SIMULTANEOUS ADAPTATION OF MULTIPLE GENETIC ALGORITHM PARAMETERS USING FUZZY LOGIC CONTROLLERS

by

ABDULLATIF SALEH NASSER GHALLAB

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

UNIVERSITI SAINS MALAYSIA

2010

ACKNOWLEDGEMENTS

This thesis would not have been possible without help of others who I would to thank. First, and for most, I thank Allah for all his blessings and guidance.

I would like to thank my supervisor Associate Prof. Dr. Ahamad Tajudin bin Khader for his supervision, encouragements, guidance, constructive, and for all of his help during my research work and preparation of this thesis. Much appreciation to my co-supervisor Associate Prof. Dr. Azman bin Samsudin for his help and support during my Ph.D. study. And, also my thanks for Prof. Dr. Francisco Herrera for the discussion and sharing of his knowledge with me.

I express my special thanks to School of Computer Sciences for all facilities and support to conduct this research. My sincere thanks and love to my parents for their prayers, loves and cares. I am deeply thankful to my closest friends and my pillars of support, Hasan Salim, Abdulqader, Abdulhameed, Rais, Mahfuz and Majed for their help and support during my thesis preparation. I am grateful to my family for their time and support to finish this research.

TABLE OF CONTENTS

CHAPTER 3 - SIMULTANEOUS ADAPTATION OF MULTIPLE GENETIC ALGORITHM PARAMETERS USING FUZZY LOGIC CONTROLLERS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ABF	Average of Best Fitness
AGA	Adaptive Genetic Algorithm
ANE	Average Numbers of Evaluations
ANG	The Average Number of Generations
ARGAF	Fuzzy Adaptive Real-coded GA
CoGA	Cooperative Genetic Algorithms
$Conv_m$	Convergence measure
DGA	Deterministic Genetic Algorithm
DPGA	Dynamic Parametric GA
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategy
EER	Exploration and Exploitation Relationship
FAGA	Fuzzy Adaptive Genetic Algorithm
FAGA 1	Fuzzy Adaptive Genetic Algorithm 1
FAGA 2	Fuzzy Adaptive Genetic Algorithm 2
FGA	Fuzzy Genetic Algorithm
FL	Fuzzy Logic
FL C	Fuzzy Logic Controller
FRGA	Fuzzy Reduction GA
GA	Genetic Algorithm
GP	Genetic Programming
GD	Genotypical Diversity

$\mathbf{Gen_{P_c}}$	The number of crossover per generation
$\mathbf{Gen_{P_m}}$	the number of mutations per generation
HS	Harmony Search
SAMGAP	Simultaneous Adaptation of Multiple Genetic Algorithm Parameters
NABF	Number of Achieved Best Fitness
NFLT	No Free Lunch Theorem
PD	Phenotypical Diversity
SAGA	Self-Adaptive Genetic Algorithm
Std Dev	Standard Deviation
TSP	Traveling Salesman Problem

LIST OF SYMBOLS

P_c	Crossover rate
P_m	Mutation rate
N	Population size
l	The string length of the chromosome
T	Maximum number of generations
t	The number of current generation
P_m^i	Mutation probability for all the genes in bitstring
f_i	The chromosome's fitness
$f_{\max}$	Maximum of best fitness
$\bar{f}$	Average of best fitness
$\Delta f_{avg}(t-1)$	The change of average fitness for last generation (t-1)
Δf_{avg}	The change of average fitness in generation for current generation
$\Delta m(t)$	The change of mutation rate
$\Delta c(t)$	The change of crossover rate
$Pc(t)$	The crossover rate for generation t
$Pm(t)$	The mutation rate for generation t
$\Delta Pc(k+1)$	The change of crossover rate in the (k+1)th generation
$\Delta Pm(k+1)$	The changes of mutation rate in the (k+1)th generation
FLC_N	Fuzzy logic controller for adaptive population size
FLC_{P_m}	Fuzzy logic controller for adaptive mutation rate
FLC_{P_c}	Fuzzy logic controller for adaptive crossover rate
f_{Best}	The best fitness of the current population

f_{Avg}	The average fitness for the current population
$f_{Average}$	The average distance of the chromosomes in the population from the best one
f_{Maxt}	The maximum distance of the chromosomes in the population from the best one
f_{Worst}	The minimum distance of the chromosomes in the population from the best one
Gen_{P_m}	The number of mutations per generation
L	Length of chromosome
$Conv_m$	Convergence measure
$f_{CurrentBest}$	The fitness of the current best element found so far
$f_{LastBest}$	The fitness of the best element found before the last G generations
Gen_{P_c}	The number of crossover per generation
REQ_N	Priority value of firing FLC_N (FLC of population size)
REQ_{Pm}	Priority value of firing FLC_{Pm} (FLC the mutation rate)
REQ_{Pc}	Priority value of firing FLC_{Pc} (FLC of crossover rate)
PT	Priority table of firing FLCs
$REQ_{parm}(t)$	Priority value of calling any FLCs for any parameter Parm at generation t
FLC_{Parm}	FLC for adapt any parameter Parm
(t)	Current generation t
$(t-1)$	Last generation t-1
f_1	Test function 1 - sphere model
f_2	Test function 2 - the generalized Rosenbrock function

f_3	Test function 3 - the Quartic function with noise
f_4	Test function 4 - generalized Rastrigin function
f_5	Test function 5 - Travel Salesman Problem
n	The dimension of the test functions
L	The lower bound of the test function of interval
H	The upper bound of the test function of interval
$f_i(\vec{x})$	The value of test function fi
$fi(x^*)$	Global fitness of function fi
$gauss$	Gaussian noise for test function 3
A	Variable control of the amplitude of the modulation for test function 4
w	Variable control of the frequency of the modulation for test function 4
π	*Pi* value (*22/7*)
Ci,j	The cost of traveling from city *i* to city *j*
χ_{online}	Online performance
$\mathcal{S}$	The search strategy
e	The environment of search strategy
$f_e(t)$	The objective function
$\chi_{offline}$	Offline performance
$f_e^*(t)$	the value of best function obtained up to time t

PENYESUAIAN SERENTAK PARAMETER ALGORITMA GENETIK BERBILANG MENGGUNAKAN PENGAWAL LOGIK KABUR

ABSTRAK

Kajian ini bertujuan untuk mereka bentuk satu kaedah mudah suai dalam talian bagi mengawal parameter Algoritma Genetik. Kecekapan Algoritma Genetik menuntut keseimbangan yang sepadan yang perlu sentiasa dikekalkan antara tinjauan dan eskploitasi yang bergantung pula kepada penentuan beberapa parameter. Parameter ini tidak bebas dan memiliki interaksi yang rumit antara satu sama lain ketika larian. Sekiranya interaksi antara parameter-parameter yang telah dimudah suai itu tidak dihiraukan atau memudah suai satu parameter sahaja, ini boleh mengakibatkan impak yang buruk terhadap parameter-parameter lain yang berkaitan. Dengan itu, pencapaian Algoritma Genetik akan menjadi tidak cekap. Tetapi sebahagian besar alternatif yang ada tidak dapat menyelesaikan masalah ini dengan berkesan. Teknik Algoritma Genetik Kabur yang di mudah suai telah digunakan untuk kawalan parameter, tetapi masih terdapat beberapa kelemahan. Usaha ini menyumbang kepada reka bentuk Algoritma Genetik Mudah suai Kabur yang bersifat teguh. Ia memperkenalkan suatu kaedah Algoritma Genetik Mudah suai Kabur yang baru dan ia berasaskan kepada tiga pengawal logik kabur. Kaedah ini mengawal beberapa parameter secara serentak dengan pertimbangan dalam talian untuk ia saling bergantungan semasa larian. Ia menyesuaikan tiga parameter strategik utama Algoritma Genetik, iaitu, saiz populasi, kadar mutasi dan kadar pindah silang.

Kesahihan kaedah ini tunjnk dengan satu siri eksperimen ke atas satu set masalah ujian yang piawai yang sering digunakan untuk menilai teknik Algoritma Genetik. Hasil yang bersifat empirik menunjukkan bahawa kaedah yang disarankan berjaya mengekalkan nilai-nilai mudah suai yang baik bagi parameter bersama dengan tahap pencapaian yang baik.

SIMULTANEOUS ADAPTATION OF MULTIPLE GENETIC ALGORITHM PARAMETERS USING FUZZY LOGIC CONTROLLERS

ABSTRACT

This study aims at designing an online adaptive method to control multiple parameters of the Genetic Algorithm. The efficiency of Genetic Algorithm requires maintaining an appropriate balance between exploration and exploitation, which in turn greatly depends on the settings of several parameters. The parameters are not independent and have complex interactions with each other during a given run. Ignoring the interaction between the adapted parameters or adapting one single parameter may have negative impact on the other related parameters, resulting in poor performance. However, most of the available alternatives cannot solve this problem effectively. Fuzzy Adaptive Genetic Algorithm techniques have been used recently for parameter control, but still suffer from some defects. This work contributes towards the design of a robust Fuzzy Adaptive Genetic Algorithm. It presents a new Fuzzy Adaptive Genetic Algorithm method based on three fuzzy logic controllers. This method controls multiple parameters simultaneously with online consideration for their interdependencies during a run. It adapts three main strategic parameters of the Genetic Algorithm, namely, population size, mutation, and crossover rates. The validity of the proposed work is illustrated with a series of experiments on a set of standard test problems often used to evaluate Genetic Algorithm techniques. The empirical results indicated that the proposed method maintains good adaptive values of parameters with better performance.

CHAPTER 1

INTRODUCTION

1.1 Problem Overview

Genetic Algorithms (GAs) are one of the Evolutionary Algorithms (EAs) paradigms which have been considered as a serious contender to solve difficult optimization problems in many fields in the past three decades (F. G. Lobo, 2007).

GAs consists of five key components: a genotype format that specifies how genetic information is represented in a data structure; a development scheme which maps that information into a phenotypic design; a fitness function that assigns a fitness value to each phenotype; a set of genetic operators that modifies and replicates the genotypes from one generation to the next; and a set of evolutionary parameters such as population size (N), mutation rate (P_m), crossover rate (P_c) and selection pressure that governs how evolution runs (Goldberg, 1989).

The parameters of GA determine the general context for evolution and the quantitative details of how the genetic operators work (Fernando and David, 2004). The performance of GA is highly affected by the strategy of parameters control (Furutani et al., 2007). The values of the GA's parameters are problem dependent. Therefore, the flexibility in applying GAs to optimize general real-world problems optimally needs an appropriate set of GAs parameters values (Goldberg, 2003). One of the major issues for

GAs is to find the optimal parameters, especially, population size, mutation rate and crossover rate (Smith, 2003). The values of these parameters determine the algorithm's behavior in finding an optimal solution during the exploitation and exploration of the search space. These parameters interact with each other and affect the performance of GA. These interdependencies make the operation of controlling parameters a very long standing challenge (Eiben et al., 2007).

In general, there are two major approaches of parameter settings: parameter tuning and parameter control (Eiben et al., 2007). *Parameter tuning* means finding good parameter values before the run of the GA, and then running the algorithm using these values which remain fixed during the run. The optimal value for any parameter is problem dependent. For example, the recommended values used for numeric problems are: population size equal to 30, crossover probability equal to 0.95, and mutation probability equal to 0.01 (Grefenstette, 1986). These parameter values give reasonable performance when applied for certain classes of test functions.

The second approach of parameter setting is the *parameter control* which starts with certain initial parameter values. Then, these values are changed during the GA run either in an adaptive way by using feedback information during the GA run or by using a preset formula (Grefenstette, 1986, Smith J. E., 1997).

A common classification of parameter control methods proposed by Eiben et al. (1999) is based on the type of the parameter change. The parameter control methods are classified to deterministic, adaptive and self-adaptive. In the deterministic approach, the

parameters are changed according to some predetermined heuristic formulas which usually depend on time schedule and use no feedback from the GA run (Smit and Eiben, 2009). In adaptive parameter control methods, feedback information is extracted on how well the search is going (James, 1985) (Aine et al., 2009). This feedback is used to control the values and direction of the parameters values. So, the value of the parameter is updated based on some feedback from the search progress of the population. The self-adaptive parameter control combines each chromosome in the population with its own parameter value as a component of the chromosome structure (Hinterding et al., 1996). This value can be subjected to the evolutionary process like mutation, recombination, and selection. Therefore, the value of a parameter is updated and evaluated via GA itself by encoding the parameter values into the chromosome (Meyer-Nieberg and Beyer, 2007).

GA parameters are not independent. The adaptation or tuning for only one parameter at a time neglects the other parameters and leads to inefficient choices (Whitacre, 2007). The tuning operation entails to try all different parameters combinations in order to find a comprehensive tuning. Thus, this operation is time-consuming and does not grantee obtaining the optimal parameters setting (A. E. Eiben, 1999).

In spite of the fact that there is a variety of GA parameter control methods, there are many issues that need to be improved (Michalewicz, 2007). Most of the existing methods only adapt one parameter for certain optimizing problems (Aine et al., 2009).

There are some methods that adapt more than one parameter, but these parameters are treated independently without any feedback among them (Whitacre, 2007).

The relationship between the parameters is not fixed (Kalyanmoy Deb, 1998). It varies according to the stage of the search. Each search period needs different values for these parameters and the amounts of parameters' changes during process affect the performance and the population diversity (D. Quagliarella, 1999). The successful adaptation of many parameters is required in such a way that can keep a good balance between exploration and exploitation and also avoid the undesirable premature convergence problem (Lin and Gen, 2009). The adaptation of more than one parameter during the course of the runs makes the operation of parameter control very complex (De Jong, 2007).

Recently, there is an increasing interest in the use of Fuzzy Logic Controller (FLC) for adapting GA control parameters, to fill the purpose of improving the performance of the search algorithm (Herrera and Lozano, 2009). Fuzzy Logic (FL) has been used as an adaptive method to control GA parameters. The GA resulting from this integration is known as Fuzzy Adaptive GA (FAGA) (Herrera and Lozano, 2003).

The FLC is a control tool for the systems which are hard to control due to mathematical complexity, ambiguity, or system uncertainties like the GA parameter control problem (Sozio, 1999). However, there are several shortcomings of FAGA such as high complexity computations during usage and calculating the knowledge rulebase

(Subbu et al., 1998), ignoring the feedback among GA parameters, the frequency of firing the FLC rules and the appropriate inputs for FLC (Herrera and Lozano, 1996a, Herrera and Lozano, 2003) .

This work contributes to solve the GA parameter control problem using an improved FAGA. The design of the proposed FAGA overcomes the above mentioned weaknesses of FAGAs.

1.2 Research Motivation

The performance of the GA is highly related to the choice of values for its parameters (Eiben et al., 1999). Furthermore, static values are not suited for the search process where the requirement for exploration and exploitation changes as the search progress (Whitacre, 2007). As such, parameter control mechanisms are more successful (Aine et al., 2009). The adaptation of GA parameters is not easy since these parameters have high interdependencies. They interact with each other during the run of the algorithm in a complicated way (Odetayo, 1997, De Jong and Spears, 1991, Kalyanmoy Deb, 1998).

The majority of earlier researches of parameter control focus on finding optimal crossover, mutation rate or population size independently (Grefenstette, 1986, Schaffer et al., 1989, Alander, 1992, Bäck, 1993, Srinivas, 1994, Angeline, 1995, Back and Martin, 1996b, Zhu, 2004, Yu et al., 2007). Most of the FAGA (Subbu et al., 1998) (Herrera and Lozano, 2003) (Last and Eyal, 2005) (Qing et al., 2007) instances

presented in the GA literature only adapt the GAs parameters while ignoring the interaction between parameters.

The motivation of conducting this research is to improve FAGA methods to adapt multiple parameters of GA. Some FAGA use a large rulebase of their membership functions (Lee and Takagi, 1993, Herrera and Lozano, 1996a, Herrera and Lozano, 2000, Jun, 2005, Qing et al., 2006). These large rules need complex computations when using many values of rulebase for FLCs' inputs and outputs. This will cause a high cost of computational resources when calculating the fuzzy rules (Subbu et al., 1998).

It is also considered that there are many shortcomings of current FAGA which needs to be improved (De Brito et al., 2006) (Qing et al., 2006) (Im and Lee, 2008). In (Herrera and Lozano, 2003) and (Herrera and Lozano, 2009), there are many problems of FAGA which have been recommended to be solved. One of these problems the firing of the FLC rules at each generation is fixed or scheduled for permanent number of generations (Herrera et al., 1995a). In many cases, the previous value is still suitable and does not take enough time to affect the search (Herrera and Lozano, 2003). Consequently, it is inappropriate to fire these rules regularly.

Generally, current FAGAs adapt many GA parameters during the run, and use separate rules for each parameter without considering their interdependencies. It is better to design new FLCs taking into account the action of each genetic operator in relation to

the behavior of each one of the remaining ones (Herrera and Lozano, 2003) (Herrera and Lozano, 2009). Moreover, it is important to investigate new additional inputs of FLCs. These inputs should be appropriate variables which can describe the search progress and the population diversity. These additional inputs introduce performance improvements (Herrera and Lozano, 2009). For example, using the population size as input when to control an appropriate mutation rate of the GA.

The significance of this research comes from the importance of the GA parameters' effect on the search behavior and the influence of FLCs on enhancing the GA performance. Thus, some approaches concerning the above mentioned FAGA's components are considered to improve the performance of GA. It is important to note that the adaptive method of this study will be applied on GA as a one paradigm of the EAs. The successful of the proposed adaptive method on GA can be generalized for other parameter of EAs models, Genetic Programming (GP), Evolutionary Strategy (ES) and Evolutionary Programming (EP).

1.3 Goal and Objectives

The adaptation of multiple parameters of GA during the run is important to improve its performance since the parameters have effects on GA behavior. The population size (N), crossover rate (P_c) and mutation rate (P_m) parameters have different strategic influences on the GA search progress. These parameters play a significant rule to balance between exploration and exploitation of genetic search.

The main goal of this thesis is to develop a new FAGA for simultaneous online (during run) control of GA parameters. To achieve this goal, the thesis will pursue the following objectives:

1. To determine the effectiveness of different GA parameters on its performance.
2. To propose a new FAGA method to control P_m, P_c and N parameters with keeping a good balance between search exploration and exploitation.
3. To perform validation of the new FAGA on some of the standard test problems.

1.4 Scope and Limitations of the Thesis

This thesis proposes a new adaptive parameter control method for three parameters of GA: population size, mutation rate and crossover rate. It controls these parameters online, during the course run of the algorithm, based on simultaneous FLCs. The algorithm uses three FLCs to adapt the GA parameters based on feedback from GA performance and population diversity.

Different test problems are used to evaluate the proposed method against related work. In each experiment, a subset of the test suite is used for the comparison. The selection of each subset depends on what is being used in the literature.

In the experiments, the proposed algorithm has been compared to some of the parameter control and the FAGA methods. The choosing of the comparative methods is limited because the performance comparisons with other techniques, such as experimental design based, are not available.

1.5 Contributions

This research has successfully adapted multiple parameters of GA during the run while considering the interaction between them to increase the performance of GA. Following are the contributions of this research:

- Adapted the population size, mutation rate and crossover rate during the course of run simultaneously by considering the interaction between these parameters. This interaction and the feedback of GA performance are used to keep a good balance of exploration-exploitation relationship. They are also used to determine the direction and magnitude of the change in the parameter value.
- Developed a new FAGA using three FLCs, FLC_{Pm} for adaptive P_m, FLC_N for adaptive N and FLC_{Pc} for adaptive P_c. These FLCs have improvements as follows:
- **Low computations**: The design of the proposed algorithm reduces the required amount of computations for each FLC rules through the refinement operation of the FLCs rulebases and the proposed fuzzy controllers firing system.
- **Feedback among parameters**: FLCs have several effects on the adapted parameters. Based on the actions of each parameter, the effects of parameters have different influences on GA behavior during the run of the algorithm. The feedback between parameters' actions allows making a suitable balance between their actions and GA performance. Taking this issue into consideration improves the performance of the GA.
- **Appropriate inputs of FLCs**: FLCs outputs will be the values of GAs parameters which need to be changed based on the inputs. Therefore, inputs of FLC are very important to describe the current performance of GAs. These inputs are critical

indicators for doing or ignoring the effectiveness of parameter adaptation. The inputs can be some information about search progress and population diversity in addition to the values of current parameters. The proposed algorithm has extra inputs which make FLCs more accurate for generating a suitable parameter set.

1.6 Thesis Organization

The work reported in thesis will be presented in six chapters organized as follows:

In Chapter 1, the present the basic concepts, scope, limitations, contributions, goal and main objectives of this work are presented.

Chapter 2 describes a literature review of the related work in the two domains of this research: genetic algorithms parameter control and the fuzzy logic controllers for adaptive genetic algorithms. This chapter describes some difficulties of GAs parameters for practical problems in general. Finally, the chapter focuses on the main challenges in the fuzzy adaptive genetic algorithms and how previous research addressed such problems.

Chapter 3 explains the proposed methodology for Simultaneous Adaptation of Multiple Genetic Algorithms Parameters (SAMGAP) technique using Fuzzy Logic Controllers (FLCs). It also presents several definitions for fuzzy Logic controllers and their usage in the thesis. In order to understand the relationship among different GA

parameters during run, how they affect performance, the simultaneous adaptation of GA parameters using FLCs will be explained in detail with some features to improve the SAMGAP.

Chapter 4 gives the particulars of the experimental setup for the experiments which have been carried out to investigate the merits of SAMGAP with several test problems. This chapter explains the different measures of GA performance for the selected test problems.

In Chapter 5, the results of applying different parameter control methods of GAs are reported. The SAMGAP is compared here with the related works of adapting the GA parameters with the same test problems. It includes a discussion to demonstrate the advancement in SAMGAP.

Chapter 6 concludes the thesis, summarizes the major contributions of this work and presents some directions for future research works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The aim of this chapter is to provide a literature review of this thesis. It presents an introductory paragraph to set GA in context with other Evolutionary Algorithms (EAs). This involves a survey of previous literature that are related to the present research. The literature review presented here covers the three directions of the work. Figure 2.1 shows the main areas of the research literature review and the overlap between the elements of the research.

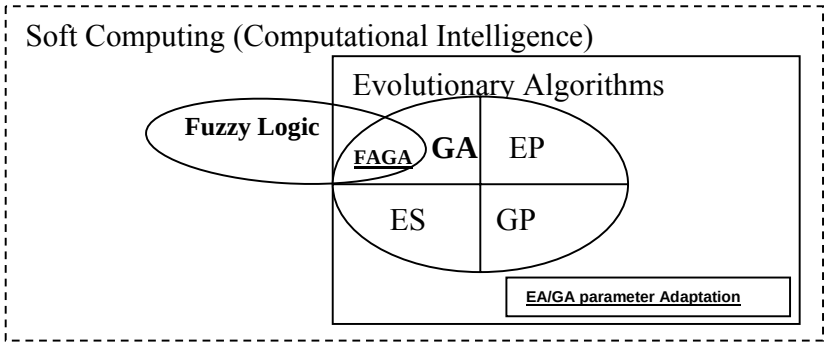

Figure 2.1: Main areas of this study

The next Section (2.2) gives a quick review of the simple GA. It demonstrates the operators and the role of each operator in the search performance. Section 2.2 explains GA parameters and their influences on GA behavior. Section 2.3 gives some information about the two major terminologies in search: exploitation and exploration which are related to parameter adaptation. Section 2.4 investigates the problem of GA

parameter adaptation. Sections 2.5 and 2.6 investigate the FAGA structures and design, the related work on the FAGA and some further considerations to improve the FAGA algorithms. Finally, Section 2.7 summarizes and concludes this chapter.

2.2 An Overview of Genetic Algorithms

Evolutionary Computations (EC) uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems (Kenneth De Jong et al., 1997). In recent years, EAs have been applied to a variety of optimization problems in different applications such as planning, scheduling, design, control systems, electrical power systems, pattern recognition, and classification problems (Smith, 2003).

Generally the classes of EAs can be distinguished by their representation of the search space and the specialization of the various operators used. For instance, in 1975, Holland introduced GAs with binary or finite discrete representations (Goldberg, 1989, Back, 1996).

The process of GAs depends on the perceived performance (the fitness) of the individual structures as defined by the environment. More precisely, GA algorithms maintain a population of structures that evolves according to some rules of selection and other operators such as recombination and mutation. Each individual in the population has a measure of its fitness in the environment. Selection focuses attention on high fitness individuals, thus exploiting those individuals for further improvement

(Davis, 1991).

The recombination and mutation perturb the individuals providing general opportunities for exploration and exploitation. So, from a biologist's view, GAs are sufficiently complex to provide robust and powerful adaptive search mechanisms (Ashlock, 2006).

There are many ways to implement GA beside the standard GA such as Generational GA and Steady State GA (Goldberg, 1989). The next subsections give a quick overview of the main standard GA components.

2.2.1 Structure of Genetic Algorithms

A GA is a search procedure inspired by principles from nature where the best gene is selected for the next generation. That means the survival of the fittest. GA is used as a general purpose optimization method for solving problems without deep knowledge about the search space (Ashlock, 2006). GA can generate effective global solutions better than the traditional search techniques (Menon, 2004).

The operation of standard GA starts with generating randomly encoded chromosomes by initializing a number of individuals (chromosomes) as an initial population of feasible solutions. Each individual is equivalent to a particular candidate solution to the problem to be solved, and it consists of several genes. The initial population consists of several individuals defined by their chromosomes. Each individual of the population is evaluated using some measurement of fitness (Mitchell,

1999). Then, the selection operation determines the best specifications individuals from the whole population. Therefore, the individuals with highest fitness factors survive in this case (Smith, 2003).

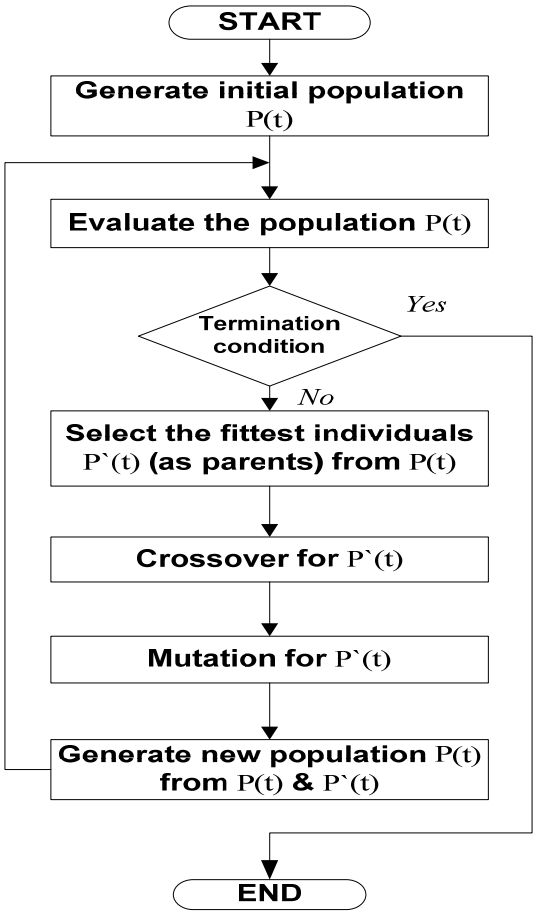

Figure 2.2: General structure of genetic algorithms. Adapted from (Eiben and Smith, 2007)

After that, the selected parents are recombined using a crossover rate (P_c) to create new population individuals which are new offspring to be used for next generation. All offspring will be mutated by altering some genes in a chromosome with a certain mutation rate (P_m). The offspring are inserted into the population replacing the parents and producing a new population (Kenneth De Jong et al., 1997).

This process is repeated and will lead to better individuals until the optimization criteria are reached. Figure 2.2 shows the general structure of GA.

The design of a GA for any particular problem (Davis, 1991) consists of the following five main components:

Genetic representation is the encoding technique to represent the potential solutions of the problem in the chromosome structure (Davis, 1991). Every particular solution represents a point in that search space, and a search space is a set of all possible solutions. Typically, GAs use string structures containing binary decision variables (Chambers, 2001). The structure that encodes a solution is called a chromosome or an individual of the population; the decision variable is called a gene and its value is called an allele. Illustration of GA population and individual is shown in Figure 2.3.

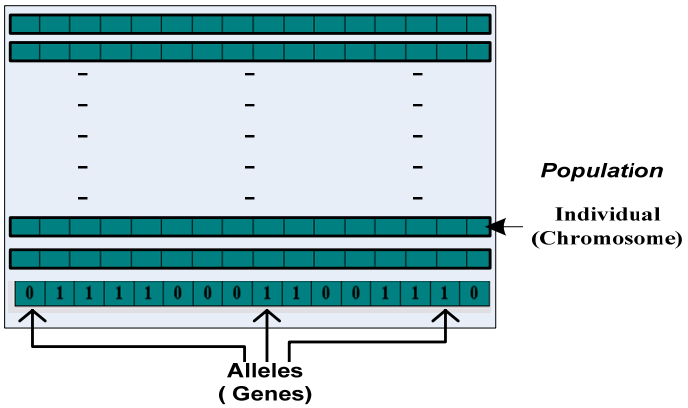

Figure 2.3: Genetic material in the GA population and individual

Initialization procedure is the way to create an initial population of solution.

Fitness evaluation function rates the solutions in terms of their fitness to the environment (Kenneth De Jong et al., 1997). It represents how well the individual adapts to the search space, so the GA discovers solutions that have high fitness values among the set of all possible solutions.

Genetic operators allow parents to be selected and can also modify individuals in order to generate offspring. In a simple GA, there are three genetic operators: selection, crossover (recombination) and mutation (Davis, 1991). The main genetic operators will be demonstrated with some details in the next sections of this chapter.

Parameter settings are the values for different parameters used by GA such as the population size and the probabilities of applying genetic operators like selection, crossover and mutation (Eiben et al., 1999). Particular details of this GA part will be presented in details throughout the following sections.

2.2.2 Reproduction Operators of Genetic Algorithm

The reproduction of a GA population is required to evolve from one population to another better population using different operators (Tuson, 1995). Primarily, the GA uses three operators for this purpose: mutation, crossover and selection (Back et al., 2000b). Due to the influence of the crossover and mutation on GA performance, the two operators are the most important operators of GA (Eiben et al., 2007). The following sections give some details about these operators.

Selection: The selection operator of GA simulates the “survival-of-the-fittest” principle. It controls the search towards the promising regions which may include better solutions in the search space (Goldberg, 1989). The major principle of selection is to choose the parents for the recombination process which adherence to the selection of fitter individuals. The purpose of selection is to keep individuals with high fitness values and remove those with low fitness values (Forrest, 1993). The individuals with high fitness values are selected to ensure that their good genes will be carried forward into the next generation.

The selection operator can be implemented using various mechanisms in order to choose the best chromosomes such as roulette wheel selection, Boltzman selection, tournament selection (Forrest, 1993), rank selection and steady selection state (Goldberg and Deb 1991; Thierens and Goldberg 1994; Blickle and Thiele 1995; Bäck 1996).

Roulette wheel selection is the most commonly used selection strategy (Schaefer, 2007b). In order to ensure that highly fit individuals have better probabilities to be selected through the crossover and mutation for the next generation, the probability of being selected using Roulette Wheel selection is based on the individuals’ proportional fitness (Smith, 2003).

An advanced strategy called "Elitism” can be used together with the selection methods (Ahn, 2006a). It ensures that the best individuals of the population are preserved during the selection process to be used when creating the new population by crossover and mutation (Ahn, 2006b). Therefore, Elitism is able to improve the

performance of GA. It guarantees that the maximum fitness of the population can never be reduced from one generation to the next and, thus, it prevents losing the best found solution (Bo and Gallagher, 2005).

The selection pressure is the probability of applying the selection operator (Sarma and De Jong, 1998). Using high selection pressure might lead to premature convergence (Back, 1994) because most of the population parts consist of identical chromosomes which represent sub-optimal solutions (Vajda et al., 2008). Using low selection pressure, on the other hand, makes ineffective selection and makes the search progress slower than necessary. Therefore, it is recommended to use low selection pressure at the beginning stages of the genetic search and higher values at the end in order to narrow the search space (Back, 1994).

Crossover : The main idea of crossover is that good solutions can be recombined to generate better solutions in the next generations (Michalewicz, 1996). Crossover operator exchanges the information of two or more parents to produce new offspring. The chromosomes portions of the two parents are swapped to create a new "child" chromosome that is a combination of genes from both parent chromosomes (Eiben and Smith, 2007).

The process of crossover starts with the selection of two individuals randomly and, then, determining the crossover points as breakpoints for string segments exchange. One or more crossover points are selected randomly and the same points are used on

both parent chromosomes. After the exchange, two combined individual offspring are produced.

In GA, crossover is considered as the main search operator and it is responsible for most of the search performed by GA (Michalewicz, 2007). A crossover process gives a chance for partial solutions to be mixed in order to discover better solutions. It works based on the idea of retaining better building blocks within the population for the next generation. The crossover operator alone is insufficient. It can cause stagnant population ultimately because its inability to introduce new material into the population as it uses existing materials (Spears, 2000). A simple example for a commonly used operator, a single-point crossover is illustrated in Figure 2.4.

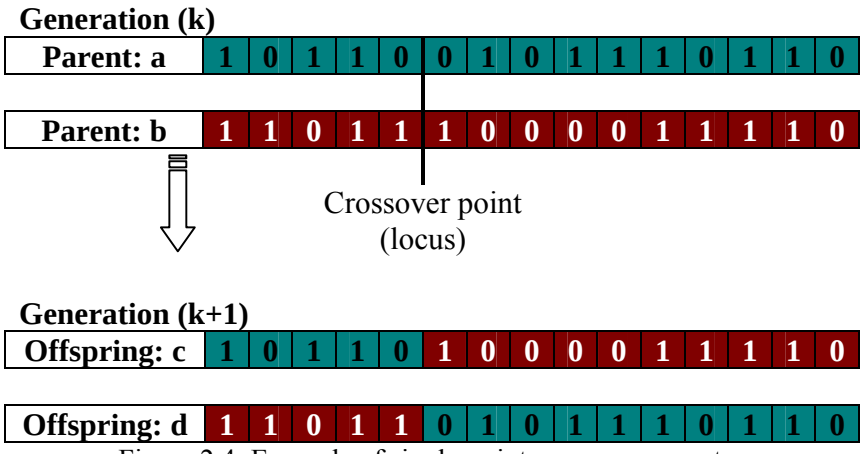

Figure 2.4: Example of single-point crossover operator

In this single-point crossover, a random bit position (locus) is selected along the two parent chromosomes, and the genetic materials of sub-sequences before and after the determined locus of two parents are swapped to create the new two offspring (Back, 1994).

There are many methods to apply a crossover operator on GA chromosomes, but the main idea is to exchange portions of one solution with that of another solution. The common method depends on the crossover points such as single-point crossover, two-point crossover, multi-point crossover and uniform crossover (Kenneth De Jong et al., 1997).

The crossover gives an opportunity for producing new superior solutions better than the two parents since the encoding of the two individual parent chromosomes will end up in a single chromosome. However, this chance of getting better offspring by using crossover can not be ensured. Usually, crossover operator is applied with a high rate (P_c) when compared with the probability of the mutation (P_m).

Mutation: In GA, if only the crossover operator is applied, the search can get stuck around local optimal solutions (Spears, 2000). The recombination process is unable to explore the regions of search space which are not represented in the population's genetic structures (Hong et al., 2002). Mutation helps prevent the population from stagnating at any local optima. It introduces new genetic structures into the population by altering some of the bits of the chromosome randomly.

This operator occasionally flips one bit value or more at randomly selected locations in a chromosome (Kenneth De Jong et al., 1997). The modification of the mutation operator is totally random. Consequently, the new added genetic materials, new alleles in particular, do not exist previously in the population. The mutation produces different structures related to other regions of the search space (Cludio et al.,

2005). An example of mutation is illustrated in Figure 2.5. The arrows in the figure represent the positions of mutation.

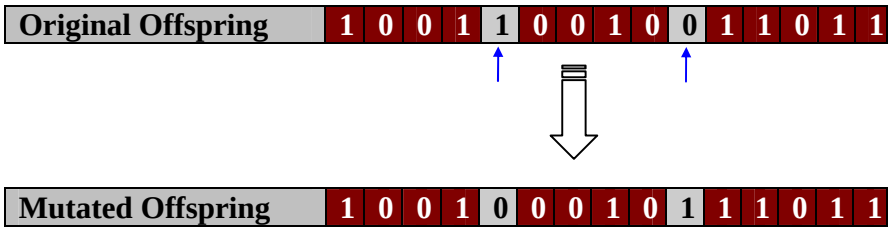

Figure 2.5: Example of mutation operator

The mutation can occur at each bit position in the individual with some probability. The value of P_m controls the probability which is usually very low and represents a chance for each bit to be changed (Schaefer, 2007a). If the P_m is 1.0 (100%), this means that all the bits in the chromosome will be inverted when the mutation operator is applied (Andre and Godelle, 2005).

Standard GA uses a point mutation operator style in which every locus in the individual is subject to the same P_m (Mitchell and Taylor, 1999). If the gene is selected, then the change of mutation operator is applied to this gene (Spears, 2000).

2.3 Genetic Algorithms Parameters

The process of GA normally involves many parameters. This section gives an overview of the main parameters which are typically used in the parameter control and will be included in this study.

2.3.1 Population Size

Population size (N) has been considered as a one of the most important GA parameters (Arabas et al., 1994). If the population size is too small, the GA may converge too quickly. On the other hand, if it is too large, the GA may waste computational resources. Consequently, the adaptation of N is important because small values of N lead to poor performance of GA as the population provides an insufficient sample size. Meanwhile large values of N cause more evaluations per generation (Fernandes et al., 2001) (Yu et al., 2007). This results in an inappropriately slow rate of convergence (GA performs a more informed search). Consequently, if N is too small, there will be a quality penalty, and if it is too large, there will be a time penalty (Furutani et al., 2007). Figure 2.6 illustrates the situation of GA population size difficulty.

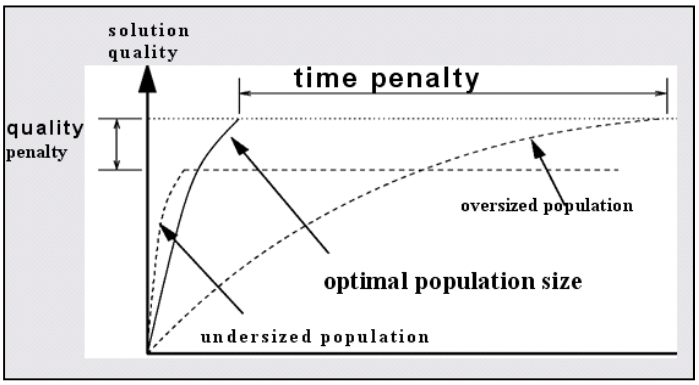

Figure 2.6: Genetic algorithm population sizing penalty (Furutani et al., 2007)

Lobo and Lima (2007) concluded that population size (N) is the most essential factor that robustly influences the convergence speed of GA search process. Therefore, they recommended that this parameter should be modified according to the problem to be solved.

Some of the early studies recommended an optimal N to be between 30 and 100 (De Jong and Spears, 1991) (Alander, 1992) . Other studies (John J. Grefenstette, 1986) (Alander, 1992) (Robert, 1993) and (Arabas et al., 1994) emphasize that the N must be adjusted for different problems according to the problems complexity.

Many recent methods have been developed to adapt the GA population size. These methods can be categorized into two main types. The first group is based on the GA population sizing theory. It focuses on determining the population size according to problem difficulty (Yong, 2003) (Fernando and David, 2004), (Tian-Li et al., 2005) and (Yu et al., 2007). The second group mimics the age and lifetime of the individual in population (Arabas et al., 1994) and (Lobo and Lima, 2007).

2.3.2 Mutation Rate

The mutation operator works for renovation of the genetic material. The probability of applying this operator on individual genes is called mutation probability (P_m) or mutation rate (Back, 1993). This parameter controls the speed of GA in exploring a new area of search space (Lin et al., 2003). The low value of P_m is desired to avoid the premature convergence of GA to a suboptimal solution, whereas high value of this parameter transforms GA search to an essentially random search algorithm (Menon, 2004).

The mutation reinforces the ability of the GA to find a near optimal solution to a given problem. The variety provided by the mutation operator is needed to ensure that

the entire solution space is searched (Thierens, 2002).It is also noted that the high value of P_m causes the loss of the good genes of the individuals. This decreases the exploitation of the regions with high fitness solutions (Cervantes and Stephens, 2006).

The adaptation of mutation rate value is one of the most common topics which have been researched in GA parameter control field. Goldberg (1989) concluded that P_m must be set to a very low value (0.0001). Some researchers state that the P_m of 0.001 is better (Ochoa et al., 1999). However some empirical tests have shown that a higher value of P_m is better in the beginning of the search process, and then using lower value in the long run (Ochoa et al., 1999). More recent studies on this parameter have been done (Cervantes and Stephens, 2006), (Zhang et al., 2008) and (Lau et al., 2009).

2.3.3 Crossover Rate

The crossover rate (P_c) controls the frequency with which the crossover operator is applied. It determines the probability at which the solutions are affected by the crossover. Crossover rate manages the exploitative capability of GAs in locating a local optima (Tzung-Pei et al., 2002).

A high value of P_c leads to quicker exploitation in order to introduce new structures into the population (Back et al., 2000b). Crossover rate (P_c) with very high values discards good individuals faster than they can be exploited. The low values of this parameter may stagnate the search due to the lower exploration rate (Tuson, 1995).