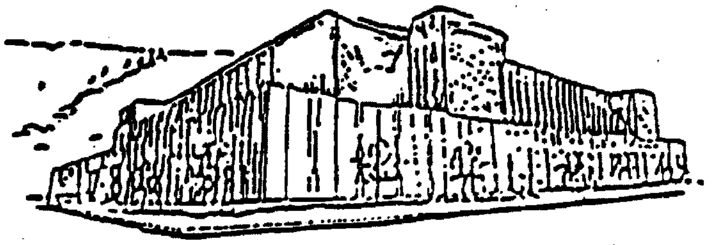
2000

Using Markov chain model to compare a steady-state and a generational GA

Cuixiu Zheng

The University of Montana



Using Markov Chain Model to Compare A Steady-state and A Generational GA

By

Cuixiu Zheng

Presented in partial fulfillment of the requirements

for the degree of

Master of Science

In Computer Science

The University of Montana-Missoula

May 2000

Approved by [illegible]

Chairperson

Dean, Graduate School [illegible]

Date 5-24-2000

Cuixiu Zheng, M.S., May 2000 Computer Science

Using Markov Chain Model to Compare A Steady-state and A Generational GAs

Director: Alden H. Wright [illegible]

Genetic algorithms were invented by Holland in 1960s and were developed and used through several decades. Genetic algorithms are methods for searching optimal or high quality solutions to a problem by applied genetic operators such as selection, crossover and mutation. Markov chain model of a GA is an intuitive method in simulating the process of a GA. In this paper, the Random heuristic Search model is reviewed. The steady-state and generational GAs are defined and their Markov chain models are constructed based on the Random Heuristic Search model. Some experimental results of running Markov chain simulations on both steady-state and generational GAs are compared in terms of the probability of obtaining at least one copy of the optimum by each generation and the expected waiting time to obtain one optimum.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank Dr. Wright for his continuous consult in this work and countless hours for reviewing this document. His comprehensive knowledge and clear explanation helped me better understand this subject. His academic attitudes deeply impressed and push me not only in this work, but also in the future. It is his soundless encouragement and support that made a big different in my second student life. I would like to thank Dr. Optiz. He gave me the first class on genetic algorithms. He had many other processing tasks when he took on this project. I'm so grateful for him in this project. Also I would like to thank Dr. Kayll. He checked the draft of this document word by word, corrected the spelling, grammar and the mathematical representation mistakes I had made. My thanks also go to my lovely five-year-old son, Huadian. He kept smiling, waiting and tried to understand while I buried my head in this work. As always, thank you, Mom and Dad, who give me the pressure and encouragement at the same time, who teach me never to quit at the half way. Without their help and support, I will not have finished this work. Finally, I would like to thank my husband, Jie, who is my biggest spirit support in my life.

CHAPTER I

INTRODUCTION

Genetic algorithms were invented by John Holland in 1960s and were developed and used through several decades. Genetic algorithms are methods for searching for optimal or high quality solutions to a problem. When searching a space of candidate solutions, a genetic algorithm only examines a small fraction of the possible candidate solutions. Although there are different implementations of genetic algorithms, they all have similar structure: Patterned after biological evolution, beginning with an initial population, the algorithm generates a sequence of populations. The next population is generated from the current population by means of genetic operators, which include selection according to fitness, crossover and mutation. At each step the better-fit individuals are selected from the current population as seeds for replacing a fraction of the current population to produce the next generation.

By modeling the evolutionary process as a mathematical model, we think about it as a computational process. Modeling a GA as a Markov chain provides a set of tools for describing the behavior of the GA. Measures of GA behavior include the probability of finding an optimal solution within a given number of steps as well as the time to convergence. Generally, different GAs combine exploitation and exploration in different ways. A GA that emphasizes exploitation rapidly converges toward high fitness individuals. A GA that emphasizes exploration will maintain a diverse population for a longer time. Selection adds exploitation to GA behavior. Mutation and crossover add exploration to GA behavior. The pressure of mutation and crossover balances the pressure of selection. Different genetic algorithms have different searching strategies. For example, in a steady-state GA, only a few elements of the population are replaced in every time step. The difference between parent generation and child generation is small. In the generational GA, a large number of elements of the population are replaced during every time step; the difference between parent generation and child generation is significant. How these factors affect the GA behavior is of great interest.

Following are two interesting and important attributes of GA behavior:

(1) The probability that the GA population will contain at least one copy of an optimum at generation k, where $k = 0,1,2,...$

(2) Expected waiting time to a population first containing a copy of the optimum.

This project is based on the context of previous work done by Dr. Wright and Yong Zhao [Zhao]. Beginning with understanding the Random Heuristic Search model algorithm and the Markov chain model of genetic algorithms, this project compared generational and steady-state GAs by running Markov chain simulations. Experiments were run on generational GA and steady-state GA algorithms, with various fitness functions and various genetic operators. The purpose is to better understand the behavior of the Markov chain model of GAs and gain insight into how genetic algorithms and GA operators affect the probability of finding an optimum and the expected time to converge. In this project, we defined steady-state and generational genetic algorithms and construct the Markov chain model for them based on Random Heuristic Search model. After running experiments for the performance of steady-state and generational GA on their Markov chain models, we verified and justified our hypothesis on the performance of steady-state and generational GA.

In order to be able to run larger models, the code to run the Markov chain simulation was implemented in C and tested using the GNU development system.

CHAPTER II

BACKGROUND

§2.1 Notation

A GA is a learning method by a set of GA operators that repeatedly recombine and mutate selected elements of the current population. Most commonly, members of a GA's population are encoded as bit strings. The advantage of bit string encoding is GA operators can easily manipulate populations.

We assume that members of a GA's population are encoded as binary strings. Let l be the length of the binary strings. Then $n = 2^l$ is the total number of possible strings. Let Ω be the set of length-l binary strings. We identify Ω with the set of integers in the interval $[0, n)$. In this paper, Z denotes the set of nonnegative integers, and R denotes the set of real numbers. Λ denotes the set of p_i, where $p_i \in R$, $0 \le p_i \le 1$, $\sum_{i \in \Omega} p_i = 1$. If $x, y \in \Omega$, then $x \oplus y$ is the bitwise EXCLUSIVE-OR of x and y, $x \otimes y$ is the bitwise AND of x and y, and $\bar{x}$ is the one's complement of x.

Let r be the size of the population. In the incidence vector representation of a population, $X =< X_0, X_1, X_2, \ldots, X_{n-1} >$, its nonnegative entry X_i is the number of times $i \in \Omega$ occurs in the population. The size of the population, r, is:

$$r = \sum_{i=0}^{n-1} X_i .$$

The corresponding probability vector representation x of X, denoted as

$$x = X/_r = < X_0/_r , X_1/_r , X_2/_r ,\dots, X_{n-1}/_r > .$$

So x_i is the probability that i is selected in population X.

§2.2 **GA operators**

There are many GA operators that have been used in GA applications. The most commonly used are *selection, crossover* and *mutation.* Selection is performed before *crossover* and *mutation.*

§2.2.1 **Selection**

Selection defines the strategy of how to choose elements in the current population for inclusion in the next generation. In order to evaluate the elements of a population, a *fitness function* $f : \Omega \to R^+$ is assumed. The fitness vector $\langle f_0, f_1, \dots, f_{n-1} \rangle$ is defined by $f_i = f(i)$ for all $i \in \Omega$. The fitter element has a higher fitness value.

A *selection scheme* $F : \Lambda \to \Lambda$ is a heuristic function where $F(x)_i$ denotes the probability that $i \in \Omega$ will be selected for the population after selection is applied to the population probability vector x. In a *proportional selection scheme*, the probability that an individual element i is selected is the ratio of its fitness to the total fitness of all elements of the current population. This defines the heuristic function F by

$$F(x)_i = \frac{f_i x_i}{\sum_{j\in\Omega} f_j x_j}.$$

For a size-r population X, $F(X/r)$ denotes the selection probability over a set of populations λX, whose population size can be any of λr, where $\lambda \in Z$.

For example:

A population vector: $X = <1,2,0,0>$.

A fitness vector: $f = <3,5,2,1>$.

The size of population: $r = 3$.

The probability distribution of $X: x = X/r = <1/3, 2/3, 0, 0>$.

The result of the proportional selection scheme: $F(X/r) = <3/13, 10/13, 0, 0>$.

§2.2.2 **Crossover**

Applying the *crossover operator* to two selected parent strings means copying selected bits from each parent string to produce two new strings. The choice of bits selected to produce new strings is determined by another bit string i, called a *crossover mask*. Let x, y be two selected parent strings. Applying i to x, y will produce two new strings:

$$(x \otimes i) \oplus (y \otimes \bar{i})$$

$$(y \otimes i) \oplus (x \otimes \bar{i}).$$

According the number of block of continuous 1's in i, we name three types of *crossover* operation as *one-point, two-point or uniform crossover.* The *crossover rate* c determines the probability of a nontrivial crossover. The probability that $i \in \Omega$ is chosen as the *crossover mask,* χ_i, is defined based on the type of *crossover* as follows:

$$\text{For one-point crossover, } \chi_i = \begin{cases} \dfrac{c}{l-1} & \text{if } i = 2^u - 1, \text{for } 1 \le u < l \\ 1 - \mathrm{c} & \text{if } i = 0. \end{cases}$$

$$\text{For two-point crossover, } \chi_{\mathrm{i}} = \begin{cases} \dfrac{c}{\binom{l}{2}} & \text{if } i = (2^{u-1} - 1) \oplus (2^{v-1} - 1), \text{ where } 0 \le v < u < l \\ 1 - \mathrm{c} & \text{if } i = 0. \end{cases}$$

$$\text{For uniform crossover, } \chi_{\mathrm{i}} = \begin{cases} \dfrac{c}{2^l} & \text{if } i \ne 0 \\ 1 - c + \dfrac{c}{2^l} & \text{if } i = 0. \end{cases}$$

For example:

Let $\begin{cases} u = 11101001 \\ v = 00001010 \end{cases}$ be two selected parent strings.

Crossover type	crossover mask	new strings
One-point	$00000111 = 2^3 - 1$	$\begin{cases} 11101010 \\ 00001001 \end{cases}$

Two-point	$00111110 = (2^6 - 1) \oplus (2^1 - 1)$	$\begin{cases} 11001011 \\ 00101000 \end{cases}$
Uniform	10011010	$\begin{cases} 10001000 \\ 01101011 \end{cases}$

A *crossover scheme* $C: \Lambda \to \Lambda$ is a heuristic function. $C(x)_i$ denotes the probability that i will be produced from the population probability vector x by crossover operation. This was shown in [Vose] to be:

$$C(x)_i = \sum_{u,v,k \in \Omega} x_u x_v \frac{\chi_k + \chi_{\bar{k}}}{2} \left[(u \otimes k) \oplus \left(v \oplus \bar{k} \right) = i \right]$$

where [expr] = 0 if expr isfalse; [expr] = 1 if expr is true.

§2.2.3 **Mutation**

Mutation insures that the population does not converge to a fixed string pattern. It adds diversity into the new generation. The mutation operator produces a new string from a single selected parent string by randomly flipping some bits of the parent string. The m*utation rate,* u, denotes the probability of each bit being flipped. The probability of a string $i \in \Omega$ being chosen as a mutation mask, denoted as μ_i, is:

$$\mu_i = (u)^{|i|}(1-u)^{l-|i|} \quad \text{where } i \in \Omega, \text{ and } |i| \text{ denotes the number of 1's in } i.$$

A *mutation scheme* $\mathrm{U}:\Lambda\to\Lambda$ is a heuristic function. $\mathrm{U}(x)_i$ denotes the probability that $i\in\Omega$ will be produced from the population probability vector x by the mutation operation. It has been shown to be [Vose]:

$$\mathrm{U}(x)_i = \sum_{u\in\Omega} x_u \mu_{u\oplus i}\,.$$

§2.3 **Random Heuristic Search Model**

Let F define the fitness heuristic scheme, let $M = U\circ C$ be the mixing scheme that depends on the mutation rate u, the crossover rate c, as well as the type of the crossover. The random heuristic scheme G of a simple GA is the composition of a mixing scheme M and a selection scheme F:

$$G = M\circ F = U\circ C\circ F\,.$$

For a size-r population X, the function $G\left(X/r\right)$ gives the next generation probability distribution over Ω, based on the current population X/r, as:

$$G\left(X/r\right) = M\circ F\left(X/r\right) = M\left(F\left(X/r\right)\right).$$

§2.4 **Description of steady-state GA and generational GA**

Here we provide pseudocode describing the two types of GA's we'll be considering.

§2.4.1 **The generational GA**

```
1.    Given an initial population $X$ of size r;
2.    Let $Y$ be empty population;
3.    For integer $k \leftarrow 1$ to $r$ do
          3.1    Select $y \in \Omega$ based on the probability distribution $G\left(X/r\right)$;
          3.2    Add element $y$ to population $Y$;
      Endfor
4.    Replace $X$ with $Y$;
5.    Go to step 2;
```

§2.4.2 **The steady-state GA**

```
1.    Given an initial population $X$ of size $r$;
2.    Select $y \in \Omega$ based on the probability distribution $G\left(X/r\right)$;
3.    Let $Y$ be a population consisting of a single element $y$. That is
      $Y_y = \begin{cases} 1 & \text{if } i = y \\ 0 & \text{if } i \neq y; \end{cases}$
4.    Delete the worst element of $X + Y$ to form $Z$;
5.    Replace $X$ with $Z$;
6.    Go to step 2;
```

§2.5 **Markov chain model**

A Markov chain is a model of a stochastic system that moves from state to state, and which satisfies the Markov property. *A process satisfies the Markov property* *if the probability of being in a given state of the process at the time step $t+1$ depends only on the state at time step t and not on the state at time steps* $1,2,\ldots,t-1$.

Most genetic algorithms satisfy the Markov property. A GA that satisfies the Markov property can be modeled as a Markov chain process, where the states are the populations [Nix & Vose]. The genetic algorithm moves from one population to another. The population of the genetic algorithm at any time step is dependent only on the immediately preceding population. The selection, crossover and mutation schemes used determine the probabilities for the next population. The Markov chain model can be used to evaluate alternative ways of doing selection, crossover and mutation.

For a GA, the number N of all possible size-r populations drawn from Ω, corresponding to the number of possible states in the Markov chain is: [Nix & Vose]

$$N = \binom{n+r-1}{r}.$$

In other words, for a GA, which has N possible populations, the transition probability is $N \times N$. Notice that N is simply the number of multiset of size-r population chosen from Ω.

We assume an ordering of the populations, and we identify them with the integers $[0, N)$.

In an $N \times N$ Markov chain transition matrix P, the XY entry P_{XY} of P is the transition probability that the system goes from state X into state Y, where $X, Y \in \{0,1,2,...,N-1\}$. For any given $X \in \{0,1,2,...,N-1\}$, $P_{X0} + P_{X1} + ... + p_{X(N-1)} = 1$. This reflects the fact that the system in a given state X will be in one of the possible N states at the next time step.

One of the special classes of Markov Chains is the class of *absorbing chains*. Let $P = (P_{XY})$ be an $N \times N$ transition probability matrix. A state X of the Markov chain is an *absorbing state* if $P_{XX} = 1$. The Markov chain represented by P is an *absorbing chain* if: (1) it contains at least one absorbing state, and (2) from every state it is possible to reach an *absorbing state*. Let m be the number of absorbing states, and then, by rearranging rows and columns, P can be written as

$$P = \begin{bmatrix} I & 0 \\ R & Q\_transient \end{bmatrix}.$$

where I is the $m \times m$ identity matrix, R is the $(N-m) \times m$ transition matrix from transient states to absorbing states, and $Q\_transient$ is the $(N-m) \times (N-m)$ transition matrix between transient states.

Using the next theorem, we can predict all future probability distributions over all possible N populations at each time step determined by the initial probability distribution over all possible N populations drawn from Ω.

Theorem I *Let P be the transition matrix of a Markov process, let $Z^{(0)}$ be the initial probability distribution row vector over all states of the Markov process, and let $Z^{(t)}$ denote the probability distribution vector at time step t. Then the probability distribution vector at time step $t+1$ is:* [Isaacson & Madsen]

$$Z^{(t+1)} = Z^{(t)}P = Z^{(t-1)}P^2 = \ldots = Z^{(2)}P^{t-1} = Z^{(1)}P^t .$$

§2.6 **Selection of an initial population**

Assume that the GA population is initialized uniformly at random. By the multinomial theorem, the probability of GA population X at the time step 0, denoted by $P(X@0)$ is:

$$P(X@0) = \frac{r!}{X_0!X_1!X_2!\ldots X_{n-1}!} * \left(\frac{1}{n}\right)^r .$$

Let $X^0, X^1, X^2, \ldots, X^{N-1}$ be the set of all possible populations. Then the initial probability distribution over populations is:

$$\left[P\left(X^0 @ 0\right), P\left(X^1 @ 0\right), P\left(X^2 @ 0\right), \ldots, P\left(X^{N-1} @ 0\right)\right].$$

§2.7 **Probability of obtaining at least one copy of an optimum at time step t**

Let Q be a transition matrix. The XY entry of Q is the transition probability that the GA will be in state Y at time t given that it was in state X at time step $t-1$. Then the probability of the GA being in state Y at time t, denoted by $P(Y@t)$, is:

$$P(Y @ t) = \sum_{X} P(X @ (t-1)) Q_{XY} = \sum_{X} P(X @ 0) \left(Q^{t}\right)_{XY} .$$

Let J be the set of populations that contains at least one copy of an optimal element (with highest fitness value). Then the probability that the GA is in one of the states in J at time step t, is:

$$P(J @ t) = \sum_{j \in J} P(j @ t).$$

§2.8 **Probability of obtaining at least an optimum by each time step**

Suppose $Z^{(0)}$ is the initial probability distribution over all possible populations and $Z^{(t)}$ is the probability distribution at time step t. Let us arrange the set of all possible populations into two subsets J and K. Let J be the set of populations that contain an optimum, and let K be the set of populations that do not contain an optimum.

We can arrange the initial probability distribution in the form $Z^{(0)} = \left(Z_J^{(0)} \mid Z_K^{(0)}\right)$, where $Z_J^{(0)}$ is the probability distribution over the population set J of $Z^{(0)}$, and $Z_k^{(0)}$ is the probability distribution over the population set K of $Z^{(0)}$. If 1 denotes the all ones vector, then starting from $Z^{(0)}$, the probability that we are in J at time step 0 is $Z_J^{(0)}1$, and the probability that we are in J for the first time at time step t is $Z_K^{(0)} Q^{t-1} R 1$. [where 1 is all ones vector].

Thus the probability that we have obtained at least an optimum by time step t is:

$$\left(Z_K^{(0)} + Z_K^{(0)}R + Z_K^{(0)}QR + Z_K^{(0)}Q^2R + \ldots + Z_K^{(0)}Q^{t-1}R\right)1$$
$$= \left(Z_J^{(0)} + Z_K^{(0)}\left(\sum_{i=0}^{t-1} Q^i\right)R\right)1.$$

§2.9 **Transition probability**

(1) Generational GA

Let X and Y be populations of size r. We now show how to compute the transition probability $P(X,Y)$. Recall that Y_i is the number of copies of element $i \in \Omega$ in Y. Let p_i be the probability that $i \in \Omega$ is selected in the next generation, which is encapsulated in the G function via $p_i = G\left(X/r\right)_i$. The probability of selecting the first element 0 of Ω Y_0 times is $\binom{r}{Y_0} p_0^{Y_0}$. The probability of selecting the second element 1 of Ω Y_1 times is $\binom{r-Y_0}{Y_1} p_1^{Y_1}$ and so on until finally which we have probability of selecting the last element $n-1$ of Ω Y_{n-1} times: $\binom{r-Y_0-Y_1-Y_2-\ldots-Y_{n-2}}{Y_{n-1}} p_{n-1}^{Y_{n-1}}$. So given the probability distribution $p =< p_0, p_1, p_2, \ldots, p_{n-1} >$, the probability from X to Y after r independent samples is:

$$\binom{r}{Y_0}\binom{r-Y_0}{Y_1}\binom{r-Y_0-Y_1}{Y_2}\cdots\binom{r-Y_0-Y_1-Y_{n-2}}{Y_{n-1}}(p_0)^{Y_0}(p_1)^{Y_1}(p_2)^{Y_2}\cdots(p_{n-1})^{Y_{n-1}}.$$
$$= r!\prod_{i\in\Omega}\frac{(p_i)^{Y_i}}{(Y_i)!}.$$

Because $p_i = G\left(X/r\right)_i$, $P(X,Y)$, the probability of forming a size-r population Y from X, is:

$$P(X,Y) = r!\prod_{i\in\Omega}\frac{\left(G\left(X/r\right)_i\right)^{Y_i}}{(Y_i)!}.$$

(2) Steady-state GA:

Let A be a size-r population, B be a size-k subpopulation of A. Then $\rho_A(B)$ denotes the probability of choosing B from A without replacement and without regard fitness, which is given by the *multiple hypergeometric probability distribution*:

$$\rho_A(B) = \frac{\prod_{j\in\Omega}\binom{A_j}{B_j}}{\binom{r}{k}}.$$

where A_j is the number of times that $j\in\Omega$ is chosen in A and B_j is the number of times that $j\in\Omega$ is chosen in B.

Let Y be the population consisting of the single element y. In other words,

$$Y_i = \begin{cases} 1, \text{if } i = y \\ 0, \text{if } i \neq y. \end{cases}$$

In step 2 and 3 of the steady-state GA paradigm (§2.4.2), the transition probability from X to Y, $P(X,Y)$ is described by the Random Heuristic Search model. That is defined as:

$$
\begin{aligned}
&P(X,Y) \\
&= 1!\prod_{i\in\Omega}\frac{\left(G\left(X/r\right)_i\right)^{Y_i}}{(Y_i)!} \\
&= \frac{G\left(X/r\right)_y^{Y_y}}{Y_y!} \\
&= G\left(X/r\right)_y.
\end{aligned}
$$

In step 4 of the steady-state GA paradigm (§2.4.2), a population of Z is formed by deleting the worst element of $X+Y$ based on fitness value. Z could have multiple choices if the fitness function is not an injective function. Let $\beta_r(X+Y)$ be the set of subpopulations consisting of the best r elements of $X+Y$. The conditional probability of choosing Z from X, given that population Y was chosen at step 2 and $Z \in \beta_r(X+Y)$, is

$$\frac{\rho_{X+Y}(Z)}{\rho_{X+Y}(\beta_r(X+Y))}.$$

Consider all the possible choices of Y; the Markov chain transition probability from X to Z is:

$$P(X,Z) = \sum_{y\in\Omega}[Z \in \beta_r(X+Y)]P(X,Y)\frac{\rho_{X+Y}(Z)}{\rho_{X+Y}(\beta_r(X+Y))}.$$

where $[\text{expr}] = 0$ if expr isfalse; $[\text{expr}] = 1$ if expr is true.

§2.10 Expected Waiting Time to obtain one copy of optimal element

Another question addressed in this project is the expected waiting time until a population first contains a copy of the optimum. To answer this question, let us treat each state whose population has at least one optimal element as an absorbing state. In other words, we redefine the transition probabilities so that $P(X,Y)=0$ if X contains one copy of the optimum and $X \neq Y$; $P(X,Y)=1$ if X contains one copy of the optimum and $X = Y$. Obviously, if the population starts with an absorbing state, the desired expected waiting time is 0. Now let us consider the situation in which the GA starts from a transient state. For an absorbing Markov chain, let Q be the transition matrix between transient states; then the ij entry of Q^n denotes the probability that the GA goes from transient state i to transient state j in exactly n steps.

We need a definition and two facts from [Isaacson & Madsen]

Definition 1 *A matrix Q with elements Q_{ij} is called substochastic if $q_{ij} \geq 0$ for all i and j, and if $\sum_j Q_{ij} \leq 1$ for all i.*

Lemma 1 *Let Q be the substochastic matrix corresponding to transitions among the transient states of a finite Markov chain. Then*

$I + Q + Q^2 + Q^3 + ... = (I - Q)^{-1}$ exists.

Lemma 2 *If 1 denotes a column vector of ones, then* $(I-Q)^{-1}\mathbf{1}$ *is a column vector, in which the* ith *entry is the expected absorption time given that the GA starts from transient state* i.

CHAPTER III

HYPOTHESES and METHODOLOGY

§3.1 Experimental parameters

Experiments were set up with the following choices of parameters for both the steady-state and generational GA:

(1) Fitness function alternatives:

- The counting-one fitness function, defined by

  $F(i) = |i|$ for $i \in \Omega$, $|i|$ denotes the number of 1's in i.

  This is an example of an easy fitness function.

- The deceptive function, defined by

  $F(i) = \begin{cases} l - |i| & \text{if } |i| \neq l \\ l+1 & \text{if } |i| = l \end{cases}$ for $i \in \Omega$, $|i|$ denotes the numbers of 1's in i.

  This is an example of a hard fitness function.

(2) Type of crossover alternatives:

One point crossover vs. uniform crossover.

(3) Crossover rate alternatives:

The crossover rate varied from 0 to 1.0.

§3.2 Experimental hypotheses

In this project, the GA performance is measured by the probability of obtaining at least one copy of an optimum by each generation (i.e. accumulative probability, as described in §2.8), and the expected waiting time to obtain one copy of an optimum (as described in §2.10). In the generational GA, at each time step, after r fitness evaluations the current population (i.e. current generation) will switch to the next population (i.e. next generation). But in the steady-state GA, at each time step, there is only one fitness evaluation, that is one generation means one fitness evaluation. The objective of the experiments is to assess the difference between the performance of a steady-state GA and a generational GA. We run experiments under same experimental parameters to compare the performance of steady-state GA and generational GA. Also we run experiments by varying one of the parameters (§3.2) to get insight into the effects of type of fitness function, the type of crossover and increasing crossover rate on the performance of each GA.

We believe the following hypotheses:

(I) The steady-state GA has a higher probability of containing one copy of the optimum than the generational GA by each generation. (Note that the Markov chain used here is modified to make populations containing the optimum into absorbing states.)

(II) The steady-state GA has a shorter expected waiting time to find the optimum than the generational GA. (Note that the Markov chain used here is modified to make populations containing the optimum into absorbing states, i.e. we use transition matrix between transient states.)

In other words, the steady-state GA has better performance than the generational GA on both an easy problem and a hard problem.

(III) Uniform crossover as opposed to one-point crossover should result in a shorter average waiting time to find a population containing an optimum.

(IV) Increasing the crossover rate will result in a shorter average waiting time to find a population containing an optimum.

Our experiments are designed to test these hypotheses.

§3.3 **Experimental methodology**

All experiments will start with the same initial probability distribution over populations (§2.6.1). The same fitness, proportional selection, crossover and mutation schemes will be used for both the steady-state and generational GAs (§2.4) at every run.

To test hypothesis (I) and (II), the experimental settings are:

(a) The counting-ones fitness function

Uniform crossover

Crossover rate: 0.8

(b) The deceptive fitness function

Uniform crossover

Crossover rate: 0.8

The deletion step in the steady-state GA always deletes the worst element before the new generation is formed. In other words, the process always keeps the best and better elements. So the probability of obtaining one copy of an optimum in the steady-state GA **at** each time step is equal to the probability of obtaining one copy of an optimum **by** each time step.

But in the generational GA, forming a new generation is performed by Random Heuristic Search, which does not guarantee that the optimum in the current generation is kept in the next generation. So in order to get the probability of obtaining one copy of an optimum by each time step, we can modify the original transition matrix P by: for any population X that contains an optimum, $P_{XX} = 1, P_{XY} = 0$, if $X \neq Y$; otherwise P_{XY} remains unchanged.

To test hypothesis (III), the experimental settings are:

(a) The counting-ones fitness function

Uniform crossover

Crossover rate: 0.8

(b) The counting-ones fitness function

One-point crossover

Crossover rate: 0.8

To test hypothesis (IV), the experimental settings are:

(a) The counting-ones fitness function

Uniform crossover

Crossover rate: from 0 to 1.0 in steps of 0.1

(b) The deceptive fitness function

Uniform crossover

Crossover rate: from 0 to 1.0 with 0.1 difference every run

§3.4 **Maple vs. C**

Maple is a general-purpose symbolic language. It has a set of rich procedures to complete computational tasks. Also Maple is an interpreted language, which can have dynamic value and type binding. Maple program cans be run using either numerical or symbolic parameters. It is relatively easy to write a program with a large amount of computations in Maple.

Under dynamic binding, type checking is done during run-time by inserting extra code into the program to detect impending errors, which takes up time and space and is inefficient. So, Maple programs turn out to be slow.

Compiled languages use a type system, where type checking is done during compile time. A compiler can infer from the source code that a function f should be applied to a correctly typed operand a each time the expression $f(a)$ is executed. In this case, the program is checked and data layout is done statically as far as possible during compilation.

The Markov Chain model involves a large amount of matrix computation. The dimension of a matrix, in particular a transition probability matrix, grows exponentially with the length of the binary strings. Using a compiled language to do this computation work will speed processing and make simulations of the Markov Chain model more practical.

CHAPTER IV

RESULTS

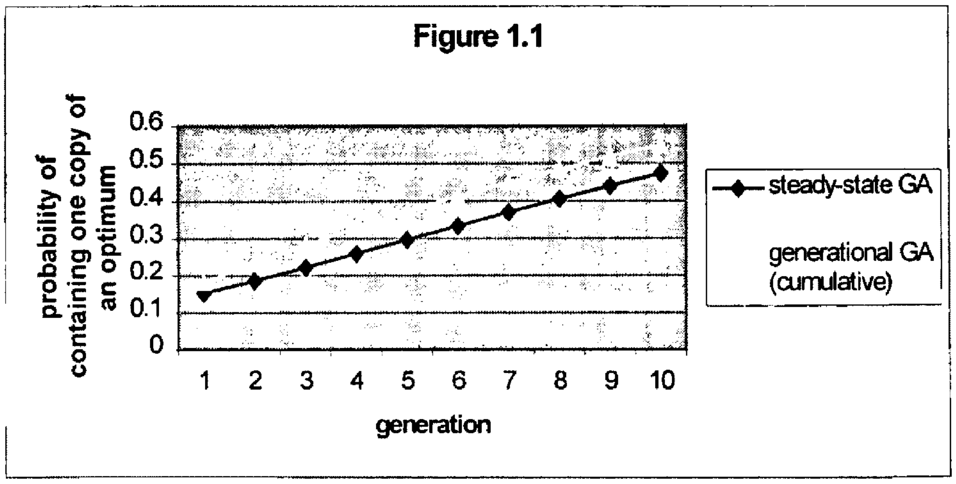
1. The result of testing hypothesis (I)

We set $l = 4$, $r = 2$, $u = 0.1$ and $c = 0.8$.

1.1 The probability of obtaining at least one copy of an optimum by the given generation using counting-ones fitness function

Table 1.1:

Generation	Steady-state GA	generational GA (by each generation)
1	0.152475	0.181734
2	0.186533	0.234430
3	0.222386	0.282879
4	0.259154	0.328306
5	0.296237	0.371085
6	0.333195	0.411358
7	0.369687	0.449218
8	0.405446	0.484762
9	0.440256	0.518094
10	0.473951	0.549324



1.2 The probability of obtaining at least one copy of an optimum by the given generation using deceptive fitness function

Table 1.2:

Generation	Steady-state GA	Generational GA (by each generation)
1	0.141902	0.161437
2	0.15242	0.188301
3	0.158911	0.21001
4	0.163428	0.229196
5	0.166805	0.246863
6	0.169447	0.26348
7	0.171577	0.279305
8	0.173333	0.294497
9	0.174803	0.309163
10	0.176052	0.323378

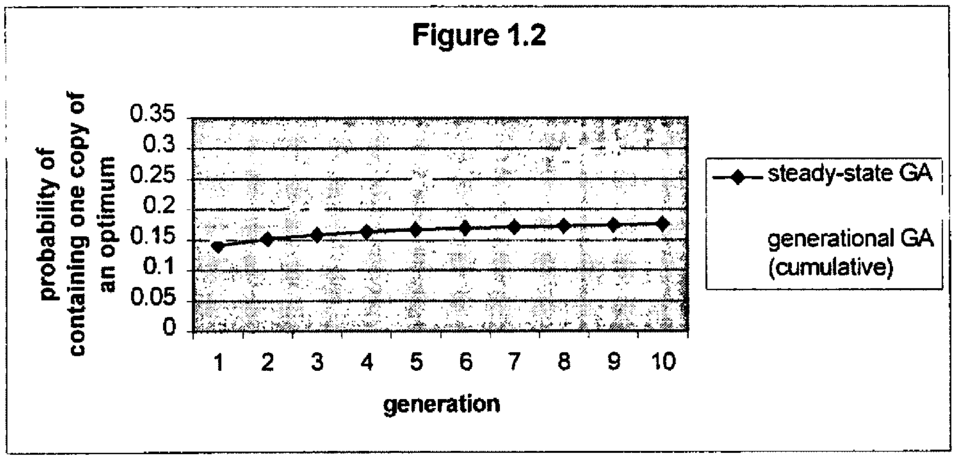


Figure 1.1 and Figure 1.2 show that the steady-state GA has a lower probability of containing one copy of an optimum than the generational GA on both easy and hard problems, which is in conflict with our hypothesis (I). We now think it is because the generational GA is more explorative than the steady-state GA, which retains more diversity than the steady-state GA. When the GA starts with a

transient state, the steady-state GA is more likely to remain in the given transient state; but the generational GA is more likely to change to different states (including the states containing an optimum). So starting from the same initial population distribution, in generational GA, the probability that we have seen an optimum by each time step is higher opposed to the steady-state GA.

2. The result of testing hypothesis II, the average expected waiting time of obtaining one copy of an optimum

Table 2: we set $l = 4$, $r = 2$, $u = 0.1$ and $c = 0.8$.

GA	Average EWT / counting-ones fitness	average EWT / deceptive fitness
steady-state	8.5	4631.4
Generational	15.5	47.2

Table 2 shows the steady-state GA has shorter expected waiting time than the generational GA on the easy problem, which consistent with our hypothesis (II).

But table 2 also shows the steady-state GA has a longer expected waiting time than the generational GA on the hard problem, which is again in conflict with our hypothesis (II). The deceptive fitness function creates a hard situation for GAs. So generally, the average expected waiting time is longer when the deceptive function is applied compared with applying the counting-one fitness function. When deceptive fitness function and low mutation rate are applied, generational GA's more exploration retains larger diversity of populations. So the generational GA is easier to recover from local optimum than the steady-state GA, which results that generational GA has shorter average waiting time until containing the first optimum. When string length increases, this result is likely to be more significant.

3. The result of testing hypothesis III on easy problem, the average expected waiting time of obtaining one copy of an optimum

Table 3: We set $l = 3$, $r = 3$, $u = 0.001$, $c = 0.8$.

GA	average EWT/uniform crossover	average EWT/one-point crossover
steady-state	325.6	315.6
generational	365.7	357.4

Theoretically, the uniform crossover has no position bias, which makes larger changes easier than one-point crossover. This higher changing rate would seem to results in higher probability of producing the first population containing an optimum. But Table 3 is in conflict with our hypothesis (III). The reason remains unclear.

4. The result of testing hypothesis IV on easy problem, the average expected waiting time of obtaining one copy of an optimum

Table 4: We set $l = 3$, $r = 3$, $u = 0.1$.

crossover rate	average EWT / steady-state GA	average EWT / generational GA
0.0	375.3	425.8
0.1	367.7	414.3
0.2	360.6	404.4
0.3	353.9	395.9
0.4	347.6	388.4
0.5	341.7	381.7

0.6	336.0	375.8
0.7	330.7	370.5
0.8	325.6	365.7
0.9	320.8	361.3
1.0	316.2	357.3

Table 4 shows the effects of increasing crossover rate. Crossover adds exploration to GAs, which results in higher probability of making a transition from one state to a different state. Larger crossover rate means larger changes more easily, so increasing crossover rate speeds up the conversion to an optimum. Table 4 confirms our hypothesis (IV).

CHARPTER V

CONCLUSIONS

In this paper, the Random Heuristic Search model is reviewed. The steady-state GA and the generational GA are defined. The Random Heuristic Search model is extended to both steady-state and generational GAs. The Markov chain models of both steady-state and generational GAs are described and applied to evaluate their performance in terms of the probability of obtaining an optimum at each generation and expected time to obtain at least one copy of an optimum. The effect of the crossover operator is investigated. Some experimental results are presented on a simple fitness function and a hard fitness function.

The experimental results of Markov chain simulation on GAs show that the generational GA has higher probability to obtain the optimum by each generation than the steady state GA. We think that this shows the importance of retaining the diversity of populations. The results verify that a higher crossover rate will result in a shorter expected waiting time. The results also show that deceptive fitness function is difficult for a GA to optimize.

It would be nice to see what the effects of other GA operators (e.g. mutation) and parameters (e.g. population size, n; the length of binary string; l) have on the GA performance. The experiment on whether the steady-state GA with a high mutation rate and a high crossover rate is equivalent to the generational GA with a lower mutation rate and a lower crossover rate could be run. The size of the Markov chain matrix grows exponentially with the length of the binary strings. Further work is needed in order to scale up to more realistic values of n and l. One solution could be to reduce the size of Markov chain matrix of a GA by lumping the states containing the optimum into one single state and/or lumping the similar states no containing the optimum into one single state without significant loss of accuracy.

REFERENCES

[Zhao] Yong Zhao (1999). *Markov chain Models of Genetic Algorithms*, Masters thesis. University of Montana.

[Tom M. Mitchell] Tom M. Mitchell (1997). *Machine Learning.* 249-270. The McGraw-Hill Companies, Inc.

[Vose] Mark D. Vose (1999). *The Simple Genetic Algorithms, Foundation and Theory.* The MIT Press. Cambridge MA.

[Melanie Mitchell] Melanie Mitchell (1996). *An introduction to Genetic Algorithms.* The MIT Press.

[Isaacson & Madsen] Chris Rorer & Howard Anton (1977). *Applications of Linear Algebra, Markov chains.* John Wiley & Sons.

[Norris] JR. Norris (1997). *Markov chains.* Cambridge University Press.

[De Jong] Kenneth A. De Jong, William M. Spears, Diana F. Gordon. *Using Markov chains to Analyze GAFOs.* In L. Darrell Whitley and M. D. Vose, eds. Foundations of Genetic Algorithms.3, San Francisco, CA, Morgan Kaufmann Publishers.

[Isaason & Madsen] Isaason D. and Madsen R. (1976). *Markov Chains Theory and Applications*. John Wiley & Sons, New York.

[Nix & Vose] A. E. Nix and M. D. Vose (1992). *Modeling Genetic Algorithms with Markov Chains.*