A Bi-objective Multi-population Genetic Algorithm with Applications to Function Optimization and Ellipse Detection

Jie Yao

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

January 2008

Your file Votre référence
ISBN: 978-0-494-37767-3
Our file Notre référence
ISBN: 978-0-494-37767-3

Abstract

A Bi-objective Multi-population Genetic Algorithm with
Applications to Function Optimization and Ellipse Detection

Jie Yao, Ph.D.
Concordia University, 2008

This dissertation presents a novel Bi-objective Multi-population Genetic Algorithm (BMPGA) for multimodal optimization problems. BMPGA is distinguished by its use of two separate but complementary fitness objectives designed to enhance the diversity of the overall population and exploration of the search space. This is coupled with a multi-population strategy and a clustering scheme, both of which together focus selection pressure within sub-populations, resulting in improved exploitation of promising optimum areas as well as effective identification and retention of potential optima.

The practical value of BMPGA is demonstrated in several applications. In optimization of benchmark multimodal functions, it shows clear superiority over other typical multimodal GAs: Multinational GA [1], Dynamic Niche Clustering [2] and Clearing [3], with respect to overall effectiveness, general applicability and reliability.

In the application of imagery ellipses detection, BMPGA is compared with both widely used Randomized Hough Transform (RHT) [4] and Sharing Genetic Algorithm (SGA) [5]. In thorough and fair experimental tests, utilizing both synthetic and real-world images, BMPGA exhibits solid advantages over RHT and SGA in terms of accuracy of

recognition - even in the presence of noise or/and multiple imperfect ellipses in an image - and speed of computation.

Finally, we successfully extend BMPGA to the segmentation of microscopic cells, which is a necessary first step of many automated biomedical image processing procedures.

Acknowledgements

I have always enjoyed the research that has gone into this thesis. I am grateful to all those who have contributed to this thesis and helped me.

First of all, I would like to express my gratitude to my supervisors, Dr. Nawwaf Kharma and Dr. Peter Grogono, for their support and guidance during all these years. I feel very privileged to have worked with them. I learned a lot from them in the field of the research as well as technical writing skills. Their suggestions, ideas and positive criticism on the theoretical aspects of this work inspired me to work harder for success. They helped me grow as a researcher and a professional.

I would also like to give my special thanks to my family: my husband and my parents. They were extremely understanding and supportive of my studies. Without their encouragement and help, I would not have endured until finally coming to fruition.

Finally, I want to thank my wonderful friends and colleagues for their help and advice during my research.

Jie Yao

Montreal, Canada

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Multi-modality, as suggested by its name, indicates the existence of one or more local optimum, besides the global optima, on a function's surface. This property unavoidably complicates the problem to be optimized. If the ultimate goal is the identification of a global optimum, the existence of local optima, to a large extent, tends to affect the search negatively, due to entrapment around local optima. There are situations, however, in which the local optima are as important as, and sometimes even more important, than the global optimum.

Multimodal optimization refers to the general problem of locating the local (and global) optima on a function's surface. It is a problem with practical significance in many application domains. Some of them naturally necessitate search of local optima.

Pattern matching and recognition is a typical example [6-8]. Given a template pattern and an image, all patterns that match the template perfectly or not so perfectly are of interest. A perfect match corresponds to a global optimum, whereas imperfect matches correspond to local optima. These local optima should not be dismissed, as they represent target patterns, though imperfectly formed.

In some other applications, local extrema or sub-optimal solutions provide a useful pool of candidates for further experimental investigation. In [9, 10], the problem consists of determining the sequence of DNA clones from restriction-fragment data. The molecular biologists aim to look for an exact sequence that is global optimum in their mathematical formulation. The search for this global optimum, however, is tightly tied up with the selection of the optimization criterion, which is fabricated by specific

approaches. With such criterion, the scientists cannot find the *correct* solution using mathematical methods. Therefore, the authors proposed to search for the *k*-best sub-optima.

Similarly, in the applications of path planning for mobile robots [11, 12], the availability of alternative suboptimal paths is an important part of the solution. The search maps of path planners are usually discretized approximations of actual environments. An optimal path found in the search map may not be the best path in the real workspace. Several important factors must be taken into consideration when deciding on the optimization criteria, which are often very complex and subjective. Hence, locally optimal and globally suboptimal paths provide a useful pool of candidates for further investigation. This is of particular significance in dynamic environments, where one or more of these paths may unexpectedly become infeasible. An alternative path must then be chosen from the pool of feasible paths.

Job scheduling [13-15] is another typical example. The objective is to find multiple solutions so that the human experts can choose the solution that better adapts to the actual conditions, e.g. politics or unexpected happenings.

These are just a few examples that illustrate the considerable significance of multi-modal optimization, for scientists and engineers, among others.

Fig. A–1 in Appendix A exhibits a typical multimodal surface of function D_1. D_1 has a finite set of discrete or independent optima. These optima represent a finite set of potentially useful solutions to a multimodal problem. This type of multimodal function is interesting to us in this study.

Hill climbing [16], [17], [18] is a classical optimization technique. Starting from a

random state, it always moves to a successor state that is better than the current one. The direction and the length of each move between consecutive states may be adjusted adaptively.

As a typical exploitation search method, a hill climber potentially ignores exploration of other areas in the search space. Therefore, it works well on uni-modal landscapes, but tends to get trapped in local optima on multi-modal landscapes.

An iterative hill climbing algorithm [16] may partially solve this dilemma, simply by iteratively running the local hill climbing method starting from different random points. However, a full exploration of all existing optima is not guaranteed. And it is subject to a high risk of repeated visits within the same area, depending on the location of the random initial points.

In recent years, many researchers have shed light on Genetic Algorithms (GA) for multimodal optimization [1, 2, 11, 12, 19-30], which, as a major branch of evolutionary computation, are promising for search in complex multimodal landscapes.

Several studies have compared various forms of hill climbing methods with GAs [31-34]. They believe that an ideal GA has quicker and more thorough search than a hill climber. In [18], Mahfoud implemented a parallel hill climber. It starts with a randomly generated population and forces each individual to converge to its nearest local optimum. By comparing this hill climber with several other GA approaches, the author concluded that the parallel hill climbing is best for easy problems with less than ten peaks. It may work reasonably well on problems of medium complexity with a moderate number of peaks, however, it shows incapability on complex landscapes and displays both isolation and misleadingness.

Similarly, Corne [35] suggested superiority of GA to the hill climbing technique on multimodal landscapes, given the condition that the GA is adequately configured to maintain multiple niches. The niches can be achieved *via* a large population size or a good inherent diversity maintenance mechanism.

In [21], the Species Conserving Genetic Algorithm maintained a good diversity of the population by conserving different species and outperformed the hill climbing method on some benchmark multimodal landscapes.

Indeed, GA's inherent properties make it theoretically advantageous for multimodal optimization problems. However, how to make good use of these properties remains a challenge. Existing GAs are known to suffer from various drawbacks, such as an inferior performance on irregular multimodal surfaces [2], [3], inability to preserve species [1], [2], [3] as well as limited applicability relying on landscapes and parameter settings [1], [2], [3].

In this dissertation, a novel Bi-objective Multimodal Genetic Algorithm (BMPGA) is proposed and studied. Its main features are:

- A bi-objective mechanism that enhances the diversity of the population and expedites exploration of the search space;
- A novel clustering algorithm that forms subpopulations of individuals around potential optima effectively and stably, without *a priori* knowledge of the landscape;
- A multi-population scheme that consolidates diversity and speciation within the population and intensifies exploitation of areas of potential optima.

BMPGA is applied to several applications including optimization of benchmark multimodal functions, ellipse extraction and microscopic cell segmentation. With augmented diversification and intensification, it achieves excellent results.

BMPGA is compared to a number of well known algorithms in these applications. In multimodal function optimization, it is compared to Dynamic Niche Clustering [2], Multinational GA [1] and Clearing [3]. It is also compared to its own sibling: the Multi-population Genetic Algorithm (MPGA). It consistently returns better results than all of these GAs in terms of its ability to *find* and *maintain* optima, as well as its generality and consistency.

In ellipse extraction, BMPGA is compared to one of the most investigated approaches – the Randomized Hough Transform [4] and another GA based method – the Sharing GA [5]. BMPGA exhibits solid advantages over these two algorithms in terms of accuracy - even in the presence of noise or/and multiple imperfect ellipses in an image, and efficiency.

The dissertation is structured as follows:

Chapter 2 of this thesis introduces necessary background. It first presents a brief overview of a classical GA and its basic flowchart. Typical evolutionary operators of a GA are discussed. A fundamental theory describing the behavior of GAs – the Schema Theorem is briefly presented. GA's advantages are analyzed in five aspects. Limitations of a classical GA on multimodal optimization are also pointed out.

Following the introduction of classical GAs, GAs for multimodal optimization are hierarchically categorized and investigated. All multimodal GAs can be roughly divided

into two categories: sequential GAs and parallel GAs. The parallel GAs can be further categorized into genetic operator based GAs and population based GAs. The characteristics, the advantages and the disadvantages of each category and sub-category of GAs are discussed. The chapter is concluded with comparison between major categories (sub-categories) of multimodal GAs: sequential GAs versus parallel GAs and genetic operator based GAs versus population based GAs.

Chapter 3 discusses clustering approaches adopted in multimodal GAs. Three major categories of clustering methods existing in the literature are introduced. They are distance based approaches, topology based approaches and hybrid approaches. Following that, the methods to compute the center of each cluster are also described and compared.

After that, a new clustering algorithm - Recursive Middling (RM) is proposed. Validity of RM is empirically tested in a Monte Carlo process and a typical multimodal GA - Dynamic Niche Clustering [2]. RM is compared to other clustering techniques, i.e. those based on Euclidean distance and Ursem's HV function [1] and demonstrates clear superiority over them.

The Bi-objective Multi-population GA (BMPGA) for multimodal function optimization is studied in Chapter 4. After the overall framework of BMPGA is introduced, major processes, including fitness evaluation, clustering and evolution within this framework, are discussed in detail. Motivation and mechanism to use two objectives (fitness terms) are interpreted. Three clustering behaviors – migration, splitting and merging are presented. In evolution, concepts of full elitism and population control are discussed. BMPGA is compared to several typical multi-modal GAs, including Multinational GA [1], Dynamic Niche Clustering [2] and Clearing [3], and demonstrates

superiority over them.

Chapter 5 presents a real world application using BMPGA - geometric shape extraction in images (so far we aim to extract ellipses). This application adopts the basic framework of BMPGA in Chapter 4. However, it also has application-specific implementations such as chromosome encoding, fitness evaluation, clustering and some evolutionary operators, e.g. crossover and mutation. Implementation and experimental results of this application are discussed in detail. BMPGA is compared to one of the most popular and most investigated algorithms – Randomized Hough Transform [4] and another multi-modal GA – Sharing GA [5]. BMPGA outperforms both algorithms in terms of accuracy, tolerance to salt and pepper noise as well as efficiency.

Chapter 6 further presents a real world application based on Chapter 5 – segmentation of microscopic cells. This application entails two stages. Stage 1, called blob extraction, extracts contours of blobs of cells and outputs them to stage 2 - ellipses detection, which then uses the approach in Chapter 5 to approximate the given contours with ellipses. Both visual and statistical results are given and analyzed.

The thesis is concluded with potential future work as described in Chapter 7.

Chapter 2 Background

2.1 Classical Genetic Algorithms (GA)

2.1.1 A Classical GA

The Genetic Algorithm (GA) [36] is an optimization technique inspired by evolution and Darwin's natural selection theory. A GA performs a parallel beam search, which may be viewed as a combination of hill climbing and random search. As a major branch of evolutionary computation, it is a promising technique for optimization problems, especially those with complex multimodal landscapes.

A classical GA encodes a potential solution to a specific problem as a chromosome and evolves a population of chromosomes from generation to generation until the termination condition is satisfied. It typically starts with a randomly generated population, each individual (chromosome) of which is assigned a fitness value (indicating the *goodness* of the potential solution). After ranking all the chromosomes with respect to their fitness values, a proportion of the best candidates are kept for the new population (elitism). Pairs of parents are selected within the current population in such a way that a chromosome that is more fit has more reproductive opportunities. New offspring are generated, by exchanging the genetic information of their parents, and put into the new population. This behavior is called reproduction, recombination or crossover. Mutation is then applied to the population with a small rate. After that, the new population is passed through to the next generation.

The overall flowchart of a classical GA is shown in Fig. 2-1. Essentially, it

contains the following genetic operations:

- Selection: selecting elites to be kept in the next population and parents to be mated;
- Crossover: recombining parental chromosomes to generate offspring;
- Mutation: randomly changing some chromosomes.

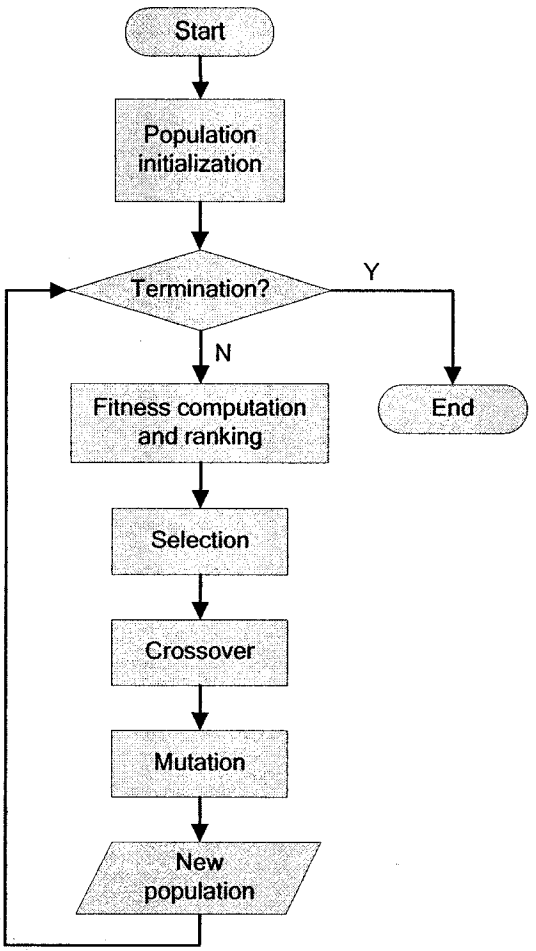


Fig. 2-1 The flowchart of a typical GA

2.1.2 Genetic Operations

As introduced above, a typical GA involves three major operations: selection, crossover and mutation.

2.1.2.1 Selection

Selection is a process that selects individuals in the population to survive and reproduce. A basic principle is that fitter individuals get more opportunities to be selected.

1.1.1.1.1 Elitism

Elitism was first introduced by De Jong [37]. It helps to preserve the best individuals in the population and pass them through to the next generation without alteration. These individuals, called elites, may otherwise be lost if they are not selected to reproduce or if they are disrupted by crossover or mutation.

1.1.1.1.2 Fitness Proportionate Selection

Fitness proportionate selection is the most common selection method in GA. With this scheme, the possibility (P_i) of an individual i to be selected to reproduce is that individual's fitness divided by the sum of the fitness of the population:

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.1}$$

where N is the size of the population, and f_i is the fitness of the individual i.

The expected number of times i is selected (E_i) is:

$$E_i = N \cdot P_i = N \cdot \frac{f_i}{\sum_{j=1}^{N} f_j} = \frac{f_i}{\sum_{j=1}^{N} f_j / N} = \frac{f_i}{\bar{f}} \tag{2.2}$$

where $\bar{f}$ is the average fitness of the population. Therefore, the expected number of times an individual to be selected to reproduce is its fitness divided by the average fitness of the whole population.

Fitness proportionate selection can be implemented with the Roulette-wheel algorithm. For a population of N individuals, a circular roulette wheel of N slices is constructed. A slice represents an individual within this population. The size of the slice is proportionate to the probability of selection. Selection of an individual is equivalent to spin the wheel. The pointer of the wheel finally stops at the individual to be selected. To select N individuals from the population, the wheel is spun for N independent runs.

Fig. 2-2 gives an example of a wheel made up of five individuals, A, B, C, D, and E. The fitness values of these individuals and their sum are listed in the left lower bottom of the picture. It can be seen that E has the largest fitness value, hence its slice (possibility of selection) accounts for 33% (5/15) of the wheel; whereas A's slice only accounts for 7% (1/15) of the wheel.

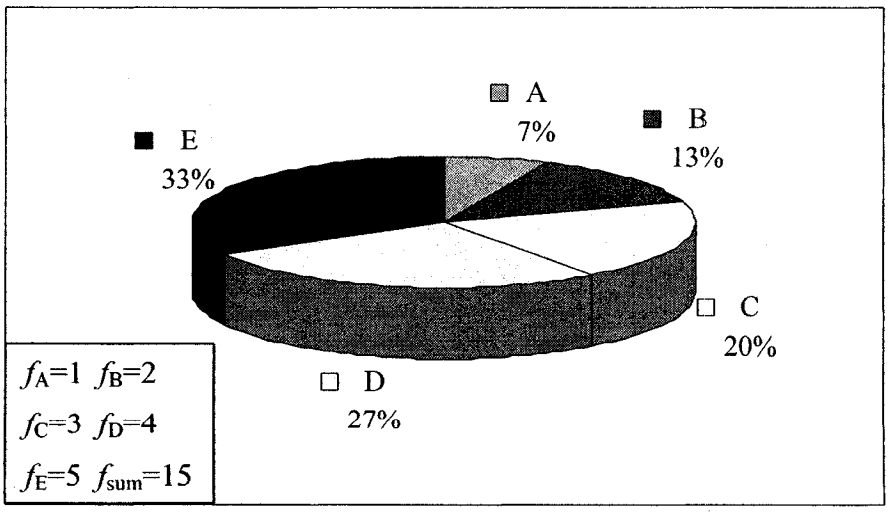


Fig. 2-2 A example of a roulette wheel

The roulette wheel selection can be implemented as follows:

1. Rank the fitness of all individuals from high to low;
2. Sum the total fitness of the population, denoted as S;
3. Repeat N times:
   a) Choose a random number r between 0 and S.
   b) Loop through the individuals in the population and sum their fitness values until the sum is greater than or equal to r. The last individual whose fitness puts the sum over r is selected.

Fig. 2-3 demonstrates 3 random trials of the roulette wheel selection based on the data above. Selected individuals are A, C, and D, respectively.

Trial #	r
1	4.65
2	10.5
3	14

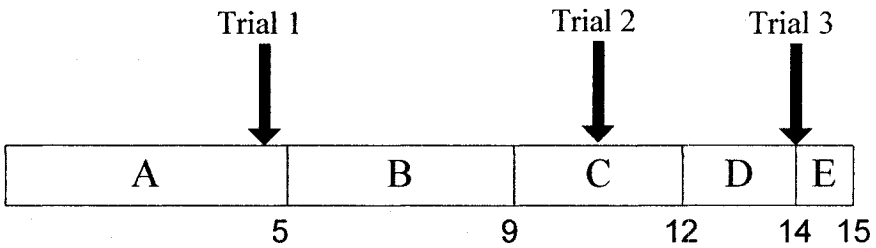


Fig. 2-3 Roulette wheel selection

Statistically, the roulette wheel approach generates the expected number of offspring for each individual. However, the actual number of offspring obtained may often be far from its expected value, especially with a relatively small population. Baker [38] proposed a Stochastic Universal Sampling (SUS) method to minimize this "spread", i.e. the range of possible actual values, given an expected value.

Unlike the roulette wheel algorithm that selects N individuals from the population

by repeated N random samplings, SUS adopts a random sampling once to select all of the individuals at evenly spaced intervals. The implementation is:

1. Rank the fitness of all individuals from high to low;
2. Sum the total fitness of the population, denoted as S;
3. Choose a random number r between 0 and S/N;
4. Let *sum* = 0, for all individual i = [0...N-1] in the population, repeat N times:

    If *sum* < r and *sum* + fitness of i > r

        Select i;

        $r = r + S/N$;

        continue;

    sum = *sum* + fitness of i;

Fig. 2-4 demonstrates 3 random trials of the roulette wheel selection based on the data above. Selected individuals are A, B, and D, respectively.

interval = *15/3* = *5*
r = 2.5 ▯ [0, interval]

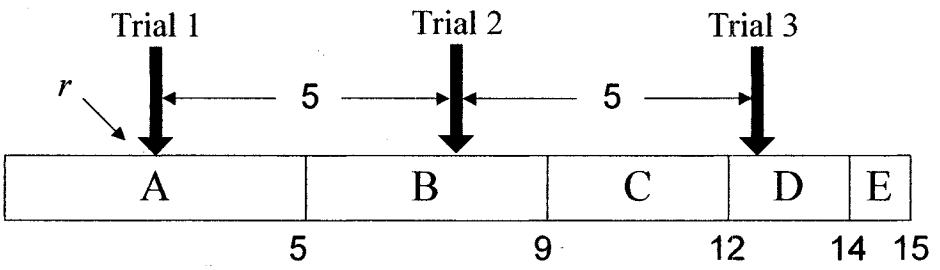


Fig. 2-4 Stochastic Universal Sampling

SUS ensures to select individuals with quantity close to the expected number.

1.1.1.1.3 Fitness Rank-based Selection

With fitness proportionate selection, if the fitness variance of the population is high and a

small number of individuals are much better than other individuals in the population, those worse individuals will have very few chances to be selected. Consequently, the group of highly fit individuals will generate an excessive number of offspring. The search is narrowed down too quickly without exploration of other regions of the search space. This phenomenon is called *premature convergence*. On the other hand, when the fitness variance of the population is low, the evolution tends to stagnate since there is not enough selection pressure to select highly fit individuals.

Fitness ranking selection overcomes these problems by assigning the expected value of each individual based on its rank rather than its absolute fitness value. This is a simple and effective way of controlling selection pressure. It avoids giving too much emphasis to a small group of highly fit individuals, and thus reduces the selection pressure when the fitness variance is high. It also keeps up selection pressure when the fitness variance is low since absolute fitness differences are not taken into account. The ratio of expected values of individuals ranked i and $i + 1$ is the same no matter how much their absolute fitness differs.

Implementation of fitness ranking selection is simple. It first ranks the population from the best to the worst individuals. Each individual then receives a fitness value from N to 1, with the best having N, and the worst having 1. Similar to SUS, the probability of each individual being selected for mating depends on its fitness normalized by the total fitness of the population.

Fitness ranking selection may depress selection pressure and make GA slower in finding highly fit individuals. However, it also results in increased preservation of diversity in the population. When searching in a multi-modal space, this scheme is more

favorable to quick convergence resulting from fitness proportionate selection.

1.1.1.1.4 Tournament Selection

Fitness ranking selection is a potentially time-consuming procedure since it needs to sort the entire population. As a noisy version of fitness ranking selection, tournament selection dramatically decreases the computational cost. It runs a *tournament* among a group of g ($1<g<N$) individuals chosen at random from the population and selects the best one for mating.

Selection pressure can be easily adjusted by changing the tournament size. If the tournament size is 1, the selection is completely random. If the tournament size is larger, weak individuals have a smaller chance to be selected.

2.1.2.2 Crossover

Crossover, also termed reproduction or recombination, mimics biological recombination between two single-chromosome organisms. In this process, pairs of parental chromosomes exchange their genetic material and generate new offspring. Different crossover techniques exist, depending on the structure of the chromosomes. In this section, several popular crossover approaches are introduced.

1.1.1.1.5 Single Point Crossover

In single point crossover, a crossover point is selected at random. All genes beyond that point in the parent strings are swapped between the two parents:

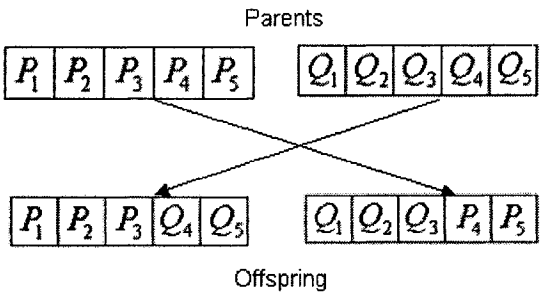


Fig. 2-5 Single point crossover

The single point crossover is conceptually simple. However, it has some shortcomings. According to Eshelman, Caruana, and Schaffer [39], it suffers from *positional bias*, that is, strong dependence of creation or destruction of schemas[1] on the location of the crossover point. For example, it can not generally combine instances of $P_1{*}{*}{*}P_5$ and $**Q_3**$ to generate $P_1{*}Q_3{*}P_5$. And it tends to destroy schemas with long defining length.

1.1.1.1.6 Two Point Crossover

Two point crossover selects two crossover points at random and exchanges the segments between them. Fig. 2-6 illustrates this process. Two point crossover reduces positional bias to some extent in that it is less likely to disrupt schemas with large defining lengths and can combine more schemas than single point crossover.

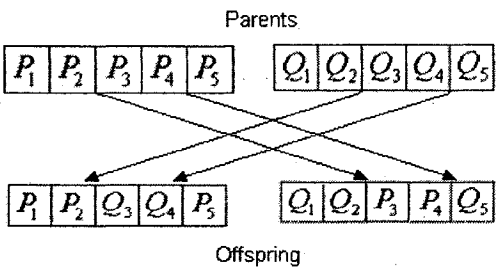


Fig. 2-6 Two point crossover

[1] For an explanation of the concept of schemas, please refer to Section 2.1.3.1 for details.

Note that two point crossover does not completely remove positional bias since it is still unable to combine some schemas.

1.1.1.1.7 Uniform Crossover

Uniform crossover makes every locus a potential crossover point. Each pair of individual genes in both parents is swapped with a probability (e.g. 0.5). As demonstrated in Fig. 2-7, pairs of genes at the 1st and the 4th locus are exchanged. All other genes remain unaltered in their parents.

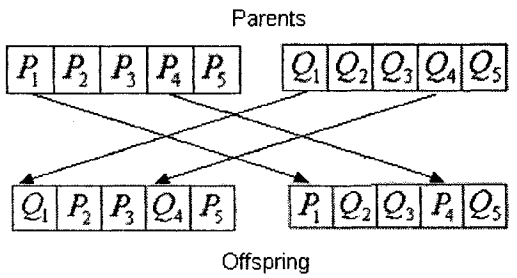


Fig. 2-7 Uniform crossover

Uniform crossover does not have positional bias. However, it is highly disruptive as it may destroy any schema.

2.1.2.3 Mutation

Resulting from copying errors, mutation is a phenomenon in which single genes of parents change in offspring. Mutation can occur at each locus in a chromosome with a small probability (e.g. 0.005). Implementation of mutation may vary, depending on different encoding schemes. For example, for a binary chromosome, mutation randomly flips some of the bits in the chromosome. For a chromosome with real valued genes, the

gene to be mutated is replaced by a value randomly selected between its given ranges.

Mutation promotes the diversity of the population by avoiding fixation of gene values. Although crossover is the major source of power in GA, mutation can become more productive than crossover when the population converges.

2.1.3 How GA Works

2.1.3.1 Schemas

Holland [36] introduced the concept of schemas in 1975. In principle, schemas are building blocks that GA processes effectively under evolutionary operators, i.e. selection, mutation, and single-point crossover. A schema is a set of bit strings that can be described by a template made up of "0", "1", and "*" that represent "don't cares". All strings fitting this template are called *instances* of this schema. For example, a schema S = 1**1 has four instances: 1001, 1011, 1101, and 1111.

The number of *defined* bits (non "*" bits) within a schema is called the *order* of the schema. Its value is between [0, l], where l is the length of the string. For example, S is a schema of order 2. Schemas with higher order are more specific.

The distance between the outermost defined bits of a schema is called the *defining length* of the schema, in short, the *length* of the schema. For example, the length of S is 3, while the length of the schema ***1 is 0. The length of a schema falls between [0, l-1]. Note that it is different from the string length l since the latter is the number of bits in the schema.

2.1.3.2 Schema Theorem

The GA's behavior can be described by the increase and the decrease in numbers of

instances of given schemas in the population. The growth of the schema can be estimated by the Fundamental theorem of the GA - the *Schema Theorem* [36], which gives a lower bound on the effect of the genetic operators from one generation to the next in a classical GA. As introduced in Section 2.1.2, a classical GA consists of such genetic operators as selection, crossover, and mutation.

Let S be a schema with at least one instance present in the population at time t. Let $n(S, t)$ be the number of instances of S at time t, and let $\hat{u}(S, t)$ be the average fitness of instances of S in the population at time t. We have:

$$\hat{u}(S,t) = \frac{\sum_{x \in S} f(x)}{n(S,t)} \tag{2.3}$$

Assuming that the fitness proportionate selection is carried out, the number of offspring of an individual x is equal to its fitness $f(x)$ divided by the average of fitness in the population $\bar{f}(t)$. With selection only, the expected number of instances of S at time $t + 1$, $E(n(S, t + 1))$, is given by:

$$E(n(S,t+1)) = \sum_{x \in S} \frac{f(x)}{\bar{f}(t)} = \frac{\hat{u}(S,t) n(S,t)}{\bar{f}(t)} \tag{2.4}$$

Hence the increases or the decreases of schema instances in the population depend on $\hat{u}(S, t)$. Although the GA does not compute this quantity explicitly, it implicitly evaluates it during the evolution process.

We now take into account the destructive effects of crossover and mutation. Let P_c be the probability that a single-point crossover will be applied to an individual. Assuming that an instance of S is picked to be a parent, S is said to *survive* under single-point crossover if one of the offspring is also an instance of S. Crossovers occurring within the defining length of S may destroy S. Hence an upper bound on the probability

that S will be destroyed is $P_c \frac{d(S)}{l-1}$, where $d(S)$ is the defining length of S, and l is the length of S. The lower bound on the probability of survival $P_c(S)$ can then be obtained by subtracting this value from 1:

$$P_c(S) \geq 1 - P_c \frac{d(S)}{l-1} \tag{2.5}$$

It can be seen that the probability of survival under crossover is higher for schemas with shorter defining length.

Let P_m be the probability of any bit being mutated. For each bit, the probability that this bit will not be mutated is 1 - P_m. Then $P_m(S)$, the probability that a schema S will survive under mutation, is equal to $(1 - P_m)^{o(S)}$, where $o(S)$ is the order of S. Obviously $P_m(S)$ is higher for lower-order schemas.

The disruptive effects of both crossover and mutation are:

$$\begin{aligned} E(n(S,t+1)) &\geq \frac{\hat{u}(S,t)n(S,t)}{\bar{f}(t)} P_c(S)P_m(S) \\ &\geq \frac{\hat{u}(S,t)n(S,t)}{\bar{f}(t)}(1-P_c\frac{d(S)}{l-1})(1-P_m)^{o(S)} \end{aligned} \tag{2.6}$$

This inequality gives an insight into how GA works and why it can sample big search spaces very fast. Short, low-order schemas whose average fitness remains above the mean fitness of the population will receive exponentially increasing numbers of samples over time, since the number of samples of these schemas increases by a factor of $\hat{u}(S,t)/\bar{f}(t)$ at each generation.

These schemas serve as building blocks that can be recombined, via crossover, into instances with increasingly higher order and higher observed fitness. This process is known as the *Building Block Hypothesis* [40], which describes the constructive effects of

crossover.

2.1.4 The Appeal of GA

GA's power lies in the following six aspects:

- It imitates the process of natural selection by conserving the instances of such schemas – the stronger the individuals, the higher probabilities they are assigned for survival, reproduction and propagation;
- It is not only able to conserve good candidates, but also to create even better candidates by effectively recombining instances of good schemas;
- It allows mutation of individuals to explore new unvisited areas in the search space. As a result of these genetic operators, it strikes a good balance between exploration and exploitation of the search space by focusing its search on promising regions, while still actively exploring other unvisited regions;
- GA's inherent properties easily enable its implementation on distributed processors so that super-linear speed up can be achieved [41];
- The implementation of a GA is conceptually simple: population evolves by means of random variation (recombination, mutation, and other operators), followed by natural selection, which picks up the fittest to survive and reproduce.

2.1.5 Limitation of a Classical GA

Despite its intrinsic merits, a classical GA has its own limitation on multimodal optimization problems. It typically manipulates one population, which, driven by selection pressure, finally converges to one winner. If there is more than one equivalently good optimum, a random winner is obtained. This phenomenon is obviously undesirable

for multimodal optimization. There have been quite a few studies to address this problem. In Section 2.2, we will discuss these GAs hierarchically and systematically.

2.2 Genetic Algorithms for Multimodal Search

As introduced in Section 0, a classic GA typically manipulates one population, which tends to converge to a random winner if more than one optimum exists. This phenomenon is obviously undesirable for multimodal problems. In the literature, there have been many enhanced versions of GAs tackling this problem. In this Section, these methodologies will be hierarchically categorized and studied.

Multimodal GAs can be broadly divided into two major categories: sequential GAs and parallel GAs. The main sequential GA, called *Sequential Niche Technique* [19], runs a simple GA iteratively and extracts the optima one at a time. The parallel GA, on the other hand, manages explicit or implicit clusters of individuals in a parallel fashion, with each cluster gathering around a potential optimum. More than one optimum may be located simultaneously.

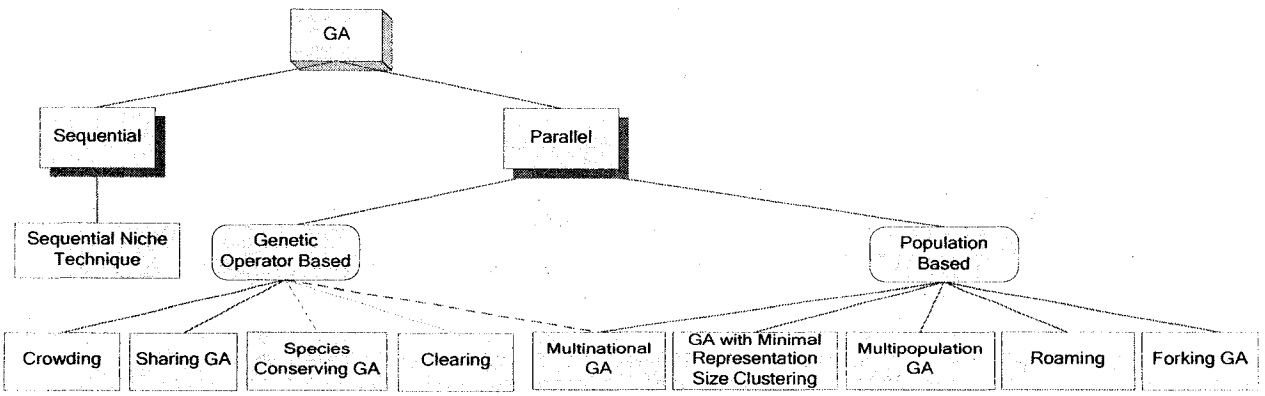


Fig. 2-8 Hierarchy of multimodal GAs

As demonstrated in Fig. 2-8, the parallel GAs can be further divided into two sub-

categories: the genetic operator based techniques and the population based techniques. In essence, a genetic operator based technique uses specially designed genetic operators to maintain implicit clusters, and the population is evolved as a whole. On the other hand, a population based approach manipulates a number of explicit subpopulations, each of which is evolved independently.

Crowding [37], Sharing GA [20, 22, 24, 30, 42-44], Species Conserving GA [21] and Clearing [45] are typical genetic operator based GAs. On the other hand, GA with Minimal Representation Size Clustering (GA-MRSC) [11, 12], Multipopulation Genetic Algorithm (MPGA) [25], Roaming [23, 46] and Forking GA (FGA) [27] are population based GAs. Ursem's Multinational GA (MGA) [1] is a peculiar case. Although it is a population based GA, it also bears some properties of a genetic operator based GA. Hence as shown in Fig. 2-8, a dashed line links MGA to the genetic operator based GA. All these GAs will be discussed in detail below.

2.2.1 Sequential GAs

The Sequential Niche Technique [19] is a typical sequential GA. It iterates a traditional GA and searches for one optimum at each iteration. In order to avoid repeated search within previously visited areas, individuals in the vicinity of a discovered optimum are punished by a fitness derating function. The raw fitness of these individuals is multiplied by the fitness derating function, which results in a reduction of the fitness of these individuals.

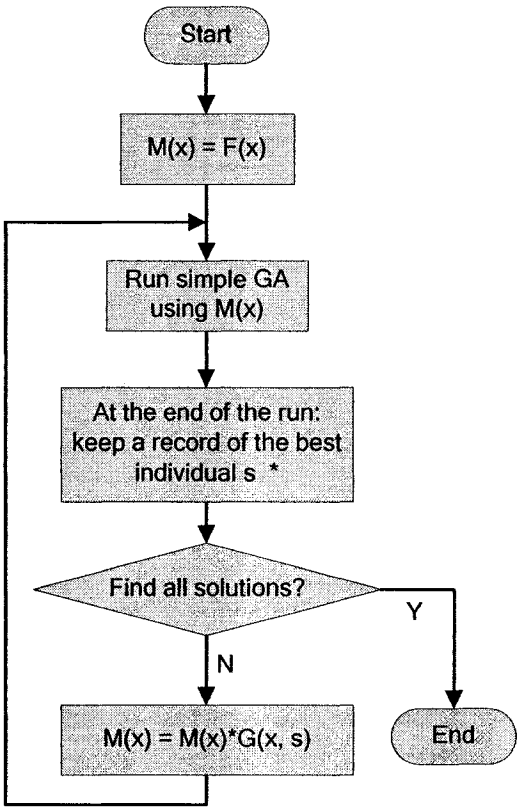


Fig. 2-9 The Sequential Niche Technique

Fig. 2-9 presents the basic framework of this algorithm. F(x) is the raw fitness function. M(x) is the modified fitness function, obtained from F(x) multiplied by a series of fitness derating functions G(x, s). Deciding when to stop a GA run and start a new iteration is not a trivial task (marked by * in Fig. 2-9). The authors suggested several techniques. They record the average fitness of the population over *h* generations. If, at any generation, the average fitness of the population is not greater than that of the population *h* generations earlier, the population is deemed to be uniform and cannot be improved any more. Consequently, the run is terminated. A run is also terminated if the raw fitness of the best individual s is greater than a solution threshold or a maximum number of generations are reached.

The first approach above may cause premature termination of a run. In this case, no optimum will be found. If the termination of a run is judged by the goodness of the best individual found, the underlying assumption is that the target or the solution is approximately known, which, without *a priori* knowledge of the landscape, is usually unavailable.

The Sequential Niche Technique assumes a known number of optima. The algorithm terminates when all of them have been found. It also assumes that the optima are evenly distributed throughout the search space in order to estimate a niche radius r. Each optimum on the function's landscape is hypothetically surrounded by a hyper-sphere of radius r. Individuals falling in the hyper-sphere that is centered at a found optimum are derated. Unfortunately, both these two assumptions do not normally hold.

Even worse, the fitness derating function itself does not guarantee the elimination of redundant search. It may also cause loss of optima or phantom optima because it arbitrarily alters the landscape of the function. In [18], the author observed that the sequential GA performed worse than a parallel hill climber; whereas the parallel niching GA (Sharing GA [20]) outperformed the sequential GA and achieved the best stability on problems of different complexity.

2.2.2 Parallel GAs

Most multimodal GAs adopt a parallel scheme [2, 9-11, 20, 22, 23, 30, 46-56]. They strive to maintain diversity in the population by explicitly or implicitly managing different sub-populations or species in parallel. Each species explores the area of a potential optimum. The availability of parallel and/or distributed computers makes such implementations at least as efficient as sequential GAs.

The crux of these approaches is diversity, that is, the population has to be diversified enough to cover as many promising regions as possible. Based on the method of maintaining diversity in the population, parallel GAs can be further divided into two sub-categories: genetic operator based techniques and population based techniques.

In essence, a genetic operator based GA uses specially designed genetic operators to promote diversity and prevent convergence to a single optimum. It typically evolves a global population of fixed size, which may contain one or more variable sized species. Mating and selection are applied globally to the whole population.

Unlike the genetic operator based approaches that maintain diversity *via* specific genetic operators, a population based GA consolidates diversity explicitly with multiple subpopulations. Each subpopulation is evolved independently toward its own potential optimum. Evolutionary operations (e.g. elitism, mating and mutation) are typically applied locally within each subpopulation.

2.2.2.1 Genetic Operator Based Techniques

1.1.1.1.8 Crowding

Jong [37] proposed a crowding model that attempts to distribute individuals evenly among niches. This model is motivated by a biological phenomenon that similar individuals within the same species compete for limited resource, whereas different individuals in different species do not tend to compete.

Crowding follows a standard GA except that only a fraction of the population dies and is replaced by new offspring in each generation. A newly generated individual replaces an individual that is most similar to it among a pool of individuals randomly

selected from the population. The size of this pool is called a *crowding factor*. When the population reaches its equilibrium, the number of individuals within each niche is expected to be stable.

Crowding is able to maintains existing diversity, but is unable to generate new species [57]. It exhibits mediocre performance on multimodal problems due to its replacement errors [57], [48], i.e. an individual is replaced by another individual that is not the same species. Thomsen suggested that a higher crowding factor (e.g. one equals to the population size) could avoid replacement errors [58]. However, when optima are close to each other, the replacement errors may still occur since a new individual may replace another similar individual that belongs to a different species.

Moreover, a higher crowding factor may degrade the performance [37, 57]. It was demonstrated in [57] that Crowding with a crowding factor equal to the size of the population exhibited little, if any, selection pressure on a simple unimodal function. That is not unexpected since Crowding consistently replaces the closest individual in the population, which is very likely to bear a similar fitness value too. The consequence is little or no improvement for the population.

Such a high crowding factor also limits parallelism and adds an order of complexity to the algorithm [57]. The complexity in this case is similar to that of the standard Sharing, which also cycles through the whole population in order to compute the shared fitness for each individual.

Mahfoud [57] improved standard Crowding by introducing a method called Deterministic Crowding. In this algorithm, the population of size N is randomly paired into $N/2$ pairs of individuals for crossover. Offspring generated then compete with

parents for replacement. The offspring replace their closest parent if they are of greater fitness. The closeness is computed based on the average distance between the pairs of parent-child. As shown in Fig. 2-10, the procedure is repeated for $N/2$ times.

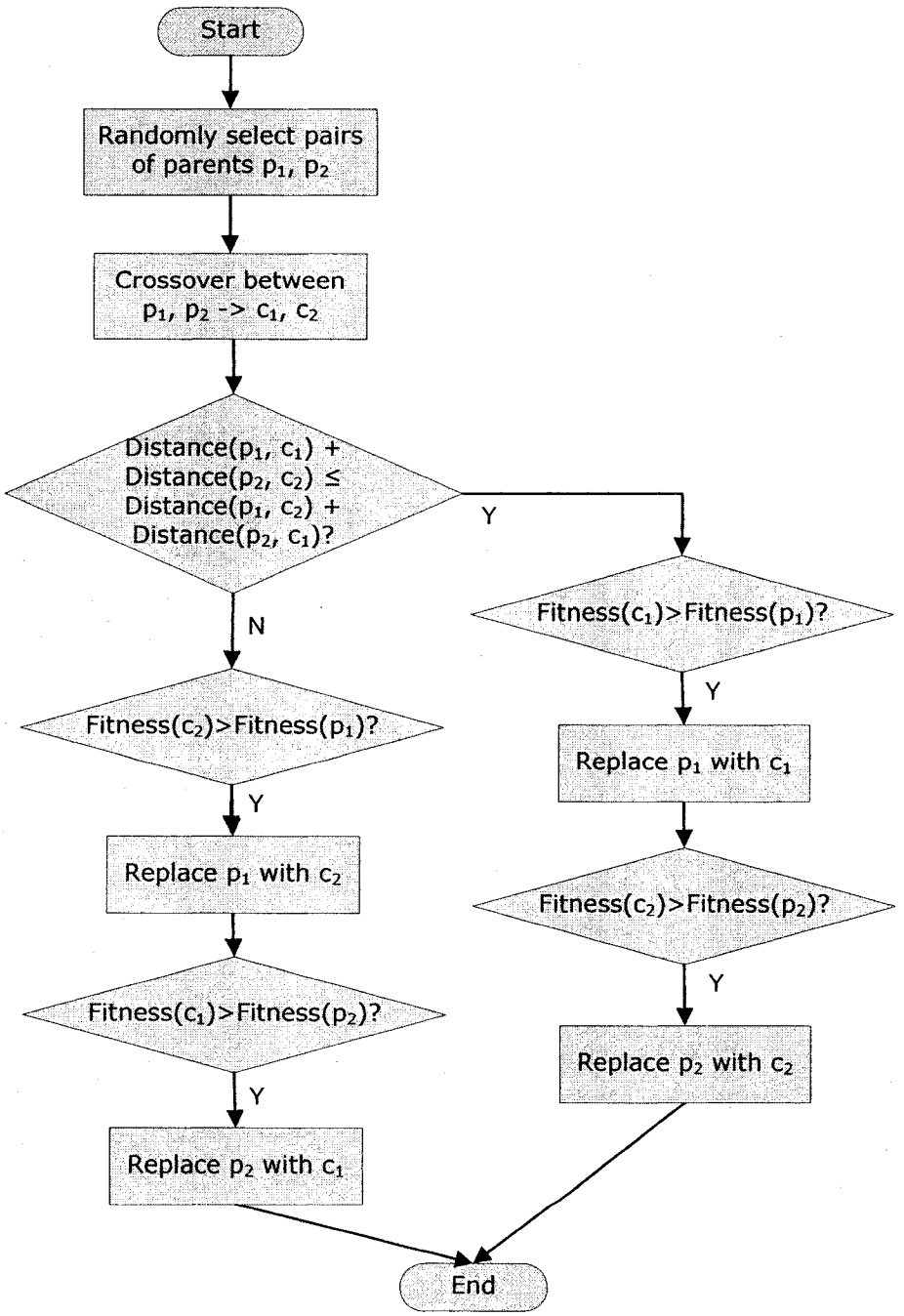


Fig. 2-10 Deterministic Crowding

Deterministic Crowding uses a distance measure to determine similarity between individuals. But it does not require the use of a radius. This relaxes the requirement of *a priori* domain knowledge. However, Deterministic Crowding always prefers higher fitness individuals over lower fitness individuals. This finally leads to a loss of niches, and hence decreased diversity of the population.

1.1.1.1.9 Sharing

Sharing GA (SGA) was first proposed by Goldberg and Richardson [20]. It attempts to maintain the diversity of the population by altering the fitness of potential optima to approximately the same level. Sharing proposed a concept of niches. A niche is a group of individuals within the same potential optimum area. In each niche, the raw fitness f_i of an individual i is scaled in the following manner:

$$f_{sh,i} = \frac{f_i}{m_i} \tag{2.7}$$

where m_i is a niche count. In standard Sharing, m_i is given by summing the sharing function $sh(d_{ij})$ of i:

$$m_i = \sum_{j=1}^{N} sh(d_{ij}) \tag{2.8}$$

where d_{ij} is the phenotypic or genotypic distance between two individuals i and j. N is the total number of individuals in the population. The sharing function, $sh(d_{ij})$, is defined as:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^{\alpha} & if\ d_{ij} < \sigma_{share} \\ 0 & otherwise \end{cases} \tag{2.9}$$

Here σ_{share} is a niche radius used to separate niches, and α is a constant to

configure the shape of the sharing function (and is usually set to 1). σ_{share} is assigned a value based on the estimated number of optima in the domain, which, unfortunately, is usually unavailable.

As can be seen from equation (2.7), in SGA, the fitness is shared among similar individuals. Since in GA, stronger individuals will have higher probabilities to survive and hence tend to generate more offspring, the fitness of such individuals is decreased in dense neighborhood. Thus, population diversity is maintained by encouraging exploration of less crowded regions of the fitness surface.

The standard Sharing GA suffers from a high time complexity of $O(N^2)$, incurred by the computation of the sharing function $sh(d_{ij})$ (equation (2.8) and (2.9)). The applicability and usefulness of the standard sharing scheme are hence severely restricted.

One of the major improvements of Sharing GAs is the introduction of cluster analysis [2, 24, 28, 30, 59]. Individuals are clustered into different groups based on their location in the search space. When computing the sharing function value of an individual, the individual is compared to the centre of each cluster, rather than all other individuals in the population. Hence the overall cost is reduced to $O(NK)$, where K is the number of clusters, ideally, the number of optima present.

Yin and Germay [30] used McQueen's Adaptive KMEAN algorithm to perform cluster analysis. The fitness of each individual i is computed using equation (2.7), with the niche count m_i given by:

$$m_i = n_i - n_i\left(\frac{d_i}{d_{\max}}\right)^{\alpha} \tag{2.10}$$

where α is a constant, n_i is the number of individuals in the niche that individual i

belongs to; d_i is the distance between individual i and the center of the cluster; and d_{max} is the maximum radius of the cluster.

A new cluster is created around an individual i if the minimum distance between i and the centers of all the existing clusters is greater than d_{max}. Two clusters are merged if the distance between their centers is smaller than another threshold d_{min}.

Yin and Germay's approach effectively reduces the computational cost. Furthermore, it does not make any assumption about the number of optima. Although an initial estimate of this number is required, it does not significantly influence formation of actual clusters, as clusters are formed based on the distribution of individuals. Another improvement worthy of note is that this method enhances the stability of existing niches by implementing mating locally within each niche, rather than globally within the whole population, as is the case in the standard Sharing GA.

Nevertheless, this approach requires estimates of two parameters: d_{max} and d_{min}, which play similar roles to σ_{share} in standard Sharing. These two parameters are vital factors for success.

In some other algorithms, the niche count is simply the size of the niche [2, 24, 28, 59], that is, n_i in equation (2.10) above.

Miller and Shaw proposed a *Dynamic Niche Sharing* (DNS) technique [24]. It identifies dynamic niches using a greedy approach, as presented in Fig. 2-11. The size of the population is N. The dynamic niche set D contains the centers (up-to-date best individuals) of the niches. An individual belongs to one of these niches if it is within σ_{share} of the niche's center. Otherwise this individual belongs to a "non-peak" category. A "non-peak" individual still has a niche count, which is computed using equation (2.8)

above as in standard Sharing. DNS assumes that the number of optima is known and that any and all pairs of optima are at least $2\sigma_{share}$ apart from each other. These two assumptions are also adopted by standard Sharing and are not always true.

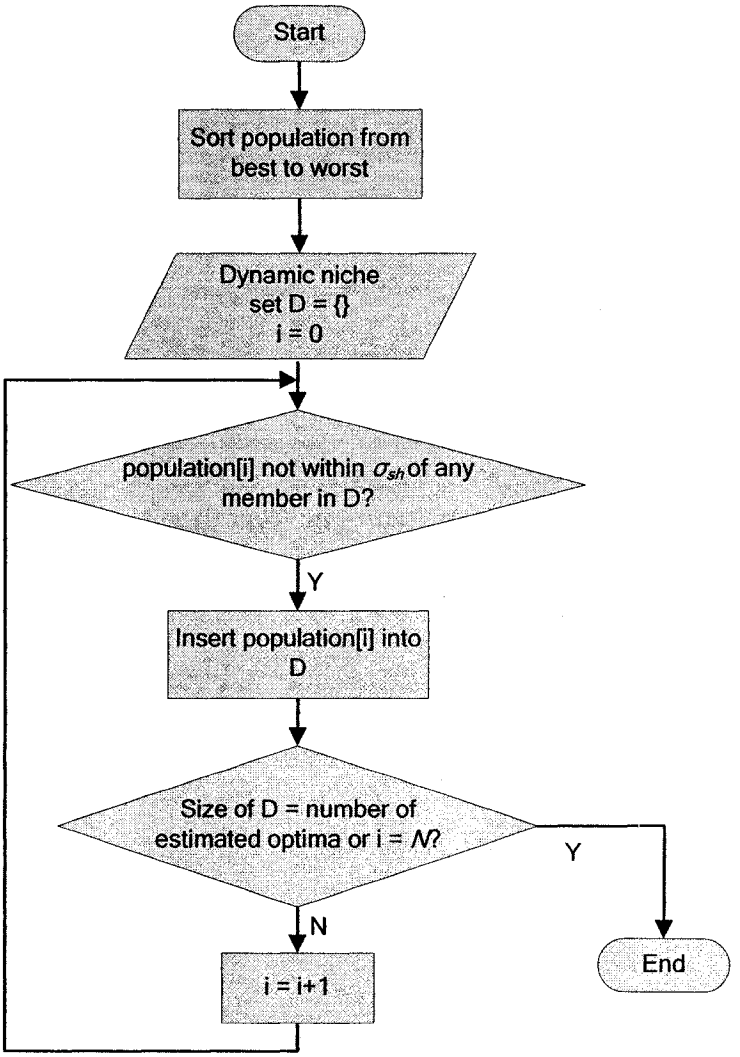


Fig. 2-11 Dynamic niche identification

Zhang, Shao and Feng [59] used a similar algorithm. They first use a DBSCAN clustering algorithm [60] to identify dynamic niches. After that, the fitness of the members within a niche is scaled by the same scaling factor – the size of the niche. They claimed that this scheme allows more thorough exploitation of the areas of potential

optima than standard Sharing. That is because the standard Sharing GA adjusts the fitness of an individual based on the distance to its neighbors, each individual has pressure to maximize the distance between itself and its neighbors. As a result, individuals will be dispersed and fewer individuals are able to populate and exploit areas of potential optima.

This algorithm also has "non-peak" individuals that do not belong to any niches. Unlike Miller and Shaw's work, which computes the niche count of the "non-peak" individuals as in standard Sharing, this algorithm simply sets the niche count of these individuals to 1. The authors explained that this scheme will give the "non-peak" individuals more chances to be selected into the next generation than the niche members that are scaled by their niche count. Consequently, it tends to attract more individuals to explore such "non-peak" areas. Therefore, this scheme makes a good trade-off between exploitation and exploration.

Song and Yu [61] proposed a hierarchical clustering based fitness sharing method. They assume that the number of optima is given. With this information, they separate individuals into niches via a hierarchical cluster tree and adapt fitness values of niche members by:

$$f_i' = \frac{f_i}{\left(\frac{nicheNum \cdot nichePop}{pop}\right)^{\alpha}} \tag{2.11}$$

where *nicheNum* is the number of clusters, *nichePop* is the size of the niche the individual *i* belongs to, *pop* is the size of the overall population size, and α is a factor to control the reduction level.

Unlike most Sharing methods that perform selection and crossover globally within the whole population, this algorithm uses special designed inner-species selection

as well as independent inner-species and inter-species crossover to improve the search ability.

Gan and Warwick [2, 28] proposed a *Dynamic Niche Clustering* (DNC) approach. Starting from N small niches (clusters) with given initial radii, it merges niches approaching the same optimum and splits niches focusing on different optima. Each niche has an independent radius, which is dynamically adjusted throughout the evolutionary process.

DNC can be implemented as follows:

1. Calculate niche members;

   Similar to other Sharing GAs, an individual belongs to a niche if it is within the radius of this niche to the center of this niche. Unlike other Sharing GAs, however, the radii of the niches are different.

   The initial radius of each niche is given by:

$$\sigma_{initial} = \frac{\lambda\sqrt{d}}{\sqrt[d]{N}} \tag{2.12}$$

   where d is the dimensionality, and λ is a constant (with a value of 1.0).

   Initially, each individual is a niche, which is centered around the individual itself.

2. Adjust the midpoint of each niche according to its members;

   Calculation of the midpoint of each niche is detailed in Section 3.2.

3. Calculate Euclidean distance between all niche pairs;
4. Sort these niche pairs with respect to their Euclidean distance;
5. Check niche pairs for merging;

Two niches i and j are merged into i if they satisfy:

$$|mid_i - mid_j| < \frac{\sigma_i}{2} \quad \text{or} \quad |mid_i - mid_j| < \frac{\sigma_j}{2} \tag{2.13}$$

where mid_x is the center of niche x, and σ_x is the radius of niche x. j is deleted from the niche set and the niche pairs list. All niche pairs that contain i are recalculated because the midpoint of i may have moved. The center of i is updated using the method introduced in Section 3.2.

6. Check all large niches for splitting;

   A niche is split if its population size is greater than 10% of the total population size and a valley is detected between a pair of randomly selected individuals i and j from within the niche. Two new niches are created with midpoints centered on these two individuals. The niche radius of each niche is calculated by:

$$\frac{|mid_i - mid_j|}{2} \tag{2.14}$$

   The members of each niche are recalculated by comparing all the individuals within the original niche and the midpoint of each of the new niches.

7. Apply sharing function within each niche;

8. Evolve the population.

Steps 1 - 6 are clustering operations aiming to identify valid clusters. Merging and splitting of niches, together with sorting of the niche pairs list, induce a great amount of extra cost to the algorithm. This negative effect manifests itself when the size of the niche pair list is large.

DNC is able to identify niches of variable radii. It also allows some overlap

between niches. These innovative facts introduce a degree of flexibility when searching complex multimodal landscapes. Additionally, unlike Yin and Germay's approach, which recalculates all clusters at every generation [30], DNC retains niches from generation to generation. This strategy significantly reduces the computational overhead.

Dilettoso and Salerno [50] incorporated the niche radius as an additional variable of the optimization problem. In this way, each individual has its own radius. The fitness of an individual is derated only if there are individuals with higher fitness values in this individual's own niche. However, as the authors themselves observed, the radius of many individuals tends to shorten quickly. The authors did propose to increase the radius of an individual if there are not any fitter individuals within the niche. However, the effectiveness of this tactic in counterbalancing the problem of shrinking radii remains an open question. After all, selection pressure tends to favor smaller radius, since in that case the fitness of an individual is shared by fewer individuals, resulting in a larger scaled fitness value.

1.1.1.1.10 Clearing

Unlike Sharing GAs that share available resources within the same species, *Clearing* [45] fully attributes these resources to dominant (best) individuals within the species. These individuals constitute the mating pool. The non-dominant individuals are simply removed from the population and do not participate in evolution.

Clearing implements a full local elitist strategy by preserving the winners with fitness values greater than the average fitness of the population before clearing. If the preservation of all the winners immobilizes a too great fraction of the population, the author suggested an alternative by preserving only the best individual of each species, as

demonstrated in Fig. 2-12.

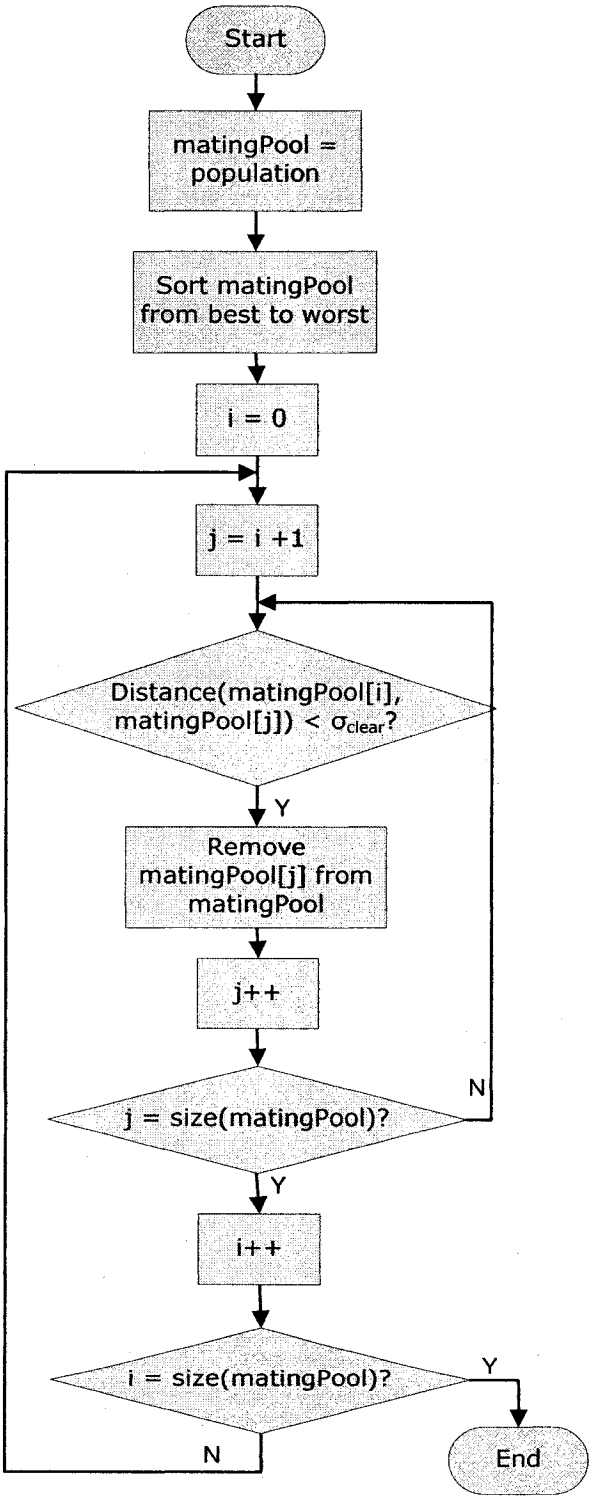


Fig. 2-12 Clearing

Like the sharing method, the Clearing algorithm uses a dissimilarity measure between individuals to determine if they belong to the same species or not. This value could be geometric distance (e.g. Hamming or Euclidian) for either genotypes or phenotypes. A threshold σ_{clear}, called the clearing radius, plays a similar role to σ_{share} in the Sharing GA.

Since the mating pool is constituted by winners from all species, Clearing carries out a global crossover scheme similar to Sharing GA. Compared to local restricted mating within species, whether mating between different species plays more constructive or destructive roles in convergence to optima is still in question.

The author theoretically analyzed advantages of Clearing over Sharing GA in [45]. Its complexity is generally lower than that of the sharing method. Additionally, in Sharing GA, if a species contains more individuals than expected due to the selection noise, each of them will have an expected number of offspring less than one in order to restore the equilibrium. When this happens with proportionate selection schemes such as Roulette Wheel Selection (RWS) or Stochastic Universal Selection (SUS) [38], an individual of such a species could not have offspring. If this is the case for all the individuals of the species, unless there is a sufficiently large population so that individuals for each desirable species have high survival probabilities, this species is exposed to extinction. Therefore, the Sharing GA demands a large population. In contrast, clearing with proportionate selection ensures that the number of offspring of the dominant individuals is always greater or equal to one if their fitness is above the average fitness of the population. Consequently, this species survives with certainty. The lower bound of the overall population size required by clearing is smaller than that of Sharing

GA. The superiority of Clearing over Sharing GA is also demonstrated in [49].

Dick [49] proposed an algorithm called Localized Clearing (LC). Rather than applying the clearing procedure globally within the whole population, this algorithm applies clearing to every deme at every location in the space. Mating is restricted between individuals within the same deme. The author demonstrated that LC outperformed globally-applied clearing on well-known and difficult multimodal problems.

LC adopts a particular spatial structure - the ring topology, on which N individuals are placed at equidistant locations. The sizes of the demes are all the same and given by a parameter. The demes overlap. An individual participates in multiple demes and undergoes the clearing process for each deme. Obviously these demes do not necessarily correspond to actual species/niches in the space. This mechanism is subject to a high risk of mix-up of different species in one deme or dispersion of the same species in multiple demes. The former will result in loss of species. The latter will induce extra cost in manipulating and maintaining redundant demes.

Stoean, Preuss, Gorunescu and Dumitrescu also proposed localized mating in [62]. Their methodology is based on a radii-based multimodal evolutionary framework - Genetic Chromodynamics [51] (GC). Each individual, denoted as c here, is only recombined with its neighboring individuals within the same mating region. If the mating region around c is empty, it is mutated to produce one offspring in such a way that the offspring is still kept in the mating region of c.

After recombination and mutation, the population is subject to a procedure similar to Clearing. This procedure, called merging in [62], removes all but the best individual within the same merging region. Merging always leads to decreasing population sizes.

Species crowding in different basins of attraction become smaller and smaller. Finally only one individual is left for each species and undergoes a hill climbing strategy via mutation.

This method is similar to Clearing in that a winner dominates a species. However, the authors also pointed out their differences:

- GC performs crossover only between individuals belonging to the same mating region, whereas clearing performs global recombination;
- GC sets different radii for the identification and maintenance of niches in the population, whereas Clearing utilizes a single radius;
- GC consistently decreases population size, whereas in Clearing the population size remains constant.

GC, which can be deemed as a hybridization of genetic operator based GA and a population based GA, is intriguing and potentially promising. However, this approach entails radii for three regions - besides the mating region and the merging region discussed above, there is also a replacement region, within which the worst individual is replaced by a new offspring if it is even worse than this offspring. As the authors themselves admitted, it is often difficult, or even impossible, to find these values, especially when the sizes of basins of attraction have large variance.

Singh and Deb proposed another interesting algorithm that makes Clearing more effective and reliable [3]. In original Clearing, inferior individuals are of no use since they are cleared and do not participate in evolution. However, they still occupy population slots. That is a waste of resources. Instead in [3], these individuals are reallocated outside the range of their respective best individuals. After that Clearing is

performed again. This action is equivalent to drastically mutating all bad individuals. Despite an extra computational cost, this method introduces more diversity into the population and leads to the exploration of more interesting areas in the search space.

The modified Clearing has all strengths of the original Clearing. However, it has its drawback too, that is, the dissimilarity between individuals is measured by their inter-distance, which is thresholded by a single global threshold.

1.1.1.1.11 Species Conservation

Li, Balazs, Parks and Clarkson introduced a Species Conserving GA (SCGA) [21]. This method is based on a concept of distributed elitism. It divides the population into several species according to their similarity. Each of these species is constructed around a dominant individual, called the species seed.

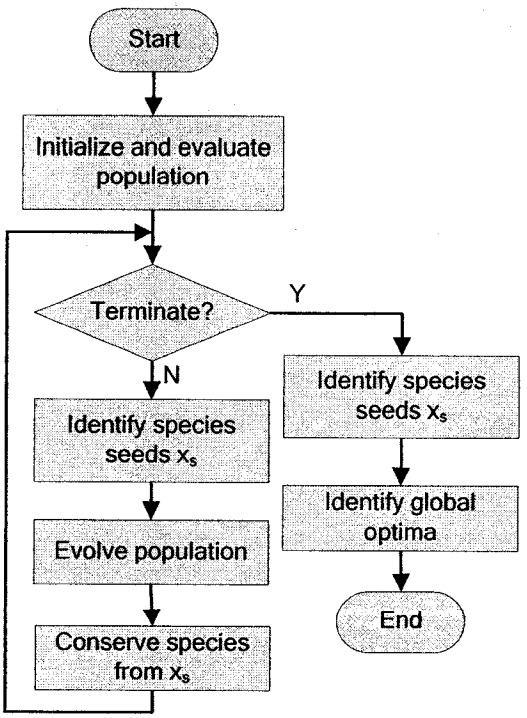


Fig. 2-13 Species Conserving GA

Similar to Clearing in 1.1.1.1.10, species seeds found in each generation are conserved and passed through to the next generation. A notable difference between these two algorithms is that unlike Clearing, which excludes all individuals, except for winners of each species, from evolution, SCGA evolves all individuals as a traditional GA.

As presented in Fig. 2-13, once the species have been found, the population is evolved using usual genetic operators, i.e. selection, crossover and mutation. Since some species may not survive following these operations, the authors used a species conserving process to copy these species into the new population. Obviously, the difference between SCGA and a classic GA is the introduction of two processes: the identification of seeds and the conservation of species. The authors claimed that the additional overhead associated with these two processes is no higher than that of standard fitness Sharing.

In SCGA, the process of identification of species seeds is similar to the process of dynamic niche identification in Dynamic Niche Sharing [24] (Section 1.1.1.1.9). Readers can refer to Fig. 2-11 for details.

Fig. 2-14, on the other hand, shows the process of the conservation of species. For each species seed $x \in x_s$ (the list of species seeds), the population is searched for individuals belonging to x. x then replaces the worst individual found, if x is better than the latter. If, however, no individual belongs to x, x simply replaces the worst unmarked individual in the population. With this method, all species are guaranteed to survive.

The last step of SCGA (see Fig. 2-13) is to identify the global optima. Since trivial but significantly different individuals could also be conserved, an individual x is identified as an optimum if the following inequality is satisfied:

$$f(x) \geq (f_{\max} - f_{\min}) \times r_f \tag{2.15}$$

where f_{max} is the fitness of the best species seed, f_{min} is the fitness of the worst individual in the final population, and r_f is a solution acceptance threshold ($0 < r_f \leq 1$). r_f needs to be carefully set as its inappropriate value may cause false positive or negative results.

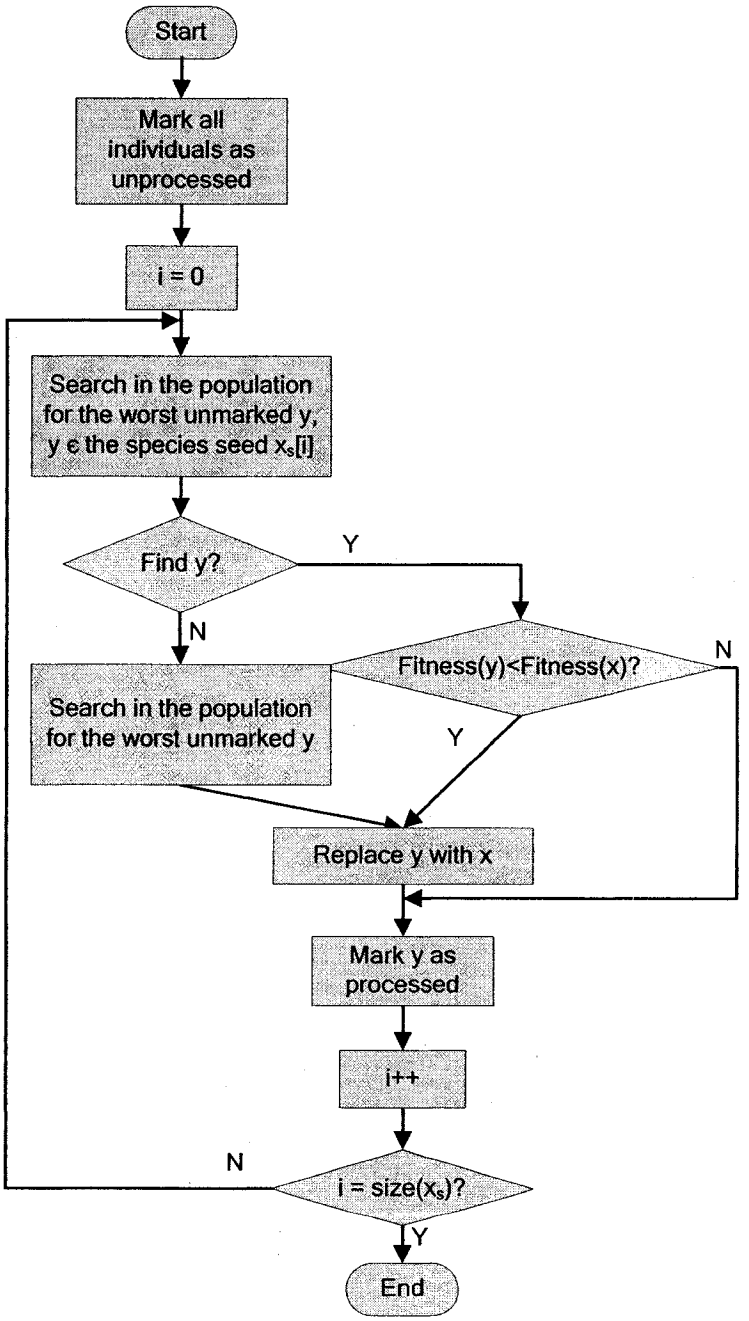


Fig. 2-14 Conserving species in SCGA

SCGA relies on an important parameter σ_s, called *species distance*, to separate different species. An individual y belongs to a species seed x if distance between x and y is less than $\sigma_s / 2$. The role of σ_s is like that of σ_{share} in sharing GA and *clearing radius* in Clearing discussed above. Both of them are crucial global thresholds and are based on an assumption that all optima are equidistant to each other. This assumption is unrealistic and problematic.

In summary, SCGA is a typical genetic operator based GA. It enforces elitism on the species level to preserve diversity. The only difference between this technique and a classical GA is the introduction of two processes: the selection of seeds and the conservation of species.

2.2.2.2 Population Based Techniques

1.1.1.1.12 Multinational GA

Multinational GA (MNGA) was proposed by Ursem [1]. It explicitly maintains and evolves a number of nations. Each nation corresponds to a promising optimum area in the search space. Mating is restricted locally within individual nations. Selection is performed either globally (weighted selection) or locally (national selection).

As shown in Fig. 2-15, the algorithm starts from a single nation, which contains all individuals in the population. Following that it enters a loop to separate individuals into different nations and evolve the nations until termination.

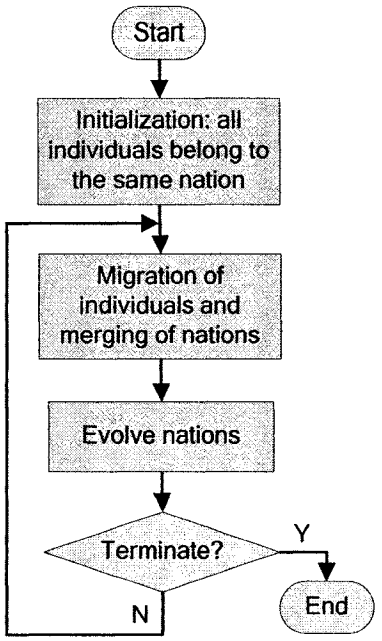


Fig. 2-15 Multinational GA

In MNGA, each individual belongs to exactly one nation. Individuals may, however, migrate from one nation to another. Nations may be merged if they are evolved toward the same optimum. The timing of these actions is determined by the relationship between individuals, which is identified by a Hill-Valley function. This function is based on the topology between concerned individuals, rather than the distance between them. The Hill-Valley method has been proved to outperform traditional distance based approaches [2, 58].

There are two selection schemes in MNGA: weighted selection and national selection. Weighted selection is performed globally, whereas national selection is performed locally. In weighted selection, the fitness of each individual is scaled by the number of individuals within its nation. This behavior resembles the fitness sharing scheme in Sharing GA. Hence, MNGA with weighted selection is virtually a genetic

operator based GA. This is indicated by a dashed line in Fig. 2-8.

On the other hand, in MNGA with national selection, individuals only compete with other individuals from the same nation. MNGA with national selection is therefore a typical population based GA, which, as its name suggests, explicitly evolves a set of nations independently, with evolutionary operations, such as selection and mating, performed locally.

Whichever selection approach MNGA adopts, its mating is always restricted between chromosomes of the same nation. This is an important characteristic of population based GAs.

1.1.1.1.13 Minimal Representation Size Clustering

Hocaoglu and Sanderson [11, 12] developed a multiple-population GA to identify different species (Fig. 2-16). A simple GA is used to evolve each species separately. Mating is not always restricted within the same species. Rather, cross-species mating occurs periodically at the $2^k \times CI_{\text{th}}$ generation, where k is an incremental integer starting from 1, and CI is a constant called cluster interval. After evolution, all species are merged into a pool for a *Minimal Representation Size Cluster* (MRSC) analysis, again, at the $2^k \times CI_{\text{th}}$ generation. Individuals are then distributed into a new set of species, which are passed into the new loop for evolution.

Hocaoglu and Sanderson's approach has the following benefits:

- Mating between different species may be useful when the function's landscape has plateaus where pseudo species may be formed;
- Clustering analysis is performed occasionally with a decreasing frequency, rather than at each generation in most other algorithms [20, 22, 24, 30, 42-44],

hence the computational cost is dramatically decreased;

- Unlike many algorithms that assume *a priori* knowledge of the number of potential optima [19, 20, 40], the MRSC analysis does not need this assumption.

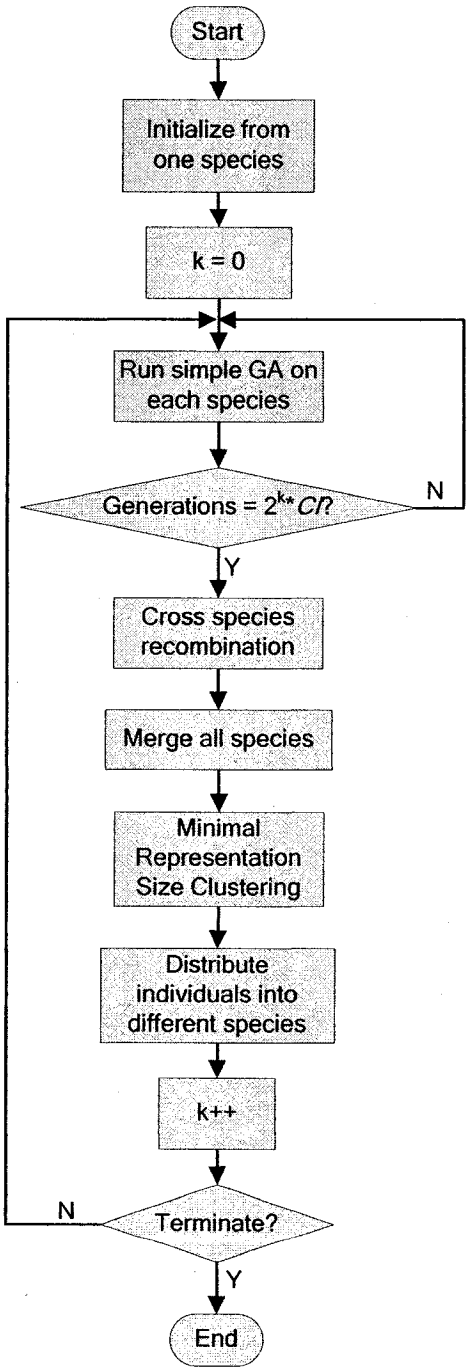


Fig. 2-16 GA with Minimal Representation Size Clustering

A potential problem of MRSC is that it assumes a Gaussian distribution of species. Although the authors claimed good performance even when the actual species are not normally distributed, this assumption is still problematic and needs to be further justified.

1.1.1.1.14 Multipopulation GA

Siarry, Pétrowski and Bessaou [25] proposed a Multipopulation Genetic Algorithm (MPGA) that divides a traditional GA into a sequence of two processes: exploration (diversification) and exploitation (intensification). Exploration locates areas of potential optima. A speciation method forms species (subpopulations) around potential optima. After that, exploitation allows intensification in the detected areas by allocating a separate portion of the search space and applying a simple GA to each species. The processes of exploration and exploitation are repeated following a particular regime.

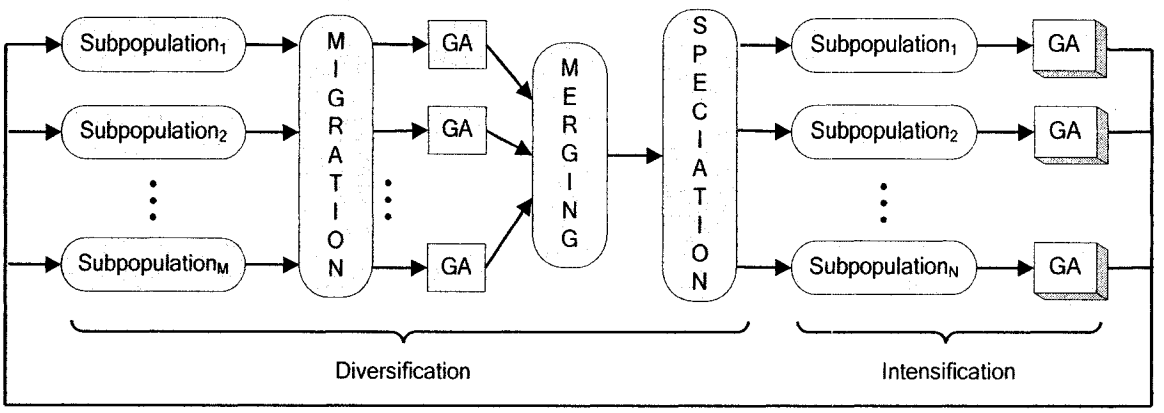


Fig. 2-17 Diversification and intensification in Multipopulation GA

As shown in Fig. 2-17, the diversification process consists of several procedures. Firstly, random individuals migrate between different subpopulations. Individuals to be migrated and individuals to be replaced are selected at random, excluding the best individuals within each species. Following that, a simple GA is applied to each

subpopulation for one generation. These two procedures together may prevent premature convergence of species and generate new individuals belonging to unexplored areas. Next, all subpopulations are merged and a speciation method forms species (subpopulations) around each optimum in the exploration process.

The authors used a speciation tree [63] to form species because of the following reasons:

- Its computational complexity is $O(n\log n)$;
- It only needs two individuals to identify an optimum;
- It does not need to estimate the niche radius – a problem dependent parameter.

A normalized Euclidean distance is used to measure similarity between individuals.

After diversification, intensification consists in applying a GA independently on each species for a fixed number of generations.

1.1.1.1.15 Roaming

Roaming [23, 46] (Fig. 2-18) is based on the novel concept of subpopulation stability. It identifies optima using isolated subpopulations, which are characterized as stable or unstable using a stability measure *SM*. An unstable subpopulation evolves in isolation until it becomes stable, that is, when a potential optimum is found within it. This subpopulation then roams toward other regions of the search space by mutating all its members with a probability 1.

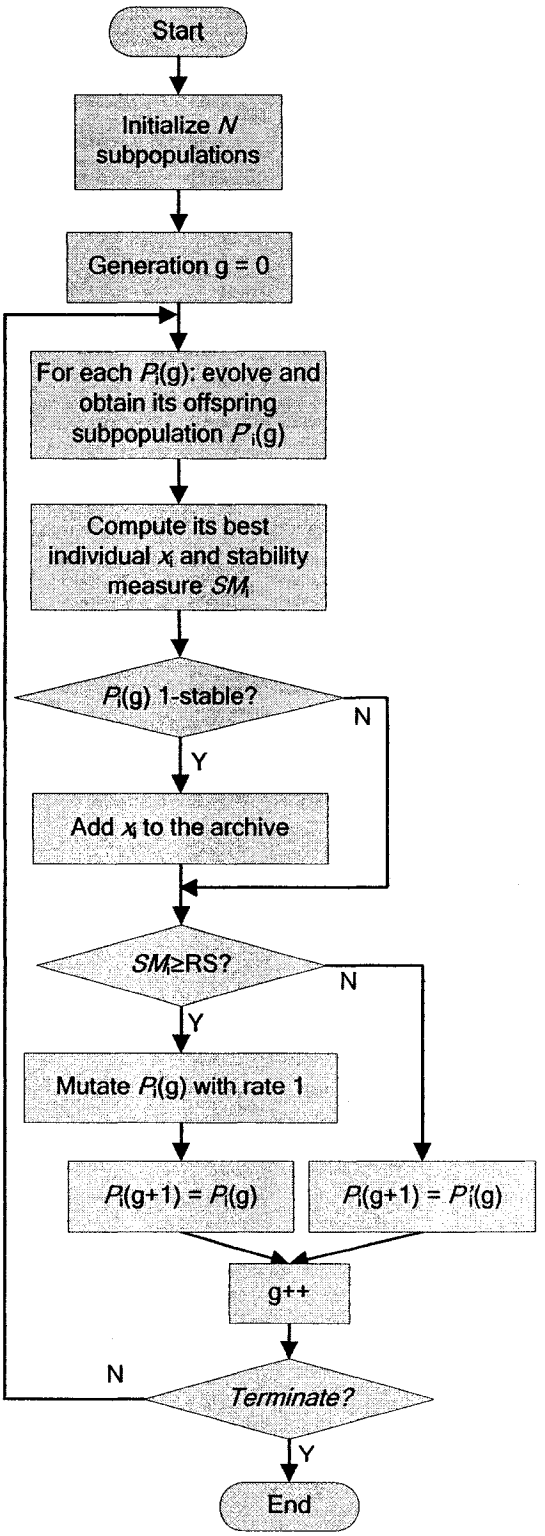


Fig. 2-18 Roaming

SM of a subpopulation P_i is defined as:

$$SM_i = 1 - \frac{card(B(x_i))}{card(P_i)} \tag{2.16}$$

where x_i is the best individual in P_i; $B(x_i)$ is the set of individuals in the offspring subpopulation of P_i that are better than x_i; *card*(A) is cardinality of a set A.

The value of SM_i is between [0, 1]. If it is 1, P_i is called 1-stable and x_i is a potential optimum, which is stored into an external population called the archive. P_i is called σ-stable if $SM_i \geq \sigma$.

As shown in Fig. 2-18, given a threshold RS, if SM_i of a subpopulation P_i is RS-stable, P_i is deemed a roaming subpopulation. It roams toward other regions of the search space by mutating all its members with a probability 1. There is no interaction between subpopulations at any stage.

One significant advantage of Roaming is that the number of subpopulations is *not* dependent on the expected number of optima – a value which is usually unobtainable without *a priori* information about the landscape. Instead, this number is a pre-defined parameter of the algorithm. This scheme confers flexibility and robustness on the algorithm.

Roaming of all RS-stable subpopulations is problematic too. As long as RS is less than 1, the subpopulations that have not reached their 1-stability are forced to migrate and deviate from their original directions. All previous efforts in evolving these subpopulations are fruitless. This leads to a serious waste of computational resources.

It is also noticed that there is no clustering behavior in this algorithm. This may reduce its cost. However, a subpopulation is very likely to cover different areas of

interest. Since any stable subpopulation is subject to roaming with strong mutation of 100% probability, many promising individuals are also altered. As a result, all previous efforts to evaluate and evolve these individuals are rendered fruitless.

Another problem comes from the addition of a solution x into the archive. This happens when x is a new local optimum or is better than another optimum that is already in the archive and in the same region. For each individual a in the archive, the global minimum of the fitness function within x and a is calculated. If the minimum indicates there is a *valley* between x and every a, x is added. Otherwise if x is better than a, x is also added. This method does eliminate the need to evaluate the distance between individuals. However, to compute the global minimum within a region in the search space is hardly achievable.

1.1.1.1.16 Forking GA

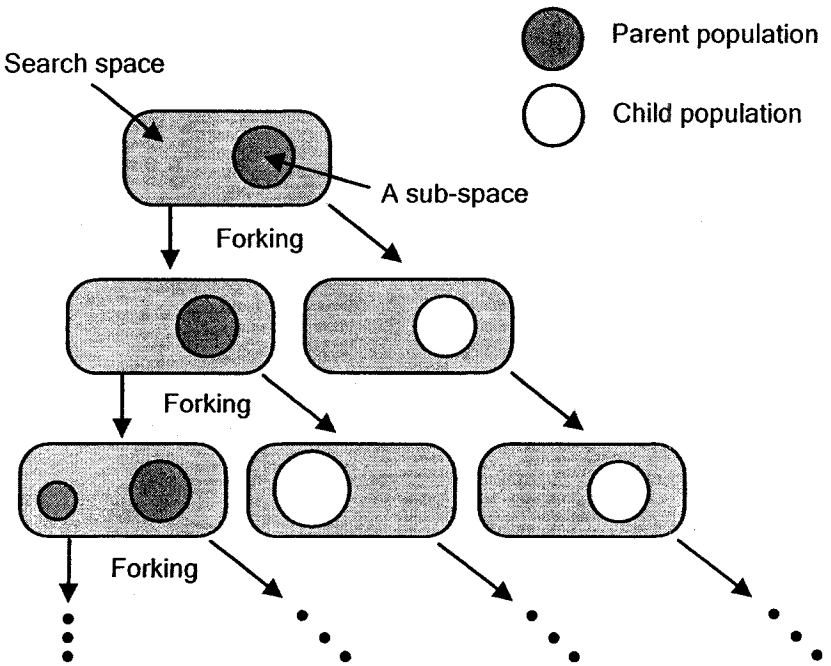


Fig. 2-19 Population forking

Forking GA (FGA) [27] uses a multi-population scheme, which involves one parent population that explores one subspace and one or more child populations that exploit other subspaces. A population is forked into different subpopulations (a parent population and child population(s)) if the diversity of the population collapses or if its fittest individual remains unchanged for a pre-set number of generations. The whole search space is then divided into non-overlapping subspaces, with each subpopulation responsible for searching each of them independently.

2.2.3 Summary

2.2.3.1 Sequential GAs versus Parallel GAs

As discussed in Section 2.2.1, the largest advantages of a Sequential GA are its conceptual simplicity and relative efficiency [19]. However, Mahfoud [18] observed that the Sequential GA was not necessarily faster than a parallel GA. It failed to solve hard problems with a vast number of peaks. A possible reason is that once the sequential GA has found several peaks on the landscape, it is getting progressively harder to locate remaining peaks because those peaks are isolated like needles in a huge haystack.

The disadvantages of the sequential approach are far more than that. Although it has a derating function, the algorithm itself still could not guarantee to avoid repeated search within visited areas and repeated convergence to the same optimum. Furthermore, the derating function severely alters the original landscape. It is subject to high risks of loss of actual optima and generation of phantom optima.

Mahfoud empirically compared the sequential GA with two other parallel GAs: the deterministic crowding and the fitness sharing. In general, the parallel methods

outperform the sequential method because they are faster and their results are better. In addition, the parallel method can be easily implemented on parallel machines.

Therefore, it is not surprising that most multimodal GAs adopt a parallel scheme [1, 2, 11, 12, 20-30].

2.2.3.2 Genetic Operator Based GAs versus Population Based GAs

Both the Genetic Operator Based GA and the Population Based GA strive to maintain diversity in the population so that the search space can be explored widely. One or more regions may be searched in parallel. However, they differ in the way of diversification and various aspects of the evolutionary process, as listed in Table 2-1 below.

Table 2-1 Genetic Operator Based GA versus Population Based GA

Algorithms	**GOB**	**PB**
Diversity maintenance	Genetic operators	Explicit subpopulations
Object to be evolved	The overall population	Every independent subpopulation
Elitism	Global	Local
Selection	Global	Local
Mating and Mutation	Global	Local

Note that Table 2-1 only characterizes typical behavior of these two GA categories. In some circumstances, exceptions may apply. For example, some genetic operator based GAs adopt local elitism either partially [2, 28] or fully [21, 22]. Although most population based GAs enforce a mating restriction, some variations allow more or less inter-species exchange, such as Minimal Representation Size Clustering [11] and Multipopulation GA [25].

In summary, an essential difference between a genetic operator based GA and a population based GA is whether the species/subpopulations are evolved alone or the population is evolved as a whole.

Since most evolutionary operators of genetic operator based GAs act globally, selection pressure is applied globally. As the authors pointed out in [2], individuals are spread around the peaks, rather than clustered tightly at the apex. Local elitism may more or less induce some local selection pressure, but convergence and accuracy are still in question. Conversely, in population based GAs, each subpopulation is focused on a particular region. Selection pressure is hence zeroed in on the exact apex. This mechanism is more likely to ensure maximum exploitation of all regions.

Chapter 3 Clustering in Multimodal GAs

Despite the multiformity of existing GA techniques on multimodal optimization problems, all of them essentially involve a clustering-like behavior, that is, identification of groups of individuals that lie within vicinities of potential optima.

In Sequential Niche GA [19], individuals sufficiently close to a discovered local optimum are punished using a fitness derating function, in order to avoid repeated search within previously explored areas.

The clustering actions are more manifest in parallel GAs. In sharing GA [2, 20, 24, 28, 30], vicinal individuals are clustered together to form niches. Fitness is then shared among the same niche. In [1, 21, 29], explicit clusters, called species [21], nations [1], or sub-populations [29], are maintained and evolved independently toward different potential optima.

Ideally, a parallel GA would eventually form a number of stable clusters, the number of which is approximately equal to the actual number of optima. Unfortunately, without *a priori* knowledge of the function surface, this is hard (though not impossible) to achieve. Effective identification of clusters is hence of key importance to GAs, especially to those exploring a multimodal landscape.

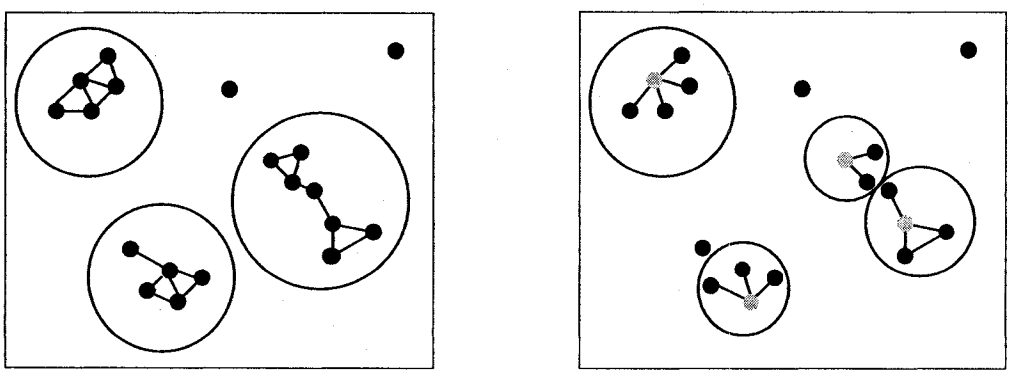
Before digging into details of our Bi-objective Multi-population Genetic Algorithm, we shall study various clustering methodologies in this chapter, and propose a novel clustering technique that effectively overcomes the shortcomings of other methods. For the sake of clarification, unless otherwise specified, such terms as niches, subpopulations, nations, or species are all called clusters in the sequel.

3.1 Clustering Algorithms

Based on the criterion to judge whether two individuals belong to the same cluster, there are two major categories of clustering algorithms: approaches based on distance metrics and approaches based on topology.

3.1.1 Distance Metrics Based Methods

Distance metrics are the most popularly used criteria for clustering in GAs. A basic principle is that if the distance between two individuals is smaller than a given threshold, these individuals are deemed to lie in one cluster. Typical distance metrics include phenotypic Euclidean distance [19-21, 25, 26] and genotypic Hamming distance [48].



(a) (b)

Fig. 3-1 Distance based clustering approaches
(a) Clustering based on distance between all individuals;
(b) Clustering based on distance between the centers of the clusters and the individuals

There are two major categories of distance based approaches. One measures the distance between all pair-wise individuals. Two individuals belong to the same cluster if the distance between them is smaller than a threshold. As shown in Fig. 3-1 (a), a straight line connects pairs of individuals, between which the distance satisfies the condition above. All connected individuals belong to the same cluster, represented with a circle in

the figure.

The other measures distance between individuals and the centers of the clusters. As shown in Fig. 3-1 (b), the centers of the clusters are marked in gray. All individuals, whose distance to one of these centers is smaller than a threshold, are placed into the same cluster.

The first category includes standard Sharing [20] and density based clustering approaches [26, 59, 60, 64].

In standard Sharing [20], pair-wise distance between all individuals is computed. If the distance is smaller than a threshold, the individuals belong to the same niche and contribute to fitness sharing within this niche. It is assumed that each of the p optima is surrounded by a hyper-sphere of radius r. These hyper-spheres do not overlap and completely fill the d-dimensional space. If the parameter range for each dimension is normalized to [0, 1], the radius r can be computed by:

$$r = \frac{\sqrt{d}}{2\sqrt[d]{p}} \tag{3.1}$$

Streichert *et al.* used a density based clustering algorithm [26], which identifies clusters by connecting individuals if the distance between them is smaller than a threshold σ_{dist}. Any interconnected groups of individuals with size greater than a minimum value *MinPts* is recognized as a cluster. This algorithm allows clusters of varying shape and does not need *a priori* knowledge of the number of clusters. Its computational cost is $O(NlogN)$ [60]. Nevertheless, its two parameters σ_{dist} and *MinPts* need to be carefully tuned. The density based approach is also used in [59, 64].

In another school of multimodal GAs, a cluster is a hyper-sphere, where all its

members are within a certain distance (the radius of the hyper-sphere) to its center. To judge whether an individual belongs to a cluster or not, its distance to the center of the cluster d, rather than the distance to another individual in this cluster, is computed and compared to the radius of the cluster r. If d is smaller than r, it belongs to the cluster, otherwise not.

Yin and Germay [30] proposed to use MacQueen's K-means cluster analysis [65]. It starts with k initial clusters with only one member. Vicinal clusters with their distance less than a threshold d_{min} are merged and the center of the new cluster is updated. All other individuals are assigned to the cluster with the nearest center. If the distance between an individual and the nearest center is larger than d_{max}, a new cluster is created around it. The computational cost of this approach reduces is $O(NK)$, where K is the number of clusters. However, it is hard to estimate the values of d_{max} and d_{min} appropriately.

In [21], Li *et al.* proposed to conserve seeds – the up-to-date best individuals for each species (cluster). These seeds form the centers of the clusters. All unmarked individuals are first ranked with respect to their fitness. Starting from the best unmarked individual, if its distance to any existing seeds is smaller than a threshold, it belongs to the corresponding cluster; otherwise it is a new seed. This threshold is empirically determined.

[45] and [47] also adopt a similar approach to Li *et al.*'s Species Conserving GA to identify the centers of clusters.

In many situations, for those algorithms that cluster individuals according to their inter-distance, the distance between two individuals does not denote their relationship

correctly. For example, the distance between two individuals that belong to the same cluster but lie on two extremities is not necessarily smaller than the given threshold. Unless there happen to exist intermediate individuals that connect these extremities, they will definitely be separated into different clusters. On the other hand, individuals on the borders of adjoining clusters are very likely to be put in the same cluster since they are sufficiently close. This may cause an undesired mix-up of clusters.

Therefore, it is plausible to favor those algorithms that treat each cluster as a hyper-sphere, and categorize individuals according to their distance to the centers of the clusters. However, a simple hyper-sphere may not always suffice, especially on highly irregular landscapes. It is not easy to choose an appropriate radius without *a priori* knowledge of the landscape.

In any case, all the approaches discussed above employ a single global radius, based on an underlying assumption that all optima are of approximately the same hyper-volume and are equidistant to each other. Most multimodal GAs adopt this scheme [19, 21, 24, 47-49, 52]. Obviously this assumption does not hold for all actual multimodal landscapes.

Gan and Warwick [2, 28] proposed a dynamic niche clustering (DNC) approach. The algorithm starts from N small niches with a given initial radius. Each niche has an independent radius, which is then dynamically adjusted throughout the evolutionary process.

The initial radius $\sigma_{initial}$ is given by:

$$\sigma_{initial} = \frac{\lambda\sqrt{d}}{\sqrt[d]{N}} \tag{3.2}$$

where d is the dimensionality, N is the population size, and λ is a constant typically set

with a value of 1.0.

The value of $\sigma_{initial}$ directly affects the performance of this algorithm. As the authors themselves pointed out, if $\sigma_{initial}$ is too small, a large number of niches are manipulated with a high computational cost; whereas if $\sigma_{initial}$ is too big, niches will be merged rapidly, resulting in inappropriately large niches that cover more than one actual optimum [2, 28].

As seen from equation (3.2) above, for a specific problem, the dimensionality d is fixed. Hence the value of $\sigma_{initial}$ is determined by the population size N. Generally speaking, N would have to be quite large in order to guarantee a thorough exploration of the search space. N is usually much larger than the actual number of the optima. Therefore the first phenomenon discussed in the paragraph above is more frequently observed. $\sigma_{initial}$ is often considerably smaller than the average distance between the optima, and as a consequence the initial niches take a long time to grow into their effective size. Of course this phenomenon can be mitigated by increasing λ or decreasing N. However, as stated above, this may result in a $\sigma_{initial}$ that is too large. Therefore, the selection of N has to be a compromise between these two extremities.

Hence, although the dynamic niche scheme appears to be more interesting and more practical than a static global threshold. The initial niche radius $\sigma_{initial}$, which is tightly tied to the population size N, continues to play a key role in the success or failure of clustering.

It is conjectured that there may exist optimal values for both population size and niche radius, with which the best possible sharing effects are achieved. Cioppa, Stefano

and Marcelli [42] proposed a method to estimate these two parameters by characterizing the dynamic behavior of the sharing GA as a function of both the niche radius and the population size. However, the authors still adopted a global static niche radius value. They admit that the optima of irregular fitness landscape may be indistinguishable if a single global radius is utilized.

All the methods discussed above entail a concept of niche radius or threshold. In the literature, there also exist distance based clustering methods that do not need this concept. The hierarchical clustering method [61] is a typical example. The basic procedure of this algorithm consists of three steps. Firstly, distance between every pair of individuals in the data set is calculated. Secondly, these individuals are grouped into a binary hierarchical cluster tree based on their mutual distance. Every time a pair of samples (or a pair of sample sets consisting of several samples) that has the minimum distance in the data set is selected, and joined into one set. This procedure is repeated until all samples are joined together into one single set. Finally, the hierarchical tree is cut into a preset number of clusters□

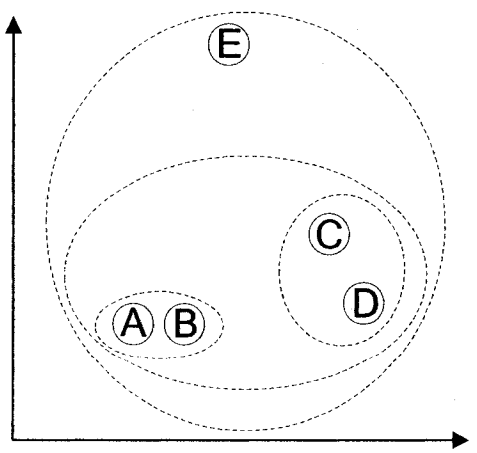


Fig. 3-2 Example of hierarchical clustering

Fig. 3-2 shows an example of the clustering procedure. Initially, A and B are selected to join into one set {A, B} since the distance between them is minimum. Next, C and D are joined into {C, D} since their pair-wise distance is minimum among the pair-wise distances of C, D, E and {A, B}. Following that, {A, B} and {C, D} are joined into {A, B, C, D}. Finally, {A, B, C, D} and E are joined into one single set.

The binary cluster tree finally formed is shown in Fig. 3-3.

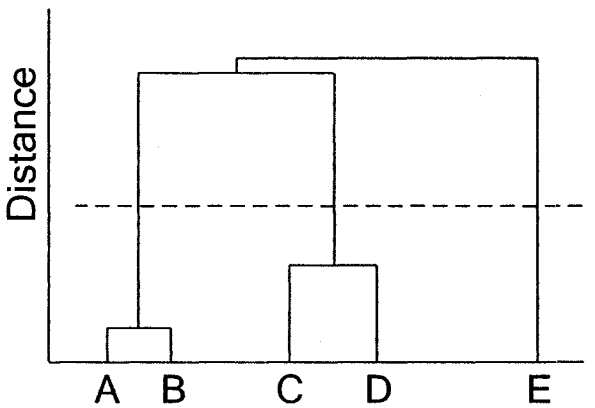


Fig. 3-3 Binary cluster tree

With the binary cluster tree, creating clusters from a specified arbitrary cluster number is simple. For example, to divide the 5 samples into 3 clusters, just find where the cluster tree has 3 branches, and cut it there, as the dashed line shows in Fig. 3-3. The three clusters are {A, B}, {C, D} and E. The heights at which two branches join represent the distances between two sample sets. But the actual cutting distances are not concerned because clustering with a given number is based on the branch numbers rather than the cutting distances. Hence this algorithm eliminates the need of a distance threshold.

Nevertheless, the hierarchical clustering approach has a fatal pitfall. That is, it requires the knowledge of the number of clusters, which is usually unavailable in most situations.

3.1.2 Topology Based Methods

As an intuitive measure of *closeness* or *similarity* of individuals, the distance metrics is simple to understand and easy to implement. However, the distance based approaches have various limitations, as discussed in Section 3.1.1 above. Rather than using distance between individuals, topology based methods employ the topological characteristics of the function's surface. An intuitive notion is that optima are separated by valleys, if the function is to be maximized. Hence, one might suggest that two points belong to the same cluster if there are no valleys separating them. It is vice versa if the function is to be minimized.

In line with this arguments, Ursem [1] proposed a *Hill-Valley* (HV) function. Given two points p and q, the HV function verifies the existence of the valley by checking the fitness values of several interior sampling points $int[i]$ between p and q. These interior samples are given by:

$$int[i] = p + (q - p) \cdot samples[i] \tag{3.3}$$

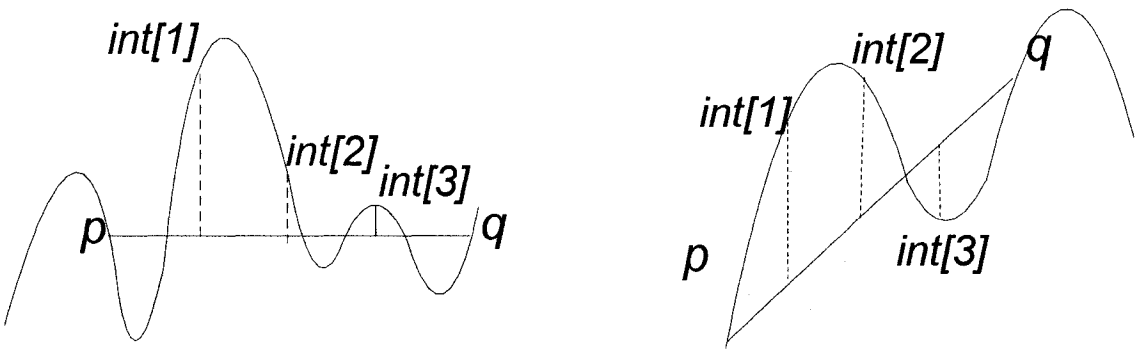
where *samples* is an array of sampling constants between [0, 1], e.g. [0.25, 0.5, 0.75].

According to this technique, if none of these interior points has a lower fitness value than either p or q, there is no valley between them. As a result, p and q belong to the same cluster.

Ursem's approach is more tolerant to irregularity of the landscape than distance based methods. Gan and Warwick [2] adopted it in their algorithm and demonstrated improvements over their previous version [28] that merely used a distance metrics for clustering.

Nevertheless, the HV function still has serious limitations. Fig. 3-4 demonstrates

two typical flaws of the HV algorithm. p and q are two points on different peaks.



(a) (b)

Fig. 3-4 Two drawbacks of Ursem's hill-valley algorithm

In Fig. 3-4 (a), the samples are too sparse or are not appropriately positioned for a proper sampling of points between p and q. To avoid this problem, a properly designed sampling vector is required. However, this is not possible without information about the landscape. Moreover, a sampling vector perfectly fitting one type of landscape does not necessarily fit other types.

In Fig. 3-4 (b), the fitness of q is larger than the fitness of all interior points. Hence it is simply impossible to find an interior point with lower fitness than both p and q due to the topology of the surface between them.

The two cases presented above are indeed encountered in many scenarios. As a result, p and q are mistakenly clustered together, and actual peaks are usually overlooked.

In [1], Ursem suggested an asymmetric interior sampling vector [0.02, 0.25, 0.5, 0.75, 0.98] for merging of nations, since, as the authors claimed, merging nations is a more drastic operation than migration of individuals. Indeed, the latter is more helpful than the symmetric vector when, given two end points, e.g. A and B in Fig. 3-5, one of their vicinal interior samples (C) happens to lie on the descending direction of either end

point (A) and has a lower fitness value than that of both A and B. Nevertheless, if the vicinal interior points (F and G in Fig. 3-5) are on ascending directions of the end points (D and E in Fig. 3-5), or if the topology is similar to that in Fig. 3-4 (b), the undesired merging of distant clusters will still happen.

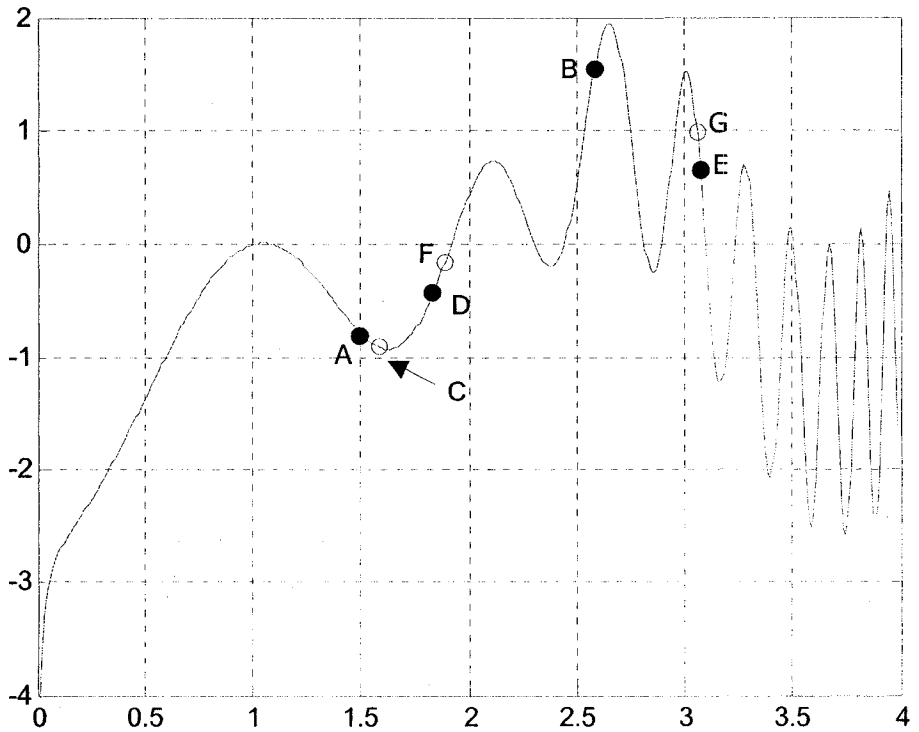


Fig. 3-5 Vicinal interior samples

3.1.3 Hybrid Methods

Some algorithms use both distance and topology information to perform clustering. Similar to algorithms discussed in Section 3.1.1 and 3.1.2, Lin and Wu's algorithm [22] also assumes that a cluster is approximately hyper-spherical. The individual with the highest fitness is the center of the first cluster. All other individuals are ranked with respect to their distance to this center. Starting from the nearest individual, the algorithm compares fitness between pairs of adjoining individuals (i and i+1) in sequence. If the fitness of the $(i+1)_{th}$ individual is larger than that of the i_{th} individual, the latter is

preliminarily qualified as a boundary between two clusters.

In order to tolerate small bumps on the landscape, a *bump ratio* is defined:

$$B_i = \frac{F(i+1) - F(i)}{F_{\max} - F_{\min}} \tag{3.4}$$

where $F(i)$ is the fitness of the i_{th} individual, F_{max} and F_{min} are the maximum and minimum fitness in the population, respectively.

If the bump ratio of an individual i is less than a given threshold, this trivial fitness increment is ignored. i still belongs to current cluster. Conversely, if the bump ratio exceeds the threshold, i is the boundary of the niche. The radius of this niche is the distance between i and the center.

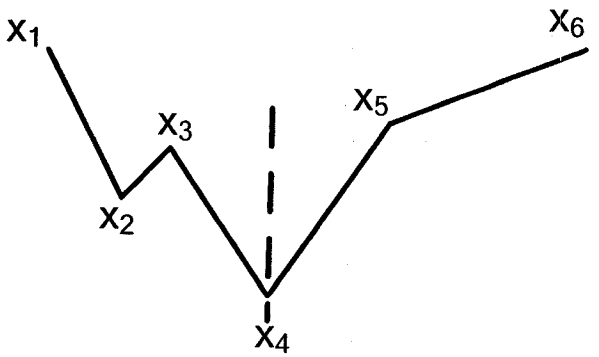


Fig. 3-6 Identification of a boundary individual and a trivial bump

Fig. 3-6 shows an example of a boundary individual and a trivial bump. It can be seen that $x_2 - x_3$ pair is a small bump since the fitness increase from x_2 to x_3 is small; whereas x_4 is the boundary of the cluster centered at x_1 because the bump ratio between x_4 and x_5 is larger than the threshold.

The individual with the highest fitness among the remaining individuals is used as the center of a new cluster. The same process is iterated until all individuals are assigned to a cluster. A cluster is discarded if its size is smaller than 10% of the population. The

resulted clusters are then manipulated so that there is no overlapping between them. All individuals are finally redistributed.

The problem of this algorithm is the same as those algorithms presuming hyper-spherical clusters. Especially in high dimensional space, the distance between the center of the cluster and the extremity on every dimension is not guaranteed to equal. Additionally, this approach needs a sufficiently large number of samples on the function's surface, which naturally entails a large population. Finally, the bump ratio threshold is an important parameter that significantly influences the final results.

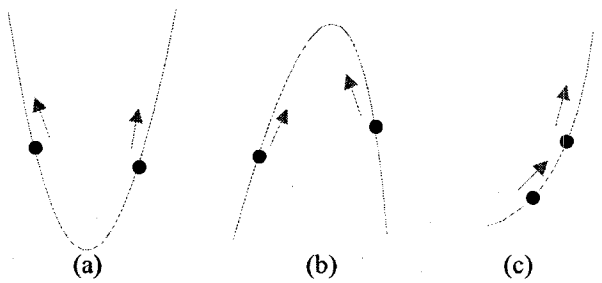


Fig. 3-7 The relative ascending directions of two given individuals
(a) back to back; (b) face to face; (c) one-way

Leung and Liang proposed a method to determine the relationship between two individuals based on their relative ascending direction and distance [66]. There are only three types of relative ascending direction between two given individuals: back to back, face to face, and one way, as demonstrated in Fig. 3-7 above.

If the relative ascending directions of two individuals are back to back, they must lie on different peaks and hence belong to different clusters. Otherwise if the relative ascending directions of two individuals are face to face *or* one-way, *and* the distance between them is smaller than a threshold, these two individuals are located on the same

peak and hence belong to the same cluster.

With this method, two close individuals in different clusters will never be put into the same cluster because their relative ascending direction is considered first. But two distant individuals within the same cluster may still be split because the distance between them exceeds the threshold. This problem is common in most distance based clustering techniques. A single global threshold does not suffice.

3.2 Centers of Clusters

The center of a cluster can be, but is not necessarily, the current fittest individual within the cluster. Most clustering approaches use the up-do-date best individual as the center of a cluster. Some other approaches take into account the fitness of the cluster members when computing the center of the cluster.

For example, DNC [2, 28] starts from N (population size) one-member clusters and as such has each member as the center of its own cluster.

After clustering, individuals are redistributed. The center of each niche is adjusted using the formula:

$$mid_i = mid_i + \frac{\sum_{x=1}^{n_i}(v_x - mid_i)\cdot f_x}{\sum_{x=1}^{n_i} f_x} \tag{3.5}$$

where n_i is the size of the niche i, v_x is the position of individual x in the parameter space, f_x is the fitness of x.

If two niches i and j are merged, the new center becomes:

$$mid_{new} = mid_m + \frac{\sum_{x=1}^{n_i}(v_x - mid_m)\cdot f_x + \sum_{x=1}^{n_j}(v_x - mid_m)\cdot f_x}{\sum_{x=1}^{n_i} f_x + \sum_{x=1}^{n_j} f_x} \tag{3.6}$$

where mid_m is given by:

$$mid_m = \frac{mid_i + mid_j}{2} \tag{3.7}$$

Unfortunately, this method may alter the original direction of the cluster. The updated center strays from its own cluster since it is affected by the fitness values of all members of this cluster.

Let us give a simple example. Fig. 3-8 is a one dimensional multimodal function with uneven distribution of peaks. Now there are two niches A and B to be merged. Each niche contains one member, which is also the center of the niche. The centers of these niches are plotted in Fig. 3-8 and listed in Table 3-1.

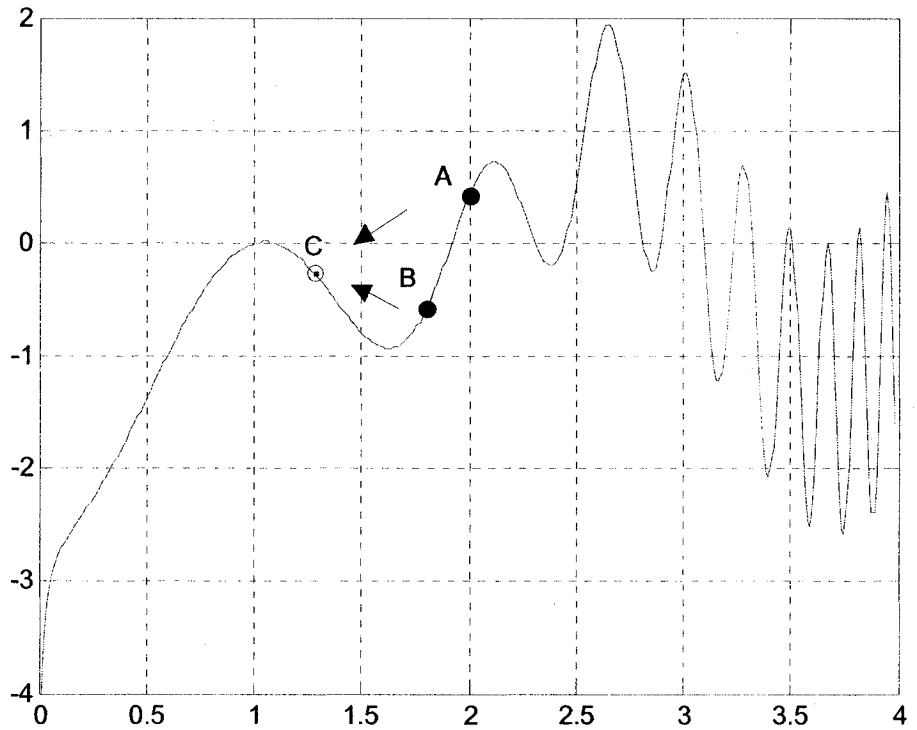


Fig. 3-8 Center deviation from weighted average of individuals

The new center C, as given by equation (3.6) and (3.7), is computed as:

$$mid_m = \frac{mid_A + mid_B}{2}$$
$$mid_C = mid_m + \frac{(v_A - mid_m)\cdot f_A + (v_B - mid_m)\cdot f_B}{f_A + f_B} \tag{3.8}$$

Table 3-1 Original and updated centers after merging of niches

Niche	Parameter	Fitness
A	2.0000	0.4259
B	1.8000	-0.5902
C	1.2816	-0.2708

From Fig. 3-8 it can be seen that C has moved to a point that does not belong to the original cluster. This straying behavior is unfavorable because it indicates that all previous efforts in evolving the cluster of A and B may be fruitless. It may even cause loss of mature clusters.

The center of a cluster can also be the average of all [30] or some [1] members of the cluster. For example, in MNGA [1], each cluster has a *government*, which is comprised of the best k individuals within this nation (k is a pre-defined constant). The center of this cluster is an average of the government members. This method naturally incurs extra computational cost. Even worse, if it is used in the Hill-Valley algorithm, clusters are subject to an even higher risk of straying because their centers may be obtained via mutual interactions of individuals that lie within different nations.

Fig. 3-9 demonstrates such an example. Suppose the government size is 2, that is, the center of a cluster is an average of the best 2 individuals within a nation, and the sampling vector is [0.25, 0.5, 0.75], as in [1]. *P* and *Q* are mistakenly placed within the same cluster, since none of their three sampling points *S-1*, *S-2*, and *S-3* has lower fitness than both *P* and *Q*. The midpoint of this cluster is hence an average of *P* and *Q* - *S-2* in the picture. Obviously *S-2* does not in any sense represent this cluster. The center of this

cluster may continue to stray as pairs of *P* - *S-2* and *Q* - *S-2* are not within the same clusters either.

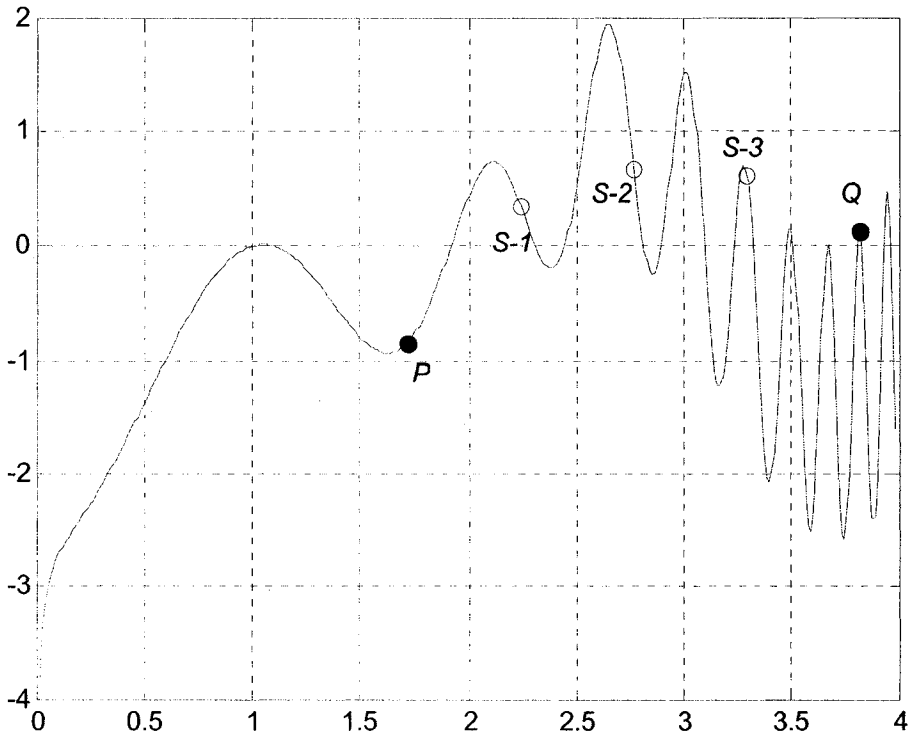


Fig. 3-9 Center deviation from an average of individuals

As a matter of fact, In MNGA, if the government size exceeds one, more alien individuals may be involved in the calculation of the midpoint. And this cluster is expected to be more unstable. We further compared the performance of MNGA [1] with different government size schemes. Two multimodal functions D_1 (Fig. A–1) and D_2 (Fig. A–2) are used. The average number of clusters formed and peaks found are graphed in Fig. 3-10 and Fig. 3-11 respectively.

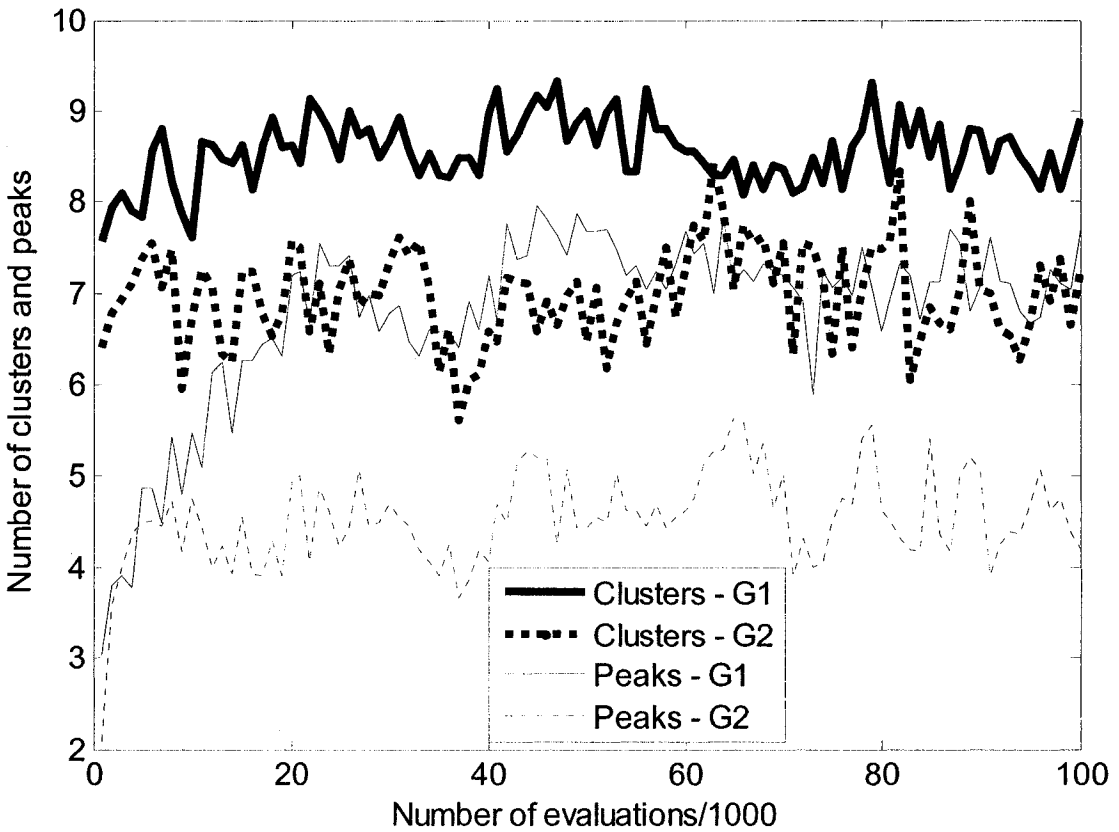


Fig. 3-10 Clusters formed and peaks found via hill-valley function in Multinational GA – function D_1

In both Fig. 3-10 and Fig. 3-11, G1 (in solid line) is the case of government size = 1 (that is, the up-do-date best individual is the center of the cluster), while G2 (in dotted line) is the case of government size = 2. For both two schemes, the program was run 30 runs with the same amount of chromosome evaluations at each run (10^5 for function D_1 and 10^6 for function D_2, respectively). The number of clusters and peaks present in the population were counted periodically (every 1000 evaluations for function D_1 and every 10000 evaluations for function D_2, respectively). The overall population size for D_1 is 45. The overall population size for D_2 is 125.

From Fig. 3-10, it can be seen that the hill-valley function identifies less clusters with G2 scheme than with G1 scheme. The reason is that the arithmetic average of more

than one individual induces more straying effects to the clusters. Seen globally, it causes more chaotic mix-up of clusters - and hence decreased capacity of the population to hold clusters. In Fig. 3-10, the number of peaks found with G2 is also less than that with G1. From a local aspect of view, clusters stray away more frequently and become more unstable. As a result, some clusters are not evolved consistently toward a fixed direction. The consequence is slower convergence to the optimum and less peaks found.

It should also be noted that the number of peaks is less than the number of clusters for both schemes. This is expected since, as discussed above, the hill-valley function itself cannot completely avoid straying clusters due to its inherent limitations. Furthermore, MNGA adopts a fixed sized population with a varied number of varied sized nations. Not all nations are assigned the same amount of resources. Affected by these factors, some clusters cannot be fully exploited and evolved.

In Fig. 3-10, we can also clearly see that the "Peaks – G1" curve is closer to the "Clusters – G1" curve than the "Peaks – G2" curve to the "Clusters – G2" curve. That means the G1 scheme locates more consistent and stable clusters than the G2 scheme so that mature peaks can be identified within these clusters.

Another interesting phenomenon in Fig. 3-10 is that all curves fluctuate violently. As stated in [1], migration of individuals is followed by merging of clusters approaching the same peaks. Two reasons account for decrease of clusters. Firstly, false merging of different clusters frequently occurs. Secondly, the selection pressure within each cluster may also eliminate immature alien individuals.

After distant clusters are merged, new alien individuals are generated via mating of parents from different nations, which, in turn, causes new migrations and emergence of

new nations. This behavior results in increase of clusters.

Therefore, the number of clusters and peaks oscillates around a certain level, which is the capacity of the population to hold the clusters and the peaks.

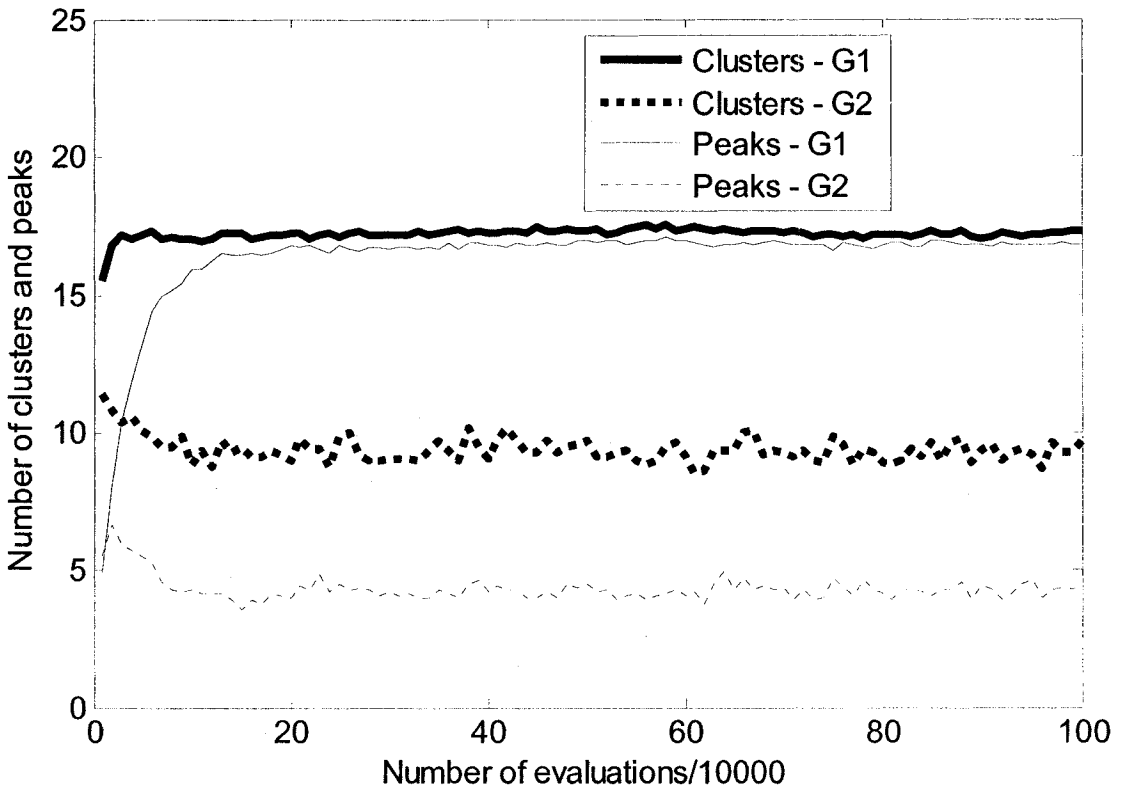


Fig. 3-11 Clusters formed and peaks found via hill-valley function in Multinational GA – function D_2

Similarly in Fig. 3-11, G1 identifies more clusters and peaks than G2. Meanwhile, the number of peaks is less than the number of clusters for both schemes. Note that both the amplitude and the frequency of the oscillation are lower than those in Fig. 3-10. That is because 25 narrow peaks are distributed evenly on a large plateau region (Fig. A–2). The volume of these peaks is approximately the same. The possibilities of clustering different species together and merging distant clusters are decreased greatly. Hence frequent and drastic alteration of number of clusters is lessened.

In summary, in this section, we presented and compared different approaches to obtain the center of a cluster. A weighted average of individuals may result in an alien center, even if all these individuals belong to this cluster. A simple average of cluster members may not generate alien centers. However, if it is used in the hill-valley method, it worsens the algorithm by inducing more roaming unstable clusters. If it is used in other algorithms, it only entails extra computations without any solid benefits. Therefore, we will use the up-to-date best (dominating) individual as the center of a cluster, as in [21, 45].

3.3 Recursive Middling

From the discussion in Section 3.1.1, we can see that most distance based clustering algorithms work under one or more following assumptions:

- All optima are distributed evenly in the search space, and the distance between them can be empirically estimated;
- The shape of a cluster is a hyper-sphere;
- The number of optima is known *in priori*.

Topology based approaches generally do not need these assumptions. However, they still have their own limitations (Section 3.1.2).

Therefore, a more effective clustering algorithm is in demand. This algorithm does not need any empirical threshold that is hard to estimate. It does not assume even distribution of the optima or basic shapes of the clusters. And it does not need any *a priori* information about landscape either.

Here we will propose a novel Recursive Middling (RM) algorithm that effectively overcomes drawbacks of aforementioned approaches and completely fulfills the

requirements above with an acceptable amount of overhead. Before presenting the details of the RM algorithm, we first present some concepts and rules, based on which the algorithm is developed.

3.3.1 Concepts and Rules

Concept 1: Given two points x_1 and x_2, a point x is an interior point of them if:

$$x = \alpha x_1 + (1-\alpha)x_2, \qquad \alpha \in [0,1] \tag{3.9}$$

It can be seen that an interior point with respect to two points is simply a linearly interpolated point between them.

Concept 2: Given two points x_1 and x_2, if there exists an interior point x with:

$$F(x)<F(x_1) \quad \& \quad F(x)<F(x_2) \tag{3.10}$$

where $F(x)$ is the function to be studied, we say there is a valley between x_1 and x_2 on (x, $F(x)$).

From Fig. 3-4 in Section 3.1.2, we can see that even there is no valley on any sampled interior points of x_1 and x_2, it does not necessarily mean there is no valley between them. The missing of an actual valley may be due to sparse sampling, as in Fig. 3-4 (a), or the topology of the landscape, as in Fig. 3-4 (b). Therefore, the concept of a valley is extended as follows:

Concept 3: Given two points x_1 and x_2, if there exists an interior point x with:

$$F(x)<F(x_1') \quad \& \quad F(x)<F(x_2') \tag{3.11}$$

where $F(x)$ is the function to be studied, x_1' and x_2' are interior points of x_1 and x_2, then there is a valley between x_1 and x_2 on (x, $F(x)$).

It should be noted that it is vice versa if minimization of the function is pursued.

The roles of *maximum* and *minimum* as well as *valley* and *peak* are simply exchanged. All *smaller* relations ("<") are replaced with *larger* relations (">").

3.3.2 The Algorithm

Based on the concepts above, the RM algorithm is proposed to detect a valley between two given points. In a similar manner to the HV method, RM is also based on the topology of the landscape. However, it effectively avoids the pitfalls of HV and returns better clustering results.

The main idea of RM is to check the fitness values of a pair of points (x, y) against the fitness value of the midpoint m between them. If the fitness of the midpoint m is the lowest of the three, from the concepts and rules above, there is at least one valley between x and y, and these two points are deemed to belong to different clusters. Otherwise, RM is applied to both (x, m) and (m, y), recursively, until either a valley is found or the endpoints finally converge. Convergence occurs when the distance between the final set of points is less than a preset threshold σ. In the latter case, x and y are deemed to belong to the same cluster.

The distance between the points can be, but is not limited to, Euclidean, city block or hamming distance. In this study, the Euclidean distance is used. The threshold σ is more or less problem dependent, but it is tied to specific problems less tightly than the global threshold in [19-21]. This is because σ is used to decide whether two points are so close that they are practically the same point, whereas a global radius is used to decide whether two points belong to a single cluster with a pre-specified but not necessarily correct radius.

Generally, σ should be very small to avoid mix up of adjacent optima so that in

most fitness landscapes, no optima would co-exist within such a small interval. Of course σ could be even smaller, but that only incurs much more unnecessary computation.

```
RM(x, y)
  If converge(x, y)
    Return false;
  m = (x+y)/2;
  If (fitness(m)<fitness(x) and fitness(m)<fitness(y))
    Return true;
  Else
    Return (RM(m, x) or RM(m, y));
  End if
End
```

Fig. 3-12 The Recursive Middling algorithm

Fig. 3-12 shows the pseudo code of the algorithm. The function RM returns true if a valley is found between x and y. If these points finally converge, it returns false.

Note that this algorithm, by its nature, is able to identify all potential clusters (optima) in most landscapes. Hence if there exist trivial local optima that may not be interesting, a problem dependent threshold is required to filter them out. In this paper, we assume that all local optima are of interest and aim to find all of them.

3.3.3 Validity Tests

3.3.3.1 Recursive Middling versus Hill-Valley

The validity of the RM algorithm is tested on two typical multi-modal functions D_1 (Fig. A–1) and D_2 (Fig. A–2), respectively. Ursem's HV function [1] is also implemented for comparison. In both algorithms, the center of each cluster is the up-to-date best individual in the cluster. Each individual is randomly generated. After its fitness is evaluated, the clustering algorithm is applied on it immediately. This individual is then put into either one of the existing clusters or a new cluster, depending on the result of the clustering

algorithm. This is a Monte Carlo process as we want to see the pure effects of the clustering algorithms without the interference of any other factors such as evolution.

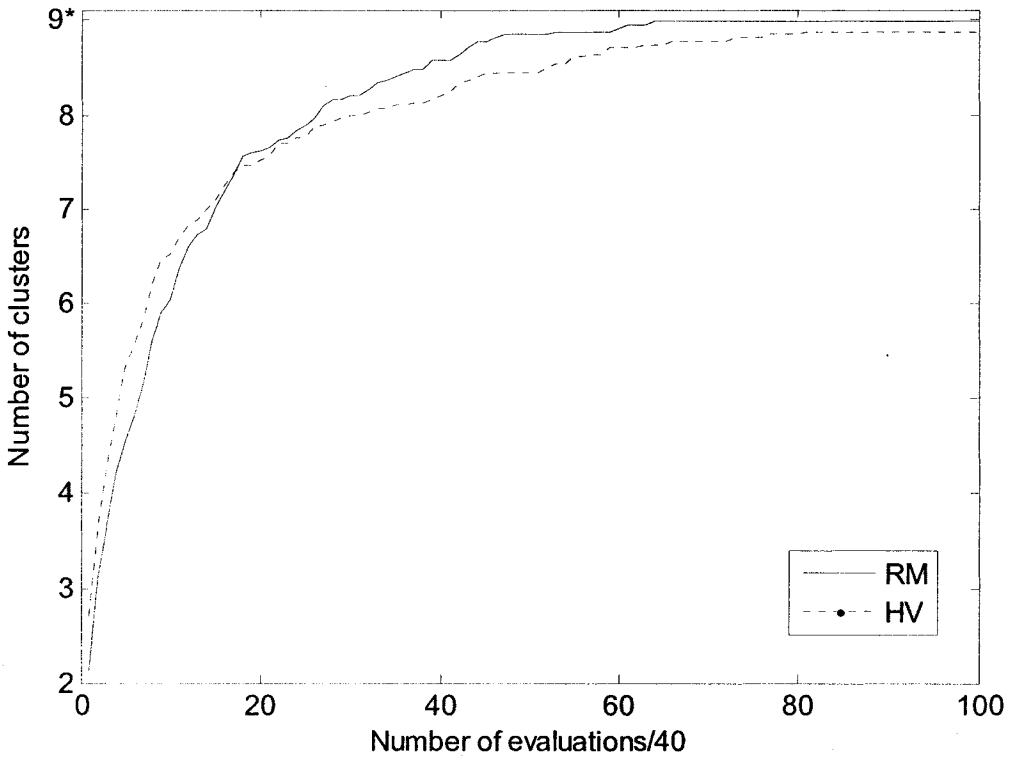


Fig. 3-13 Comparison of RM and HV - D_1

Table 3-2 Performance of RM and HV on D_1

Algorithms	Average evaluations	Average number of chromosomes	Successful runs
RM	1516.7	64.48	29
HV	1649.8	233.33	16

Fig. 3-13 and Fig. 3-14 show the average clustering effects of both algorithms with respect to the number of evaluations after 30 runs. The desired number of clusters is marked with an asterisk in both figures. A run is deemed to succeed if all clusters are located. Both Table 3-2 and Table 3-3 list the number of successful runs as well as the average number of evaluations and chromosomes assessed when the number of clusters reaches the desired number in those successful runs. Note that the average number of

evaluations far exceeds the average number of chromosomes since both algorithms evaluate many extra interior samples between pairs of chromosomes.

From both Fig. 3-13 and Fig. 3-14, it can be seen that although HV initially forms clusters faster than RM, it is slower than RM after a certain number of evaluations. This phenomenon is more manifest in Fig. 3-14, in which HV tends to stabilize at a lower number of clusters. On the other hand, RM locates slightly more than the desired number (25) of clusters (25.66) in Fig. 3-14. This can be remedied by periodically merging clusters that are surrounding the same peak.

RM successfully locates all clusters of function D_1 for 29 runs in a total of 30 runs, whereas HV successfully locates all clusters for only 16 runs (Table 3-2). The average numbers of evaluations required to locate all clusters do not differ much for these two algorithms. But the average number of chromosomes for RM (approximately 64) is much less than that for HV (approximately 233). This is reasonable since for each pair of chromosomes, RM evaluates, on average, more interior samples than HV.

HV fails to locate all clusters for function D_2 completely (0 successful runs in Table 3-3). From Fig. 3-14 it can be seen that it reaches the equilibrium of approximately 16 clusters early. This is because of the symmetric nature of the landscape. As demonstrated in Fig. A–2, 25 peaks of D_2 are distributed evenly in a 5 X 5 square. These peaks are equidistant to each other and have approximately the same volume. Recall that we use the sampling vector [0.25, 0.5, 0.75]. This vector is symmetric and the resulted interior samples are equidistant to each other too. Hence no valley is found between distant species when there are other species lying between them. These species may be placed in one cluster, resulting loss of actual species. One possible solution is to use a

different sampling vector other than [0.25, 0.5, 0.75]. However, this reveals a fact that the sampling vector needs to be carefully designed and it is almost impossible to get an appropriate vector without *a priori* knowledge of the landscape.

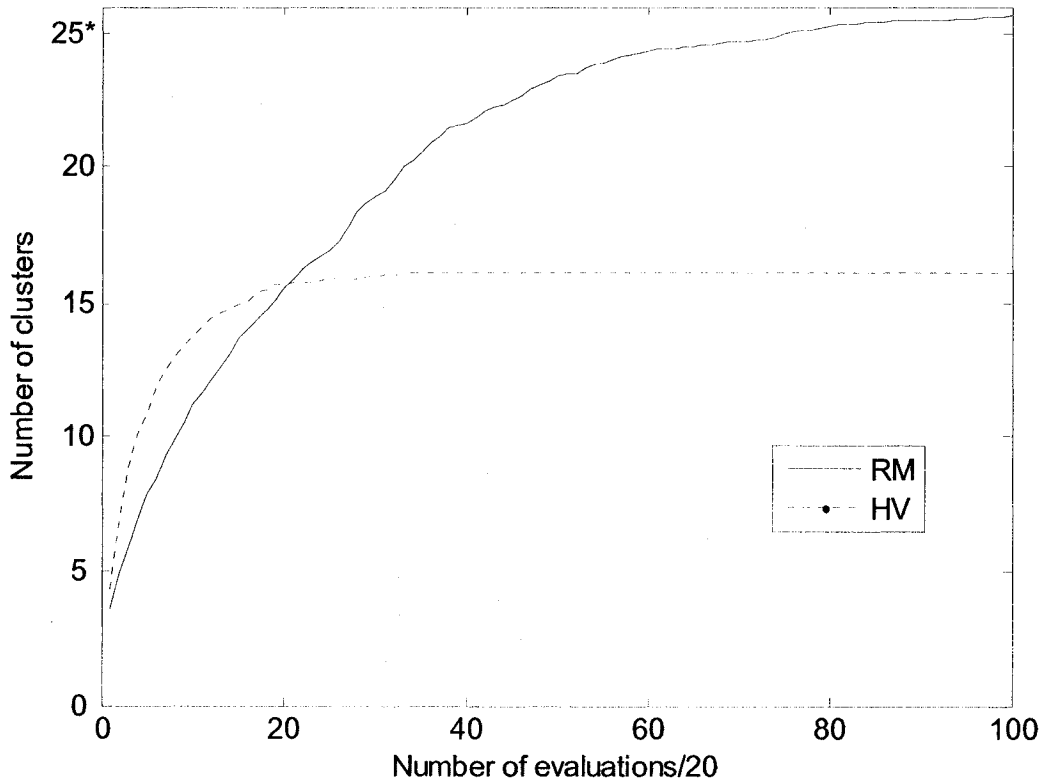


Fig. 3-14 Comparison of RM and HV - D_2

Table 3-3 Performance of RM and HV on D_2

Algorithms	Average evaluations	Average number of chromosomes	Successful runs
RM	12788	212.58	26
HV	N/A	N/A	0

RM also has parameters that need to be tuned as well. It needs a threshold to judge whether two points are sufficiently close so that they can be deemed to converge to one point. Unlike HV's sampling vector that is more tightly tied to specific landscapes, this threshold can fit more different landscapes. Once set, it can be universally applied on most multimodal landscapes. Furthermore, unlike the sampling vector that may cause

inferior performances, this threshold does not generally affect the performance of the algorithm. As long as it is within a reasonable range, RM is always guaranteed to find all clusters before it reaches the equilibrium. Smaller threshold only incurs more fitness evaluations since more interior samples are evaluated before their convergence.

3.3.3.2 Dynamic Niche Clustering

We further apply RM in Dynamic Niche Clustering (DNC) in contrast to its original version without RM [2]. There are two main reasons to choose DNC for testing. First of all, up to our knowledge, it is the only version of Sharing GA that manipulates niches of variable radii. Secondly, DNC in [2] incorporated both Euclidean distance metrics and Ursem's HV function. Hence RM can be compared with both forms of clustering techniques.

Initially, the DNC algorithm randomly generates a group of individuals. A new niche, with an initial radius r computed using equation (3.2) in Section 3.1.1, is created for each individual. Pairs of niches i and j are merged if certain conditions are satisfied. For the original form of DNC, these conditions are:

- The HV function does not detect any valley between i and j;
- The Euclidean distance d between the midpoints of i and j satisfies:

$$d \le \frac{\sigma_i}{2} \qquad \text{or} \qquad d \le \frac{\sigma_j}{2} \tag{3.12}$$

where σ_x is the radius of niche x.

For the sake of brevity, the implementation of DNC is not listed here. Interested readers may refer to [2] or Section 1.1.1.1.9 for details.

For DNC that adopts our RM scheme, a group of niches are initiated in the same way as the original DNC. However, these niches do not have initial radii. Two niches are

merged as long as the RM function does not find any valley between them. The main process is also slightly different to that of the original DNC:

1. Recalculate niche members;
2. Adjust the midpoint of each niche according to its members;
3. Check niche pairs for merging using the RM algorithm;
4. Apply sharing function within each niche.

It can be seen that DNC-RM does not need to sort and manipulate the niche pair list. It does not check niches for splitting either, because a large niche is almost unlikely to contain more than one niche, due to the nature of the RM algorithm.

We test three benchmark functions, all of which are typical multi-modal functions taken from literature [1, 26, 37]. The number of clusters and optima found are tracked along the generations. Because clustering is a major overhead of computation, the average CPU running time as well as the proportion of clustering and evolution are also demonstrated. The overall performance, averaged on 50 runs for each function, is gauged by the following measurements:

- *Success rate*: percentage of successful runs;
- O_{avg} / O_{max} / O_{min}: the average/maximum/minimum number of optima found at the end of a run;
- C_{avg} / C_{max} / C_{min}: the average /maximum/minimum number of clusters at the end of a run;

Note that a run is deemed to succeed if and only if all present optima are found.

Various parameters required by the evolutionary process are empirically given as follows:

- Population size: suppose each optimum is assigned 10 chromosomes, on average. For each function, the total population size is 10 times the number of actual optima, as listed in Table 3-4 below;

Table 3-4 Number of optima and population size for each function

Function	Number of Optima	Population Size
D_1	9	90
D_2	10	100
D_4	125	1250

- Elitism rate: 0.1;
- Crossover rate: 0.8;
- Mutation rate: 0.1;
- Number of generations: 80.

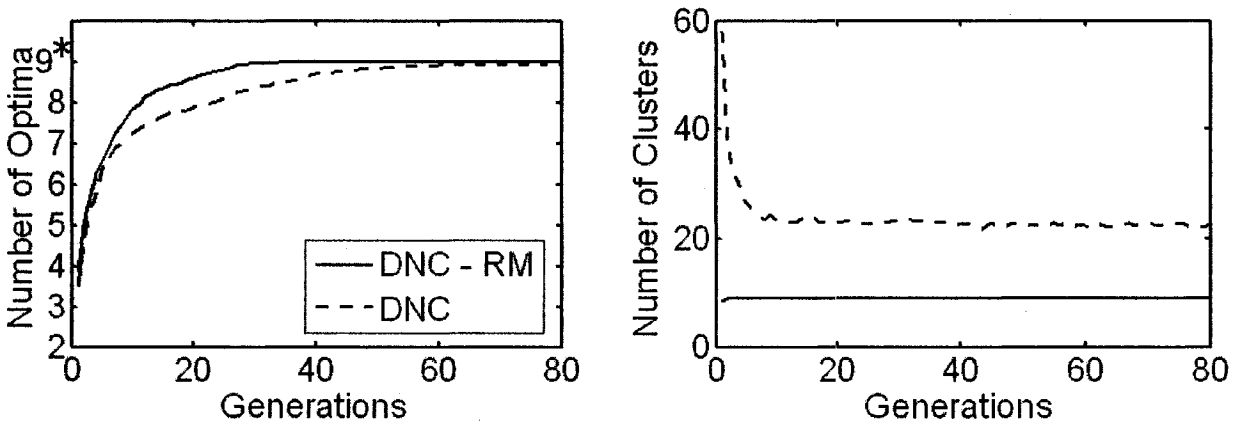
In all figures (Fig. 3-15, Fig. 3-16, and Fig. 3-17), the solid line denotes DNC with RM, while the dashed line denotes the original version of DNC. Fig. 3-15, Fig. 3-16, and Fig. 3-17 (a) are number of optima found, and Fig. 3-15, Fig. 3-16 and Fig. 3-17 (b) are number of clusters formed, against generations, respectively. Table 3-5, Table 3-6 and Table 3-7 list performance measurements of these functions.

Note that we do not plot the number of identified peaks/clusters against the number of fitness evaluations, which unjustifiably assumes that the main cost of the algorithm attributes to fitness calculation and neglects the cost of other processes, e.g. evolutionary operations and manipulation of the niche pair list. Rather, as graphed in Fig. 3-15, Fig. 3-16 and Fig. 3-17, we compare the number of peaks/clusters found against the number of generations. Further, in order to compare the efficiency of these algorithms, their implementation time is listed in Table 3-8.

All experiments were run on a Sun Fire 280R machine, with dual 1015-MHz UltraSPARC III processors and 4.0 GB of RAM, running SunOS Release 5.8.

Function D_1

Function D_1 is shown in Fig. A–1. From Fig. 3-15 and Table 3-5, it can be seen that DNC forms much more clusters than DNC-RM, whereas its performance is slightly worse than DNC-RM, in that it reaches the equilibrium later, with lower success rate and less optima found.



(a) (b)

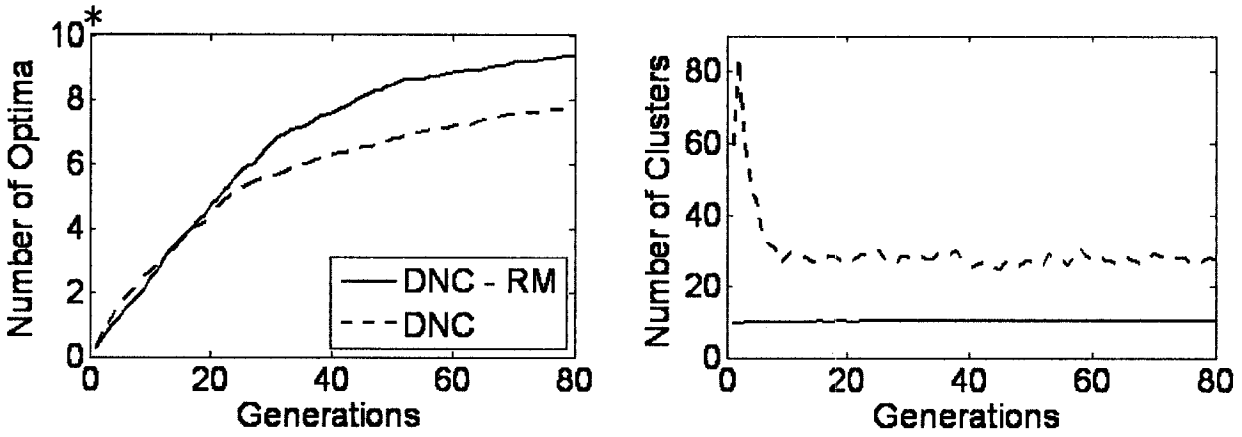
Fig. 3-15 Number of optima and clusters found for D_1

Table 3-5 Performance measurements on function D_1

Measurements	DNC - RM	DNC
Success Rate (%)	100	92
$O_{avg} / O_{max} / O_{min}$	9 / 9 / 9	8.92 / 9 / 8
$C_{avg} / C_{max} / C_{min}$	9 / 9 / 9	22.7 / 30 / 15

Function D_3

D_3 is typical two-dimensional multi-modal function with irregular optima, as shown in Fig. A–3. Still, DNC-RM outperforms DNC in all measurements shown in Table 3-6.



(a) (b)

Fig. 3-16 Number of optima and clusters found for D_3

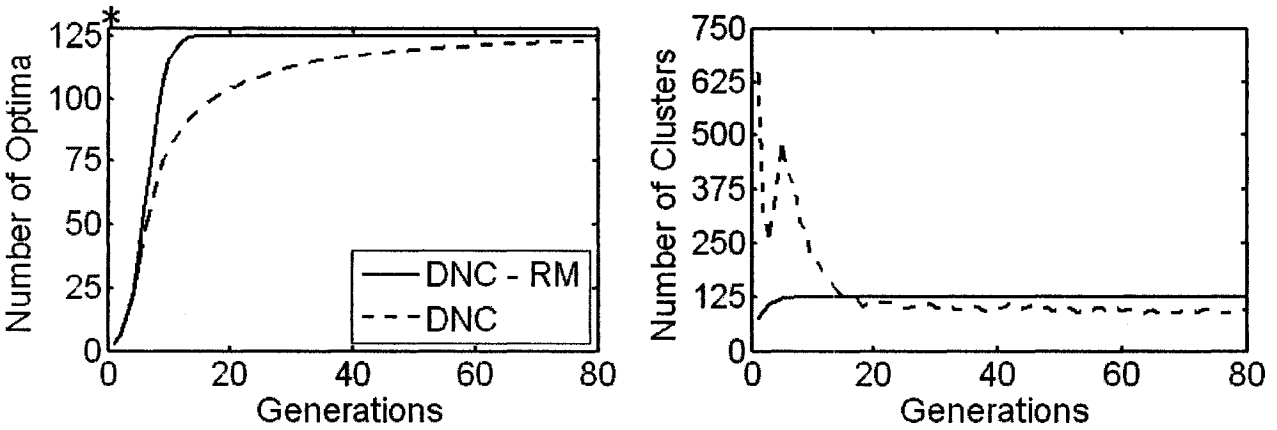
Table 3-6 Performance measurements on function D_3

Measurements	DNC - RM	DNC
Success Rate (%)	46	2
O_{avg} / O_{max} / O_{min}	9.36 / 10 / 8	7.78 / 10 / 6
C_{avg} / C_{max} / C_{min}	10.22 / 11 / 10	27.52 / 82 / 17

Function D_4

The optima in function D_4 are distributed evenly, but it can be easily extended in any n+1 dimensional space, with 5^n peaks. Fig. A–4 shows the function in a three dimensional space with n = 2.

To test the applicability of RM in a higher dimensional space, we set n = 3. There are altogether 125 peaks in this four dimensional space. As illustrated in Fig. 3-17 (b), DNC forms less clusters than DNC-RM from slightly earlier than the 20th generation. This is different from functions D_1 and D_2, with which DNC forms more clusters than DNC-RM.



(a) (b)

Fig. 3-17 Number of optima and clusters found for D_4

Table 3-7 Performance measurements on function D_4

Measurements	DNC - RM	DNC
Success Rate (%)	100	14
$O_{avg} / O_{max} / O_{min}$	125 / 125 / 125	123.02 / 125 / 120
$C_{avg} / C_{max} / C_{min}$	125 / 125 / 125	96.3 / 248 / 54

It is interesting to note that, with DNC, the average number of clusters is 96.3, less than the average number of optima found - 123.02 (Table 3-7). This is because we cumulate the number of optima throughout the process. The initial radius is large so that some niches may cover more than one optimum. Hence different optima may co-exist within one niche and emerge alternately. With RM scheme, however, niches are strictly separated with each of them focusing on a single optimum. Hence the number of clusters is always approximately equal to the number of actual peaks, as in Table 3-7 and Fig. 3-17 (b).

Finally, Table 3-8 lists CPU running time as well as the proportion of clustering and evolution for each function (in both seconds and percentage), respectively.

Table 3-8 CPU running time for functions D_1 - D_4

Functions	Evolution (s)		Clustering (s)		Miscellaneous (s)		Total (s)	
	DNC-RM	DNC	DNC-RM	DNC	DNC-RM	DNC	DNC-RM	DNC
D_1	0.3987 (32.00%)	0.5766 (57.69%)	0.8456 (67.88%)	0.4218 (42.21%)	0.0015 (0.12%)	0.001 (0.10%)	1.2458	0.9994
D_2	0.5367 (9.21%)	0.8379 (39.36%)	5.2686 (90.42%)	1.283 (60.27%)	0.0213 (0.37%)	0.0079 (0.37%)	5.8266	2.1288
D_3	429.9513 (61.84%)	512.1076 (57.74%)	258.4687 (37.18%)	373.7481 (42.14%)	6.7997 (0.98%)	1.0611 (0.11%)	695.2197	886.92

As expected, for all three functions, RM helps DNC to form desired number of clusters without *a priori* knowledge of the landscape. Furthermore, it does not rely on the size of the population. In fact, this size only needs to be sufficiently large so that each optimum can be allocated sufficient resources (chromosomes).

On the contrary, DNC with Euclidean distance metrics and HV function demonstrates demands on an appropriate size of the population. As discussed in Section 3.1.1, if this size is too large or small, the clustering effect will be affected negatively. This claim is supported by the facts observed in the experiments. It can be seen that, for functions D_1 and D_2, the average number of clusters formed (22.7 and 27.52, respectively) is much larger than the actual number of optima (9 and 10, respectively), because the population size for these two functions (90 and 100, respectively) is too large, resulting in many niches of small radii. On the other hand, however, the average number of clusters for function D_3 (96.3) is much smaller than the desired number (125), as the population size is too small for function D_3 and yields large niches that cover more than one optima.

A possible criticism of the RM algorithm is its scalability to high dimensional space due to its computational cost. It is true that clustering with RM accounts for most of the implementation time for functions D_1 and D_2. However, for higher dimensional

functions with many optima, e.g. D_3, the cost of evolution exceeds that of clustering substantially (Table 3-7). This is because the main cost of RM is its recursive fitness computation for midpoints. When the dimensionality increases, the cost of evolutionary operators grows dramatically and exceeds the cost of fitness calculation, which is more or less increased, but not as much as the former. As a matter of fact, as long as the cost of fitness calculation remains feasible, the RM algorithm itself is applicable even in a high dimensional space.

In terms of the total implementation cost, DNC-RM is generally higher than DNC (functions D_1 and D_2), but it is still acceptable with the same magnitude as the latter. However, for function D_3, DNC-RM is faster than DNC. As discussed above, this is because DNC entails sorting and manipulation of a large list of niche pairs. These two actions unavoidably induce a non-trivial computational cost to the whole process.

From Table 3-8, we can also see that for all three functions, DNC-RM spends less time in evolution than DNC, at the cost of clustering. It can be concluded that, with RM, the algorithm is able to form correct clusters effectively and stably, at the cost of acceptable overhead for clustering. When there are many optima in the landscape and the evolution process accounts for the majority of the computational cost, the overall algorithm is expected to be faster than those without RM.

Of course, the computational cost of RM is still a potential pitfall - especially when two individuals are on the same peak, the algorithm does not terminate until they finally converge, resulting in a much higher cost than the case when two individuals are on different peaks. Even worse, if the cost of fitness calculation is too high, RM may not be applicable.

One possible solution is to reduce runs of RM. Since the population is partially replaced with new chromosomes, RM can be applied on new individuals only. This is another advantage of RM over the distance metrics based techniques. With RM, survived old individuals are already clustered and will remain in their own clusters steadily. With the distance based methods, however, alteration of the midpoint or the niche radius may change the membership of the individuals within the cluster.

With a population of size N and P niches, suppose the number of new individuals is Q, the runs of RM are reduced from $N*P$ times to $Q*P$ times. Q is a proportion of N and is dependent on various evolutionary rates (e.g. elitism, crossover, mutation). $(N-Q)*P$ runs of high cost RM (between two individuals on the same peak) are also avoided.

Another possible solution to reduce runs of RM is to gradually decrease the frequency of clustering when the population tends to stabilize. Initially, clustering is performed at each generation. It is then performed every 2, 4 or 8 generations until termination.

In case the fitness calculation is prohibitively expensive, RM can still be applied in the offline training, that is, to analyze the distribution of the potential optima beforehand and provide *a priori* information about the topology of the landscape.

Finally, we note that although DNC-RM achieves better success rates than DNC, its performance for some functions is still not satisfactory. For example, for function D_2, the average number of optima found is 9.36 (desired: 10) and the success rate is 46%. This is due to the limitation of the DNC algorithm itself. Most evolutionary operators of DNC act globally, and selection pressure is realized globally as well. As the authors pointed out in [2], individuals are spread around the top of peaks, rather than clustered

tightly at the apex. If RM is applied in a multi-population based GA [29], where independent sub-populations (niches) are evolved separately and the selection pressure is focused locally about the peaks and within each sub-population, the performance is expected to be better.

3.3.3.3 Summary

Effective and efficient clustering plays a key role in evolution algorithms aiming for multi-modal optimization. In this section, a novel clustering algorithm Recursive Middling (RM) is proposed and compared to other clustering techniques, i.e. those based on Euclidean distance and Ursem's HV function [1], alone and within the framework of an evolution algorithm – Dynamic Niche Clustering (DNC) [2].

Experiments on benchmark functions show that, with an acceptable overhead, RM quickly forms stable clusters around actual optima, and considerably facilitates the evolution of the population. It outperforms other clustering techniques in terms of the correctness and stability of the clusters formed, as well as the number of optima found in the evolutionary algorithm tested.

Chapter 4 Bi-objective Multi-population Genetic Algorithm for Multi-modal Optimization

There are two major issues in multi-modal optimization: diversification and intensification. From a global point of view, diversification aims to maintain sufficient diversity within the population so that individuals are spread out widely within the search space. On the other hand, intensification is seen locally. It allows for congregation of individuals around potential optima so that each optimum region is fully exploited. Both diversification and intensification play key roles in the optimization process.

In BMPGA, diversification is realized *via* a novel bi-objective mechanism and a multi-population scheme. Unlike most other GAs that evolve the population towards a single fitness objective, BMPGA introduces a second *complementary* objective that is an essential property of all sought optima. Hence, the subpopulations of BMPGA are evolved toward both objectives, separately and simultaneously. This results in more diverse subpopulations than ones which would have resulted from a focused search guided by a single all-encompassing fitness objective.

In multimodal GAs, unjustified loss of species means loss of diversity. Inspired by the island model of GAs [67], BMPGA explicitly maintains a set of subpopulations, each clustered around one potential optimum. Hence, species are maintained consistently through these subpopulations.

As for the intensification, each subpopulation is evolved separately toward its optimum. This allows for focused exploitation of each optimum region.

4.1 Methodology

4.1.1 Overall Framework

The Bi-objective Multi-population Genetic Algorithm adopts dual fitness terms to enhance weaker optima and hence diversity of the population. Stable subpopulations focusing on different potential optima are formed *via* an effective and robust clustering algorithm. Each subpopulation is evolved independently so that the selection pressure is zeroed in on the apex of each optimum.

The overall framework of BMPGA is presented in Fig. 4-1. This process is akin to a standard GA, except that multiple subpopulations are evolved independently and are manipulated by clustering operations that work hand-in-hand with the evolutionary processes.

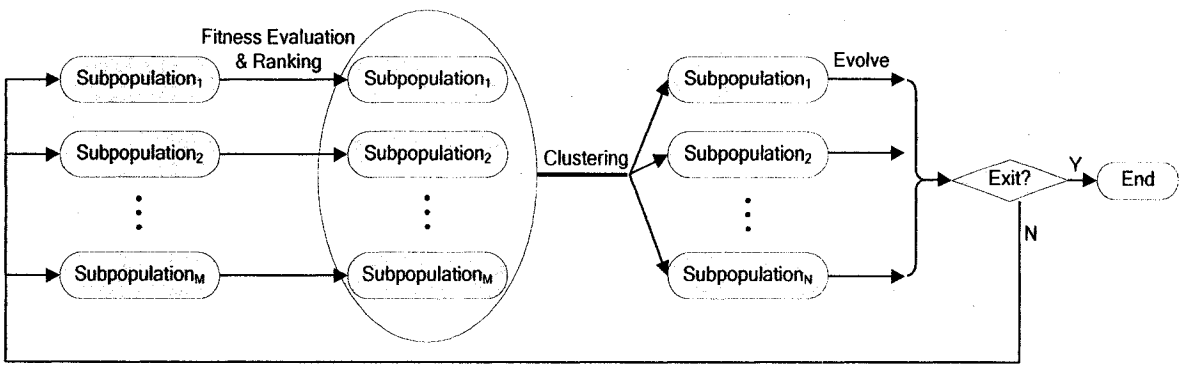


Fig. 4-1 The Bi-objective Multi-population Genetic Algorithm

As shown, the algorithm manipulates M subpopulations, with $M = 1$, initially. After evaluating and ranking the fitnesses of the individuals, these subpopulations are manipulated by a clustering process. Individuals around the same potential optimum are placed and maintained in the same subpopulation; and subpopulations approaching the same potential optimum are merged. After clustering, the number of subpopulations

becomes N, where N may or may not equal M. Each of these N subpopulations is evolved toward its own potential optimum. The algorithm is run repeatedly until the termination criteria are satisfied.

The overall outline of BMPGA can also be presented as high-level pseudo code.

```
Initialization;
While (! terminate) do
  Fitness evaluation & ranking;
  Clustering;
  For each sub-population do
    Evolution;
  End For
End While
```

Fig. 4-2 Pseudo code of BMPGA

BMPGA is divided into the following major processes:

1. Initialization (Section 4.1.4);
2. Fitness evaluation and ranking (Section 4.1.5): For each subpopulation: the chromosomes are evaluated and ranked with respect to two fitness terms;
3. Clustering (Section 4.1.6): Subpopulations are manipulated so that each of them is focused on a specific potential optimum area in the search space;
4. For each subpopulation do evolution (Section 4.1.7).
5. Steps 2-4 are repeated until the termination criterion is satisfied (Section 4.1.8).

4.1.2 Chromosome Encoding

In principle, BMPGA does not necessitate a specific chromosomal representation. Each gene, which is a parameter of the object function, can adopt a binary or real-valued format. One of the algorithms to be compared to BMPGA works in a normalized

parameter space with 15-bit binary genes [2, 28]. For a fair comparison between these algorithms, we adopt the same representational format.

4.1.3 Parameters

The parameters of BMPGA are:

- N_s: The size of subpopulations: all equal;
- R_e: Elitism percentage;
- R_c: Crossover possibility;
- R_m: Mutation possibility;
- R_d: Percentage of least fit individuals discarded;
- σ: Convergence threshold – two individuals are deemed to converge to one if the distance between them is smaller than σ.

4.1.4 Initialization

The algorithm starts with a single subpopulation of size N_s. Although N_s is empirically devised, it is problem-independent and can be applied to multimodal landscapes of variable dimensionality and number of optima. Each gene of each chromosome is randomly assigned a value within [0, 1].

4.1.5 Fitness Evaluation and Ranking

In most GAs, the probability of selection of an individual is proportional to its relative fitness or ranking within the population [68]. Regardless of the specific selection scheme used, a basic principle is that fitter individuals, on average, must have a greater probability of survival than the rest.

While fully aware of that principle, we propose a bi-objective scheme in order to

promote diversity in search without compromising the optimality of the ultimate goal. This scheme does not make BMPGA a typical GA-based multi-objective optimizer, which attempts to satisfy several conflicting objectives, resulting in a set of Pareto-optimal solutions [69]. The two objectives used in BMPGA (say A and B) do not conflict with each other. Rather, they represent different but complementary quality considerations allowing for less focused search for the optima. Both fitness objectives reach their optima *concurrently* at the each optimum.

A simple scenario will help to interpret this scheme. Suppose an individual *I* is ranked low with respect to the fitness objective A, but high with respect to B within its population. If, during the evolution, only A is used as the measure of fitness (as in most traditional GAs), *I* has trivial likelihood of survival and will very likely be replaced by better individuals soon. However, this does not necessarily mean that *I* was a bad solution. *I* might be a promising candidate that leads to a much better solution after a number of evolutionary cycles or even an optimal solution in another optimum area. Therefore, preservation of promising candidates like *I* is important. If B is incorporated as a second fitness term, *I* is much more likely to survive. Consequently, diversity of the population is promoted.

A question that arises is why not use more than two objectives to further enhance diversity. Generally speaking, too many objectives will induce more unnecessary costs and slower convergence of the subpopulations since individuals are more divergent rather than focused. Therefore, we believe that the bi-objective scheme is a reasonable compromise between diversity and efficiency.

After fitness evaluation, individuals are sorted with respect to both fitness

objectives, and as such two ranking lists are maintained.

Generally speaking, the design of the objectives should follow the following criteria:

- They should be one of the intrinsic characteristics of the optimum we are seeking;
- The two objectives should not be completely independent, that is, their optima usually occur concurrently.

Obviously the objectives are application-dependent. Details of specific objectives for a different application will be discussed in Chapter 5.

BMPGA uses two fitness terms. One of the fitness terms, naturally, is the given objective function itself, denoted as $f(x)$.

For a differentiable function, there usually is:

$$\nabla f(x^*) = 0 \tag{4.1}$$

where x^* is a local maximum/minimum of f(x), and $\nabla f(x^*)$ is its gradient. The gradient of an n-dimensional function $f(x)$ has a component for each direction. Hence we have:

$$g(x) = \frac{\sum_{i=1}^{n} \left| \frac{\partial f(x_i)}{\partial x_i} \right|}{n} \tag{4.2}$$

as the second objective function, the minimum of which is to be sought.

Alternatively, $g(x)$ can also be the norm of the vector $[\frac{\partial f(x_1)}{\partial x_1}, \frac{\partial f(x_2)}{\partial x_2}, \ldots \frac{\partial f(x_2)}{\partial x_2}]$. We leave it for future work.

Our current implementation handles functions for which a gradient can be

estimated numerically. If the function is non-differentiable and badly-behaved, e.g. a function with large discontinuities, the gradient may not be an appropriate measure. There may still be ways to design the secondary fitness objectives, but this is beyond the scope of this thesis.

4.1.6 Clustering

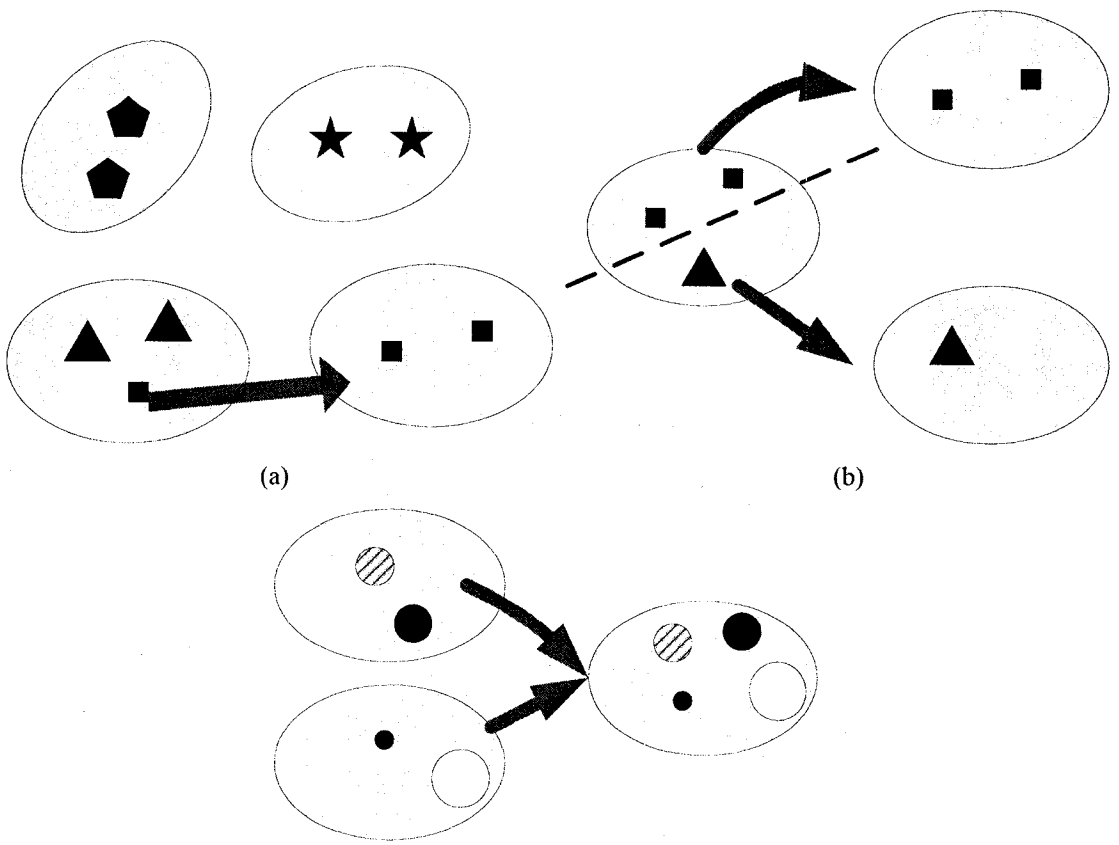


Fig. 4-3 The Clustering Actions
(a) Migration; (b) Splitting; (c) Merging

The multi-population scheme of BMPGA demands effective and robust identification of subpopulations. These subpopulations, also called clusters, are manipulated *via* three major clustering operations: migration, splitting, and merging, as described below and shown in Fig. 4-3.

- Migration (Fig. 4-3 (a)): An individual that does not belong to its own cluster is moved to an existing cluster more appropriate for it.
- Splitting (Fig. 4-3 (b)): An individual that does not belong to its own or any other cluster is removed from its own cluster and placed in a newly created cluster.
- Merging (Fig. 4-3 (c)): Different clusters are merged into one cluster, if they are deemed to be converging towards the same optimum. This is done by summing all of their individuals, and then selecting the best N_s (cluster size) for the new cluster.

```
SUB: set of current subpopulations

Migration & Splitting:
For each subpopulation s in SUB
  For each new individual i in s
    If i does not belong to s
      For each subpopulation s* in SUB (s* ≠ s)
        If i belongs to s*
          i migrates to s*;
          Break;
        End if
      End for
      If i does not belong to all s*≠s
        Create a new subpopulation around i;
      End if
    End if
  End for
End for

Merging:
For subpopulation i = 1 to NumberOf(SUB)
  For subpopulation j = i+1 to NumberOf(SUB)
    If i and j are around the same optimum
      Merge i and j;
      Remove j from SUB;
    End if
  End for
End for
```

Fig. 4-4 Pseudo code of clustering

Fig. 4-4 presents the pseudo code description of the three clustering operations.

Once an individual is correctly placed into an appropriate subpopulation s, it will stay in s without straying to other subpopulations, unless s itself is merged with other subpopulations. Hence, re-clustering of *old* individuals inherited from previous generations is simply unnecessary. Only *new* individuals generated initially or through evolution are examined to determine their appropriate subpopulations. This strategy effectively lessens the cost of clustering without causing straying effects.

Merging does not occur in every generation. Rather, it is implemented periodically to save computational cost. In addition, merging is not carried out during earlier generations to allow for formation of stable subpopulations around potential optima. If the number of individuals in a merged subpopulation exceeds N_s, the best N_s individuals are maintained in that subpopulation, while the rest are simply discarded.

It is worth noting that there are two relationships in clustering: individual-cluster and cluster-cluster. Since a cluster can be uniquely represented by its center (i.e. the up-do-date fittest individual within the cluster), both relationships can be represented as individual-individual relationships. This allows for the introduction of our own Recursive Middling clustering algorithm, explained in full in Chapter 3.

4.1.7 Evolution

In BMPGA, each subpopulation is evolved independently towards its own optimum. Evolution proceeds mainly *via* selection and diversification. Selection tends to eliminate those individuals in the population with lower than average fitness, focusing the search on promising areas of the fitness surface. Diversification expands the search towards new and potentially promising areas. In BMPGA, selection is realized using elitism and rank-

based fitness proportionate selection [68]. Diversification is realized *via* recombination and mutation.

The basic outline of evolution is shown in Fig. 4-5. Its basic steps are similar to those of a standard GA, except that a proportion (R_d) of the worst individuals are discarded and replaced by newly generated individuals.

```
SUB: set of current subpopulations

For each s in SUB:
   Create a new subpopulation s* = empty;
   s* = Elitism(s);
   s* = s* □Recombination(s);
   Mutation(s*);
   While (Size(s*)<Ns)
    s* = s* □ randomly generated individual;
   End while
End for
```

Fig. 4-5 The evolution process

If the size of a subpopulation is less than N_s, the gap is filled with randomly generated individuals. Hence for a subpopulation of size N (where $N \leq N_s$), the number of new individuals generated by this action is

$$N_{new} = N \cdot R_d + N_s - N \tag{4.3}$$

This action introduces new individuals into the subpopulation and enhances its diversity.

4.1.7.1 Elitism

Full local elitism is adopted in the sense that a proportion (R_e) of the fittest individuals in both ranking lists are copied to the next generation without alteration. Since a new emergent subpopulation may start from a couple of seeds, the computed number of elites

may be less than 1. In such cases, the fittest individual is nevertheless maintained in the subpopulation in order to preserve this subpopulation from extinction. As a result, all existing species are maintained firmly and will not be subject to extinction.

```
s*: new subpopulation

Elitism(subpopulation s)
  Num_elite = Size(s) * Re;
  If (Num_elite<1)
    Num_elite = 1;
  End if
   Copy best Num_elite individuals from
f(x) ranking list of s to s*;
   Copy best Num_elite individuals from
g(x) ranking list of s to s*;
End Elitism
```

Fig. 4-6 Elitism

4.1.7.2 Recombination and Mutation

Recombination only occurs between individuals in the same subpopulation. Two pairs of parents are selected *via* rank based proportional selection from the two ranking lists, respectively. With a given probability R_c, these parents generate two pairs of offspring, which are then put into the new subpopulation s^*. This process is repeated until the size of s^* reaches $(1\text{-}R_d)N$ (where N is the size of the original subpopulation s).

```
s*: new subpopulation

Recombination(subpopulation s)
  Lim = Size(s) * (1-Rd);
  While (Size(s*)<Lim)
    Select pairs of f-parents from f(x) list of s;
    f-offspring = recombine f-parents;
    Select pairs of g-parents from g(x) list of s;
    g-offspring = recombine g-parents;
    Put f-offspring and g-offspring in s*;
  End while
End Recombination
```

Fig. 4-7 Recombination

The default recombination is realized *via* 1-point crossover (Section 1.1.1.1.5). A pivot is selected at random, and the parents' genes on either side of the pivot are swapped to create two offspring.

Mutation is carried out with probability R_m. It is generally higher than the normal rate of 0.01-0.05, in order to enhance the diversity of the population. A gene to be mutated is randomly selected from the chromosome, and then arbitrarily assigned a new value from its valid range.

It should be noted that both crossover and mutation may vary, depending on different applications.

4.1.7.3 Population Control

As discussed above, after recombination and mutation, the new population reaches $(1-R_d)N$. The gap in the new subpopulation (R_dN) is then filled with new individuals until the size of s^* finally reaches N_s, as shown in Fig. 4-5.

We have:

$$N_s = N_{elite} + N_{recomb} + N_{new} = (1 - R_d) \cdot N + N_{new} \tag{4.4}$$

where N_{elite} is the number of elites in s, N_{recomb} is the number of offspring; the sum of N_{elite} and N_{recomb} is $(1-R_d)*N$. The remaining vacancy is filled by N_{new} new individuals, with N_{new} given by equation (4.3).

As a partial reshuffle of the population, the population control is a more drastic action than regular mutation, and it further enhances the diversity.

4.1.8 Termination Criterion

In general, the termination criterion can be, but not limited to, a maximum number of

fitness evaluations or a maximum number of generations.

4.2 Experiments

4.2.1 BMPGA versus Compared Algorithms

BMPGA is a population based parallel GA. It explicitly maintains a set of subpopulations, each focused on a potential optimum. The main features of BMPGA are:

1. Bi-objective scheme: individuals are evaluated and manipulated relative to two fitness terms;
2. Recursive middling: individuals are clustered based on topological information;
3. Local evolution: each subpopulation is evolved independently toward its potential optimum.

Multi-population GA (MPGA) is a variant version of BMPGA and also a typical population based GA. The main difference between MPGA and BMPGA is that BMPGA uses a bi-objective scheme instead of MPGA's single-term fitness. Hence, a comparison between MPGA and BMPGA would demonstrate the impact of the bi-objective scheme on GA based multi-modal optimization.

Multi-national GA (MNGA) [1] is another population based GA. MNGA uses the Hill-Valley function for clustering. As such, MNGA's clustering technique belongs to the family of topology based clustering approaches. It is interesting to compare MNGA to GAs utilizing distance based clustering methods, as well as our own BMPGA, which uses RM. Another interesting feature of MNGA is that, unlike other typical population based GAs [11], [25], [23], which evolves a variable-sized population containing a number of fixed-sized clusters, MNGA evolves an overall population of fixed size, containing a

number of variable-sized clusters.

Dynamic Niche Clustering (DNC) [2] is a typical genetic operator based GA. It belongs to the family of Sharing GAs [20], which are one of the most popular multimodal GAs to date. Unlike population based GAs that evolve local subpopulations separately, DNC evolves the whole population globally. It uses a hybrid clustering method that makes use of both distance and topological information. It is interesting to compare this method to RM, which only uses topological information. Finally, according to our knowledge, DNC is the only version of Sharing GA that is able to identify clusters of varied radii. That gives DNC a potential advantage over other multi-modal GAs in the literature.

Clearing is another interesting genetic operator based GA [45]. It is similar to most Sharing GAs in that it applies a single global threshold to all clusters. It is interesting to compare this static distance based approach to other dynamic distance based approach (such as DNC) and topology based approaches (such as those utilizing RM or HV).

BMPGA, MPGA and MNGA adopt full local elitism in the sense that the elites of every subpopulation, regardless of its size, are preserved. In contrast, both DNC and Clearing implement partial local elitism. Their implementations are slightly different though. In DNC, only elites within large niches (> 5% population size) are preserved. On the other hand, Clearing preserves the winners of each species with fitness values greater than the average fitness of the whole population.

In this study, we compare BMPGA to MPGA, MNGA [1], DNC [2] and an enhanced Clearing [3] GA (denoted as CLEARING) that outperforms its original version.

Table 4-1 presents the main features of the algorithms used in the empirical study (to follow). These features are: the categories of the algorithms, their evolution style and clustering approaches, the number of fitness objectives they use and population sizing, which indicates whether the size of a subpopulation or that of the global population is kept constant.

Table 4-1 BMPGA versus other multimodal GAs

Algorithms		BMPGA	MPGA	MNGA [1]	DNC [2]	CLEARING [3]
Category		Population	Population	Population	Genetic operator	Genetic operator
Evolution		Local	Local	Local	Global – partial local elitism	Global – partial local elitism
Fitness		Two	One	One	One	One
Clustering	Approach	Topology - RM	Topology - RM	Topology - HV	Dynamic distance - Euclidean + Topology- HV	Static distance - Euclidean
	Center	Up-to-date best	Up-to-date best	Average of the government	Weighted average of relative physical positions of all cluster members	Up-to-date best
Population Sizing		Subpopulation	Subpopulation	Population	Population	Population

4.2.2 Benchmark Functions

Four diverse benchmark functions of different dimensionality (from 1 to 3) and landscapes (evenly or unevenly distributed optima of equal or different heights and volumes) are used. These functions, denoted as D_1, D_2, D_3 and D_4 in Appendix A, have been used by a number of researchers in assessing the quality of their respective multimodal GAs [1, 2, 23, 26, 28, 50]. The aim is to locate the maxima of these functions within a given range.

4.2.3 Configuration

We are interested in comparing the performance of BMPGA, MPGA, DNC, MNGA and CLEARING with different population sizes. Suppose each potential optimum is assigned

N_s individuals, on average. For BMPGA and MPGA, the size of each subpopulation is N_s (N_s is 5, 10, 15 and 20, respectively). For DNC, MNGA and CLEARING, on the other hand, the total population size equals N_s (average cluster size) times the number of actual optima. All other parameters (described in Section 4.1.2) are given identical values:

- $R_e = 0.1$
- $R_c = 0.8$
- $R_m = 0.1$
- $R_d = 0.15$
- $\sigma = 0.025$ (D_1, D_3, D_4); 0.5 (D_2)

Note that σ is empirically devised and may well differ on different landscapes. σ is also used to verify the optimality of an individual. An individual is deemed to be an optimum if the Euclidean distance between this individual and the actual optimum is less than σ.

For each function, all five GAs are run for the same number of fitness evaluations E_{max}. The value of E_{max} for each function is:

- For D_1, $E_{max}=10^5$
- For D_2, $E_{max}=10^6$
- For D_3, $E_{max}=10^6$
- For D_4, $E_{max}= 2*10^6$

There are also some algorithm specific parameters. CLEARING requires a global threshold σ_{dis} to separate species. σ_{dis} is estimated using Deb and Goldberg's method [48], as equation (3.1) in Section 3.1.1. σ_{dis} for each function is given in Table 4-2.

Table 4-2 Global threshold for CLEARING

Function	Dimensionality	Number of optima	σ_{dis}
D_1	1	9	0.056
D_2	2	25	0.1414
D_3	2	10	0.2236
D_4	3	125	0.2

In MNGA, the sampling vector for clustering is [0.25, 0.50, 0.75]. However, when it is about to merge clusters, two extra samples are calculated. Hence, the sampling vector becomes [0.02, 0.25, 0.50, 0.75, 0.98] because, as explained in [1], the merging of clusters is a more drastic operation than the migration of individuals. The government size is 2.

4.2.4 Performance Evaluation

The performance of the five GAs is evaluated with respect to the following measures:

- *Percentage* of successful runs: *Suc*;
- Average/*maximum*/minimum number of peaks found at the end of a run: $O_{ave}/O_{max}/O_{min}$;
- Standard deviation of *number* of peaks found in all runs: O_{cons};
- Average/maximum/*minimum* number of clusters found at the end of a run: $C_{ave}/C_{max}/C_{min}$;
- Standard deviation of *number* of clusters found in all runs: C_{cons};
- Average CPU running time.

Suc, $O_{ave}/O_{max}/O_{min}$, and O_{cons} reflect the capability of the algorithms of finding and holding all optima. $C_{ave}/C_{max}/C_{min}$ and C_{cons} indicate the ability of the algorithms to form stable clusters. O_{cons} and C_{cons} measure the consistency of the algorithms in locating

the optima and forming clusters, respectively. Finally, average CPU time reflects the efficiency of the algorithms.

4.2.5 Results

For each population size, each algorithm was run 30 times. All the experiments were run on an Intel Xeon 2.66 GHz with 512 KB of cache and 512 MB DDR RAM, running Red Hat Linux 8.0.3.2-7.

For the sake of clarity, from now on, we use the term *cluster* in all algorithms unless otherwise stated. As discussed above in Chapter 3, a cluster is a group of chromosomes deemed to explore the same potential optimum area. It is equivalent to a subpopulation in BMPGA and MPGA, a nation in MNGA, a niche in DNC and a species in CLEARING.

We also use the term *species* to represent an actual optimum surrounded by a neighborhood inferior to this optimum. Ideally, the number of clusters formed by various algorithms is equal to the number of actual optima or species. However, depending on specific algorithms, a cluster may contain one or more species, and a species may spread in several clusters. A key to the success of an algorithm is its ability to form correct clusters as well as to find and preserve species.

During each run, the population is checked periodically (every $E_{max}/100$ evaluations) for the number of optima found and the number of clusters formed. That gives us 100 samples per run. Fig. 4-8, Fig. 4-9, Fig. 4-10 and Fig. 4-11 chart the results for functions D_1 to D_4, respectively. Within each figure, sub-figures (a), (c), (e) and (g) show the number of optima found against the number of evaluations; sub-figures (b), (d), (f) and (h) show the number of clusters formed against the number of evaluations. The x

axis of all sub-figures represents the number of evaluations, scaled to [0, 100]. The y axis of all sub-figures has the actual number of optima marked by an asterisk.

In sub-figures (a), (c), (e) and (g) of Fig. 4-8 - Fig. 4-11, each one of DNC, MNGA and CLEARING has two curves named "algorithm" and "algorithm-R", respectively. "algorithm" counts all optima previously and currently found. It reflects an algorithm's ability to *find* the optima (but not necessarily *preserve* them). It is a historically accumulated result and is non-decreasing. On the other hand, "algorithm-R" only counts the optima that exist in the population at the point of sampling. It is a "real time" result and its curve may vibrate. "algorithm-R" reflects an algorithm's ability to both *find* and *preserve* the optima.

A common approach to explore unknown landscapes sees the algorithm run for a pre-set duration (e.g. number of fitness evaluations or generations) and obtain the results at the end of the run. In this sense, "algorithm-R" is of more practical significance. As described above, the samplings of "algorithm-R" are independent since the results of each sampling are not influenced by the results of any previous samplings. This is equivalent to run the algorithm for different duration (for example, $E_{max}/100$ evaluations, $2*E_{max}/100$ evaluations ... $100*E_{max}/100$ evaluations, and so on) and then count the optima at the end of the duration.

It is observed that both "algorithm" and "algorithm-R" curves of BMPGA and MPGA overlap perfectly (possible reasons will be discussed later). Therefore, each of these two algorithms only has one curve describing its performance.

Here we will compare the performance of BMPGA and MPGA against that of DNC, MNGA, and CLEARING. Unless specifically stated, all results of DNC, MNGA

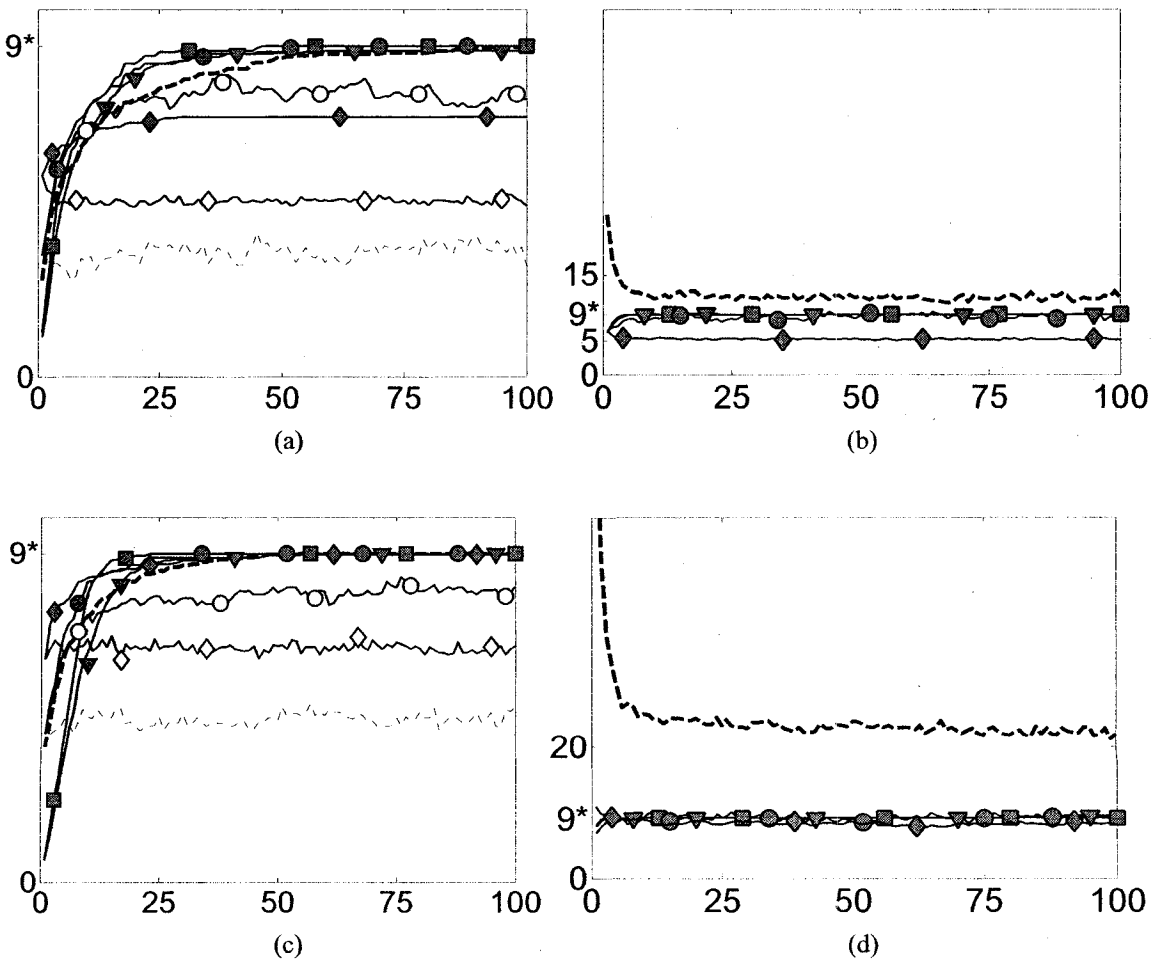
and CLEARZNG are "algorithm-R" results.

At each sampling point, the number of clusters in the population is counted. This gives us a "real-time" value. Hence sub-figures (b), (d), (f) and (h) do not have "algorithm-R" curves.

Table 4-3 - Table 4-22 provide the results of various measurements described above. "Real" represents real-time results. "Accu" represents accumulated results. All results (including figures and tables) are for cluster sizes (N_s) 5, 10, 15 and 20.

Function D_1

Function D_1 (Fig. A–1) is a typical 1-dimensional multimodal function with 9 optima. It features an uneven distribution of optima of varied heights and volumes.



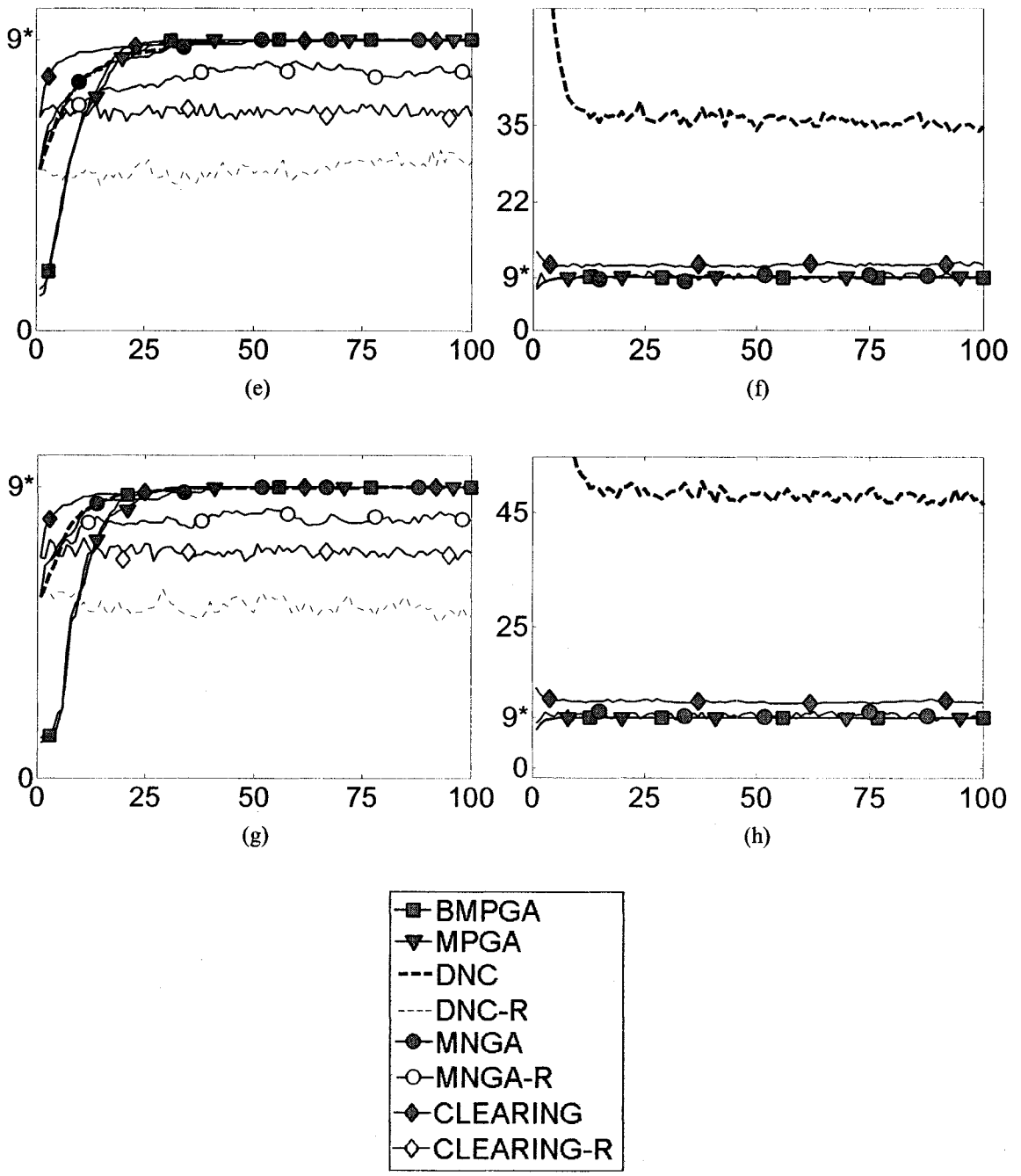


Fig. 4-8 Performance on Function D_1

Number of optima found against number of evaluations: (a) $N_s = 5$; (c) $N_s = 10$; (e) $N_s = 15$; (g) $N_s = 20$;
Number of clusters formed against number of evaluations: (b) $N_s = 5$; (d) $N_s = 10$; (f) $N_s = 15$; (h) $N_s = 20$.

As expected, all "algorithm-R" results of DNC, MNGA and CLEARING are much worse and less consistent than their corresponding "algorithm" results. Most

"algorithm-R" curves also fluctuate more drastically than their corresponding "algorithm" curves. These facts indicate that DNC, MNGA and CLEARING are unable to preserve species, even though the species may have reached their optima.

Now we take a closer look at each of these algorithms. As discussed above, the initial radius σ_{init} of DNC is calculated using equation (3.2). The value of σ_{init} is usually very small because N (the total population size) >> P (the number of optima). Hence DNC starts with a lot of small clusters, which decreases in number rapidly due to merging (Fig. 4-8 (b), (d), (f), (h))). At a certain point in time, the population reaches equilibrium. The number of the clusters remains roughly constant. It may still decrease, but only slightly and insignificantly. The number of the clusters increases as N_s increases.

Table 4-3 Measurements for function D_1, N_s = 5

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	90	0	96.67	10	100	0	3.33
$O_{ave}/O_{max}/O_{min}$	9/9/9	8.87/9/7	3/5/2	8.97/9/8	7.53/9/6	9/9/9	4.63/6/2	7.07/9/4
O_{cons}	0	0.4342	0.7878	0.1826	0.8193	0	0.7649	1.2847
$C_{ave}/C_{max}/C_{min}$	9/9/9	9.03/10/9	11.50/24/7		8.93/11/7		7.0/8/6	
C_{cons}	0	0.1826	3.2879		1.5742		0.5872	

Table 4-4 Measurements for function D_1, N_s = 10

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	100	0	100	20	100	0	96.67
$O_{ave}/O_{max}/O_{min}$	9/9/9	9/9/9	4.87/6/3	9/9/9	8.1/9/7	9/9/9	6.50/8/5	8.97/9/8
O_{cons}	0	0	0.8996	0	0.5477	0	0.6823	0.1826
$C_{ave}/C_{max}/C_{min}$	9.10/10/9	9.43/11/9	22.13/30/14		9.03/11/8		8.07/11/6	
C_{cons}	0.3051	0.6261	3.4614		0.8899		1.4368	

Table 4-5 Measurements for function D_1, $N_s = 15$

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	100	0	100	16.67	100	0	96.67
$O_{ave}/O_{max}/O_{min}$	9/9/9	9/9/9	5.23/8/3	9/9/9	7.90/9/7	9/9/9	6.63/8/6	8.97/9/8
O_{cons}	0	0	1.1651	0	0.6618	0	0.6687	0.1826
$C_{ave}/C_{max}/C_{min}$	9/9/9	9.17/10/9	34.87/42/24		9.60/11/9		11.23/14/7	
C_{cons}	0	0.3790	4.0830		0.7701		1.5687	

Table 4-6 Measurements for function D_1, $N_s = 20$

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	100	0	100	20	100	0	100
$O_{ave}/O_{max}/O_{min}$	9/9/9	9/9/9	4.9/6/4	9/9/9	7.97/9/7	9/9/9	6.97/8/6	9/9/9
O_{cons}	0	0	0.8277	0	0.6687	0	0.7184	0
$C_{ave}/C_{max}/C_{min}$	9/9/9	9.10/10/9	46.50/59/31		9.73/11/9		11.90/13/8	
C_{cons}	0	0.3051	5.9582		0.9072		1.0289	

Table 4-7 Average CPU running time for function D_1

Pop size	BMPGA	MPGA	DNC	MNGA	CLEARING
5	0.48	2.28	0.48	0.21	1.60
10	0.49	0.24	1.16	0.22	2.54
15	0.53	0.22	3.11	0.24	4.48
20	0.54	0.21	6.40	0.26	7.52

In CLEARING, σ_{dis} is 0.056. However, the actual minimum distance σ_{min} between adjoining optima is 0.032 (All these distances are computed in a normalized space). As a matter of fact, D_1 has 9 peaks. Hence, there are altogether 8 pairs of neighboring optima. The distance between 3 pairs of peaks is less than σ_{dis}. Consequently, it is not unusual that two adjacent species are mistakenly placed in the same cluster and the weaker one is

cleared out. As can be seen from Fig. 4-8 and Table 4-3 - Table 4-7, CLEARING-R has 0 success rates for all N_s values; the average number of optima found is only better than DNC-R.

MNGA-R outperforms both DNC-R and CLEARING-R. Its success rates are all below 20%. Although it forms approximately the correct number of clusters, the average number of optima found starts from 7.53 at $N_s = 5$ and increases slightly to about 8 at $N_s \geq 10$.

In contrast to DNC, MNGA and CLEARING, BMPGA and MPGA are able to maintain species. Initially, both BMPGA and MPGA find less optima than DNC-R, MNGA-R and CLEARING-R (this phenomenon is more manifest when $N_s \geq 10$). However, BMPGA and MPGA quickly exceed them. It can be seen that BMPGA works perfectly with 100% success rates for all N_s values. MPGA, on the other hand, is slightly inferior to BMPGA at $N_s = 5$, where its success rate is 90%. However, the performance of MPGA improves with increasing N_s values. Its success rates reach 100% for $N_s \geq 10$. The curves of BMPGA and MPGA almost overlap when $N_s = 20$.

Since DNC forms more than the required number of clusters, the average number of individuals assigned to each species is less than N_s. This can cause insufficient exploitation of areas of potential optima. Therefore, DNC constantly fails in finding all the optima. The performance of DNC-R is the worst among all GAs.

In terms of overall performance, the GAs can be listed from best to worst as: BMPGA, MPGA, MNGA-R, CLEARING-R, DNC-R.

In terms of CPU running time (Table 4-7), MPGA is the slowest when $N_s = 5$. Nevertheless, its running time drops quickly from $N_s = 10$. CPU time of MNGA is a

fraction of CPU time of MPGA at N_s = 5, whereas CPU times of both algorithms are approximately the same at other N_s values. So generally speaking, MNGA is faster than MPGA. CLEARING is the slowest since it is only faster than MNGA at N_s = 5 and remains the slowest for all N_s values. The CPU running times of DNC and CLEARING have approximately identical orders of magnitude. The CPU running times of BMPGA, MPGA and MNGA have approximately the same orders of magnitude. The CPU running times of DNC and CLEARING are generally one order of magnitude greater than the CPU running time of BMPGA, MPGA and MNGA. As N_s increases, the differences become more marked.

These algorithms can be listed from fastest to slowest as: MNGA, MPGA, BMPGA, DNC, CLEARING.

Function D_2

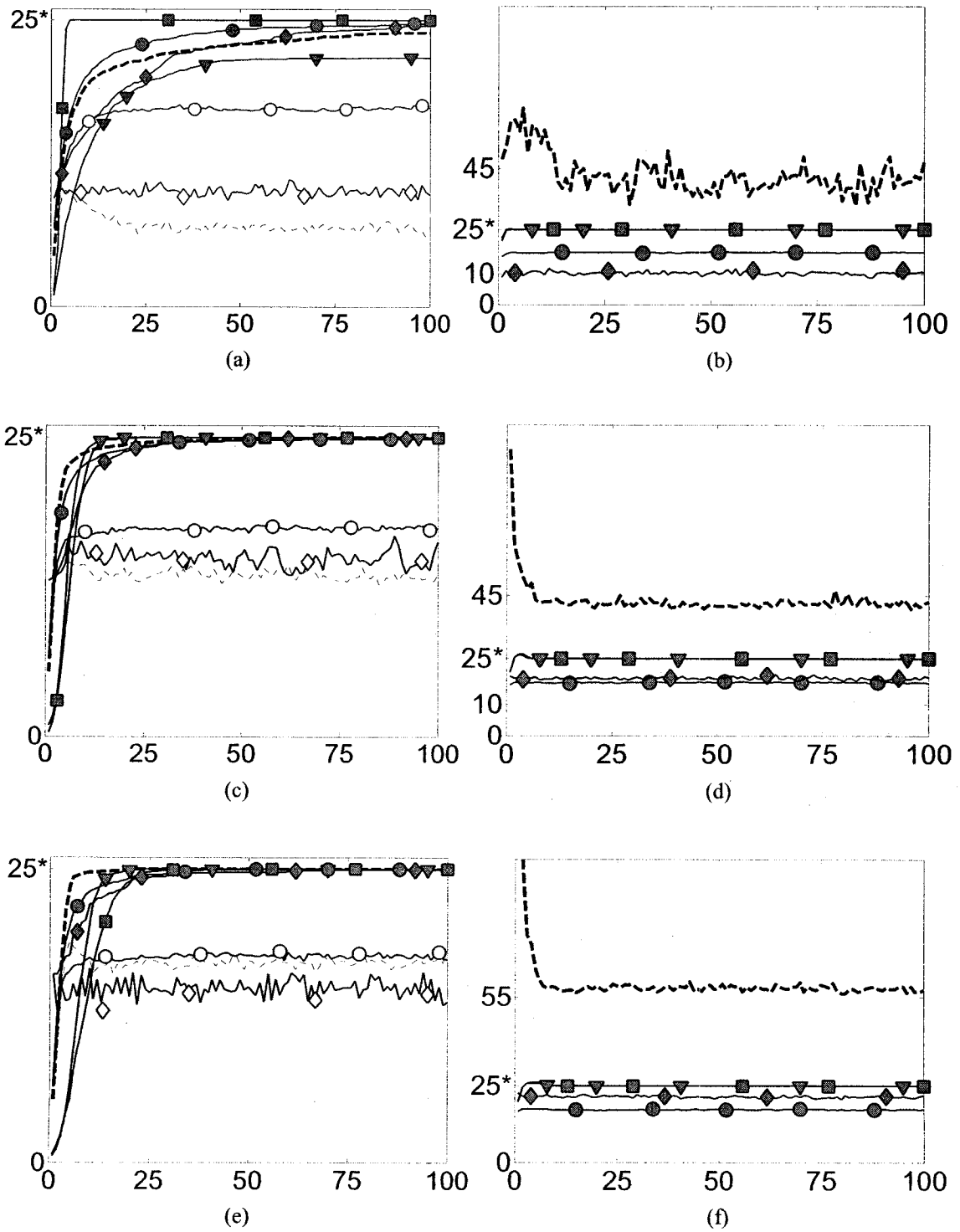
D_2 is a 2-dimensional multimodal function with 25 optima. Its optima are distributed evenly in a 5×5 square but the differences between their magnitudes are large (Fig. A–2).

All "algorithm-R" results of DNC, MNGA and CLEARING are worse than their corresponding "algorithm" results.

Again, BMPGA achieves 100% success rates for all N_s values. Similar to D_1, BMPGA is better than MPGA in terms of success rates, average number of optima found and consistency at N_s = 5. MPGA achieves100% success rates starting from N_s = 10.

For CLEARING, σ_{dis} = 0.141 and σ_{min} = 0.122 ($\sigma_{dis} > \sigma_{min}$). This is the same case as that of D_1. Unlike D_1, however, which has an uneven distribution of optima, the

optima of D_2 are distributed evenly in a 5×5 square and are equidistant to each other. Therefore, the possibility of placing neighboring peaks in one cluster is much higher for D_2.



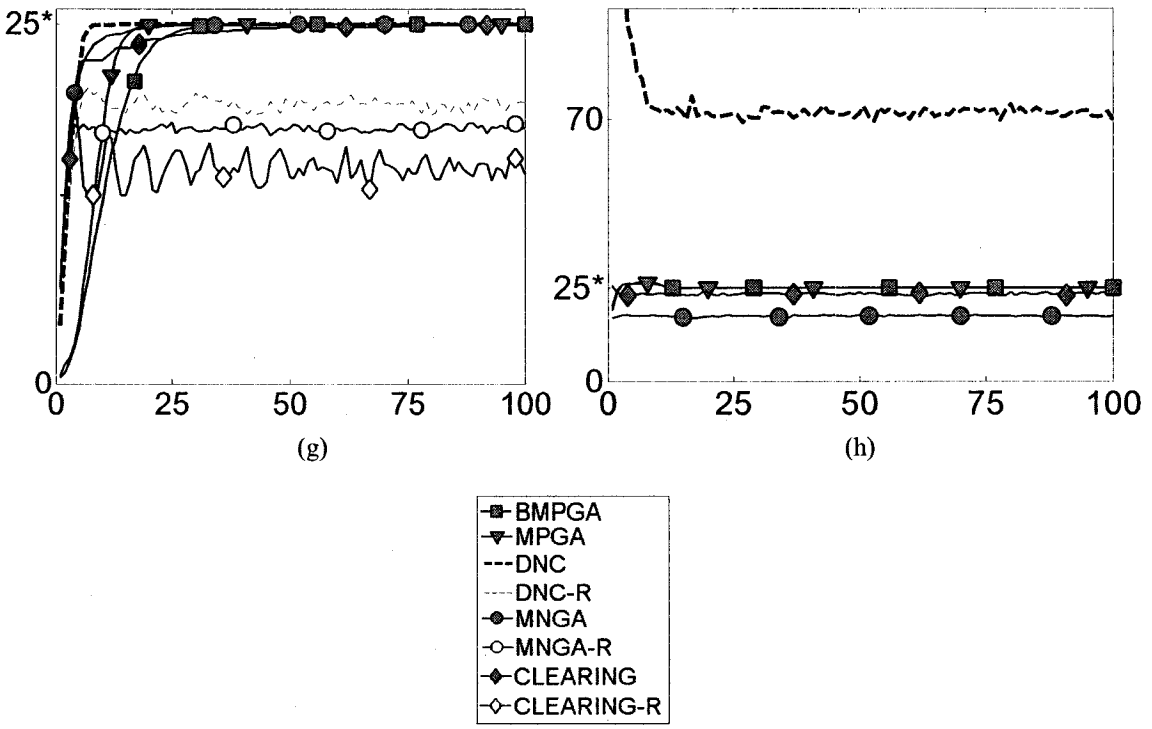


Fig. 4-9 Performance on Function D_2

Number of optima found against number of evaluations: (a) $N_s = 5$; (c) $N_s = 10$; (e) $N_s = 15$; (g) $N_s = 20$;
Number of clusters formed against number of evaluations: (b) $N_s = 5$; (d) $N_s = 10$; (f) $N_s = 15$; (h) $N_s = 20$.

The success rates of MNGA-R are all equal to 0. On average, MNGA-R detects 17 out of 25 optima and forms about 17 clusters for all N_s values. The loss of clusters may be due to the Hill-Valley clustering function used and to the symmetric nature of the landscape. Recall that we use the sampling vector [0.25, 0.5, 0.75] for the migration of individuals. These interior samples are equidistant to each other. Since the optima of function D_2 are also equidistant to each other and have approximately the same volume, usually no valley is found between species. The result is that distant species, even though there are other species lying between them, may be placed in one cluster. This is the case in Fig. 3-4 (a).

Table 4-8 Measurements for function D_2, $N_s = 5$

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	30	0	33.33	0	73.33	0	93.33
$O_{ave}/O_{max}/O_{min}$	25/25/25	21.70/25/7	6.17/9/4	23.9/25/21	17.33/20/16	24.7/25/23	12.47/20/8	24.87/25/23
O_{cons}	0	4.3004	1.2341	1.1250	0.8023	0.5350	3.0027	0.5074
$C_{ave}/C_{max}/C_{min}$	25/25/25	25/25/25	47.20/122/16		17.30/19/16		11.97/16/10	
C_{cons}	0	0	27.8201		0.5960		1.5862	

Table 4-9 Measurements for function D_2, N_s =*10*

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	100	0	100	0	86.67	3.33	100
$O_{ave}/O_{max}/O_{min}$	25/25/25	25/25/25	13.60/17/9	25/25/25	17.40/20/15	24.87/25/24	16.23/25/13	25/25/25
O_{cons}	0	0	1.7734	0	0.9685	0.3457	4.0231	0
$C_{ave}/C_{max}/C_{min}$	25/25/25	25/25/25	42.97/56/35		17.03/20/16		18.77/23/15	
C_{cons}	0	0	5.1960		0.8899		2.2542	

Table 4-10 Measurements for function D_2, N_s =*15*

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	100	0	100	0	96.67	0	90
$O_{ave}/O_{max}/O_{min}$	25/25/25	25/25/25	18.27/21/14	25/25/25	17.4/19/16	24.97/25/24	13.87/24/13	24.90/25/24
O_{cons}	0	0	2.0500	0	0.9322	0.1826	2.2397	0.3051
$C_{ave}/C_{max}/C_{min}$	25/25/25	25/25/25	57.30/64/50		17.53/21/16		21.90/25/17	
C_{cons}	0	0	3.7522		1.2794		1.8071	

Table 4-11 Measurements for function D_2, N_s =*20*

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc rate (%)	100	100	0	100	0	100	0	93.33
$O_{ave}/O_{max}/O_{min}$	25/25/25	25/25/25	19.47/22/16	25/25/25	17.73/20/16	25/25/25	14.50/23/13	24.93/25/24
O_{cons}	0	0	1.6554	0	1.0483	0	3.0822	0.2537
$C_{ave}/C_{max}/C_{min}$	25/25/25	25/25/25	69.87/76/63		17.33/19/16		22.87/25/20	
C_{cons}	0	0	3.1594		0.9223		1.3830	

Table 4-12 Average CPU running time for function D_2

POP SIZE	BMPGA	MPGA	DNC	MNGA	CLEARING
5	52.35	26.40	185.16	11.35	39.180
10	46.92	11.61	117.79	12.60	99.116
15	46.91	11.52	243.01	14.55	183.434
20	46.30	11.44	425.08	20.32	302.985

The performance of BMPGA, MPGA, DNC-R, MNGA-R and CLEARING-R are listed from best to worst as: BMPGA, MPGA, MNGA-R, CLEARING-R, DNC-R.

However, DNC exceeds both CLEARING and MNGA at N_s values of 15 and 20. At these values, the ordered list changes to: BMPGA, MPGA, DNC-R, MNGA-R, CLEARING-R.

The GAs ranked from fastest to slowest (Table 4-12) are: MPGA, MNGA, BMPGA, CLEARING, DNC.

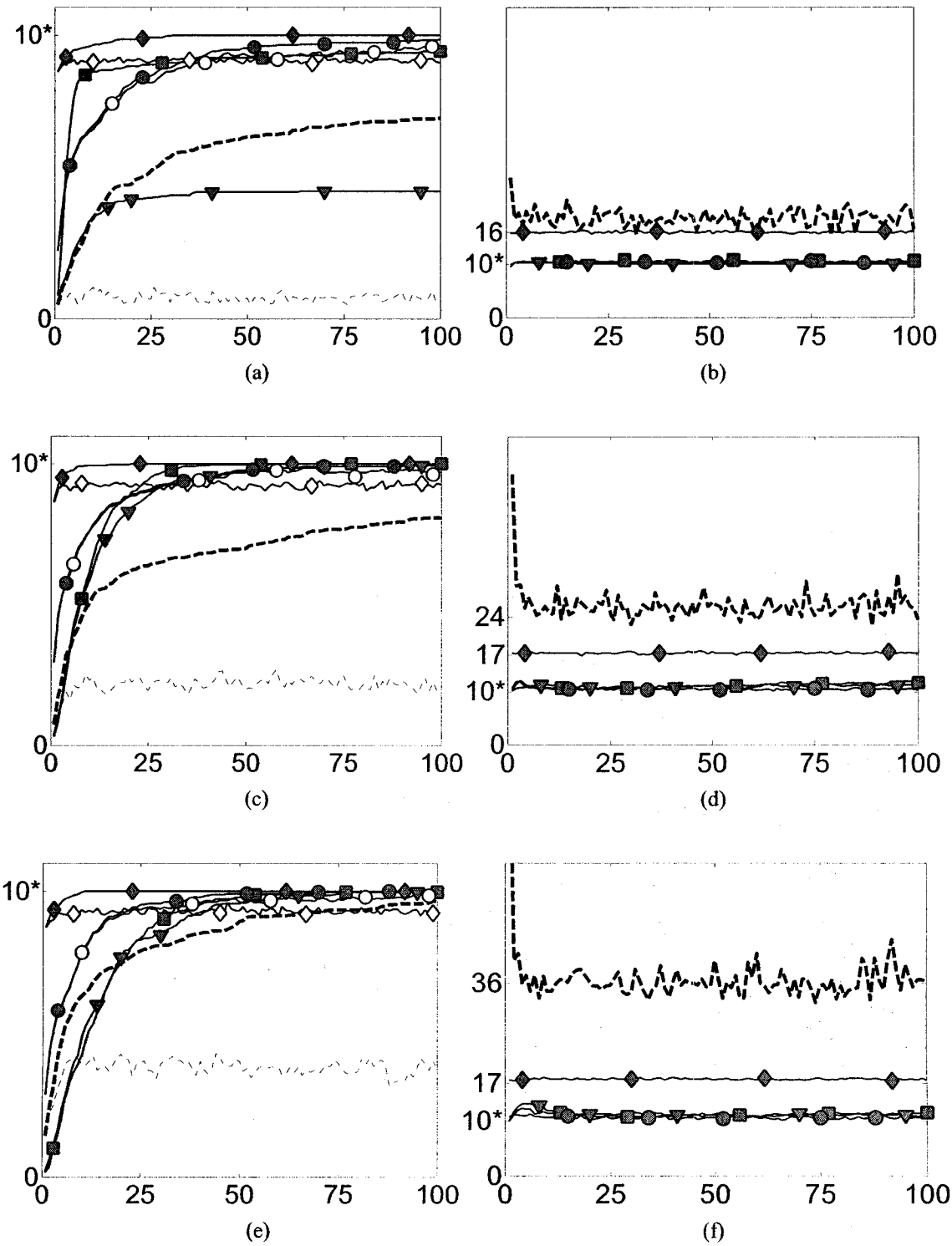
Function D_3

Function D_3 (Fig. A–3) is similar to D_1 in that the optima are of different heights and volumes and are non-equidistant to each other.

At $N_s = 5$, the success rate of BMPGA is 56.67%; it detects an average of 9.43 out of 10 peaks. The success rate of MPGA is 0 with an average of 4.49 out of 10 optima. BMPGA still outperforms MPGA in success rates, average number of optima found and consistency at most N_s values.

In CLEARING, σ_{min} (0.320) > σ_{dis} (0.224). This is in contrast to the situation of D_1 and D_2. In D_1 and D_2, pairs of adjacent optima may be put into one cluster. In D_3, however, even the closest optima should be safely separated because their distance exceeds the threshold (σ_{dis}). As a result, CLEARING-R is able to find more than 9 (of 10)

peaks at all N_s values. Although its success rates are still low (< 50%), its overall performance far exceeds that of D_1 and D_2.



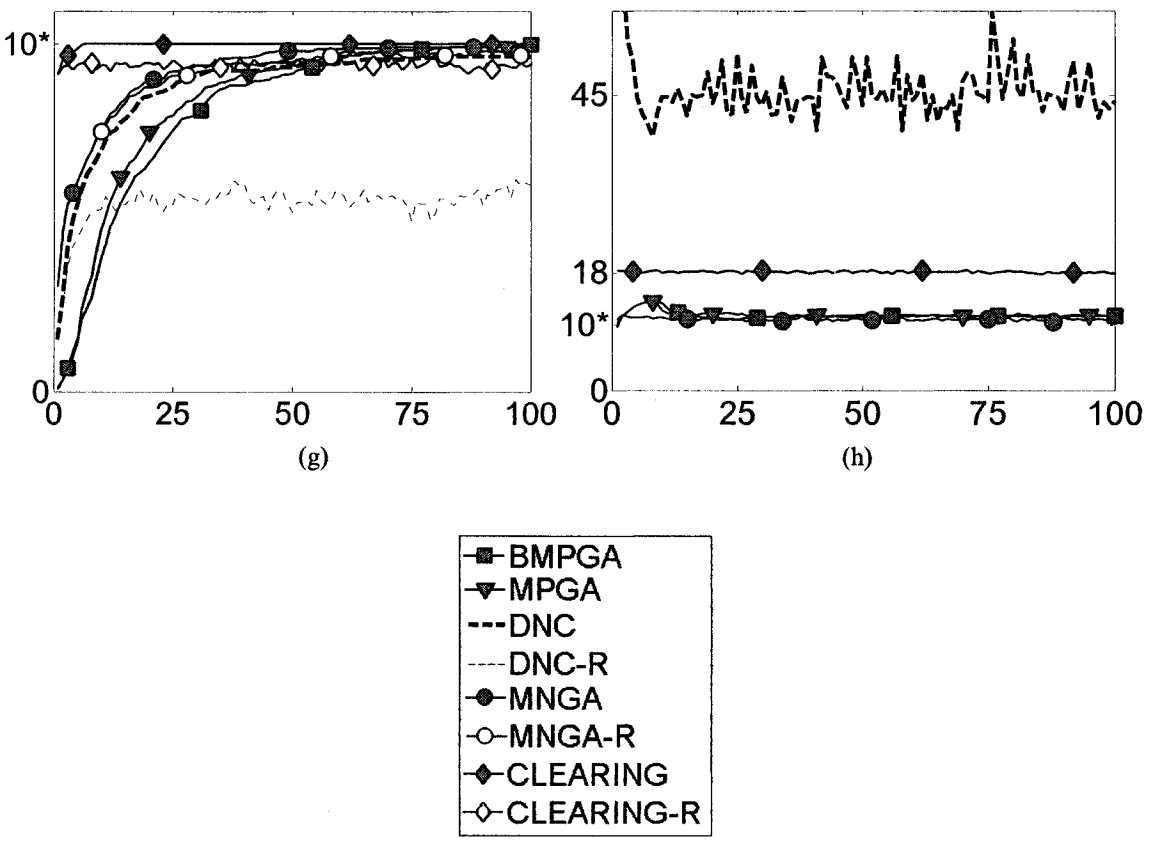


Fig. 4-10 Performance on Function D_3

Number of optima found against number of evaluations: (a) $N_s = 5$; (c) $N_s = 10$; (e) $N_s = 15$; (g) $N_s = 20$;
Number of clusters formed against number of evaluations: (b) $N_s = 5$; (d) $N_s = 10$; (f) $N_s = 15$; (h) $N_s = 20$.

The low success rates of CLEARING-R may be attributed to the irregularity of the landscape. From Fig. A–3 it can be seen that some optimum regions are large and malformed. Such regions may be partitioned into several clusters because of the small σ_{dis}. That means that one species may occupy more than one cluster. Since the overall amount of resources (the total population size) is fixed, resources available to other species are reduced. Hence, as shown in Table 4-13 - Table 4-16, although CLEARING forms more clusters than the desired number of species, the actual number of species it covers is less than that.

MNGA is able to generate the desired number of clusters. Both the success rate (63.33%) and the average number of optima found (9.57) by MNGA-R are the highest among all algorithms at N_s = 5. But MNGA is beaten by BMPGA and MPGA for all other N_s values.

Table 4-13 Measurements for function D_3, $N_s = 5$

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	56.67	0	0	0	63.33	86.67	3.33	100
$O_{ave}/O_{max}/O_{min}$	9.43/10/7	4.49/8/0	0.6/2/0	7.07/9/6	9.57/10/8	9.87/10/9	9.03/10/9	10/10/10
O_{cons}	0.7739	2.2928	0.6747	0.9803	0.6261	0.3457	0.1826	0
$C_{ave}/C_{max}/C_{min}$	10.53/13/10	9.97/11/9	16.47/28/9		10.33/12/10		16.07/18/14	
C_{cons}	0.7303	0.4901	4.4079		0.5467		0.9444	

Table 4-14 Measurements for function D_3, N_s =*10*

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	93.33	0	6.67	73.33	96.37	23.33	100
$O_{ave}/O_{max}/O_{min}$	10/10/10	9.93/10/9	2.10/4/0	8.07/10/6	9.70/10/8	9.97/10/9	9.23/10/9	10/10/10
O_{cons}	0	0.2537	1.0939	1.0483	0.5350	0.1826	0.4302	0
$C_{ave}/C_{max}/C_{min}$	11.80/15/10	11.03/13/10	23.63/38/14		10.23/12/9		17.33/19/16	
C_{cons}	1.3235	1.0334	6.3381		0.5683		0.8023	

Table 4-15 Measurements for function D_3, N_s =*15*

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	96.67	96.67	0	70	86.67	100	26.67	100
$O_{ave}/O_{max}/O_{min}$	9.97/10/9	9.97/10/9	3.77/5/2	9.67/10/8	9.87/10/9	10/10/10	9.27/10/9	10/10/10
O_{cons}	0.1826	0.1826	0.8976	0.5467	0.3457	0	0.4498	0
$C_{ave}/C_{max}/C_{min}$	11.63/15/10	11.07/12/10	36.37/68/25		10.73/12/10		17.77/20/17	
C_{cons}	1.2452	0.8683	11.1710		0.6915		0.7739	

Table 4-16 Measurements for function D_3, N_s =20

MEASUREMENTS	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	93.33	0	66.67	83.33	100	43.33	100
$O_{ave}/O_{max}/O_{min}$	10/10/10	9.93/10/9	5.97/8/4	9.67/10/9	9.80/10/8	10/10/10	9.43/10/9	10/10/10
O_{cons}	0	0.2537	1.0981	0.4795	0.4842	0	0.5040	0
$C_{ave}/C_{max}/C_{min}$	11.43/14/10	11.60/14/10	43.97/105/29		10.53/12/10		18.10/19/17	
C_{cons}	1.4065	1.0034	16.9309		0.7303		0.7120	

Table 4-17 Time Measurements for function D_3

POP SIZE	BMPGA	MPGA	DNC	MNGA	CLEARING
5	12.3	61.61	7.87	3.09	29.41
10	9.23	2.65	22.65	3.34	31.33
15	8.92	2.64	47.73	3.51	51.90
20	8.73	2.59	82.13	3.74	78.38

DNC's overall performance is the worst among all these algorithms. The success rates of DNC-R are 0 for all N_s values. It finds less than 60% of the actual optima.

The algorithms listed in order of their overall performance (best to worst) are: BMPGA, MPGA, MNGA-R, CLEARING-R, DNC-R.

The algorithms listed from fastest to slowest give us (Table 4-17): MPGA, MNGA, BMPGA, DNC, CLEARING.

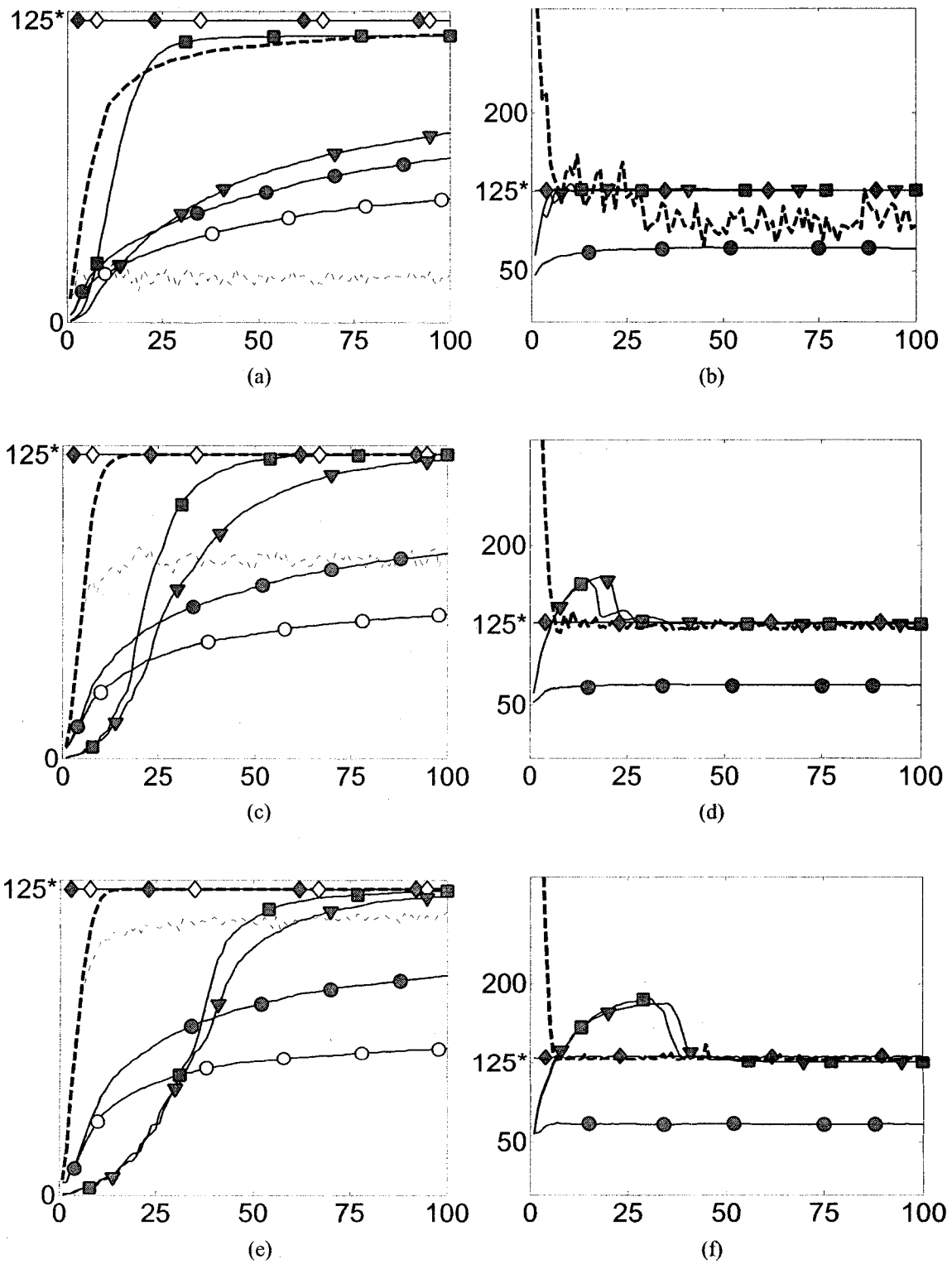
Function D_4

The optima of D_4 are equidistant to each other and have equal heights and volumes. The dimensionality of $D_4(n)$ is configurable.

Fig. A–4 shows a 2-dimensional D_4. But we actually used a 3-dimensional D_4 in order to demonstrate the extendibility of these algorithms to spaces of higher dimensions.

BMPGA fails to find all the optima at N_s = 5. It achieves 100% success rates at N_s

= 10. At N_s = 15 and N_s = 20, the success rates drop to 43.33% and 56.67%, respectively. Both cases find approximately 124 out of the 125 peaks.



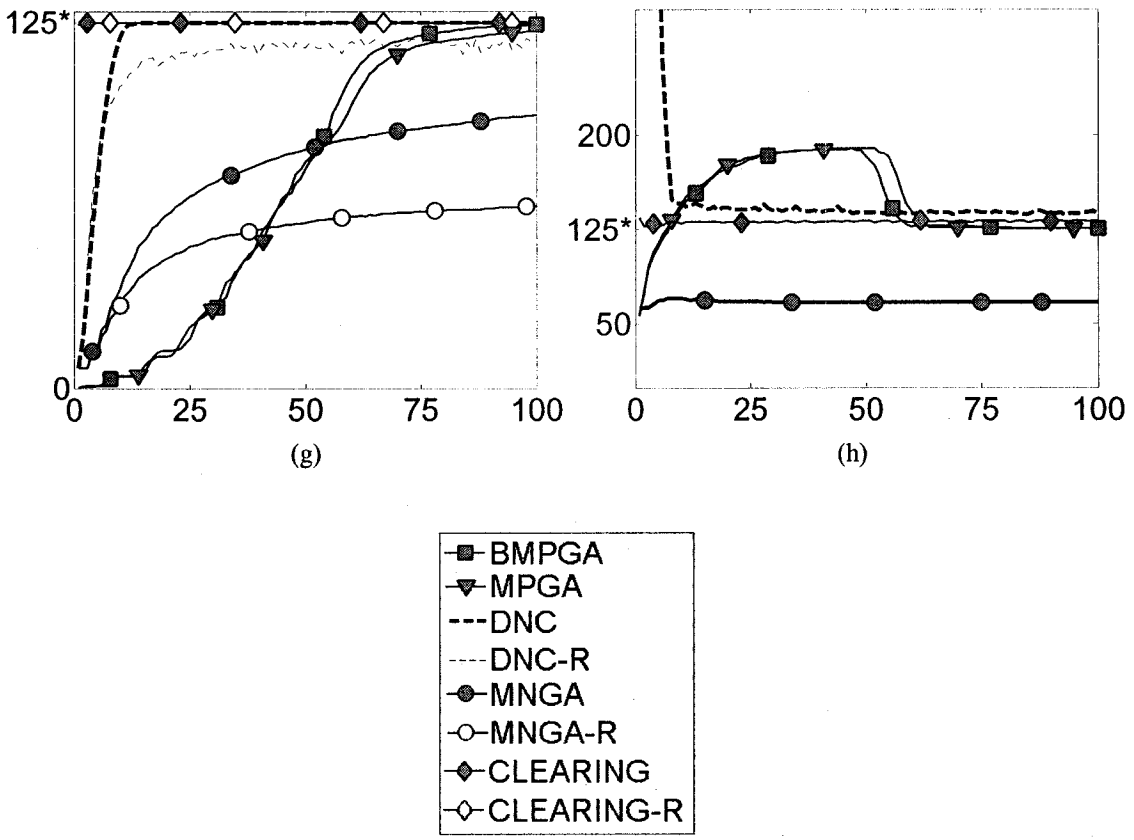


Fig. 4-11 Performance on Function D_4

Number of optima found against number of evaluations: (a) $N_s = 5$; (c) $N_s = 10$; (e) $N_s = 15$; (g) $N_s = 20$;

Number of clusters formed against number of evaluations: (b) $N_s = 5$; (d) $N_s = 10$; (f) $N_s = 15$; (h) $N_s = 20$.

Performance of MPGA is worse than that of BMPGA. All of its success rates are less than 10%. At $N_s = 5$, it only finds 78.3 out of 125 peaks. It finds up to 122 peaks for $N_s > 5$.

Table 4-18 Measurements for function D_4, $N_s = 5$

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	0	0	0	30	0	0	100	100
O_{ave} / O_{max} / O_{min}	118.53/124/114	78.30/90/62	20.10/40/4	118.97/125/100	50.93/62/46	68.33/81/60	125/125/125	125/125/125
O_{cons}	2.7258	5.7063	7.7776	7.1558	3.3930	4.7149	0	0
C_{ave} / C_{max} / C_{min}	125.06/126/125	125.43/127/125	91.33/302/29		69.83/76/65		125.20/126/125	
C_{cons}	0.2537	0.6261	78.5504		2.5875		0.4068	

Table 4-19 Measurements for function D_4, N_s =10

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	100	3.33	0	100	0	0	100	100
$O_{ave}/O_{max}/O_{min}$	125/125/125	123/125/118	83.13/106/42	125/125/125	58.83/64/53	84.13/95/76	125/125/125	125/125/125
O_{cons}	0	1.2865	15.1856	0	3.0181	4.6737	0	0
$C_{ave}/C_{max}/C_{min}$	125.17/126/125	125.07/126/125	125.90/252/112		67.77/73/65		126.84/133/125	
C_{cons}	0.3790	0.2537	24.2208		1.9061		2.9676	

Table 4-20 Measurements for function D_4, N_s =15

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	43.33	3.33	0	100	0	0	100	100
$O_{ave}/O_{max}/O_{min}$	124.37/125/123	122.07/125/118	115.76/123/107	125/125/125	60.63/63/58	90/99/79	125/125/125	125/125/125
O_{cons}	0.6149	1.6174	3.9007	0	1.3257	4.3786	0	0
$C_{ave}/C_{max}/C_{min}$	125.23/128/125	125.13/126/125	128.78/137/124		66.90/71/64		131.87/134/127	
C_{cons}	0.6261	0.3457	3.2994		1.9001		1.8889	

Table 4-21 Measurements for function D_4, N_s =20

Measurements	BMPGA	MPGA	DNC		MNGA		CLEARING	
			Real	Accu	Real	Accu	Real	Accu
Suc (%)	56.67	6.67	0	100	0	0	100	100
$O_{ave}/O_{max}/O_{min}$	124.33/125/122	122.17/125/118	120.19/123/117	125/125/125	62/66/57	93.23/99/83	125/125/125	125/125/125
O_{cons}	0.9589	1.8020	1.8787	0	2.0844	4.6512	0	0
$C_{ave}/C_{max}/C_{min}$	125.13/127/125	125.07/126/125	136.93/150/131		66.57/70/64		131.65/134/127	
C_{cons}	0.4342	0.2537	3.8431		1.5687		2.1992	

Table 4-22 Time Measurements for function D_4

Pop size	BMPGA	MPGA	DNC	MNGA	CLEARING
5	15.11	9.17	3073.167	8.63	2430.53
10	13.30	6.73	6504.74	16.62	3408.87
15	10.03	6.49	7874.34	33.91	4708.46
20	9.90	6.40	8792.01	57.91	5750.22

CLEARING gives excellent results. It has 100% success rates for all N_s values. This is because its σ_{min} (0.2) is greater than σ_{dis} (0.173). All the optima have the same

volume and are evenly distributed, and as such are perfectly separable. From Table 4-18 - Table 4-21, we can see that CLEARING forms approximately the desired number of clusters at N_s = 5 and N_s = 10. Although the number of clusters for N_s > 10 slightly exceeds the desired number, the performance of CLEARING does not degrade. It appears that the large size of the population ensures adequate exploration of all species.

It can also be seen from Fig. 4-11 (a), (c), (e) and (g) that CLEARING finds all optima from as early as the first sampling point. That is possibly because the clustering approach in CLEARING does not evaluate fitness. Hence all fitness evaluations contribute to evolution and all species are evolved for a large number of generations before the first sampling point.

The topology of D_4 is similar to that of D_2. On such symmetric landscapes, MNGA forms less than 70 clusters, which is much less than the desired number - 125. MNGA consistently fails to find all the optima. It is better than DNS at N_s = 5 but remains the worst performer for all other N_s values.

At N_s = 5, σ_{init} of DNC approximately equals σ_{min} (equation (3.2)). After merging and enlarging clusters, the number of clusters falls below the desired number (125). However, starting from N_s = 10, the number of clusters grows closer to the desired number. As a result, the performance of DNC is enhanced considerably with increasing N_s. At N_s = 20, the results of DNC and those of DNC-R are very close. DNC gives its best performance on function D_4. It is expected to perform better at larger population sizes.

The GAs are listed in decreasing order of their overall performance as: CLEARING-R, BMPGA, MPGA, DNC-R, MNGA-R.

The algorithms listed from fastest to slowest are (Table 4-22): MPGA, BMPGA, MNGA, CLEARING, DNC.

4.2.6 Analysis

From the experiments we can see that although DNC, MNGA, and CLEARING have the potential to find all optima, none of them is able to consistently retain those found optima (except for CLEARING on function D_4). Their real time results are worse and have more fluctuations than their historically accumulated results.

Conversely, BMPGA and MPGA are able to preserve species consistently. Their real time results match their accumulated results almost perfectly. The curves of BMPGA and MPGA are generally smoother than those of DNC, MNGA and CLEARING.

Each algorithm is discussed in detail below.

4.2.6.1 DNC

DNC implements partial local elitism by only preserving elites in large clusters with sizes exceeding 5% of the size of the overall population. However, DNC usually forms far more than the desired number of clusters and the majority of those clusters are small. As a result, only a small proportion of clusters, and hence species, are maintained.

In those small clusters, their members may or may not be sufficiently promoted to ensure survival in the population. These species may be subject to extinction. Even if one of the species reaches its optimum, the discovered optimum may not be retained.

Therefore, the elitism scheme, the clustering approach, and the sharing mechanism, result in noticeable fluctuation of the DNC-R curves.

The sharing technique itself suffers from other problems. In DNC (and all Sharing

GAs), fitnesses values are shared within the same cluster. Hence individuals are dispersed within their clusters rather than focused on the optima. Sharing GAs force overall diversity in the population, but lack local convergence [58]. They also implement global evolution. This may improve the population as a whole, but each species may not be fully evolved.

For the reasons given above, the overall performance of DNC is far from satisfactory. It completely fails to find and preserve all optima of the tested benchmark functions.

The performance of DNC is dependent on its initial radius σ_{init}. If σ_{init} is greater than or equal to the distance between actual optima, DNC will generate less than the desired number of clusters because the clusters will be merged and enlarged, as is the case with function D_4 at $N_s = 5$. However, if σ_{init} is slightly smaller than the distance between actual optima, DNC will form the correct number of clusters quickly, as in the cases with function D_4 at $N_s \geq 10$. Of course, if σ_{init} is too small, the number of clusters will first drop quickly and then stagnate or decrease very slowly. The number of clusters remains far above the desired number. This is the situation that is most frequently observed in the experiments, because the overall population size is usually much larger than the number of optima and σ_{init} is very small (equation (3.2)).

The performance of DNC also improves rapidly with population size. This result matches the claim that Sharing GAs generally demand a sufficiently large population to work properly [45], [47]. However, quickly increasing the population size raises the computational cost dramatically, as can be seen from Table 4-7, Table 4-12, Table 4-17 and Table 4-22.

In summary:

- DNC is unable to preserve species for the following reasons:
  - The elitism scheme is unable to preserve small clusters;
  - The sharing method does not guarantee preservation of promising clusters;
  - The clustering method is unable to form stable clusters around potential optima;
- DNC does not fully explore all promising areas due to its sharing and global evolution mechanisms;
- The performance of DNC is dependent on its initial radius σ_{init}. It works properly if and only if it starts from an appropriate σ_{init};
- The performance of DNC also depends on the population size. Its performance is improved with increasing population size, which in turn increases computational cost.

4.2.6.2 MNGA

MNGA uses a Hill-Valley function for clustering. This method is generally better than distance-based methods since it is more tolerable to uneven landscapes. Nevertheless, the symptoms demonstrated in Fig. 3-4 and discussed in Section 3.1.2 above are frequently observed. This results in unstable clusters with individuals from different species. Consequently, the previously found optimum may be replaced by fitter individuals from different species, and may appear again in other clusters later. Therefore, the number of optima present in a population may fluctuate significantly.

Performance is expected to worsen if the government size increases, because the center, which is the average of government members, is more likely to misrepresent its

cluster when more individuals are involved.

MNGA works better on uneven landscapes (functions D_1 and D_3) than even landscapes (functions D_2 and D_4). This is because the Hill-Valley function with the symmetric sampling vector [0.25, 0.5, 0.75] generates a desired number of clusters on uneven landscapes, but misses clusters on even landscapes. The situation may be improved if an asymmetric or randomly generated sampling vector is used.

Although MNGA outperforms DNC and CLEARING for functions D_1 and D_3, it is still inferior to BMPGA and MPGA. A possible reason is that MNGA runs an overall population of fixed size, within which the size and number of clusters are flexible. This does not guarantee an even distribution of individuals among species. As a result, small species may not be fully evolved and could not reach the goal at the end of a run. Conversely, BMPGA and MPGA have a varied number of subpopulations, the sizes of which are constant. As a result, all subpopulations (species) are assigned equal resources and are evolved in the same manner.

MNGA is characterized below:

- MNGA is unable to preserve species because of its clustering method, which produces unstable clusters;
- The sampling vector plays an important role in MNGA. A sampling vector that works appropriately on a certain type of landscape may not suit other types of landscapes. Hence different types of landscapes may require different sampling vectors;
- MNGA runs a population of fixed size, which may affect its performance negatively even when it is able to form the desired number of stable clusters.

4.2.6.3 CLEARING

CLEARING only preserves winners of clusters with fitness values greater than the average fitness of the population. Promising individuals with relatively low fitness values are unfortunately discarded.

Mating is carried out between individuals from different clusters. The offspring may belong to a cluster other than their parents'. With such elitism and mating schemes, frequent loss of species is expected.

The performance of CLEARING is dependent on its global threshold σ_{dis} and the landscape of the function to be optimized. For functions D_1 and D_2, σ_{dis} is greater than σ_{min} (the minimum distance between adjacent peaks). Different species will be put into the same cluster if the distance between them is less than σ_{dis}. Weaker species are then cleared out. The number of species explored is less than the actual number of optima. Hence, CLEARING fails to find all optima when it is applied to D_1 and D_2.

For functions D_3 and D_4, however, σ_{dis} is less than σ_{min}. The likelihood of different species clustering together is small. However, if σ_{dis} is too small, a large optimum region will be occupied by more than one cluster. Resources assigned to other optimum regions are hence limited. This is the case with function D_3. Although CLEARING performs better on D_3 than on functions D_1 and D_2 (in terms of success rates and average number of optima found), its success rates on D_3 still fall below 50%.

CLEARING works best for function D_4. The optima of D_4 are evenly distributed. The volumes of the peaks are equal and are not as large as those of function D_3. Hence, each species can be safely isolated within its own cluster.

In CLEARING, clustering only computes distances between given points without

any fitness evaluations. That means that in CLEARING, all fitness evaluations contribute to evolution. However, in other algorithms, a large proportion of fitness evaluations are performed during clustering. Since the total number of fitness evaluations for all algorithms is the same, the species in CLEARING are more likely to be fully evolved. Hence, in function D_4, the performance of CLEARING exceeds that of all other algorithms when they are compared against fitness evaluations.

The characteristics of CLEARING are summarized below:

- CLEARING is unable to preserve species because:
  - It fails to preserve the elites of species with relatively low fitness values;
  - Its global mating strategy does not guarantee the propagation of existing species;
- The performance of CLEARING is dependent on its global threshold σ_{dis} - a σ_{dis} less than or equal to σ_{min} is generally required;
- Even with $\sigma_{dis} \leq \sigma_{min}$, CLEARING does not necessarily work well on irregular landscapes. It works best on landscapes with evenly distributed optima of the same volume.

4.2.6.4 BMPGA and MPGA

BMPGA and MPGA implement full local elitism in the sense that at least one of the fittest individuals is maintained in each subpopulation, regardless of the size of the subpopulation. Once a species is found, it is never subject to extinction.

With Recursive Middling, BMPGA and MPGA always form the required number of subpopulations. The possibility of a single subpopulation containing more than one species is very small. The center of a subpopulation is its up-do-date fittest individual and

is not influenced by other individuals as in DNC and MNGA. Hence, subpopulations in BMPGA and MPGA are stable. They are able to maintain stable species and preserve all found optima.

Initially, BMPGA and MPGA always find a smaller number of optima than other algorithms. This is because they start with a single subpopulation of size N_s, whereas DNC, MNGA and CLEARING start with a population of size N_sP (P is the number of optima). The initial differences are greater when N_s increases.

The performance of BMPGA may be affected by N_s. For example, for functions D_3 and D_4, the results at $N_s = 5$ are inferior to the results at $N_s \geq 10$. That is because the subpopulations are too small to sufficiently explore promising areas.

For function D_4, on the other hand, the best results are obtained at $N_s = 10$. Performance worsens and equilibrium point delays at larger N_s values. The average number of fitness evaluations at each generation increases rapidly with N_s because Recursive Middling involves recursive evaluation. Given that the maximum number of fitness evaluations is fixed, the number of generations for evolution drops with increasing N_s. Hence at large N_s values, the population is not evolved as thoroughly as with small N_s values. This phenomenon manifests itself when P is large, as in function D_4.

Nevertheless, BMPGA successfully finds all optima for all functions at $N_s = 10$. This fact indicates that there may exist an appropriate value of N_s that can be applied to multimodal landscapes of variable dimensionality and number of optima.

The influence of N_s on MPGA is greater than its influence on BMPGA. At $N_s = 5$, MPGA shows inferior performance and its execution time is many times greater than that of BMPGA. The performance gap between MPGA and BMPGA narrows down as the

value of N_s increases. This is because MPGA generates less diversity than BMPGA. At low N_s values, the subpopulations of MPGA are prone to stagnation about sub-optima. At large N_s values, however, those subpopulations may contain sufficient diversity. The larger N_s the more diversity the subpopulations of MPGA have. Therefore, the performance of MPGA becomes closer to that of BMPGA when N_s increases.

Characteristics of BMPGA and MPGA are:

- Both are able to preserve species because:
  - Their clustering approach forms the desired number of stable subpopulations;
  - Their elitism strategy preserves species regardless of their sizes;
- The performance of BMPGA and MPGA may be affected by population size;
- BMPGA is generally a better performer and is less dependent on population size than MPGA.

4.2.6.5 Comparison

In most optimization problems, the approximate location and the value of optima are unknown. Hence optima are rarely extracted during a run, but rather at the end of the run. No matter how many optima are found historically, only optima existing at the point of termination are counted. Therefore, in this dissertation, we mainly observe and compare the real time results of various algorithms. We believe that these results demonstrate an algorithm's capability to both *find* and *preserve* species.

Although DNC, MNGA and CLEARING can more or less promote some diversity within a population, they are unable to *preserve* it. Even if some of the species happen to find the optima, they are still subject to the risk of extinction. For DNC,

MNGA and CLEARING, there is usually a large discrepancy between the "real-time" results and the "historically accumulated" results. Most of the real-time results are not satisfactory.

DNC finds less than 77% optima for all the tested functions. CLEARING performs as badly as DNC on functions D_1 and D_2. For function D_3, it finds up to 94% of the optima, but its success rate still falls below 44%. MNGA completely fails on function D_2 and D_4. It does however achieve reasonable success rates for functions D_1 (up to 20%) and D_3 (up to 87%).

In contrast to DNC, MNGA and CLEARING, BMPGA and MPGA are able to *find* and *preserve* species well. BMPGA successfully finds all optima for functions D_1 and D_2. For function D_3, its success rates range from 57% at N_s = 5 to 100% at N_s = 20. For all the tested functions, BMPGA finds at least 94% of the optima.

MPGA has lower success rates and average number of optima found than BMPGA at N_s = 5. It successfully finds all the optima at all other N_s values when applied to functions D_1 and D_2. For function D_3, its success rates range from 0 to 97%. For all the functions, it finds a minimum of 0 and a maximum of 99% of the optima.

Obviously, both BMPGA and MPGA outperform DNC, MNGA and CLEARING in terms of success rates and average number of optima found for functions D_1 to D_3. From Fig. 4-8 - Fig. 4-10 as well as Table 4-3 -Table 4-17, we can also see that the results of BMPGA and MPGA are more consistent than those of DNC, CLEARING and MNGA.

The results for function D_4 are slightly different. CLEARING achieves the best results with 100% success rates for all N_s values. BMPGA is worse than CLEARING

since its success rate is 100% only at N_s = 5. MPGA is worse than BMPGA with lower success rates (6.7%) and a smaller number of optima found (122). Nevertheless, BMPGA and MPGA are still superior to DNC and MNGA, whose success rates are all 0.

DNC, MNGA and CLEARING only work appropriately under certain conditions or on specific landscapes. For example, MNGA suits irregular landscapes. On these landscapes (e.g. D_1, D_3), MNGA achieves acceptable results (>83% peaks found) and outperforms both DNC and CLEARING. DNC demands an appropriate initial radius and a sufficiently large population size. This is exactly the case with function D_4, on which DNC returns its best performance (up to 96% peaks found). CLEARING works best with an appropriate global threshold σ_{dis} and on landscapes with an even distribution of optima, such as function D_4.

BMPGA and MPGA are more robust than DNC, MNGA and CLEARING because they have satisfactory results (>99% peaks found with BMPGA, >97% peaks found with MPGA) for all the functions, provided the size of the subpopulations is large enough ($N_s \geq 10$). In fact, with an appropriate N_s value (N_s = 10), BMPGA successfully finds all the optima of all the functions tested. Wider applicability is a clear advantage that BMPGA and MPGA have over DNC, MNGA and CLEARING.

A potential concern with BMPGA and MPGA is their computational cost, since Recursive Middling involves recursion. However, as can be seen from Table 4-7, Table 4-12, Table 4-17 and Table 4-22, the actual cost is much less than expected. The running time of DNC and CLEARING is orders of magnitude larger than that of BMPGA and MPGA. That is because clustering in DNC and in CLEARING does not consume any fitness evaluations, whereas clustering in BMPGA and MPGA is responsible for a large

proportion of fitness evaluations. Since all the algorithms have the same overall population size and the same parameter values, we can assume that all the algorithms use approximately the same number of fitness evaluations for evolution. At each generation, the average number of fitness evaluations of BMPGA and MPGA far exceeds that of DNC and CLEARING. Hence given the same number of maximum fitness evaluations, BMPGA and MPGA are expected to run much fewer generations than DNC and CLEARING. Although clustering in DNC and CLEARING does not involve fitness evaluations, it still has other non-negligible costs. For example, in DNC, the cost of sorting and manipulating a large list of niche pairs is $O(N_n^2)$, where N_n is the number of the niches; which is much larger than P, the actual number of optima. This cost, which may even exceed that of evolution, grows greatly when the population size increases.

Similarly, MNGA is generally slower than MPGA (except for $N_s = 5$) because its HV function requires less fitness evaluations than RM in MPGA and its generations are more than that of MPGA (but less than that of DNC and CLEARING). The average running time of MNGA increases with population size.

For each function, the execution time of BMPGA and MPGA is the longest at $N_s = 5$. It then drops to a fraction at $N_s = 10$. From then on, the running time either remains approximately constant or decreases slowly when N_s increases. BMPGA is generally slower than MPGA because BMPGA evaluates more fitness values.

Based on the facts presented above, we conclude that the computational cost of a GA should not be always attributed solely to fitness evaluation since many operations, such as crossover, mutation and clustering, do not necessarily evaluate fitness but still require computation. These costs, in some circumstances, should also be taken into

account.

Hence, when the cost of fitness evaluation is not very high, the overall cost of a GA involving a large amount of fitness evaluations may not be higher than the costs of a GA with other costly operations. That is exactly the case of BMPGA and MPGA versus DNC and CLEARING. The overhead induced by RM is thus practically acceptable.

It is also noted that DNC, CLEARING and MNGA run more generations than BMPGA and MPGA, but end up with worse results, most of the time. If all of these algorithms are run with the same number of generations, the performance gap between DNC, CLEARING as well as MNGA and BMPGA as well as MPGA is expected to be greater.

4.3 Algorithm Discussion

4.3.1 Fitness: Bi-objective versus Single Objective

The bi-objective scheme is a novel aspect of BMPGA in that all other multimodal GAs adopt one fitness term.

In the optimization of multimodal functions, the second objective is complementary to the original objective. It is similar to add small perturbation to the problem so that more diversity of the population is expected. Accordingly, we design a Multi-population GA (MPGA). It is identical to BMPGA except that it only uses one fitness term (the original objective) as its fitness function.

From the experiments in this chapter, it can be seen that BMPAG outperforms MPGA in overall success rates and average number of optima found. It is also less dependent on varying population size than MPGA. This fact empirically demonstrates the success of the bi-objective scheme.

Although so far in the literature, it is not usual to see the usage of multi-objective GAs for multi-modal optimization problems, a few researchers have shed light on this interesting subject. Watanabe and Sakakibara [70] proposed to transform a single objective optimization problem into a multi-objective one by adding noise as another objective. Their experiments demonstrate that this approach effectively increases the paths to the global optimum and the diversity of the population. As a result, it achieved better solutions than the original single objective problem when the function to be optimized is multimodal and separable. This is a good proof of the effectiveness of the bi-objective scheme.

4.3.2 Evolution: Local versus Global

As a typical population based GA, BMPGA implements all these operations locally within each subpopulation. Nevertheless, most evolutionary operations in a typical genetic operator based GA are performed within the entire population. Local evolution of separate subpopulations and global evolution of the whole population is the core difference between a population based GA and a genetic operator based GA.

Basic elements of an evolutionary process include selection, recombination (mating restriction), and mutation. In the literature, some genetic operator based GAs have adopted local selection and have demonstrated advantages. For example, DNC [2] used partial local elitism to preserve diversity of the population. Top 10% individuals in a niche that is greater than 5% of the overall population size are retained in the next generation. CLEARING [45] also adopted partial local elitism by preserving species winners with fitness greater than the average fitness of the overall population.

However, as already discussed in this chapter, these elitism schemes cannot fully

preserve species. They are subject to unnecessary loss of species such as loss of small niches in DNC or species whose winners bear low fitness values in CLEARING. Therefore, we believe that a strong local elitism method, as used in BMPGA, is required. That is, the best individual of any subpopulation, despite its size, is passed unaltered to the next generation. This is of great significance to the preservation of species in a multimodal space.

Besides partial local selection, mating restriction is used in some genetic operator based GAs [48], [59], [61]. Zhang *et al.* [59] selected the first parent using binary tournament selection in the whole population. If this parent is a niche member, another parent is the corresponding niche seed in the same niche. Otherwise another parent is selected using the binary tournament selection again in the whole population.

Song and Yu [61] adopt both inner-species and inter-species crossover in their Sharing based approach. But the rate of the inter-species crossover is much lower than the rate of the inner-species crossover in order to prevent the destructive effects of the former to the niches already formed.

Although the genetic operator based GAs above use some local operations, they are, in essence, still global methods that focus on evolving the whole population. Our experiments have shown that these GAs are generally inferior to population based GAs. That is because in the former, individuals are spread around the top of peaks, rather than clustered tightly at the apex. On the contrary, subpopulations in the latter are relatively independent and evolved separately. The selection pressure within each subpopulation is focused locally about the peaks.

That does not mean local evolution is perfect. For example, it is generally

believed that disallowing recombination between individuals from different species may avoid destroying potentially good building blocks [48]. However, exchange of genetic information between individuals from different species can help to promote the diversity of the population, especially when stable species have been formed and inner-species mating can only generate offspring within the same species. Hence, some population based GAs [11, 12, 64] use cross-species mating in their framework.

4.3.3 Sizing: Fixed versus Variable Sized Population

Some methodologies [1, 2, 21, 28, 63] manipulate an overall population of fixed size. The population contains a variable number of clusters, the size of which is also variable. In contrast, BMPGA and another school of approaches [11, 12, 23, 25, 29] fix the size of each cluster. The number of clusters and the size of the overall population are dynamic.

For those algorithms that adopt a fixed sized population, their performance is tightly tied to the population size [42, 44], which hence becomes an important parameter to be carefully tuned. However, without *a priori* knowledge of the landscape, it is hard to estimate an appropriate size of the population.

If the population size is too small, it may result in insufficient coverage and exploration of the landscape. Therefore, a large population is generally required in order to maintain the diversity. Nevertheless, a sufficiently large population does not guarantee an even distribution of individuals around potential optima since resources (chromosomes) are usually randomly allocated among species. A possible consequence is imbalanced exploration of the search space. Crowded areas are exploited until maturity, whereas areas with sparse neighborhood are not fully explored due to limited resources. Moreover, if the population is too large, it may also incur a lot of extra cost.

If, instead, the size of each cluster is fixed, each optimum is assigned the same amount of resources (chromosomes). The size of the entire population is dependent on the landscape. Each promising area is guaranteed to be explored with the same amount of resources. This scheme potentially eliminates the need to tune the overall population size.

The experiments empirically prove the advantages of fixed sized subpopulations over fixed sized population. BMPGA, MPGA and MNGA are all population based GAs. BMPGA and MPGA run a number of fixed sized subpopulations, whereas MNGA runs an overall population of fixed size. In BMPGA and MPGA, the overall population size starts from N_s and gradually reaches the equilibrium of $N_s \cdot k$, where N_s is the preset size of each subpopulation, and k is the number of optima. In MNGA, on the other hand, the overall population size is $N_s \cdot k$ from the beginning. It can be seen that MNGA is generally worse than BMPGA and MPGA for all tested multimodal functions. This is the case even when MNGA is able to form the desired number of stable clusters. That is because, as discussed above, the sizing scheme of MNGA does not guarantee an even distribution of individuals among species. As a result, small species may not be fully evolved and could not reach their optimum at the end of a run.

4.3.4 To Share or Not to Share

Sharing GA [20] is one of the most popular multimodal GAs up-do-date. Since its invention, to share or not to share remains a question of debate.

Standard Sharing suffers from a high complexity of $O(N^2)$, incurred by the computation of the sharing function. Furthermore, in order to estimate the global radius, standard Sharing requires a given number of expected optima, which is usually

unavailable.

As discussed in Section 1.1.1.1.9, in the literature, there have been large quantities of studies improving the drawbacks of the standard Sharing [2, 24, 28, 30, 50, 59]. Some of them effectively reduce the computational cost [2, 24, 28, 30, 59]. Others adopt various clustering approaches to eliminate the need of *a priori* information about the number of optima [2, 28, 50].

Nevertheless, the core of all Sharing based GAs – the fitness sharing mechanism itself is problematic. As seen from equation (2.7) in Section 1.1.1.1.9, the fitness of an individual is shared with its neighbors in the same niche. It is assumed that weak individuals will have a sparsely populated neighborhood, whereas highly fit individuals will be seen within dense areas. Consequently, the fitness of the latter will be degraded to approximately the same level as the fitness of the former. However, this assumption is not always true (example can be seen in Fig. 5-6 and Section 5.1.2). Even if it does hold, a weak individual may or may not be sufficiently enhanced for survival, since the shared fitness of this individual is proportionate to its raw fitness. Less fit but promising individuals in unvisited areas are at high risk of extinction, resulting in a general loss of diversity. On the other hand, since fitness sharing arbitrarily alters the fitness landscape, loss of potential optima or introduction of phantom optima may be caused.

Most Sharing GAs carry out global evolution. And selection pressure is applied globally. Although some more recent versions of Sharing GAs adopt local evolutionary operators, such as local elitism [2, 28] and restricted mating [48, 59, 61], most evolutionary operators act globally. As discussed in Section 4.3.2, global evolution of the entire population is inferior to local evolution of each promising area.

Another problem of the Sharing GA is its population sizing. All Sharing GAs use a population of fixed size. The number and the size of the niches within the population are variable. As already discussed in Section 4.3.3, this scheme is inappropriate for search in multimodal space. Researchers have found that the Sharing GA generally needs a large population to work [42, 45, 56, 71].

In this chapter, a sharing based GA – Dynamic Niche Clustering [2, 28] is compared to our BMPGA. With the same number of fitness evaluations, BMPGA outperforms DNC considerably on all benchmark functions tested in terms of a comprehensive coverage of optima in multi-modal landscapes as well as speed of operation.

In summary, the sharing mechanism, the global evolution and the population sizing constitute pitfalls of the Sharing GAs. We believe that, unless specific conditions are satisfied, the Sharing GA is unsuitable for multimodal optimization problems due to its inherent limitations.

4.4 Conclusions

This chapter applies BMPGA to the application of multimodal function optimization. As its name suggests, BMPGA uses a bi-objective scheme - the subpopulations are evolved toward two objectives. Individuals that are weak in one of the fitness terms but promising in the other are given a chance at survival. The overall diversity of the population is hence effectively enhanced.

BMPGA adopts a new clustering method, called Recursive Middling. This method helps BMPGA form the desired number of clusters (subpopulations) around potential optima and maintain stable species throughout the optimization process.

BMPGA carries out local evolution of subpopulations. Each subpopulation is evolved independently toward its optimum. The selection pressure is focused within the subpopulations rather than dispersed in the whole population. Full local elitism of BMPGA prevents extinction of small subpopulations.

In BMPGA, the size of each subpopulation is fixed. Hence each promising area is explored using the same amount of computational resources.

BMPGA is compared to other multimodal GAs with distinguished features that make them as interesting competitors: MNGA [1], DNC [2], CLEARING [3] and MPGA. DNC and CLEARING are genetic operator based GAs. MNGA is a population based GA. MPGA is a variant of BMPGA, which uses one fitness objective as opposed to BMPGA's two.

These algorithms were tested on four benchmark multimodal functions of different dimensionality and landscapes. For each function, each algorithm was run 30 times. The results were then averaged to produce the final results.

With an acceptable computational overhead (induced by Recursive Middling), BMPGA and MPGA outperform DNC, MNGA and CLEARING in relation to:

- Effectiveness: BMPGA and MPGA not only find but also preserve species consistently. DNC, MNGA and CLEARING are unable to fully preserve species. BMPGA and MPGA achieve higher success rates and find more optima than DNC, MNGA and CLEARING.
- Generality and dependence on parameters: DNC, MNGA and CLEARING can only function effectively under specific parameter conditions and/or on certain

landscapes. BMPGA and MPGA are much less dependent on particular parameter settings and are able to work on different types of landscapes.

- Consistency: The results of BMPGA and MPGA are more consistent than those of DNC, MNGA and CLEARING.

The computational costs of BMPGA and MPGA are not necessarily higher than those of DNC, MNGA and CLEARING.

BMPGA further outperforms MPGA in overall success rates and average number of optima found. BMPGA is considerably less sensitive to population size than MPGA.

As such, we conclude that BMGA is arguably a better choice for those who aim to use GAs for multi-modal function optimization than DNC, MNGA, CLEARING and MPGA.

Finally, it is worth noting that we have successfully applied BMPGA to the detection of elliptical curves [6] and the segmentation of microscopic cells [56], which will be introduced in detail in Chapter 5 and Chapter 6, respectively.

Chapter 5 Application I: Multiple Ellipses Detection

A shape is a set of geometrically distributed visual points, together with their mutual special relationship. The shape is one of the most important visual features in machine vision. It denotes one of the essential and intrinsic characteristics of an imagery object and remains perceptually similar and recognizable under affine transform such as translation, rotation, and scaling. A user survey [72] on cognitive aspects of image retrieval shows that people tend to show more interest in shape than color and texture in the retrieval process.

It is true that human brain tends to process local simple features before global complex features during its visual information retrieval. It is suggested that shape should be described by a combination of local primitives, which can be computed relatively efficiently and would not affect the overall shape even if some of them are changed slightly [73]. It is hence intriguing to decompose an object into elementary shape primitives and group them appropriately as higher level features that represent the object.

The decomposition process entails efficient and robust extraction of basic shape primitives, including circles, ellipses, polygons, and straight lines. Extraction of these primitives is an important image analysis technique with significant applicability in many industrial applications, especially those with manufactured objects or artificial environments. Typical applications include motion tracking/recognition in video, roads extraction from satellite images, engineering drawings analysis, robotics vision, and

medical imaging.

An ellipse is the projection of a circle on a plane and is found frequently in real world objects. It plays an important role in recognition of any objects with elliptical curves as salient features, such as microscopic cells, optical characters, human faces, and transportation signs.

An ellipse has five parameters – the center, the major/minor axes, and the orientation. If the length of the major axis is equal to the length of the minor axis, the ellipse becomes a circle. Hence the ellipse is a super set of the circle. Detection of any elliptical features naturally involves detection of circular features as well. Since the ellipse lies in a 5D search space, it is generally more complex than straight lines, the dimensionality of which is only 2.

In this study, we aim to detect multiple elliptical curves within noisy images. It can be easily extended to other lower order primitives. BMPGA are applied to this application since it is a typical multi-modal optimization problem - all primitives beyond acceptable levels of goodness are of interest.

5.1 Previous Work

In the literature, there have been a large amount of studies on detecting geometric primitives. Among various techniques, Hough Transform and its variations are most popularly used. A more recent technique utilizes the Genetic Algorithm and has yielded promising results. In this section, these techniques are investigated. Some approaches that are not popular but provide novel angles of view are also briefly introduced.

5.1.1 Hough Transform (HT)

The Hough Transform (HT) is one of the most widely used techniques for the detection

of various geometric shapes [74]. The basic idea of a standard Hough Transform is to represent a geometric shape by a set of appropriate parameters. For example, a circle could be represented by the coordinates of its centre and radius, hence three parameters. An ellipse, on the other hand, could be represented by its centre, long axis, short axis, and rotation angle, hence five parameters. Each foreground pixel is mapped into the space formed by the parameters, which are quantized into a number of bins. These bins are then accumulated. Peaks in the bins provide the best indication of where shapes may be.

In an HT, the intervals of the bins directly affect the accuracy of the results and the computational effort, since the parameters are quantized into discrete bins. For fine quantization of the space, the algorithm returns more accurate results, while suffering from large memory loads and expensive computation - especially in high dimensional spaces. Hence, the HT is most commonly used in two-dimensional or three-dimensional feature spaces and is unsuitable for higher dimensional spaces.

One of the fastest and most widely used variant of the Hough transform is the Randomized Hough Transform (RHT) proposed by Xu *et al.* [4]. The idea is to randomly pick n pixels from the image (n is the dimensionality of the geometric shape to be extracted), and then solve n parallel equations to get a set of parameters, with which, if a potential object is present, a score is assigned to the bin corresponding to the candidate shape. Hence in RHT, a bin represents a candidate shape, rather than a set of quantized parameters in HT. However, like HT, RHT also goes through an accumulation process for the bins. The bin with the highest score represents the best approximation of an actual shape in the target image.

McLaughlin's work [75] shows that RHT produces improvements in accuracy and

computational complexity, as well as a reduction in the number of false positives (non-existent ellipses), when compared with the original HT and a number of its improved variants.

Many HT based methods have been developed. They improve on the efficiency of the HT by exploiting geometric characteristics of the ellipses, such as edge gradient [76-79] and symmetry [74, 80, 81], by using intelligent means of dimensionality reduction [76, 77, 80, 82-84], or by image preprocessing to restrict the search space [78].

In [76] and [77], pairs of points on the ellipse are used to form lines that intersect at the center of the ellipse. As illustrated in Fig. 5-1, P_1 and P_2 are two points on an ideal ellipse, T_{12} is the intersection of two tangent lines from P_1 and P_2, and M_{12} is the middle point of the segment line P_1P_2. It has been proved that for an ideal ellipse, M_{12} and T_{12} are on the line that crosses the center O [85]. Therefore, given another pair of points P_3 and P_4, the center of the ellipse can be located by intersecting lines $M_{12}T_{12}$ and $M_{34}T_{34}$. Pairing points with unparallel gradients obtains a two dimensional accumulator of intersections. Local maxima of this accumulator indicate promising candidates of the centers of the ellipses.

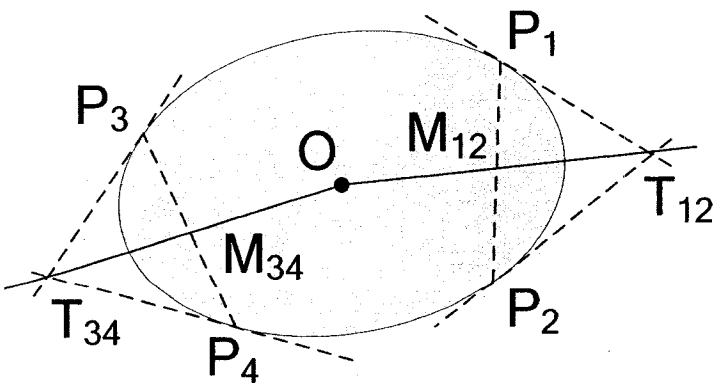


Fig. 5-1 Location of ellipse center by edge gradient

After the identification of ellipse centers, Guil *et al.* [76] and Zhang *et al.* [77] find the semi-axis ratio r (long axis/short axis) and the orientation of the ellipse θ through a polling process that employs another two-dimensional accumulator.

$$-r^2 = \frac{(q_1 - \tan(\theta))(q_2 - \tan(\theta))}{(1 + q_1 \tan(\theta))(1 + q_2 \tan(\theta))} \tag{5.1}$$

where q_1 and q_2 are slopes for lines P_1P_2 and $T_{12}M_{12}$, respectively.

The parameters of the major and minor axes of the ellipse are finally computed, using the set of four parameters found previously.

Cheng and Liu [78] and Troter *et al.* [79] take three points from the image and adopt the same principle in Fig. 5-1 to locate the center of the ellipse. They further compute the other three parameters of the ellipse using the coordinate of the three points and the center obtained. The candidate ellipses are then placed in an accumulator, like that used in the RHT, with each bin representing a candidate ellipse.

Chen and Liu [78] improve the efficiency of the RHT by restricting the search scope and neglecting invalid points (hence called Restricted RHT), including the operations of image preprocessing, image thinning and converting images to graphs to locate independent curves in the image. Each group of three points is randomly selected from an independent curve, rather than globally from the whole image.

The combination of gradient information and multistage processing reduces the computational complexity. However, evaluation of the tangents of the edge contours is generally very sensitive to noise. It is only suited for images containing less than 10% of speckle noise [86]. Moreover, the efficiency of these algorithms greatly relies on the number of candidate points.

Another school of researchers make use of the symmetry of the ellipses. Ho and

Chen [74] and Sewisy and Lebert [81] proposed two closely related algorithms to find lines of symmetry via a Hough transform applied to straight lines. They first scanned an ellipse from left to right and top to bottom. If each horizontal scan line intersects the ellipse at XL_i and XR_i, the midpoint of XL_i and XR_i - XM_i lies on the same straight line L_v (Fig. 5-2 (a)). Similarly, if the ellipse is scanned from top to bottom and left to right, each vertical scan line intersects the ellipse at YT_i and YB_i. The midpoint of YT_i and YB_i - YM_i lies on the same straight line L_h (Fig. 5-2 (b)). Ho and Chen proved that L_v and L_h intersect at the center of this ellipse, as illustrated in Fig. 5-2 (c).

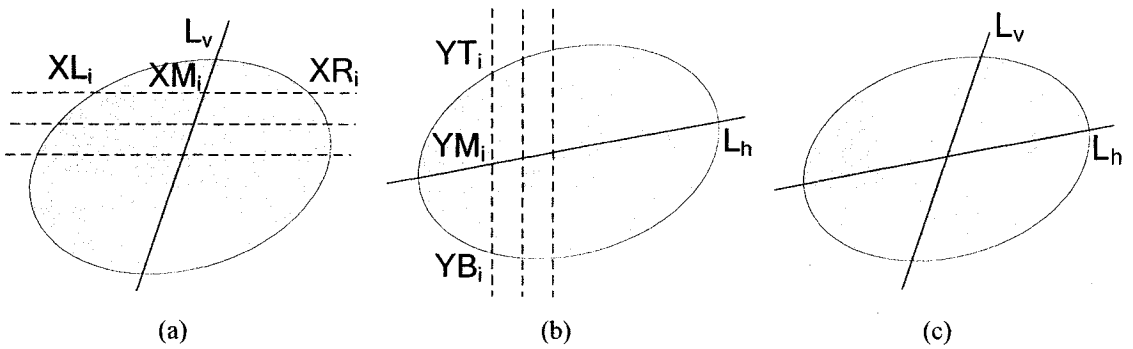


Fig. 5-2 Ellipse symmetry in Ho & Chen's algorithm

(a) The middle point XM_i of XM_i and XM_i lies on the straight line L_v;

(b) The middle point YM_i of YT_i and YB_i lies on the straight line L_h;

(c) The cross point of L_v and L_h is the center of the ellipse.

Hence this algorithm contains two phases. In phase 1, the image is scanned horizontally and vertically. A Hough Transform is applie to all midpoints XM_i and YM_i to extract candidates of L_v and L_h. Intersections of L_v and L_h then generate possible ellipse centers. For each ellipse center, all pairs of points relative to L_v or L_h are grouped into the same subimage. In phase 2, an RHT is adoped in each subimage. Possible groups of points on the ellipse are used to compute the remaining three parameters - the major and the minor axes as well as the orientation of the ellipse. These parameters sets are then

put into a three-dimensional accumulator, the peak of which is the candidate ellipse.

Lei and Wong [80] also proposed to detect symmetric axes of ellipses from contours. As illustrated in Fig. 5-3, as long as pairs of points P_1 and P_2 are symmetric about a symmetric axis (either major axis or minor axis) of the ellipse, their central line CL will coincide with this axis (T_2 in Fig. 5-3). Obviously, all central lines of such pairs of points have the same value of (s, θ).

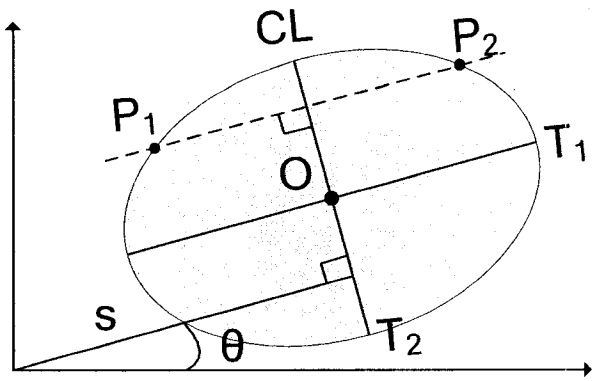


Fig. 5-3 Symmetric axes of an ellipse

This algorithm transforms a high-dimensional problem into two two-dimensional ones. The authors first used a Hough-based approach to obtain candidates of (s, θ). As such, all centers and orientations of ellipses can be found. After that, they used another Hough transform to find out the lengths of the major and minor axes.

Following the use of the geometric properties, some researchers also decompose the problem into multiple stages so that the dimensionality is effectively reduced. For example, in [76], [77] and [80], the authors used two two-dimensional accumulators in two polling processes.

In [82], Xie and Ji only used a one-dimensional accumulator to identify the most likely length of the minor axis, based on the ellipse geometry in Fig. 5-4. Given two endpoints (x_1, y_1) and (x_2, y_2) on the major axis of the ellipse, the center (x_0, y_0), the half-

length of the major axis a and the orientation of the ellipse α can be computed:

$$x_0 = \frac{x_1 + x_2}{2} \tag{5.2}$$

$$y_0 = \frac{y_1 + y_2}{2} \tag{5.3}$$

$$a = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{2} \tag{5.4}$$

$$\alpha = \text{atan}(\frac{y_2 - y_1}{x_2 - x_1}) \tag{5.5}$$

Given another point (x, y) on the ellipse, it satisfies:

$$\sqrt{(x - f_{1x})^2 + (y - f_{1y})^2} + \sqrt{(x - f_{2x})^2 + (y - f_{2y})^2} = 2a \tag{5.6}$$

where

$$f_{1x} = x_0 - \cos|\alpha|\sqrt{a^2 - b^2} \tag{5.7}$$

$$f_{1y} = y_0 - \sin|\alpha|\sqrt{a^2 - b^2} \tag{5.8}$$

$$f_{2x} = x_0 + \cos|\alpha|\sqrt{a^2 - b^2} \tag{5.9}$$

$$f_{2y} = y_0 + \sin|\alpha|\sqrt{a^2 - b^2} \tag{5.10}$$

where (f_{1x}, f_{1y}) and (f_{2x}, f_{2y}) are the coordinates of the foci of the ellipse f_1 and f_2, respectively.

The half-length of the minor axis b can be derived:

$$b^2 = \frac{a^2 d^2 \sin^2\theta}{a^2 - d^2\cos^2\theta} \tag{5.11}$$

where

$$\cos^2\theta = \frac{a^2 + d^2 - f^2}{2ad} \tag{5.12}$$

and d and f are the distance between (x, y) and (x_0, y_0) and the distance between (x, y) and (x_2, y_2), respectively. A one-dimensional accumulator collects votes for b.

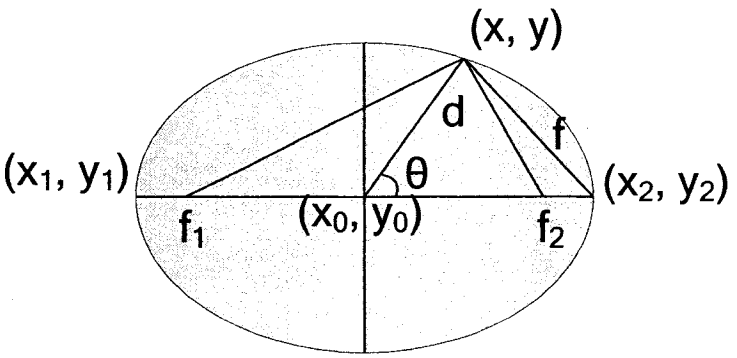


Fig. 5-4 Ellipse geometry: f_1 and f_2 are ellipses foci

[83] and [84] also used the same ellipse geometry and similar algorithms as above.

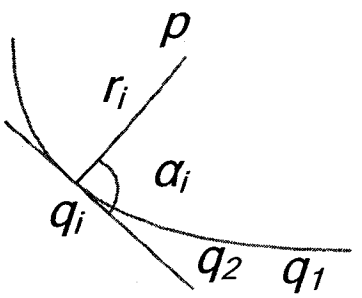


Fig. 5-5 Vector and tangent angle pair in generalized Hough Transform

Finally, HT can also be extended for arbitrary non-analytic curves, including polygons [87]. The basic idea is to select a reference point p and a set of boundary points q_i on the template model. Pairs of vectors r_i between p and q_i and tangent angles α_i at q_i are recorded in a lookup table (Fig. 5-5), indexed by quantized α. A 2-D accumulator array, which indicates possible x and y coordinates of p is created. For each edge point in the scene, its tangent angle is computed and corresponding entry for its vector is found in

the lookup table. The coordinates of p can thus be calculated and the corresponding bin in the accumulator array is increased by 1. Finally, the highest score indicates the position of the reference point. For rotated and scaled curves, a 4-D accumulator can be used, with quantized tangent angle α, rotation angle β, and scale factor μ.

5.1.2 Genetic Algorithms (GA)

As early as 1992, Roth *et al.* [88] proposed a way of extracting geometric shapes using GAs. Since then, a number of GA-based techniques have been developed for the purpose of detecting specific geometric shapes such as straight lines [89], ellipses [5, 86, 90, 91], and polygons [5, 90].

Some GA-based methods extract shapes *sequentially*, as in [88, 89, 91]. This method removes detected shapes from the image, one at a time, and iterates until there are no more shapes in the image. It is clear that this approach involves a high degree of redundancy and is computationally inefficient.

Lutton and Martinez [5] used a Sharing GA, first introduced by Goldberg *et al.* [20], to detect geometric primitives. The fitness of an individual i is given by:

$$newFitness(i) = \frac{oldFitness(i)}{\mu_i} \tag{5.13}$$

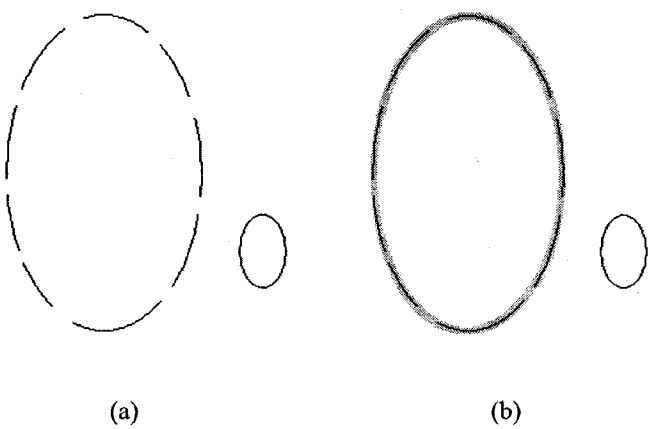
where

$$\mu_i = \sum_{k=1}^{N} (oldFitness(k) * sh(d_{ik})) \tag{5.14}$$

newFitness reflects the relative fitness of i with respect to its own subpopulation. N is the number of individuals in the subpopulation and $sh(d_{ik})$ is a sharing function defined in the same manner as that in standard Sharing [20]. They claimed that this scheme allows to "inhabit" more peaks than the classical Sharing techniques, which is a

beneficial factor for a typical multi-modal problem.

Unfortunately, the implementation of the fitness sharing is based on an assumption that the neighborhood of local optima is less crowded than the neighborhood of the global optimum and therefore, the fitness of local optima will be enhanced by sharing. This assumption is not valid in the application of geometric primitive detection, because imperfect primitives may actually attract many neighbors with a high probability, as long as they contain a sufficiently large number of pixels. This will deflect the search from exploring potentially promising areas, and will, often, result in missed detection.



(a) (b)

Fig. 5-6 Global and local optima
(a) A large imperfect ellipse (left) and a much smaller perfect ellipse (right)
(b) Locally-optimum candidate ellipse overlaid on top of left ellipse

Fig. 5-6 provides a concrete example. Fig. 5-6 (a) shows an imperfect ellipse and a perfect ellipse. The former, as a local optimum, is physically much larger than the latter, the global optimum. In the GA process, the local optimum, marked in grey in Fig. 5-6 (b), is at the centre of the densest subpopulation, because it attracts more samples around it. Hence, if the sharing function is applied, the fitness of the sub-optimal individual will be shared with the rest of its dense subpopulation, and on the other hand, the fitness of the

optimal individual will be shared with the rest of its less dense subpopulation. The result is in contrast to what is desired in the sharing scheme - the relative fitness of the global optimum will be more pronounced, but the relative fitness of the sub-optimum will be weakened. Indeed, our own experiments show that, when multiple primitives or/and high speckle noise are present in a target image, the Sharing GA fails miserably (Section 5.3.5.).

Smith *et al.* [16] further stated that the computation of the distance of an individual to any/all other individuals in a population has a time complexity of $O(N^2)$, where N is the size of the population. And since setting of σ_{share} is dependent on a uniform distribution of peaks in the search space, peaks that are distributed with less uniformity may be overlooked.

Jiang *et al.* [92] proposed a parallel GA for cell image segmentation. Since most of the cells in the human body have ellipse-like boundaries, the algorithm essentially extracts ellipses. A multi-population GA is used. The number of the subpopulations and the size of each subpopulation are both preset. Each subpopulation is evolved independently. Migration occurs in a random number of subpopulations, which send their best individuals to the neighboring subpopulations and replace the worst individuals there.

Ayala-Ramirez *et al.* [93] used a GA to detect circles. The chromosome is constituted by three edge points on the candidate circle. For each individual c, the fitness function $F(c)$ counts the number of ideal edge points that are actually present in the edge image:

$$F(c) = \frac{\sum_{i=1}^{N_s} E(x_i, y_i)}{N_s} \tag{5.15}$$

where (x_i, y_i) is obtained by uniform sampling of the circle boundary:

$$x_i = x_0 + r \cdot \cos\frac{2\pi i}{N_s} \tag{5.16}$$

$$y_i = x_0 + r \cdot \sin\frac{2\pi i}{N_s} \tag{5.17}$$

N_s is the number of samples on the circumference of the circle and r is the radius of the circle. r can be computed from the three edge points in the chromosome.

The authors claimed that their approach detects circles with sub-pixel accuracy.

5.1.3 Tabu Search

Ke *et al.* [94] proposed to use a Tabu search technique. Tabu search uses a tabu list (short-term memory) that keeps track of recent solutions to prevent the search from becoming trapped in a local minimum. At each iteration, a number of new solutions around the current solution are generated. Among them, the best solution that is not tabu, or that is tabu but satisfies certain so-called aspiration conditions is set as the current solution. The search stops when a certain termination criterion is satisfied.

5.1.4 Upwrite Method

The Upwrite method [95] uses a spot algorithm to compute local models of pixels within a small resolution radius r. If r is small enough, the data in the neighborhood can be modeled with a straight line. The set of local models positioned along the same geometric primitive are then grouped together. Finally, each geometric feature is characterized by Zernike moments as: lines, circles, ellipses or others.

5.1.5 Hybrid Approaches

In the literature, there also exist some hybrid approaches that consist of different algorithms. It is interesting to find that these approaches usually adopt a GA and another algorithm. Examples are given below.

5.1.5.1 GA + Local Search

Yin [96] adopts a hybrid scheme consisting of a GA phase and a local search phase. During the GA phase, a number of candidates with fitness values above a certain threshold are fed into the next phase. During the local search phase, the neighborhood of the candidates is explored, based on the observation that the fitness values of the true candidates can be improved by adjusting the parameter values locally, whereas the false ones are rarely improved by looking to their neighbors.

5.1.5.2 GA + HT

Kasemir and Betzler [86] proposed to combine a modified version of HT with a GA to detect ellipses of limited eccentricity. Assuming that the orientation of the ellipse is fixed, there is:

$$(x-x_0)^2+\frac{(y-y_0)^2}{e^2}=r^2 \tag{5.18}$$

where (x_0, y_0) is the center of the ellipse, e is the ratio of the vertical and the horizontal semi axes of the ellipse:

$$e=\frac{r_y}{r_x},\quad r=r_x \tag{5.19}$$

The GA searches a two-dimensional space $\{x_0, y_0\}$. For each chromosome of the GA (x_0, y_0), the HT searches a two-dimensional space $\{r, e\}$ and finds the best candidate

with the highest votes. Hence for a population of size N, each iteration of the GA's main loop requires N runs of the HT. This algorithm has very high computational demands.

5.1.5.3 GA + Tabu Search

In [97] and [98], a GA and a Tabu Search are incorporated into the proposed method. Specifically, the idea of "the survival of the strongest" in the GA is adopted. The algorithm has two level selections. The first level starts with a guess of N feasible candidates called parents, chosen randomly in the search space. Each parent can generate a number of children, which form a family. The Tabu Search is then used to select the child as parent for the next generation.

The second-level selection is the competition between the families, in which the number of children (NTS) that could be generated in the next generation is proportional to their fitness values:

$$NTS = \frac{N_0 \cdot NTS_0 \cdot F}{S} \tag{5.20}$$

where N_0 is the initial number of the families, NTS_0 is the initial number of children generated for each family, S is the sum of the fitness values of all families, and F is the average fitness value of the family.

5.2 HT versus GA

Procter *et al.* [91] made an interesting comparison between GA and RHT. These two techniques have the following features in common:

– Representation of geometric shapes using minimal sets of parameters;

– Random sampling of image data;

– Sequential extraction of multiple shapes.

Their experiments clearly demonstrate that GA based techniques return superior results to those produced by RHT methods when a high level of salt and pepper noise is present in the image, but RHT methods are more attractive for relatively noise-fee images.

Indeed, for an elliptical curve of L pixels long in an image with a total of A pixels, the probability of locating this curve from a single sample is:

$$P = \frac{C_L^n}{C_A^n} \tag{5.21}$$

where n is the dimensionality of the geometric shape and C_X^y is the unordered selection of y pixels from the pixel set X.

With the same probability P, for each sampling and the same number of samples (say N), RHT gets N independent chances of detection when exploring the search space sequentially. In contrast, a GA explores the search space in parallel (using a population with size M), while guiding the search towards promising areas within N/M generations. Moreover, unlike the RHT, which locates peaks in the fitness surface after an exhaustive search of the space, a GA generates new improved individuals from existing ones, mainly through crossover and mutation. The RHT executes a blind sequential search, whereas a GA is able to search the space iteratively with feedback and in parallel. Hence, GA based algorithms have inherent strengths which, if properly used, can make them a better approach than any RHT based algorithm.

Of course, if there is only a small amount of noise in the image ($L \approx A$ and $P \approx 1$), RHT will converge quickly, since each sampling has a high probability of locating the target shape. However, in cases where there is a lot of noise, multiple ellipses, or partial ellipses with some noise, RHT tends to overlook small elliptical curves, since C_L^n

decreases dramatically with a small L, which leads to an extremely small P. In this case, a GA based algorithm is likely to converge faster, and exhibit more tolerance to noise as well as accuracy than the RHT. This conclusion matches the experimental results of Procter *et al.* [91] and is further supported by our own experiments

In summary, the core of an HT-based technique is a polling process, which is equivalent to a blind sequential search of the search space. On the other hand, the core of a GA based technique is a parallel semi-directed search of the search space. This inherent disparity between these two approaches is at the root of their widely differing performances, as exhibited in our own experiments in Section 5.3.5.

5.3 Methodology

When using GA in the framework of pattern recognition, there are several issues that must be addressed [7]:

- The reformulation of typical pattern recognition problems as optimization ones;
- The encoding of the solution into the genotype;
- The definition of the fitness function on the attainable performance: in many cases of interest, it leads to a multimodal fitness function. Therefore the further issue of making a GA able to search for many optima arises.

These issues will be studied in detail in this section.

5.3.1 Methodology Overview

The algorithm basically follows the flow in Fig. 4-1. Initially, a single subpopulation is generated by creating a number of chromosomes whose genes are randomly selected from the set of foreground pixels in the image (see Section 5.3.3). The subpopulation is

then ranked in terms of both *similarity* and *distance* (Section 5.3.4) and searched for good candidates - if any. A clustering technique (Section 5.3.5) is used to divide the chromosomes into a number of clusters (or subpopulations). From that moment on, all the subpopulations are evolved, in parallel. The program, as a whole, terminates when all (full and partial) ellipses are found, or when a pre-set maximum number of generations or evaluations is reached.

If one of the subpopulations converges on an optimal or a suboptimal chromosome, that whole subpopulation and the corresponding ellipse are removed. This has the positive side effect of accelerating the search process, since the number of foreground pixels is decreased and as a result, the size of candidates' pool (may contain both phantom and actual ellipses) is also decreased.

5.3.2 Ellipse Geometry

Chromosomes in BMPGA are no more than candidate ellipses, and hence understanding the geometry of ellipses is essential to understanding chromosomal representation. The ellipse equation can be written as:

$$ax^2 + 2hxy + by^2 + 2gx + 2fy + 1 = 0 \qquad (5.22)$$

Assuming we have five distinct points on the perimeter of an ellipse, we can solve five linear equations to obtain a, h, b, g and f using an efficient numerical technique - the LU Decomposition algorithm [99]. The geometric parameters (the long and short radii of an ellipse (r_{max} and r_{min}), the co-ordinates of the center (x_0, y_0) and the orientation with respect to the X axis θ) for an ellipse can be computed, by substituting the values of a, h, b, g and f into a set of five equations, as in [91].

$$x_0 = \frac{hf - bg}{C} \tag{5.23}$$

$$y_0 = \frac{gh - af}{C} \tag{5.24}$$

$$r_{\max} = \sqrt{\frac{-2\Delta}{C(a+b-R)}} \tag{5.25}$$

$$r_{\min} = \sqrt{\frac{-2\Delta}{C(a+b+R)}} \tag{5.26}$$

$$\theta = \frac{1}{2}\arctan\left(\frac{2h}{a-b}\right) \tag{5.27}$$

where,

$$C \equiv ab - h^2 \tag{5.28}$$

$$R^2 \equiv (a-b)^2 + 4h^2 \tag{5.29}$$

$$\Delta \equiv \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & 1 \end{vmatrix} \tag{5.30}$$

Again, the LU Decomposition algorithm [99] is used to compute Δ efficiently.

5.3.3 Representation and Initialization

Chromosome encoding is application dependent. Specifically, for the task of extraction geometric primitives, chromosomes are typically encoded in two ways. One comes from Roth and Levine's approach [88]: a minimal set of N (N is the dimensionality of the shape) points on the shape's boundary constitutes the chromosome, where each point is a gene. The parameters of the shape can then be computed from N parallel equations.

The other way encodes the chromosome using the geometric parameters of primitives directly. Here use an ellipse as the example. Mainzer represents an ellipse as [90]:

$$\begin{aligned} x_i &= x_0 + a_i k_x \cos(\alpha) + b_i k_y \sin(\alpha) \\ y_i &= y_0 + a_i k_x \sin(\alpha) + b_i k_y \cos(\alpha) \end{aligned} \tag{5.31}$$

where a_i and b_i are defined circle coordinates, with

$$\begin{aligned} a_i &= \cos(\frac{2\pi i}{N}) \\ b_i &= \sin(\frac{2\pi i}{N}),\ i \in < 0; N-1 > \end{aligned} \tag{5.32}$$

x_i and y_i are transformed coordinates, x_0, y_0 are coordinates of transformed ellipse centre, k_x, k_y are coefficients of ellipse stretch in orthogonal direction and α is spin of transformed ellipses. The chromosome is set of x_0, y_0, k_x, k_y and α.

This representation is similar to an ellipse chromosome encoded as a, b, x_0, y_0, and θ (Fig. 5-7), where a and b are long and short axis of the ellipse, respectively, x_0, y_0 are the coordinates of the center, and θ is the rotation angle of the ellipse.

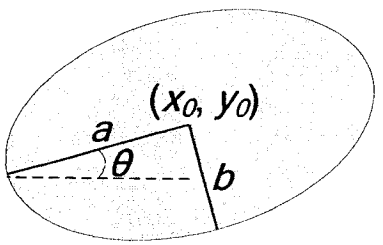


Fig. 5-7 Geometric parameters of an ellipse

Lutton and Martinez [5] encodes the ellipse chromosome with the center O, the point P, and the rotation angle a.

They claimed that their individual chromosome represents a unique ellipse, while Roth and Levine's chromosomes have redundancy since different chromosomes may

represent the same ellipse with the same set of points in different permutation.

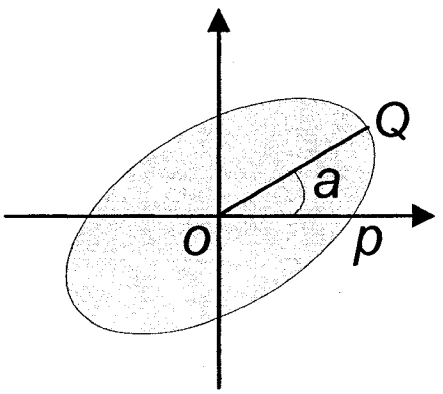


Fig. 5-8 Lutton & Martinez's encoding for an ellipse chromosome

The encoding of ellipses *via* their direct geometric parameters may have problems on initialization of the population and generation of new individuals. First of all, it is not easy to estimate the range of the parameters. For example, the length of the long axis of an ellipse could be between 0 and $\frac{\sqrt{width^2 + height^2}}{2}$ (*width* and *height* are the width and height of the image). However, it is also possible that a huge partial ellipse exists with its major axis beyond the upper limit. Secondly, without making good use of the domain knowledge, i.e., the distribution of pixels in the image, these techniques provide no guarantee that the resulting candidate ellipses will contain any point from the actual ellipses in the target image. Hence they usually spend a long time evaluating the fitness of many invalid chromosomes.

Therefore, we choose to encode the chromosome with a set of points as in [88]. We disallow *identical* individuals in the population. Two chromosomes are identical if their phenotypes (geometric parameters) are identical.

Note that the encoding scheme allows for some measure of redundancy. This occurs when chromosomes have sufficiently close, but *not* identical, phenotypes. This, however, is a *desired* property, which, if properly utilized, will facilitate the generation of good phenotypes. It is observed that a good genotype (i.e., a chromosome with all genes lying on a good ellipse) does not necessarily result in an equivalently good phenotype, due to digital displacement errors.

As shown in Fig. 5-9, a chromosome with genes ($P_1P_2P_3P_4P_5$) gives a sub-optimal ellipse S, which deviates slightly from the optimal ellipse O, since all the genes lie densely on a small segment of the ideal curve. A better chromosome, on the other hand, tends to distribute genes evenly along the boundary of the ellipse. It can be obtained through either mutation or the crossing over of two sub-optimal chromosomes with *close* phenotypes.

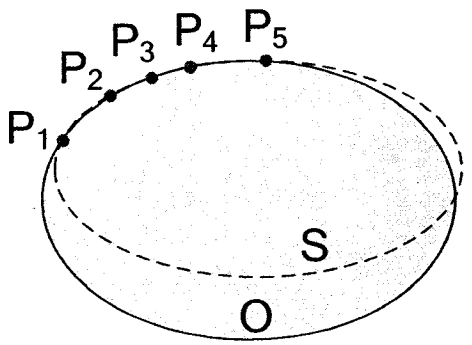


Fig. 5-9 A good genotype that does not correspond to a good phenotype

5.3.4 Fitness Evaluation

Most of the reported work [5, 90] evaluate the fitness of a candidate ellipse by counting the number of foreground pixels in the target image that coincide with the perimeter of the candidate ellipse. To enhance the robustness of the matching function, Mainzer [90] distinguishes pixels lying near the perimeter of a candidate (ideal) ellipse from those far

from it, and assigns a penalty to the latter. Mainzer defines fitness S of a given candidate ellipse as:

$$S = \sum_{x,y} \left(\underset{\forall i,j}{MAX} (E(x+i, y+j) - \frac{1}{d}(|i| + |j|)) \right) \tag{5.33}$$

$E(x+i, y+j)$ is a function that returns 1 if there exists a foreground pixel $(x+i, y+j)$ coincident with, or close to a pixel (x, y) on the candidate ellipse. Otherwise the function returns 0. i and j are the horizontal and vertical displacements, respectively, between the two pixels. d is a constant determined by the nature of the image. Finally, *MAX* returns a maximal value over all combinations of i and j. The maximum is obtained whenever $E(x+i, y+j)$ returns 1 and the second-term $\frac{1}{d}(|i| + |j|)$ returns a minimum. Equivalently, *MAX* looks for an appropriate pixel $(x+i, y+j)$ that is closest to (x, y). If there exist points in the image that exactly (i.e. $i = 0$ and $j = 0$) match every point in the candidate ellipse, this candidate ellipse will receive the maximum fitness of 1.

In [5], a grey-level "distance image", in which each pixel's grey value indicates its distance to the nearest contour point, is used. This distance image is computed from the original image using two morphological masks. Similar to the approach above, the points not exactly on the perimeter of a candidate ellipse are punished via a displacement factor. However, the construction of the distance image requires a large amount of extra storage. And the manual tuning of the two distance parameters in the mask requires extra preparatory work before the algorithm can be used.

Lutton *et al.* [5] also introduced another fitness term that counts effective contour pixels "to favor bigger primitives". However, bigger primitives are not necessarily better ones; and the extent to which an actual pattern matches a candidate shape has nothing to

with the pattern's absolute length or size.

It can be seen that most GA-based methods gives us a fitness function that lacks necessary flexibility for the detection of *multiple* primitives. A fitness function with a single term will drive the whole population towards a single global optimum. Hence, in the presence of multiple optima, the final winner is obtained randomly. Moreover, when there are both perfect *and* imperfect candidates, the latter, being *locally* optimum, will most likely be replaced by better individuals during evolution, and eventually ignored.

In order to detect full as well as partial multiple ellipses with varying types and degrees of imperfections, we propose a bi-objective scheme, i.e., the fitness of a chromosome is measured in two terms: *Similarity* and *Distance*.

(A) *Similarity*

Similarity (S) indicates how much the actual pixels match the perimeter of an ideal complete ellipse. S is defined as:

$$S = \frac{\sum_{(x,y)} \frac{E(x+i, y+j)}{d_{x+i,y+j}}}{\#total} \tag{5.34}$$

Like Mainzer's definition above, for a given point (x, y) on the candidate ellipse, the term $E(x+i, y+j)$ returns 1 if there is a point $(x+i, y+j)$ that coincides with, or is close to (x, y); otherwise $E(x+i, y+j)$ returns 0. The terms i and j represent the horizontal and vertical displacements between the *ideal* pixel (x, y) and the actual pixel $(x+i, y+j)$, respectively. #total is the total number of pixels on the ideal ellipse's perimeter. $d_{x+i,y+j}$ is a distance factor (note that it is different from the fitness term *distance* defined later) used to punish those pixels far away from the ideal pixel (x, y). It is given by:

$$d_{x+i,y+j} = e^{\frac{|i|+|j|}{4}} \tag{5.35}$$

where e is the natural logarithm base, and i and j are the same terms of i and j in equation (5.34). The value of S belongs to [0, 1], with 1 indicating a perfect match and 0 no match at all.

Fig. 5-10 shows how an actual point (Q) is determined with respect to an ideal point (P) and how the distance factor $d_{x+i,y+j}$ between them is computed. The dashed arc belongs to an ideal template with center C. The solid arcs are the actual pixels in the image. If P does not coincide with any point in the image, a line is originated from C passing through P. A fast search, based on Brensenham's algorithm [100], is initiated from P, both outward and inward along this line, until the closest actual point is reached (Q). The horizontal and vertical displacements i and j between them are used to compute $d_{x+i,y+j}$ as in equation (5.35).

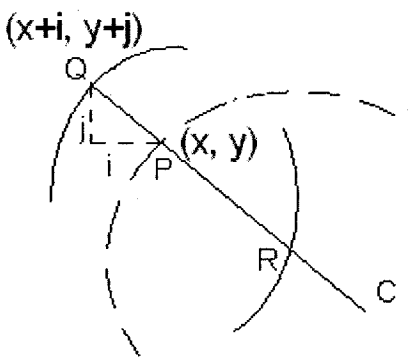


Fig. 5-10 Matching of a candidate ellipse, point by point, to potential actual ellipses in an image

(B) *Distance*

Distance (D) indicates how far or close is the actual pattern to the ideal ellipse. It is defined as:

$$D = \frac{\sum_{(x,y)} d_{x+i,y+j}}{\#eff} \tag{5.36}$$

The term $d_{x+i,y+j}$ is defined in equation (5.35) above. #eff is the total number of actual points that are successfully matched with points on the ideal ellipse. D ranges from $[0, \infty)$.

Fig. 5-11 and Table 5-1 provide an intuitive visual interpretation of these two fitness terms. In Fig. 5-11, the dashed arcs are ideal template ellipses, which are superimposed on three grey actual ellipses 1, 2 and 3. Ellipse 1 is a perfect ellipse, with Similarity S=0.995758 and Distance D=0.010167 (Ideally, the values should be 1 and 0, respectively. However, they are immensely close to the desired values due to digital rounding errors). Ellipse 2 is an incomplete pattern with the lowest Similarity among the three ellipses, whereas its Distance is as good as ellipse 1, since all its actual points lie exactly on the ideal template. Ellipse 3 is a complete but irregular ellipse. Its Similarity value lies between those of ellipses 1 and 2, whereas its Distance value is greater than both, due to its irregular segment AB, which is somewhat distant from the ideal trace. Hence we have:

$$Similarity_1 > Similarity_3 > Similarity_2$$

while

$$Dstance_2 \approx Dstance_1 < Dstance_3$$

Ellipse 1 has an almost perfect match to the template. Ellipse 3 is more *complete*, and therefore more *similar* to the template than ellipse 2, with a larger *Similarity* value. On the other hand, Ellipse 2 is *closer* to the template than 3, since its *Distance* is *smaller*. Generally, we aim to seek candidates with sufficiently large similarity *and* small distance.

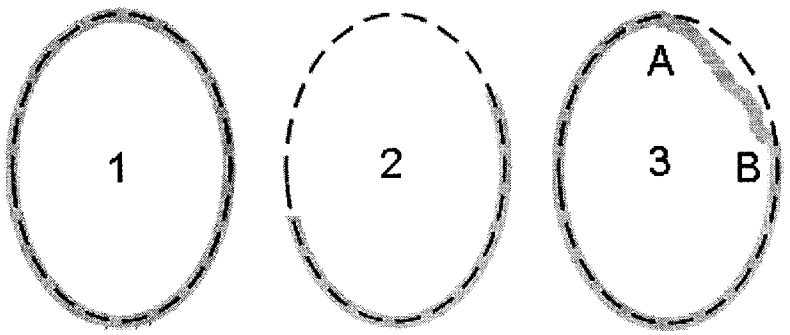


Fig. 5-11 Perfect and imperfect ellipses

Table 5-1 Fitness of Ellipses in Fig. 5-11

Ellipse	Similarity	Distance
1	0.995758	0.010167
2	0.475883	0.009647
3	0.804562	0.402721

The bi-objective fitness scheme enhances the probability of survival of candidate ellipses that have low fitness relative to one of the two fitness terms and high fitness relative to the other term. Indeed, if only one of the two fitness terms is used, then many imperfect but promising ellipses (such as ellipse 2 or 3) will eventually be deselected, resulting in a significant reduction in population diversity.

It is also worth stressing that the detection of more than one primitive shapes, ellipse or otherwise, in an image, is in essence a multi-modal optimization problem, where several optima are present. The essential challenge of such problems is to craft algorithms that do not focus the search on any one optimum point. Instead, the algorithm should proceed to explore all areas of potential optima. Hence, measures that increase the diversity of a GA population, such as the use of two fitness terms, may well lead to a more comprehensive search and, as our results will show, more comprehensive results.

To compute the fitness efficiently, we further assume that the ideal template is

centered at the origin of coordinates with a horizontally aligned long axis (Fig. 5-12).

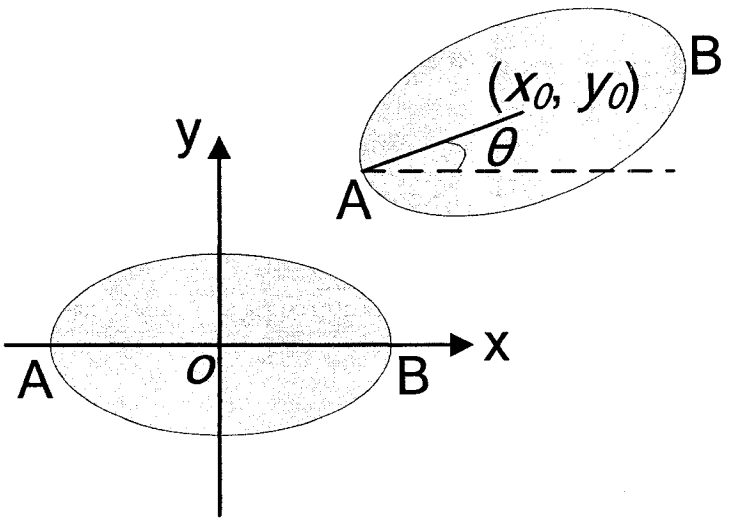


Fig. 5-12 2D Geometric transformation of ellipse

A classic midpoint ellipse algorithm [100] is then used to traverse the perimeter of this candidate ellipse. This algorithm favors integer computation, and it only computes a quarter of the ellipse's perimeter. All the other points in the remaining 3 quadrants are obtained from symmetry. Each computed pixel is matched to its *actual* ideal position using:

$$\begin{bmatrix} x^T \\ y^T \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & x_0 \\ \sin\theta & \cos\theta & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5.37}$$

where (x, y) are the original coordinates and (x^T, y^T) are the transformed coordinates. θ is the orientation. Finally, the term $E(x+i, y+j)$ is replaced by $E^T(x+i, y+j)$, giving us the final form of equation (5.34):

$$S = \frac{\sum_{(x,y)} \frac{E^T(x+i, y+j)}{d_{x+i,y+j}}}{\#total} \tag{5.38}$$

5.3.5 Clustering

As discussed in Section 4.1.6, the clustering process in BMPGA involves migration, splitting, and merging. Different to the application of multimodal function optimization in Chapter 4 though, the Recursive Middling algorithm is not used. Since the fitness computation of each ellipse chromosome is costly, the cost of recursive fitness calculation will be prohibitively high. Hence the Euclidean distance ED is used as the clustering criterion.

ED is computed as follows: given two sets of ellipse parameters (a_1, b_1, x_1, y_1, ω_1) and (a_2, b_2, x_2, y_2, ω_2), a_i and b_i are the long and short axes, respectively; (x_i, y_i) is the center and ω_i is the orientation.

$$ED = \frac{\sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2 + (x_1 - x_2)^2 + (y_1 - y_2)^2 + (\omega_1 - \omega_2)^2}}{5} \tag{5.39}$$

The center of a cluster is its up-do-date best individual. ED between a chromosome and existing cluster centers determines whether it remains in its own cluster or migrates.

If ED between a chromosome and its own cluster centre is higher than an empirically derived threshold, this chromosome migrates to another cluster with the closest centre to it. The resulted vacancy in the original cluster is filled by the processes of evolution (see Section 5.3.6). If there is not any cluster sufficiently close to the migrating chromosome, a new cluster is created around this chromosome. Two clusters are merged if ED between their centers is lower than the threshold.

5.3.6 Evolution

As discussed in Chapter 4, BMPGA is not a typical multi-objective optimizer that seeks a

set of Pareto-optimal solutions [69], which satisfy several (usually conflicting) objective functions. The two objectives used in BMPGA do not conflict with each other. On the contrary, they usually reach their optima concurrently. Therefore, the evolutionary strategies used are especially tailored to meet our requirements, and are not meant as general multiple-objective optimization strategies.

The basic evolutionary process is the same as that discussed in Section 5.3.6. Crossover (recombination) and mutation are specifically introduced here.

5.3.6.1 Crossover

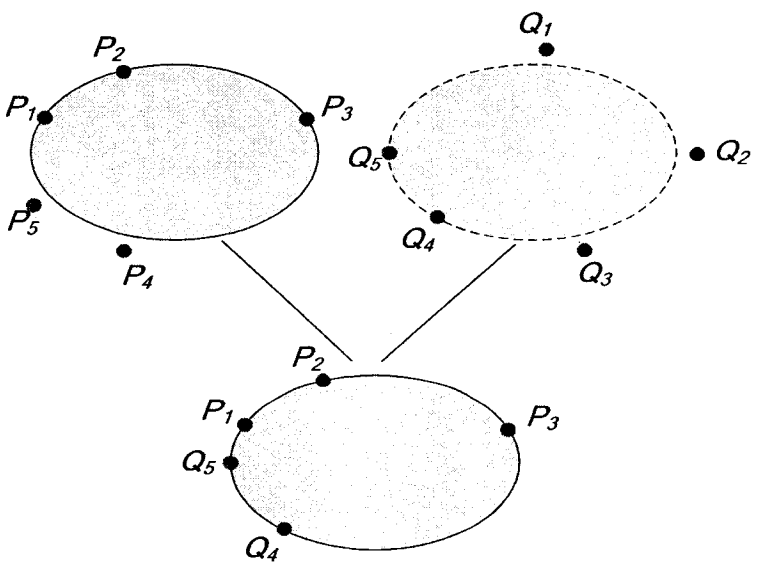


Fig. 5-13 Result of crossing-over two chromosomes: phenotypic view

Given the fact that each chromosome is defined by a set of points on the perimeter of an ellipse, we can assert that simple single point crossover is an effective method. A pivot is selected at random, and the parent chromosomes' genes on either side of the pivot are swapped to create the offspring, as shown in a genotypic view in Fig. 2-5 in Section 1.1.1.1.5.

A possible positive effect of the crossover is shown in a phenotypic view in Fig.

5-13.

5.3.6.2 Mutation

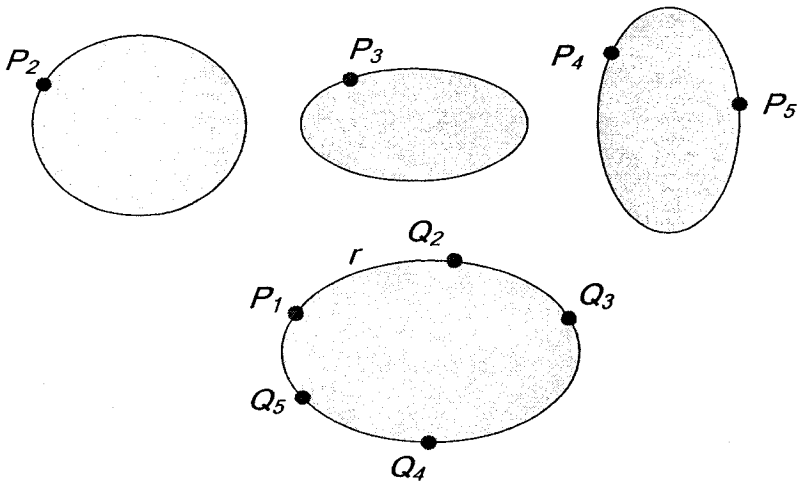


Fig. 5-14 Mutation operation

A standard mutation randomly selects a gene (in our case: a pixel), and reassigns it a new value. This operation is global, in the sense that the new value of the gene is picked, at random, from all foreground pixels in the image. Alternatively, we define a localized mutation operator configured specifically for our application.

First, a point is randomly selected from the chromosome ($P_1P_2P_3P_4P_5$) that we intend to mutate, as shown in Fig. 5-14. Starting from this point (P_1), a local path r is traversed with a trace tracking algorithm, until a preset maximum number of points, or an end/intersection point, is reached. If r is large enough, the remaining four genes Q_2–Q_5 are picked from it randomly. Therefore, the original chromosome ($P_1P_2P_3P_4P_5$) is mutated into a new one ($P_1Q_2Q_3Q_4Q_5$), with all the genes (P_1, Q_2, Q_3, Q_4, and Q_5) lying on the same curve. Clearly, as long as the starting point lies on a promising candidate, it is highly possible that the other four points will too.

In summary, a localized mutation actually replaces four genes within a chromosome, whereas a standard mutation replaces only one. The localized mutation randomly picks up new points on a local path originating from a selected starting point, whereas the latter picks up new points, indiscriminately, from the whole image. Therefore, the new localized mutation dramatically enhances the possibility of mutating an existing candidate into a fitter one.

5.4 Experiments

This section compares the performances of the BMPGA, Sharing GA (SGA) [5], and Randomized Hough Transform (RHT) [4], using both synthetic and real world images.

As discussed in Section 1.1.1.1.9, SGA is an important multimodal GA and has attracted a lot of studies and research efforts on it [18, 20, 22, 24, 30, 42-44]. SGA is already used in a wide range of pattern recognition tasks [5, 7, 101-106], including the application of ellipses extraction [5]. It is interesting to see how BMPGA is compared to SGA in this application.

RHT, on the other hand, is one of the most popular methods on geometric shape detection. As a matter of fact, it has been a milestone in this field. There have been many improved versions of RHT in recent years [74, 76-84]. As studied in Section 5.1.1, the performance of RHT is enhanced by improving peripheral aspects of the algorithm, such as reducing the dimensionality [76, 77, 80, 82-84] or making use of the geometric properties [74, 76-81]. The *core* of RHT is unaltered - it is still in essence a blind enumeration and accumulation process without any heuristics. Hence RHT is a good benchmark algorithm to compare with. But it is worth noting that we are more interested in comparing its core scheme with our BMPGA. Those peripheral strategies can be

inserted into the algorithm for possible acceleration. But they never play crucial roles in the algorithm itself.

There also exists a hybrid method that involves both GA and RHT [86]. However, the demanding cost of the algorithm makes its feasibility limited.

To carry out a fair comparison, we use the same numerical technique to compute the geometric parameters of an ellipse (see Section 5.3.2) and the same method to calculate the fitness (described in Section 5.3.4). For GA-based techniques (SGA and BMPGA), the same genetic operators are used (i.e. crossover and mutation).

The performance of these algorithms is measured by validation accuracy, false positive rate, and average cpu time of the core algorithm excluding image pre-processing. Accuracy is defined as the ratio of correctly detected ellipses with respect to the total number of ellipses actually present in the target image. Over detection is reflected in the false positive (fp) statistics. All the experiments were run on an Intel Xeon 2.66 GHz w/ 512 KB of cache, 512 MB DDR RAM and running Red Hat Linux 8.0.3.2–7.

5.4.1 Synthetic Images

The synthetic images are comprised of two sets, set A and set B. Set A is partitioned into seven collections of 50 images each, totaling 350 images. These collections are used to test the algorithms' performances on images with different number of ellipses. The first collection contains 50 images of single ellipses, the second collection contains images of two ellipses, and so on. The last collection contains 50 images of seven ellipses.

Set B, on the other hand, is used to test the algorithms' performance on noisy images. This set is made of five collections of 50 images each, totaling 250 images. The first, second, third, fourth and fifth collections contain images with 0, 0.5, 1, 3 and 5%

salt-and-pepper noise, respectively.

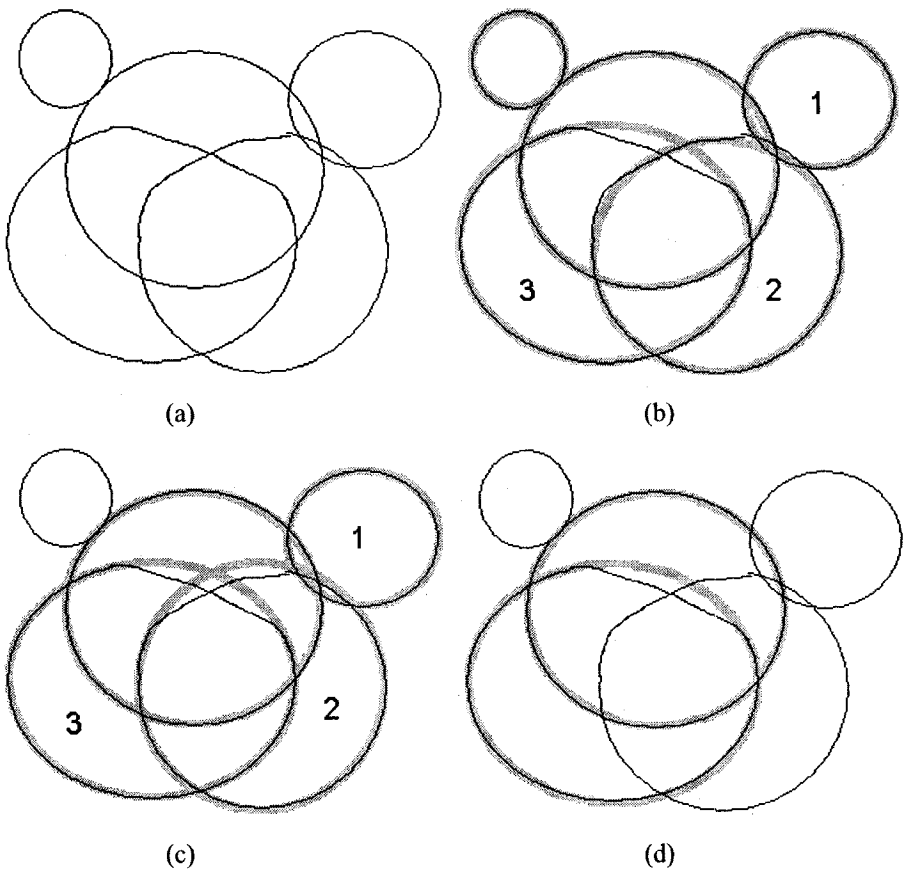


Fig. 5-15 A typical image with multiple ellipses: detected ellipses overlaid in thick grey

(a) Original Image; (b) Ellipses Detected by BMPGA;

(c) Ellipses Detected by RHT; (d) Ellipses Detected by SGA

The ellipses in the synthetic image database are not all full and perfectly formed ellipses; we make sure that some images in both sets have at least one partial or deformed ellipse. A typical example is presented in Fig. 5-15, in which there are five ellipses, two of which are malformed. The figure also shows the results of BMPGA, RHT, and SGA by superimposing the detected ellipses on the original image.

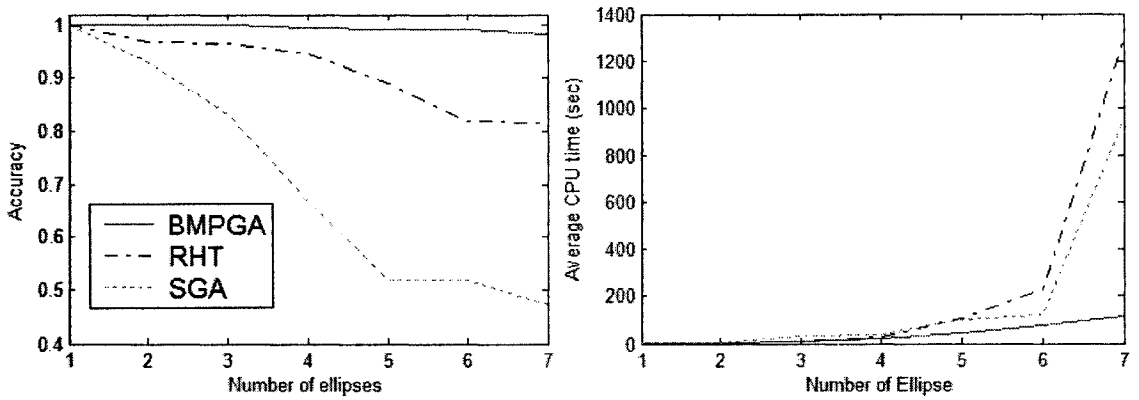
As seen in Fig. 5-15 (c), RHT misses the smallest ellipse since, in line with

equation (5.21) in Section 5.2, the probability of locating it is smaller than the probability of locating the other larger ellipses. The fitness values shown in Table 5-2 are Similarity only because Similarity is the only measure/factor of fitness used by all these three algorithms. Uniquely and exclusively, the bi-objective scheme is part of the core mechanism of the BMPGA algorithm, while RHT and SGA do not use a second fitness term.

Table 5-2 Parameter and fitness values for ellipses detected in Fig. 5-15

Algorithms	Center	Major Axis	Minor Axis	Orientation (°)	Fitness	Time (sec)
BMPGA	(71, 54)	29	29	0	0.970238	19.93
	(152, 123)	81	73	0.76	0.955702	
	[1](259, 80)	47	42	1.03051	0.934348	
	[2](196, 176)	78	72	-3.9616	0.822264	
	[3](125, 168)	91	74	0.533168	0.790112	
RHT	(152, 123)	81	73	0.964792	0.976567	31.25
	[1](260, 79)	48	42	-1.38724	0.810691	
	[2](195, 172)	78	78	0	0.680005	
	[3](124, 169)	90	73	4.89973	0.663584	
SGA	(153, 123)	81	73	1.05523	0.968314	172.42
	(126, 169)	91	75	2.18694	0.678744	

In reference to Table 5-2 and Fig. 5-15 (b) and (c), BMPGA returns better fitness values than RHT for ellipses 1, 2 and 3. This is because BMPGA, via evolution, executes a parallel search, focused at different localities of the search space, and progressively improves the fitness of the candidates; whereas RHT carries out a one-shot blind search through the whole space.



(a) (b)

Fig. 5-16 Comparing the performance of RHT, SGA and BMPGA in detecting multiple ellipses

(a) Accuracy versus number of ellipses; (b) CPU Time versus number of ellipses

Fig. 5-16 contrasts the performance of the three algorithms in terms of accuracy and average CPU running time. Fig. 5-16 (a) shows that the accuracy of SGA decreases dramatically as the number of ellipses increase. Similarly, the accuracy of RHT also decreases gradually from 100% for one ellipse to approximately 80% for seven ellipses. In contrast, MPGA outperforms the other two algorithms by maintaining an almost perfect level of detection accuracy, and slightly dipping to around 98%, for images with seven ellipses.

Fig. 5-16 (b) graphs the amount of CPU time used by various algorithms with 1–7 ellipses. BMPGA has clear advantage over both RHT and SGA with regards to speed. There is almost no difference in speed for images with four or less ellipses. However, for five or more ellipses, BMPGA realizes a clear and widening gap.

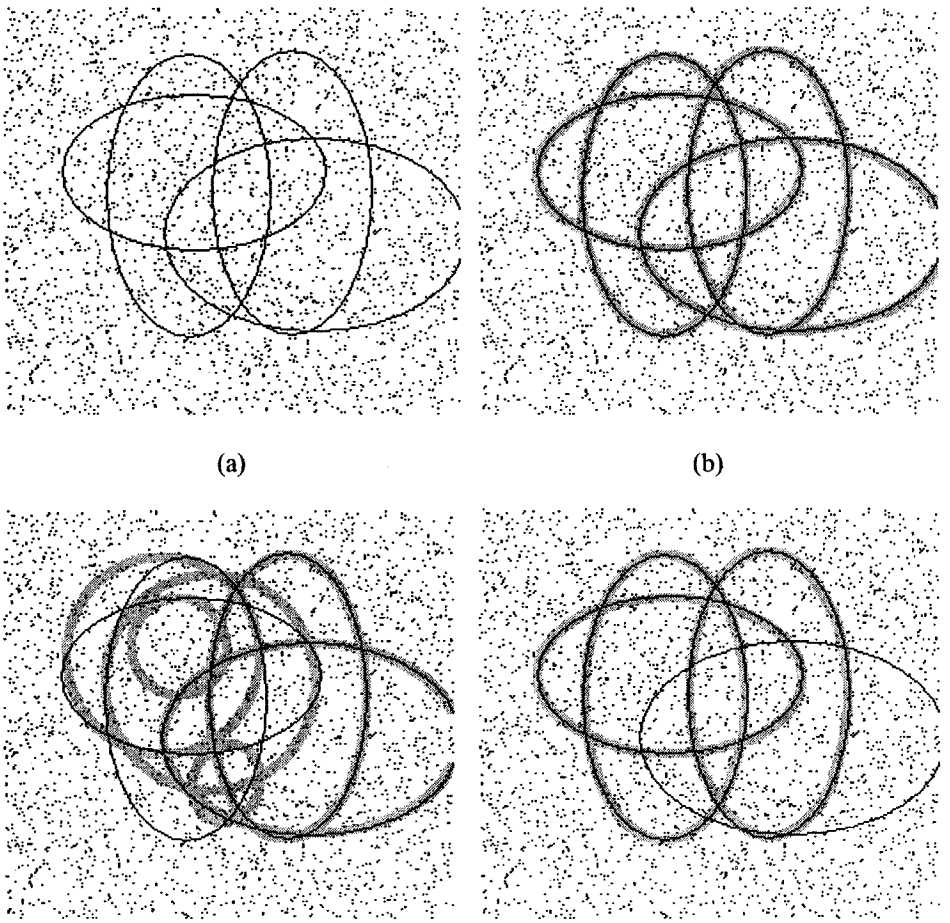


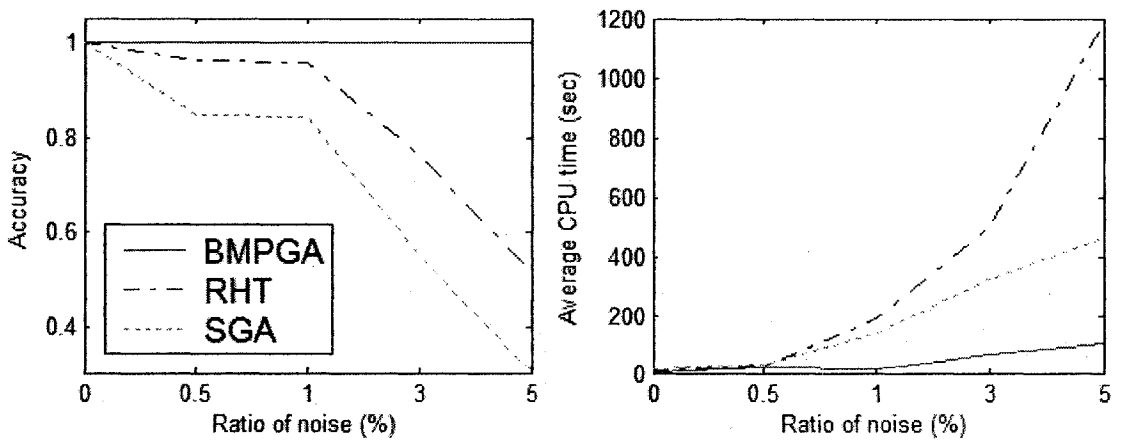
Fig. 5-17 A typical noisy image with %5 salt & pepper noise: detected ellipses overlaid in thick grey
(a) Original image; (b) Ellipses detected by BMPGA;
(c) Ellipses detected by RHT; (d) Ellipses detected by SGA

Fig. 5-17 (a) shows a typical image with 5% pepper-and-salt noise. Table 5-3 lists parameters and fitness values of ellipses found. It can be seen that BMPGA is much faster than both RHT and SGA. RHT is the slowest among the three. Its computation time is two times of that of SGA and more than 40 times of that of BMPGA. It should be also noted that RHT is more liable to false positives (i.e. the detection of non-existing ellipses), as demonstrated in Fig. 5-17 (c).

Table 5-3 Parameter and fitness values for ellipses detected in Fig. 5-17

Algorithms	Center	Major Axis	Minor Axis	Orientation (°)	Fitness	Time (sec)
BMPGA	(103, 104)	78	46	90	0.991525	54.88
	(161, 102)	79	45	89.2954	0.968645	
	(106, 91)	74	43	-0.5067	0.955948	
	(174, 127)	85	53	0.56718	0.847435	
RHT	(161, 103)	79	45	-89.5019	0.933849	2670.24
	(174, 126)	84	53	0	0.82058	
	(117, 95)	60	60	0	0.531464	
	(90, 80)	55	55	0	0.511258	
	(99, 75)	27	27	0	0.417344	
	(126, 151)	22	19	-67.7653	0.338096	
SGA	(103, 104)	79	45	90	0.984774	1267.67
	(161, 102)	79	44	89.3833	0.979715	
	(106, 92)	74	43	0.698993	0.96429	

Fig. 5-18 shows the performance of these algorithms on noisy images. Again, BMPGA shows considerable tolerance to salt-and-pepper noise, while simultaneously operating faster than the other two algorithms.



(a) (b)

Fig. 5-18 Comparing the performance of RHT, SGA and BMPGA on detecting ellipses in noisy images
(a) Accuracy versus ratio of noise; (b) CPU time versus ratio of noise

Many false positives are observed for RHT with high noise, as Fig. 5-19 shows. This fact further fortifies our claim that RHT searches the space and accumulates votes blindly.

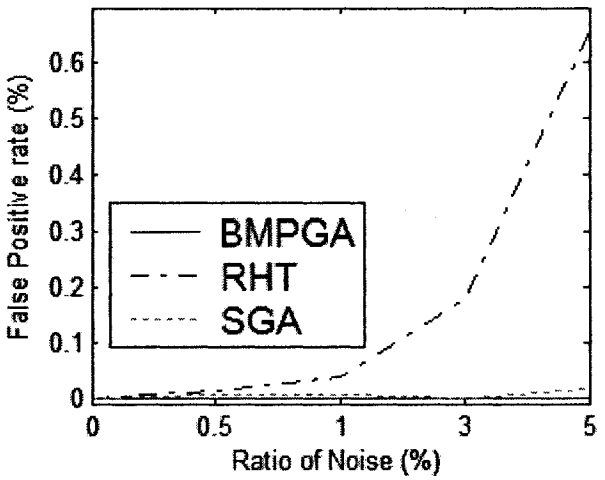


Fig. 5-19 False positives for different noise level

5.4.2 Real World Images

A large, carefully constructed database of synthetic images provides the bases for comprehensive tests, which give specific feedback about potential problems with detection algorithms. Nevertheless, it is real-world images that determine the difference between a genuinely useful and purely academic novel algorithm. Hence, we have amassed three databases of intentionally different types of images: hand-written English letters, microscopic cell images and road signs.

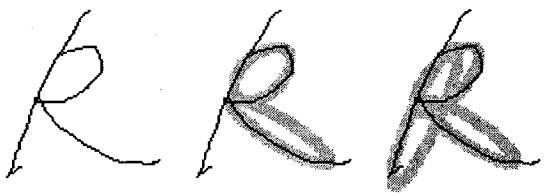
The databases are of varying size, but contain images that were collected, prior to even the design of BMPGA. From each of the databases, 50 images are selected at random and used for assessing the real world performance of BMPGA, RHT, and SGA. On visual inspection, we noticed that these images contain: all kinds of combinations of perfect and deformed ellipses, noise levels, and other geometric shapes (e.g. lines,

polygons).

These images are either colorful or gray-scale. All color images are converted to gray-scale images. The gray-scale images are then subject to morphological noise reduction and canny edge detection. Resulted binary edge images are inputted to the ellipse detection algorithm.

Fig. 5-20 (a) shows a typical handwritten character with elliptical curves detected. SGA failed completely.

In Table 5-4, we can see that BMPGA is slower than RHT. As discussed in Section 5.2, when the image is relatively simple, noise-free and does not have many overlapping patterns, RHT is expected to run faster than BMPGA. However, RHT still suffers from false positives, as Fig. 5-20 (c) shows.



(a) (b) (c)

Fig. 5-20 A typical handwritten character: detected elliptical curve overlaid in thick grey
(a) Original image; (b) Elliptical curves detected by BMPGA;
(c) Elliptical curves detected by RHT

Table 5-4 Parameter values of ellipses detected in Fig. 5-20

Algorithms	Center	Major Axis	Minor Axis	Orientation (°)	Fitness	Time (sec)
MPGA	(75, 52)	17	10	-42.8732	0.871649	8.50
	(84, 81)	29	6	31.5684	0.486901	
RHT	(84, 81)	27	6	32.9764	0.493482	1.08
	(72, 55)	22	9	-46.8089	0.466301	
	(61, 73)	38	7	-65.1084	0.25563	
SGA	n/a					

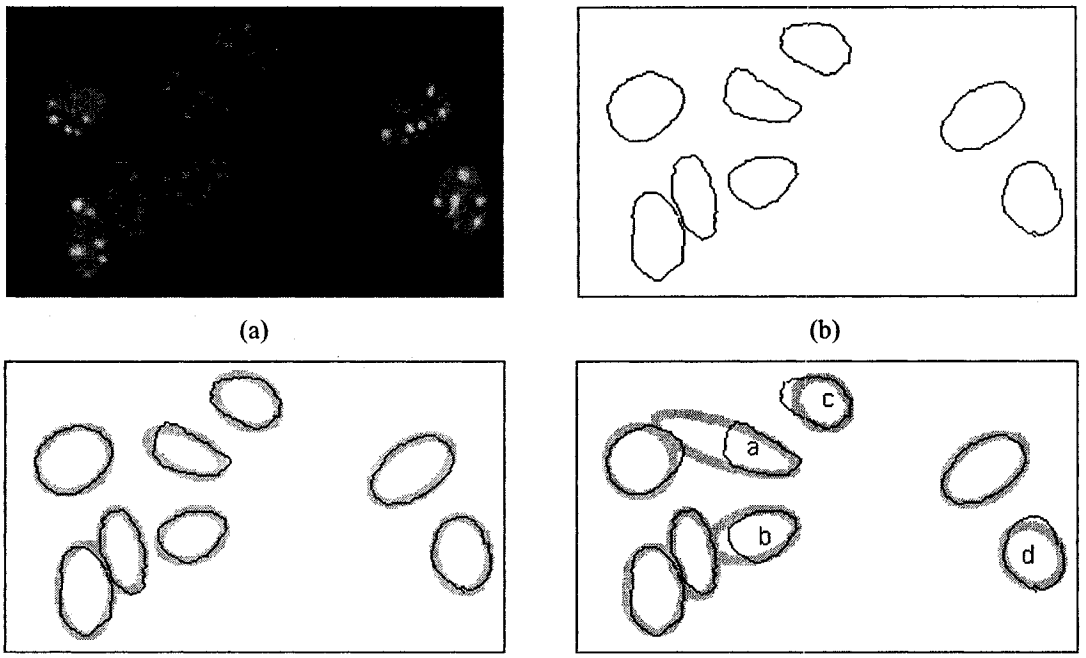


Fig. 5-21 A typical image of cells Taken through a Microscope - SGA failed to detect a single ellipse
(a) Original image; (b) Pre-processed image; (c) Ellipses detected by BMPGA;
(d) Ellipses detected by RHT

Table 5-5 Parameter values of ellipses detected in Fig. 5-21

Algorithms	Center	Major Axis	Minor Axis	Orientation (°)	Fitness	Time (sec)
BMPGA	(70, 110)	24	12	78.7455	0.884863	26.71
	(41, 57)	23	18	-26.4053	0.871432	
	(111, 101)	20	14	-19.6434	0.855267	
	(272, 112)	22	16	82.1506	0.844027	
	(48, 133)	27	16	-89.4256	0.841953	
	(242, 63)	27	15	-33.6307	0.837715	
	(144, 23)	20	14	26.6459	0.734521	
	(107, 53)	24	11	22.4682	0.649748	
RHT	(47, 133)	25	17	87.6509	0.747763	1015.98
	(38, 58)	20	20	0	0.644028	
	(70, 112)	25	13	78.18	0.74799	
	(243, 63)	27	16	33.7181	0.805789	
	(89, 48)	44	12	18.0898	0.34089	
	(147, 24)	17	14	34.4332	0.639102	
	(106, 101)	26	14	-14.8053	0.478279	
	(271, 113)	19	16	48.1021	0.627391	
SGA	n/a					

Fig. 5-21 (a) is a typical microscopic image of cells at 40x (Table 5-5). Again, BMPGA shows its obvious advantage over RHT and SGA in both accuracy and speed. Although RHT approximates all eight cells, it suffers from large misplacement (see cells labeled: a, b, c and d in Fig. 5-21 (d) and long running time; SGA fails to detect even a single ellipse.

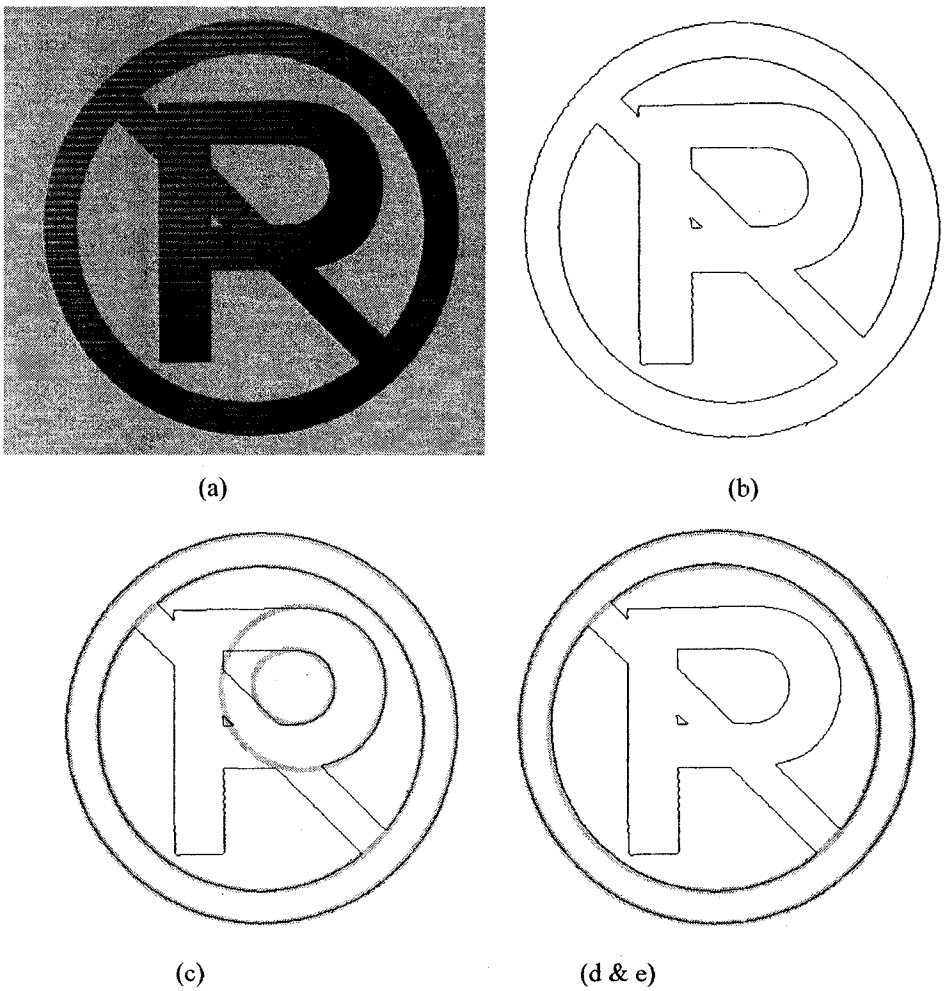


Fig. 5-22 A typical road sign image

(a) Original image; (b) Pre-processed image; (c) Ellipses detected by BMPGA; (d) Ellipses detected by RHT; (e) Ellipses detected by SGA

Fig. 5-22 is a typical image of a road sign (Table 5-6). We can see that neither

RHT nor SGA is able to detect partial ellipses, especially when there are also perfect ellipses in the image. They are generally slower than BMPGA in this kind of complicated images.

Table 5-6 Parameter values of ellipses detected in Fig. 5-22

Algorithms	Center	Major Axis	Minor Axis	Orientation (°)	Fitness	Time (sec)
BMPGA	(224, 200)	188	188	0	0.993062	140.68
	(223, 200)	157	157	0	0.908382	
	(255, 160)	39	36	9.34395	0.654582	
	(264, 161)	79	79	0	0.401485	
RHT	(224, 200)	188	188	0	0.993062	335.08
	(223, 201)	158	156	0	0.908279	
SGA	(224, 200)	188	188	0	0.993062	378.74
	(222, 200)	157	157	0	0.886644	

Table 5-7 gives out the overall performance for real world images. From the table, we can see that SGA is almost totally ineffective with complicated real world images; it returns an average accuracy of less than 20%. RHT suffers from long computation times and high false positive rate. There are some false positives (6.9048%) for BMPGA as well, because the algorithm may sometimes approximate polygons with ellipses. One possible solution is to detect low-dimensional shapes first (e.g., lines) and remove them from the image before detecting circles and ellipses. This way, the process can be made both more efficient and more accurate.

Table 5-7 Performance of BMPGA, SGA, and RHT for real world images

Algorithms	Accuracy (%)	Average cpu time (sec)	False positives (%)
BMPGA	92.761	134.58	6.9048
SGA	15.113	370.39	0
RHT	64.387	809.73	18.633

5.5 Algorithm Discussion

From the experiments in this chapter, we can draw similar conclusions to those presented in Section 4.3 in the following aspects:

- Bi-objective versus single objective: in this application, two objectives are naturally better than one objective since the two objectives are equivalently important. Ignorance of either of them will result in unnecessary loss of promising candidates.
- Local evolution versus global evolution: local evolution in BMPGA generally outperforms global evolution in Sharing since the former exploits every promising area, whereas the later explores the whole search space without focusing on any optima. The consequence is that BMPGA yields better match between found ellipses and template ellipses than Sharing.
- Locally fixed subpopulation size versus globally fixed population size: if the size of a subpopulation is fixed, each promising optimum area can be allocated the same quantity of resources (chromosomes). This is better than fixing the size of the overall population size, which is hard to estimate without any information about the landscape and does not guarantee a balanced distribution of resources among species.
- Sharing: BMPGA is more efficient and accurate than Sharing on the detection of multiple potentially deformed ellipses in noisy images. From the experiments we can see that Sharing showed an overall unstable performance and sometimes may even be outperformed by RHT.

5.6 Conclusions

In this study, BMPGA was thoroughly tested on a large number of synthetic and three types of real world images, and compared to the very widely used RHT and one of the best available GA based ellipse detection technique - SGA. Generally speaking, BMPGA is more efficient and accurate than both RHT and SGA on the detection of multiple, potentially deformed, full and partial ellipses in noisy images. It can be easily extended to detect other geometric primitives, e.g. straight lines, too.

Chapter 6 Application II: Microscopic Cell Segmentation

Proteins, the basic components of all living cells, are made up of different combinations of amino acids, which fold into place to form proteins. Protein conformational disorders such as Alzheimer's disease, Huntington's disease, Parkinson's disease and oculopharyngeal muscular dystrophy are associated with particular proteins that misfold and aggregate to form clumps (or inclusions) in specific tissues. The presence of inclusions in patients' affected tissues has led to the suggestion that protein aggregation is a critical molecular component of such diseases. The cellular toxicity associated with protein aggregates has been suggested to result from the sequestration of essential proteins with a variety of functions, such as transcription, maintenance of cell shape and motility, protein folding and degradation.

Conducting research in protein conformational disorders requires the identification of cell as well as the inclusions within them from microscopic imagery. Cell image segmentation hence becomes a necessary first step of many automated biomedical image processing procedures. Programs that carry out automatic identification and quantification of cells and their sub-cellular structures are immediately useful tools for researchers in the fields of pathology and cytology.

If the counting of cells and/or their sub-cellular structures can be fully automated, this will relieve human experts of quantitative population studies from difficult and time-consuming work and, furthermore, improve the objectivity of the quantitative results [107-109]. Hence, there is an increasing demand for an automatic quantification system

to process the digitized histological images and extract useful information reasonably accurately from the images. Yet, rugged automation of these tasks is far from realization, since cellular images are, on the whole, noisy and non-uniform, and objects on the image often overlap each other or occur in heaps.

In this chapter, we present an algorithm that combines iterative thresholding with ellipse detection into a potent cell segmentation mechanism. The aim of the project goes beyond cell segmentation, into sub-cellular/nuclear object segmentation. The results of the experiments provide strong evidence that the proposed approach can segment circular/elliptical cells from background, effectively and efficiently.

The program (embodying the theoretical methodology) was tested on 98 randomly selected microscopic images, and returned an overall accuracy value of 96.35%.

The challenge facing us may be divided into the following sub-problems: (1) pre-processing of images, in terms of noise elimination/reduction and extraction of foreground objects or 'blobs'; (2) extraction of individual cells from blobs.

Pre-processing is achieved using an algorithm that combines morphological operators with adaptive thresholding in order to simultaneously eliminate noise and locate blob regions, hence compute the best possible local threshold levels for 'blob' extraction.

The main challenge of extracting cells is handled quite differently. We use the approach discussed in Chapter 5 - ellipse detection based on BMPGA to identify elliptical objects within blobs, which are of an acceptable size. This algorithm has proven itself very effective and efficient in detecting the boundaries of circular/ elliptical cells, even if these boundaries were deformed or significantly occluded by other cells, or non-

cellular objects.

6.1 Previous Work

Theoretically, in the case of no or low noise, an appropriate image segmentation algorithm (i.e., object contour detection followed by object classification) could accurately quantify the cellular nuclei in an image. In practice, however, no straightforward attempt is very successful or robust [110]. This is due to two major hurdles facing any simple object segmentation scheme. One is the non-uniformity of image, the other is the cellular occlusion (i.e., cells touching and clustering to form cell heaps).

There are, at least, four categories of methods in the cell image processing literature that deal with the non-uniformity of images. The most conventional method partitions the whole image into smaller (generally square) patches, and assume that these smaller patches are uniform [107, 111]. In many cases, however, the patch itself is not uniform. Also, foreground objects of interest may lie on the edge of a patch or be included in more than one patch.

The second approach multiplies each image by a background control matrix (BCM) [112]. Obviously, a specific BCM can only be optimally suited to a narrow class of images. And, generating an optimized BCM is itself a non-trivial task.

The third method use certain morphological operations (specifically, erosion and dilation) to remove all relevant foreground objects, while maintaining the background, then applies an image subtraction operation to remove the background from the original image [113]. This method is not very accurate in removing foreground objects since the morphological operations may treat some of the background as part of foreground objects,

especially in cases where considerable non-uniformities exist, within and outside these objects.

Hence, the adaptiveness of all the three methods above is limited. Furthermore, their control parameters (such as patch size, BCM coefficients, and size of structuring element) need to be tuned before the methods can be effectively deployed.

The fourth approach covers various hierarchical image processing techniques. These include the two-step segmentation strategy [107, 114], the multi-resolution approaches [115-117] and the recursive/iterative thresholding technique [109].

6.2 Methodology

6.2.1 Overview

The overall operation of the algorithm is presented graphically in Fig. 6-1.

The first phase is blob extraction. It applies a multi-region adaptive thresholding method that is used to segment the objects of interest in the image (blobs), while eliminating the noise. A non-linear function from Otsu [22] is employed. This function includes not only Otsu's original threshold, but also statistical measures of the images including (but not limited to) mean and standard deviation of the pixel values of the whole image. Once the original image is segmented using this global (typically low) threshold, each extracted region of the image (not yet considered a blob) will be further processed using its own *local* threshold values. This process will continue and iterate until the extracted foreground blobs cannot be improved further. The cells in the image are labeled and extracted, in order to provide the working ground for the second phase.

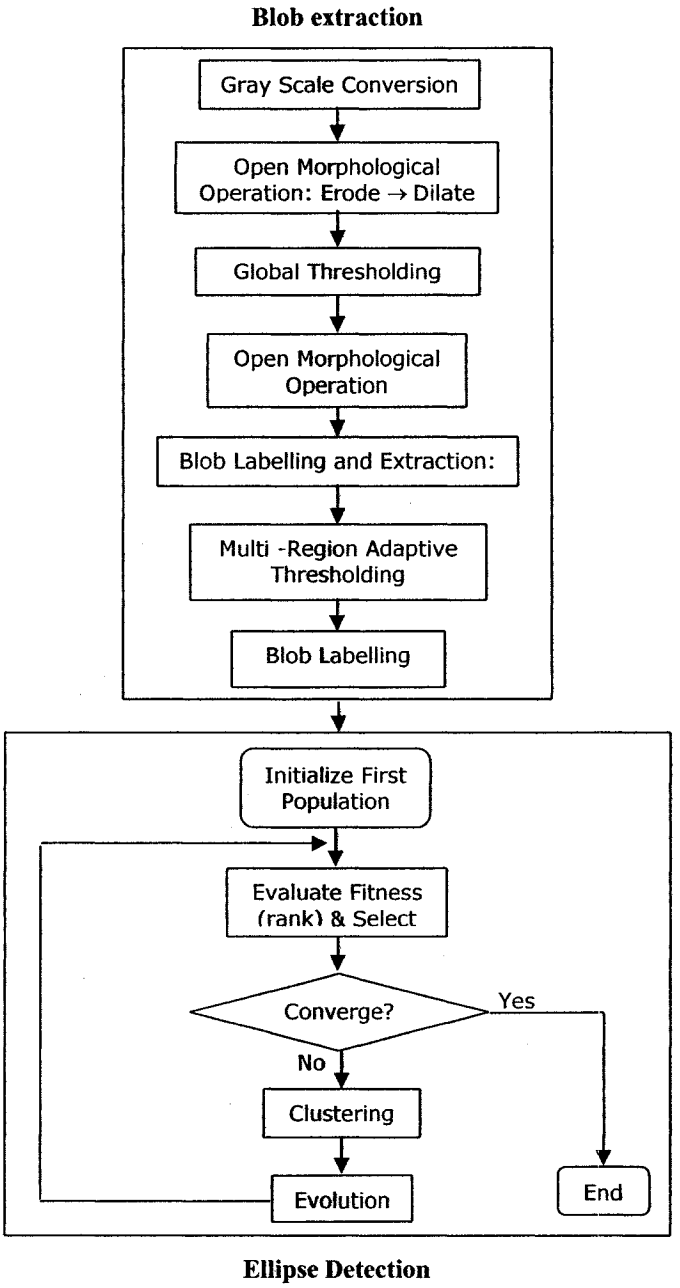


Fig. 6-1 Overview of two-phase algorithm for the segmentation of cells

The second phase generates a single population by creating a number of chromosomes (or candidate ellipses) whose genes are randomly selected from the set of foreground pixels in the image. The population is then ranked in terms of both similarity

and distance and searched for good candidates, if any. If, by chance, all the ellipses in the image are included in the first generation, the program terminates. Otherwise, a clustering technique is used to divide the chromosomes into a number of clusters (or subpopulations). From that moment on, all the subpopulations are evolved in parallel. If one of the subpopulations converges on an optimal or a sub-optimal chromosome, then that whole subpopulation and the corresponding ellipse in the image are removed. This has the positive side effect of accelerating the search process by decreasing the size of the candidate pool. The program, as a whole, terminates when all (full and partial) ellipses are found, or when a preset maximum number of generations are reached.

The final result of both phases is a set of ellipses, most of which identify the various cells in the image. These ellipses are used to extract the cells from the image, which can then be further processed, or simply counted.

6.2.2 Blob Extraction

This phase applies a multi-region adaptive thresholding method to segment the objects of interest (blobs) in the image, while eliminating the noise. The blobs of cells are then extracted, labeled, and fed into the second phase, which identifies cells within these blobs.

Initially, all original color images are converted to grayscale images via:

$$I = 0.299 * R + 0.587 * G + 0.114B \tag{6.1}$$

where I is the grayscale intensity of each pixel, R, G, B are red, green and blue values of each pixel, respectively.

The grayscale image obtained is then de-noised using an open morphological operator [114] with a circular structuring element of radius r, where:

$$r = average\ radius\ of\ blob\ (in\ pixels)/10 \tag{6.2}$$

The gray scale image is further thresholded via a threshold given by:

$$total\_threshold = a * \left(1.3 + (\frac{\mu - \sigma}{256})\right) * OTSU \tag{6.3}$$

where μ is the mean value andσ is the standard deviation of all the RGB values of the pixels, respectively. *OTSU* is a threshold proposed by Otsu [118]. *a* is empirically calculated, and is dependent on the color and intensity of the blobs; its default value is 1.

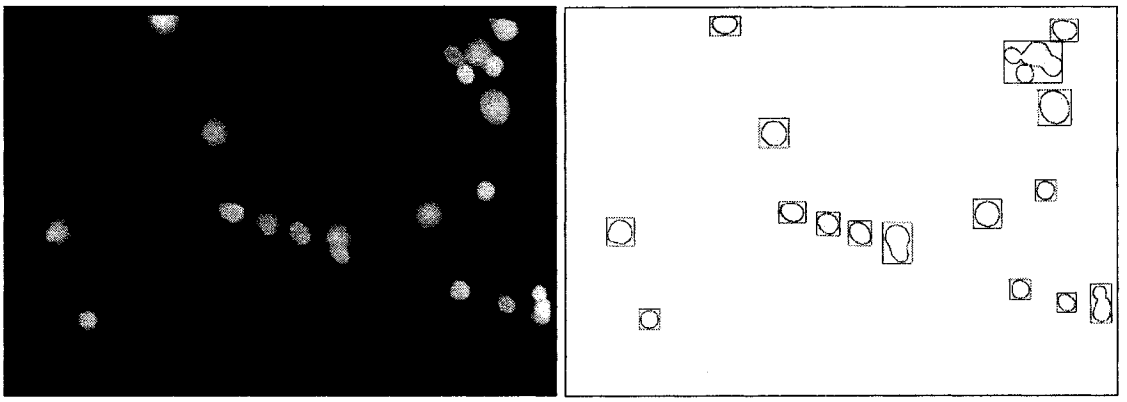
The resulted image is de-noised again using the same open morphological operator. After that a region labeling algorithm denotes each blob with a label using an 8-connected component operator [119]. A blob is identified by placing a minimally-fitting bounding box around it. A multi-region adaptive thresholding method is applied to each box in the original gray scale image with a threshold given by:

$$adaptive\_threshold = b * (\mu + 3 * \sigma) \tag{6.4}$$

where μ is the mean value andσ is the standard deviation of all the RGB values of the pixels within the box, respectively. *b* is empirically calculated, and is dependent on the color and intensity of the blob; its default value is 0.38.

The blobs resulting from this step are labeled again using 8-connected labeling method. The blobs are identified to produce the final image.

Fig. 6-2 shows an example of an original color image and the resulted binary image with bounding boxes. The latter is input to the second phase.



(a) (b)

Fig. 6-2 A example of 10X magnification cell images

(a) Original color image; (b) Thresholded binary image

6.2.3 Cell Extraction

For each bounding box, BMPGA is used to extract all elliptical objects above a certain size, which correspond to the cells in the pre-processed image. Interested readers may refer to Chapter 5 for implementation details.

6.3 Experimental Procedure

Experiments were carried out on a set of images relating to oculopharyngeal muscular dystrophy (OPMD) disease. A collection of 98 images, which contains two classes of cells: viable green cells (Fig. 6-3) and nonviable yellow/orange cells (Fig. 6-4), was obtained under a fluorescence microscope and stored in a database. The percentage of was determined The program used all the images in the database, and was run on an Intel Xeon 2.66 GHz w/ 512 KB of cache, 512 MB DDR RAM and running Red Hat Linux 8.0.3.2-7.

Each image was first analyzed by three biomedical collaborators, whose consensus opinions are reported. They were asked to mark the exact boundaries of the blobs as well as of individual cells. These marked images were used as the ground truth

for future analysis. The ellipses discovered by the program are compared with the actual cells identified by the expert.

Perfect results for phase 1 mean that all blobs identified by the expert are also identified by the program (without any significant amount of over- or under-coverage), and that the program does not identify any other blob. Perfect results for phase 2 mean that every cell marked by the biomedical expert is also identified by an ellipse, and that there are no ellipses identifying more than one cells or non-existent cells.

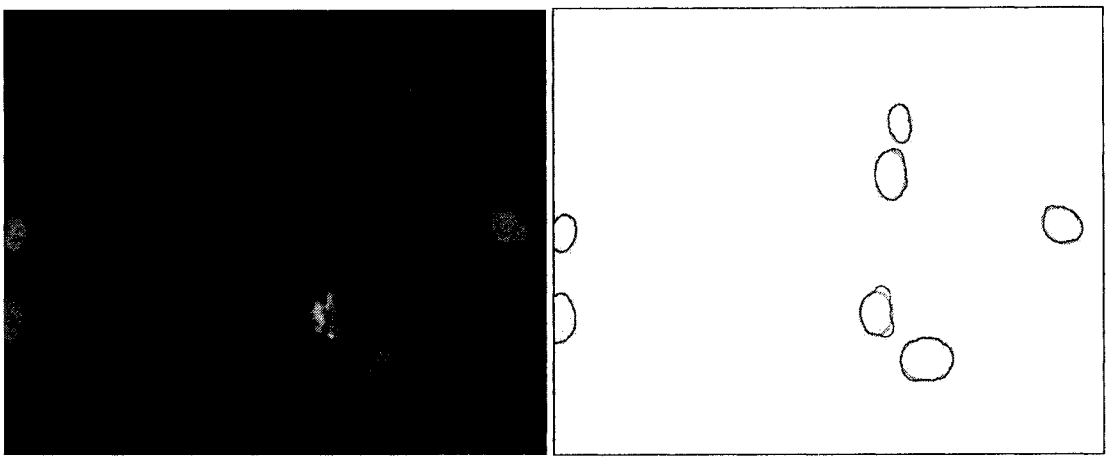
In the following section, we discuss the results of the application, first in some detail, using concrete examples, and then in the form of statistical tables, for each of the two phases of the program.

6.4 Results and Discussion

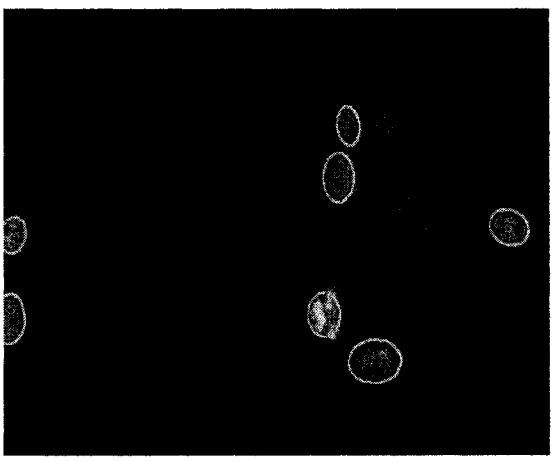
We present the results of the experiments in visual as well as tabular form. We present examples of successful application, as well as examples of some situations that caused erroneous results. However, it is worth noting that the program achieved an overall final rate of accuracy > 96%, and that the average time for processing an image (over all images) was 52.33 seconds.

6.4.1 Visual Results

Fig. 6-3 and Fig. 6-4 show two typical scenes of detection. Fig. 6-3 and Fig. 6-4 (a) are original images. Fig. 6-3 and Fig. 6-4 (b) are extracted outlines with detected ellipses imposed on it. Fig. 6-3 and Fig. 6-4 (c) show the detected ellipses imposed on the original images.



(a) (b)



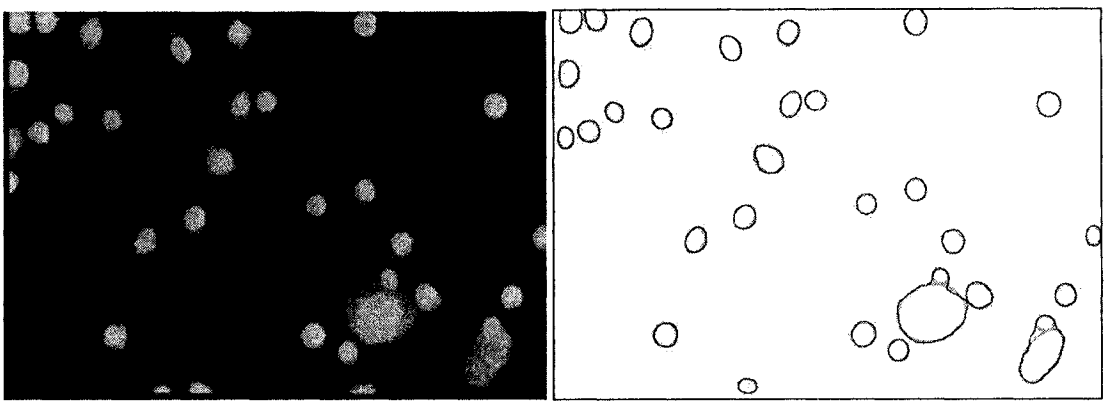
(c)

Fig. 6-3 Segmentation results I

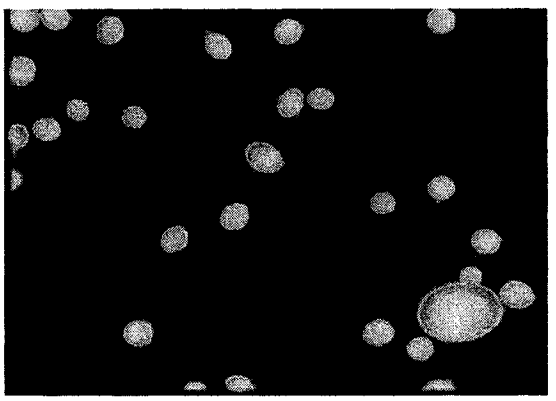
(a) Original color image; (b) Detected ellipses imposed on binary image;

(c) Detected ellipses imposed on original image

In Fig. 6-3, partial cells are successfully approximated with ellipses. In Fig. 6-4, on the other hand, ellipses are also successfully approximated on overlapping/occluded cells. More examples of successfully approximating partial cells are shown in Fig. 6-5.



(a) (b)

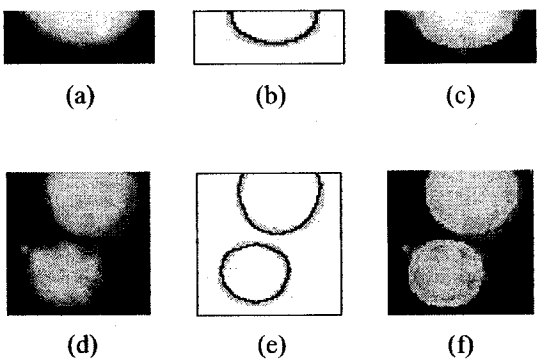


(c)

Fig. 6-4 Segmentation results II

(a) Original color image; (b) Detected ellipses imposed on binary image;

(c) Detected ellipses imposed on original image



(a) (b) (c)

(d) (e) (f)

Fig. 6-5 Segmentation results of partially visible cells

(a), (d) Original image; (b), (e) Detected ellipse imposed on binary image

(c), (f) Detected ellipses imposed on original image

Fig. 6-6 and Fig. 6-7 further show the success of BMPGA in this application. The

performance of BMPGA is compared against that of RHT. Particularly, the success of BMPGA in separating overlapped/occluded cells has been shown in Fig. 6-6.

In Fig. 6-6, it can be also seen that RHT tends to ignore partial/small candidates that typically have worse fitness.

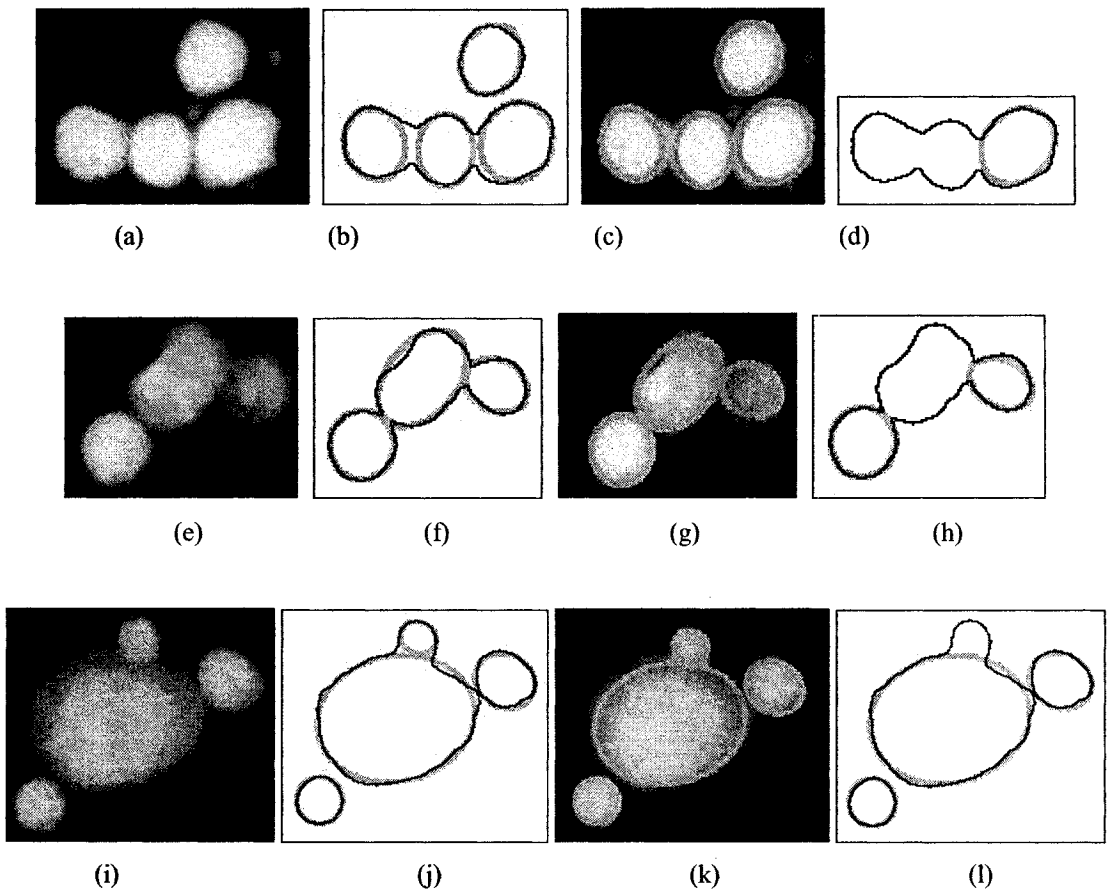
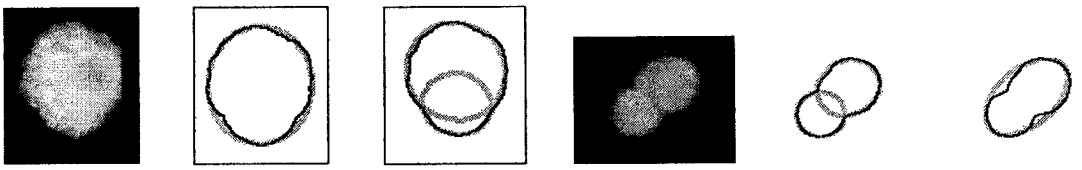


Fig. 6-6 Successful segmentation of overlapping cells by BMPGA and undetected cells by RHT
(a), (e), (i) Original image; (b), (f), (j) Detected ellipse by BMPGA imposed on binary image;
(c), (g), (k) Detected ellipses by BMPGA imposed on original image;
(d), (h), (l) Detected ellipse by RHT imposed on binary image

Fig. 6-7 further shows a typical example of over-detection via RHT.



(a) (b) (c) (d) (e) (f)

Fig. 6-7 Comparison of BMPGA and RHT

(a), (d) Original image; (b), (e) Ellipse extracted by BMPGA; (c), (f) Ellipse extracted by RHT

Recall Fig. 5-21 in Chapter 5 that the ellipses obtained via BMPGA maximally approximate the contours of the cells, whereas the ellipses obtained via RHT are much coarser than the former. Therefore in these experiments, again, BMPGA is shown to be superior to RHT. As already discussed in Chapter 5, besides its low efficiency, RHT also suffers from the following drawbacks in terms of the quality of its results:

- Ignorance of weak/small candidates;
- Misplacement of ellipses;
- Over-detection of non-existent ellipses.

However, it does not mean that our system is perfect. The experiments also present some problems, some of which are demonstrated in Fig. 6-8. In Fig. 6-8 (b) and (c), a cell, denoted as A in Fig. 6-8 (a), is missing due to limited outline extraction. A possible solution could be better extraction of blobs from the original image, so that the outlines are as close as possible to the originals. As Fig. 6-8 (a) shows, if the dark gap between blobs A and B is clearly identified, i.e., they are separated as two independent blobs, the outline of cell A will not be ignored as noise.

Fig. 6-8 (e) and (f) show a similar example. If the gap A in Fig. 6-8 (d) is clearly identified so that the relative completeness of the boundary of cells B and C is maintained, it is highly possible to separate them as two ellipses.

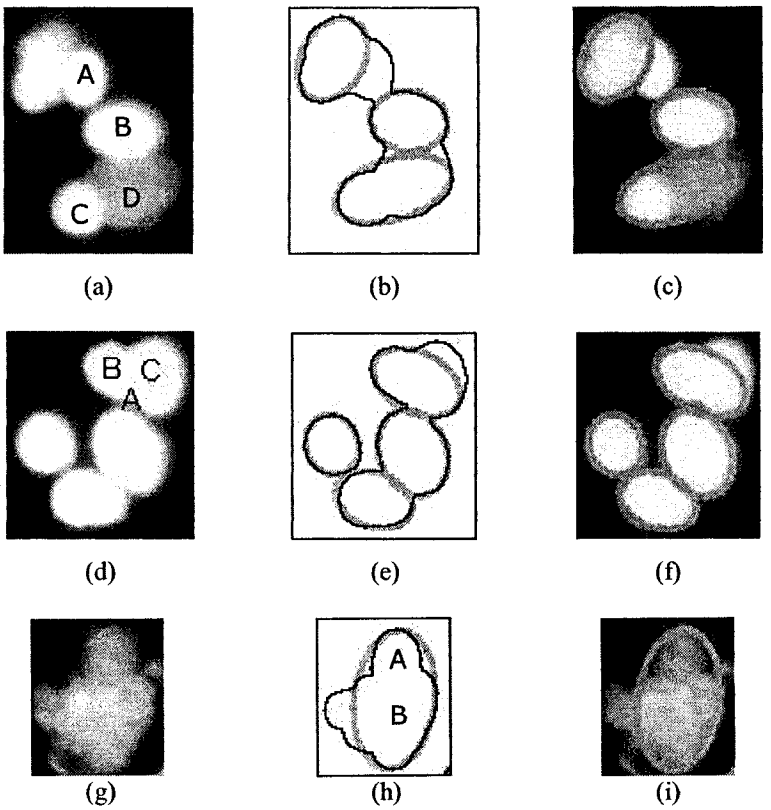


Fig. 6-8 Incorrect segmentation of cells

Fig. 6-8 (b) and (c) also shows two cells identified (C and D in Fig. 6-8 (a)) as one ellipse. As a matter of fact, it is impossible to identify cell D as an independent cell because its outline does not look, remotely, as an ellipse - complete or partial. A possible solution for this kind of problem is to differentiate different cells according to color, and not just shape - cell D is orange in color while the others are mostly yellow.

Similarly, in Fig. 6-8 (h) and (i), if the outlines of cell A and cell B can be separated using their color information, they should be able to be identified as two ellipses rather than one.

Fig. 6-9 shows another case of failed detection due to the mismatch between the extracted and the original outlines. In Fig. 6-9 (a) and (b), the shape of the boundary is

affected negatively by the shade within the cell. However, if the relatively bright area on the right side of the cell could be detected, the original outline of the cell would be maintained and thus a correct ellipse would fit it.

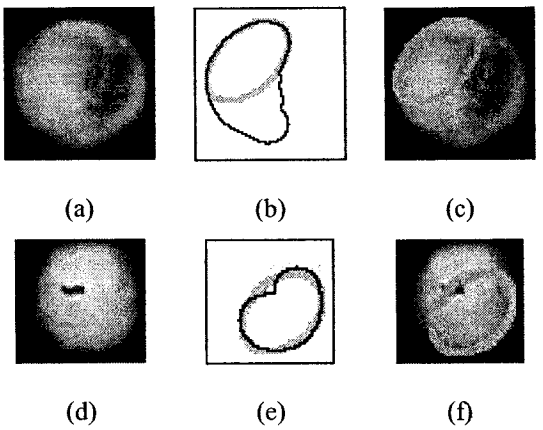


Fig. 6-9 Mismatch between extracted and original cells

In Fig. 6-9 (d) and (e), a small black area within the cell affects the boundary of the whole cell. If all bright pixels in the cells are identified correctly, this will not be a problem.

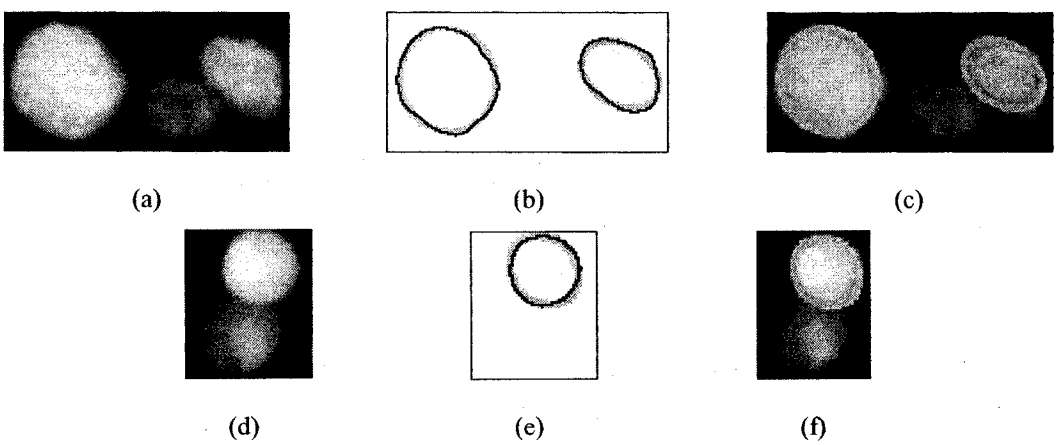


Fig. 6-10 Missed cells due to local thresholding effects

Fig. 6-10 shows missing cells due to their relatively low brightness. Only highly visible blobs are segmented in phase I. This shows that our algorithm needs better local adaptation of the threshold.

In summary, we have very few problems, but they prevent perfectly accurate segmentation of cells. It is possible, however, to remedy the situation by implementing the following embellishments:

- Stricter identification of the boundaries of blobs. This demands careful choice of both global and local thresholds.
- Inclusion of color information in the criteria used to segment cells from blobs (in addition to shape information).
- Reducing the effect of shading within blobs/cells: these can be filtered out/ blurred a little using a standard Butterworth filter.

6.4.2 Statistical Analysis

To summarize the statistical results, we devised a number of specific measures. Accuracy is the ratio of the number of correctly detected cells to the total number of cells actually present in the target image. The false negative rate (%fn) is the percentage of cells missed by the program. As such accuracy added to %fn always comes to 100%. The false positive rate (%fp) is the percentage of non-existent cells.

Table 6-1 Performance of BMPGA and RHT on Cell Clumps

Algorithms	**BMPGA**			**RHT**		
Images	Accuracy	%fn	%fp	Accuracy	%fn	%fp
Class A	96.78	3.22	4.14	70.12	29.88	12.13
Class B	87.93	12.07	15.56	55.19	44.81	25.69
Overall	94.33	5.67	8.21	66.15	33.85	15.78

In Table 6-1 and Table 6-2, Class A is the nonviable cells, and Class B is the viable cells. Table 6-1 shows the performance of our GA and RHT on cells clumps, on

the aspects of accuracy, fp and fn rate. Table 6-2 shows the overall performance, i.e., the average accuracy, fp as well as fn rate we have achieved.

Table 6-2 Overall Performance of Cell Segmentation

Images	Accuracy	%fn	%fp
Class A	98.11	1.89	3.34
Class B	90.93	9.07	13.76
Overall	96.35	3.65	5.24

6.5 Conclusion

We apply a new Bi-objective Multi-population GA (BMPGA) to the problem of elliptical/circular cell segmentation, with a fair measure of success. The overall algorithm is implemented in two phases, Phase 1 uses iterated thresholding to identify and mark foreground objects or 'blobs' with an overall accuracy of 97%. Phase 2 of the method uses a BMPGA to identify cells quickly and reliably. Our empirical evaluation on a sample of 98 images returned an accuracy ranging between 90% and 98%, with an overall false negative less than 14% and an overall false positive less than 10%. It is worth noting here that this ellipse detection technique has worked well despite the imperfect nature of the perimeter of cells (e.g. deformations, missing sections and overlapping). As such it is our belief that this ellipse detection algorithm is not only reasonably fast but is also quite tolerant to imperfections widely observed in imagery of cells and other natural (as opposed to engineered) patterns. The method holds promise for other biomedical image processing applications, in which elliptical patterns are sought, but validating its general value is left for future investigation.

Chapter 7 Conclusions and Future Work

7.1 Conclusions

This thesis studies GAs for multimodal optimization problems. A novel population based approach – the Bi-objective Multi-population Genetic Algorithm (BMPGA) is proposed. It is different from other genetic operator based GAs and population based GAs in that it offers:

- An bi-objective scheme: a new more diversity-preserving method of fitness evaluation;
- Locally-focused evolution of the sub-populations with full elitism.
- Recursive Middling: better allocation of resources (i.e. chromosomes) to the various neighborhoods of potential optima through a novel clustering algorithm;

BMPGA is applied in different application areas and compared to other leading algorithms in corresponding fields.

In optimization of benchmark multimodal functions, BMPGA is compared to MNGA [1], DNC [2], CLEARING [3] and MPGA. DNC is a leading Sharing GA that is able to identify niches of variable radii. CLEARING is a genetic operator based GA that has shown some advantages over Sharing GAs. MNGA is a population based GA, whose Hill-Valley clustering method is interesting and is used in many peer's works. Finally,

MPGA is a variant of BMPGA with one objective.

BMPGA outperforms DNC, CLEARING and MNGA in terms of its overall effectiveness in finding and preserving optima, general applicability on different landscapes, reliability on specific parameters, as well as consistency of the results. It also outperforms MPGA in overall success rates and average number of optima found. It is less dependent on varying population size than MPGA, too.

In the detection of multiple perfect/imperfect elliptical curves in noisy images, BMPGA is compared to RHT – the most popular geometric extraction algorithm and Sharing GA – the best available GA based ellipse detection method. BMPGA demonstrates more efficiency, accuracy and tolerance to noise than both RHT and SGA in a test of a large number of synthetic and real world images.

BMPGA is also extended to automatically extract cells from microscopic imagery. The mechanism, as a whole, has an accuracy rate 96% and takes <1 min (given our specific hardware configuration) to operate on a microscopic image.

7.2 Future Work

BMGA is arguably a better choice for those who aim to use GAs for multimodal optimization problems. However, that does not mean there is no room for improvement. Possible future work can be roughly categorized into *algorithm* side and *application* side:

1. Algorithm
    i. Clustering:

In BMPGA, clustering is implemented at each generation. Initially, migration of individuals and merging of species occur frequently. However, these actions decrease dramatically when the population reaches the equilibrium with stable species. At this

time, frequent clustering operations will incur a lot of unnecessary costs. Hence performing clustering at each interval, the duration of which increases gradually, rather than at each generation, may lead to better results. Similar approach is seen in Hocaoglu and Sanderson's work [66].

ii. Interbreeding:

As already discussed in Section 4.3.2, interbreeding between different species may induce more diversity into the population. This is fairly important when each cluster in the population has been saturated with individuals from the same species.

2. Application

i. Ellipse geometry:

Section 5.1.1 introduces many approaches that make use of the ellipse's geometric properties to reduce the cost of the algorithm. They can also be used in BMPGA. More efficiency is expected.

ii. Edge detection:

Replace the current pre-processing module with one that performs edge detection more critically, provides specific features, and allows the main ellipse detection step to execute faster than our current version (i.e. segment a 100-object field in 1 second or less on a 1-Xeon Linux PC)

iii. Interface:

Build a graphical user interface that makes it easy for end-users to install, learn, use, and upgrade the software; also, build a different interface, which would allow our software to be used as a licensed component (a DLL) by other companies as part of their own products;

iv. Software:

Complete a software package called *Ellipzoid 1.0*, which detects elliptic curves (including all ellipses and arcs) in binary images, even when those curves are imperfect, incomplete, or riven by noise. Thoroughly test the new version of the software for accuracy, speed, usability, reliability, maintainability, and other relevant software qualities.

v. Patent:

Apply for a patent for the ellipse detection software.

vi. Other applications:

Finally, it would be interesting to extend BMPGA to other application areas of practical significance.

Appendix A Benchmark Functions

Function D_1

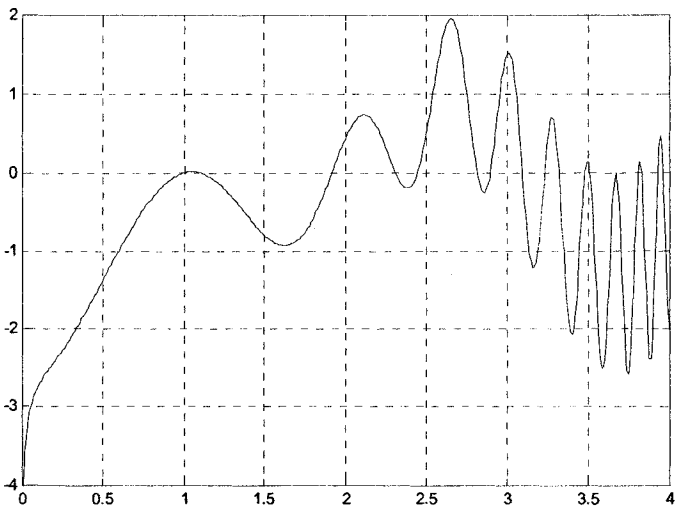


Fig. A–1 Multimodal function D_1

$$D_1(x) = \ln(x) \cdot (\sin(e^x) + \sin(3x)), \quad x \in (0,4] \tag{7.1}$$

Function D_1 is a typical one-dimensional multimodal function. It features an uneven distribution of peaks of varied height.

Function D_2

$$D_2(x) = \frac{1}{\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{2} (x_j - a_{ij})^6}} \qquad x_j \in [-65.536,\ 65.536] \tag{7.2}$$

D_2 is a two-dimensional multimodal function. Its optima are distributed evenly in a 5 X 5 square but the contrast between their altitudes is huge.

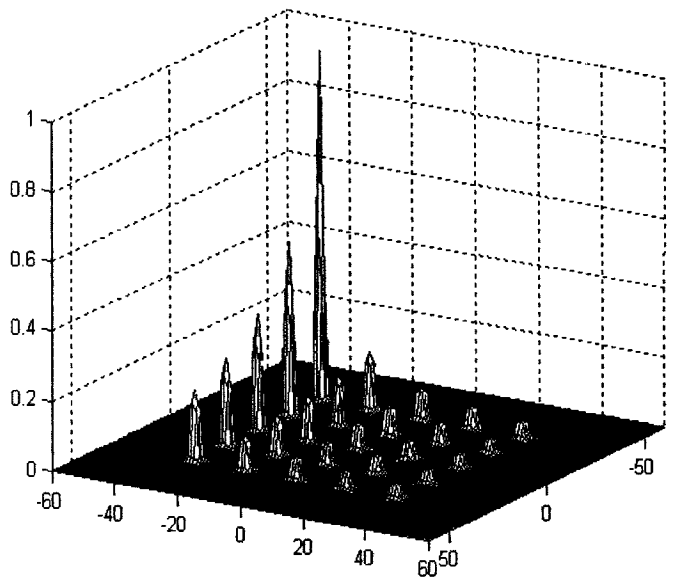


Fig. A–2 Multimodal function D_2

Function D_3

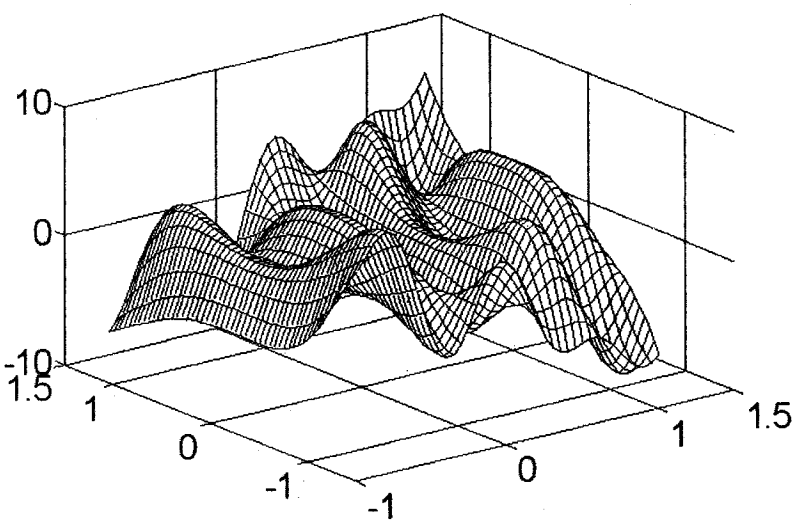


Fig. A–3 Multimodal function D_3 in 3D space

$$D_3(x,y) = (0.3x)^3 - (y^2 - 4.5y^2)xy$$
$$-4.7cos(3x - y^2(2+x))sin(2.5\pi x) \qquad (7.3)$$
$$x \in [-0.9, 1.2], \quad y \in [-1.2, 1.2]$$

The two-dimensional function D_3 is a similar to D_1 in that its optima are of different volume and apex value and are non-equidistant to each other.

Function D_4

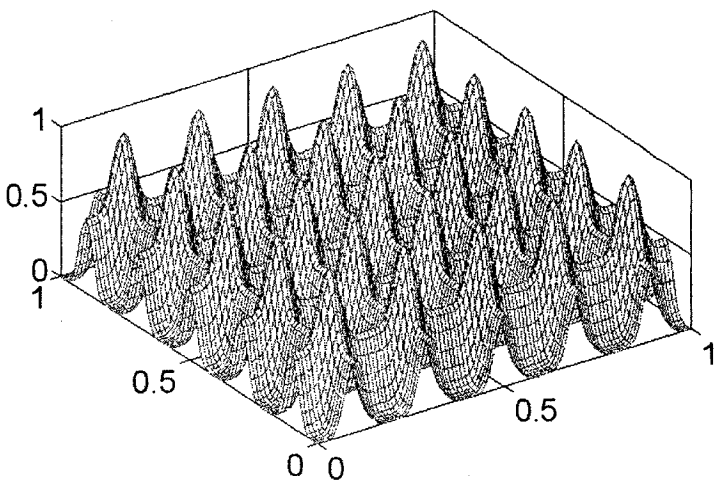


Fig. A–4 Multimodal function D_4

$$D_4(\bar{x}) = 1 - \frac{1}{n}\sum_{i=1}^{n}(1 - \sin^6(5\pi x_i)) \quad x_i \in [0,1] \tag{7.4}$$

The optima of D_4 are equidistant to each other and are of the same volume and altitudes. Its dimensionality can be extended to higher than 3.

References

[1] Ursem, R.K., *Multinational evolutionary algorithms. IEEE Congress on Evolutionary Computation*, pp. 1633-1640, 1999.

[2] Gan, J. and Warwick, K., *Dynamic niche clustering: a fuzzy variable radius niching technique for multimodal optimization in GAs. IEEE Congress on Evolutionary Computation*, pp. 215-222, 2001. Seoul, Korea.

[3] Singh, G. and Deb, K., *Comparison of multi-modal optimization algorithms based on evolutionary algorithms. the 8th Annual Conference on Genetic and Evolutionary Computation*, pp. 1305 - 1312, 2006. Seattle.

[4] Xu, L., Oja, E., and Kultanen, P., *A new curve detection method: randomized hough transform (RHT).* Pattern Recognition Letters, 1990. 11(5): pp. 331-338.

[5] Lutton, E. and Martinez, P., *A genetic algorithm for the detection of 2D geometric primitives in images. the 12th International Conference on Pattern Recognition*, pp. 526-528, 1994.

[6] Yao, J., Kharma, N., and Grogono, P., *A multi-population genetic algorithm for robust and fast ellipse detection.* Pattern Analysis & Applications, 2005. 8(1-2): pp. 149-162.

[7] Marcelli, A. and Stefano, C.D., *Evolutionary algorithms for pattern recognition*, in *The 18th International Conference on Pattern Recognition.* 2006: Hong Kong.

[8] Wang, Y.-K. and Fan, K.-C., *Applying genetic algorithms on pattern recognition: an analysis and survey. the 13th International Conference on Pattern Recognition*, pp. 740-744, 1996.

[9] Cedeño, W., Vemuri, V., and Slezak, T., *Multi-Niche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments.* Evolutionary Computation, 1995. 2(4): pp. 321-345.

[10] Cedeño, W., *The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications*, in *Computer Science* 1995, University of California: Davis

[11] Hocaoglu, C. and Sanderson, A.C., *Multimodal function optimization using minimal representation size clustering and its application to planning multi-paths.* Evolutionary Computation, 1997. 5(1): pp. 81-104.

[12] Hocaoglu, C. and Sanderson, A.C., *Planning multiple paths with evolutionary speciation.* IEEE Transactions on Evolutionary Computation, 2001. 5(3): pp. 169-191.

[13] Pérez, E., Herrera, F., and Hernández, C., *Finding multiple solutions in job shop scheduling by niching genetic algorithms.* Journal of Intelligent Manufacturing, 2003. 14(3-4): pp. 323-339.

[14] Daoud, M.I. and Kharma, N., *An efficient genetic algorithm for task scheduling in heterogeneous distributed computing systems*, in *IEEE Congress on Evolutionary Computation*. 2006: Vancouver. pp. 3258-3265

[15] Liang, Y., Lueng, K., and Mok, T.S., *Automating the drug scheduling with different toxicity clearance in cancer chemotherapy via evolutionary computation. the 8th Annual Conference on Genetic and Evolutionary Computation*, pp. 1705 -1712 2006. Seattle.

[16] Charbonneau, P., *An introduction to genetic algorithms for numerical optimization.* 2002, NCAR.

[17] Rich, E. and Knight, K., *Artificial Intelligence.* second ed. 1991, New York: McGraw-Hill.

[18] Mahfoud, S.W., *A comparison of parallel and sequential niching methods. the Sixth International Conference on Genetic Algorithms*, pp. 136- 143, 1995.

[19] Beasley, D., Bull, D.R., and Martin, R.R.A., *Sequential niche technique for multimodal function optimization.* Evolutionary Computation, 1993. 1(2): pp. 101-125.

[20] Goldberg, D.E. and Richardson, J., *Genetic algorithms with sharing for multimodal function optimization. 2nd Int. Conference on Genetic Algorithms*, pp. 41-49, 1987.

[21] Li, J., Balazs, M.E., Parks, G.T., and Clarkson, P.J., *A species conserving genetic algorithm for multimodal function optimization.* Evolutionary Computation, 2002. 10(3): pp. 207-234.

[22] Lin, C.-Y. and Wu, W.-H., *Niche identification techniques in multimodal genetic search with sharing scheme.* Advances in Engineering Software, 2002. 33(11-12): pp. 779 - 791.

[23] Lung, R.L., *A subpopulation stability based evolutionary technique for multimodal optimization. GECCO*, 2004.

[24] Miller, B.L. and Shaw, M.J., *Genetic algorithms with dynamic niche sharing for multimodal function optimization. IEEE International Conference on Evolutionary Computation*, pp. 786-791, 1996.

[25] Siarry, P., Pétrowski, A., and Bessaou, M., *A multipopulation genetic algorithm aimed at multimodal optimization.* Advances in Engineering Software, 2002. 33(4): pp. 207-213.

[26] Streichert, F., Stein, G., Ulmer, H., and Zell, A.A., *Clustering based niching EA for multimodal search spaces. 6th International Conference on Artificial Evolution*, pp. 293-304, 2003.

[27] Tsutsui, S., Fujimoto, Y., and Ghosh, A.R., *Forking genetic algorithms: GAs with search space division schemes.* Evolutionary Computation, 1997. 5(1): pp. 61 - 80.

[28] Warwick, K. and Gan, J., *A genetic algorithm with dynamic niche clustering for multimodal function optimisation. Int. Conference on Artificial Neural Nets and Genetic Algorithms*, pp. 248-255, 1999.

[29] Yao, J., Kharma, N., and Grogono, P., *BMPGA: a bi-objective multi-population genetic algorithm for multi-modal function optimization. IEEE Congress on Evolutionary Computation*, pp. 816 - 823, 2005.

[30] Yin, X. and Germay, N., *Improving genetic algorithms with sharing through cluster analysis. the 5th International Conference on Genetic Algorithms*, pp. 100 - 101, 1993.

[31] Ackley, D.H., *An empirical study of bit vector function optimization*, in *Genetic Algorithms and Simulated Annealing*, L. Davis, Editor. 1987, Morgan Kaufmann: Los Altos, CA. pp. 170-204.

[32] Forrest, S. and Mitchell, M., *What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation.* Machine Learning, 1993. 13: pp. 285-319.

[33] Mitchell, M., Holland, J.H., and Forrest, S., *When will a genetic algorithm outperform hill climbing?*, in *Advances in Neural Information Processing Systems*, J.D. Cowan, G. Tesauro, and J. Alspector, Editors. 1994, Morgan Kaufmann: San Mateo, CA. pp. 51-58.

[34] Davis, L., *Bit-Climbing, representational bias, and test suite design. the Fourth International Conference on Genetic Algorithms*, pp. 18 - 23, 1991.

[35] Corne, D., *Stepping stones and hidden haystacks: When a genetic algorithm defeats a hillclimber. IEEE International Conference on Evolutionary Computation*, pp. 139 - 142, 1997.

[36] Holland, J.H., *Adaptation in Natural and Artificial Systems*. 1975, Ann Arbor, MI: University of Michigan Press.

[37] Jong, K.A.D., *An analysis of the behavior of a class of genetic adaptive systems*. 1975, University of Michigan: Ann Arbor.

[38] Baker, J.E., *Reducing bias and inefficiency in the selection algorithm. the 2nd Int. Conf. on Genetic Algorithms*, pp. 14-21, 1987.

[39] Eshelman, L.J., Caruana, R.A., and Schaffer, J.D., *Biases in the crossover landscape. the Third International Conference on Genetic Algorithms*, pp. 10 - 19, 1989.

[40] Goldberg, D.E., *Genetic Algorithms in Search, Optiniization, and Machine Learning*. 1989, Reading, MA: Addison-Wesley.

[41] Andre, D. and Koza, J.R., *A parallel implementation of genetic programming that achieves super-linear performance. Int. Conf. Parallel and Distributed Processing Techniques and Applications*, pp. 421-428, 1996.

[42] Cioppa, A.D., Stefano, C.D., and Marcelli, A., *On the role of population size and niche radius in fitness sharing.* IEEE Transactions on Evolutionary Computation, 2004. 8(6): pp. 580-592.

[43] Sareni, B. and Krahenbuhl, L., *Fitness sharing and niching methods revisited.* IEEE Transactions on Evolutionary Computation, 1998. 2(3): pp. 97 - 106.

[44] Mahfoud, S.W., *Population sizing for sharing methods*, in *Foundations of Genetic Algorithms 3*, M. Kaufmann, Editor. 1994: San Francisco.

[45] Pétrowski, A., *A clearing procedure as a niching method for genetic algorithms. IEEE International Conference on Evolutionary Computation*, pp. 798-803, 1996.

[46] Lung, R.I. and Dumitrescu, D., *A new subpopulation model for evolutionary multimodal optimization. Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 4, 2005.

[47] Cioppa, A.D., Stefano, C.D., and Marcelli, A., *Where are the niches? dynamic fitness sharing.* IEEE Transactions on Evolutionary Computation, 2006. In press.

[48] Deb, K. and Goldberg, D.E., *An investigation of niche and species formation in genetic function optimization. Third International Conference on Genetic Algorithms*, pp. 42-50, 1989.

[49] Dick, G., *A comparison of localised and global niching methods. The 17th Annual Colloquium of the Spatial Information Research Centre*, pp. 91 - 101, 2005. Dunedin, New Zealand.

[50] Dilettoso, E. and Salerno, N., *A self-adaptive niching genetic algorithm for multimodal optimization of electromagnetic devices.* IEEE Transactions on Magnetics, 2006. 42(4): pp. 1203-1206.

[51] Dumitrescu, D., *Genetic chromodynamics.* Ser: Informatica, 2000. 45(1): pp. 39-50.

[52] Hanagandi, V. and Nikolaou, M., *A hybrid approach to global optimization using a clustering algorithm in a genetic search framework.* Computers Chem. Engng, 1998. 22(12): pp. 1913-1925.

[53] He, X., Liu, Z.-M., Wang, W.-K., Zhou, J.-L., Li, J., and Zhou, Z.-Y., *A novel genetic algorithm based on multi-species. International Conference on Machine Learning and Cybernetics*, pp. 1344 - 1348, 2002.

[54] Li, J., Balazs, M.E., Parks, G.T., and Clarkson, P.J., *A species conserving genetic algorithm for multimodal function optimization.* Evolutionary Computation, 2002. 10(3): pp. 207-234.

[55] Ling, Q., Wu, G., Yang, Z., and Wang, Q., *Crowding clustering genetic algorithm for multimodal function optimization.* Applied Soft Computing, 2008. 8: pp. 88-95.

[56] Kharma, N., Moghnieh, H., Yao, J., Guo, Y.P., Abu-Baker, A., Laganiere, J., Rouleau, G., and Cheriet, M., *Automatic segmentation of cells from microscopic imagery using ellipse detection.* Image Processing, IET, 2007. 1(1): pp. 39 - 47.

[57] Mahfoud, S.W., *Crowding and preselection revisited,* in *Parallel Problem Solving for Nature.* 1992, Elsevier Science. pp. 27-36.

[58] Thomsen, R., *Multimodal optimization using crowding-based differential evolution. Congress on Evolutionary Computation*, pp. 1382 - 1389, 2004.

[59] Zhang, G., Yu, L., Shao, Q., and Feng, Y., *A clustering based GA for multimodal optimization in uneven search space. the 6th World Congress on Intelligent Control and Automation*, pp. 3134 - 3138, 2006.

[60] Sander, J., Ester, M., Kriegel, H.-P., and Xu, X.W., *Density-based clustering in spatial databases, the algorithm GDBSCAN and its applications.* Data Mining and Knowledge Discovery, 1998. 2(2): pp. 169-194.

[61] Song, S. and Yu, X., *Multi-peak function optimization using a hierarchical clustering based genetic algorithm. the Sixth International Conference on Intelligent Systems Design and Applications*, pp. 425 - 428, 2006.

[62] Stoean, C., Preuss, M., Gorunescu, R., and Dumitrescu, D., *Elitist generational genetic chromodynamics - a new radii-based evolutionary algorithm for multimodal optimization. IEEE Congress on Evolutionary Computation*, pp. 1839- 1846, 2005.

[63] Petrowski, A. and Genet, M.G., *A classification tree for speciation. Congress on Evolutionary Computation*, pp. 204 - 211, 1999.

[64] Yang, H.-Z., Li, F.-C., and Wang, C.-M., *A density clustering based niching genetic algorithm for multimodal optimization. the Fourth International Conference on Machine Learning and Cybernetics*, pp. 1599 - 1604, 2005.

[65] MacQuee, J.B., *Some methods for classification and analysis of multivariate observations. 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281 - 297, 1967.

[66] Leung, K.-S. and Liang, Y., *Adaptive elitist-population based genetic algorithm for multimodal function optimization.* Lecture Notes in Computer Science, 2003: pp. 1160-1171.

[67] Nowostawski, M. and Poli, R., *Parallel genetic algorithm taxonomy. Third International Conference on Knowledge-Based Intelligent Information Engineering Systems* pp. 88 - 92, 1999.

[68] Melanie, M., *An Introduction to Genetic Algorithms.* 1996: MIT Press.

[69] Coello, C.A.C., *An updated survey of GA-based multiobjective optimization techniques.* ACM Computing Surveys, 2000. 32(2): pp. 109-143.

[70] Watanabe, S. and Sakakibara, K., *Multi-objective approaches in a single-objective optimization environment. IEEE Congress on Evolutionary Computation*, pp. 1714 - 1721, 2005.

[71] Darwen, P. and Yao, X., *A dilemma for fitness sharing with a scaling function. IEEE International Conference on Evolutionary Computation*, pp. 166 - 171, 1995.

[72] Lambert, S., Leau, E.d., and Vuurpijl, L., *Using pen-based outlines for object-based annotation and image-based queries. the Third International Conference on Visual Information and Information Systems*, pp. 585 - 592, 1999.

[73] Pope, R., *Model-based object recognition - a survey of recent research.* 1994, University of British Columbia.

[74] Ho, C.T. and Chen, L.H., *A fast ellipse/circle detector using geometric symmetry.* Pattern Recognition, 1995. 28(1): pp. 117-124.

[75] McLaughlin, R.A., *Randomized hough transform: improved ellipse detection with comparison.* Pattern Recognition Letters, 1998. 19(3-4): pp. 299-305.

[76] Guil, N. and Zapata, E.L., *Lower order circle and ellipse Hough Transform* Pattern Recognition, 1997. 30(10): pp. 1729-1744.

[77] Zhang, S.C. and Liu, Z.Q., *A robust, real-time ellipse detector.* Pattern Recognition, 2005. 38: pp. 273-287.

[78] Cheng, Z. and Liu, Y., *Efficient technique for ellipse detection using restricted randomized Hough Transform. International conference on information technology: coding and computing*, pp. 714 - 718 2004.

[79] Le Troter, A., Bulot, R., Boi, J.-M., and Sequeira, J., *Arc of ellipse detection for video image registration. IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 365 - 367, 2005

[80] Lei, Y. and Wong, K.C., *Ellipse detection based on symmetry.* Pattern Recognition Letters, 1999. 20(1): pp. 41-47.

[81] Sewisy, A. and Lebert, F., *Detection of ellipses by finding lines of symmetry in the images via an Hough transform applied to straight lines.* Image and Vision Computing, 2001. 19(12): pp. 857-866.

[82] Xie, Y. and Ji, Q., *A new efficient ellipse detection method. International Conference on Pattern Recognition*, pp. 957-960, 2002.

[83] Basca, C.A., Talos, M., and Brad, R., *Randomized Hough Transform for ellipse detection with result clustering. The International Conference on Computer as a Tool*, pp. 1397 - 1400, 2005.

[84] Chia, A.Y.S., Leung, M.K.H., Eng, H.-L., and Rahardja, S., *Ellipse Detection with Hough Transform in One Dimensional Parametric Space. IEEE International Conference Image Processing*, pp. V - 333 - V - 336 2007.

[85] Li, H., Lavin, M.A., and Master, R.J.L., *Fast Hough transform: a hierachical approach.* J. Comput. Vision Graphics Image Process, 1986. 36: pp. 139-161.

[86] Kasemir, K.U. and Betzler, K., *Detecting ellipses of limited eccentricity in images with high noise levels.* Image Vision Computing, 2003. 21: pp. 221-227.

[87] Ballard, D.H., *Generalizing the Hough Transform to detect arbitrary shapes.* Pattern Recognition, 1981. 13(2): pp. 111-122.

[88] Roth, G. and Levine, M.D., *Geometric primitive extraction using a genetic algorithm. IEEE* Transactions on Pattern Analysis and Machine Intelligence, 1994. 16(9): pp. 901 - 905.

[89] Chakraborty, S. and Deb, K., *Analytic curve detection from a noisy binary edge map using genetic algorithm. International Conference on Parallel Problem Solving from Nature*, pp. 129-138, 1998.

[90] Mainzer, T., *Genetic algorithm for shape detection.* 2002, University of West Bohemia.

[91] Procter, S. and Illingworth, J., *A comparison of the randomized hough transform and a genetic algorithm for ellipse detection*, in *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, E. Gelsema and L. Kanal, Editors. 1994, Elsevier Science Ltd. pp. 449-460.

[92] Jiang, T., Yang, F., Fan, Y., and Evans, D.J., *A parallel genetic algorithm for cell image segmentation.* Electronic Notes on Theoretical Computer Science, 2001. 46: pp. 1-11.

[93] Ayala-Ramirez, V., Garcia-Capulin, C.H., Perez-Garcia, A., and Sanchez-Yanez, R.E., *Circle detection on images using genetic algorithms.* Pattern Recognition Letters, 2006. 27: pp. 652-657.

[94] Ke, Q., Jiang, T., and Ma, S., *A tabu search method for geometric primitive extraction.* Pattern Recognition Letters, 1997. 18(14): pp. 1443-1452.

[95] McLaughlin, R.A. and Alder, M.D., *The hough transform versus the upwrite.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998. 20(4): pp. 396-400.

[96] Yin, P.Y., *A new circle/ellipse detector using genetic algorithms.* Pattern Recognition Letters, 1999. 20: pp. 731-740.

[97] Chai, J.X., Jiang, T., and Ma, S.D., *Evolutionary tabu search for geometric primitive extraction*, in *Soft Computing in Engineering Design and Manufacturing*, P. Chawdhry, R. Roy, and R.K. Pant, Editors. 1998, Springer-Verlarg: London. pp. 190-198.

[98] Jiang, T. and Yang, F., *An evolutionary Tabu Search for cell image segmentation.* IEEE Transactions on Systems, Man and Cybernetics, 2002. 32(5): pp. 675-678.

[99] Press, W.H. and Teukolsky, S.A., *Numerical Recipes in C: The Art of Scientific Computing*. Second Edition ed. 1992, 43-50: Cambridge University Press.

[100] Hearn, D. and Baker, M.P., *Computer Graphics, C Version*. 2nd Edition ed. 1996: Prentice Hall.

[101] Cordon, O. and Herrera, F., *Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems.* Fuzzy Sets and Systems, 2001. 118(2): pp. 235-255.

[102] Yuen, S.Y. and Ma, C.H., *Genetic algorithm with competitive image labelling and least square.* Pattern Recognition, 2000. 33(12): pp. 1949-1966.

[103] Eick, C.F., Kim, Y.-J., and Secomandi, N., *Enhancing diversity for a genetic algorithm learning environment for classification tasks. Sixth International Conference on Tools with Artificial Intelligence*, pp. 820 - 823, 1994.

[104] Saito, H. and Mori, M., *Object modeling from multiple images using genetic algorithms. the 13th International Conference on Pattern Recognition*, pp. 669 - 673, 1996.

[105] Mclntyre, A.R. and Heywood, M.I., *Toward co-evolutionary training of a multi-class classifier. IEEE Congress on Evolutionary Computation*, pp. 2130 - 2137 2005.

[106] Saito, H. and Kimura, M., *Shape modeling of multiple objects from shading images using genetic algorithms. IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2463 - 2468, 1996.

[107] Fang, B., Hsu, W., and Lee, M.L., *On the Accurate Counting of Tumor Cells.* IEEE Trans. Nanobioscience, 2003. 2(2): pp. 94 - 103.

[108] Garrido, A. and de la Blanca, N.P., *Applying deformable templates for cell image segmentation.* Pattern Recognition, 2000. 33(5): pp. 821-832.

[109] Wu, H., Barba, J., and Gil, J., *Iterative thresholding for segmentation of cells from noisy images.* Journal of Microscopy, 2000. 197(3): pp. 296-304.

[110] Malpica, N., Santos, A., Tejedor, A., Torres, A., Castilla, M., Garcia-Barreno, P., and Desco, M., *Automatic quantification of viability in epithelial cell cultures by texture analysis.* Journal of Microscopy, 2003. 209(1): pp. 34-40.

[111] Anoraganingrum, D., Kröner, S., and Gottfried, B., *Cell segmentation with adaptive region growing. 10th International Conference on Image Analysis and Processing* pp. 27 - 29, 1999. Italy.

[112] Malpica, N., de Solorzano, C.O., Vaquero, J.J., Santos, A., Vallcorba, I., Garcia-Sagredo, J.M., and del Pozo, F., *Applying watershed algorithms to the segmentation of clustered nuclei.* Cytometry, 1997. 28: pp. 289-297.

[113] Woods, R.E. and Gonzales, R.C., *Digital Image Processing.* 1993, Reading, MA: Addison-Wesley.

[114] Wu, K., Gauthier, D., and Levine, M.D., *Live cell image segmentation.* IEEE Transactions on Biomedical Engineering, 1995. 42(1-12).

[115] Bamford, P. and Lovell, B., *A water immersion algorithm for cytological image segmentation. APRS Image Segmentation Workshop*, pp. 75-79, 1996. Sydney, Australia.

[116] Jeacocke, M.B. and Lovell, B.C., *A multi-resolution algorithm for cytological image segmentation. New Zealand Conference on Intelligent Information Systems*, pp. 322-326, 1994.

[117] Spann, M. and Wilson, R., *A quadtree approach to image segmentation that combines statistical and spatial information.* Pattern Recognition, 1985. 18: pp. 257-269.

[118] Otsu, N., *A Threshold Selection Method from Grey-Level Histograms.* IEEE Transactions on System, Man and Cybernetics, 1979. 9(1): pp. 377-393.

[119] Bovik, A., *Handbook of Image & Video Processing*. 2000: Academic Press.