439	3.8432	58.39
7000	4.1206	5.8952	69.90	2.2226	3.8432	57.83
7500	4.1298	5.8952	70.05	2.2292	3.8432	58.00
8000	4.1457	5.8952	70.32	2.2528	3.8432	58.62
8500	4.1307	5.8952	70.07	2.2546	3.8432	58.66
9000	4.1564	5.8952	70.50	2.2398	3.8432	58.28
9500	4.1780	5.8952	70.87	2.2620	3.8432	58.86
10000	4.1642	5.8952	70.64	2.2591	3.8432	58.78

Table 5.4 shows that when a state space has $2^9 = 512$ states and the (fixed) number of generations is set to 2000, more than 50% of $Trace((CMS)^{n'})$ can be estimated if n' is between 10 and 26. In addition, small n' (e.g., $n' = 10$) has better estimated precision than large n' (e.g., $n' = 26$). However, the power n' cannot be set too small since with small n', relatively small eigenvalues are still non-zero, which causes some noise if the purpose is to predict a degree

of convergence. Moreover, $Trace((CMS)^{n'})$ with too small n' has large number of non-zero diagonal elements, which makes the estimation difficult for the proposed metric because the sample size is relatively small with respect to the number of states in the state space.

5.3.3 Discussions on the Precision of the Estimation

Now, let us compare the estimated $Trace((CMS)^{n'})$ of fitness functions and their actual values. In order to accomplish this comparison, I have run the experiments on a High-Performance-Computer. The maximal order of the transition matrices (square matrices) estimated for the actual trace values is 2^{10}. In this case, we can either set $(m, l) = (2, 5)$, or $(m, l) = (5, 2)$. Both of them draw the same conclusion. Table 5.5 presents the estimated and actual values of $Trace((CMS)^{n'})$ for $(m, l) = (2, 5)$ with the (fixed) number of generations being set to 10000. Other settings with $ml \leq 10$ have similar results.

Table 5.5 Comparison with Actual Trace Values

	Fitness f_2			Fitness f_4		
The Power n'	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
10	11.2222	19.8676	56.48	10.9152	19.8817	54.90
12	9.6180	18.1384	53.03	9.0834	18.1525	50.04
14	8.2574	16.6158	49.70	7.7609	16.6331	46.66
16	7.1721	15.2567	47.01	6.7049	15.2781	43.89
18	6.2690	14.0383	44.66	5.8291	14.0643	41.45
20	5.4732	12.9436	42.29	5.0959	12.9746	39.28
22	4.9405	11.9583	41.31	4.5871	11.9946	38.24
24	4.3463	11.0701	39.26	4.0685	11.1118	36.61
26	3.9340	10.2680	38.31	3.6454	10.3154	35.34
	Fitness f_5			Fitness f_8		
The Power n'	Estimated Value	Actual Value	%	Estimated Value	Actual Value	%
10	11.7743	19.9793	0.5893	11.6067	19.9302	0.5824
12	10.0742	18.2940	0.5507	9.8011	18.1948	0.5387
14	8.7976	16.8133	0.5233	8.3236	16.6830	0.4989
16	7.6219	15.4948	0.4919	7.2105	15.3389	0.4701
18	6.7025	14.3155	0.4682	6.4481	14.1368	0.4561
20	5.3872	13.2577	0.4063	5.7171	13.0590	0.4378
22	5.1549	12.3069	0.4189	5.0500	12.0908	0.4177
24	4.8047	11.4504	0.4196	4.5648	11.2197	0.4069
26	4.4481	10.6772	0.4166	4.1386	10.4346	0.3966

In Table 5.5, if we compare the estimated $Trace((CMS)^{n'})$ with n' being fixed, we get that all of the estimated $Trace((CMS)^{n'})$ have close percentages of accuracy, except f_4. The estimated $Trace((CMS)^{n'})$ of f_4 drops a little because the landscape of the fitness function f_4 is almost flat. The fitness value is not sensitive with respect to the modifications of the value x. With this type of fitness function, proportional selection cannot discriminate "good" solutions from "bad" ones. Therefore, during a GA run, most states have non-zero probabilities

remaining the same in the next generation. In other words, this type of fitness function results in a transition matrix which has a large number of non-zero diagonal entries. This feature will cause problems for the proposed methodology since there may not be enough samples to reach all of the states and accurately estimate $Trace((CMS)^{n'})$, $n' \in N$.

Comparing the percentages of accuracy among powers $n' = 10$, $n' = 12$, ..., and $n' = 26$, the same conclusion in the previous subsection is drawn that small n' has better estimated precision than large n'. With this feature, we can be sure that if a fitness function, which is not flat, has larger trace value for a certain n', the proposed evaluation metric will not underestimate its $Trace((CMS)^{n'})$ compared to the estimated traces of other fitness functions.

However, a concern may be raised that the discussion and observation above are based on the condition that the number of samples is greater than the number of states. The relation between estimated $Trace((CMS)^{n'})$ with a fixed n' and a number of generations n with n smaller than a predefined number of states is not specified. The empirical investigation shows that the estimated $Trace((CMS)^{n'})$ and the number of generations n are linearly related if n is less than the number of states. Figure 5.6 illustrates their relationship with respect to f_2 and f_8 when the setting is $(n', m, l) = (10, 2, 16)$. That means, the estimated traces with respect to fitness functions should have close percentages of accuracy with n smaller than the number of states, if they have close percentages of accuracy with n larger than or equal to the number of states.

5.3.4 Discussions on the Convergence Trend

Another feature in the data shown in Table 5.3 is that with the number of generations being fixed, the estimated $Trace((CMS)^{n'})$ decreases as n' increases. Therefore, the convergence trend of the estimated $Trace((CMS)^{n'})$ remains the same as the convergence trend of the theoretical value of $Trace((CMS)^{n'})$. This feature can be applied to evaluate the degrees of convergence asserted earlier in this dissertation.

To analyze the convergence trends of the estimated $Trace((CMS)^{n'})$ with respect to the fitness functions, the number of individuals within a generation m is set to 2, the length of

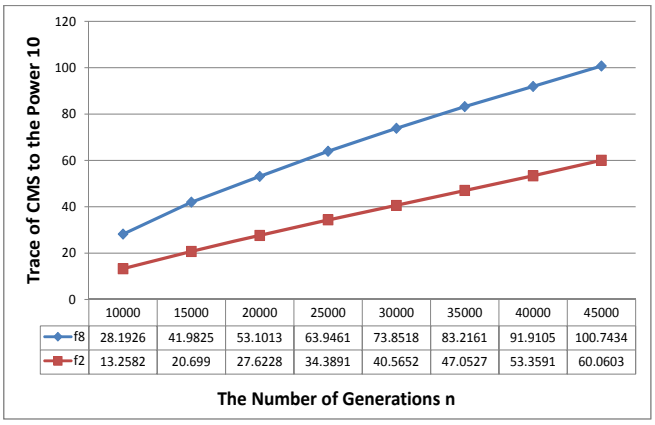


Figure 5.6 The relationship between estimated traces and numbers of generations

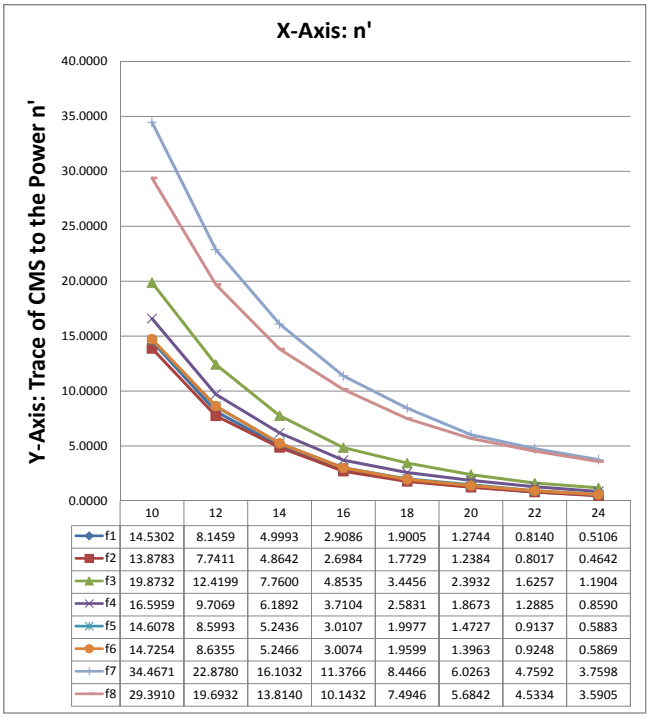
an individual (a feasible solution) l is set to 16, the number of generations is fixed to 10000, and the power n' of $(CMS)^{n'}$ is set from 10 to 24, respectively. The derived result, shown in Figure 5.7, is the average from 40 trials of GA runs.

Figure 5.7 shows the relationship between n' and the estimated $Trace((CMS)^{n'})$, and demonstrates that no matter what type the fitness function, as n' increases, the estimated $Trace((CMS)^{n'})$ decreases. All of the estimated values of $Trace((CMS)^{n'})$ approach to a value between 0 and 1 with increasing n'. Moreover, the fitness functions are clustered based on their difficulties for GAs. Fitness functions f_1 to f_6 can be considered to be in the same cluster, and f_7 to f_8 can be considered to be in another one. This phenomenon shows that the proposed evaluation metric has the ability to distinguish the degrees of difficulty of GAs.

5.3.5 Discussions on the Confidence Intervals

Further investigation on the relationship between n' and the estimated value $Trace((CMS)^{n'})$ is conducted. The 95% standard confidence intervals (SCIs) are computed for all of the fitness functions f_1 to f_8 (see Figure 5.8 to Figure 5.15). The settings remain the same as in the previous discussion on the convergence trend.

The SCI tends to shrink as n' increases. Among f_1 to f_8, f_7 has the largest confidence



	10	12	14	16	18	20	22	24
f1	14.5302	8.1459	4.9993	2.9086	1.9005	1.2744	0.8140	0.5106
f2	13.8783	7.7411	4.8642	2.6984	1.7729	1.2384	0.8017	0.4642
f3	19.8732	12.4199	7.7600	4.8535	3.4456	2.3932	1.6257	1.1904
f4	16.5959	9.7069	6.1892	3.7104	2.5831	1.8673	1.2885	0.8590
f5	14.6078	8.5993	5.2436	3.0107	1.9977	1.4727	0.9137	0.5883
f6	14.7254	8.6355	5.2466	3.0074	1.9599	1.3963	0.9248	0.5869
f7	34.4671	22.8780	16.1032	11.3766	8.4466	6.0263	4.7592	3.7598
f8	29.3910	19.6932	13.8140	10.1432	7.4946	5.6842	4.5334	3.5905

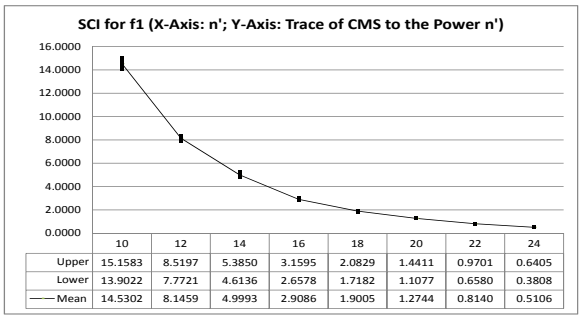
Figure 5.7 The convergence trends of $Trace((CMS)^{n'})$

intervals for $n' = 10$ and $n' = 24$, whose lengths are around 2.02 and 0.60 (less than 1), respectively. Combined with the conclusion made in the previous discussion, it can be confirmed that the clustering of fitness functions with respect to the estimated values of $Trace((CMS)^{n'})$ is valid and has a strong discrimination since the confidence intervals of $Trace((CMS)^{n'})$ with any n' between 10 to 24 for f_7 and f_8 disjoint from the confidence interval of $Trace((CMS)^{n'})$ for any fitness function from f_1 to f_6. In reality, f_7 and f_8 are considered the most difficult fitness functions among f_1 to f_8 for GAs.

In the literature, researchers who investigate confidence intervals have asserted that the actual coverage probability of the SCI may sometimes not be equal to the nominal level claimed, especially when the sample is small [Brown, L. D. and Cai, T. T. and DasGupta, A. (2001)]. Several alternative intervals are recommended. The Wilson interval (WI) is one of them and used to compare with the SCI here (see Figure 5.16 to Figure 5.23). As we can see, the WI also tends to shrink as n' increases. Most of the mean values and the lengths of WIs are slightly smaller than those of SCIs. Figure 5.24 shows the mean values of WI and SCI for the fitness functions f_1 and f_8, respectively. Carefully comparing the WIs and SCIs for the eight fitness functions, we can get that when $n' \geq 16$, the two intervals overlap each other. No matter which interval (WI or SCI) is used for the comparison, the clusters formed earlier are still valid.

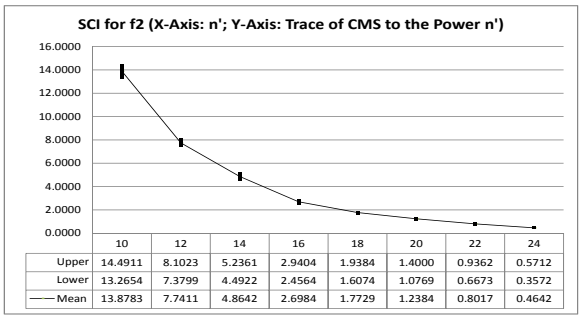
5.3.6 Discussions on the Smallest Number of Samples Needed for the Estimation

Through above discussions, we know that the proposed evaluation metric has the ability to determine the order of fitness functions sorted by the difficulty levels for GAs. This subsection simply discusses the smallest number of samples needed for the estimation. Similar as before, the fitness functions, except f_4, with the setting $(n', m, l) = (10, 2, 16)$ are investigated. Assume that for the same n', using a larger number of samples derives the estimated $Trace((CMS)^{n'})$ closer to the actual trace value. The order of fitness functions when the number of generations $n = 20000$ is compared with smaller numbers of generations. When $n = 20000$, the order of the fitness functions from the most difficult one (i.e., the one has the largest $Trace((CMS)^{10})$)



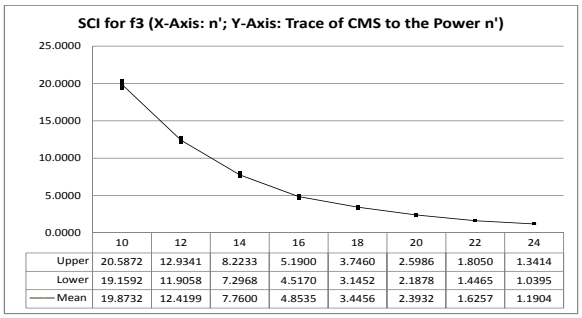
	10	12	14	16	18	20	22	24
Upper	15.1583	8.5197	5.3850	3.1595	2.0829	1.4411	0.9701	0.6405
Lower	13.9022	7.7721	4.6136	2.6578	1.7182	1.1077	0.6580	0.3808
Mean	14.5302	8.1459	4.9993	2.9086	1.9005	1.2744	0.8140	0.5106

Figure 5.8 95% SCI of $Trace((CMS)^{n'})$



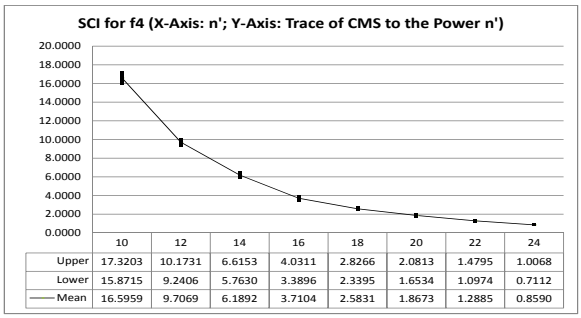
	10	12	14	16	18	20	22	24
Upper	14.4911	8.1023	5.2361	2.9404	1.9384	1.4000	0.9362	0.5712
Lower	13.2654	7.3799	4.4922	2.4564	1.6074	1.0769	0.6673	0.3572
Mean	13.8783	7.7411	4.8642	2.6984	1.7729	1.2384	0.8017	0.4642

Figure 5.9 95% SCI of $Trace((CMS)^{n'})$



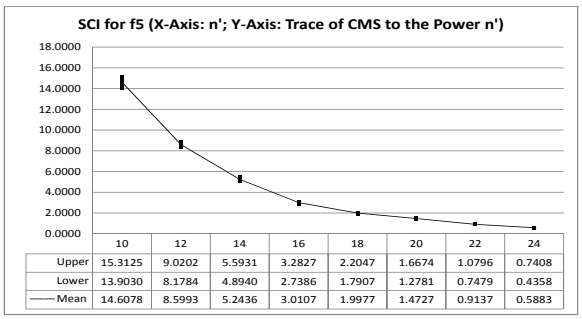
	10	12	14	16	18	20	22	24
Upper	20.5872	12.9341	8.2233	5.1900	3.7460	2.5986	1.8050	1.3414
Lower	19.1592	11.9058	7.2968	4.5170	3.1452	2.1878	1.4465	1.0395
Mean	19.8732	12.4199	7.7600	4.8535	3.4456	2.3932	1.6257	1.1904

Figure 5.10 95% SCI of $Trace((CMS)^{n'})$



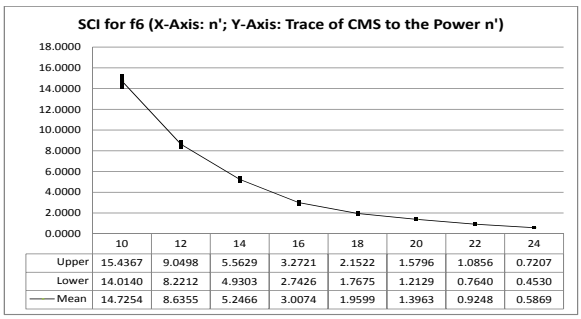
	10	12	14	16	18	20	22	24
Upper	17.3203	10.1731	6.6153	4.0311	2.8266	2.0813	1.4795	1.0068
Lower	15.8715	9.2406	5.7630	3.3896	2.3395	1.6534	1.0974	0.7112
Mean	16.5959	9.7069	6.1892	3.7104	2.5831	1.8673	1.2885	0.8590

Figure 5.11 95% SCI of $Trace((CMS)^{n'})$



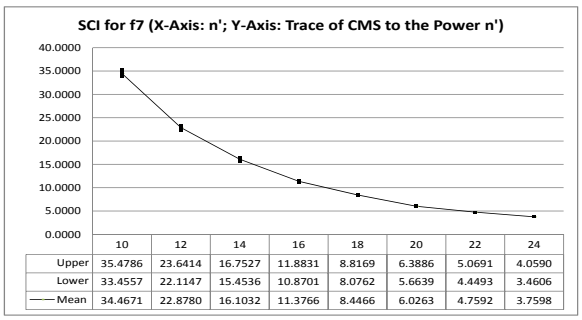
	10	12	14	16	18	20	22	24
Upper	15.3125	9.0202	5.5931	3.2827	2.2047	1.6674	1.0796	0.7408
Lower	13.9030	8.1784	4.8940	2.7386	1.7907	1.2781	0.7479	0.4358
Mean	14.6078	8.5993	5.2436	3.0107	1.9977	1.4727	0.9137	0.5883

Figure 5.12 95% SCI of $Trace((CMS)^{n'})$



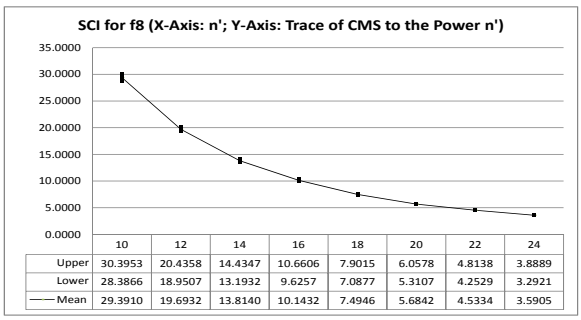
	10	12	14	16	18	20	22	24
Upper	15.4367	9.0498	5.5629	3.2721	2.1522	1.5796	1.0856	0.7207
Lower	14.0140	8.2212	4.9303	2.7426	1.7675	1.2129	0.7640	0.4530
Mean	14.7254	8.6355	5.2466	3.0074	1.9599	1.3963	0.9248	0.5869

Figure 5.13 95% SCI of $Trace((CMS)^{n'})$



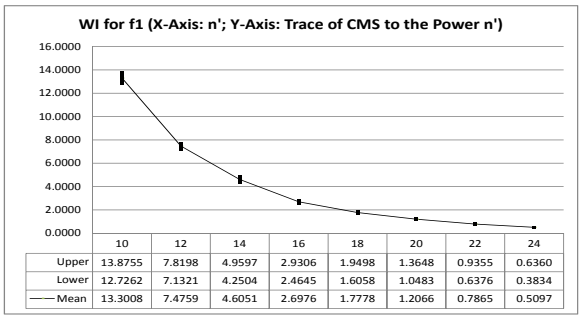
	10	12	14	16	18	20	22	24
Upper	35.4786	23.6414	16.7527	11.8831	8.8169	6.3886	5.0691	4.0590
Lower	33.4557	22.1147	15.4536	10.8701	8.0762	5.6639	4.4493	3.4606
Mean	34.4671	22.8780	16.1032	11.3766	8.4466	6.0263	4.7592	3.7598

Figure 5.14 95% SCI of $Trace((CMS)^{n'})$



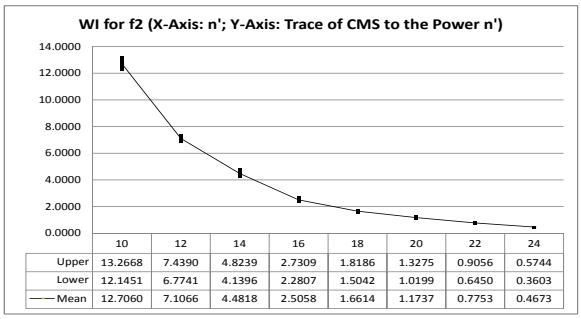
	10	12	14	16	18	20	22	24
Upper	30.3953	20.4358	14.4347	10.6606	7.9015	6.0578	4.8138	3.8889
Lower	28.3866	18.9507	13.1932	9.6257	7.0877	5.3107	4.2529	3.2921
Mean	29.3910	19.6932	13.8140	10.1432	7.4946	5.6842	4.5334	3.5905

Figure 5.15 95% SCI of $Trace((CMS)^{n'})$



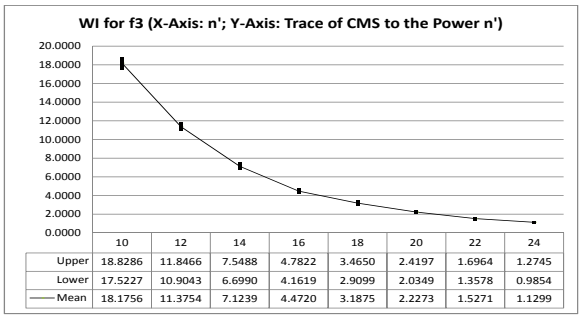
	10	12	14	16	18	20	22	24
Upper	13.8755	7.8198	4.9597	2.9306	1.9498	1.3648	0.9355	0.6360
Lower	12.7262	7.1321	4.2504	2.4645	1.6058	1.0483	0.6376	0.3834
Mean	13.3008	7.4759	4.6051	2.6976	1.7778	1.2066	0.7865	0.5097

Figure 5.16 95% WI of $Trace((CMS)^{n'})$



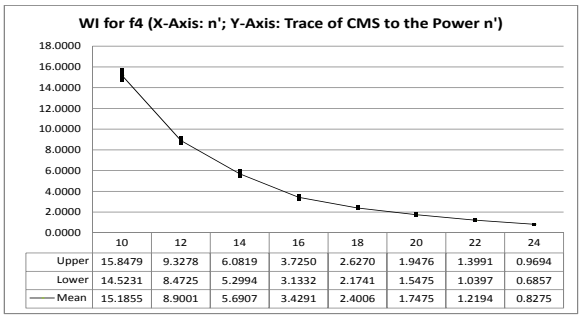
	10	12	14	16	18	20	22	24
Upper	13.2668	7.4390	4.8239	2.7309	1.8186	1.3275	0.9056	0.5744
Lower	12.1451	6.7741	4.1396	2.2807	1.5042	1.0199	0.6450	0.3603
Mean	12.7060	7.1066	4.4818	2.5058	1.6614	1.1737	0.7753	0.4673

Figure 5.17 95% WI of $Trace((CMS)^{n'})$



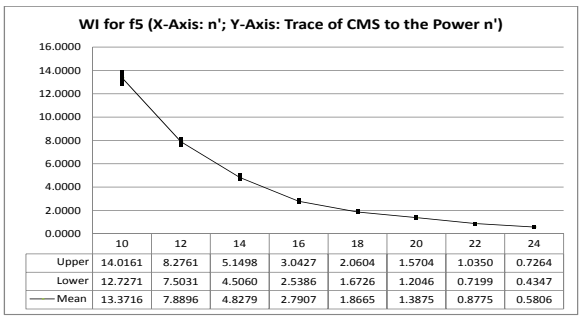
	10	12	14	16	18	20	22	24
Upper	18.8286	11.8466	7.5488	4.7822	3.4650	2.4197	1.6964	1.2745
Lower	17.5227	10.9043	6.6990	4.1619	2.9099	2.0349	1.3578	0.9854
Mean	18.1756	11.3754	7.1239	4.4720	3.1875	2.2273	1.5271	1.1299

Figure 5.18 95% WI of $Trace((CMS)^{n'})$



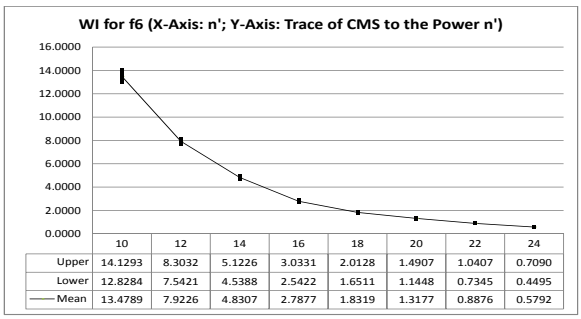
	10	12	14	16	18	20	22	24
Upper	15.8479	9.3278	6.0819	3.7250	2.6270	1.9476	1.3991	0.9694
Lower	14.5231	8.4725	5.2994	3.1332	2.1741	1.5475	1.0397	0.6857
Mean	15.1855	8.9001	5.6907	3.4291	2.4006	1.7475	1.2194	0.8275

Figure 5.19 95% WI of $Trace((CMS)^{n'})$



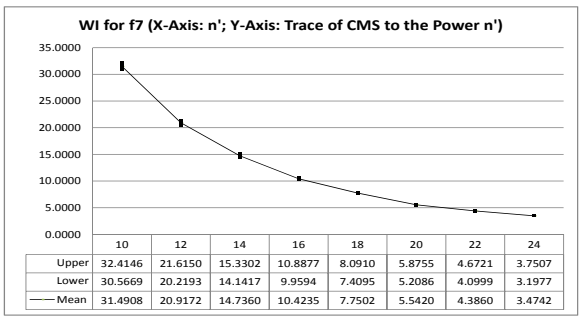
	10	12	14	16	18	20	22	24
Upper	14.0161	8.2761	5.1498	3.0427	2.0604	1.5704	1.0350	0.7264
Lower	12.7271	7.5031	4.5060	2.5386	1.6726	1.2046	0.7199	0.4347
Mean	13.3716	7.8896	4.8279	2.7907	1.8665	1.3875	0.8775	0.5806

Figure 5.20 95% WI of $Trace((CMS)^{n'})$



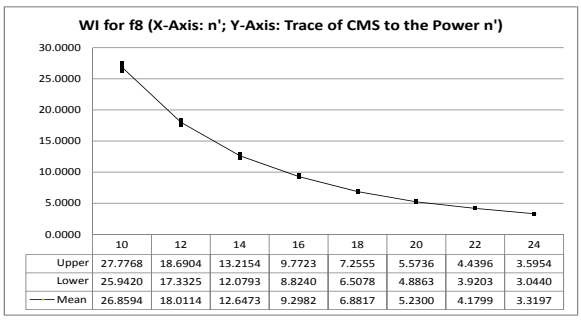
	10	12	14	16	18	20	22	24
Upper	14.1293	8.3032	5.1226	3.0331	2.0128	1.4907	1.0407	0.7090
Lower	12.8284	7.5421	4.5388	2.5422	1.6511	1.1448	0.7345	0.4495
Mean	13.4789	7.9226	4.8307	2.7877	1.8319	1.3177	0.8876	0.5792

Figure 5.21 95% WI of $Trace((CMS)^{n'})$



	10	12	14	16	18	20	22	24
Upper	32.4146	21.6150	15.3302	10.8877	8.0910	5.8755	4.6721	3.7507
Lower	30.5669	20.2193	14.1417	9.9594	7.4095	5.2086	4.0999	3.1977
Mean	31.4908	20.9172	14.7360	10.4235	7.7502	5.5420	4.3860	3.4742

Figure 5.22 95% WI of $Trace((CMS)^{n'})$



	10	12	14	16	18	20	22	24
Upper	27.7768	18.6904	13.2154	9.7723	7.2555	5.5736	4.4396	3.5954
Lower	25.9420	17.3325	12.0793	8.8240	6.5078	4.8863	3.9203	3.0440
Mean	26.8594	18.0114	12.6473	9.2982	6.8817	5.2300	4.1799	3.3197

Figure 5.23 95% WI of $Trace((CMS)^{n'})$

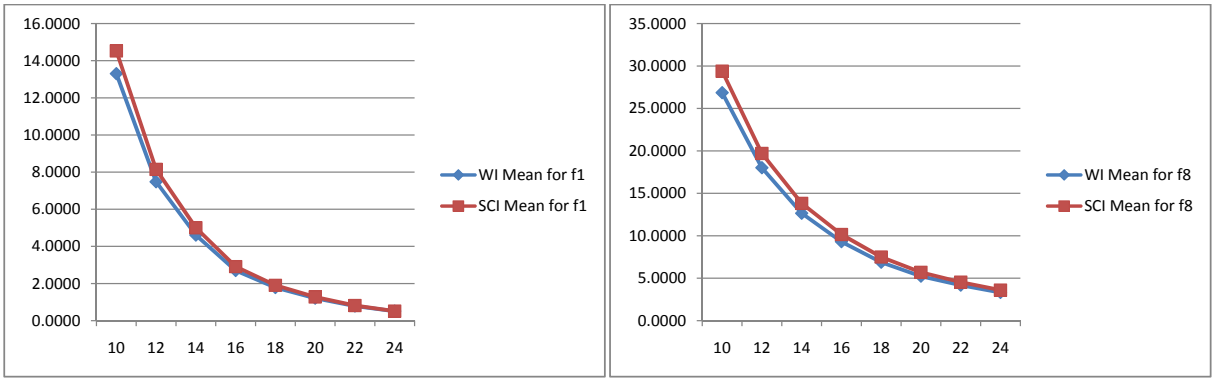


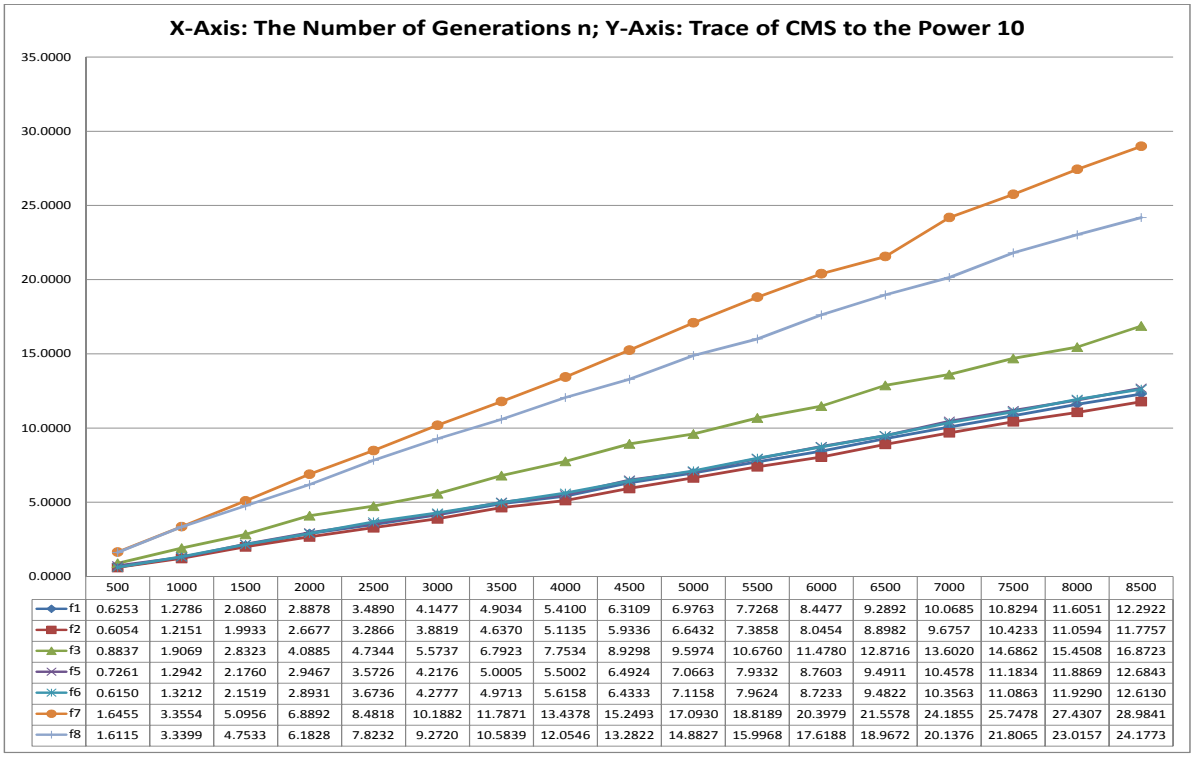
Figure 5.24 The means of WI and SCI for fitness functions f_1 and f_8, respectively

to the easiest one is

$$f_7 \to f_8 \to f_3 \to f_6 \to f_5 \to f_1 \to f_2$$

Note that when $n = 10000$, the order of fitness functions is exactly the same as the above order. Figure 5.25 shows the results obtained from using smaller numbers of generations.

As we can see, when the number of generations increases, the estimated traces increase linearly. This result is similar as the result in Figure 5.6. From $n = 500$ to $n = 8500$, the two lines representing the trace values for f_5 and f_6 are tangled. This is also the case when n is around 20000. When $n = 500$, the traces for f_7 and f_8 already have larger values than the traces for other fitness functions; however, the difference of the traces for any two fitness functions is very small, which may cause problems. For instance, we cannot know whether the fitness function which has a smaller trace is actually the easier one or not. When $n = 1000$, we get the exact same order of fitness functions presented above. This shows that the smallest number of samples needed for the setting $(n', m, l) = (10, 2, 16)$ is around 1000. Compared with the number of states (2^{32}) and the number of feasible solutions in the solution space (2^{16}), it is a small number.



	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000	8500
f1	0.6253	1.2786	2.0860	2.8878	3.4890	4.1477	4.9034	5.4100	6.3109	6.9763	7.7268	8.4477	9.2892	10.0685	10.8294	11.6051	12.2922
f2	0.6054	1.2151	1.9933	2.6677	3.2866	3.8819	4.6370	5.1135	5.9336	6.6432	7.3858	8.0454	8.8982	9.6757	10.4233	11.0594	11.7757
f3	0.8837	1.9069	2.8323	4.0885	4.7344	5.5737	6.7923	7.7534	8.9298	9.5974	10.6760	11.4780	12.8716	13.6020	14.6862	15.4508	16.8723
f5	0.7261	1.2942	2.1760	2.9467	3.5726	4.2176	5.0005	5.5002	6.4924	7.0663	7.9332	8.7603	9.4911	10.4578	11.1834	11.8869	12.6843
f6	0.6150	1.3212	2.1519	2.8931	3.6736	4.2777	4.9713	5.6158	6.4333	7.1158	7.9624	8.7233	9.4822	10.3563	11.0863	11.9290	12.6130
f7	1.6455	3.3554	5.0956	6.8892	8.4818	10.1882	11.7871	13.4378	15.2493	17.0930	18.8189	20.3979	21.5578	24.1855	25.7478	27.4307	28.9841
f8	1.6115	3.3399	4.7533	6.1828	7.8232	9.2720	10.5839	12.0546	13.2822	14.8827	15.9968	17.6188	18.9672	20.1376	21.8065	23.0157	24.1773

Figure 5.25 The order of fitness functions w.r.t. smaller numbers of generations

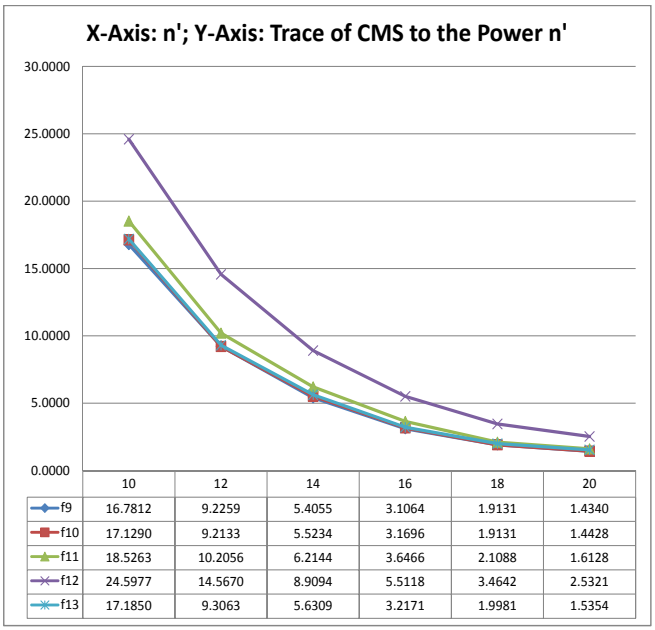
5.3.7 Multi-Dimensional Fitness Functions

In order to present more evidence for verifying the proposed methodology, five 3-dimensional fitness functions are selected, including Ackley's, Rastrigin's, and Sphere functions, with different degrees of difficulty for GAs. As it is commonly known, Ackley's, Rastrigin's, and Sphere functions are designed to be minimized. The selection technique in this dissertation, roulette wheel selection, enables a GA to maximize fitness functions. Hence, those fitness functions are slightly adjusted in this experiment. The five fitness functions investigated are in Table 5.6. Intuitively, the fitness functions f_9 and f_{10} are relatively simple among the five, since both of

Table 5.6 The Selected 3-Dimensional Fitness Functions and Their Features

Fitness Function	Optimal Value
$f_9(\mathbf{x}) = (\sum_{i=1}^{3} x_i) + 16$, where $x_1, x_2, x_3 \in [-5, 5]$	31
(Sphere Function - Unimodal and Separable) $f_{10}(\mathbf{x}) = -(\sum_{i=1}^{3} x_i^2) + 76$, where $x_1, x_2, x_3 \in [-5, 5]$	76
(Ackley's Function 1 - Multimodal, and Non-separable) $f_{11}(\mathbf{x}) = -(-20e^{-0.2\sqrt{\frac{1}{3}\sum_{i=1}^{3} x_i^2}} - e^{\frac{1}{3}\sum_{i=1}^{3} cos(2\pi x_i)} + 20 + e) + 15$, where $x_1, x_2, x_3 \in [-5, 5]$	15
(Ackley's Function 2 - Multimodal, and Non-separable) $f_{12}(\mathbf{x}) = -(-20e^{-0.2\sqrt{\frac{1}{3}\sum_{i=1}^{3} x_i^2}} - e^{\frac{1}{3}\sum_{i=1}^{3} cos(2\pi x_i)} + 20 + e) + 23$, where $x_1, x_2, x_3 \in [-32.768, 32.768]$	23
(Rastrigin's Function - Multimodal, and Separable) $f_{13}(\mathbf{x}) = -(\sum_{i=1}^{3} (x_i^2 - 10cos(2\pi x_i) + 10)) + 135$, where $x_1, x_2, x_3 \in [-5, 5]$	135

them only have one optimal solution. Comparing the features of the rest of the fitness functions, Ackley's function is more difficult for a CGA than Rastrigin's function, since Ackley's function is not only multimodal, but also non-separable (i.e., it cannot be rewritten as a sum of several functions of just one variable) [Hadley, G. (1964)]. Separable functions can be optimized for each variable in turn. Non-separable functions are more difficult to optimize as the accurate search direction depends on two or more genes. The result of this experiment, shown in Figure 5.26, is the average from 40 trials of GA runs. The confidence intervals with respect to the settings of n' are similar to the result derived by one dimensional fitness functions. From Figure 5.26, the estimated trace derived by f_{12} is larger than that of f_{11} since the range $[-32.768, 32.768]$ includes more peaks than the range $[-5, 5]$. The degree of difficulty of f_{12} is higher than that of f_{11} for a CGA. Moreover, when fitness functions are multi-dimensional,



	10	12	14	16	18	20
f9	16.7812	9.2259	5.4055	3.1064	1.9131	1.4340
f10	17.1290	9.2133	5.5234	3.1696	1.9131	1.4428
f11	18.5263	10.2056	6.2144	3.6466	2.1088	1.6128
f12	24.5977	14.5670	8.9094	5.5118	3.4642	2.5321
f13	17.1850	9.3063	5.6309	3.2171	1.9981	1.5354

Figure 5.26 The convergence trends of $Trace((CMS)^{n'})$ for multi-dimensional fitness functions

the traces estimated by the proposed evaluation metric still coincide with the earlier assertion that more difficult fitness functions, with respect to a CGA, result in higher estimated traces. The proposed evaluation metric has the ability to determine the best fitness function for a CGA among a set of fitness functions.

5.3.7.1 Observations on the Experiments

Due to the fact that usually the sample size is relatively small compared to the state space size, the estimated $Trace((CMS)^{n'})$ is only a small percentage of the actual $Trace((CMS)^{n'})$. The criterion (i.e., |Obtained Value $-$ 1|) discussed earlier for determining a degree of convergence should be slightly adjusted. Since the actual $Trace((CMS)^{n'})$ is greater than or equal to 0 for all $n \in N$, any percentage of $Trace((CMS)^{n'})$ should be greater than or equal to 0. The criterion can be modified to |Obtained Value $-$ 0| (i.e., Obtained Value) to reflect that the smaller the estimated $Trace((CMS)^{n'})$, the higher the degree of convergence.

From the experiments, we get that different settings of m and l may cause different orders

of fitness functions sorted by difficulties to a certain GA configuration. The reason is that the number of points encoded in a search space for discovering the landscape of a fitness function is based on l. Small l implies a small number of points encoded in a search space. In that case, a non-smooth fitness function may become a smooth one due to the lack of points. The setting of m impacts probabilities of selecting individuals. Another observation is that different domains of a fitness function encoded in a search space have a high possibility to result in different degrees of difficulties for a GA, which is because more of the landscape of the fitness function, such as peaks, and valleys, is included in a larger domain.

In addition, before running a GA with the evaluation metric, one has to check:

1. Sensitivity of the fitness function;
2. Fitness values of the encoding $0000\ldots0$ and $1111\ldots1$;
3. If the isolation points are known, the fitness values of those points should be computed in advance.

Checking the sensitivity of the fitness function helps researchers determine whether or not the defined fitness function has a low discrimination for "good" and "bad" solutions. In real-world applications, GAs with non-sensitive fitness functions are rare. Practitioners apply GAs to obtain (near) optimal solutions. If the fitness function is almost flat for a GA, it is meaningless to spend time searching for the global optimal solution. Besides, the low discrimination fitness function can always be reformulated to improve the performance of GAs. Through the second step, the isolation points of the fitness function in the extreme locations can be known beforehand. This step is essential because the isolation points other than $0000\ldots0$ and $1111\ldots1$ are relatively easy to produce by chance through crossover and mutation operators. Checking the extreme cases first helps GAs to have a better chance to achieve the global optimal solution.

CHAPTER 6. A Case Study - Evolutionary Testing

6.1 Overview of Evolutionary Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding [Pressman, R. S. (2005)]. Due to its importance, at least 50% of the total cost (including human effort) involved in a software development project is typically consumed by testing [Beizer, B. (1990); Tracey, N. and Clark, J. and Mander, K. (1998)]. Although a lot of effort has been allocated to the task of testing, complete testing is usually impossible in practice because of the vast amount of possible input situations [Wegener, J. (2005)]. For the same reason, effective methods for automatic test data generation are in great demand. Many techniques focusing on that purpose have been proposed and developed. Evolutionary Testing (ET) is one of them and it gradually forms an important branch of the research area in automatic test data generation.

ET exploits evolutionary algorithms (EAs), such as GA, to generate test data. According to the objectives of testing [Pressman, R. S. (2005)], it is believed that ET has its potential to become a major technique for automatically generating test data since it possesses the following features:

- Like Random Testing (RT), it has the ability to search feasible solutions in the search space. However, unlike RT, it can guide solutions to the desired ones through an evaluation criterion. With guidance, ET is capable of producing effective solutions even for complex and poorly understood search spaces with many dimensions. Without it, ET behaves exactly the same as RT. Hence, ET is more powerful than RT.

- ET is able to recombine the input data based on the higher fitness values. That is, it has

the capability to adjust the input data from the testing history, and create the desired ones.

A better encoding of EAs is one that can explore more possible solutions within a fixed time duration. In other words, a good design for ET can cover lots of combinations of inputs and discover errors with a high probability.

Among all the software testing activities (i.e., test case design, test execution, monitoring, test evaluation, test planning, test organization, and test documentation), test case design is essential. Systematic generation of test cases based on the test design is indispensable to the quality of software. However, for most of the test objectives, it is difficult to automate the generation of test cases due to the reasons listed below [Sthamer, H. and Wegener, J. and Baresel, A. (2002)]:

- The generation of test cases for functional testing is usually impossible because in general no formal specifications can be applied in industrial practice.
- Structural testing is difficult to automate due to the limits of symbolic execution.
- No specialized methods and tools exist for testing the temporal behavior of systems.
- A generation of test cases for testing safety constraints is generally impossible.

Hence, test cases have to be defined manually, which is not efficient. Therefore, the researchers have sought other techniques, e.g., ET, to effectively solve these problems.

According to the literature review, ET has been applied for functional testing (black box testing), structural testing (white box testing), and real-time testing [Last, M. and Eyal1, S. and Kandel, A. (2006); Michael, C. C. and McGraw, G. and Schatz, M. A. (2001); Baresel, A. and Pohlheim, H. and Sadeghipour, S. (2003); Baresel, A. and Binkley, D. and Harman, M. and Korel, B. (2004); McMinn, P. (2004); Wegener, J. (2005)] to automatically generate test data. Among all of the existing work, structural testing is the method that has been investigated and developed the most.

In structural testing, various methods to analyze the coverage of program structures have been proposed. Depending on their test aims, fitness functions of ET are formulated. The fitness functions summarized below are commonly seen in the literature.

The test data generation problem for structural testing is to find a set of program inputs that achieves the desired coverage [Tracey, N. and Clark, J. and Mander, K. (1998)]. This type of problems has been converted to optimization problems since the test criteria are specifically known. Most of the work defines the fitness function based on the test criterions. The methods to construct fitness functions can be categorized in Figure 6.1 [McMinn, P. (2004)].

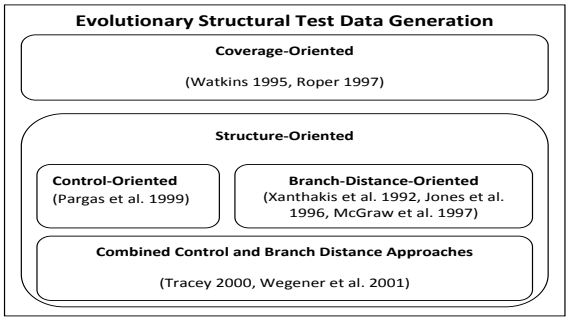


Figure 6.1 Classification of Dynamic Structural Test Data Generation Techniques Using EAs [McMinn, P. (2004)]

From Figure 6.1, we can see that researchers started to apply EAs for software testing around two decades ago. They first set their goal in terms of coverage. For instance, the fitness functions are designed to penalize the individuals that follow already covered paths. Later, they found that it is difficult for the coverage-oriented approach to discover some specific paths. It gradually changed to structure-oriented approach. In this approach, researchers employed different information to form the fitness functions. The most updated one has combined control and branch distance together. The branch distance is usually normalized to the range $[0, 1)$ [McMinn, P. (2004); McMinn, P. and Binkley, D. and Harman, M. (2005)]. With it (fractional part), one may evaluate two feasible solutions with the same fitness value on the approach level (integer part) [Levin, S. and Yehudai, A. (2007)]. The basic form of that to be minimized is

$$Fitness = Approach\_Level + Branch\_Distance, \tag{6.1}$$

where the *Approach_Level* is the number of target condition statements which are not executed by the given test data, and *Branch_Distance* is the distance from the test data to the input values of the target branch, and it can be computed in various ways. The formula to calculate both depend on the test goal, i.e., which types of coverage are needed [McMinn, P. and Holcombe, M. (2003); McMinn, P. (2004); Wegener, J. (2005)].

6.2 The Flag Problem

Many works have applied EA for structural testing using fitness functions to guide the feasible solutions to form sets of test data for the desired branches. However, problems exist that inhibit the search and have not been perfectly solved yet. The flag problem is one of them. Typically, it occurs when the source code has flag variables. A flag variable is any variable that takes on one of two or more (finite) discrete values [Baresel, A. and Binkley, D. and Harman, M. and Korel, B. (2004)]. For instance, boolean variables are flag variables. In the following, an example is given to show how the flag variable influences the performance of the search.

Suppose Figure 6.2 is part of the tested source code.

1. flag = false;

2. ... /*Nothing related to 'flag'*/

3. if (switch == 5) flag = true;

4. ... /*Nothing related to 'flag'*/

5. if (flag) /* Test aim */

Figure 6.2 An Example: The Source Code

In Figure 6.2, there is an boolean variable called $flag$. The test aim is in line 5, in which $flag$ should be equal to 1 (TRUE). In that case, part of the fitness function related to the $flag$ variable should be of some formula like $|flag - 1|$, i.e., the absolute value of $(flag - 1)$. Now,

given two or more sets of input test data automatically generated by EA, if all of them do not go to the branch in which the variable *switch* is equal to 5 before line 3, all of the fitness values related to the flag variable are equal to $|0-1|$. That is, the fitness values related to the flag variable are the same for all sets of test data with $flag$ not being TRUE in line 5. Hence, the fitness function does not provide any guidance for reaching the test aim as the flag can be always FALSE and cannot become TRUE. The search degenerates to random search.

6.3 Discussion

The reason of selecting ET as a case study is that

- There is a difficult flag problem in the context of ET so that the difficult problems can be compared with easier problems (i.e., the problems dealing with testing of flag-free programs);
- The fitness function of ET (Equation 6.1) has two types of guidances — *Approach_Level* and *Branch_Distance*. In other words, improper fitness functions (i.e., the fitness functions with two different guidances) can be formulated to verify the proposed evaluation metric;
- From the final fitness value, we can always get information about the quality of solutions. The *Approach_Level* (i.e., integer part) represents which branch the generated test case is located in.

6.4 Experiment Settings

The flag problem is a difficult problem for ET. In order to demonstrate that the proposed evaluation metric has the ability to discern the applicability of GAs for real-world optimization problems, two types of programs are designed (see Figure 6.3 and Figure 6.4). Three fitness functions are formulated for the programs (see Table 6.5). CGA with best solution maintained over time is applied to generate test cases for those types of programs with respect to the three fitness functions.

```
Input (a, b, c, d)
1.    if (a==18.75)
2.    {
3.        if (b==37.5)
4.        {
5.            if (c==56.25)
6.            {
7.                if (d==75)
8.                {
9.                    Test Aim
10.               }
11.           }
12.       }
13.   }
```

Figure 6.3 Program with Flags

```
Input (a, b, c, d)
1.    if (a>=1)
2.    {
3.        if (b>=2)
4.        {
5.            if (c>=3)
6.            {
7.                if (d>=4)
8.                {
9.                    Test Aim
10.               }
11.           }
12.       }
13.   }
```

Figure 6.4 Program without Flags

The first type of the program (called P_1) has flags which are discrete values. For this type of program, the fitness function formulated using Equation 6.1 has isolation points. The second type of the program (called P_2) is a flag-free program.

Based on the description in Section 6.3, two fitness functions are formulated for P_1. f_1 is an improper fitness function since the *Approach_Level* and *Branch_Distance* guide the solutions to two different points in the search space. f_2 and f_3 are typical fitness functions for P_1 and P_2, respectively (see Table 6.5). The *Approach_Level* in Table 6.5 is in Figure 6.5.

The CGA with best solution maintained is applied to generate test cases corresponding to P_1 and P_2 with their fitness functions, respectively. With the setting $n = 10000$, $l = 16$, and $m = 2$, the result in Section 6.5 is derived.

6.5 The Result of the Case Study

Table 6.2 presents the average values of $Trace(CMS)^{n'}$ derived from the proposed evaluation metric from 40 trials. As we can see, all of them find the corresponding test cases for test aims. Since f_1 leads the solutions to two different points in the search space, the corresponding

Program	Fitness Function
P_1	$f_1 = 5 - (Approach\_Level + \frac{1}{1101}(\lvert a - 18.75\rvert + \lvert b - 9.375\rvert + \lvert c - 6.25\rvert + \lvert d - 4.6875\rvert))$
P_1	$f_2 = 5 - (Approach\_Level + \frac{1}{1101}(\lvert a - 18.75\rvert + \lvert b - 37.5\rvert + \lvert c - 56.25\rvert + \lvert d - 75\rvert))$
P_2	$f_3 = 5 - (Approach\_Level + \frac{1}{1190}(f_{a'}(a) + f_{b'}(b) + f_{c'}(c) + f_{d'}(d)))$, where $f_{a'}(a) = \begin{cases} \lvert a-1\rvert, & \text{if } a \geq 1 \\ \lvert a-1\rvert + 180, & \text{if } a < 1 \end{cases}$ $f_{b'}(b) = \begin{cases} \lvert b-2\rvert, & \text{if } b \geq 2 \\ \lvert b-2\rvert + 180, & \text{if } b < 2 \end{cases}$ $f_{c'}(c) = \begin{cases} \lvert c-3\rvert, & \text{if } c \geq 3 \\ \lvert c-3\rvert + 180, & \text{if } c < 3 \end{cases}$ $f_{d'}(d) = \begin{cases} \lvert d-4\rvert, & \text{if } d \geq 4 \\ \lvert d-4\rvert + 180, & \text{if } d < 4 \end{cases}$

Table 6.1 The Fitness Functions of P_1 and P_2

$Trace(CMS)^{n'}$ is higher than that of f_2 with respect to P_1 for all n'. Moreover, f_3 with P_2 has lowest $Trace(CMS)^{n'}$ for all n' because it is the easiest for ET. The results coincide the conclusions made from Chapter 5. From the results, we can also know that the isolation points sometimes can be visited under the condition that they are always encoded as points in the search space of GAs.

n'	f_1 vs. P_1	f_2 vs. P_1	f_3 vs. P_2
10	14.9973	14.3495	13.9687
12	9.2076	8.4146	7.9792
14	5.6073	4.9971	4.7871
16	3.3241	2.9591	2.7624
18	2.2372	2.1697	1.8531
20	1.7440	1.6540	1.5847
22	1.5246	1.5015	1.3598
24	1.3912	1.3851	1.2680
26	1.2905	1.2795	1.1730
Optimal Value	4.7945	4.8680	4.9388

Table 6.2 The Results of the Case Study

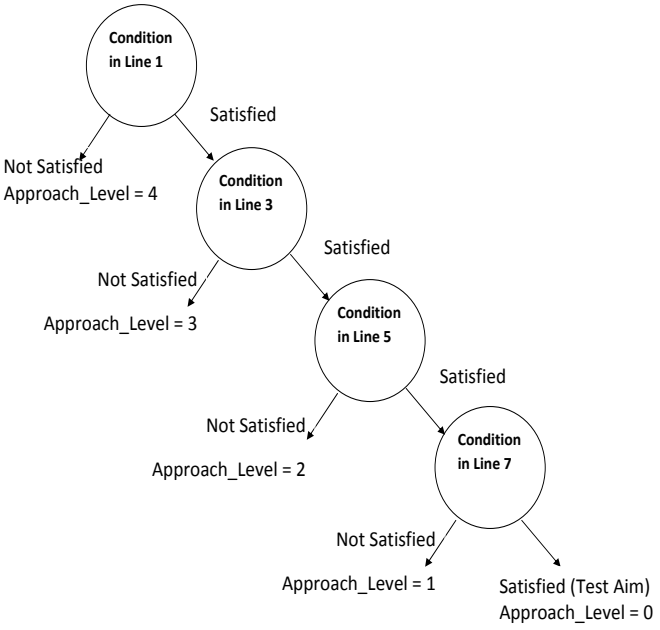


Figure 6.5 The Control Flow Graph for P_1 and P_2

CHAPTER 7. The Estimation of Global Convergence

This chapter presents a possible approach using the proposed evaluation metric in Chapter 5 for estimating the number of generations needed for the best solution generated by a CGA to converge to the global optima of a fitness function.

7.1 The Overview of the Estimation

The estimation includes the following three phases (see Figure 7.1):

- the accurate estimation on $Trace((CMS)^{n'})$ and $Trace((CMS)^{n'+k})$, for some $n', k \in N$,
- the generation of a fitting curve based on $Trace((CMS)^{n'})$ and $Trace((CMS)^{n'+k})$, and
- the derivation of the number of generations (for the global convergence) from the fitting curve and a small fixed ϵ.

Each phase is introduced below.

7.1.1 The Empirical Estimation on the Traces and the Fitting Curve

In the experiments of Chapter 5, estimated traces derived by the proposed evaluation metric are merely certain proportions of the actual ones. Without any other information, the exact actual trace is difficult to determine from the estimated trace. For accurately estimating a trace, one has to know the number of states visited by a GA run, which can be computed by directly counting the number of nodes in the linked list in Figure 5.4. Being aware of the number of states visited by a GA run, the exact proportion of the visited states over the total number of states is derived, which is used to estimate the actual trace below.

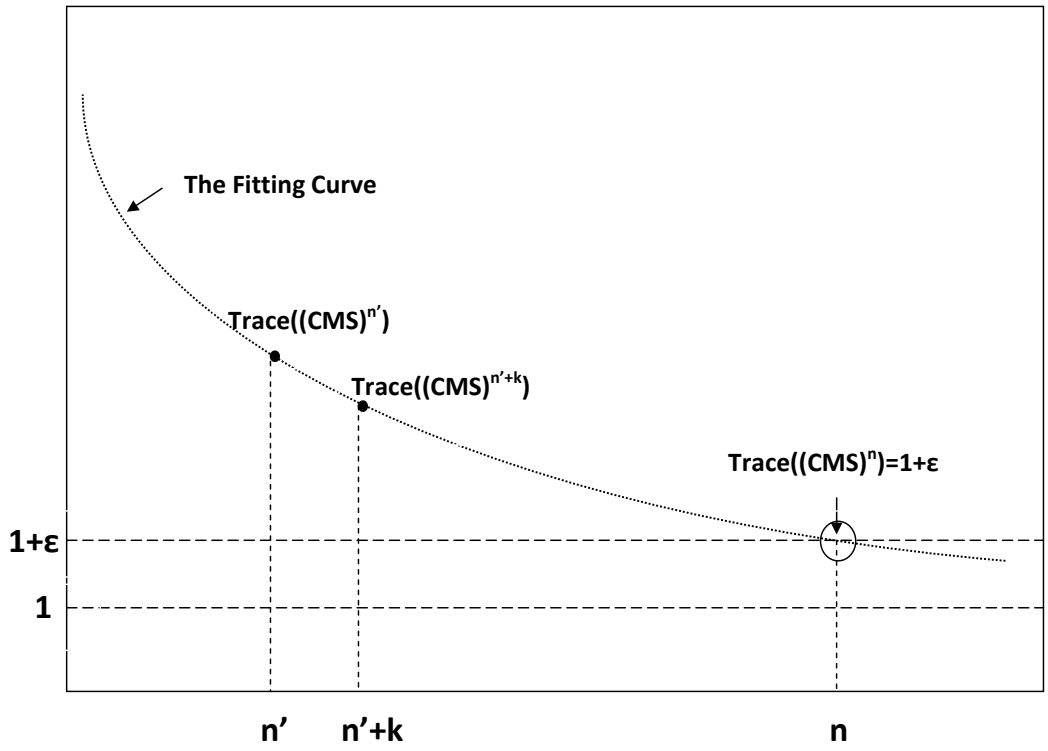


Figure 7.1 The Overview of the Estimation

Due to the limitation of the computer capability, the maximal order of the transition matrices (square matrices) estimated for the actual traces is 2^{10}. For this reason, the setting $m = 2$ is leveraged to develop the entire estimation framework. Similar frameworks can be derived by the same way for other settings of m. Table 7.1 and Table 7.2 show the relationships among the estimated trace, the proportion of the visited states over the total number of states, and the actual trace with respect to the fitness function $f_1(x) = x$. Note that the actual traces for the settings $l > 5$ are guessed by fitting a curve to the previous four points (i.e., $(m, l) = (2, 2), (2, 3), (2, 4)$ and $(2, 5)$).

As shown in both Table 7.1 and Table 7.2, the value, derived by dividing the estimated trace by the proportion of the visited states over the total number of states, is greater than the actual trace when the setting $m = 2$ and $l \geq 4$. Moreover, the relationship between the value and the setting of l is in Figure 7.2, which is based on the results from the setting $n' = 10$. Figure 7.3 shows the difference between the setting $n' = 10$ (the power of CMS) and $n' = 12$

Fitness Function: $f_1(x) = x$, $n' = 10$							
Settings	$m = 2, l = 2$	$m = 2, l = 3$	$m = 2, l = 4$	$m = 2, l = 5$	$m = 2, l = 6$	$m = 2, l = 7$	$m = 2, l = 8$
The Number of Generations: 500							
The (Average) Number of States Visited	10	27	54	89	121	157	186
Percentages of Visited States (%)	62.50	42.19	21.09	8.69	2.95	0.96	0.28
Estimated Trace by the Proposed Metric	1.9325	2.7694	3.6092	4.1131	4.4298	3.8777	3.5605
(Estimated Trace/Percentages of Visited States) ($*_{500}$)	3.0919	6.5644	17.1103	47.3240	149.9540	404.6681	1254.5194
($*_{500}$)/(The Actual Trace)	0.9357	1.0933	1.5674	2.3842	4.1543	6.1644	10.5078
The Number of Generations: 1000							
The (Average) Number of States Visited	13	35	77	141	208	284	352
Percentages of Visited States (%)	81.25	54.69	30.08	13.77	5.08	1.73	0.54
Estimated Trace by the Proposed Metric	2.2072	3.0862	4.6629	6.0658	6.8859	6.9667	6.5767
(Estimated Trace/Percentages of Visited States) ($*_{1000}$)	2.7166	5.6433	15.5027	44.0521	135.6001	401.9119	1224.4665
($*_{1000}$)/(The Actual Trace)	0.8221	0.9399	1.4202	2.2193	3.7566	6.1224	10.2561
The Number of Generations: 1500							
The (Average) Number of States Visited	13	39	90	176	279	398	507
Percentages of Visited States (%)	81.25	60.94	35.16	17.19	6.81	2.43	0.77
Estimated Trace by the Proposed Metric	2.2457	3.4820	5.2121	7.2691	8.6713	9.3192	9.3039
(Estimated Trace/Percentages of Visited States) ($*_{1500}$)	2.7640	5.7140	14.8255	42.2932	127.3033	383.6316	1202.6409
($*_{1500}$)/(The Actual Trace)	0.8365	0.9516	1.3581	2.1307	3.5268	5.8439	10.0733
The Number of Generations: 2000							
The (Average) Number of States Visited	14	42	100	203	335	498	647
Percentages of Visited States (%)	87.50	65.63	39.06	19.82	8.18	3.04	0.99
Estimated Trace by the Proposed Metric	2.3166	3.5612	5.5706	7.7670	10.0028	11.1726	11.5273
(Estimated Trace/Percentages of Visited States) ($*_{2000}$)	2.6475	5.4265	14.2607	39.1792	122.3023	367.5734	1167.6285
($*_{2000}$)/(The Actual Trace)	0.8012	0.9038	1.3064	1.9738	3.3882	5.5993	9.7800
The Number of Generations: 5000							
The (Average) Number of States Visited	15	51	139	306	570	933	1342
Percentages of Visited States (%)	93.75	79.69	54.30	29.88	13.92	5.69	2.05
Estimated Trace by the Proposed Metric	2.4505	4.0357	6.4425	9.8537	14.0102	17.9755	20.9455
(Estimated Trace/Percentages of Visited States) ($*_{5000}$)	2.6139	5.0644	11.8654	32.9744	100.6768	315.6592	1022.8634
($*_{5000}$)/(The Actual Trace)	0.7911	0.8434	1.0870	1.6612	2.7891	4.8085	8.5675
The Actual Trace	3.3043	6.0044	10.9161	19.8492	36.0964 (Guess)	65.6460 (Guess)	119.3893 (Guess)

Table 7.1 The Relationships among the Estimated Trace, the Proportion of the Visited States over the Total Number of States ($n' = 10$)

Fitness Function: $f_1(x) = x$, $n' = 12$							
Settings	$m = 2, l = 2$	$m = 2, l = 3$	$m = 2, l = 4$	$m = 2, l = 5$	$m = 2, l = 6$	$m = 2, l = 7$	$m = 2, l = 8$
The Number of Generations: 500							
The (Average) Number of States Visited	10	27	54	88	121	156	185
Percentages of Visited States (%)	62.50	42.19	21.09	8.59	2.95	0.95	0.28
Estimated Trace by the Proposed Metric	1.8528	2.5426	3.1317	3.2154	3.2348	3.1077	2.6261
(Estimated Trace/Percentages of Visited States) ($*_{500}$)	2.9644	6.0269	14.8466	37.4151	109.5020	326.3875	930.2841
($*_{500}$)/(The Actual Trace)	0.9303	1.0595	1.4620	2.0641	3.3843	5.6512	9.0237
The Number of Generations: 1000							
The (Average) Number of States Visited	13	35	77	140	207	284	351
Percentages of Visited States (%)	81.25	54.69	30.08	13.67	5.05	1.73	0.54
Estimated Trace by the Proposed Metric	2.0982	2.8219	4.0908	4.9186	5.4002	5.4528	5.0476
(Estimated Trace/Percentages of Visited States) ($*_{1000}$)	2.5824	5.1601	13.6007	35.9760	106.8554	314.5709	942.4512
($*_{1000}$)/(The Actual Trace)	0.8104	0.9071	1.3393	1.9847	3.3025	5.4466	9.1417
The Number of Generations: 1500							
The (Average) Number of States Visited	13	39	90	176	279	397	506
Percentages of Visited States (%)	81.25	60.94	35.16	17.19	6.81	2.42	0.77
Estimated Trace by the Proposed Metric	2.1311	3.1494	4.6002	5.9594	6.8408	7.3889	7.0296
(Estimated Trace/Percentages of Visited States) ($*_{1500}$)	2.6228	5.1682	13.0850	34.6728	100.4298	304.9358	910.4647
($*_{1500}$)/(The Actual Trace)	0.8231	0.9085	1.2886	1.9128	3.1039	5.2798	8.8315
The Number of Generations: 2000							
The (Average) Number of States Visited	14	42	100	203	334	497	647
Percentages of Visited States (%)	87.50	65.63	39.06	19.82	8.15	3.03	0.99
Estimated Trace by the Proposed Metric	2.1711	3.2489	4.8517	6.3587	7.9047	8.7813	8.7114
(Estimated Trace/Percentages of Visited States) ($*_{2000}$)	2.4812	4.9507	12.4203	32.0755	96.9392	289.4821	882.3972
($*_{2000}$)/(The Actual Trace)	0.7786	0.8703	1.2231	1.7695	2.9960	5.0122	8.5592
The Number of Generations: 5000							
The (Average) Number of States Visited	15	51	139	306	570	933	1342
Percentages of Visited States (%)	93.75	79.69	54.30	29.88	13.92	5.69	2.05
Estimated Trace by the Proposed Metric	2.3152	3.7109	5.6716	8.3251	11.1602	13.9268	16.0126
(Estimated Trace/Percentages of Visited States) ($*_{5000}$)	2.4696	4.6569	10.4456	27.8591	80.1966	244.5628	781.9699
($*_{5000}$)/(The Actual Trace)	0.7750	0.8186	1.0286	1.5369	2.4786	4.2344	7.5851
The Actual Trace	3.1866	5.6887	10.1548	18.1266	32.3562 (Guess)	57.7557 (Guess)	103.0934 (Guess)

Table 7.2 The Relationships among the Estimated Trace, the Proportion of the Visited States over the Total Number of States ($n' = 12$)

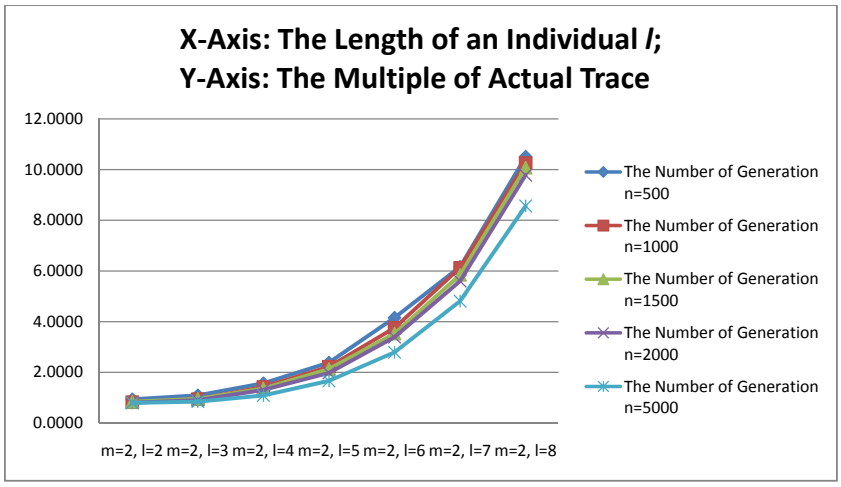


Figure 7.2 The relationship between the value and the setting of l

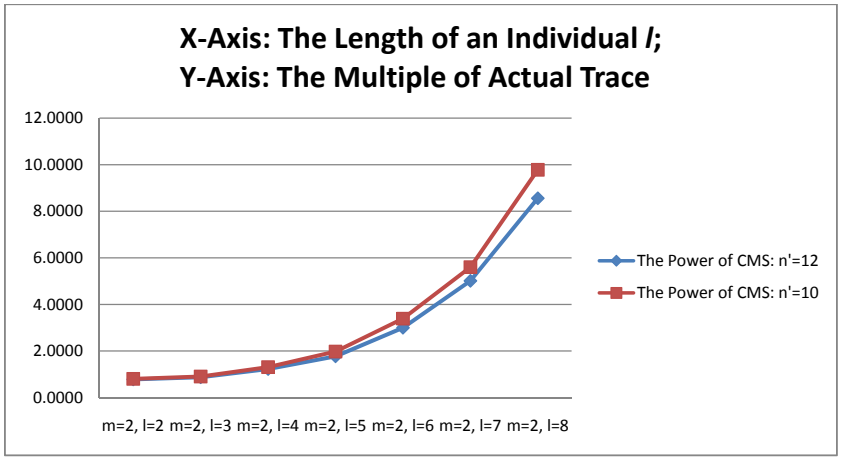


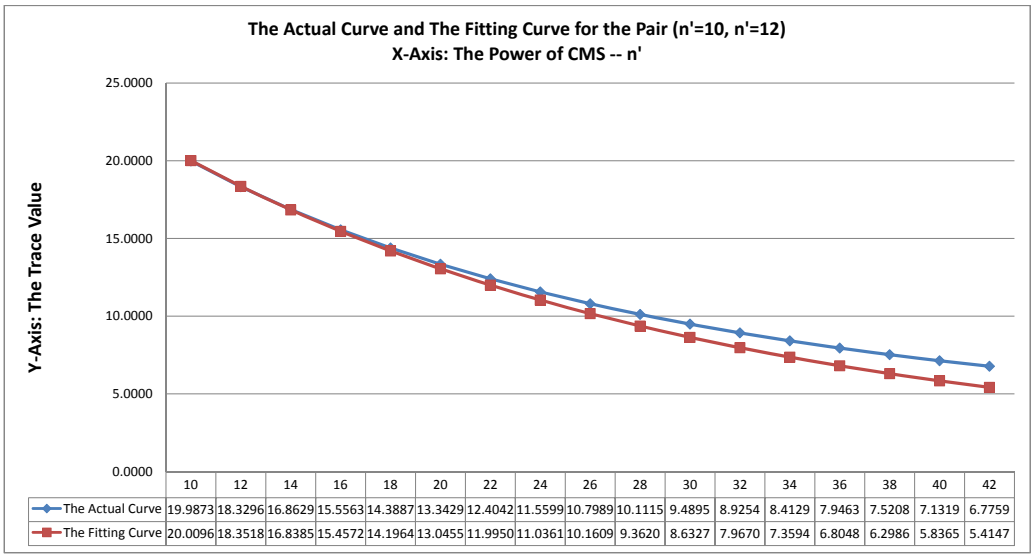
Figure 7.3 The relationship between the value and the setting of l

when the number of generations is set to 2000. The observations on other fitness functions have similar results, except the values being slightly different.

Based on my investigation, different fitness functions, with fixed numbers of generations, have different multiple functions (as shown in Figure 7.2) for actual traces, but all of them can be fitted by the curve $y = \alpha^{l-3}$ when $l \geq 4$, where $\alpha \geq 1$. Note that different settings on the number of generations have different values for α. For each fitness function, we can always compute its own multiple functions to derive the (estimated) actual traces for estimating the number of generations for the empirical global convergence; however, that idea has some

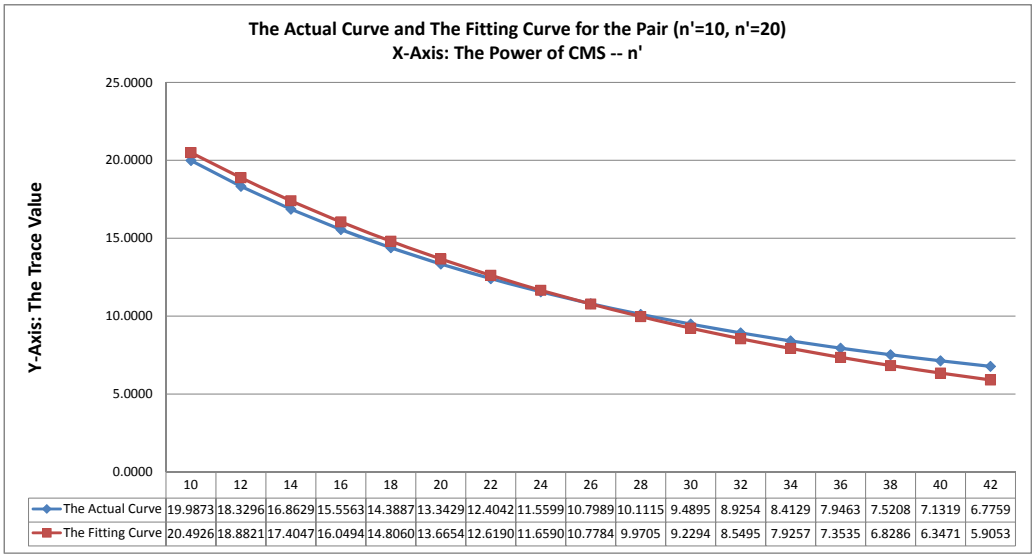
drawbacks. We have to compute everything for each fitness function, which costs a lot of computation time. In addition, computation errors of the multiple functions may sometimes result in the situation that the estimated λ_2 (i.e., the second largest eigenvalue in absolute value) is greater than 1. In order to conquer the above problems and provide a robust method for the estimation, the entire estimation framework is analyzed and simplified. The following explains how the framework works.

Let us discuss the equation planned to be used as the fitting curve in Figure 7.1 first. Theoretically, the curve should be $y = f(x) = 1 + (m_2\lambda_2)^x + (m_3\lambda_3)^x + \cdots + (m_q\lambda_q)^x$, where $m_2, \ldots, m_q$ and $\lambda_2, \ldots, \lambda_q$ are defined in Theorem 4 in Chapter 3; however, to inlcude all of the parameters ($m_2, \ldots, m_q$ and $\lambda_2, \ldots, \lambda_q$) costs a lot of computation overhead. As it is commonly known that $\lambda_2, \ldots, \lambda_q$ are all less than 1, for a large enough x, the terms $(m_3\lambda_3)^x, \ldots, (m_q\lambda_q)^x$ can be ignored. The equation $y = f(x) = 1 + (m_2\lambda_2)^x$ is thusly applied to fit the curve. In this case, merely two points are needed for the fitting. A proper set of fitness functions with different difficulty levels is chosen for estimating λ_2. With an arbitrary setting of m and l, the estimations of λ_2 derived by three pairs of points, $(n' = 10, n' = 12), (n' = 10, n' = 20)$, and $(n' = 20, n' = 22)$, are compared. Based on the results, it is considered true that λ_2 computed based on the pair $(n' = 20, n' = 22)$ is always the largest. For instance, with respect to the fitness function $y = f_3(x) = 1+sin(x)$ with the setting m equal to 2 and l equal to 5, λ_2 derived by the pair $(n' = 10, n' = 12)$ is equal to 0.9554, λ_2 derived by the pair $(n' = 10, n' = 20)$ is equal to 0.9578, and λ_2 derived by the pair $(n' = 20, n' = 22)$ is equal to 0.9612. The reason is that, at the point $x = 10$, $(m_3\lambda_3)^x + \cdots + (m_q\lambda_q)^x$ is not really close to 0, which causes some noises for the estimation. Although the λ_2 derived by the pair $(n' = 20, n' = 22)$ is the best among the three, the noises caused by $(m_3\lambda_3)^x + \cdots + (m_q\lambda_q)^x$ still exist, which makes the estimated λ_2 slightly smaller than the actual λ_2. The fitting curves based on those three λ_2's and their corresponding m_2's are drawn in Figure 7.4, Figure 7.5, and Figure 7.6, respectively. Comparing three of them, the curve generated by the pair $(n' = 20, n' = 22)$ is the best fit for the actual curve. Note that the n' cannot be too large when it is applied to estimate λ_2 for any fitness function. Too large n' may cause some errors due to precision problems.



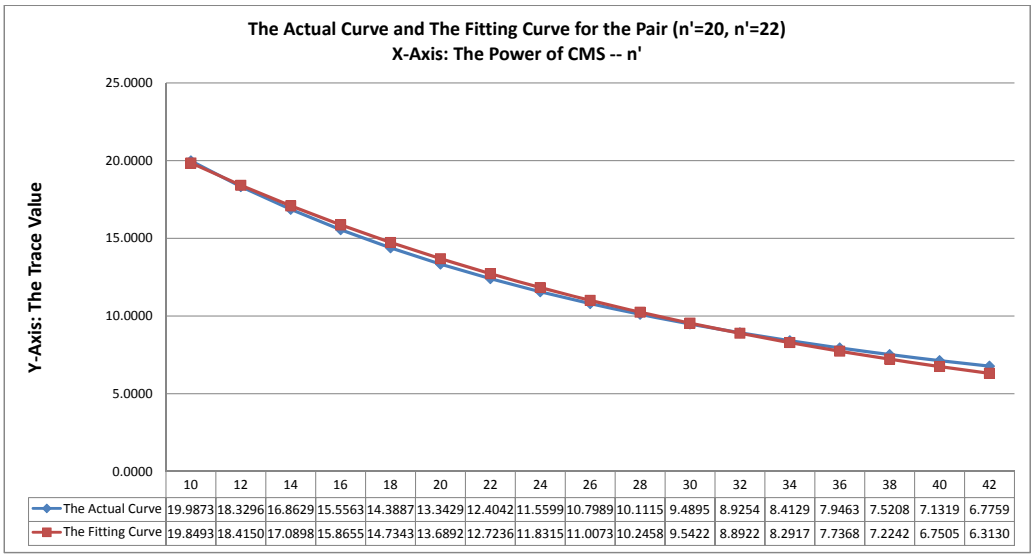
	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42
The Actual Curve	19.9873	18.3296	16.8629	15.5563	14.3887	13.3429	12.4042	11.5599	10.7989	10.1115	9.4895	8.9254	8.4129	7.9463	7.5208	7.1319	6.7759
The Fitting Curve	20.0096	18.3518	16.8385	15.4572	14.1964	13.0455	11.9950	11.0361	10.1609	9.3620	8.6327	7.9670	7.3594	6.8048	6.2986	5.8365	5.4147

Figure 7.4 The relationship between the value and the setting of l



	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42
The Actual Curve	19.9873	18.3296	16.8629	15.5563	14.3887	13.3429	12.4042	11.5599	10.7989	10.1115	9.4895	8.9254	8.4129	7.9463	7.5208	7.1319	6.7759
The Fitting Curve	20.4926	18.8821	17.4047	16.0494	14.8060	13.6654	12.6190	11.6590	10.7784	9.9705	9.2294	8.5495	7.9257	7.3535	6.8286	6.3471	5.9053

Figure 7.5 The relationship between the value and the setting of l



	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42
The Actual Curve	19.9873	18.3296	16.8629	15.5563	14.3887	13.3429	12.4042	11.5599	10.7989	10.1115	9.4895	8.9254	8.4129	7.9463	7.5208	7.1319	6.7759
The Fitting Curve	19.8493	18.4150	17.0898	15.8655	14.7343	13.6892	12.7236	11.8315	11.0073	10.2458	9.5422	8.8922	8.2917	7.7368	7.2242	6.7505	6.3130

Figure 7.6 The relationship between the value and the setting of l

This paragraph mainly discusses and analyzes the approach adopted for estimating traces with certain powers n' and the corresponding λ_2. Consider an arbitrary fitness function. With respect to this fitness function, we can compute two multiple functions for both $n' = 20$ and $n' = 22$. Suppose the value of the multiple function for $n' = 20$ at $l = 16$ is $Mul_{20}(16)$, and the value of the multiple function for $n' = 22$ at $l = 16$ is $Mul_{22}(16)$. The traces for $n' = 20$ and $n' = 22$ are $\frac{\text{Estimated Trace } (n'=20)}{\text{Percentages of Visited States}}/Mul_{20}(16)$ and $\frac{\text{Estimated Trace } (n'=22)}{\text{Percentages of Visited States}}/Mul_{22}(16)$, respectively. Note that the fraction $\frac{\text{Estimated Trace}}{\text{Percentages of Visited States}}$ is used for estimating traces because the proportion of visited states roughly presents the percentages of traces discovered by the evaluation metric. From Table 7.1 and Table 7.2, we know that the fraction overestimates the traces by some multiples when $l \geq 4$. The multiple functions are leveraged to solve the overestimated problem. The reason for the overestimation is that usually the sum of the relatively larger diagonal elements is discovered and computed by the evaluation metric. The amount is already included in the estimated trace. To estimate the sum of smaller diagonal elements using the sum of the relatively larger diagonal elements causes the problem. After deriving the

estimated traces for $n' = 20$ and $n' = 22$, the equations

$$\begin{cases} 1 + m_2\lambda_2^{20} = Trace((CMS)^{20}) \\ 1 + m_2\lambda_2^{22} = Trace((CMS)^{22}) \end{cases}$$

are applied to estimate λ_2. From the equations, we obtain

$$\begin{aligned} \lambda_2^2 &= \frac{Trace((CMS)^{22}) - 1}{Trace((CMS)^{20}) - 1} \\ &= \frac{Trace((CMS)^{22})}{Trace((CMS)^{20})} + \left(\frac{Trace((CMS)^{22}) - 1}{Trace((CMS)^{20}) - 1} - \frac{Trace((CMS)^{22})}{Trace((CMS)^{20})}\right) \\ &= \frac{Trace((CMS)^{22})}{Trace((CMS)^{20})} - \left(\frac{1}{Trace((CMS)^{20}) - 1} - \frac{Trace((CMS)^{22})}{Trace((CMS)^{20})(Trace((CMS)^{20}) - 1)}\right). \end{aligned} \tag{7.1}$$

Note that the second term of the last line in Equation (7.1) can be ignored if $Trace((CMS)^{20})$ is large enough (e.g., $Trace((CMS)^{20}) \geq 1000$). We use a function $Correction(Trace((CMS)^{20}), Trace((CMS)^{22}))$ to denote it. Combined with the computations mentioned earlier, Equation (7.1) becomes

$$\begin{aligned} \lambda_2^2 =& (\frac{\text{Estimated Trace } (n' = 22)}{\text{Percentages of Visited States}}/Mul_{22}(16))/(\frac{\text{Estimated Trace } (n' = 20)}{\text{Percentages of Visited States}}/Mul_{20}(16)) \\ & - Correction(Trace((CMS)^{20}), Trace((CMS)^{22})), \end{aligned} \tag{7.2}$$

which can be re-written as

$$\begin{aligned} \lambda_2^2 =& \frac{Mul_{20}(16)}{Mul_{22}(16)}\left((\frac{\text{Estimated Trace } (n' = 22)}{\text{Percentages of Visited States}})/(\frac{\text{Estimated Trace } (n' = 20)}{\text{Percentages of Visited States}})\right) \\ & - Correction(Trace((CMS)^{20}), Trace((CMS)^{22})). \end{aligned} \tag{7.3}$$

If both of $Trace((CMS)^{20})$ and $Trace((CMS)^{22})$ are estimated by the proposed evaluation metric at the same time (i.e., they are computed using the same set of random numbers), Equation (7.3) can be simplified and re-written as

$$\lambda_2^2 = \frac{Mul_{20}(16)}{Mul_{22}(16)}\left(\frac{\text{Estimated Trace } (n' = 22)}{\text{Estimated Trace } (n' = 20)}\right) - Correction(Trace((CMS)^{20}), Trace((CMS)^{22})), \tag{7.4}$$

since both of them have the same value for *Percentages of Visited States*.

Now let us carefully consider an approximation equation for Equation (7.4) using the limited known information. As mentioned before, it is time consuming to compute the multiple functions for each fitness function. In order to get more precise results, one may choose to perform that. My goal is to simplify the computation process. Hence, an approximation approach is suggested. Based on my observation, the results show that with respect to the same setting of l, the rates $\frac{Mul_{20}(16)}{Mul_{22}(16)}$ of different fitness functions are close; however, fitness functions which have smaller numbers of states visited by GAs tend to have slightly larger rates. The fitness function $f_7(x) = sin^6(5\pi[x^{3/4} - 0.05])$ is chosen as the referenced fitness function. (Note that other fitness functions can also be used as the referenced fitness function.) For different l ($l \geq 4$), the rate $\frac{Mul_{20}(l)}{Mul_{22}(l)}$ of f_7 (when the number of generations n is set to 2000) is approximate to

$$\frac{Mul_{20}(l)}{Mul_{22}(l)} = (1.0072)^{l-3}.$$

Some adjusted coefficients need to be applied for other fitness functions. The one suggested is the fraction $\frac{\text{The Number of States Visited for } f_7}{\text{The Number of States Visited for the Fitness Function}}$ for a specific fitness function. The approximation equation for Equation (7.4) can be

$$\lambda_2^2 \approx (1.0072)^{l-3}(\text{The Adjusted Coefficient}) \left(\frac{\text{Estimated Trace } (n' = 22)}{\text{Estimated Trace } (n' = 20)} \right), \tag{7.5}$$

where $l \geq 4$. The $Correction(Trace((CMS)^{20}), Trace((CMS)^{22}))$ is ignored because for most of the cases, we need Equation (7.5) to estimate λ_2 for large scale problems, in which case, l is greater than 14. When $l \geq 14$ and $n' = 20$, the traces are around or greater than 1000.

7.2 The Number of Generations for the Empirical Global Convergence

After obtaining λ_2, the equation $1 + m_2\lambda_2^{20} = Trace((CMS)^{20})$ is used to estimate m_2. Before that, $Trace((CMS)^{20})$ needs to be computed first. By the computation in the previous section, for each fitness function, $Trace((CMS)^{20})$ is equal to the fraction $\frac{\text{Estimated Trace } (n'=20)}{\text{Percentages of Visited States}}$ divided by a number $Mul_{20}(l)$ corresponding to the fitness function. According to the results on the values of $Mul_{20}(l)$ for fitness functions in the observed set, the largest $Mul_{20}(l)$ is around ten times of the smallest one. As mentioned before, it is not very efficient to compute

the multiple functions for each fitness function. Hence, using one of the multiple functions instead of all of them is suggested. Note that the selection of the value $Mul_{20}(l)$ only impacts m_2, which is employed in computing n from the equation $1 + m_2\lambda_2^n = 1 + \epsilon$ (as in Figure 7.1), in the entire estimation framework. An equivalent equation for $1 + m_2\lambda_2^n = 1 + \epsilon$ is

$$n = \frac{log(\frac{\epsilon}{m_2})}{log(\lambda_2)}. \tag{7.6}$$

From Equation (7.6), we can get that λ_2 is more sensitive than m_2. Moreover, Figure 7.6 shows that the fitting curve computed by the proposed estimation method is eventually lower than the curve drawn by the actual traces after n exceeds a value. That means the number of generations for the global convergence estimated by the proposed method is smaller than the actual one. Hence, the smallest $Mul_{20}(l) = 158.5566 \approx 159$ (from the observed set) is leveraged so that the largest m_2 can be used to solve n in Equation (7.6).

7.3 The Verification of the Proposed Estimation Framework

The numbers of generations for the empirical global convergence with respect to the fitness functions f_1 to f_8 in Chapter 5 are derived in Table 7.3.

Table 7.3 The Empirical Global Convergence for Each Fitness Function

Fitness Function	λ_2	m_2	The Number of Generations Derived from the Proposed Approximation Method	The Observed Number of Generations for the Empirical Convergence
f_1	0.8490	102814	106	7000
f_2	0.8434	150613	105	6000
f_3	0.8890	76333	145	8500
f_4	0.9339	26663	234	13500
f_5	0.8646	87450	119	8000
f_6	0.8699	84670	124	8500
f_7	0.9439	85533	298	16000
f_8	0.9265	133100	231	11000

The last column of Table 7.3 derived by observing the convergence trends of the fitness functions from 40 trials. For the non-periodic fitness functions, the values are the numbers

of generations in which GAs reach the global optimal (or near optimal) solutions. Note that the isolation points of the fitness functions f_5 and f_6 are ignored here. In other words, when $n \geq 8000$, the best solutions for the fitness function f_5 reach 90000.0. Sometimes, they can reach 100000.0 for some trials. However, based on my observation, the best solutions have certain difficulty to reach 100000.0 for all trials even for a very large n. Similar results are found for f_6. For the periodic fitness functions f_3 and f_7, it is almost impossible for their best solutions to reach only a single point since they have more than one global optimal solutions. To solve this problem, a GA is considered as converged if the best solution maintained over time x satisfies

$$|f_i(x) - \text{Global Optimal Value}| < 10^{-6},$$

where $i \in \{3, 7\}$.

Comparing the numbers of generations derived by the proposed method and the observed numbers of generations for the empirical convergence, we can get that the later ones are in certain multiples of the former ones. The multiples are in between 47 to 67. Hence, the number of generations derived by the proposed method is useful for providing some information about the empirical global convergence if a certain multiple greater than 67 (e.g., 100) is applied on it.

Another important result derived from Table 7.3 is that, the convergence mainly depends on the value λ_2. m_2, related to the constant in Inequality (4.3), does not have significant impact on it. This coincides with our earlier result (see Inequality (4.3)) and many work in the literature that the second largest eigenvalue represents the convergence rate of a Markov chain.

CHAPTER 8. The Generalization of the Proposed Methodology

As the aforementioned, the proposed evaluation metric is based on the condition that CMS is regular, which is obtained from the assumption that CGA is applied to solve the optimization problems. In this chapter, I am going to prove that in addition to CGA, the evaluation metric is possibly to be applied to any type of selection, and crossover operators, as long as bit mutation is performed after selection and crossover operators.

Theorem 9 *Regardless of the types of selection and crossover operators, the transition matrix SCM is regular if and only if bit mutation is performed after selection and crossover.*

Proof. Since S and C are stochastic matrices, SC is a stochastic matrix (i.e., sum of elements in a row is equal to 1 and each element of SC is nonnegative). Moreover, because bit mutation is performed, for all i and j, the element m_{ij} in M is

$$m_{ij} = \prod_{k=1}^{m} p_m^{H(\pi_k(i),\pi_k(j))}(1-p_m)^{l-H(\pi_k(i),\pi_k(j))},$$

where $H(\cdot,\cdot)$ is the Hamming distance of the strings (chromosomes). Hence, m_{ij} is positive for all i and j. That means M is positive.

Since SC is a stochastic matrix and M is positive, SCM is positive. Therefore, SCM is regular.□

With the condition that SCM is regular, the evaluation metric can be applied generally — not limited to certain types of selection and crossover operators. The corresponding properties investigated and checked in previous chapters should also be further investigated if different selection or crossover operators are used.

Theorem 10 *The transition matrix A constructed by Rudolph's version can be converted to a transition matrix B in Vose's version. Moreover, A is positive implies that B is positive.*

Proof. Let us consider an arbitrary state, $State_i$, of the transition matrix A constructed by Rudolph's version. Suppose the number of different individuals in $State_i$ is r, each individual is assigned a distinct number between 1 and r, and m_j, $j \in \{1, 2, \ldots, r\}$, is the corresponding number of occurrences for $Individual_j$ (in State i). Note that $m_1 + m_2 + \cdots + m_r = m$.

By Rudolph's state representation, there are $\frac{m!}{m_1!m_2!\ldots m_r!}$ states which are a permutation of individuals in $State_i$ in the transition matrix A. In Vose's version, those states, including $State_i$, are all in one state since each state represents only the number of occurrences of the individuals. To convert the transition matrix A in Rudolph's version to a transition matrix B in Vose's version, one can simply add up the columns that correspond to the destination states (in Rudolph's version) considered as the same state in Vose's version. Similarly, the rows which correspond to the source states (in Rudolph's version) considered as the same state in Vose's version are added up. For the computation of rows, a multiple of $1/\frac{m!}{m_1!m_2!\ldots m_r!}$ needs to be applied to the result. The reason is that given a state with r different individuals in Vose's version, there is $1/\frac{m!}{m_1!m_2!\ldots m_r!}$ possibility that this state is permutation k, where $k \in \{1, 2, \ldots, \frac{m!}{m_1!m_2!\ldots m_r!}\}$, in Rudolph's version (see Figure 8.1).

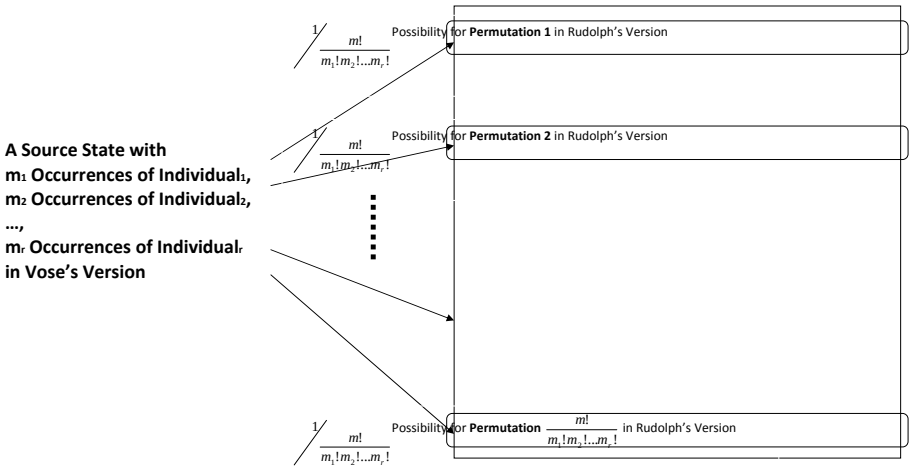


Figure 8.1 An Illustration of the Conversion

Based on the computation above, we get that if A is positive, then B is positive since the adding and multiplying manipulations of positive numbers are still positive. □

In Rudolph's paper [Rudolph, G. (1994)], it is shown that CMS constructed by CGA is positive. With Theorem 9, we know that SCM constructed by a GA, with bit mutation being performed after any selection and crossover operations, is also positive. The proposed evaluation metric can be generalized to a GA with bit mutation being performed after any selection and crossover operations using Vose's version for state representation, since the converted transition matrix in Vose's version is positive (also regular). More experiments on the performance of the proposed evaluation metric with states being represented by Vose's version need to be conducted. I believe that the estimated trace of a transition matrix in Vose's version will be more accurate than the estimated trace of the transition matrix in Rudolph's version. This will be one of the directions of my future work.

CHAPTER 9. Conclusion and Future Work

The conclusion of this dissertation includes three sections. The first section describes the contributions of this work; the second section presents the limitations of the evaluation metric and the proposed approximation approach. In the final section, future work is discussed.

9.1 Contributions of this Work

The contributions of this work include the following:

- I derive the convergence rate of PU (the extended transition matrix) from Perron's Formula.

- I propose a framework for estimating the applicability of GAs for real-world optimization problems, formulate several corresponding theorems, conduct experiments for the proposed framework and analyze the experimental results, and study several cases in Software Testing and verify the evaluation metric.

- With the evaluation metric, researchers can decide a cluster of better fitness functions among a set of fitness functions by inputting the set of fitness functions with the same GA configuration. This feature is very important for real-world applications, since researchers or practioners may consider using more than one fitness function when they attempt to solve optimization problems. The evaluation metric can rank the fitness functions from the best to the worst by outputting the fitness functions sorted by degrees of convergence.

  The traditional way (also the only way) to discriminate the fitness functions with respect to a GA configuration is to compute their Markov transition matrices. With a number of 2^{ml} states, at least $2^{ml} \times 2^{ml} \times m$ fitness evaluations need to be performed, which costs

a lot of time. Comparison of the fitness values after a fixed number of generations is not a trustworthy method, since the chosen fixed number may be too large, so that all of the transition matrices empirically converge to their limits.

- An approximation approach is presented for applying the proposed evaluation metric to estimate the number of generations for the global convergence of GAs.

9.2 Limitations

Both the evaluation metric and the proposed approximation approach have limitations as follows:

- The proposed evaluation metric has the ability to rank the difficulties of fitness functions with respect to a configuration of GAs; however, it is difficult for the metric to derive a precise value of trace. A too small n makes the metric arduous to predict the relative convergence behavior of a flat fitness function. For a large enough n, an absolute numerical value related to a degree of convergence can be derived.

- The proposed approximation approach for predicting the number of generations for the global convergence of GAs has difficulty in dealing with fitness functions that have isolation points. The empirical results show that with respect to such a fitness function, when n is equal to the number of generations derived from the approximation approach, the best solution maintained over time reaches a global optimal solution of the fitness function other than the isolation points in continuous 40 trials. Only a few of the trials obtain the best solutions equal to the isolation points.

Further investigation is needed to solve the limitations.

9.3 Future Work

My future work will be focused on extending and improving the precision of the proposed evaluation metric. Currently, the metric is able to evaluate the applicability of CGAs with best solution maintained over time, and is extended to a more general case in Chapter 7 (e.g.,

the applicability of EAs that perform bit mutation after selection and crossover operators). Further investigation on more classes of EAs are necessary. Chapter 8 also proves that the state representation using Rudolph's version can be converted to the state representation using Vose's version. I believe that this conversion is helpful for improving the precision of the trace estimated by the proposed evaluation metric, since certain states represented using Rudolph's version are combined into a state in Vose's version. The diagonal elements of the transition matrices constructed by Vose's version have larger values than those of the transition matrices constructed by Rudolph's version, and the state space in Vose's version is smaller than that in Rudolph's version. These reasons make the estimation easier in Vose's version. More research work and experiments need to be conducted along this line.

An alternative approach or adjustment of the approximation is another future direction. The approximation can currently predict the number of generations for the global convergence of GAs; however, a solution to deal with its limitation is needed. I will investigate the Markov transition matrices constructed by the fitness functions with isolation points and solve the problem by either adjusting the proposed approximation or developing a novel approach.

The design of fitness functions of multi-objective problems is the other direction for future research. If the fitness functions cannot be well-formed for multi-objective problems, some of the objectives will not be considered during the GA runs. In the future, I will follow this approach, and derive suitable rules for researchers to deal with this problem.

BIBLIOGRAPHY

Baresel, A. and Pohlheim, H. and Sadeghipour, S. (2003). *Lecture Notes in Computer Science: Structural and Functional Sequence Test of Dynamic and State-Based Software with Evolutionary Algorithms*. Springer-Verlag Berlin Heidelberg.

Baresel, A. and Binkley, D. and Harman, M. and Korel, B. (2004). Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach. *Proceedings of the International Symposium on Software Testing and Analysis*, 108–118.

Beizer, B. (1990). *Software Testing Techniques (second edition)*. International Thomson Computer Press.

Berndt, D. and Fisher, J. and Johnson, L. and Pinglikar, J. and Watkins, A. (2003). Breeding Software Test Cases with Genetic Algorithms. *proceedings of Hawaii International Conference on System Sciences*, 338–347.

Berndt, D. J. and Watkins, A. (2004). Investigating the Performance of Genetic Algorithm-Based Software Test Case Generation. *proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 261–262.

Berndt, D. J. and Watkins, A. (2005). High Volume Software Testing using Genetic Algorithms. *proceedings of Hawaii International Conference on System Sciences*, 318–326.

Bethke, A. D. (1980). Genetic Algorithm as Function Optimizers. *Ph.D. Dissertation, University of Michigan*.

Briand, L. C. and Labiche, Y. and Shousha, M. (2004). Performance Stress Testing of Real-Time Systems Using Genetic Algorithms. *Technical Report: SCE 03-23, Carleton University.*

Briand, L. C. and Labiche, Y. and Shousha, M. (2005). Stress testing real-time systems with genetic algorithms. *proceedings of Conference on Genetic and Evolutionary Computation*, 1021-1028.

Bridges, C. L. and Goldberg, D. E. (1991). *The Nonuniform Walsh-Schema Transform.* San Mateo, CA: Morgan Kaufmann.

Brown, L. D. and Cai, T. T. and DasGupta, A. (2001). Interval Estimation for a Binomial Proportion. *Statistical Science*, *16*(2), 101–117.

Burden, R. L. and Faires, J. D. (2005). *Numerical Analysis, 8th Edition.* Brooks/Cole.

Chang, C. K. and Christensen, M. J. (1999). A Net Practice for Software Project Management. *IEEE Software*, *16*(6), 80–88.

Chang, C. K. and Christensen, M. J. and and Zhang, T. (2001). Genetic Algorithms for Project Management. *Annals of Software Engineering 11*, 107–139.

Chang, C. K. and Cleland-Haung, J. and Hua, S. and Kuntzmann-Combelles, A. (2001). Function-Class Decomposition: A Hybrid Software Engineering Method. *Computer*, *34*(12), 87–93.

Chao, C. (1995). SPMNET: A New Methodology For Software Management. *Ph.D. Dissertation, University of Illinois at Chicago.*

Ching, W. K. and Ng, M. K. (2006). *Markov Chains: Models, Algorithms and Applications.* Springer Science + Business Media, Inc.

Clark, J. and Dolado, J. J. and Harman, M. and Hierons, R. M. and Jones, B. and Lumkin, M. and Mitchell, B. and Mancoridis, S. and Rees, K. and Roper, M. and Shepperd, M. (2003). Reformulating Software Engineering as a Search Problem. *IEE Proceedings - software*, *150*(3), 161–175.

Coley, D. A. (1999). *An Introduction to Genetic Algorithms for Scientists and Engineers*. New Jersey: World Scientific Publishing Co. Pte. Ltd.

Davis, T. E. and Principe, J. C. (1993). A Markov framework for the simple genetic algorithm. *Evolutionary Computation*, *1*(3), 269–288.

DeJong, K. A. and Spears, W. M. and Gordon, D. F. (1995). Using Markov Chains to Analyze GAFOs. *Foundations of Genetic Algorithms 3*, 115–137.

Ding, L. and Yu, J. (2005). Some Theoretical Results About the Computation Time of Evolutionary Algorithms. *proceedings of Conference on Genetic and Evolutionary Computation*, 1409–1415.

Droste, S. and Jansen, T. and Wegener, I. (1998). A rigorous complexity analysis of the (1+1) evolutionary algorithm for linear functions with Boolean inputs. *Proceedings of International Conference on Evolutionary Computation (ICEC)*, 499–504.

Eiben, A. E. and Aarts, E. H. L. and Hee, K. M. V. (1991). Global convergence of genetic algorithms: A markov chain analysis. *Parallel Problem Solving from Nature*, *496*, 3–12.

Fogel, D. B. (1995). *Evolutionary Computation*. New York: IEEE Press.

Forrest, S. and Mitchell, M. (1993). What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation. *Machine Learning*, *13*, 285–319.

Ge, Y. (2004). Capability-based Software Project Scheduling with System Dynamics and Heuristic Search. *Master's Thesis, Iowa State University*.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional.

Grefenstette, J. J. (1992). *Foundations of Genetic Algorithms 2: Deception Considered Harmful*. San Mateo, CA: Morgan Kaufmann

Hadley, G. (1964). *Nonlinear and Dynamics Programming*. Addison Wesley

He, J. and Kang, L. (1999). On the convergence rates of genetic algorithms. *Theoretical Computer Science*, *229*, 23–39.

He, J. and Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, *127*(1), 57–85.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press.

Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press.

Horn, J. and Goldberg, D. E. (1995). *Foundations of Genetic Algorithms 3: Genetic Algorithm Difficulty and the Modality of Fitness Landscapes*. San Francisco, CA: Morgan Kaufmann.

Horn, J. (1995). Genetic Algorithms, Problem Difficulty, and the Modality of Fitness Landscapes. *Master's Thesis, University of Illinois at Urbana-Champaign*.

Iosifescu, M. (1980). *Finite Markov Processes and Their Applications*. John Wiley and Sons.

Jiang, H. and Chang, C. K. and Zhu, D. and Cheng, S. (2007). A Foundational Study on the Applicability of Genetic Algorithm to Software Engineering Problems. *proceedings of Congress on Evolutionary Computation*, 2210–2219.

Jiang, H. and Chang, C. K. (2008). Deriving Evaluation Metrics for Applicability of Genetic Algorithms to Optimization Problems. *proceedings of Conference on Genetic and Evolutionary Computation*, 1113–1114.

Kosters, W. A. and Kok, J. N. and Leiden, P. F. (1999). Fourier Analysis of Genetic Algorithms. *Theoretical Computer Science*, *229*, 143–175.

Last, M. and Eyal1, S. and Kandel, A. (2006). *Lecture Notes in Computer Science: Effective Black-Box Testing with Genetic Algorithms*. Springer-Verlag Berlin Heidelberg.

Leemis, L. H. and Park, S. K. (2005). *Discrete-Event Simulation: A First Course*. Prentice Hall.

Levin, S. and Yehudai, A. (2007). *Lecture Notes in Computer Science: Evolutionary Testing: A Case Study*. Springer Berlin/Heidelberg.

McMinn, P. and Holcombe, M. (2003). The state problem for evolutionary testing. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*, 2488-2497.

McMinn, P. (2004). Search-Based Software Test Data Generation: A Survey. *Software Testing, Verification and Reliability*, *14*(2), 105–156.

McMinn, P. and Binkley, D. and Harman, M. (2005). Testability Transformation for Efficient Automated Test Data Search in the Presence of Nesting. *Proceedings of the Third UK Software Testing Workshop (UKTest 2005)*, 165–182.

Michael, C. C. and McGraw, G. and Schatz, M. A. (2001). Generating Software Test Data by Evolution. *IEEE Transactions on Software Engineering*, *27*(12), 1085–1110.

Mitchell, B. S. and Mancoridis, S. (2002). Using Heuristic Search Techniques To Extract Design Abstractions From Source Code. *proceedings of Conference on Genetic and Evolutionary Computation*, 1375–1382.

Mitchell, M. (1996). *An Introduction to Genetic Algorithm*. Cambridge, MA: MIT Press.

Naudts, B. and Kallel, L. (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *4*, 1–15.

Nix, A. E. and Vose, M. D. (1992). Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, *5*(1), 79–88.

Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. *Proceedings of IEEE International Conference on Evolutionary Computation*, 798–803.

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach (6th edition)*. The McGraw-Hill Companies, Inc.

Rudolph, G. (1994). Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks*, *5*(1), 96–101.

Rudolph, G. (1996). Convergence of Evolutionary Algorithms in General Search Spaces. *proceedings of Congress on Evolutionary Computation*, 50–54.

Sareni, B. and Krahenbuhl, L. (1998). Fitness Sharing and Niching Methods Revisited. *IEEE Transactions on Evolutionary Computation*, *2*(3), 97–106.

Schoenauer, M. et al. (2007). Bridging the Gap between Theory and Practice. *MIT Press Journals - Evolutionary Computation*, *15*(4), iii–v.

Setnes, M. and Roubos, H. (2000). GA-Fuzzy Modeling and Classification: Complexity and Performance. *IEEE Transactions on Fuzzy Systems*, *8*(5), 509–522.

Sthamer, H. and Wegener, J. and Baresel, A. (2002). Using Evolutionary Testing to improve Efficiency and Quality in Software Testing. *Proceedings of the 2nd Asia-Pacific Conference on Software Testing Analysis and Review (AsiaSTAR2002)*, 50–54.

Suzuki, J. (1995). A Markov chain analysis on simple genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, *25*(4), 655–659.

Suzuki, J. (1995). A Further Result on the Markov Chain Model of Genetic Algorithms and Its Application to a Simulated Annealing-Like Strategy. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, *28*(1), 95–102.

Tracey, N. and Clark, J. and Mander, K. (1998). Automated Program Flaw Finding Using Simulated Annealing. *Proceedings of the International Symposium on Software Testing and Analysis*, 73–81.

Vieira, F. E. and Menezes, R. and Braga, M. (2006). Using genetic algorithms to generate test plans for functionality testing. *proceedings of Annual Southeast Regional Conference*, 140–145.

Vose, M. D. and Liepins, G. E. (1991). Punctuated Equilibria in Genetic Search. *Complex Systems*, *5*(1), 31–44.

Wegener, J. (2005). *Lecture Notes in Computer Science: Evolutionary Testing Techniques*. Springer-Verlag Berlin Heidelberg.

Xu, R. and Qian, L. and Jing, X. (2003). CMM-based software risk control optimization. *IEEE International Conference on Information Reuse and Integration (IRI)*, 499–503.