Niching in Evolution Strategies

Ofer M. Shir

5th November 2004

Contents

List of Figures

Chapter 1

Introduction

The term *Evolutionary Algorithms* (EAs) refers to a collection of search methods for highly demanding problems, computationally wise, such as function optimization, classifications and machine learning, simulations of real-time complex systems and many other applications. The EAs field got its origins from the Biology area and, in particular, from the Darwinian Evolution theory. It had been inspired by the famous claim which guarantees "the survival of the fittest". Those methods aim to imitate evolution, and evolve proposed solutions for the given problem into the best solution. EAs have three main streams: *Genetic Algorithms* (GAs), developed by J. Holland in the U.S. [14], *Evolution Strategies* (ES), developed in Germany by I. Rechenberg [18] and H.P. Schwefel [19], and *Evolutionary Programming* (EP), developed by L.J. Fogel et al. in the U.S. [10]. Whereas ES and EP are close and share many basic characteristics in common [2], the principal difference between them and GAs is the encoding of the genetic information. GAs encode the genome with discrete values (as in nature), whereas ES as well as EP do that with continuous real values. The three fields had been developed independently, but at some point began to be influenced by each other.

EAs, and in particular GAs as well as ES, have the tendency to converge quickly into one solution, which means that all the individuals of the artificial population (i.e. proposed solution for the problem) evolve to become nearly identical. Hence, given a problem with multiple solutions, the traditional GAs and ES will locate a single solution. This is of course the desired result for many complex tasks, but a problem arises when we deal with multimodal domains, where we are interested in multiple solutions for the given problem.

Niching methods aim to maintain the diversity of certain properties within the population, and by that to allow parallel convergence within those subpopulations into multiple good solutions for the given problem. Niching methods are thus a desired tool for classification and machine learning, multimodal function optimization, and simulation of complex systems. Up to date, *niching methods* have been studied only in the field of GAs. The research in this direction has yielded a various of successful methods which have been proved to satisfy the requirements and to find multiple solutions efficiently.

In the context of real-valued multi-variable functions optimization, ES are obviously the most natural environment among all the branches of EAs. This is simply due to their straightforward encoding, as well as to their successful performance in this domain in comparison with other methods. The higher the dimensionality of the function is, the more obvious becomes the advantage of ES in comparison to GAs. Since many complex search problems require the location of multiple solutions, and since the natural framework for those problems is ES, it is clear that the development of ES niching methods is needed.

The purpose of this work, which is defined as a study of *niching methods*, is to propose a successful ES niching algorithm. In order to achieve this goal, we develop the appropriate framework for presenting the *niching methods* of the GAs field, and then analyze their main characteristics. This is followed by the description of our proposed method and the experimental setup in which we tested our new algorithm.

It is important to remark that our search for an ES *niching method* has been mainly multimodal function optimization oriented and, in particular, we have been seeking for a fast method which aims to find efficiently and in parallel all the multiple solutions of the given problem. In that sense, we may have ignored some properties of the artificial population, which had been of interest in GAs' niching methods researches, and we let ourselves focus solely on getting the final multiple solutions.

The remainder of the thesis is as follows. Chapter 2 presents the framework: the *Evolution Strategies* field. It gives an overview of the basic elements of the theory, and describes the main algorithms and properties of the traditional ES.

Chapter 3 begins by giving some motivation for *niching methods* - introducing the *diversity* property, which had been the original motivation for developing such methods. This is followed by a brief overview of the main benchmarks of niching methods in the GAs framework. In particular, we introduce the important concepts of *Crowding* and *Sharing,* which will have a role in our proposed method.

Chapter 4 presents our proposed niching method, "The ES Dynamic Niching Algorithm". We begin with a short review of diversity elements in the standard ES, which gives the motivation for developing such a method. This is followed by the presentation of our proposed method: we elaborate on the main concepts of the method, and then we give a detailed description of our algorithm. The chapter ends with the derivation of auxiliary formulas which are in use.

Chapter 5 describes the Experimental Setup for testing our algorithm, including a discussion concerning the performance criteria. This is followed by the Experimental Results which our simulations yielded, and a short discussion over the results.

Chapter 6, the final chapter, provides a short summary of our work, and gives our conclusions. It also outlines further directions for study in this field.

Chapter 2

Evolution Strategies

Evolution Strategies (ES) are a set of general purpose probabilistic search methods, which are based upon the theory of natural Evolution. Inspired by the genetic concepts and motivated by the "survival of the fittest" principle, the basic idea in ES is to create an artificial environment which encodes a search problem into biology-like terms. By defining the appropriate artificial genetic operators, ES imitate the natural evolution process, and lets the artificial environment evolve into the production of the best solution.

This chapter is based mainly on the research done by Bäck [4].

2.1 The Standard ES: Basic Definitions

The purpose of this section is to formally define the basic terms of our framework - the standard ES.

2.1.1 Fitness

Given a search problem, it is usually straightforward to define a quantitative term to describe the fitness of a proposed solution. In optimization problems, which are of our main interest, such a term is naturally the objective function value.

Let us begin by introducing the elementary terminology of a parameter optimization problem. Given the objective function, also called the target function,

$$f \; : \; M \subseteq \Re^n \rightarrow \Re, \quad M \neq \emptyset$$

where M is the set of feasible solutions

$$M = \{\vec{x} \in \Re^n \mid g_j(\vec{x}) \geq 0 \; \forall j \in \{1, ..., q\}\}, \quad g_j(\vec{x}) \; : \Re^n \rightarrow \Re$$

with q inequality constraints $g_j(\vec{x})$, our goal is to find a vector $\vec{x} \in M$ which satisfies

$$\forall \vec{x} \in M \; : \; f(\vec{x}) \leq f(\vec{x}^*) \equiv f^*$$

and then f^* is defined as the *global maximum* and $\vec{x}^*$ is the *maximum location.*

Due to

$$min\{f(\vec{x})\} = -max\{-f(\vec{x})\}$$

it is straightforward to convert each *minimization problem* into a maximization problem. Thus, without loss of generality, we shall stick to maximization.

A *local maximum* $\hat{f} = f(\hat{\vec{x}})$ is defined in the following manner:

$$\exists \epsilon > 0 \, \forall \vec{x} \in M \; : \; \left\| \vec{x} - \hat{\vec{x}} \right\| < \epsilon \Rightarrow \hat{f} \geq f(\vec{x})$$

2.1.2 The individuals

ES consider a population of proposed solutions to the given problem. An individual is associated with a chromosome of real values, which holds a potential solution for the problem, as well as control evolution parameters.

Every individual of the population is represented by

$$\vec{a} = (\vec{x}, \vec{s})$$

where $\vec{x} \in \Re^n$ denotes the *object variable vector*, i.e. a representation of a solution: x_i are the *decision parameters* to be optimized. Furthermore, $\vec{s} \in \Re^m$ denotes the *strategy parameters vector*, which holds the control parameters for the genetic operators. The dimension m of the strategy parameter space is subject to the desired parameter control approach, to be presented shortly.

2.1.3 The Strategy Parameters: The Self-Adaptation Principle

The strategy parameters are incorporated into the individual's representation according to the so called *self-adaptation* principle. Due to this strong principle, the control parameters are an integral part of an individual, and they evolve along with the decision parameters through the evolution process. This principle does not exist in the traditional GA, where the control parameters define the environment, and are set globally in a deterministic manner, based on various considerations.

The strategy parameters of ES can basically be composed of all the control parameters of the algorithm - the mutation rates, the step sizes, the crossover rates etc. However, we limit ourselves to contain only mutation rates, as will be presented.

Given a decision parameter space of dimension n, a general mutation-control mechanism introduces the *covariance matrix* Σ^{-1}; its elements are the $\frac{n\cdot(n-1)}{2}$ variances and the covariances of the decision parameters to be optimized. This yields altogether $m + n = \frac{n(n+1)}{2}$ parameters for the representation of an individual. This general mechanism is often reduced - a diagonalized covariance matrix is introduced, where the decision parameters are assumed to be *i.i.d.*

2.1.4 The Mutation Operator

A vector of random variables with a joint-normal distribution $\vec{z}$, determines the additive modification of the object variable vector $\vec{x}$. Given Σ^{-1}, the covariance matrix of $\vec{x}$, the *multivariate normal distribution*, represented by its *p.d.f.*, is introduced (zero mean is assumed):

$$\vec{z} \sim N\left(\vec{0}, \Sigma\right) \quad : \quad \Phi\left(\vec{z}\right) = \sqrt{\frac{\det \Sigma}{\left(2\pi\right)^{n}}} \cdot \exp\left(-\frac{1}{2} \cdot \vec{z}^{T} \cdot \Sigma \cdot \vec{z}\right)$$

where Σ^{-1} is assumed to be positive definite.

Then, <u>the mutation operator acts as follows</u>:

$$\vec{x}' = \vec{x} + \vec{z}$$

According to the self-adaptation principle which was given in the previous section, the covariance matrix elements also evolve. It is convenient to introduce the rotation angles α_{ij} in order to present the non-diagonal elements of Σ^{-1} in the following manner:

$$covariance(x_i, x_j) = \frac{1}{2}\left(\sigma_i^2 - \sigma_j^2\right) \cdot \tan\left(2\alpha_{ij}\right)$$

where σ_i^2, σ_j^2 are the variances of x_i, x_j respectively.

The mutation procedure is given below, where the first two steps are the adaptation of the control parameters, and the last step is the mutation of the object variable vector:

$$\begin{aligned}
&\sigma_i' = \sigma_i \cdot \exp\left(\tau' \cdot N\left(0,1\right) + \tau \cdot N_i\left(0,1\right)\right)\\
&\alpha_j' = \alpha_j + \beta \cdot N_j\left(0,1\right)\\
&\vec{x}' = \vec{x} + \vec{N}\left(\vec{0}, \Sigma\right)
\end{aligned}$$

$N(0,1)$, $N_i(0,1)$ and $N_j(0,1)$ denote independent random variables, and τ, τ' and β are constants. Once the first two steps are completed, the covariance matrix $\Sigma = \Sigma\left(\vec{\sigma}, \vec{\alpha}\right)$ can be updated according to the relations specified above.

2.1.5 The Recombination Operator

Due to the continuous nature of the variables, there are two ways to recombine two parents alleles:

- *Discrete recombination*: one of the alleles is randomly chosen.
- *Intermediate recombination*: the two values are averaged.

Moreover, many recombination mechanisms were derived based on those two ways (for example, global recombination, where more than 2 parents are involved in the recombination process, was introduced).

The recombination operator's action can be summarized as follows:

1. For each *object variable* choose two parents, and apply *discrete recombination* on the corresponding variables.

2. For each *strategy parameter* choose two parents, and apply *intermediate recombination* on the corresponding variables.

2.1.6 The Selection Operator

Two deterministic operators are introduced in the standard ES using an elegant notation due to Schwefel [19]. The notation characterizes the selection mechanism, as well as the number of parents and offsprings involved:

- $(\mu + \lambda)$-selection: the next generation of parents will be the best μ individuals selected out of the union of current parents and offsprings.

- (μ, λ)-selection: the next generation of parents will be the best μ individuals selected out of the current offsprings.

It is worth mentioning an additional selection mechanism, probabilistic, which will play a role in chapter 4 - the *Tournament Selection.* This selection mechanism works as follows: given a parameter k, each one of the next generation individuals is selected independently as the winner of a fitness-competition among k random individuals.

All the three selection mechanisms which were given here choose the best individuals out of partial sets of the population, or the whole population. Hence, they obviously produce high selective pressure, as will be discussed later on.

2.2 The Standard ES: The Dynamics

The standard ES randomly initializes a population of μ individuals, where all the parameters, decision and strategy, are randomly generated. The evolution process proceeds for a given number of *generations*, unless it satisfies earlier a halting criterion. During each generation, the standard ES applies the recombination operator, followed by the mutation operator, followed by the selection operator. The traditional algorithm can be summarized as follows:

Algorithm 1 (μ, λ)-ES, $(\mu + \lambda)$-ES

```
$t = 0$;
initialize population: $P(0) := \{\vec{a}_1(0), ..., \vec{a}_\mu(0)\}$;
evaluate $P(0)$;
while termination criterion not fulfilled do
  recombine: $\vec{a}'_k(t) := Recomination\,\{P(t)\}\ \ \forall k \in \{1, ..., \lambda\}$;
  mutate: $\vec{a}''_k(t) := Mutate\,\{\vec{a}'_k(t)\}$;
  evaluate $P'(t) := \{\vec{a}''_1(t), ..., \vec{a}''_\lambda(t)\}$;
    $(\{f(\vec{x}''_1(t)), ..., f(\vec{x}''_\lambda(t))\})$;
  select $P(t+1) := if\,(\mu, \lambda)$-ES
    $then\ Selection\,\{P'(t)\}$;
    $else\ \ Selection\,\{P'(t) \cup P(t)\}$;
  $t := t + 1$;
od
```

Chapter 3

Genetic Algorithms' Niching Methods

We will begin this chapter by providing the motivation for studying and using *niching methods*: the *diversity* term and its properties. This will be followed by the description of the main niching methods in the GAs field.

This chapter is based mainly on the research done by Mahfoud [16].

3.1 GAs' Diversity

The promotion of *diversity* in the traditional GA had been originally the main motivation for the development of niching methods. Promoting diversity can contribute in two directions. The first is to prevent *premature convergence* (a term which refers to convergence to undesirable solutions, such as local optima, or to non-optima points), and by that to encourage further exploration of the search space, while seeking for the global optimum. The second direction is to find multiple solutions for the given problem. It should be noted that the two directions are not necessarily mutually exclusive: achieving convergence delay does not apply locating multiple solutions, and vice versa. However, we find it important to study diversity properties, and in particular to analyze the main elements of EAs which prevent diversity, in order to gain a better understanding of niching methods.

3.1.1 General: The Traditional GA

The traditional GA loses diversity due to three main effects: *selection pressure, selection noise, and operator disruption.* We shall give a brief analysis of these three effects, but first we would like to note that we are aware of the fact that much can be elaborated on these three effects and that each one of the subsections deserves a broader discussion. Moreover, this paper lacks the appropriate introduction for a proper discussion on those issues, i.e. GAs' background. However, those elements exceed the scope of this work, so we will limit ourselves just to give a brief review of these three directions.

3.1.2 Selection Pressure

The traditional GA applies a probabilistic *Selection* mechanism, the *Roulette-Wheel Selection (RWS).* This mechanism belongs to a broad set of selection mechanisms which follow the fitness-proportionate selection principle. According to this principle, the higher your fitness is, the higher are your chances to have an offspring in the next generation. Obviously, this is the most intuitive cause for the loss of diversity in the traditional GA.

Selection pressure is associated with the 1*st Moment of the selection operator*, or its *Expectation.* It has been demonstrated by Mahfoud [16] that the selection pressure, or equivalently the non-zero expectation of the selection operator, prevents the evolution process from the formation and the convergence to non-globally optima.

3.1.3 Selection Noise

Selection noise is associated with the 2*nd Moment of the selection operator*, or its *Variance.* Mahfoud [16] demonstrated that the high variance of the RWS, as well as of other selection mechanisms, is responsible for the fast convergence of a population into one solution, even when there exists a set of identically fit solutions.

3.1.4 Operator Disruption

Evolution operators in general, and the *Mutation* and *Recombination* operators in particular, boost the evolution process towards exploration of the search space. In that sense, they have a constructive effect on the process, since they allow locating new and better solutions. However, their action has also a destructive effect. This is due to the fact that by applying them we might lose good solutions that have been located previously.

The Mutation operator usually has a small effect, since it acts in small steps - low mutation probability in the traditional GA, which means infrequently occurrence of bit flips. In that sense, the mutation operator can be considered to have a negligible disruption.

The Recombination operator, on the other hand, has a much more dramatical effect. In the GAs field, where the *Crossover* operator is in use (single-point, two-point or n-point crossovers), it has been shown to have a disruptive nature by breaking desired patterns within the population (the well known *Schema Theorem* discusses the schema disruption by the crossover operator and states that schemata with high defining length will most likely be disrupted by the crossover operator).

3.2 GAs' Niching Methods

Genetic Algorithms' Niching Methods have been studied during the past few decades. The research has yielded a variety of different methods, and we will give in this section a short overview of the main known methods within the framework of GAs, and introduce the important concepts of *Crowding* and *Sharing*.

3.2.1 Crowding

This was one of the pioneering methods in this field, and was introduced by De Jong in 1975 [7]. The Crowding approach aims to reduce changes in the population distribution between generations, in order to prevent pre-mature convergence. Next, we will describe the method briefly.

Given the traditional GA, a proportion G of the population is selected in each generation via fitness-proportionate selection to undergo crossover and mutation - out of which a part is chosen to die and to be replaced by the new offsprings. Each offspring finds the elements it replaces by taking a random sample of CF (Crowding Factor) individuals from the population, and replaces the most similar individual from the sample. An appropriate Similarity Metric should be chosen, whereas the one proposed in the original paper was a genotype based metric.

This mechanism was later improved by Mahfoud [15], to a mechanism known as Deterministic Crowding.

3.2.2 Fitness Sharing

The Sharing mechanism was also one of the pioneering methods in this field. It was first introduced by John Holland in 1975 [14], and later expanded by Goldberg and Richardson [11]. This strong approach of considering the fitness as a shared resource has become a principal element in the GAs' niching field, and has yielded one of the only successful techniques of multimodal function optimization within GAs. A short description of the method follows.

Given the Similarity Metric of the population, which can be genotype or phenotype based, let us introduce the traditional *Sharing Function*:

$$sh(d_{i,j}) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\sigma_{sh}}\right)^{\alpha_{sh}} & \text{if } d_{i,j} < \sigma_{sh} \\ 0 & \text{otherwise} \end{cases}$$

where $d_{i,j}$ is the distance between individuals i and j, determined by the given Similarity Metric, σ_{sh} is the fixed radius of every niche, and α_{sh} is a control parameter, usually set to 1.

Using the *Sharing Function*, we define the *Niche Count* as follows:

$$m_i = \sum_{j=1}^{N} sh(d_{i,j})$$

Given an individual's fitness f_i, we define its *shared fitness* as follows:

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

Thus, the basic idea behind sharing is reducing the fitness of individuals that have similar members within the population, and by that reducing redundancy.

Few remarks regarding this mechanism should be made. It is important to note that the formulas for determining the value of σ_{sh}, which are omitted in this chapter but will be given partially later, are dependent on q, the number of peaks of the target function. Hence this approach holds two strong assumptions - q can be estimated, and all peaks are at least in distance $2\sigma_{sh}$ from each other. In practice, an accurate estimation of the expected number of peaks q in a given domain may turn to be extremely difficult. Moreover, peaks may vary in shape, and this would make the task of determining σ_{sh} rather complicated.

3.2.3 Dynamic Niche Sharing

In order to improve the Sharing mechanism, a dynamic approach was adopted. This approach aims to dynamically recognize the q peaks of the forming niches, and with this information to classify the individuals as either members of one of the niches, or as members of the "non-peaks domain". This method was developed by Miller and Shaw [17].

Putting it formally, let us introduce the *Dynamic Niche Count*:

$$m_{dsh,i} = \begin{cases} n_j & \text{if ind.}\, i \text{ is within dyn. niche}\, j \\ m_i & \text{otherwise (non-peak individual)} \end{cases}$$

where n_j is the size of the jth dynamic niche, and m_i is the standard *Niche Count*, as defined in the previous section.

The shared fitness is now defined respectively:

$$f_{dsh,i} = \frac{f_i}{m_{dsh,i}}$$

The identification of the dynamic niches can be done in the greedy approach, as presented in the given algorithm outline (algorithm 2), which was given by Miller and Shaw [17]. This approach proved to be efficient as well as accurate.

Algorithm 2 Greedy Dynamic Peak Identification

```
input:      $Pop$ - array of population members
            $N$ - population size
            $q$ - number of peaks to identify
            $\sigma_{sh}$ - niche radius.

Sort Pop in decreasing fitness order
$i = 1$
$NumPeaks = 0$
$DPS = \emptyset$ (Dynamic Peak Set)
loop until $NumPeaks = q$ or $i = N + 1$
   if $Pop[i]$ is not within $\sigma_{sh}$ of peak in $DPS$
      $Pop[i] \rightarrow DPS$
      $NumPeaks = NumPeaks + 1$
   endif
   $i = i + 1$
endloop

output:   Dynamic Peak Set
```

3.2.4 Sequential Location of Niches

A method presented by Beasley, Bull and Martin [5]. This method, in contrast to the other methods presented earlier, does not modify the genetic operators or any characteristics of the traditional GA, but rather creates a general search framework suitable for locating multiple solutions. The search process turns in this method into a sequence of independent runs of the traditional GA, where the basic idea is to depress the fitness function at the end of each simulation at the observed optimum of that run, in order to prevent the search from revisiting that optimum. The traditional GA is run multiple times sequentially: given the best solution of each run, firstly it is stored as a possible final solution, and secondly the fitness function is depressed in all the points within the neighbourhood of that optimum up to a desired radius. This modification is done immediately after each run. Its purpose is to discourage the following runs from revisiting those optima, and by that to encourage the exploration of other areas in the search space - and the location of all the optima of the search problem. It should be noted that each function modification might yield artificial discontinuities in the fitness function. This method focuses only in locating the multiple optima of the given search problem, without considering the concepts of parallel evolution and subpopulations formation. In that sense, it has been claimed that it could not be considered as a niching method.

3.2.5 Over-Specification

This mechanism has been developed originally for environments that change over time. The concept of over-specification had been motivated by the fact that a large percentage of the genome of higher organisms has no known functionality up to date, as well as by the well known concept that redundancy protects against the loss of information. The basic idea is to promote the diversity of the individuals by redundantly encoding their genetic information - where the redundant bits are assumed to store the history information of the environment. When the environment keeps changing, previously good solutions are stored in the redundant bits. Those solutions may be stored in case the search reaches similar states. This memory property is a crucial element of this approach, and is implemented using hidden activation bits within the chromosomes. The original primary goal of this approach was to simulate biological genetic systems, whereas problem solving was only secondary. However, this approach has been implemented successfully for problem solving in dynamic environments, and in particular for 2-optima oscillating systems. An important remark is that diversity, expressed particularly in redundant data, is known to disappear when the environment is being static for a long time. No method which uses the over-specification technique for niching methods in static environments has been introduced.

An example of an implementation of such a method for a dynamic environment is *The Structured GA*, by Dasgupta and McGregor [6].

3.2.6 The Regional Population Model - Migration

This is a very intuitive approach which could be easily motivated by the diversity of human beings around the globe. This mechanism divides the population into multiple subpopulations, which evolve independently of each other for a fixed number of generations (known as the *isolation time*). After this period, a number of individuals is distributed among the other subpopulations - a phase which is known as the *migration* process.

The genetic diversity and the amount of information exchange between subpopulations are determined by the following parameters - the number of exchanged individuals, the *migration rate*, the selection method of the individuals for migration (uniformly at random, or elitist fitness-based approach) and the scheme of migration: complete net topology, ring topology or neighbourhood topology - as shown in figures 3.1, 3.2, and 3.3 respectively. The figures were taken from the Toolbox for Genetic and Evolutionary Algorithms [20].

Figure 3.1: Unrestricted migration topology (Complete net topology)

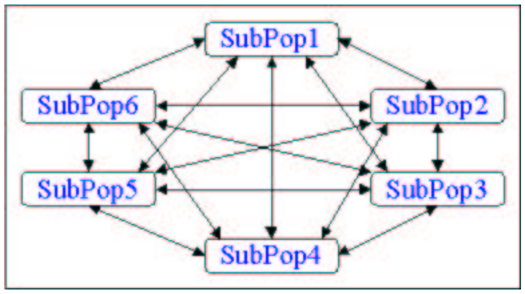


Figure 3.2: Ring migration topology; left: distance 1, right: distance 1 and 2

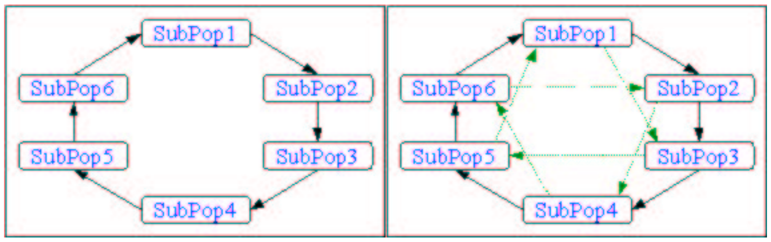
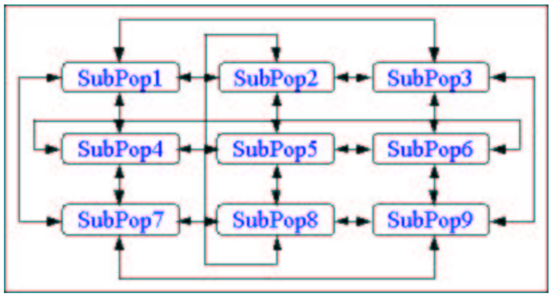


Figure 3.3: Neighbourhood migration topology (2D grid)



Chapter 4

Evolution Strategies: A Proposed Niching Method

In this chapter we present a niching method for the ES framework. We begin by giving the motivation for developing such a method through a brief analysis of the *diversity* properties in ES. After that we give a detailed description of our algorithm.

4.1 ES Diversity

4.1.1 General

The standard ES has a strictly deterministic, rank-based approach. In the two traditional strategies, (μ, λ) and $(\mu+\lambda)$, an approach of deterministically selecting the best individuals (out of the appropriate set - the next generation or the union of the two generations, respectively) is applied, which intuitively implies high selective pressure. However, due to the deterministic nature of the selection operator, *selection pressure* (expectation) and *selection noise* (variance) do not exist. We consider two main effects which cause the standard ES to lose diversity: *Selective Pressure* and *Operator Disruption*. A detailed explanation follows.
A remark regarding the notation should be made - the careful reader will distin-

guish between the two rather confusing terms, Selection Pressure and Selective Pressure, which refer to two different things, and should avoid abuse of notation.

4.1.2 Selective Pressure

Due to the crucial role of the selection operator within the evolution process, its impact within the ES field has been widely investigated. Goldberg and Deb introduced the important concept of *takeover time* [12], which gives a quantitative description of selective pressure:

Definition: *The takeover time* τ^* is the minimal number of generations until repeated application of the selection operator yields a uniform population filled with copies of the best individual.

The selective pressure has been further investigated by Bäck, who analyzed all the ES selection mechanisms also with respect to takeover times [3]. Let us introduce the results for the takeover times of the main selection mechanisms, where we chose to omit the derivations.

Theorem: The takeover time of (μ, λ)-selection is :

$$\tau^*_{(\mu,\lambda)} = \frac{\ln(\lambda)}{\ln\left(\frac{\lambda}{\mu}\right)}$$

Theorem: The takeover time of $(\mu + \lambda)$-selection is given implicitly by:

$$\lambda = \frac{\left(\alpha_1^{\tau^*+1} - \alpha_2^{\tau^*+1}\right)}{\sqrt{\frac{\lambda}{\mu} \cdot \left(\frac{\lambda}{\mu} + 4\right)}}$$

whereas

$$\alpha_{1,2} = \frac{\lambda}{2\mu} \pm \frac{1}{2} \cdot \sqrt{\left(\frac{\lambda}{\mu}\left(\frac{\lambda}{\mu} + 4\right)\right)}$$

Theorem: The takeover time of k-Tournament selection is given by:

$$\tau^*_k = \frac{(\ln(\lambda) + \ln(\ln(\lambda)))}{\ln(k)}$$

It is easy to verify that with the substitution of the traditional values of the standard ES, we get very short takeover times for the given selection mechanisms, which mean high selective pressures.

4.1.3 Operator Disruption

In the standard ES the mutation operator also typically has a small effect - low $\triangle x$ values (or z in the notation of chapter 2), which mean staying in the neighbourhood. In that sense, the mutation operator can be regarded also in the standard ES as an operator with negligible disruption.

The Recombination operator, as in the GAs, has a bigger effect. In the standard ES, where *discrete* and *intermediate recombination operators* are in use, the disruptive nature is also highly intuitive - modifying a coordinate of the decision parameters to be optimized, not in a local manner (averaging or taking a value from a different individual), has the potential to shift the offspring not in a negligible way.

4.1.4 Conclusions

The standard ES is exposed to several strong effects which interrupt the formation and maintenance of multiple solutions and push the evolution process towards a rapid convergence into a single solution. When multiple solutions are required for a given problem, it is obvious that the standard ES will not be able to qualify for this task.

4.2 The ES Dynamic Niching Algorithm

We present an algorithm for *Multimodal Function Optimization* for the Evolution Strategies framework. As far as we know, this is the first niching algorithm ever presented within the ES framework. However, our work is naturally influenced by the various niching algorithms of the GAs field, which have been briefly presented in the previous chapter. In particular, we have been motivated by the *Fitness Sharing* and *Crowding* concepts and by the *Dynamic Niche Sharing* method. Nevertheless, it is important to note that we do not apply at any stage any kind of sharing function but we only apply niche resources sharing, as will be presented shortly.

It is important to emphasize that our goal is to get the multiple optima of the multimodal function, and keep a stable subpopulation in each niche, without considering at all the distribution of the population among the niches. Our performance criteria will be defined accordingly.

4.2.1 The Strategy

The basic idea of our proposed algorithm is to *dynamically identify the various fitness-peaks of every generation* which define the niches, *classify all the individuals into those niches*, and apply *a Mating Restriction Scheme* which allows competitive mating only within the niches.

The algorithm assumes that the number of expected/desired peaks q is given, and accordingly estimates the required radius of each niche, σ_{sh}. This estimation is based on a formula which will be given shortly with its derivation. Moreover, the algorithm tries to keep those q niches in a fixed proportion, meaning fixed mating resources, as will be explicitly explained shortly.

As the reader may conclude, our method has been inspired by various GAs' niching methods and concepts. First, we apply a resources-sharing mechanism, which limits every niche and forces its individuals to share the offsprings resource. Second, our unique selection mechanism replaces individuals from each niche only with individuals from the same niche - an idea which originates from the *Crowding* method. Finally, we imitate the *dynamic niche formation technique* of the Dynamic Niche Sharing method.

4.2.2 The Algorithm

Given a Population of individuals, a standard ES **Mutation Operator** is applied as the first step.

This is followed by the fitness-peaks identification - a greedy approach is applied in identifying the dynamic peaks of each generation, as given previously in the *Greedy Dynamic Peak Identification Algorithm* (algorithm 2). Using the estimated niche radius σ_{sh}, it is straightforward to classify all the individuals of the population into those peaks niches.

At this point the mating phase starts, which is a closed competitive mating session within every niche. Each niche gets fixed mating resources (number of individuals in the next generation), i.e. independent of the fitness value of its peak. In this manner we prevent the best niche to take over the population's resources and flood the next generation with its offsprings.

In particular, we consider a uniform distribution of the resources:

$$\tilde{\mu} \equiv \frac{\mu}{q} \quad , \quad \tilde{\lambda} \equiv \frac{\lambda}{q}$$

meaning that each niche will produce $\tilde{\mu}$ individuals for the next generation, out of which $\tilde{\lambda}$ individuals are new offsprings.

For each niche the same **Selection Mechanism** is applied as follows - at the first stage, $\tilde{\lambda}$ individuals are selected within the niche in a *Tournament Selection.* Those individuals are still singles at this stage, and soon a matching will be found for them from the niche - the second parent will be the best individual in the niche which is different than the first parent. In case that the niche contains only one individual, the second parent will be the best individual of another niche.

Given the $\tilde{\lambda}$ pairs of parents, the **Recombination Operator** is defined as in the standard ES - *Intermediate Recombination* for the strategy parameters and *discrete recombination* for the decision parameters.

The rest of the individuals which will stay in the next generation, explicitly $\delta \equiv \tilde{\mu} - \tilde{\lambda}$ individuals, are chosen in an elitist manner: the best δ individuals of that niche will survive with no recombination. If the niche does not have sufficient individuals, new randomly-generated individuals will be added on that niche's resources.

We define the following notation for our new selection strategy:

$$(\mu \rightarrow \lambda)$$

Our algorithm can be summarized in the following pseudo-code given here:

Algorithm 3 One Generation Loop of the ES Dynamic Niching Algorithm

```
input:    $Pop$ - array of random population members
          $\tilde{\mu}$ - Niche's population size
          $\tilde{\lambda}$ - number of offsprings in each Niche
          $q$ - number of peaks to identify
          $\sigma_{sh}$ - niche radius.

1.  Apply Mutation on Pop
2.  Compute the Dynamic Peak Set using the Greedy Dynamic Peak
Identification Algorithm
3.  For every Niche Apply the following Selection and
Recombination:
  4.  For $\tilde{\lambda}$ offsprings do:
    5.  Choose 1st parent via Tournament Selection
    6.  Choose the best individual of that niche as the 2nd
parent
    7.  Apply Recombination
  8.  Add the best $\tilde{\mu} - \tilde{\lambda}$ individuals of the Niche as offsprings
9.  Join all $\tilde{\mu}$ offsprings from the $q$ Niches to yield the new Pop

output:   Pop
```

4.2.3 The Niche Radius σ_{sh}

The original formula for σ_{sh} for Phenotypic Sharing in GAs was derived by Deb and Goldberg [8]. By following the trivial analogy and considering our decision parameters as the decoded parameter space of the GA, it is straightforward to apply the same formula. Its derivation is given below.

Given that our Individual Space (the decision parameters space) is of dimension p , the distance $d_{i,j}$ is calculated using a suitable distance norm in the p-dimensional space. In particular, let us consider the Euclidean Distance in p-dimensional space.

Given the individuals $\overrightarrow{x_i} = [x_{1,i}, x_{2,i}, ..., x_{p,i}]$ and $\overrightarrow{x_j} = [x_{1,j}, x_{2,j}, ..., x_{p,j}]$ the metric $d_{i,j}$is calculated as:

$$d_{i,j} = \sqrt{\sum_{k=1}^{p}(x_{k,i} - x_{k,j})^2}$$

Given q, the number of peaks in the solution space, we would like to consider every niche as surrounded by a p-dimensional hypersphere with radius σ_{sh} which occupies $\frac{1}{q}$ of the entire volume of the space.

The volume of the hypersphere which contains the entire space is

$$V = cr^p$$

where c is a constant, given explicitly by:

$$c = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}, \qquad \Gamma(n) = \int_0^{\infty} x^{n-1}e^{-x}dx$$

The radius r is calculated as

$$r = \frac{1}{2}\sqrt{\sum_{k=1}^{p}(x_{k,max} - x_{k,min})^2}$$

If we divide the volume into q parts, we may write

$$c\sigma_{sh}^p = \frac{1}{q}cr^p$$

which yields

$$\sigma_{sh} = \frac{r}{\sqrt[p]{q}}$$

Chapter 5

Experimental Results

5.1 Experimental Setup

5.1.1 The Test Functions

We investigated the performance of our algorithm within four multimodal domains in various space dimensions. Our experiments considered the following four theoretical test functions:

1. Introduce $\mathcal{H}$, *Himmelblau's Function* [13]:

   $$\mathcal{H}(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

   In the interval $x_1, x_2 \in [-6, 6]$, this 2-variables function has nine stationary points - four global minima, one maximum, and four saddle points. We shall consider a minimization problem. The four global minima are:

   $$\begin{aligned} \mathcal{H}_{min}^1 &= \mathcal{H}(3, 2) = 0 \\ \mathcal{H}_{min}^2 &= \mathcal{H}(3.5844, -1.8481) = 0 \\ \mathcal{H}_{min}^3 &= \mathcal{H}(-3.7793, -3.2831) = 0 \\ \mathcal{H}_{min}^4 &= \mathcal{H}(-2.8051, 3.1313) = 0 \end{aligned}$$

   Two figures are attached - a three-dimensional visualization in figure 5.1, as well as a two-dimensional contour map in figure 5.2.

Figure 5.1: Himmelblau's Function

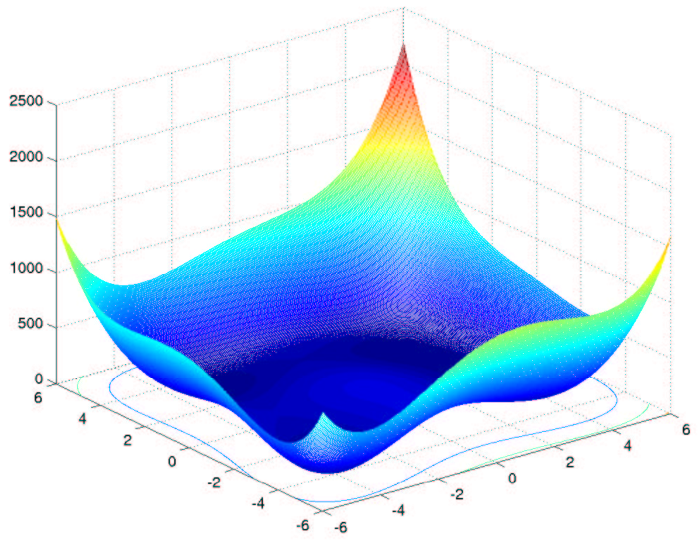
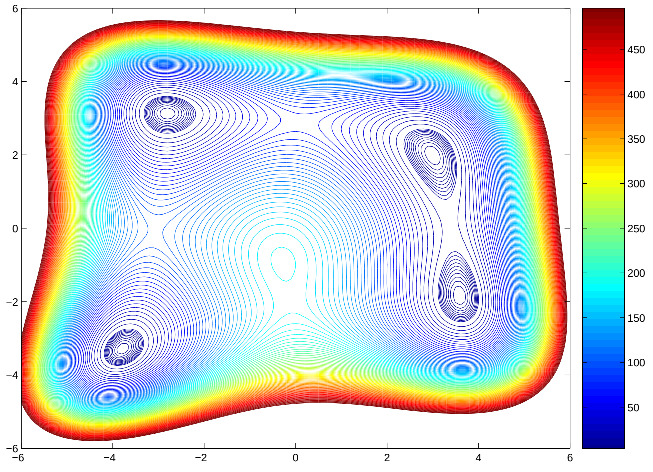


Figure 5.2: Contour Map of the *Himmelblau's Function*



2. Introduce $\mathcal{L}$:

$$\mathcal{L}(\vec{x}) = \prod_{i=1}^{n} \sin^k(5.1 \cdot \pi \cdot x_i + 0.5) \cdot \exp\left(-4 \cdot \ln(2) \cdot \left(\frac{x_i - 0.0667}{0.8}\right)^2\right)$$

where n is the dimensionality and the parameter k determines the sharpness of the peaks in the function landscape (see a two-dimensional plot for $\{n = 1,\ k = 6\}$ in figure 5.3, three-dimensional plot for $\{n = 2,\ k = 4\}$ in figure 5.4, as well as a two-dimensional contour map for the latter in figure 5.5).
$\mathcal{L}$ has one global maximum, regardless of n and k:

$$\forall n \forall k\ \ \mathcal{L}_{max} = 1$$

The case of $n = 1,\ k = 6$ is known as the *multimodal domain* $F2$, as had been originally introduced in Goldberg and Richardson [11]. This periodic function, a sinusoid trapped in an exponent envelope, has been a popular test function for GA niching methods - it is investigated within the interval $x \in [0, 1]$, where it has 5 equally spaced peaks of varying height. We will consider various values of n.

Figure 5.3: $\mathcal{L}$: $n = 1,\ k = 6$, AKA $F2$

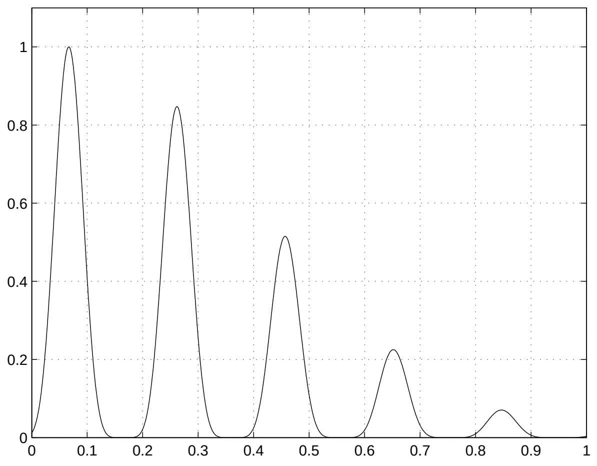


Figure 5.4: $\mathcal{L}$: $n = 2,\ k = 4$

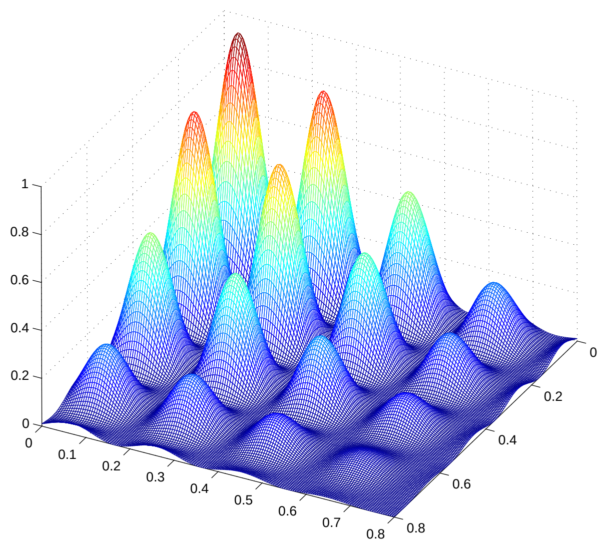
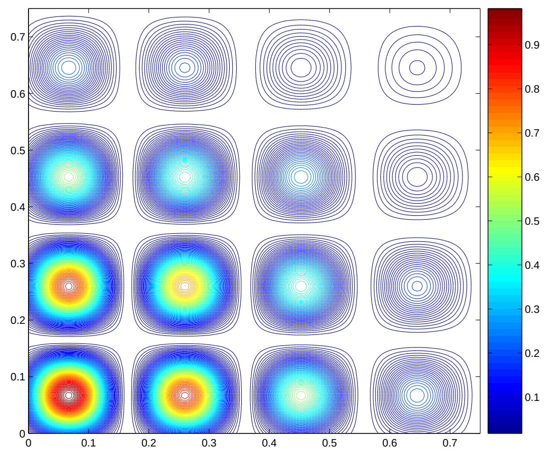


Figure 5.5: Contour Map for $\mathcal{L}$: $n = 2,\ k = 4$



3. Introduce $\mathcal{A}$, *Ackley's Function* [1]:

$$\mathcal{A}(\vec{x}) = -c_1 \cdot \exp\left(-c_2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} cos(c_3 \cdot x_i)\right) + c_1 + e$$

This n-variable function has one global minimum, regardless of its dimension n:

$$\forall n \ \ \mathcal{A}_{min} = \mathcal{A}(\vec{x}^* = \vec{0}) = 0$$

This minimum is surrounded isotropically by $2n$ local minima in the first circle, followed by exponentially increasing number of minima in the upgoing circles (see visualization for $n = 2$: three-dimensional plot in figure 5.6 and a two-dimensional contour map in figure 5.7).

Figure 5.6: Ackley's Function ($n = 2$)

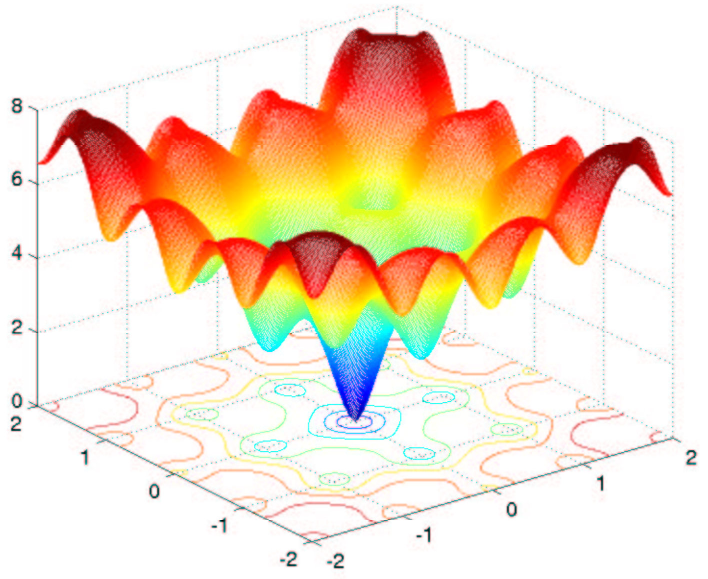
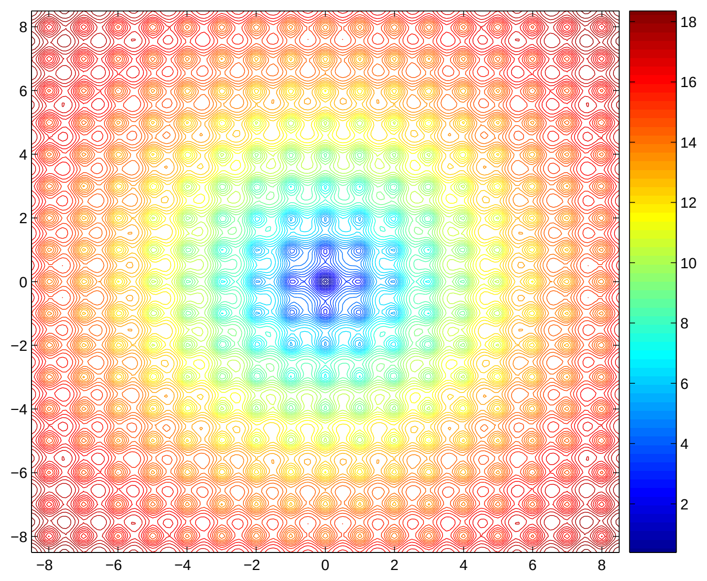


Figure 5.7: Contour Map of the *Ackley's Function*



Ackley's function has been investigated in the context of Evolutionary Computation by Bäck [4].

4. Introduce $\mathcal{F}$, *The Function after Fletcher and Powell* [9, 19, 4]:

$$\begin{aligned}
\mathcal{F}(\vec{x}) &= \textstyle\sum_{i=1}^{n} (A_i - B_i)^2 \\
A_i &= \textstyle\sum_{j=1}^{n} (a_{ij} \cdot sin(\alpha_j) + b_{ij} \cdot cos(\alpha_j)) \\
B_i &= \textstyle\sum_{j=1}^{n} (a_{ij} \cdot sin(x_j) + b_{ij} \cdot cos(x_j))
\end{aligned}$$

where $\mathbf{A} = (a_{ij})$, $\mathbf{B} = (b_{ij})$, and $\vec{\alpha} = (\alpha_j)$ have random elements:

$$a_{ij},\, b_{ij} \in [-100, 100]; \quad \vec{\alpha} \in [-\pi, \pi]^n$$

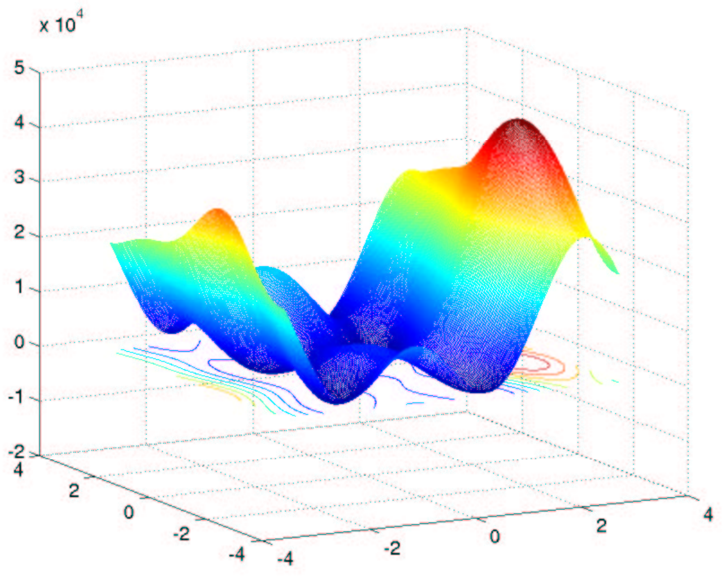
$\mathcal{F}$ in fact is an example of a nonlinear parameter estimation. In our context, this is a multimodal objective function of dimension n, with up to 2^n extrema which satisfy $\vec{x} \in [-\pi, \pi]^n$.

A global minimum is obviously given by:

$$\mathcal{F}_{min} = \mathcal{F}\left(\vec{x}^* = \vec{\alpha}\right) = 0$$

This function introduces two properties which the rest of our test functions do not have. Firstly, the function's extrema are distributed randomly, and secondly, it has no symmetry. Naturally, these two properties make this interesting function a challenging test for our method. Based on the data given by Bäck [4], a three-dimensional visualization is given in figure 5.8, followed by its two-dimensional contour map in figure 5.9.

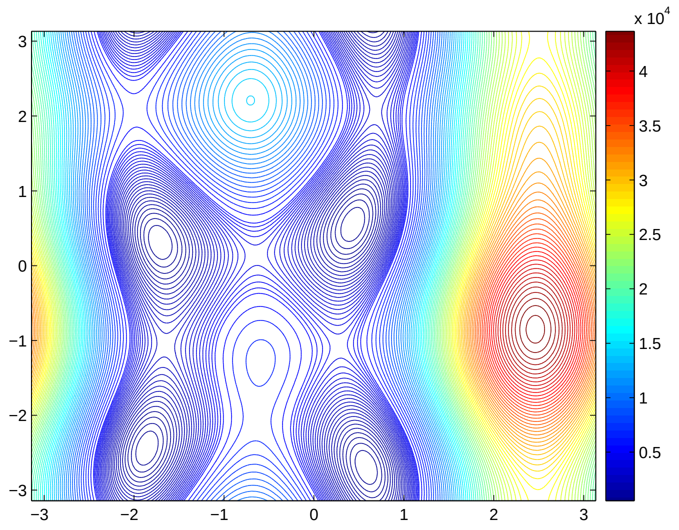
Figure 5.8: The Function after Fletcher and Powell: $n = 2$



5.1.2 Performance Criteria

The traditional GA's niching methods research had been highly interested in the distribution of the final population compared to a desired profile, most likely to a fitness-proportionate profile. This had been measured by the *Chi-square-like performance statistic* [8], which estimates the deviation of the actual

Figure 5.9: Contour Map of the function after Fletcher and Powell



distribution of individuals N_i from an ideal distribution (characterized by mean μ_i and variance σ_i^2) in all the i subspaces (q peak subspaces and the non-peak subspace):

$$\text{Chi-square-like performance metric} = \sqrt{\sum_{i=1}^{q+1}\left(\frac{N_i-\mu_i}{\sigma_i}\right)^2}$$

where the ideal-distribution characteristic values are derived per function.

Our research focuses in the ability of identifying global as well as local optima and converging in those directions through time, with no particular interest in the distribution of the population. Moreover, our algorithm attempts to populate in every generation the niches according to its predefined distribution (the uniform distribution in our implementation), regardless of the objective function. Hence, we find no reason to apply this performance metric in our experiments.

Thus, we adopt another performance metric, called the *maximum peak ratio statistic*, which has also been in use in GAs' niching methods [17]. This metric measures the quality as well as the number of optima given as a final result by the evolutionary algorithm. Given the fitness of the optima in the ending population $\left\{\tilde{f}_i\right\}_{i=1}^{q}$, and the actual optima of the objective function $\left\{\hat{\mathcal{F}}_i\right\}_{i=1}^{q}$, the maximum peak ratio is defined as follows:

$$\text{Maximum Peak Ratio} = \frac{\sum_{i=1}^{q} \tilde{f}_i}{\sum_{i=1}^{q} \hat{\mathcal{F}}_i}$$

The actual optima of the objective function cannot always be obtained analytically, particularly in complex problems. Numerical optima are computed and given when needed. We will consider each simulation's *maximum peak ratio* as a random variable, and we shall supply the appropriate statistics as a performance criterion, i.e. the expectation and the variance.

Moreover, we will consider as a failure of the algorithm every non-stable convergence process - where the algorithm is not capable of maintaining stable good niches along the evolution process. By stable convergence we mean that every niche has its individuals surrounding the optimum in a decreasing radius as time passes - niche convergence. This property is crucial to test our algorithm, since due to its nature the niches are formed in each generation (recall that the peaks identification procedure is called in every generation). Our implementation allows us to track this property, and we will supply an appropriate report on each simulation.

5.2 Implementation

5.2.1 Environment

We chose **MATLAB** 7.0 as the framework for our experiments. The code for the simulations has been written in a compact matrix notation, aiming to exploit **MATLAB**'s well known optimizations for matrices-operations.

5.2.2 The Individuals

We chose to implement our experiments using one control parameter per individual - each individual is represented by the object variables plus one control parameter, which is its standard deviation σ:

$$\vec{a_i} = (\vec{x_i}, \sigma_i)$$

In other words, the covariance matrix Σ is diagonalized.

5.2.3 Parameter Tuning

The parameters of the algorithm, for example the initial random distribution profile of the individuals or the mutation constants, were tuned manually after few runs of each simulation. The parameter q was set according to the a-priori knowledge of the test functions. However, the important population-values $\{\mu, \lambda\}$ were tuned per test function, as differences had been found in the behaviour of the various functions. See further remarks in the discussion in chapter 6.

5.3 Experimental Results

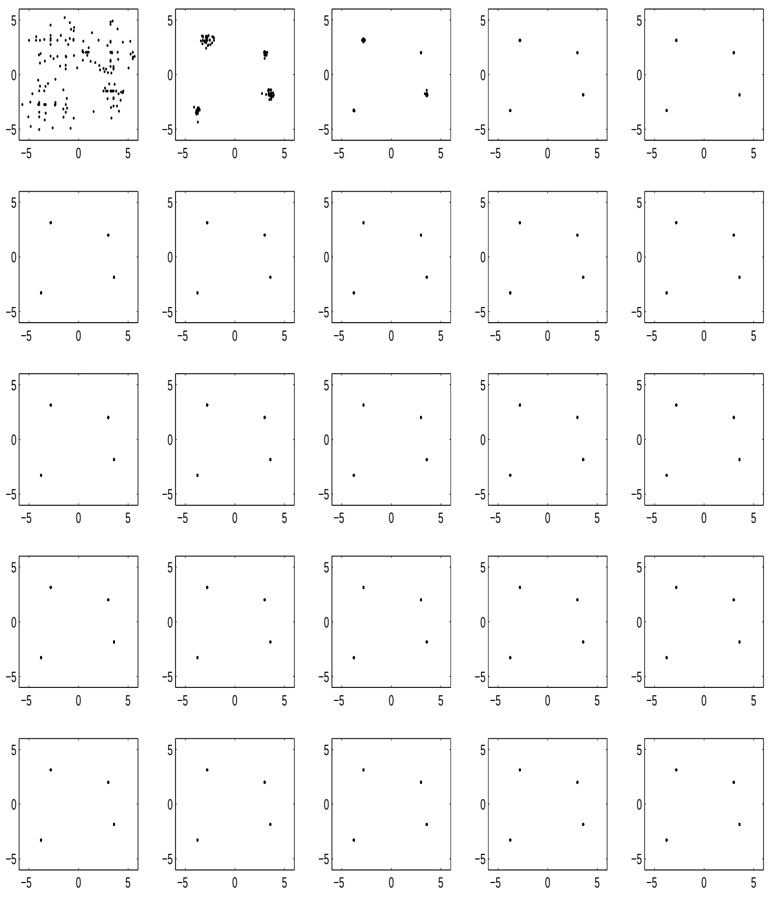
This section summarizes the results of the application of the *ES Dynamic Niching Algorithm* in finding multiple optima for the specified multimodal test functions.
We begin with presenting a short review regarding the simulation of each test function in separated subsections, where visualization of the results is given when possible. This is followed by the presentation of the results. The obtained final results refer to an average over multiple runs on each test function. They are concentrated in a table at the end of this section.

5.3.1 Simulating the *Himmelblau's Function*

This 2-variables function, with its four sparsely located minima, seemed to be an easy task for our method. Using the analytical a-priori knowledge, and considering a minimization task, we set the desired number of peaks to $q = 4$. The convergence as well as the niche formation processes have been rapid, and extremely accurate final results have been achieved.

Due to the function's dimensionality, a visualization of the convergence and niche formation processes is possible. A snapshot gallery is shown in figure 5.10, where in each frame the whole population is plotted on a two-dimensional grid. Each frame is taken within five generations.

Figure 5.10: *Himmelblau's Function*: The Niche Formation Process



Our method performed in the best way under the following population parameters:

$$(200 \rightarrow 120)$$

5.3.2 Simulating $\mathcal{L}$

We have tested this function in a various of dimensions.
For the traditional multimodal domain $F2$, our algorithm achieved a very fast niches formation as well as fast convergence to the actual optima (order of magnitude of ten generations): when setting q to the desired value of five niches, our algorithm performed perfectly with no particular problems.

Our method kept performing very well also in higher dimensions. The niche formation process was fast but, naturally, the convergence process towards the actual optima was slower, due to the dimensionality.

The best performance was under the following population parameters:

$$(24 \cdot q \rightarrow 16 \cdot q)$$

5.3.3 Simulating the *Ackley's Function*

We have focused in locating the global minimum and the first circle of local minima. Hence, in our simulations, the desired number of peaks was set to:

$$q_{\mathcal{A}} = 2n + 1$$

Our method also performed well in this test case - fast niche formation and fast convergence have been achieved. A visualization of the convergence and niche formation processes is also available for this function, in the cases of $n = 2, 3$. The two snapshot galleries are given in figures 5.12 and 5.11.

The best performance was under the following population parameters:

$$(100 \cdot q \rightarrow 50 \cdot q)$$

Figure 5.11: *Ackley's Function* $\{n = 2\}$: The Niche Formation Process

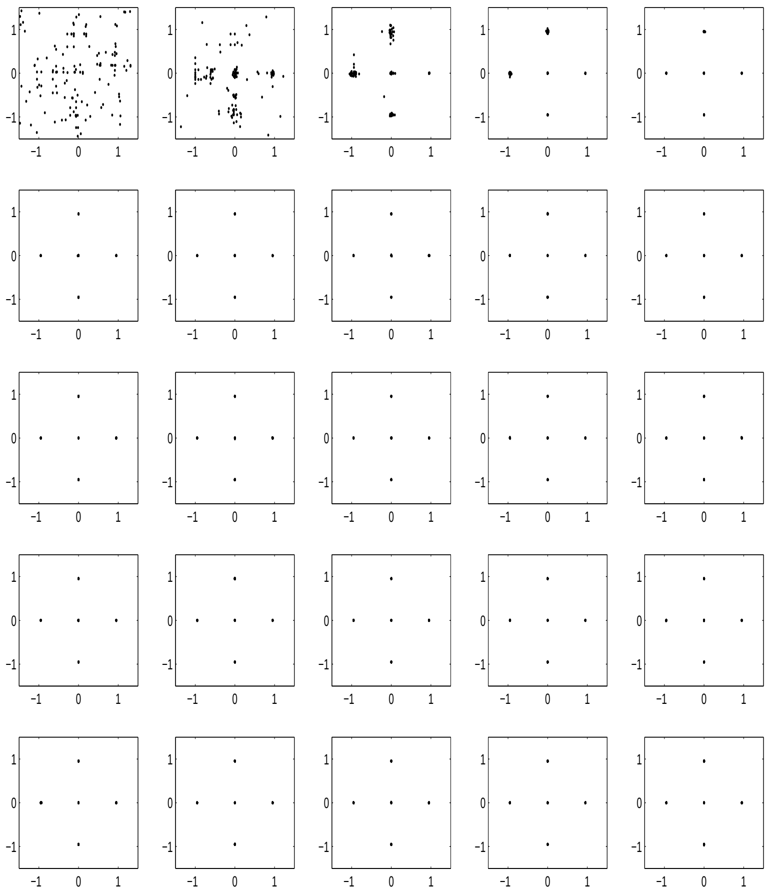
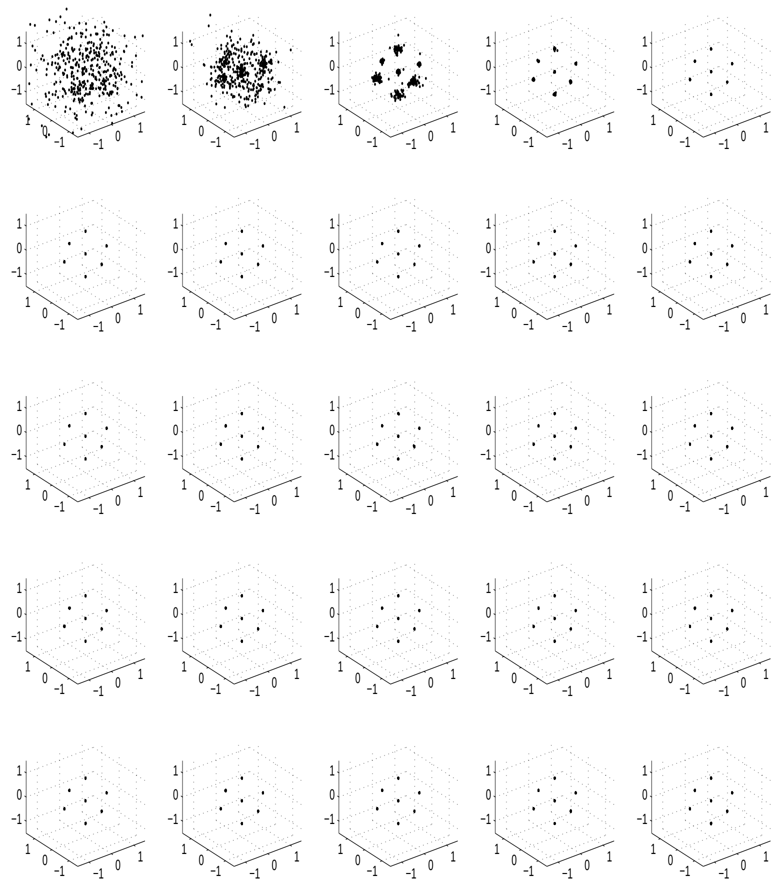


Figure 5.12: *Ackley's Function* $\{n = 3\}$: The Niche Formation Process



5.3.4 Simulating the *Function after Fletcher and Powell*

We have tested this function for a various of dimensions: $2 \leq n \leq 30$. As pointed earlier, we have used the data given by Bäck [4] for initializing the random matrices $\mathbf{A}$, $\mathbf{B}$ and $\vec{\alpha}$, and we have followed his scaling manner for $n < 30$.

Our method has managed to achieve highly satisfactory results for this challenging test case as well.

5.3.5 Performance Results

All simulations were ran up to an upper bound of $10,000$ generations. The final results of the simulations are concentrated in this subsection, in table 5.1. We introduce in the table three measures for each test case:

- The *maximum peak ratio* - as presented earlier. The values given here are *mean values*, where the expectation is over the multiple runs.
- The global optimum location percentage. It refers to the percentage of the runs in which the global optimum was found.
- The number of optima found. It refers to the average over the multiple runs of the number of the optima which were located - with respect to the desired number of peaks, q.

Table 5.1: Performance Results

Function	Max Peak Ratio	% Global Opt.	Optima Found w.r.t. q
$\mathcal{H}$	1	100%	4/4
$\mathcal{L}: \ n=1$	1	100%	5/5
$\mathcal{L}: \ n=2$	1	100%	5/5
$\mathcal{L}: \ n=3$	1	100%	7/7
$\mathcal{L}: \ n=4$	0.9880	100%	4.6/5
$\mathcal{L}: \ n=10$	0.9612	100%	8.2/11
$\mathcal{A}: \ n=2$	1	100%	5/5
$\mathcal{A}: \ n=3$	1	100%	7/7
$\mathcal{A}: \ n=4$	0.9904	100%	8.8/9
$\mathcal{A}: \ n=10$	0.9946	100%	19.2/21
$\mathcal{F}: \ n=2$	1	100%	4/4
$\mathcal{F}: \ n=3$	0.9321	100%	3.4/4
$\mathcal{F}: \ n=10$	0.9141	100%	2.8/4

5.4 Discussion

In this section we will analyze shortly our experimental results and discuss their consequences.

The experimental results which were presented in the last section, show clearly that our method has achieved its goals in two layers. In the evolution layer, it has achieved the goal of forming and maintaining optimal niches within the population, and in the optimization layer it has achieved its goal in finding multiple solutions for the given optimization problems. Moreover, our method performed all its tasks very well, and achieved with no doubt highly satisfactory results, as reflected in the results table.

The visualization tool which has been presented in the form of the snapshot galleries, allows us to get the feeling of the different niches formation processes in the different experiments. It should be noted as a side remark that those figures can somehow resemble another process from nature, but from the completely unrelated field of Astrophysics - a rapid *star formation* process. The niches formations, as can be seen in the figures, are very fast and occur within an order of magnitude of ten generations.
In order to quantitatively measure this property, which could be considered as ***niche formation pressure***, we introduce a new term:

Definition: *the niches formation time* $\tilde{\tau}$ is the minimal number of generations until all niches are around the actual optima, and every niche has its population within a desired ϵ-neighbourhood around the actual optima.

Referring to this definition, and by making the reasonable substitution $\epsilon = \sigma_{sh}$, we can state that the niches formation time of our experiments did not exceed an upper bound of $\tilde{\tau}_{max} = 100$, which implies a high *niche formation pressure* of our algorithm.

Chapter 6

Conclusions and Outlook

In order to conclude our study of niching methods within the Evolution Strategies field, we would like to draw conclusions from our work, and afterward to suggest possible directions for future study in this domain.

6.1 Conclusions

Our study aimed to construct as its first stage the appropriate basis for niching methods within ES. This construction process has been influenced by the equivalent research within the GAs field, where niching methods have been studied deeply. Using the analytical tools of GAs niching methods, and in particular the *diversity* property, we have performed a short analysis of the standard ES in order to give the motivation. We concluded that different factors are responsible for the loss of diversity in the standard ES with comparison to the traditional GA, and this is due to the nature of the selection operator. Two factors play that role in ES - *selective pressure* and *operator disruption.* The main conclusion of that analysis is that these factors interrupt to the formation and maintenance of multiple solutions, and that niching methods are necessary in order to satisfy those requirements and to achieve multiple solutions.

Being motivated and inspired by the primary niching methods of GAs, we have developed an ES niching method, the first ever to be presented. We have set an experimental design, and managed to show that our method qualifies as a robust niching method. Our algorithm performed well and achieved its primary goal in finding the multiple optima of the test functions.

6.2 Future Research

In this section we would like to outline possible directions for further study with respect to our work. Our proposed niching method is the first niching method to be introduced and tested within the *Evolution Strategies* framework up to date. Naturally, the research work which has been done so far by us and presented in this paper is only the introductory work in this domain. We shall describe shortly what are the possible future research directions - both in the context of our proposed method and in the general context of ES niching methods.

In the context of our given method we propose three main directions for further research work:

- Our method should be further tested with more challenging test functions. Non theoretical (i.e. empirical) test functions should be introduced and tested.
- We believe that the population parameters $\{\mu, \lambda\}$ and their influence on the convergence profile should be deeply investigated in the context of our proposed method. The intermediate conclusions from our experimental setup were that for the best performance of the algorithm the parameters should be tuned task-dependently, but we have not worked in that direction of research.
- A possible extension of our method should be to other domains which require the location and maintenance of multiple solutions, such as classification and machine learning, simulation of complex systems etc.

Regarding the general context of ES niching methods, preferably more methods should be found. In particular, we refer to an extension as a method which will be proficient at forming niches with population distribution proportionate to the fitness values of the peaks, satisfying the chi-square-like performance criterion, discussed earlier.

Bibliography

[1] D.H. Ackley. *A connectionist machine for genetic hillclimbing.* Kluwer, Boston, 1987.

[2] Th. Bäck, G. Rudolph, and H.P. Schwefel. Evolutionary Programming and Evolution Strategies: Similarities and Differences. In D.B. Fogel and W. Atmar (eds.): *Proceedings of the second Annual Conference on Evolutionary Programming* (pp. 11-22), La Jolla, CA: Evolutionary Programming Society 1993.

[3] Th. Bäck. Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. *Proceedings of the First IEEE Conference on Evolutionary Computation 1994*: (1993) 57-62.

[4] Th. Bäck. *Evolutionary algorithms in theory and practice.* Oxford University Press, New York, 1996.

[5] D. Beasley, D.R. Bull, and R.R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation 1*(2), 101-125.

[6] D. Dasgupta and D.R. McGregor. Nonstationary function optimization using the structured genetic algorithm. In R. Männer & B. Manderick (Eds.), *Parallel problem solving from nature*, 2 (pp. 145-154). Amsterdam: Elsevier, 1992.

[7] K.A. De Jong. *An analysis of a behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975. Dissertations Abstracts International, 36(10), 5140(B), University Microfilms No. 76-9381.

[8] K. Deb and D.E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In Schaffer, J.D. (Ed.), *Proceedings of the third International Conference on Genetic Algorithms* (pp. 42-50). San Mateo, CA: Morgan Kaufmann, 1989.

[9] R. Fletcher and M.J.D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163-168, 1963.

[10] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, 1966.

[11] D.E. Goldberg and J. Richardson. Genetic Algorithms with Sharing for multimodal function optimization. In Grefenstette, J.J. (Ed.), *Proceedings of the second International Conference on Genetic Algorithms* (pp. 41-49). Hillsdale, NJ: Lawrence Erlbaum Associates, 1987.

[12] D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69-93. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[13] D.M. Himmelblau. *Applied nonlinear programming*, McGraw-Hill, New-York, 1972.

[14] J.H. Holland. *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press, 1975.

[15] S.W. Mahfoud. Crowding and Preselection revisited. In R. Männer & B. Manderick (Eds.), *Parallel problem solving from nature*, 2 (pp. 27-36). Amsterdam: Elsevier, 1992.

[16] S.W. Mahfoud. *Niching Methods for Genetic Algorithms.* PhD Thesis, University of Illinois at Urbana-Champaign, 1995. IlliGAL Report No. 95001.

[17] B.L. Miller and M.J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96). IEEE; New York, NY, USA, 1996.

[18] I. Rechenberg. *Evolutionsstrategies: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart, 1973.

[19] H.P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary systems research.* Birkhäuser, Basel, 1977.

[20] The Documentation for the Genetic and Evolutionary Algorithm Toolbox: `http://www.geatbx.com/ver_3_5/`