RECEIVED

FEB 20 1996

OSTI

The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications

Walter Cedeño
PhD Thesis

September 1995

The logo of the Lawrence Livermore National Laboratory, featuring a stylized 'LL' symbol to the left of the text 'Lawrence Livermore National Laboratory' which is arranged in four lines.

Lawrence
Livermore
National
Laboratory

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This report has been reproduced
directly from the best available copy.

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (615) 576-8401, FTS 626-8401

Available to the public from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.,
Springfield, VA 22161

The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications

Walter Cedeño

**Doctor of Philosophy
Thesis**

Manuscript date: September 1995

MASTER

LAWRENCE LIVERMORE NATIONAL LABORATORY
University of California • Livermore, California • 94551

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

dlc

The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications

By

Walter Cedeño

B.S. (University of Puerto Rico, Rio Piedras) 1982

M.S. (University of Puerto Rico, Rio Piedras) 1985

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree
of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY of CALIFORNIA

DAVIS

Approved:

Approved: *V. Lao Huan*
Mrs. M. Blatter
John Lee

Committee in Charge

1995

Acknowledgments

Behind every successful effort there is a team. There are many individuals that helped me during all these years. First, I would like to mention the names of individuals from the University of California and the Lawrence Livermore National Laboratory that in one way or another helped me achieve my goal. I would like to thank my advisor and friend Prof. Rao Vemuri for his guidance and support during all these years. His advise and positive criticism helped me grow as a researcher and professional. Thanks to Dr. John Feo and my colleagues in the Computing Research Group for lending me a hand and providing a great environment for doing part of this work. Thanks to Prof. Meera Blattner for her many suggestions and for serving on the committee. Thanks to Dr. David O. Nelson for his suggestions and ideas on the theoretical aspects of this work. Thanks to Dr. Elbert W. Branscomb and Mr. Thomas R. Slezak from the Human Genome Center for giving the opportunity to work on the DNA mapping problem. Thanks to Virginia Johnson for describing the intricacies of the SUTRA application and providing us with a model for the Aquifer Management problem.

To my professors from the University of Puerto Rico, Rio Piedras, specially Dr. Oscar Moreno and Dr. Italo J. Dejter, "Muchas Gracias". They gave me the foundation to succeed and much needed encouragement to continue my graduate work. Thanks to my friends from the Research

Scientific Computing department at Sterling Winthrop, specially Dr. David J. Balaban and Dr. Jerry Verlin, for providing me their input and support.

Special thanks to my father Julio Cedeño-Morales and my mother Sylvia Rivera-Rivera for all their hard work in providing me with a good education and keeping me focussed during my younger years. Mom and Dad, I am very proud to be your son and be able to pass on your genetic material and your high moral principles to my children. By your blessing, I am sure, each of them will excell in their own selected niche!

I dedicate this dissertation to my wife Sara E., my twin sons Walter Jr. and Wilbert, and my daughter Waleska M. for putting their needs aside and giving me the opportunity to continue. For all those long weekends and week nights that I could not be with you, thank you. I dedicate this work to you because you are my heart, my soul, and the major reason to continue graduate studies.

Finally, I want to thank the countless number of friends for their support. Their encouragement gave us the strength to endure until we found the light at the end of the tunnel.

This work was supported, in part, by the Applied Mathematics Program of the Office of Energy Research (U.S. Department of Energy) under contract number W-7405-Eng-48 to the Lawrence Livermore National Laboratory.

Table Of Contents

1. INTRODUCTION	1
2. OVERVIEW OF GENETIC ALGORITHMS	10
2.1 THE CLASSICAL GENETIC ALGORITHM	11
2.2 THE STEADY STATE GENETIC ALGORITHM	15
2.3 GENETIC ALGORITHM THEORY	18
2.3.1 Schemata	18
2.3.2 The Schema Theorem	20
2.3.3 The Infinite Population Model	25
2.3.4 The Finite Population Model	26
2.4 GENETIC ALGORITHMS FOR MULTIMODAL SEARCH	27
2.4.1 Preselection	29
2.4.2 Crowding	29
2.4.3 Sharing	31
2.4.4 Deterministic Crowding	36
2.4.5 Fitness Derating	36
2.4.6 Subpopulation Schemes	37
2.4.7 Restricted Tournament Selection	38
2.4.8 Distributed and Parallel Approaches	39
3. THE MULTI-NICHE CROWDING GA	42
3.1 OVERVIEW	42
3.2 TEST FUNCTIONS	47
3.3 CROSSOVER AND MUTATION	49
3.4 PERFORMANCE METRICS	52
3.5 MNC GA PARAMETERS	54
3.6 RESULTS	56
3.7 EMPIRICAL ANALYSIS OF MNC GA PARAMETERS	64
3.7.1 Effect of Population Size	65
3.7.2 Effect of Crowding Selection Size	66
3.7.3 Effect of Crossover Probability	68
3.7.4 Effect of Mutation Probability	68
3.7.5 Effect of Interval size	71
3.7.6 Effect of Crowding Factor	72
3.7.7 Effect of Crowding Size	75
3.8 OTHER SAMPLE FUNCTIONS F_5	77
3.9 COMPARISON WITH OTHER METHODS	82
3.10 DYNAMIC LANDSCAPES	86
3.10.1 Background	87
3.10.2 Experimental Design	89
3.10.3 Results and Discussion	92
3.11 SUMMARY	94
4. MATHEMATICAL ANALYSIS OF THE MULTI-NICHE CROWDING GA	96
4.1 OVERVIEW	96
4.2 NOTATION	99
4.3 MNC GA CROWDING SELECTION: DIVERSIFYING THE MATING POOL	100
4.3.1 Example Population: Selection	106
4.4 MNC GA MATING: INTERVAL CROSSOVER AND MUTATION	108
4.4.1 Example Population: Mating	110

4.5 MNC GA REPLACEMENT: ENCOURAGING SPECIATION AND NICHING	111
4.5.1 <i>Example Population: WAMS Replacement</i>	115
4.6 THE MARKOV MATRIX: BRINGING EVERYTHING TOGETHER	118
4.6.1 <i>Example Population: The State Transition Probabilities</i>	121
4.7 A CLOSER LOOK AT CROWDING SELECTION	122
4.8 A CLOSER LOOK AT WORST AMONG MOST SIMILAR REPLACEMENT	125
4.9 COMMENTS ON THE GENERATIONAL MNC GA	128
5. APPLICATION OF THE MNC GA TO DNA MAPPING	131
5.1 BACKGROUND ON THE DNA RESTRICTION-FRAGMENT ASSEMBLY PROBLEM	132
5.2 RESTRICTION FRAGMENT DATA FOR CHROMOSOME 19	135
5.3 RELATED WORK AND SCOPE OF PRESENT WORK	138
5.4 PROBLEM REPRESENTATION	141
5.4.1 <i>Chromosome Encoding</i>	144
5.4.2 <i>Fitness Function</i>	144
5.4.3 <i>Mating and Mutation Operators</i>	148
5.4.4 <i>Similarity Function</i>	150
5.5 RESULTS	151
5.6 DISCUSSION AND COMMENTS	157
5.7 FRAGMENT DATA AND MATRICES FOR DATA SET 2	160
6. APPLICATION OF THE MNC GA TO WATER QUALITY CONTROL	161
6.1 INTRODUCTION	161
6.2 THE SIMULATION ENVIRONMENT	162
6.2.1 <i>The Aquifer</i>	163
6.2.2 <i>The SUTRA Model</i>	164
6.3 INTERFACING WITH THE MULTI NICHE CROWDING GENETIC ALGORITHM	167
6.3.1 <i>Chromosome Representation</i>	168
6.3.2 <i>Recombination</i>	169
6.3.3 <i>Replacement</i>	170
6.3.4 <i>Similarity Measure</i>	171
6.3.5 <i>Fitness Measure</i>	172
6.4 RESULTS	174
6.4.1 <i>Scenario 1: Two Objective Functions</i>	175
6.4.2 <i>Scenario 2: Three Objective Functions</i>	178
6.5 DISCUSSION AND COMMENTS	180
7. APPLICATION OF THE GENERATIONAL MNC GA TO SOLVE THE FILE DESIGN PROBLEM	185
7.1 INTRODUCTION	186
7.2 THE GENETIC ALGORITHM MODEL	190
7.3 GENETIC OPERATORS	192
7.3.1 <i>Gene Encoding</i>	193
7.3.2 <i>Similarity Metric</i>	193
7.3.3 <i>Mating Operator</i>	194
7.3.4 <i>Fitness Function</i>	196
7.4 EXPERIMENTAL DATA	199
7.5 RESULTS	200
7.6 SUMMARY	203
8. CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH	204
9. BIBLIOGRAPHY	211
10. CURRICULUM VITAE	219

List of Tables

TABLE 2.1: THE SELECTION STEP: CREATION OF MATING POOL USING FPR.....	13
TABLE 2.2: THE MATING STEP: APPLICATION OF CROSSOVER AND MUTATION	15
TABLE 3.1: PEAK PARAMETERS FOR FUNCTIONS $F_4(X,Y)$ AND $F_5(X,Y)$	48
TABLE 3.2: FUNCTION SPECIFIC PARAMETERS USED IN THE MNC GA	55
TABLE 3.3: LOCATION, WIDTH, AND HEIGHT FOR THE PEAKS IN FUNCTIONS $F_{5.1}, F_{5.2}, F_{5.3},$ AND $F_{5.4}$	77
TABLE 3.4: MNC GA PARAMETERS USED FOR EACH FUNCTION.	83
TABLE 3.5: COMPARISON OF PARALLEL HILLCLIMBING (PH), FITNESS DERATING (FD), SHARING (SH), DETERMINISTIC CROWDING (DC), AND MULTI-NICHE CROWDING (MNC) ON FUNCTIONS $F_1, F_2,$ AND F_3	84
TABLE 3.6: PARAMETERS OF THE 10 PEAKS USED FOR ALL TESTS OF THE MNC GA IN A DYNAMIC LANDSCAPE.	91
TABLE 4.1: SAMPLE STATE TRANSITION PROBABILITIES IN THE MNC GA.	121
TABLE 6.1: SAMPLE SOLUTIONS WITH TWO OBJECTIVES.....	176
TABLE 6.2: SAMPLE SOLUTIONS WITH THREE OBJECTIVES.....	179
TABLE 7.1: POSSIBLE CONFIGURATIONS FOR 12 RECORDS IN 2 FILES OF SIZE 6. THE ATTRIBUTES VALUES ARE {B, C, F, V} AND HAVE {2, 7, 1, 2} RECORDS RESPECTIVELY.....	187
TABLE 7.2: CONFIGURATION FOR ALL TEST CASES.....	199
TABLE 7.3: SOLUTION FOUND BY THE MNC GA TO TEST CASES AND THE NUMBER OF GENERATIONS NEEDED	201

List of Figures

FIGURE 2.1: CLASSICAL GENETIC ALGORITHM	12
FIGURE 2.2: STEADY STATE GENETIC ALGORITHM	17
FIGURE 2.3: INVERTED SHEKEL'S FOXHOLES.....	31
FIGURE 2.4: TEST FUNCTIONS $F_1(x)$ AND $F_2(x)$	33
FIGURE 3.1: WORST AMONG MOST SIMILAR (WAMS) REPLACEMENT POLICY	43
FIGURE 3.2: STEADY STATE MULTI-NICHE GENETIC ALGORITHM.....	45
FIGURE 3.3: TEST FUNCTIONS F_4 , "2-NEEDLES IN A HAYSTACK", (SCALED VERSION ON LEFT) AND F_5 (RIGHT).....	47
FIGURE 3.4: REGION (DELINEATE BY OUTER SQUARE WITH DASHES) FROM WHERE OFFSPRING IS SELECTED WHEN USING INTERVAL CROSSOVER DURING MATING.....	50
FIGURE 3.5: NICHE COUNT AND AVERAGE FITNESS FOR NICHES IN FUNCTION F_2	56
FIGURE 3.6: NICHE COUNT AND NICHE AVERAGE FITNESS FOR PEAKS IN FUNCTION F_1	57
FIGURE 3.7: NICHE COUNT AND AVERAGE FITNESS FOR THE PEAKS IN FUNCTION F_4	58
FIGURE 3.8: NICHE COUNT AND AVERAGE FITNESS FOR PEAKS IN FUNCTION F_5	60
FIGURE 3.9: MAXIMUM FITNESS ON EACH NICHE FOR FUNCTION F_5	61
FIGURE 3.10: NICHE COUNT AND NICHE AVERAGE FITNESS FOR PEAKS IN FUNCTION F_3	63
FIGURE 3.11: EFFECT OF DIFFERENT VALUES FOR THE CROWDING SELECTION SIZE (C_s) ON THE NICHE COUNT OF EACH PEAK. NICHE COUNT FOR VALUES OF 5 (TOP LEFT), 10 (TOP RIGHT), 20 (BOTTOM LEFT), AND 25 (BOTTOM RIGHT) ARE SHOWN ABOVE.	67
FIGURE 3.12: EFFECT OF MUTATION PROBABILITY ON THE NICHE COUNT FOR FUNCTION F_5 . THE RESULTS SHOWN HERE ARE FOR MUTATION PROBABILITY VALUES OF 0.001 (TOP LEFT), 0.01 (TOP RIGHT), 0.05 (BOTTOM LEFT), AND 0.1 (BOTTOM RIGHT).	69
FIGURE 3.13: EFFECT OF MUTATION PROBABILITY OF THE NICHE MAXIMUM FITNESS FOR FUNCTION F_5 . THE RESULTS SHOWN HERE ARE FOR MUTATION PROBABILITY VALUES OF 0.001 (TOP LEFT), 0.01 (TOP RIGHT), 0.05 (BOTTOM LEFT), AND 0.1 (BOTTOM RIGHT).	70
FIGURE 3.14: NICHE MAXIMUM FITNESS AVERAGED OVER 20 RUNS FOR DIFFERENT VALUES OF THE INTERVAL CROSSOVER SIZE ϵ . STARTING AT THE TOP LEFT WE HAVE RESULTS FOR A SIZE OF 10, IN THE TOP RIGHT RESULTS FOR A SIZE OF 100, IN FIGURE 3.9 FOR A SIZE OF 1,000, IN THE LOWER LEFT RESULTS FOR A SIZE OF 10,000, AND IN THE LOWER RIGHT RESULTS FOR A SIZE OF 100,000.	72
FIGURE 3.15: NICHE COUNT FOR CROWDING FACTOR VALUES OF 2 (LEFT) AND 6 (RIGHT).	73
FIGURE 3.16: NICHE MAXIMUM FITNESS FOR ALL PEAKS OF FUNCTION F_5 USING CROWDING FACTOR VALUES OF 2 (UPPER LEFT), 4 (UPPER RIGHT), 5 (LOWER LEFT), AND 6 (LOWER RIGHT).	74
FIGURE 3.17: NICHE COUNT FOR DIFFERENT VALUES OF CROWDING SIZE FOR THE NICHES OF FUNCTION F_5 . VALUES OF 5 (TOP LEFT), 10 (TOP RIGHT), 20 (BOTTOM LEFT), AND 25 (BOTTOM RIGHT) ARE SHOWN.	75
FIGURE 3.18: SAMPLE FUNCTIONS $F_{5,1}$ (TOP LEFT), $F_{5,2}$ (TOP RIGHT), $F_{5,3}$ (BOTTOM LEFT), AND $F_{5,4}$ (BOTTOM RIGHT).	78
FIGURE 3.19: NICHE COUNT FOR FUNCTIONS $F_{5,1}$ (TOP LEFT), $F_{5,2}$ (TOP RIGHT), $F_{5,3}$ (BOTTOM LEFT), AND $F_{5,4}$ (BOTTOM RIGHT).	79
FIGURE 3.20: SHOWN ABOVE ARE THE NICHE MAXIMUM FITNESS FOR FUNCTIONS $F_{5,1}$ (TOP LEFT), $F_{5,2}$ (TOP RIGHT), $F_{5,3}$ (BOTTOM LEFT), AND $F_{5,4}$ (BOTTOM RIGHT).	81
FIGURE 3.21: NICHE COUNT (TOP) AND NICHE MAX FITNESS (BOTTOM) FOR TEST CASE 2 USING A VALUE OF $G = 10$	94
FIGURE 4.1: PSEUDO CODE OF STEADY STATE MNC GA USED FOR ANALYSIS.	97
FIGURE 4.2: MEMBERS OF Ω FOR STRING LENGTH $L = 3$	99
FIGURE 4.3: INTERVAL OF VECTOR P ACCOUNTED BY E_{AB} WITH CHROMOSOMES HAVING A LOWER DISTANCE TO CHROMOSOME J THAN CHROMOSOME K	103
FIGURE 4.4: NUMBER OF INDIVIDUALS IN THE POPULATION WITH SIMILAR RANKING (K) AND LOWER RANKING (R) FOR CHROMOSOMES IN THE SAMPLE POPULATION.	106
FIGURE 4.5: MATE PROBABILITY UNDER CROWDING SELECTION FOR A VALUE OF $C_s = 2$	107

FIGURE 4.6: PROBABILITY THAT OFFSPRING WITH TAG VALUE 6 IS GENERATED AFTER APPLYING INTERVAL CROSSOVER AND BIT MUTATION, WITH PARAMETERS $\chi = 0.9$, $\mu = 0.1$, AND $\varepsilon = 2$, FOR ANY CHROMOSOME PAIR (J, K).....	110
FIGURE 4.7: NUMBER OF INDIVIDUALS IN POPULATION WITH EQUAL (J) AND LOWER SIMILARITY RANKING FOR ALL CHROMOSOMES IN Ω	115
FIGURE 4.8: GRAPH OF SAMPLE FITNESS FUNCTION. TWO MAXIMA AT 0 AND 6.	116
FIGURE 4.9: CROWDING PROBABILITY DURING REPLACEMENT WITH $s = 2$	117
FIGURE 4.10: WAMS REPLACEMENT PROBABILITY FOR $C_F = 2$	118
FIGURE 4.11: EXPECTED RANK VALUE, VARIANCE, AND STANDARD DEVIATION OF THE MATE UNDER CROWDING SELECTION AS THE FUNCTION OF THE CROWDING SELECTION SIZE USING A POPULATION SIZE $N = 100$	124
FIGURE 4.12: PROBABILITY DISTRIBUTION FOR THE RANK VALUES UNDER CROWDING SELECTION USING A POPULATION OF 100.....	125
FIGURE 5.1: PHYSICAL MAP FOR ISLAND USING FRAGMENTS FROM A SET OF OVERLAPPING CLONES.	138
FIGURE 5.2: COSMID CLONES WITH FRAGMENT DATA.	142
FIGURE 5.3: AN EXAMPLE OF FRAGMENT ASSEMBLY.....	144
FIGURE 5.4: THE M MATRIX ON THE LEFT WHOSE ENTRIES ARE THE NUMBER OF FRAGMENT MATCHES AND THE E MATRIX ON THE RIGHT WHOSE ENTRIES ARE THE TOTAL ERRORS BETWEEN ANY TWO CLONES WITH $E = 10$	145
FIGURE 5.5: MODIFIED GENETIC EDGE RECOMBINATION FOR CLONE SEQUENCING.....	149
FIGURE 5.6: CLONE SEQUENCES OBTAINED BY GA AND ACTUAL SEQUENCE.	154
FIGURE 5.7: POPULATION AVERAGE FITNESS (LEFT) AND BEST FITNESS (RIGHT) FOR DIFFERENT METHODS.....	156
FIGURE 6.1 SUTRA NODES AND ELEMENTS IN MESH.....	166
FIGURE 6.2: USE OF RANKING IN WORST AMONG MOST SIMILAR REPLACEMENT.....	171
FIGURE 6.3: WELL CONFIGURATIONS FOR FOURTH (TOP) AND FIFTH (BOTTOM) SOLUTIONS IN TABLE 1.....	183
FIGURE 6.4: WELL CONFIGURATIONS FOR THE FIFTH (TOP) AND SIXTH (BOTTOM) SOLUTIONS IN TABLE 2. ..	184
FIGURE 7.1: ENCODING FOR THE FILE DESIGN PROBLEM.....	193
FIGURE 7.2: USING SIMILARITY TO SELECT A MATE DURING CROWDING SELECTION.....	193
FIGURE 7.3: MATING OPERATOR FOR THE FILE DESIGN PROBLEM.....	195
FIGURE 7.4: MNC GA EXECUTION TIME IN SECONDS FOR 50 GENERATIONS.....	202

Walter Cedeño
September 1995
Computer Science

The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications

Abstract

The ability of organisms to evolve and adapt to the environment has provided mother nature with a rich and diverse set of species. Only organisms well adapted to their environment can survive from one generation to the next, transferring on the traits, that made them successful, to their offspring. Competition for resources and the ever changing environment drives some species to extinction and at the same time others evolve to maintain the delicate balance in nature.

In this disertation we present the *multi-niche crowding* genetic algorithm, a computational metaphor to the survival of species in ecological niches in the face of competition. The multi-niche crowding genetic algorithm maintains stable subpopulations of solutions in multiple niches in multimodal landscapes. The algorithm introduces the concept of *crowding selection* to promote mating among members with similar traits while allowing many members of the population to participate in mating. The algorithm uses *worst among most similar replacement* policy to promote competition among members with similar traits while allowing competition among members of different niches as well.

We present empirical and theoretical results for the success of the multi-niche crowding genetic algorithm for multimodal function optimization. The properties of the algorithm using different parameters are examined. We test the performance of the algorithm on problems of DNA Mapping, Aquifer Management, and the File Design Problem. Applications that combine the use of heuristics and special operators to solve problems in the areas of combinatorial optimization, grouping, and multi-objective optimization. We conclude by presenting the advantages and disadvantages of the algorithm and describing avenues for future investigation to answer other questions raised by this study.

1. Introduction

The ability of organisms to evolve and adapt to their environment by means of natural selection has provided mother nature with a diverse set of species. This foundation, which is part of modern evolutionary thinking, was laid by Charles Darwin after the publication of his work "On the Origin of Species by Means of Natural Selection". Only organisms well adapted to their environment can survive from one generation to the next, transferring on the traits that made them successful to their offspring. Competition for resources between organisms and the ever changing environment drives some species to extinction and at the same time others evolve to maintain the delicate balance in nature. It is through this interaction between nature and organisms, that species containing favorable traits for a given environment emerge. In this work we apply the same principles present in nature to create a genetic algorithm that evolves a population of mathematical solutions containing different categories of solutions adapted to niches in a multimodal environment.

In optimization problems over multimodal landscapes, techniques that make use of the gradient information to search for the global optimum usually get stuck at one of the local optima near the location of the first guess. These gradient based techniques work with a single solution and must be applied several times with different starting points to be able to cover the

search space as best as possible. In many problems the gradient information is not readily available, prompting the need for better techniques. Additionally, there exist many problems where the location of the best K optima are needed in order to compare different answers and point out further experimentation.

The benefits of an approach that can locate multiple optima and maintain them throughout the search are many. Consider a dynamic environment where the optima are constantly changing, a technique that can locate and maintain multiple optima can inform the user when the current configuration is no longer the best based on the parameters in the environment. In other cases abnormal situations may cause changes in the current configuration; having viable alternatives at hand can allow for a more smoother transition to the new configuration. An approach that can use a set of solutions to locate multiple optima is more practical for these types of environments.

This study is about the Multi-Niche Crowding genetic algorithm. A technique based on natural selection and genetic recombination that uses a set of solutions, called a *population*, to locate and maintain multiple optima in a multimodal landscape. This technique is able to eliminate some of the drawbacks of other genetic algorithm approaches to evolve different *species* in the niches. These species emerge naturally in the population to create a

stable set of subpopulations in the multiple niches in the multimodal landscape.

The Multi-Niche Crowding genetic algorithm uses the concept of *crowding selection* to promote mating among members having similar traits and allow many members of the population to participate in mating. This allows members of the same niche to participate in mating more often and preserve those traits that define their species. At the same time mating between different species may occur giving rise to new species.

The algorithm also uses *worst among most similar replacement* to promote competition among members with similar traits belonging to a niche while allowing competition among members of different niches as well. This replacement technique accomplishes two things. First, by promoting competition among members of the same species in a niche it applies the *survival of the fittest* rule that is so prevalent in nature. Only those that are fit to their environment can survive for many generations, thus allowing the species to evolve to their best potential within their niche. Second, by allowing competition between different species as well, those species that are a better fit for their environment tend to occupy more slots in the overall population.

The second chapter of this dissertation gives an overview of genetic algorithms. We introduce the terminology, describe the Classical Genetic

Algorithm and the Steady State Genetic Algorithm, and point out the differences between them. We present the basic theory describing the behavior of genetic algorithms. For this we introduce the concept of *schemata* and how it is exploited by the genetic algorithm. By modeling the effect of schemata on the population we develop what is known as the Schema Theorem, the fundamental theorem of genetic algorithms. Then we introduce other theoretical models of genetic algorithms that deviate from the use of schemata. The Infinite Population Model is described, how it models the trajectory of the search in a genetic algorithm, and what its implications are. In the same manner we describe the Finite Population Model which uses an exact Markov Chain model of the genetic algorithm to analyze its stochastic behavior. We end this chapter by describing other genetic algorithm approaches to locate multiple optima in a multimodal landscape and their drawbacks.

In the third chapter we describe in detail our proposed technique called the Multi-Niche Crowding genetic algorithm and how this approach overcomes the drawbacks of other genetic algorithms. We describe the selection and replacement operators called crowding selection and worst among most similar replacement respectively, the motivation for using these operators, and how they work in harmony to locate and maintain multiple optima in the population. Then a set of multimodal test functions to test the technique are defined. We introduce a crossover operator, called *interval*

crossover, and a mutation operator used in conjunction with these test functions. Then, a set of metrics to measure different performance aspects of the technique are defined. Parameters for the algorithm are given together with an explanation for their effect during the search. The results on these test functions is described using the metrics defined previously. These results show the superior performance of the Multi-Niche Crowding genetic algorithm, over other techniques, for multimodal function optimization. To analyze the effect that different parameters have on the behavior of the algorithm we evaluate its performance using different values for each parameter. We then test the technique using other functions to verify the properties exhibited by the approach are not dependent on the previous functions. Finally, the properties of the approach for multimodal function optimization are summarized, including its advantages and disadvantages.

The fourth chapter presents a mathematical analysis of the steady-state version of the Multi-Niche Crowding genetic algorithm using a Markov Chain model. To accomplish this we first define the notation used throughout the chapter. We then analyze each step, namely, selection, mating, and replacement, of the genetic algorithm individually. For selection we define a probability function that computes the probability that any two members of the population are selected for mating using crowding selection. For the mating step we define a probability function that computes *the probability* of generating a given offspring generated from two members of the population

after applying interval crossover and mutation. For the replacement step we define a probability function that computes the probability that a given offspring replaces a member of the population under worst among most similar replacement. Given these three probability functions describing each step we define a function for the transition probability between any two populations after one iteration of the algorithm. This transition probability function is then used to define the Markov chain transition probability matrix. With this matrix defined, we can then examine the asymptotic behavior of the algorithm for simple sample functions. Throughout the chapter we use a simple sample function to verify the probability functions defined for each step and to show the behavior of the algorithm. We end this chapter with a more detailed analysis of the selection and replacement steps of the algorithm and describing how the same analysis can be applied to the generational counterpart of the Multi-Niche Crowding genetic algorithm.

Now we turn our attention to the practical aspects of the Multi-Niche Crowding genetic algorithm. In the next three chapters we present results of the technique on problems with high complexity and having practical application in other areas. Specifically we applied the technique to the areas of DNA mapping, water quality control, and database design. For each application we concentrate on the ability of the technique to solve the problem by locating multiple optima in different search spaces.

The fifth chapter describes the application of the Multi-Niche Crowding genetic algorithm to the DNA restriction fragment map assembly problem. This is believed to be a NP Hard problem. Here the main goal is to reconstruct a piece of DNA from fragment data obtained from fingerprinting. Here we describe in detail the features of the problem and its importance to the Human Genome Project. We briefly summarize other approaches and their drawbacks. We then give a detailed description of the genetic operators used for this problem. We start with the definition of the encoding used. Then we define the fitness function which is based on the information from overlapping fingerprint DNA data. We explain why this fitness function was selected over others and other ways of improving it. The genetic operators for the problem are then described, their importance in this application, and what their impact is on the properties of our approach. We end this chapter with a discussion of the performance of the technique for this problem and identification of different areas where it can be improved.

The sixth chapter presents the results of the Multi-Niche Crowding genetic algorithm to a problem in multi-objective optimization arising in the context of water quality control. This is a very complex problem where we are trying to locate a set of pumping wells in and around a contaminated aquifer so that the pollutants can be removed effectively and before they leave the region. Here, again we describe the application domain and the importance of this problem for aquifer decontamination. We describe previous work in this

area and some of their drawbacks. Then we describe suitable genetic operators used for this problem, why they were chosen over others, and the impact they had on the performance of the algorithm. Additionally we describe a way to use ranking, during replacement, as a way to evolve configurations that are optimal for different objectives. This eliminates the need for a fitness function that groups multiple objective with different weighting values, thus eliminating biases implicitly added by the weight parameters. The results obtained by the technique are then evaluated. Finally, some comments are made about the suitability of the approach for multi-objective optimization and improvements to the current model.

The seventh chapter describes the application of the Multi-Niche Crowding genetic algorithm to the file design problem. A problem that is usually encountered during the design of databases. In this problem the main goal is to design a database composed of multiple files so that when queries are made a minimum number of files are accessed. This problem is NP Hard and requires the optimization over conflicting objectives. Additionally we investigate the performance of a generational version of the approach by implementing a parallel version of the algorithm. We start the chapter by describing the problem at hand, its complexity, and its importance in database design. Then we introduce the generational model used for the application together with the encoding, genetic operators, and fitness function for the problem. Here we also describe the use of heuristics in the

genetic operators and how they improve the performance of the algorithm. We finish this chapter by summarizing the results we obtained and pointing out ways to improve the application.

The last chapter of this dissertation summarizes the properties of the Multi-Niche Crowding genetic algorithm. We also mentioned the advantages over other methods and its disadvantages. More important, we give several conclusions we reached as part of this research. Finally, we point out future areas of research that were identified while doing this work.

2. Overview of Genetic Algorithms

Genetic algorithms (GAs) are general purpose search procedures based on the principle of natural selection and genetic recombination. Their first implementation is attributed to Holland (1975) at the University of Michigan during the mid 60's. During the past several years genetic algorithms have been applied to diverse types of problems ranging from scheduling to computational biology (Schaffer, 1989; Belew and Booker, 1991; Forrest, 1993; and Eshelman, 1995).

Genetic algorithms imitate nature by using the mechanics of evolution and natural selection to improve a set of initial solutions called a *population* using "recombination" and "mutation" of the "genetic material". During the process the GA insures that better solutions mate more often by assigning them a higher chance of recombination. New solutions are created and inserted into the current family of solutions; old and ineffective solutions are allowed to perish. Only the most effective solutions survive. Selection of the candidates for recombination and mutation, and the replacement of solutions in the population are driven by genetic operators governed by probability.

Genetic algorithms differ from traditional optimization techniques in several ways. GAs work with several solutions at the same time and improve them by recombining their good features while exploring new solutions in the search space. This approach has the advantage of permitting good solutions

to guide the algorithm to near optimal solutions while at the same time exploring different areas of the search space. Working with populations allows the GA to escape local optima more easily than approaches using single solutions. The genetic operators are governed by probability and the methodology of using them is problem independent; the method and the programs implementing them can be easily applied to many problems with only minor adjustments. Solutions are represented with a simple encoding (usually with binary strings) which allow easy manipulation and fast execution of the genetic operators. GAs use an objective function to determine the quality of a solution as the only information to guide the search, no derivatives or auxiliary information is needed. This property makes it possible to apply GAs to a large set of problems where traditional methods are not suitable.

Section 2.1 formally introduces what is called the classical genetic algorithm; how it works, the operators involved in the process, and the terminology used. Section 2.2 introduces the steady state genetic algorithm and compares it with the classical GA. Section 2.3 gives an overview of the theoretical results for the classical GA. Finally, Section 2.4 describes different approaches to make a GA suitable for multimodal landscapes.

2.1 The Classical Genetic Algorithm

The mechanics of the classical GA, also known as the simple GA (SGA), are easy to understand. A pseudo code of the algorithm is shown in Figure

2.1. A possible solution to a problem is referred to as an *individual* (or *organism*) which consists of one or more *chromosomes* (usually one) suitably encoded usually using binary strings. Each position (or bit, in the case of binary encoding) in the chromosome is the *locus* which can take on different values called *alleles* (0 or 1 for binary strings). The SGA starts by randomly generating a population of n individuals. *Fitness* of these individuals is evaluated using a *fitness function* that measures the quality of the solution. Better solutions get higher scores, with the global optima having the highest value. For simplicity all further discussion uses binary alleles, unless otherwise noted.

```

Generate initial population of size  $n$  at random.
Evaluate initial population.
For gen = 1 to MAX_GENERATIONS
    Create mating pool. Select  $n$  individuals from the population using
        fitness proportionate reproduction.
    Form  $n/2$  pairs of individuals and apply crossover and mutation with
        probability  $\chi$  and  $\mu$  respectively.
    Replace current population with the  $n$  offspring.
end for
Output fittest individual as the solution to the problem

```

Figure 2.1: Classical Genetic Algorithm

Given an initial population, the computational steps of *selection*, *mating*, and *replacement* are applied for an indicated number of generations or until another terminating condition is satisfied. During selection, a *mating pool* is formed by selecting n individuals from the population according to their fitness values. Those with higher fitness are selected more often than individuals with lower fitness. The probability P_i of selecting an individual I_i

whose fitness is F_i , is given by the ratio of the individual fitness to the total fitness of all the individuals in that population, i.e., $P_i = F_i / \sum_{j=1}^n F_j$. Table 2.1 shows a sample population of four individuals, their assumed fitness and the probability of selection for mating. This selection technique is known as *fitness proportionate reproduction* (FPR) or *roulette wheel selection*. As a consequence of FPR high fitness individuals in the population have a better chance to participate in mating and therefore pass their genetic material to their offspring.

Table 2.1: The Selection Step: Creation of Mating Pool using FPR

Number	Individual	Fitness (F_i)	Selection Probability (P_i)	Mating Pool
1 = I_1	10000	81	81/158 = 0.51	10000
2 = I_2	00101	10	10/158 = 0.06	01011
3 = I_3	01011	31	31/158 = 0.20	10000
4 = I_4	01100	36	36/158 = 0.23	01100
Average		39.5		
Total		158	1.00	

Notice that Individual 1 was selected into the mating pool more than once because of that individual's higher fitness being approximately twice as much as the population's average fitness. After selection, the individuals selected are divided randomly into pairs for mating. Using the information in the last column of Table 2.1 the two pairs may consist of Individual 1 with Individual 3 (10000, 01011) and Individual 1 with Individual 4 (10000, 01100). For each of the pairs *one-point crossover* and *mutation* are applied

with probability χ and μ respectively. During this crossover operation, one of the interstitial positions is selected uniformly at random. The *offspring*, i.e., new strings, are created by swapping the alleles on, say, the right side of the crossover point between the individuals. For example, suppose the pair I_1 and I_4 are selected for mating and that the interstitial position 3 from the right of the string is chosen at random (indicated by the | symbol). The new strings after crossover are:

$$I_1 = 10|000 \Rightarrow 10100$$

$$I_4 = 01|100 \Rightarrow 01000$$

The new strings are created by swapping the bits to the right of the symbol | in both strings. When χ is less than 1.0 some of the individuals survive the crossover operation unchanged (i.e. they do not experience crossover). Whether or not crossover is experienced each resulting individual is subjected to mutation with probability μ . That is, during mutation each allele in the individual is changed with probability μ . When an allele is mutated, its value is changed from 0 to 1 or from 1 to 0. Mutation allows genetic material that might otherwise be lost from the population using crossover alone to reappear. The value of μ is selected low to prevent the algorithm from resembling random search. Usually μ is chosen so that there is about one mutation per thousand bits (Goldberg, 1989). Table 2.2 shows a possible population after crossover with $\chi = 0.6$ and mutation with $\mu = 0.025$.

Following mating, the *replacement step* replaces all the individuals in the population with the new offspring obtained after mutation. The new population is composed of the individuals (10001, 01011, 10100, 01000) as shown in the fifth column of Table 2.2. After replacement the new population is used for the next generation until the maximum number of iterations is reached or some other terminating condition is satisfied.

Table 2.2: The Mating Step: Application of Crossover and Mutation

Mating Pool	Fitness	Offspring after Crossover	Fitness	Offspring after Mutation	Fitness
<i>I1</i> 10000	81	10000	81	10001	82
<i>I3</i> 01011	31	01011	31	01011	31
<i>I1</i> 10000	81	10100	90	10100	90
<i>I4</i> 01100	36	01000	27	01000	27
Average	57.25		57.25		57.5
Total	229		229		230

The individual with the best fitness after the last generation is returned as the solution for the problem. The simplicity of the algorithm and the low overhead of the genetic operators makes GAs suitable for application to a broad range of problems. The classical GA is easy to parallelize and does not require any more information about the problem than the fitness function.

2.2 The Steady State Genetic Algorithm

The steady state GA (SSGA) was introduced independently, first by Whitley (1988) and later on by Syswerda (1989). In the SSGA only a subset of the individuals in the population, called the *generation gap* (De Jong,

1975), are replaced in every generation. This generation gap is often expressed as a fraction in the interval $[0.0, 1.0]$. The mating pool size and the number of offspring generated in every generation is equal to the value of $G * n$, where G denotes the generation gap and n the population size. The selection and mating steps are applied as in the SGA. During replacement, only those individuals $G * n$ at the bottom of the fitness ladder are replaced with the offspring. The SSGA is equivalent to the SGA when the value of G is set to 1.0, i.e., when the entire population is replaced in every generation. In the simplest case, $G = 2/n$, two individuals are selected using FPR, then crossover and mutation are applied to generate two offspring. The two lowest fitness individuals in the population are replaced with the offspring. In this scenario the new offspring are part of the population and are ready for selection the next time around. The classical GA and the steady state GA with $G = 2/n$ are analogous to the Jacobi (simultaneous replacement) and Gauss-Siedel (successive replacement) methods respectively in numerical analysis. In general when we refer to the SSGA we are assuming the simple case with $G = 2/n$. Figure 2.2 shows the pseudo code for a steady state GA.

The manner in which individuals are chosen from the population for replacement, so that the offspring can be inserted into vacated slots, differs from method to method. The simplest way is to eliminate the two lowest fitness individuals from the population. Another method, called *inverse ranking* (Syswerda, 1989), has been used for the replacement step above. In

inverse ranking the population is sorted and starting with the lowest fitness member in the population, two individuals are selected for deletion with probability $v > 1/n$. With this operator, low fitness individuals are more likely to be deleted but can still be parents if they are lucky.

```

Generate initial population of size  $n$  at random.
Evaluate initial population.
For  $gen = 1$  to  $MAX\_TRIALS$ 
    Select 2 individuals from the population using FPR.
    Apply crossover and mutation with probability  $\chi$  and  $\mu$  respectively.
    Replace 2 individuals from the population with offspring.
end for
Output fittest individual as the solution to the problem.

```

Figure 2.2: Steady State Genetic Algorithm

The steady state GA is closer to what happens in nature than the classical GA. Low fitness individuals disappear faster from the population. Higher fitness individuals survive for more than one generation. The steady state GA has been shown to sample the search space (Whitley, 1988) faster than the classical GA but at the same time tends to exhibit what is known as *premature convergence*, where the highly fit individuals quickly take over the entire population. Inserting the offspring in the population right after they are generated allows highly fit offspring to rapidly contribute to the genetic pool.

A drawback of the steady state GA is that it is not easy to parallelize. By applying all three genetic operators sequentially the algorithm works with only one population and therefore the selection, mating, and replacement

steps cannot be applied in parallel for each pair of individuals. Nevertheless, the steady state GA is used by many researchers in the field.

2.3 Genetic Algorithm Theory

In this section we briefly survey some theoretical models of the SGA. Specifically we present the results of Holland (1975), Vose (1992), and Nix and Vose (1992). These works will be used as a basis for the analysis of MNC GA presented later on. Using their results we describe the problems exhibited by the classical GA. The reader is referred to the actual source for all the details.

2.3.1 Schemata

A *schema* (Holland, 1975) over the binary alphabet $A = \{0, 1\}$ is a string composed of the symbols in A plus an additional don't care symbol $*$. Consider the schema $11**0$ of length 5. This matches the four strings $\{11000, 11010, 11100, 11110\}$. That is, this particular schema matches all binary strings of length 5 starting with two 1's and ending with one 0. The schema $*****$ matches all binary strings of length 5 and 11111 is the schema representing the binary string of all 1's. Thus a schema can be thought of as a template comprised of the symbols from an alphabet A and the don't care symbol $*$.

There are two important properties that must be defined for schemata. The *order* of a schema S , denoted $o(S)$, and the defining *length* of a schema S ,

denoted $\delta(S)$. The order of a schema $o(S)$ is the number of fixed positions, i.e., non * positions, in S . For example, 11111 is a schema of order 5, while **1** is a schema of order 1. Note that the value of $o(S)$ is in the interval $[0, l]$, where l is the number of positions in the string. The higher the order, the more specific the schema.

The defining length of a schema $\delta(S)$ defines the number of crossover positions between the two outermost fixed positions in S . For example, the length of the schema 11*01 is 4, while the length for the schema ****1 is 0 since the two outermost fixed positions are the same, namely digit 1. The length of a schema is a value in the interval $[0, l-1]$. The schema length $\delta(S)$ defines the compactness of the schema and it's different from the string length l which refers to the number of positions in the schema.

In general, there are $(k + 1)^l$ schemata for strings of length l in an alphabet of size k . Each string of length l in A is matched by exactly 2^l schemata and a schema with r don't care symbols matches exactly k^r strings.

As we will see in the next section the GA samples the schemata in the population giving and increasing number of mating slots to highly fit schemata. We will also show the effect of crossover and mutation on any particular schema.

2.3.2 The Schema Theorem

The *Schema Theorem* (Holland, 1975) gives a lower bound on the effect of the genetic operators from one generation to the next in the classical GA. Recall that the classical GA, described in Section 2.1, consists of the following repeated steps; selection, crossover, mutation, and replacement. To be able to predict I^{t+1} , the population at time $t + 1$, we look at the influence of the genetic operators on the schemata of I^t , the population at time t .

In the first step, selection, individuals are chosen for mating according to the ratio of their fitness to the total fitness of the population. Let F_i be the fitness of individual I_i and

$$F^t = \sum_{i=1}^n F_i$$

the fitness of the population, then the probability of mating for individual I_i is

$$P_i = \frac{F_i}{F^t}.$$

To expand the notion of selection probability to schema, it is only necessary to define the fitness of a schema S . The fitness of a schema S at time t is defined as the average fitness of all the individuals matched by the schema. That is,

$$F_S = \sum_{i=1}^{\phi} \frac{F_i}{\phi_S^t}, \text{ for } I_i \in S.$$

Here ϕ_S^t is the number of individuals matching schema S in the population at time t . Then schema S has probability

$$P_S = \frac{F_S}{F^t}$$

to be selected.

After selection takes place we expect to have

$$n \cdot \phi_S^t \cdot \frac{F_S}{F^t}$$

individuals belonging to schema S in the mating pool prior to crossover. This result is obtained from the fact that schema S has ϕ_S^t individuals in I^t with an average probability P_S of being selected for any of the n slots in the mating pool. If we define the average fitness of the population

$$\bar{F}^t = \frac{F^t}{n}$$

then we have

$$\phi_S^{t+1} = \phi_S^t \cdot \frac{F_S}{\bar{F}^t}$$

Notice that the number of individuals matching schema S after selection grows as the ratio of the schema average fitness over the population average fitness. This means that schema with above average fitness tends to receive more mating opportunities than below average schema.

From the value of ϕ_S^{t+1} we can also observe the long term effect of above average schema in the population. If we assume that a schema S remains above average in succeeding generations, then the number of individuals belonging to the schema will increase exponentially. This is a cause for *premature convergence*, where an above average individual drives the population towards a local minimum. This situation is not rare and is a direct result of using FPR.

During the second step of the SGA crossover is applied to pairs of individuals from the mating pool. Then mutation is applied to the resulting offspring from crossover. Now we will examine the effect of these two genetic operators on the expected number of schemata ϕ_S^t . As discussed previously, crossover chooses uniformly at random a position in the chromosome and swaps the information to the right of this location between a pair of individuals. Since the mating pairs are chosen at random any two schema can undergo crossover.

For a moment, let's assume that two individuals selected for crossover are representative of the following schemata; $S_1 = (11***)$ and $S_2 = (01**0)$, respectively. Suppose the second crossover position is selected. Then the offspring resulting from crossover of $(11*|**)$ with $(01*|*0)$ will belong to the schemata $(11**0)$ and $(01***)$ respectively. One of the offspring will still

belong to schema S_1 , since 11**0 is contained in it, but the other offspring may or may not belong to the schema S_2 , because 01*** is not contained in S_2 .

As can be observed, schemata with large defining length are more probable of being destroyed than schema with shorter defining length. In general, crossover is applied with probability χ and the crossover site is selected uniformly at random over the $l - 1$ positions in the string. Therefore the probability of destruction of a schema S is

$$PD_S = \chi \cdot \frac{\delta(S)}{l-1},$$

and consequently, the probability of schema survival is

$$PS_S = 1 - \chi \cdot \frac{\delta(S)}{l-1}.$$

Notice also that there is still a chance for the schema to survive when the information passed from the other individual contain the same digits as the fixed positions in the schema. In the example above, if the individual belonging to S_1 also had a 0 in its last digit then the schema S_2 will survive the crossover. In the same manner two individuals not belonging to schema S_2 can produce an offspring in S_2 . For example, (01001) and (11100) with the crossover position 1 will produce the offspring (01000) and (11101), with the first one belonging to S_2 . For this reason the value of PS_S is a lower bound estimate of the resulting survival probability:

$$PS_S \geq 1 - \chi \cdot \frac{\delta(S)}{l-1}.$$

Combining the effect of crossover with the expected schema count obtained we get the equation:

$$\varphi_S^{t+1} \geq \varphi_S^t \cdot \frac{F_S}{F^t} \cdot [1 - \chi \cdot \frac{\delta(S)}{l-1}]$$

This equation captures the expected count for schema S in the next generation after selection and crossover are applied.

Mutation is the next operator to be considered. Mutation works by randomly changing an allele in the chromosome with probability μ . It is easy to see that a schema S will only be affected if one of its fixed positions is modified. We know that the order of a schema defines the number of fixed positions in it. Therefore the probability that a schema S survives mutation is

$$PS_S = (1 - \mu)^{o(S)}.$$

For small values of μ ($\mu \ll 1$), the survival probability due to mutation may be approximated by the expression

$$PS_S = 1 - \mu \cdot o(S).$$

Adding the effect of mutation on schema count we get the equation

$$\varphi_S^{t+1} \geq \varphi_S^t \cdot \frac{F_S}{F^t} \cdot [1 - \chi \cdot \frac{\delta(S)}{l-1}] [1 - o(S) \cdot \mu].$$

As shown, the addition of mutation changes our previous equation a little. Our previous observations about above average schema still apply. The next step, replacement does not change the equation since the new offspring replace the entire population. Ignoring the small cross product terms in the equation we may conclude that for a particular schema S the expected number of individuals belonging to S after selection, crossover, and mutation is given by the following equation:

$$\phi_S^{t+1} \geq \phi_S^t \cdot \frac{F_S}{\bar{F}^t} \cdot [1 - \chi \cdot \frac{\delta(S)}{l-1} - o(S) \cdot \mu].$$

This equation is known as the Schema Theorem, or the Fundamental Theorem of Genetic Algorithms. The implications of the theorem are very important; schemata with small defining length, low order, and above average fitness receive exponentially increasing trials in subsequent generations of the classical GA.

2.3.3 The Infinite Population Model

Vose (1991) created a model for the simple genetic algorithm where the exact distribution of the population, a vector x whose entries contain the number of chromosomes in the population for each possible chromosome in the search space, from generation to generation is given by a function $\zeta(x)$. Unlike the schema theorem, the function $\zeta(x)$ determines the exact distribution of the population on any given generation (Vose, 1993) and can be used to study the trajectory of the search in the SGA. The results obtained

with this model are based on deterministic equations and compute the evolutionary path of the SGA with infinite size populations.

He defined populations as points in a smooth "GA-surface" and analyzed the behavior of the simple GA on this surface. Their work describes population trajectories and provides a geometric interpretation of genetic search. An important result of their work is that they characterize the long and short term behavior of the genetic algorithm for large populations. They found that the short term behavior of the simple GA is to converge to the local minima of the surface that contained the initial population. These local minima correspond to the fixed points of $\zeta(x)$. The long term behavior of the population is to converge to the local minima with the largest basin of attraction. The basin of attraction of a local minima are all those populations which drive the SGA towards the local minima.

2.3.4 The Finite Population Model

Nix and Vose (1992) constructed an exact model of the simple GA with finite population using Markov chain analysis. Their main result is that the trajectory followed by a finite population in the simple GA is related to the evolutionary path predicted by the infinite population model. They also found that the SGA converges to the steady state distribution of the Markov chain. These steady states correspond to the local minima on the "GA-surface". They also proved that the steady state distribution of the finite population model

concentrates probabilistically near the fixed points of the infinite population model.

Other researchers have used Markov chain analysis (Davis and Principe, 1991; Rudolph, 1994) to prove different properties of the SGA or variants of it. Most results agree that the SGA has the property of an ergodic Markov chain, i.e., all the states of the chain have nonzero probability. Therefore the algorithm can be in any state at any given time regardless of the initial population. Later on, we will also use Markov chain analysis to examine the properties of the *multi-niche crowding* GA.

2.4 Genetic Algorithms for Multimodal Search

Finding the optimum in unimodal search spaces is an important optimization problem for which many techniques exist. When applied to multimodal search spaces, (i.e., cases where the function to be optimized exhibits several peaks) these techniques tend to converge to only one of the optima, more often to one of the local optima in the neighborhood of the first trial. Although many of the optimization problems require only the location of the global optimum, there are many other applications where the location of other local optima and their values are of interest.

Here we introduce the notion of *species*, individuals with common characteristics, and *niches*, subdomains of the search space, and how GAs can benefit from introducing niching and speciation into their operators. By

encouraging the formation of niches and species, GAs can be made to converge to more than one optimum in a multimodal search space.

In this section we describe and compare various modifications to the classical GA and the steady state GA to make them suitable for multimodal search spaces. Both the classical and steady state methods suffer from premature convergence due to the effect of FPR on schema selection which drives the algorithm to converge to a local optimum. Assigning exponential number of mating trials to above average schema in a complex multimodal search space does not allow a good amount of exploration of the search space. Due to this problem and the "deceptiveness" exhibited by multimodal search spaces (Goldberg et. al., 1992) the classical GA and the steady state GA are not appropriate for multimodal search. The term *deception* was introduced by Goldberg (1987b) to represent the class of problems that are misleading to GAs in a schema average sense.

Now we describe modifications to the genetic operators of the classical GA and the steady state GA in order to render them suitable for multimodal landscapes. We are interested in the effect of these modifications on convergence rate, algorithm complexity, convergence to global optimum, convergence to multiple optima, and the ability to maintain diversity in the population. We will show empirically that these are important issues that the GA must address to be successful in multimodal search spaces. A GA that is deficient in any of these areas would be useless in complex multimodal

search spaces. We conclude this chapter with a description of other attempts to make GAs suitable for multimodal function optimization. We describe the drawbacks of these techniques based on the issues described above.

2.4.1 Preselection

Preselection (Cavichio, 1970) is one of the modifications to the replacement step of a classical GA. In preselection not all the generated offspring are chosen for the new population. Only the offspring with higher fitness than their parents, replace their parents in the next generation. Cavichio claims to have maintained a more diverse population with relatively small population sizes ($n = 20$). According to its author, (a) the new replacement step did not add any significant amount of complexity to the SGA, and (b) the convergence of the algorithm was improved over the classical GA by keeping only the most fit individuals after mating. Like the other traditional GAs, this stratagem does not keep stable species subpopulations for many generations and it only converges to one optimum.

2.4.2 Crowding

Crowding (De Jong, 1975) is a generalization of preselection where offspring replace similar individuals from the population. In crowding, selection and reproduction are the same as in the SGA; but replacement is different. For concreteness, it is assumed that two parents are selected to produce two offspring. Two members of the population are identified to make room for these offspring. The policy of replacing a member of the present

generation by an offspring is carried out as follows, in two steps. First, a group of CF individuals is selected at random from the population. CF , called the *crowding factor*, indicates the size of the group. De Jong had success with multimodal functions using values of $CF = 2$ and $CF = 3$. Second, the bit strings in the offspring chromosome are compared with those of the CF individuals in the group using the Hamming distance, which measures the number of differing bits, as a measure of similarity. The group member that is most similar to the offspring is now replaced by the offspring. This process is repeated for the other offspring as well. This second offspring can conceivably replace its sibling that just entered the population pool, although such scenario is rather unlikely.

De Jong used the 25-peak multimodal function shown in Figure 2.3 to test his crowding model (a mirror image of this in the XY-plane is called the Shekel's foxholes function). The function is given by the formula:

$$F(x,y) = 0.002 + \sum_{i=1}^{25} \frac{1.0}{(x - X_i)^6 + (y - Y_i)^6 + i},$$

where (X_i, Y_i) denotes the location of peak i . The peaks are located on the grid intersections formed by the values

$$X = [-32, -16, 0, 16, 32] \text{ and}$$

$$Y = [-32, -16, 0, 16, 32].$$

Figure 2.3: Inverted Shekel's Foxholes

All 25 peaks have the same base width. The height of peak i is given by the expression $0.002 + 1/i$, where the peaks are located at

$$(-32, -32), (-16, -32), \dots, (32, 32).$$

Using this function, De Jong obtained nearly optimal performance in locating the global optima, with a value of $CF = 2$. Replacing individuals by similar ones allows the population to maintain more diversity for more generations. Crowding does not add any significant complexity to a SGA; in most cases it permits escape from local optima, and slows down premature convergence. The only limitation is that it does not converge to multiple solutions and after many generations one of the peaks takes over.

2.4.3 Sharing

Goldberg and Richardson (1987) used the *sharing* concept of Holland (1975) as a way of reducing the selection pressure caused by FPR. In sharing,

the fitness value of an individual is adjusted downwards according to the number of individuals in its neighborhood or niche. The more individuals in a niche, the more pressure they create on each other thus causing a lowering of fitness. The downwards adjustment of the fitness of an individual is done with the help of a suitably defined sharing function. This function takes into account the proximity of all the members in the population in the immediate neighborhood of the individual. Therefore, when many individuals belong to the same niche, their fitness are derated giving more allocations to individuals in other niches.

The sharing function assumes a value between 0 and 1 for any distance value d_{ij} between any two individuals I_i and I_j in the population. The distance between two individuals measures their similarity. The lower the distance the more similar the individuals. A common measure of distance for binary strings is the Hamming distance. Consider the sharing function

$$Sh(d_{ij}) = 1 - d_{ij} / \sigma_{share},$$

called the triangular function, defined for individuals in the neighborhood of σ_{share} , otherwise

$$Sh(d_{ij}) = 0.$$

As indicated above the fitness of an individual is derated by dividing it by the sum of all the sharing function values contributed by the individuals in

the population. That is, before selection in each generation, the fitness of each individual I_i is adjusted with the equation

$$F'_i = F_i / \sum_{j=1}^n Sh(d_{ij}).$$

This step takes $O(n^2)$ operations after each generation increasing the complexity of the SGA significantly.

Figure 2.4: Test Functions $F1(x)$ and $F2(x)$

In their work, Goldberg and Richardson apply sharing to the functions;

$F_1(x) = \sin^6(5.1\pi x + 0.5)$, having 5 peaks of equal height and

$F_2(x) = \exp^{-4(\ln 2)(x-0.0667)^2 / 0.64} F_1(x)$, with 5 peaks of un-equal height

as shown in Figure 2.4 ($F_1(x)$ on the left, $F_2(x)$ on the right). Both functions were maximized using binary chromosome encoding for numbers in the interval $0 \leq x \leq 1$. In both cases the GA with sharing was able to maintain stable subpopulations and diversity in the population. Convergence was good, but not all the peaks in F_1 had the individuals distributed close to the

top. It is not clear that in F_1 sharing will be able to maintain proportional number of individuals for larger number of generations.

Another drawback with sharing is that it is not problem independent. To define the neighborhood of an individual, the user must have some knowledge of the search space so that the neighborhood defined by the sharing function contains only a single peak. The authors also suggested a more general approach to sharing based on comparisons of individuals at the *genotypic* level (i.e., the encoded parameter space, usually binary strings) instead of the *phenotypic* level (i.e., the decoded parameter space). They defined a family of power law sharing functions of the form

$$Sh(d_{ij}) = 1 - (d_{ij} / \sigma_{share})^\alpha,$$

where d_{ij} / σ_{share} measures the relative bit difference between individuals I_i and I_j .

To improve the convergence within the different niches and improve the stability of the subpopulation in the sharing method Deb and Goldberg (1989) introduced *phenotypic sharing* in which the similarity between two individuals is computed in the decoded parameter space, thus allowing for a better metric for distance in the search space. They also introduced the concept of *mating restriction*, where only individuals belonging to the same niche were allowed to mate.

As mentioned above, phenotypic sharing and mating restriction did improve convergence of the subpopulation at the peaks and maintained stable subpopulations in the peaks. One drawback from doing this, is that it becomes harder to form other niches due to the restriction imposed in mating. Only mutation can form another niche in an unexplored area of search space and there must be at least two individuals to prevent the mutation induced niche from disappearing in the next generation.

In the work done by Yin and Gerny (1993) the concept of cluster analysis was applied with sharing to reduce the complexity of the algorithm. In their work, during each generation, an adaptive clustering algorithm groups the individuals in the population into a number of clusters. The number of clusters formed depends on the population diversity and the parameters of the clustering algorithm. These clusters are then used to compute the sharing value and the derated fitness for each individual in the population. Clustering reduces the calculation of the sharing values to an algorithm of $O(n)$. By adding clustering to a SGA using sharing reduces the complexity of the algorithm while at the same time allows the niches to form naturally. On the other hand the algorithm exhibits the same behavior as sharing alone in the sense that the parameters of the clustering algorithm are problem dependent.

2.4.4 Deterministic Crowding

In *deterministic crowding* (Mahfoud, 1992) selection pressure is eliminated and preselection is introduced to obtain a very fast GA suitable for multimodal search spaces. In this method the selection pressure induced by FPR is eliminated by allowing individuals to mate at random with any other individual in the population. Pressure is applied, however, during the replacement step using preselection. Toward this goal, each of the two offspring is first paired with one of the parents; this pairing is not done randomly, rather the pairings are done in such a manner that the offspring is paired with the most similar parent. Then each offspring is compared with its paired parent and the individual with the higher fitness is allowed to stay in the population and the other is eliminated.

Deterministic crowding does not add any complexity to the classical GA. It works well in multimodal search spaces. It does not restrict mating to elements of the same niche, thus allowing new niches to form. It is not clear if multiple solutions can be maintained for many generations (Mahfoud, 1993), although it appears that this method is able to sustain stable subpopulations, and therefore diversity, for many generations.

2.4.5 Fitness Derating

Fitness derating (Beasley et. al., 1993) allows unimodal optimization methods to be applied to multimodal problems by using the knowledge gained in a run to avoid re-searching the same area of the search space. In

fitness derating, a classical GA is first applied to the problem at hand. The solution obtained is used to derate the fitness values of the individuals in the neighborhood of the solution. This allows other solutions to be found in each subsequent application of the algorithm. This technique may be used with other optimization methods as well.

The authors used sharing functions to derate the values at every peak solution found so far. They modified the fitness function after each run by adding a derating term to lower the fitness values of individuals around a radius selected by the user. The additional term and the repetitive application of the algorithm makes the complexity of fitness derating similar to that of sharing.

As in sharing, fitness derating can succeed only with knowledge from the search space. When a derating function is defined, the fitness of all individuals around a radius, selected by the user, is derated. A low radius may form artificial peaks, while a large radius may derate fitness of an undiscovered peak. Nevertheless when good knowledge of the search space is known, fitness derating can be applied successfully.

2.4.6 Subpopulation Schemes

In this class of methods diversity is maintained by creating subpopulations (Spears, 1994) in a classical GA using tag bits. A similar tag number identifies individuals in the same subpopulation. Mating and selection is restricted to individuals belonging to the same subpopulation, i.e.

having the same tag value. This method replaces the concept of distance between individuals with labels identifying them as members of the same subpopulation. It also uses fitness sharing during each generation, but uses the number of individuals within each subpopulation as the niche count. This approach reduces the complexity of the sharing scheme while obtaining similar results.

For example, the method, called *Simple Subpopulation Scheme 1* (SSS1), is not able to sustain multiple solutions in lower peaks for many generations. Spears tried another variation (SSS2) on this method by introducing topology to the population to restrict mating even further to individuals in the neighborhood. The SSS2 approach was able to maintain more individuals at lower peaks. It is not clear if stable subpopulations can be maintain for many generations.

2.4.7 Restricted Tournament Selection

Restricted tournament selection (Harik, 1995) modifies the selection and replacement steps in the SSGA to eliminate the selection pressure caused by FPR and promote competition between members of the same niche. In this approach selection is done at random. Both parents are selected at random from the population. After crossover and mutation two offspring are generated and inserted in the population in the following manner. For each offspring a group of w individuals are chosen from the population at random with replacement. Then the individual in the group which is most similar to

the offspring is selected. The offspring replaces the chosen individual in the population if its fitness is higher, otherwise the offspring is eliminated. As we will see later this approach is a special case of our multi-niche crowding GA with replacement done greedily.

The major flaw of this approach is its greedy nature. This method only allows replacement when the offspring fitness is higher than that of the chosen individual to be replaced. This greedy stratagem prevents possible new areas of the search space to be discovered, a necessary property when applying the technique to a dynamic landscape. Although the method is successful when applied to simple functions, the way replacement is done will guarantee that smaller peaks will lose its members.

2.4.8 Distributed and Parallel Approaches

In this section we describe other methods that have exhibited good properties for multimodal search. They were not applied to multimodal search spaces directly, but the authors observed the formation of niches during the execution of the algorithm. After a number of generations the methods described below did converge only to one solution. Compared to a classical GA, these methods were able to maintain a more diverse population for longer number of generations thus avoiding convergence to local optima more often.

A method called *distributed* GA (Tanese, 1989) divides the population into several equal size subpopulations that are mapped into separate processors.

The classical GA is applied to each subpopulation locally. Periodically genetic material migrates between the subpopulations. This technique is very similar to the subpopulation scheme described above. The main differences are the use of migration between subpopulations and the absence of fitness sharing. In a distributed GA the subpopulations are configured as a binary n -cube. Migration occurs only between the immediate neighbors in the cube. As mentioned earlier, niches form during the initial generations of the algorithm, but after many generations all subpopulations converge to a single peak that takes over the entire population.

The other methods are based on a fine grain parallel GA (Manderick and Spiessens, 1989; Mühlenbein, 1989; Davidor, 1991). In this model each individual in the population is assigned to a processor in a parallel system organized as a square matrix. During each generation, each individual performs a random walk around the neighboring individuals to select a mate. After crossover and mutation, the best offspring replaces the individual at the processor. In this model, niches formed naturally because of the restriction imposed on selecting localized mates. If the GA is run for many generations the niche with highest fitness will eventually take over the population. This algorithm also maintains a diverse population for many generations and is able to escape local optima easily.

Another parallel approach using a *dynamic division algorithm* (Elo, 1994) divides the population into an increasing number of subpopulations

dynamically. This approach divides the population when the following conditions are met: (a) the population has not converged, (b) it has at least 40 individuals, and (c) there is an individual sufficiently different than the best one in the population and its fitness is adequate. A bounding box delineating the subpopulation (the whole population initially) is used to evaluate if the subpopulation has converged, if not it is divided based on its size and the fitness and similarity of another individual with respect to the best one in the subpopulation. The author obtained good results using a symmetric function. The conditions for dividing a subpopulation are complex and problem dependent. It is not clear how well the approach will do on non-symmetric functions and with fewer number of processors.

3. The Multi-Niche Crowding GA

This chapter describes the *multi-niche crowding* (MNC) genetic algorithm and presents the results on the optimization of static and dynamic multimodal functions (Cedeño and Vemuri, 1992). The main purpose in this chapter is to present the central idea and describe the behavior of the algorithm when applied to different landscapes. The main objective during each test case is to observe the ability of the algorithm in escaping local optima, maintaining stable subpopulations in different niches, and converging to the global optima.

3.1 Overview

In the MNC GA both the selection and replacement steps are modified with some type of crowding. The idea is to eliminate the selection pressure caused by FPR and allow the population to maintain diversity throughout the search. This objective is achieved in part by encouraging mating and replacement within members of the same niche while allowing some competition for the population slots among the niches. The result is an algorithm that (a) maintains stable subpopulations within different niches, (b) maintains diversity throughout the search, and (c) converges to different local optima. No prior knowledge of the search space is needed and no restrictions are imposed during selection and replacement thus allowing

exploration of other areas of the search space while converging to the best individuals in the different niches.

In MNC, the FPR selection is replaced by what we call *crowding selection*. In crowding selection each individual in the population has the same chance for mating in every generation. Application of this selection rule is done in two steps. First, a parent I_i is selected for mating. This selection can be either sequential or random. Second, its mate I_j is selected, not from the entire population, but from a group of individuals of size C_s (crowding selection group size), picked uniformly at random (with replacement) from the population. The mate I_j thus chosen must be the one who is the most “similar” to I_i . The similarity metric used here is not a genotypic metric such as the Hamming distance, but a suitably defined phenotypic distance metric.

Figure 3.1: Worst among most similar (WAMS) replacement policy

Crowding selection promotes mating between members of the same niche while still allowing individuals from different niches to mate. Unlike mating

restriction that only allows individuals from the same niche to mate, crowding selection allows some amount of exploration to occur while at the same time looking for the best individual in each niche.

During the replacement step, MNC uses a replacement policy called *worst among most similar* (WAMS). The goal of this step is to pick an individual from the population for replacement by an offspring. Implementation of this policy follows these steps. First, C_r "crowding factor groups" are created by randomly picking s (crowding group size) individuals per group from the population. Second, one individual from each group that is most similar to the offspring is identified. This gives C_r individuals that are candidates for replacement by virtue of their similarity to the offspring. The offspring will replace one of them. From this group of most similar candidates, we pick the one with the lowest fitness to die and be replaced by the offspring. Figure 3.1 shows a pictorial view of this replacement policy.

After the offspring becomes part of the population it competes for survival with other individuals when the next offspring is inserted in the population. In WAMS replacement offspring are likely to replace low fitness individuals from the same niche. It can also happen that it replaces a high fitness individual from the same niche or an individual from another niche. This allows a more diverse population to exist throughout the search. At the same time it promotes competition between members of the same niche and between members belonging to different niches. A similar technique was

used by Goldberg (1989) in classifier systems but he replaced the most similar individual out of a group of lowest fitness candidates.

```

Generate initial population of size  $n$  at random.
Evaluate initial population.
For gen = 1 to MAX GENERATIONS
  For  $i = 1$  to  $n$ 
    Use crowding selection to find mate for parent  $i$ 
    Mate and mutate
    Insert offspring in population using WAMS replacement

```

Figure 3.2: Steady state Multi-Niche Genetic Algorithm

Both the selection and replacement steps in the MNC are primarily based on a similarity metric. Fitness is also considered during replacement to promote competition between members of the same niche. Competition between members of different niches occurs as well. The pseudocode in Figure 3.2 summarizes the salient features of the steady state MNC GA.

The MNC GA can be applied using either a generational approach (Cedeño, 1992) or a steady state approach (Cedeño and Vemuri, 1993). Either way the behavior of the algorithm is the same. Niches form naturally and subpopulations are maintained throughout the run. Only convergence rate seems to be slightly adversely affected; but, more results are needed to verify this observation.

In the tests that follow the MNC GA converges consistently to the global optimum. The complexity added by the selection and replacement operators to the GA is dependent on the values of C_s and C_r * s. Using Figure 3.2 and

assuming that mating and mutation take constant time the number of comparisons between the individuals in MNC GA is given by the expression.

$$\text{MAX_GEN} * n * (C_s + C_f * s).$$

The additional operations are a result of forming the various groups from where similar individuals are selected during selection and replacement. The equation above also reflects the number of times the function that calculates the similarity between individuals is called. Keeping this function as simple as possible is important to keep the extra computational burden to a minimum. In most complex applications, as we will see later, the evaluation of the fitness function is the highest computational burden. In those cases the overhead caused by the similarity function evaluations is small relative to the time taken by the whole run.

A good rule of thumb for selecting the value of these constants is to set C_f to a value in the interval $[2, 4]$, and C_s and s to be at least two times the number of global optima one wants to maintain during the run. For example, to maintain 5 global optima a good set of values is: $C_f = 3$, $C_s = 15$ and $s = 10$. Later on when we go over analysis of the MNC GA we will examine in more detail the crowding selection and WAMS replacement policies to learn more about the effect of these parameters on niching and convergence.

3.2 Test Functions

To evaluate the performance of the MNC model we used five different test functions. Three of these functions are taken from previous work. Functions $F_1(x)$ and $F_2(x)$, shown in Figure 2.4, were described in Section 2.4.3. They correspond to the five-optima sine functions used by Goldberg in his work with sharing. Function $F_3(x,y)$, shown in Figure 2.3, called “Shekel’s foxholes” with twenty five optima, was used by De Jong in his work with crowding (Section 2.4.2).

Figure 3.3: Test functions F_4 , “2-needles in a haystack”, (scaled version on left) and F_5 (right).

We also considered other functions not exhibiting the symmetry present in the above functions. Function $F_4(x,y)$, called “2-needles in a haystack”, shown on the left in Figure 3.3, contains two optima with the same height and width but located far apart. Function $F_5(x,y)$, the sample shown on the right in Figure 3.3, contains five optima with height, width, and location

chosen at random in every run. Both of these functions are defined by the equation

$$F(x,y) = \sum_{i=1}^{np} H_i / [1 + W_i [(x - X_i)^2 + (y - Y_i)^2]],$$

where np indicates the number of peaks in the function, (X_i, Y_i) the coordinates of peak i , H_i the height of peak i , and W_i determines the width at the base of peak i . Note that the lower the value of W , the wider the base of the peak. The parameters for the peaks in functions $F_4(x,y)$ and $F_5(x,y)$ are shown in Table 3.1. The height of function $F_4(x,y)$, shown on Figure 3.3, was increased 10 times the actual height to get a better graph of the function.

Table 3.1: Peak parameters for functions $F_4(x,y)$ and $F_5(x,y)$

Function	Peak Location	Width	Height
$F_4(x,y)$ $0 \leq x,y \leq 65,535$	(45k, 2k)	0.0004	100
	(15k, 62k)	0.0004	100
$F_5(x,y)$ $0 \leq x,y \leq 65,535$	(17.1, 34.4)	1.9	50.5
	(8.7, 4.1)	0.17	44.8
	(20.3, 11.0)	3.6	87.1
	(38.3, 47.9)	3.1	51.5
	(57.9, 54.1)	3.7	55.5

The parameters for function $F_5(x,y)$ were chosen uniformly at random. The location of the peak, (X, Y) , was obtained by choosing values for X and Y in the interval $[0.0, 65.535]$. The height of the peak, H , was obtained by choosing a value in the interval $[0.0, 100.0]$. Finally, the width of the peak,

W , was obtained by choosing a value in the interval $[0.0, 5.0]$. Overlapping peaks were not allowed.

3.3 Crossover and Mutation

For reasons that are discussed below, the crossover operator used during mating is called *interval crossover*. In interval crossover only one offspring is generated. For each pair of parent chromosomes, x_1 and x_2 , (assume without loss of generality that the cardinal value of binary string x_1 is less than the cardinal value of binary string x_2), the offspring's chromosome is selected at random from the interval $[x_1 - \epsilon/2, x_2 + \epsilon/2]$. The value for ϵ is usually small, less than 0.001% of the highest cardinal value for the binary string. This allows the offspring to move outside the boundaries delineated by their parents. For this test ϵ was set at the decimal value 2,000.

We obtained better results using interval crossover over single-point crossover, due in part to the disruptive effect on niche preservation caused by single-point crossover. For example, consider the individuals 0100000000000 and 0011111111111. If these individuals are part of the same peak, a single-point crossover may conceivably generate the offspring pair 0000000000000 and 0111111111111 which may not belong to the same peak as the parents. Consider also the schema $S_1=10^{****}$ and $S_2=01^{****}$. Assume that S_1 and S_2 are the schema for two different niches. After single-point crossover it is conceivable that the offspring will belong to the schema 11^{****} and 00^{****} ,

neither of them preserving the phenotypic characteristics of their parents. The same effect was observed by Mahfoud (1993) on his work with deterministic crowding.

On the other hand interval crossover will generate offspring close to the parents and more likely the offspring will belong to the same niche. The success of interval crossover is due, in part, to crowding selection. In crowding selection the parent and its mate are likely to be near each other in the search space. Since both individuals are from the same neighborhood, interval crossover is restricted to a smaller region of the search space.

Figure 3.4: Region (delineate by outer square with dashes) from where offspring is selected when using interval crossover during mating.

There is a similar technique called *line crossover* (Michalewicz, 1992). It is a special case of interval crossover with $\epsilon = 0$ and only selecting individuals in the line defined by the two parents. Interval crossover is a more general

operator and the region from where the offspring is chosen at random depends on the number of chromosomes defining the genome in the parents. The parents denote a line when the genome contains 1 chromosome, a rectangle when the genome contains 2 chromosomes, a box when the genome contains 3 chromosomes, and so on.

Suppose we apply interval crossover to a pair of individuals with a pair of chromosomes each. Let (x_1, y_1) and (x_2, y_2) be the two parents (see Figure 3.4). Remember that each chromosome value x_i or y_i is a nonnegative integer in the range from 0 to $2^l - 1$. Let $minx = \text{Minimum}(x_1, x_2)$, $maxx = \text{Maximum}(x_1, x_2)$, $miny = \text{Minimum}(y_1, y_2)$, and $maxy = \text{Maximum}(y_1, y_2)$. The resulting offspring will have chromosomes (x, y) where $x \in [minx - \epsilon/2, maxx + \epsilon/2]$ and $y \in [miny - \epsilon/2, maxy + \epsilon/2]$. Note that (x, y) is a point in the rectangle with lower left corner $(minx, miny)$ and upper right corner $(maxx, maxy)$. This rectangle is shown in Figure 3.4 using dotted lines.

Interval crossover is an appropriate operator to apply when the chromosome is a linear transformation of the phenotype. This property assures that individual parents from a given niche are likely to produce offspring that also belong to the same niche. The region delineated by the parents in the genotype space maps to a similar region in the phenotype space. This is true for multimodal optimization problems where each coordinate in the search space will be represented with a binary chromosome.

Mutation is applied as in the SGA. Every bit in each chromosome is mutated with probability μ . The only difference is that mutation is applied only to the offspring resulting from interval crossover. If crossover does not take place between two parents (this occurs when the crossover probability $\chi < 1.0$) neither will mutation. This is contrary to the SGA where mutation is applied to the parents when crossover does not occur. This means that the probability for crossover indirectly affects the probability for mutation. Unlike the SGA, in the MNC GA individuals in the population have a chance of surviving from one generation to the next and therefore are not taken as offspring if crossover does not take place.

3.4 Performance Metrics

The metrics used to evaluate the MNC GA are intended to capture the stability of the subpopulations, convergence properties of the algorithm, and the competition between the individuals from different niches. They also show how species form naturally at different niches in the search space. We measured, for all the test cases presented in Section 3.1, the number of individuals within the neighborhood of each optima and called it the *niche count*. An individual is part of a niche if it is within a region that encompasses 90% of the optima's fitness value. This region represents the upper 90% of a peak. Individuals not belonging to any of the niches were grouped together and considered as a separate niche, and denoted as peak 0. The niche count allows us to observe, for the different peaks in a function,

the competition for slots in the population, the formation of subpopulations, and the diversity in the population. The diversity in the population can be seen by the different niches being maintained as well as the number of individuals in peak 0. Any one individual belongs to only one of the niches. Therefore, the total niche count for all niches is equal to the population size and the total number of niches is equal to the number of peaks + 1.

We also measured the average fitness of the individuals and the maximum fitness in each niche. These values allow us to examine the convergence rate in different niches and the effect of niche average fitness on niche count. We can then examine if the algorithm has converged to the optimum in every peak and was able to preserve the value for many generations. We can also look for a relationship between convergence rate and niche count with the parameters, such as height, width, and location defining a peak. The average fitness of a niche could also help us examine the competition for population slots during each generation between the niches. As the average fitness changes we will observe the impact it has on the niche count.

The last set of metrics we want to observe are the *average similarity rank* and the *average fitness rank* after each generation. The similarity rank, a relative distance of the individuals in the population to another individual, is measured during crowding selection and WAMS replacement. During crowding selection the individuals in the population are sorted in ascending

order with respect to their distance to the parent. The similarity rank of an individual is then its index, starting with 0, in the sorted population. Similarly, during WAMS replacement the individuals in the population are sorted in ascending order with respect to their distance to the offspring. The similarity rank of an individual is its index in the sorted population. In both cases the most similar individual gets a similarity rank value of 0.

The fitness rank on the other hand is obtained by sorting the population in descending order of fitness. The rank number is then given by the index, starting with 0, of the individual in the population. The highest fitness individual is assigned a fitness rank of 0.

These rank values will allow us to observe the effect of crowding selection and WAMS replacement on niching pressure. We can evaluate the effect of different values of C_s , C_r and s , on rank values and convergence to the local optima. In the next chapter we will tie some of the results observed on average rank values with the analysis of the MNC GA.

3.5 MNC GA Parameters

The simulations were done in a 486/33 MHz PC with an application developed using the "ANSI C" language. The genome for each individual is comprised of two chromosomes x and y (only x for F_1 and F_2) representing the coordinates of the test functions. Each coordinate was encoded using a 32 bit chromosome. The genome for each individual will then consist of a 64 bit

string. The individuals in the initial population were generated at random. The crossover probability (χ) was set at 1.0 with $\varepsilon = 2000$. The mutation probability (μ) was set at 0.003. Similarity between two individuals was determined by adding the distance (using the chromosome decimal value) between all chromosomes in the genome. For example, for two individuals I_1 and I_2 with genomes (x_1, y_1) and (x_2, y_2) respectively, we have the similarity between them given by the sum $|x_1 - x_2| + |y_1 - y_2|$. The MNC GA was executed for 100 generations in each run. Other parameters are summarized in Table 3.2.

Table 3.2: Function specific parameters used in the MNC GA

	$F_1 \& F_2$	F_3	F_4	F_5
Population size (n):	100	500	100	200
Number of chromosomes:	1	2	2	2
Crowding selection size (C_s):	15	75	5	15
Crowding factor (C_f):	2	2	3	3
Crowding size (s):	15	75	5	15

All the results were averaged over 20 runs. The following section shows the results obtained using these parameters. Later in this chapter we also evaluate the performance of the MNC GA for different values for some of these parameters. Specifically we evaluate the performance of the algorithm for different values of population size, mutation probability, crossover probability, crowding selection size, crowding subpopulation size, crowding

factor, and ϵ . Only one parameter is modified at a time and the other parameters are left with the values shown above. In the last section we will look at the performance of the MNC GA for different versions of function F_5 .

3.6 Results

The MNC GA was successful in locating all optima on functions F_1 , F_2 , and F_4 . The algorithm converged to all optima on these functions and maintained stable subpopulations for every niche in the functions throughout the run. After 100 generations all five global optima in function F_1 were found including the local optimum at $x = 1.0$. All five optima were also found for function F_2 . Functions F_1 and F_2 did not pose any major challenge to the MNC GA. For both functions the initial population always contained individuals in the peaks and after a very few generations converge to all optima.

Figure 3.5: Niche count and average fitness for niches in function F_2 .

Figure 3.5 shows the niche count and average fitness of each niche for function F_2 , shown in Figure 2.4. From the graph, accounting for the niche

average fitness, we can easily guess which line correspond to which peak. For example, the five graphs from the top in the picture on the right, correspond to the five peaks of the function (from left to right of F_2 in Figure 2.4). The last graph captures the average fitness of those individuals in peak 0. Notice the effect of the average fitness on the niche count, as the average fitness converges toward the peak's maximum values, the niche count of each peak levels off.

Note also that the niche average fitness is less than the maximum peak value but greater than the next lower peak. Additionally we can observe that a higher average fitness meant, in most cases, a higher niche count for the peak. In general a peak with higher average fitness will usually have a greater niche count, but there are other factors that affect the niche count. For example, note that the individuals in peak 0, denoted by the line with the highest niche count at generation 0, has a lower average fitness but higher niche count than the individuals in the smallest peak.

Figure 3.6: Niche count and niche average fitness for peaks in function F_2 .

Similar observations can be made for function F_1 . Observe the niche count and niche average fitness in Figure 3.6 for the peaks in the function. Although all the peaks have the same height and width and the average fitness was very similar with the highest average fluctuating between all niches, the niche count seems to level off and does not seem to fluctuate as much. The difference might not be statistically significant in this case and could be attributed to noise in the random number generator but becomes more apparent when we view the results for function F_3 later on.

The results for function F_4 are more significant. Although the initial population did not have any individuals in the peaks for the first few generations the MNC GA was able to locate both peaks successfully on every run. The distance between the peaks and their small width did not pose any problem to the method. Figure 3.7 shows the niche count and average fitness for the individuals in each peak.

Figure 3.7: Niche count and average fitness for the peaks in function F_4 .

In all runs the initial population and the population after the first few generations did not have any individuals in any peak. Nevertheless the

algorithm was able to locate consistently both peaks and maintain stable subpopulations in both of them. After 30 generations we can observe minor fluctuations in both niche count and average fitness. We also observed that the number of individuals outside the peaks is stable and significant, showing the ability of the MNC GA to maintain a diverse population throughout the run. This property is very important for problems where the fitness function changes dynamically. The algorithm will be able to locate new peaks and adjust to changes in the search space dynamically without the need to restart the population (Eshelman, 1991) or reseed part of it (Maresky et. al.,1995). Results on the behavior of the MNC GA when applied to dynamic landscapes are reported later in this chapter.

For the functions F_3 and F_5 only the very small peaks were not able to maintain any significant number of individuals even though some were present during the initial generations. In some runs, such individuals appeared in the last few generations. For the sample function F_5 , shown on the right side of Figure 3.3, the algorithm was able to locate and maintain subpopulations in all peaks.

Figure 3.8 shows the niche count and average fitness for the peaks listed in Table 3.1 for function F_5 . Observe the ability of the method to avoid premature convergence. Note that the niche count for peak 2, the wider and lower of all peaks, always gains many more individuals during the initial generations when its average fitness was higher. As the average fitness of

the other niches increased so did their niche count and the niche count for peak 2 decreased. In a conventional GA based on fitness proportionate reproduction the population is more likely to converge to peak 2. In fact when we ran the test using the SGA, with crossover probability set at 0.6 and all parameters left with the same value, it consistently converged to peak 2. These results are very encouraging since it shows the ability of the MNC GA to overcome premature convergence and maintain many local optimum for many generations.

Figure 3.8: Niche count and average fitness for peaks in function F_g .

From Figure 3.8 we can also observe that the average fitness is not the only factor affecting the niche count of a peak. Although the average fitness of each niche reaches a value near the maximum of its respective peak, the niche count of peaks with higher average fitness is lower than other peaks. Looking closely we can observe that peak 3, the peak with the largest height and therefore the niche with larger average fitness, does not have the highest niche count. There are a combination of factors that might explain this

behavior. The width of the peak could be small thus having a lower percentage of the search space when compared with other peaks. The location of a peak may be such that offspring from the mating of individuals in two different peaks fall to that peak. Or it could be that the location of the peak in the search space does not allow the niche to pull in individuals from other peaks. At this moment we can only speculate about which factors directly or indirectly affect the niche count in a peak. More results and a rigorous analysis is needed to be able to describe what affects the subpopulations in each niche.

Figure 3.9: Maximum fitness on each niche for function F_5 .

Another good property of the algorithm is its ability to maintain the best fit individual in each peak for many generations. This fact was evident on functions F_2 and F_5 , where the highest value in each peak was maintained for many generations after they were located. Higher peaks did not eliminate optimal values from lower peaks. This behavior allows members from different niches to co-exist in the population for many generations and allows

the researcher to identify interesting areas of the search space. This property is illustrated in Figure 3.9 for the peaks of function F_5 . Observe that all five peaks were located at different generations and maintained until the last generation. Recall that all these results were averaged over 20 runs, which means that after fifty generations the population was converged to all five peaks in every run.

Figure 3.10 shows the niche count and niche average fitness for the peaks in function F_5 . Although it is hard to follow the lines for some of the peaks we can observe that all 25 peaks had, on average, some individuals in them after 100 generations. All niches contained a similar number of individuals in the first few generations. After about 10 generations the difference between the niche counts is more notable with values fluctuating between 4 and 20. More important is niche average fitness graph which shows how successful the MNC GA was in locating and maintaining the average value in each of the niche close to the actual maximum value of the peak. We can easily account for the average of the 8 "tallest" peaks.

In function F_3 we can also observe that the number of individuals in a peak is related to more than just its height. On function F_3 where the optima are located on a 5x5 grid, peaks along the same x and y axis as the global optima had more individuals than other peaks with higher values. Some of the extra individuals can be attributed to mutation since a bit change in one of the chromosomes will cause an individual to move along the x or y axis. We

ran the same test with mutation set at 0.0 and no major changes were observed.

Figure 3.10: Niche count and niche average fitness for peaks in function F_3 .

Not shown are the count and average fitness for the individuals located in peak 0. About 200 individuals were located in peak 0 between generations 50

and 100. Their average fitness was about 0.003 during the same period. As in previous functions it shows that with the MNC GA exploration is not restricted to the areas of search space where local optima has been located. This property is essential when considering dynamic environments where the optima in the search space are changing constantly. We can allow the current population to evolve dynamically with the fitness landscape without having to re-seed the population.

3.7 Empirical Analysis of MNC GA Parameters

In this section we investigate the effect of various parameters on the performance of the MNC GA. Specifically we investigate the effect of different values for population size, crowding selection size, crossover probability, mutation probability, interval crossover size, crowding factor, and crowding size. This is done by evaluating each parameter independently. Only one parameter is changed at a time. All other parameters are set to the values described in Section 3.5. All results are averaged over 20 runs. Five different values are used for each parameter.

We want to get a good feel for the effect of different parameters on various metrics such as niche count, niche average fitness, niche maximum fitness, average similarity rank during selection, and average fitness rank during replacement. The sample function F_s , shown on the right of Figure 3.3, was used for all the tests in the following sections. The line plots and the legend shown in Figure 3.8 (niche count and niche average fitness) and Figure 3.9

(niche maximum fitness) are used as part of the results throughout the next sections.

3.7.1 Effect of Population Size

In this section we examine the effect of different population sizes (n) on the performance of the MNC GA. We ran some tests using population sizes of 50, 100, 200, 300, and 400. Nothing out of the ordinary was noted in the results. Higher population sizes allowed the MNC GA to converge to the best solution in each peak and to maintain stable subpopulations in the niches in less number of generations. The increase in this performance metric is accompanied negatively by an increase in computational burden. Doubling the population size from 200 to 400 did not make the algorithm find the optima in each peak in half the number of generations. Population sizes of 50 and 100 required more than 100 generations to locate the optimum in all peaks. For similar problems, with the same MNC GA parameters, a population size of 200 is the best choice.

For all population sizes the MNC GA was able to form stable populations in all the niches. The underlying behavior was the same. The niche count for each peak was determined, in part, by the niche average fitness. As the average fitness in each niche stabilizes in all peaks so did the niche count for all peaks. On the other hand a higher population size increased the percentage of individuals outside of the niches. This is due, in part, to an increase in mating pairs where the individuals belong to different niches.

Given that the crowding selection size (C_s) was kept at a value of 15 for all different population sizes, the probability of selecting a parent and its mate from the same niche decreases as the population size increases. This behavior is linked to the the average similarity rank. In these tests we found the average similarity rank to be linearly dependent to the population size. The average similarity rank for a population size of 50 was approximately 3.5. For a population size of 100, 200, 300, and 400 it was 6.6, 12.6, 19.1, and 25.0 respectively. Similarly, the average fitness rank was found to be linearly dependent to the population size. Later on in the next chapter we tie these results to the analysis of the MNC GA operators.

3.7.2 Effect of Crowding Selection Size

In this section we analyze the performance of the MNC GA as the crowding selection size (C_s) is increased. Crowding selection size values of 5, 10, 15, 20, and 25 were used. The following properties were observed. First, as we can see from Figure 3.11, the niche count for individuals outside the peaks decreases as the crowding selection size increases. The reason is obvious, increasing the crowding selection size decreases the probability that a parent and its mate are from different niches. This in turn decreases the chance that the offspring generated using interval crossover is outside of the any of the niches. Describing it from another angle, the probability that the parent and mate belong to the same niche is proportional to the crowding

selection size. Offspring generated are likely to belong to the same niche as their parents.

Figure 3.11: Effect of different values for the crowding selection size (C) on the niche count of each peak. Niche count for values of 5 (top left), 10 (top right), 20 (bottom left), and 25 (bottom right) are shown above.

Second, as we increase the crowding selection size the average fitness of each niche is more stable. That is, the value of the average fitness from generation to generation varied less as we increased the value of crowding selection size. We can also observe from Figure 3.11 (see legend on Figure 3.8 or Figure 3.9) that the niche count depends more on the average fitness of the niche as the crowding selection size value is increased and can therefore reverse the effects of premature convergence more easily. As a result the peaks with higher average fitness were able to increase their niche count

above some of the peaks with lower average fitness that had a higher niche count in the initial generations.

Finally, the convergence speed of the MNC GA improved for higher peaks as the crowding selection size was increased. The global optima was always located for all peaks using different values, but the higher peaks converged faster to the maximum value with higher crowding selection size values. This fact is due to the increase in mating pairs from the same niche and increase in niche count, as evident by the decrease in average similarity rank as the crowding size was increased. No effect was observed on the average fitness rank.

3.7.3 Effect of Crossover Probability

In this section we describe the performance of the MNC GA as we change the value of the crossover probability (χ). The results were obtained for crossover probability values of 0.7, 0.8, 0.9, 0.95, and 1.0. The results show what we expected. No significant changes occurred to the niche count. The difference between the niche counts was small and the same behavior was observed for all peaks. The average similarity rank and the average fitness rank was the same for the different crossover probabilities.

3.7.4 Effect of Mutation Probability

In this section we describe the performance of the MNC GA as we change the value of the mutation probability (μ). For the results described here we

used values of 0.001, 0.003, 0.01, 0.05, and 0.1 for the mutation probability.

Figure 3.12 shows the effect of the mutation probability on the niche count.

Two things can be observed from the results. First, an increase on the mutation probability increases the number of individuals outside the peaks.

This increase is natural since mutation controls, in part, the diversity of the population.

Figure 3.12: Effect of mutation probability on the niche count for function F_5 . The results shown here are for mutation probability values of 0.001 (top left), 0.01 (top right), 0.05 (bottom left), and 0.1 (bottom right).

Second, an increase in mutation probability decreased the competition among the different niches, due to the increase in the number of individuals outside the niches having lower average fitness making them perfect candidates for replacement by offspring from any of the niches. In other

words, the chances that an offspring from a niche replaces an individual from a different niche decreases as the mutation probability increases.

Figure 3.13: Effect of mutation probability of the niche maximum fitness for function F_5 . The results shown here are for mutation probability values of 0.001 (top left), 0.01 (top right), 0.05 (bottom left), and 0.1 (bottom right).

On the other hand, the effect of a higher mutation probability on the niche maximum fitness is not what we expected. Figure 3.13 shows the results for different mutation probabilities. For all mutation probability values the MNC GA was able to locate and maintain optima in the peaks. Only when the niche count fell below 10 individuals did the peak had problem maintaining the best individuals in it. The increased number of individuals outside the peaks did not turn the algorithm into a random search; convergence to the top of the peaks was slowed down but not impeded. Competition among members of the same

niche was healthy and allowed the niche members to move towards the top of the peak.

The different mutation probabilities had no effect on the average similarity rank or the average fitness rank of the individuals selected during each generation.

3.7.5 Effect of Interval size

In this section we investigate the effect of different values of the minimum interval size (ϵ), used during interval crossover, on the performance of the MNC GA. The results described here used interval size values of 10, 100, 1000, 10000, and 100000. No major changes were observed on the niche count and niche average fitness. The different values did not affect the way in which the competition between niches occur.

A significant change was observed in the convergence speed of the MNC GA. Figure 3.14 shows the line plots for the different values of the interval size. The line plot in the lower left, corresponding to an interval size 10,000, indicates that all global optima were found consistently by the 35th generation. Using a higher interval size during crossover affected the ability of the MNC GA to locate the global optima, due to the increase in random search caused by bigger interval sizes during interval crossover. More generations are needed in order to find all global optima for this function using bigger interval sizes. Smaller sizes will always locate all global optima in the given number of generations, but very small sizes took more

generations. When using smaller interval sizes the convergence to the top of the peak is slower because the interval size restricts the offspring to a smaller region bounded by the parents.

Figure 3.14: Niche maximum fitness averaged over 20 runs for different values of the interval crossover size ϵ . Starting at the top left we have results for a size of 10, in the top right results for a size of 100, in Figure 3.9 for a size of 1,000, in the lower left results for a size of 10,000, and in the lower right results for a size of 100,000.

The different interval sizes had no effect on the average similarity rank or the average fitness rank of the individuals selected during each generation.

3.7.6 Effect of Crowding Factor

In this section we describe the effect of different crowding factor (C_p) values on the performance of the MNC GA. Crowding factor values of 2, 3, 4, 5, and 6 were used for the results shown here. In general we observed that higher crowding factor values had a positive impact on the niche count. As

the value increased, peaks had a higher niche count. This behavior can be observed in Figure 3.15 for crowding factor values of 2 (on the left) and 6 (on the right). With a crowding factor value of 2 the niche count for individuals not belonging to any of the peaks was very high (between 70 and 80). When the crowding factor was increased to 6 the niche count for individuals outside the peaks decreased dramatically (to a value between 10 and 20). At the same time the niche count for individuals in the higher peaks increased. Increasing the crowding factor increases the competition for slots in the population, as well as competition among members of different niches. Those niches with higher niche average fitness are more likely to survive replacement when members of the other niches appeared in the crowding factor group. This was evident by the increase in average fitness rank as the crowding factor was increased. Indicating that it was more likely to replace lower rank members of the population. As expected no change was observed for the average similarity rank.

Figure 3.15: Niche count for crowding factor values of 2 (left) and 6 (right).

On the other hand increasing the crowding factor has a negative impact on the niche maximum fitness. Examining Figure 3.16 we will note that all global optima were consistently found in all runs for crowding factor values of 2, 3 (shown in Figure 3.9), and 4. Using crowding factor values of 5 and 6 the MNC GA was not able to find the global optima for the highest peak in all runs in the given number of generations. This is due, in part, to the small width of the peak coupled with the increase number of individuals in it, causing convergence to the top of the peak to stagnate.

Figure 3.16: Niche maximum fitness for all peaks of function F_6 using crowding factor values of 2 (upper left), 4 (upper right), 5 (lower left), and 6 (lower right).

This effect was also observed in the niche average fitness. Higher peaks could not increase their niche average fitness dramatically after its members were close to the top. Increasing the crowding size (s) will alleviate this

problem, since higher values of s increases competition among members of the same niche eliminating lower individuals from the peak faster. This behavior needs to be studied further to be able to pinpoint the cause for the stagnation on higher peaks.

Figure 3.17: Niche count for different values of crowding size for the niches of function F_s . Values of 5 (top left), 10 (top right), 20 (bottom left), and 25 (bottom right) are shown.

3.7.7. Effect of Crowding Size

In this section we examine the effect of different crowding size values on the performance of the MNC GA. Specifically we observe the performance of the algorithm for crowding size values of 5, 10, 15, 20, and 25. The first thing we observed was that increasing the crowding size increases the competition

between members of the same niche and at the same time decreases competition between members of different niches.

From Figure 3.17 we can observe that as the crowding size was increased, the niche count for a peak was not affected by the other peaks. Higher crowding size values allowed peaks with lower average fitness to maintain larger number of individuals. For example, the niche count for individuals outside the peaks increased with higher values of the crowding size. By increasing the crowding size it increases the chances that offspring replace individuals from the same peak, maintaining a more balance niche count from generation to generation.

On the other hand convergence to the global optima of each peak has slowed down. For crowding size values of 20 and 25 the MNC GA was not able to converge to the global optima in the given number of generations. An increase competition within the individuals in the same niche stagnates the improvements to the niche average fitness because there is an increase in the number of less fit offspring replacing more fit niche members. Another reason for the slow convergence to the higher peaks is the lower niche count caused by higher crowding size values. With less number of individuals in the niche, it is less likely that a parent from the niche can find a mate from the same niche. Although the average fitness rank was somewhat lower as the crowding was increased, individuals were often replaced by lower fit offspring.

3.8 Other Sample Functions F_5

In this section we examine the performance of the MNC GA on four variations of function $F_5(x,y)$. Like the original function F_5 , the variants have five peaks and their location, width, and height were generated at random. To distinguish the four variants from the original, we will refer to them as $F_{5,1}$, $F_{5,2}$, $F_{5,3}$, and $F_{5,4}$. Table 3.3 specifies the peaks for these functions. As in the previous sections all results were averaged over 20 runs. The parameters used for the MNC GA are the same as those used for function F_5 and described in Section 3.5.

Table 3.3: Location, width, and height for the peaks in functions $F_{5,1}$, $F_{5,2}$, $F_{5,3}$ and $F_{5,4}$.

Function	Peak Location	Width	Height
$F_{5,1}(x,y)$ $0 \leq x, y \leq 65.535$	(63.81, 4.36)	4.55122	29.7219
	(38.56, 37.82)	4.52074	32.2517
	(47.12, 13.78)	4.8969	33.3982
	(2.68, 57.65)	3.1818	11.5794
	(33.21, 23.79)	0.241381	74.9232
$F_{5,2}(x,y)$ $0 \leq x, y \leq 65.535$	(11.66, 58.66)	2.87159	16.3294
	(53.40, 4.66)	3.93287	14.6774
	(56.78, 64.62)	0.42059	77.4209
	(32.57, 1.85)	3.24469	71.6625
	(8.54, 58.98)	1.74803	94.3676
$F_{5,3}(x,y)$ $0 \leq x, y \leq 65.535$	(41.73, 38)	0.846273	99.2606
	(42.99, 4.96)	3.31453	99.633
	(9.47, 25.89)	1.32045	22.5901
	(62.46, 11.59)	3.30759	31.7457
	(49.4, 28.63)	3.25467	13.8121
$F_{5,4}(x,y)$ $0 \leq x, y \leq 65.535$	(6.26, 17.34)	3.82096	82.1918
	(32.58, 5.25)	2.69618	84.5886
	(27.70, 52.69)	2.22031	67.7593
	(26.82, 21.32)	3.37049	91.8288
	(4.72, 63.82)	4.76132	33.2565

Figure 3.18: Sample functions $F_{5,1}$ (top left), $F_{5,2}$ (top right), $F_{5,3}$ (bottom left), and $F_{5,4}$ (bottom right).

Figure 3.18 shows the graph for all four functions. With the exception of function $F_{5,2}$ all five peaks can be seen easily from these graphs. The upper corner of the graph denotes the location $(0, 0)$ with coordinate x going left and down and coordinate y going right and down. From Table 3.3 we can see that the first and fifth peak of function $F_{5,2}$ are very close and therefore merged to form the largest peak of the function located on the left corner of the graph.

From simple inspection we can observe that the functions have a good mix of peaks. There is a good combination of peaks with different height and width whose locations are scattered throughout the landscape.

Figure 3.19: Niche count for functions $F_{s,1}$ (top left), $F_{s,2}$ (top right), $F_{s,3}$ (bottom left), and $F_{s,4}$ (bottom right).

The results obtained with the MNC GA are very positive. For all four functions the algorithm located all peaks and maintained subpopulations for all but the smaller peaks. The inability to maintain a stable niche count was noticeable for smaller and skinnier peaks located close to wider and higher peaks. Peaks with smaller width are less likely to get offspring resulting from mutation or when a member of their niche mate with a member from another niche. By being close to higher peaks competition increases between the niches. Higher values of crowding selection size and crowding size are

needed in order to be maintain stable subpopulations in skinnier peaks. That will increase the competition between members of the same niche by increasing the chances that mating and replacement occurs among members of the same peak.

Figure 3.19 shows the niche count for all the functions. The legend for the plots is the same as the one in Figure 3.9. Notice that in function $F_{5.1}$ and $F_{5.3}$ one of the niches cannot maintain a stable niche count and in the case of function $F_{5.3}$ it goes to zero. In function $F_{5.1}$ this niche corresponds to the third peak with width equal to 4.8969 units (higher number represents a skinnier peak) and height of 33.3982 units. Note that this peak is located closer to highest peak in the function. The same occurs for the smaller peak in function $F_{5.3}$.

We can also note from Figure 3.19 that the niche count for individuals outside the peaks is higher than the niche count for any of the peaks. This is also an indication that mating among members of different niches was high, thus generating a number of individuals outside the peaks.

Figure 3.20 shows the niche maximum fitness for all four functions. From these graphs we can observe that peaks that could not maintain a stable niche count throughout the run could not maintain the best individual in the niche. Looking back at Figure 3.19 it can be seen that these peaks are the same ones corresponding to function $F_{5.1}$ and $F_{5.3}$. To verify our convergence

and stability for function $F_{5.1}$ we ran some tests with crowding selection size and crowding size set to 20, and crowding factor set to 4. As expected all peaks were located successfully and stable subpopulations were maintained in each peak.

Figure 3.20: Shown above are the niche maximum fitness for functions $F_{5.1}$ (top left), $F_{5.2}$ (top right), $F_{5.3}$ (bottom left), and $F_{5.4}$ (bottom right).

For function $F_{5.2}$ we can also note that the maximum niche value is fluctuating for two of the niches. One of these niches (fourth line from the top) corresponds to the set of individuals outside the peaks, peak 0. The other niche (fifth line from the top) corresponds to peak 1 located very close to peak 5. The high fitness in peak 0 and the fluctuations in peak 1 are due to the way in which individuals are determined as being part of a peak. Some individuals from peak 5 (a higher peak) are being considered as members of

peak 0 because the two adjacent circles (formed to calculate each peak region) with center at the top of peak 1 and peak 5 do not cover both peaks entirely.

3.9 Comparison with other Methods

In this section we compare the performance of the MNC GA to some of the methods described in Section 2.4. Specifically the Mahfoud's results (1995a, 1995b) about the performance of deterministic crowding, sharing, fitness derating (called sequential niching in their work), and parallel hillclimbing on functions F_1 , F_2 , and a scaled up version of F_3 . Using the same methodology we ran the MNC GA on the same functions and compared our results to theirs.

Parallel hillclimbing is not a GA but a heuristic that starts with a randomly created population. Then every individual climbs up greedily to the nearest peak. This technique was used as a base for comparing GAs to more traditional methods. Basically, the algorithm takes an individual in the population and moves it by the quantity h , the step size, along any of its coordinates if that move increases the individual's fitness. This is done for every individual in the population. Then the step size h is reduced in half and the process is repeated. The algorithm stops when step size is less than ϵ . Complete details about parallel hillclimbing can be found in the above cited papers.

In the cited reference they used a single performance criterion, namely the number of function evaluations, to compare the performance of different techniques. For each technique they calculated the minimum population size (which is a power of 2) required to locate and maintain the peaks in the function being tested. This is done for each function independently. For each test the GA is allowed to run until the average fitness of the population at generation t is no greater than the average fitness of the population at generation $t - 4$ plus an increment inc . The value of inc was set at 0.001 for F_1 and F_2 , and 0.1 for F_3 . After the GA converges, the parallel hillclimbing algorithm is applied. All function evaluations are counted for the GA and the combination of the GA and parallel hillclimbing.

Table 3.4: MNC GA parameters used for each function.

	F_1 & F_2	F_3
Crowding selection size (C_s):	4	10
Crowding factor (C_f):	4	7
Crowding size (s):	10	75

We implemented the parallel hillclimbing algorithm exactly as in the cited reference. We used the same step size and ϵ values. The crossover probability was set at 1.0 and the mutation probability to 0.0. Single point crossover was used in the mating step for all GAs. The only difference between the methods is in the population size increment for function F_3 ; we

started with a population size equal to the number of peaks (25) and kept doubling it until we found a population size that was sufficient to locate and maintain all peaks. As mentioned in Section 3.5, the MNC GA population size and crowding parameters can be appropriately set according to the number of peaks we want to locate. Therefore, it makes more sense, in this case, to use a population size which is a function of the number of peaks in the function. Table 3.4 summarizes the parameters used for the MNC GA.

Table 3.5: Comparison of parallel hillclimbing (PH), fitness derating (FD), sharing (SH), deterministic crowding (DC), and multi-niche crowding (MNC) on functions F_1 , F_2 , and F_3 .

Method	Average Niche Count	Average Number of Generations	Average Function Evaluations for GA	Average Function Evaluations for GA + PH
Function F_1				
PH	2.72			1,017
FD	3.68	46.40	738	4,112
SH	5.76	8.00	264	2,431
DC	2.40	28.00	380	1,246
MNC	3.14	12.30	197	1,026
Function F_2				
PH	2.72			1,021
FD	4.64	75.60	1,770	8,632
SH	8.96	8.70	442	3,827
DC	2.40	27.40	372	1,264
MNC	3.16	12.30	197	1,027
Function F_3				
PH	12.29			29,017
FD	3.58	146.30	12,202	46,657
SH	5.12	11.80	1,638	12,910
DC			> 1,500,000	
MNC	3.07	21.50	2,150	12,133

The comparative results, shown in Table 3.5, are averaged over 10 runs. The methods can be compared using any of the metrics such as average niche count (column 2), average number of generations (column 3), average number of function evaluations before the GA converges (column 4), and the average number of function evaluations for the GA and subsequent parallel hillclimbing (column 5). Table 3.5 is repeated once for each of the test functions, namely F_1 , F_2 , and F_3 .

Inspection of the table reveals the following. If the number of function evaluations (averaged over 10 runs) is the criterion for comparison, the MNC GA, working alone outperforms its peers in column 4 and does as well as parallel hillclimbing in column 5 for both F_1 and F_2 (see the bold face figures). However for test function F_3 , the sharing method converged in less number of generations and outperformed the MNC GA working alone, but the MNC GA edged out all methods when the combined number of function evaluations are accounted for in column 5. The MNC GA, on average, took more generations than sharing before the population converged. This is due to the worst among most similar replacement policy which allows the MNC GA to successfully filter out low fit individuals within each niche and get closer to the top of each peak. This meant that many of the individuals in the niche took less number of function evaluations when parallel hillclimbing was applied.

From Table 3.5 we can also observe that the MNC GA plus parallel hillclimbing gave a performance close to parallel hillclimbing alone for

functions F_1 and F_2 . It outperforms parallel hillclimbing for function F_3 . More tests are needed using other functions, not exhibiting the symmetry present in F_1 , F_2 , and F_3 , before we can determine with confidence the overall performance of the MNC GA when compared to other methods. Other performance metrics, in our opinion, are necessary and useful when comparing niching methods also. Other useful metrics include those that measure the ability of a method to locate and maintain niches in static and dynamic environments for many generations, number of function evaluations to locate all global optima, average population size needed to locate and maintain niches, and so on.

3.10 Dynamic Landscapes

In this section we examine the behavior of the MNC GA on a multimodal dynamic landscape. A multimodal dynamic landscape is a search space where the locations and heights of peaks change with time. A superior solution in a peak at time t_0 could become an inferior solution at time t_1 and vice versa. An example of such a landscape is the stock market where parameters influencing the price of stocks are constantly changing. In order to obtain the best return for the money invested, the stock brokers must constantly monitor these parameters and move the money to the solution that offers the best payoff at that time. Investment solutions that were not optimal at one time could potentially become the best investment solutions at

some other time. In such a landscape, keeping a set of possible investment solutions is smart and necessary.

A multimodal dynamic landscape presents a very challenging problem to any search technique for various reasons. First, the heights and widths are constantly changing. Second, new peaks that are constantly emerging must be located quickly to avoid missing solutions that could potentially be useful. Third, solutions already converged to a peak that has since flattened (become a peak with a small height) must be diverted to other areas of the search space in search of better solutions. Finally, solutions that have been found previously must be re-evaluated after some time to maintain an accurate value of its fitness.

A technique for multimodal dynamic landscapes must be able to overcome these obstacles by locating multiple optima, existing ones and new ones, and maintaining them until they disappear. Additionally, it must minimize the number of function re-evaluations for solutions found thus far without affecting the convergence properties of the technique. Later, in this section we will see how the MNC GA accomplishes these tasks.

3.10.1 Background

There have been many attempts to apply GAs to dynamic landscapes. None of them considered applying a niching technique to such problems. In one of the approaches part of the population is re-initialized (Eshelman, 1991; Grefenstette, 1992) after it has converged. In this approach it is very

hard for the newly introduced solutions to establish themselves when the population contains highly fit individuals. Maresky et. al. (1995) introduced an operator called *selectively destructive re-start* that improves the previous approach by reinitializing the chromosome in a solution with certain probability. The probability is given by a combination of population size, improvement of the best-in-generation individual, and the number of function evaluations. The main difficulty with this approach is to find the appropriate re-initialization probability for the problem at hand.

In another work, Cobb and Grefenstette (1993) compared a partial re-start of the population with two approaches that manipulate the mutation rate of the GA. In the first approach the mutation probability is set at a higher rate in a standard GA. In the second approach, called *triggered hypermutation*, the mutation rate is dynamically changed to high values when the time-averaged performance of the GA deteriorates. It was shown that mutation based approaches worked better than population re-initialization for environments without abrupt changes. Some of the drawbacks are the reduction in the performance of the GA regarding the improvement of the population average fitness. Additionally, the mutation rate selected affects the performance of the GA greatly when applied to abruptly changing environments.

Some other approaches used schemes to encode, in the chromosomes, previous history about the individual (Goldberg and Smith, 1987; Ng and

Wong, 1995). In these studies the chromosomes are encoded using genome structures, like diploid or triallelic schemes, that are able to preserve genetic information that will be beneficial if the environment changes. In a separate study (Dasgupta and McGregor, 1992) used a tree structure representation of the population. In this approach, called the sGA (structured GA) nodes at a higher level in the tree regulated the activation and deactivation of genes at lower levels in the tree. These approaches were shown to work well in a landscape when only two peaks are oscillating. It is unclear if these approaches will work on problems with many more peaks.

In all of these approaches the main focus was on increasing the population diversity. Increasing the diversity allowed the GA to discover new peaks while at the same time preserving the good solutions found thus far. Our MNC GA exhibits both of these properties implicitly. Solutions from multiple peaks are maintained while at the same time allowing a subset of the individuals in the population to explore other regions of the search space. In the sections below we will show how nicely the MNC GA solves problems in a dynamic landscape.

3.10.2 Experimental Design

In this section we describe the multimodal function with dynamically changing peaks that was used to test the performance of the MNC GA. We also describe the parameter settings for the MNC GA. The test function used here is based on function $F_s(x,y)$ described in Section 3.2. We want to test

three cases with this function. The first case is based on a landscape where the number of peaks (or optima) at the beginning of the run is the same as when the run ends, but the location and properties of the peak (width and height) are different. For this case we generate 10 peaks (with different width and height) at random. We start the run with the first set of 5 peaks completely manifested in the landscape. Then we dynamically decrease the height of these peaks while at the same time increase the height of the second set of 5 peaks. Toward the completion of the run all the peaks in the first set have disappeared and the peaks in the second set are fully manifested.

The second case is similar to the first case, but here we start with 2 peaks in the first set and dynamically change to the second set containing 8 peaks. This case is more challenging since the MNC GA must be able to dynamically locate and spread the individuals in the population to other peaks in the landscape.

The third and final case is a mirror image of the second case. We start with an initial set of 8 peaks and dynamically change them to a second set containing 2 peaks. This case was done for completeness only, since success in the second case, described above, will provide some results indicating the expected behavior in this simpler case.

In all three cases we used the 10 peaks (generated randomly) shown in Table 3.6. Case 1 contains peaks 1 to 5 in the first set and peaks 6 to 10 in the second set. Case 2 contains peaks 1 and 2 in the first set and peaks 3 to 10 in the second set. Finally, case 3 contains peaks 1 to 8 in the first set and peaks 9 and 10 in the second set.

Table 3.6: Parameters of the 10 peaks used for all tests of the MNC GA in a dynamic landscape.

Peak	Peak Location	Width	Height
1	(8.37208, 64.8927)	3.81693	63.0384
2	(50.1509, 13.6573)	3.54459	96.1568
3	(51.1377, 28.8592)	1.6538	68.0674
4	(51.712, 9.99087)	4.32051	89.004
5	(9.3263, 19.3406)	1.92525	27.3985
6	(30.6667, 55.8088)	4.41471	73.3882
7	(47.7256, 38.3194)	0.916318	90.1156
8	(28.3306, 12.1417)	4.93385	85.6341
9	(25.8438, 50.7668)	0.20225	13.2312
10	(11.7122, 30.492)	1.02969	82.9828

The landscape is formed from the contribution of each set of peaks to the total value of the function. The contribution of each set of peaks is increased or decreased by 10% every g generations. In our tests we used values of g equal to 10 and 20 generations. To be more exact, let $h_1(x,y)$ be the function, similar to F_s , with peaks defined from set 1. Let $h_2(x,y)$ be the function, similar to F_s , with peaks defined from set 2. The fitness value $f(x,y)$ of an individual in generation k is given by the following equation:

$$f(x, y) = (1 - 0.1 \left\lfloor \frac{k}{g} \right\rfloor) h_1(x, y) + 0.1 \left\lfloor \frac{k}{g} \right\rfloor h_2(x, y).$$

Due to the steady state nature of the MNC GA, individuals that survive for many generations must be re-evaluated to properly adjust their fitness. For these tests we re-evaluated all individuals that have been in the population for more than 10 generations. The value for C_s , the crowding selection size, was set at 20. The value of s , the crowding group size, was set at 40 and the value of C_p , the crowding factor, was set at 4. These values were selected to allow mating among individuals from different peaks more often and to increase replacement of less fit individuals from the same peak. They may not (with a high probability) be the optimal values for these tests. All other parameters for the MNC GA were set to those values used for F_s in section 3.5. As in the other tests interval crossover is used in conjunction to bit mutation during the mating step.

3.10.3 Results and Discussion

The results shown in this section are very promising. In all test cases the MNC GA was able to locate the new peaks emerging at different locations of the landscape. Moreover, the niche count adjusted dynamically among the peaks as their height and width changed. The maximum fitness of the highest peaks was found and maintained dynamically during the run.

Figure 3.21 shows the niche count and the niche maximum fitness for test case 2. We can observe the algorithm converging to the first two peaks in the

initial generations. As the other peaks emerge they were located and all peaks maintained in parallel. After generation 100 the initial two peaks disappeared and so did the individuals in the niche. From the niche maximum fitness graph, in the same figure, we can observe the steps for each peak. It indicates how the MNC GA was able to locate the maxima in the highest peaks, maintain them during the run, and adjust to the changes dynamically. As the peaks in set 1 (peaks 1 and 2) decreased so did their niche count. Between generations 50 and 60, when both sets of peaks contributed equally to the total fitness, we can see all the lines coming together in the charts. After this point the peaks in the second set took over the population.

Other tests can be done to determine with more certainty the applicability of the MNC GA to dynamic environments. For example, we can use a landscape where all peaks are changing independently of each other and new peaks are created at random. Such an environment is more practical and is an area where more research is needed. Nevertheless, the test cases used here have shown the ability of the MNC GA to adapt well to such environments. It locates and maintains the different peaks dynamically without the need of restarting the population, using a higher mutation rate, or using special encoding for the chromosomes.

Figure 3.21: Niche count (top) and niche max fitness (bottom) for test case 2 using a value of $g = 10$.

3.11 Summary

The properties exhibited by the MNC GA are very encouraging. The approach successfully locates multiple optima and maintains stable subpopulations in each peak. The formation of different subpopulations in the niches evolve naturally thus maintaining a very healthy diversity throughout the search. It is this stability between exploration and exploitation of the search space that makes the MNC GA a very good technique for complex problems in static or dynamic environments. A more

rigorous analysis must be made before we can claim the success of the algorithm when applied to other problems. The ability to search a complex space in an efficient manner and locate multiple optima will be useful in many areas where current optimization techniques do not work well. The same properties make the MNC GA applicable to problems where the search space is changing dynamically. Re-initializing the population is not always applicable when the answer must be provided in a predetermined amount of time.

We applied the MNC GA to various types of problems including multimodal function optimization of static and dynamic search spaces (Cedeño and Vemuri, 1992), a grouping and mapping problem usually encountered during the design of databases called the file design problem (Cedeño, 1992), a problem in DNA mapping called the restriction fragment assembly problem (Cedeño, 1993; Cedeño and Vemuri, 1993; Cedeño et. al., 1995), and a multicriteria optimization problem arising in aquifer management (Cedeño and Vemuri, 1995). In later chapters we will describe each of these applications in more detail. As in the results shown here we will observe the same behavior in all these problems. The MNC GA converges to multiple solutions while maintaining stable subpopulations in different niches during the run.

4. Mathematical Analysis of the Multi-Niche Crowding GA

In this chapter we present a mathematical analysis of the multi-niche crowding GA for multimodal function optimization. To simplify the analysis we focus on the application of the steady state MNC GA on an arbitrary real valued multi-modal function, $F(x)$, of one real variable. Moreover, we use interval crossover and bit mutation as the genetic operators during the mating step. Section 4.9 describes how to apply the same analysis to the generational MNC GA.

4.1 Overview

The analysis of the MNC GA is based on a Markov Chain analysis similar to that of Vose (1992), and Nix and Vose (1992) cited on Sections 2.3.3 and 2.3.4 respectively. A Markov Chain is a Markov process (or a stochastic process) with a denumerable (finite or denumerably infinite) number of states. As in any Markovian process only the last state occupied by the process is relevant in determining its future behavior. The time parameter t used in the process will be taken from the set of non-negative integers. Knowledge of the current state and the probabilities governing the transition between any two states is sufficient information to specify the probabilistic behavior of the process.

The main goals of the analysis presented in the next sections are to:

1. Define the selection probability, under crowding selection, for any two individuals in the population.
2. Define the probability that an offspring is generated after mating under interval crossover and bit mutation.
3. Define the replacement probability, under worst among most similar replacement, for any individual in the population.
4. Use the previous results to formally define the transition probability matrix for the Markov chain representing the MNC GA stochastic process so that it can be used to identify and analyze the search trajectory of the algorithm.

By accomplishing the first three goals in this list we can easily use the results to define the Markov transition probability matrix or Markov matrix. The Markov matrix defines the transition probability between any two states. In our case a state corresponds to a particular population configuration. The Markov matrix captures the asymptotic behavior of the MNC GA.

```

Generate Population of  $n$  individuals at random.
For evals = 1 to MAX_EVALS
  Select mating pair using crowding selection.
  Apply interval crossover (with probability  $\chi$ ) to pair and bit
    mutation (with probability  $\mu$ ) to offspring.
  Insert offspring in population using WAMS replacement.
End

```

*Figure 4.1: Pseudo code of steady state MNC GA used for analysis. **

Figure 4.1 shows the pseudo code that summarizes the salient points of the steady state MNC GA. The analysis presented in the next sections is based on this algorithm.

We have shown empirically that in the MNC GA a diverse set of individuals is maintained in the population for many generations. Moreover, it locates multiple peaks and maintain stable subpopulations in different niches of the search space throughout the run.

The analysis presented in this chapter is organized as follows. Section 0 defines some notation used throughout the chapter. Section 4.3 derives, for any pair of population members, the probability that they are selected under crowding selection. Section 4.4 calculates the probability of obtaining any chromosome value under interval crossover and bit mutation. Section 4.5 finds the probability of replacing any member of the population under worst among most similar replacement (WAMS). Section 4.6 uses the results from the previous sections to define the Markov matrix for the steady state MNC GA. Section 4.7 analyzes crowding selection in more detail. Section 4.8 analyzes WAMS replacement in more detail. Finally, Section 4.9 describes the differences between the steady state MNC GA and its generational counterpart and how this analysis can be applied.

4.2 Notation

In this section we define some notation and terms that will be used throughout this chapter. First, we define Ω to be the set of all length l binary strings and let $N = 2^l$. For $l = 3$, Ω would have $N = 2^3 = 8$ elements as shown in Figure 4.2. For easy identification, each element of Ω is associated with a tag in the

$\Omega = \left\{ \begin{array}{c} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \right\}$	$Tags = \left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \right\}$
---	---

Figure 4.2: Members of Ω for string length $l = 3$.

interval $[0, N-1]$, which in this case is the decimal value of the binary string. The elements of Ω are called *chromosomes*. The GA manipulates elements of Ω to find highly fit chromosomes. The members of the population, composed of elements of Ω , are called *individuals*. In our analysis we assume that individuals are haploid, i.e., they each have a single chromosome.

The population of size n can be denoted by the column vector $I = [I_0 I_1 \dots I_n]^T$, where T stands for vector transpose. A component of this vector, namely I_j , represents an individual in the population with a value in Ω . Using elements of Ω shown on Figure 4.2, the vector $I = [4 \ 1 \ 7 \ 5 \ 4 \ 3]^T$ is a valid population with six individuals whose tag values are 4, 1, 7, 5, 4, and 3. Clearly, there are different vectors I representing the same population, like for example $I = [7 \ 3 \ 4 \ 1 \ 4 \ 5]^T$ which is a permutation of the previous vector.

We now denote a population using the column vector $p = [p_0 \ p_1 \ \dots \ p_{N-1}]^T$. The components of this vector, namely p_j , are the number of copies of chromosome with tag value j in the population. Using the example cited above we have $p = [0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]^T$ which represents one copy of chromosome 1, one copy of chromosome 3, two copies of chromosome 4, one copy of chromosome 5, and one copy of chromosome 7 in the population. Evidently

$$\sum_{j=0}^{N-1} p_j = n,$$

the size of the population. The vector p also represents uniquely the state of the population and will be used in later sections to represent the Markov chain states.

4.3 MNC GA Crowding Selection: Diversifying the Mating pool

In this section we define the selection probability for any pair of individuals under crowding selection. Crowding selection selects the pair of individuals that will undergo crossover. For notational convenience we will identify the first individual selected into the pair as the *parent* and the other the *mate*. The selection step of MNC GA, described in Section 3.1, can be summarized as follows. For each mating pair the parent is chosen at random from the population and its mate is chosen using crowding. That is, the mate is chosen as the one that is most similar to the parent from a group of C_c candidates taken at random from the population, one at a time, with

replacement. It is not hard to see that fitness plays no role in this selection step.

Let $u = [u_0 \ u_1 \ \dots \ u_{N-1}]^T$ be the vector defining the parent selection probability for each chromosome in Ω . That is u_j , the j^{th} element, is the probability that chromosome j is selected as the parent for mating. We emphasize that the quantity u_j refers only to the probability of selecting a parent, not its mate. Notice also that in MNC GA, it is always true that

$$u = \frac{1}{n} p,$$

where u and p are vectors and n is a scalar. From this formulation it is clear that

$$\sum_{j=0}^{N-1} u_j = 1.$$

To facilitate the analysis of crowding selection, we choose the $N \times N$ matrix W , whose (j,k) th element, namely $W_{j,k}$, represents the probability that chromosome k is selected as the mate of parent j using crowding selection. Recall that in crowding selection the mate is the most similar individual to parent j from a group of C_c candidates taken at random (with replacement) from the population. To calculate the values of the entries $W_{j,k}$ we first need to rank the members of the population according to their similarity to parent j . Towards this end we define a $N \times N$ matrix R whose entries $R_{j,k}$ give the similarity ranking of the chromosomes in the population at iteration i . To be

more precise, the j th row of R lists the similarity ranking of the chromosomes in the population with respect to parent j . To simplify ranking we also assume that a simple linear transformation exists between a chromosome value (genotype) and the corresponding coordinate value (phenotype). Given these assumptions the similarity (or distance) between any two individuals in the population can be obtained using their chromosome tag values.

For the j th row of R , the ranking with respect to chromosome j is obtained by sorting the members of the population in ascending order of their distance to chromosome j (assume for now that there are no ties). Then, a rank value ranging from 0 to $n-1$ is assigned to the sorted list. The value of 0 is given to the population member closest to j (always itself) and $n-1$ to the one farthest from j . The smaller the distance, the more similar a chromosome is and the lower its rank value. The value $R_{j,k}$ denotes the similarity ranking of k with respect to parent j . That is, it refers to the number of individuals with lower rank with respect to chromosome k , i.e., there are $R_{j,k}$ population members closer to j (lower distance) than chromosome k .

Let e_j be a column vector of size N with a 1 at position j and 0 in all other positions. Define the vector $E_{j,k}$ as follows

$$E_{j,k} = \begin{cases} \sum_{m=j}^k e_m & \text{where } j \leq k \\ 0 & \text{otherwise} \end{cases}$$

The vector $E_{j,k}$ denotes a column vector of size N with 1s in all positions from j to k , including j and k . To get the number of individuals in the population with lower distance to j than k , i.e., the value of $R_{j,k}$ ($j \neq k$), we only need to know how many copies of the chromosomes between a and b are in the population, where

$$a = \text{Max}(j - \text{Abs}(j-k) + 1, 0) \text{ and}$$

$$b = \text{Min}(j + \text{Abs}(j-k) - 1, N-1).$$

Only those chromosomes with tag values $a, a+1, a+2, \dots, b-1$, and b will have a smaller distance to j than k .

$$p = [p_0 \ p_1 \ \dots \ p_a \ p_{a+1} \ \dots \ p_b \ \dots \ p_{N-1}]$$

$\underbrace{\hspace{10em}}_{E_{a,b}}$

Figure 4.3: Interval of vector p accounted by $E_{a,b}$ with chromosomes having a lower distance to chromosome j than chromosome k .

Figure 4.3 outlines the interval in p accounted by $E_{a,b}$. Using these values we have

$$R_{j,k} = \phi(j) \phi(k) (E_{a,b})^T p, \text{ where } \phi(j) = \begin{cases} 1 & \text{if } p_j \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

In a population where any chromosome appears at most one time, the sum of row R_j , where j is the lowest chromosome tag in the population, is equal to $n(n-1) / 2$. The same applies to row R_k , where k is the highest chromosome tag in the population.

To calculate $W_{j,k}$, the probability that chromosome k is selected as the mate of parent j under crowding selection, it is only necessary to know how many of the n^{C_s} (i.e., n to the power C_s) possible crowding selection groups will have chromosome k as the lowest rank member of the group. This value can be obtained by adding all possible groups where chromosome k appears among higher rank members of the population. In such cases chromosome k appears at least once and up to C_s times in that group. All other positions in the group are filled with an arbitrary combination of the higher ranked chromosomes. Given that chromosome k has rank $R_{j,k}$ we have $(n - R_{j,k} - 1)$ chromosomes with higher rank. The number of ways of arranging m copies of k in a group with C_s positions is given by $\binom{C_s}{m}$. The number of ways of selecting $(n - R_{j,k} - 1)$ higher rank chromosomes taking one at a time with replacement for the remaining $C_s - m$ positions is given by $(n - R_{j,k} - 1)^{C_s - m}$. Adding over all possible values of m , the number of times chromosome k is in the crowding selection group we get the following expression;

$$W_{j,k} = \frac{1}{n^{C_s}} \sum_{m=1}^{C_s} \binom{C_s}{m} (n - R_{j,k} - 1)^{C_s - m},$$

which is easily evaluated using the binomial theorem, to yield

$$W_{j,k} = \frac{(n - R_{j,k})^{C_s} - (n - R_{j,k} - 1)^{C_s}}{n^{C_s}}.$$

Equation 4.1

Equation 4.1 does not account for chromosomes having the same distance to parent j as mate k or duplicate copies of k . In these cases we have the chromosomes with the same rank value (ties). If we break ties at random we can calculate $W_{j,k}$ by averaging over all possible rankings of chromosome k and multiplying the result by the number of copies of k . Let $K_{j,k}$ represent the number of chromosomes in the population with the same distance to j as k (including all copies of chromosome k). Using the values of a and b defined above, we have that only chromosomes $(a - 1)$ or $(b + 1)$ have the same rank as individual k , and

$$K_{j,k} = \phi(j) \phi(k) (e_{a-1} + e_{b+1})^T p.$$

When $(a - 1)$ or $(b + 1)$ fall outside the interval $[0, N-1]$ the vector e_{a-1} or e_{b+1} respectively is considered the vector with all entries equal to 0. Using $R_{j,k}$ to represent the number of population members with lower rank, we can expand Equation 4.1 to obtain the average probability under crowding selection and get

$$\begin{aligned} W_{j,k} &= \frac{p_k}{K_{j,k} (n^{C_s})} \sum_{m=0}^{K_{j,k}-1} (n - (R_{j,k} + m))^{C_s} - (n - (R_{j,k} + m) - 1)^{C_s} \\ &= p_k \frac{(n - R_{j,k})^{C_s} - (n - R_{j,k} - K_{j,k})^{C_s}}{K_{j,k} (n^{C_s})} \end{aligned}$$

Equation 4.2

4.3.1 Example Population: Selection

In this section we go back to our sample population $p = [0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]^T$, and calculate the similarity ranking (K and R), parent probability (u), and mate probability (W) for all individuals in the population. For this population, $u = [0, 1/6, 0, 1/6, 2/6, 1/6, 0, 1/6]^T$. Figure 4.4 shows the entries for matrices K and R using the euclidean distance between the chromosome tag values to obtain the similarity ranking. Both matrices start at row 0 and column 0. To make things easier to visualize, chromosomes not in the population ($p_j = 0$ or $p_k = 0$) have matrix values of 0 and are shown with dashes in both matrices in Figure 4.4. Take for example entry (3, 1), shown in bold, in both matrices. The entry $K_{3,1} = 2$ (also $K_{3,5} = 2$) indicates there are two chromosomes in the population, namely 1 and 5, with the same distance to chromosome 3. The entry $R_{3,1} = 3$ (also $R_{3,1} = 3$) indicates there are three chromosomes in the population, 3, 4, and 4, more similar (lower distance and rank value) to chromosome 3 than chromosome 1 (5).

$$K = \begin{bmatrix} - & - & - & - & - & - & - & - \\ - & 1 & - & 1 & 2 & 1 & - & 1 \\ - & - & - & - & - & - & - & - \\ - & \mathbf{2} & - & 1 & 2 & 2 & - & 1 \\ - & 2 & - & 2 & 2 & 2 & - & 2 \\ - & 1 & - & 2 & 2 & 1 & - & 2 \\ - & - & - & - & - & - & - & - \\ - & 1 & - & 1 & 2 & 1 & - & 1 \end{bmatrix} \quad R = \begin{bmatrix} - & - & - & - & - & - & - & - \\ - & 0 & - & 1 & 2 & 4 & - & 5 \\ - & - & - & - & - & - & - & - \\ - & \mathbf{3} & - & 0 & 1 & 3 & - & 5 \\ - & 4 & - & 2 & 0 & 2 & - & 4 \\ - & 5 & - & 3 & 1 & 0 & - & 3 \\ - & - & - & - & - & - & - & - \\ - & 5 & - & 4 & 2 & 1 & - & 0 \end{bmatrix}$$

Figure 4.4: Number of individuals in the population with similar ranking (K) and lower ranking (R) for chromosomes in the sample population.

Using Equation 4.2 and the matrices in Figure 4.4, we can calculate the matrix W for our sample population for any value of C_i . A trivial case occurs when $C_i = 1$, that is, when all individuals in the population have the same probability, $1/n$, of being selected as a mate. In this case every row of W (whose row value is a chromosome in the population) will be identical, each being equal to u^T . Figure 4.5 shows the more interesting case of matrix W when $C_i = 2$.

$$W = \frac{1}{36} \begin{bmatrix} - & - & - & - & - & - & - & - \\ - & 11 & - & 9 & 12 & 3 & - & 1 \\ - & - & - & - & - & - & - & - \\ - & 4 & - & 11 & 16 & 4 & - & 1 \\ - & 2 & - & 6 & 20 & 6 & - & 2 \\ - & 1 & - & 4 & 16 & 11 & - & 4 \\ - & - & - & - & - & - & - & - \\ - & 1 & - & 3 & 12 & 9 & - & 11 \end{bmatrix}$$

Figure 4.5: Mate probability under crowding selection for a value of $C_i = 2$.

As one can observe from matrix W , for a given parent j (j th row of W) the mates with the higher probability correspond to those chromosomes closer to j , i.e., entries closer to the diagonal W_{jj} .

So far we have defined the parent selection probability, u , and the mate selection probability for any parent, W . We summarize these results by defining the selection probability for any pair of chromosomes j (parent) and k (mate) under crowding selection with the equation

$$P_*(j, k) = u_j W_{j,k}.$$

Equation 4.3

4.4 MNC GA Mating: Interval Crossover and Mutation

In this section we continue the analysis of the MNC GA by considering the result obtained in the mating step after applying crossover and mutation. Specifically, we want to define the probability of obtaining chromosome m after applying interval crossover to the pair of chromosomes j and k and bit mutation on the resulting offspring. In interval crossover only one offspring is generated. For each pair of parent chromosomes, j and k , (assume without loss of generality that the tag value of j is less than or equal to the tag value of k), the offspring's chromosome is selected at random from the interval

$$[c = \text{Max}(j - \varepsilon/2, 0), d = \text{Min}(k + \varepsilon/2, N-1)],$$

where ε is a user selected parameter indicating the minimum size of the interval. When either $j < \varepsilon/2$ or $k > N - \varepsilon/2$ the interval is reduced to make certain that it is contained completely in Ω . If both $j < \varepsilon/2$ and $k > N - \varepsilon/2$ then the interval becomes the entire domain Ω . To prevent interval crossover from resembling random search, a small ε value and two chromosomes j and k that are close to each other (close in the sense of Euclidean distance) are needed. Since the mating pair is selected using crowding selection, it is more likely that the Euclidean distance between them is small.

Given the above description of interval crossover we can see that only chromosomes whose "values" lie in the interval $[c, d]$ have a non-zero probability of being selected as an offspring; all other values have a zero probability. Moreover, all chromosomes in the interval have the same probability of being selected. Let χ be the probability that interval crossover is applied. Define $C_{j,k}(m)$, the probability of generating chromosome m from chromosomes j and k using interval crossover, by

$$C_{j,k}(m) = \varphi(m) \frac{\chi}{(d-c+1)}$$

$$\text{where } \varphi(m) = \begin{cases} 1 & \text{if } m \in [c, d] \\ 0 & \text{otherwise} \end{cases}.$$

We now need to calculate the probability that chromosome m' results after mutation of offspring m . Mutation changes any one bit in the chromosome with probability μ . The differences between two chromosomes are given by the Hamming distance between them. Let $\eta_{m,m'}$ denote the Hamming distance between chromosomes m and m' . Then the probability that we obtain chromosome m' after mutation of chromosome m is

$$M_{m,m'} = \mu^{\eta_{m,m'}} (1 - \mu)^{l - \eta_{m,m'}},$$

where l is the chromosome length.

Using $C_{j,k}(m)$ and $M_{m,m'}$, we now calculate the probability that any chromosome m is generated after mating in the MNC GA. There are two cases: (1) the chromosome is generated by interval crossover and no mutation

occurs and (2) interval crossover generates a different chromosome, and mutation changes it to m . Adding over all chromosomes in obtain

$$P_m(j, k, m) = \sum_{i=0}^{N-1} C_{j,k}(i) M_{i,m} = \sum_{i=c}^d C_{j,k}(i) M_{i,m},$$

Equation 4.4

the probability of generating chromosome m from chromosomes j and k after the mating step in the MNC GA. For any two chromosomes j and k in Ω , $j \leq k$, the value $P_m(j, k, m)$, calculated using Equation 4.4, gives the probability of generating any chromosome value $m \in \Omega$ under interval crossover and bit mutation.

$j \backslash k$	0	1	2	3	4	5	6	7
0	0.0045	0.0273	0.0225	0.0326	0.0285	0.1182	0.1125	0.1125
1	0.0273	0.0273	0.0225	0.0326	0.0285	0.1182	0.1125	0.1125
2	0.0225	0.0225	0.0273	0.0387	0.0326	0.1365	0.1274	0.1274
3	0.0326	0.0326	0.0387	0.0513	0.0405	0.1636	0.1485	0.1485
4	0.0285	0.0285	0.0326	0.0405	0.0297	0.1863	0.1636	0.1636
5	0.1182	0.1182	0.1365	0.1636	0.1863	0.2457	0.2025	0.2025
6	0.1125	0.1125	0.1274	0.1485	0.1636	0.2025	0.2457	0.2457
7	0.1125	0.1125	0.1274	0.1485	0.1636	0.2025	0.2457	0.3645

Figure 4.6: Probability that offspring with tag value 6 is generated after applying interval crossover and bit mutation, with parameters $\chi = 0.9$, $\mu = 0.1$, and $\varepsilon = 2$, for any chromosome pair (j, k) .

4.4.1 Example Population: Mating

In this section we return to our sample population $p = [0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]^T$ and examine the probability of generating offspring with tag value 6 (chromosome 110) under interval crossover and bit mutation. Using Equation 4.4 we need to calculate $C_{j,k}(i)$ and $M_{i,6}$ for all pairs of chromosomes in the

population. We set the crossover probability at $\chi = 0.9$, the mutation probability at $\mu = 0.1$, and the minimum interval size at $\epsilon = 2$. Using these parameters Figure 4.6 shows the probability for any pair of chromosomes.

4.5 MNC GA Replacement: Encouraging Speciation and Niching

In this section we calculate, for all individuals in the population, $P(j, k)$, the probability that chromosome k is replaced by offspring j under worst among most similar (WAMS) replacement. Recall that during replacement the MNC GA creates C_r crowding groups with s individuals each, chosen at random with replacement, from the population. Then the individual most similar to the offspring in each crowding group is chosen to form the crowding factor group, of C_r most similar candidates. From the crowding factor group the lowest fitness individual is replaced by the offspring in the population.

It is not hard to see a relation between the replacement step and crowding selection. Each one of the individuals in the crowding factor group is selected using crowding, but here the similarity ranking is based on the offspring. Therefore the creation of the crowding factor group can be viewed as the application of crowding selection C_r times with $C_s = s$ and the offspring being the parent. Given this relationship we can use the results from Section 4.3 to get the probability that a chromosome in the population is selected from one of the crowding groups.

Let V be an $N \times N$ matrix whose entries define the crowding selection probability of the individuals in the population for any chromosome Ω . That is, the entry $V_{j,k}$, denotes the probability of selecting chromosome k as the most similar candidate to offspring j from a crowding group, of size s , chosen at random with replacement from the population. As was done in the analysis of crowding selection, let us define two $N \times N$ matrices Q and J , to indicate the number of individuals with lower and equal similarity ranking respectively for all individuals in the population. The entry $Q_{j,k}$ defines the number of individuals in the population with lower similarity ranking than chromosome k with respect to offspring j in Ω . In the same manner the entry $J_{j,k}$ defines the number of individuals in the population with equal similarity ranking as chromosome k with respect to offspring j in Ω . Using Equation 4.2 we get

$$V_{j,k} = p_k \frac{(n - Q_{j,k})^s - (n - Q_{j,k} - J_{j,k})^s}{J_{j,k} (n^s)}.$$

The entries of the matrix Q can be obtained from $Q_{j,k} = (E_{a,b})^T p$. Similarly each entry of the matrix J can be obtained from $J_{j,k} = (e_{a-1} + e_{b+1})^T p$. The distinction between the matrices V and W , defined in Equation 4.2, is that V has nonzero values in all rows, because the ranking is done for all chromosomes in Ω . Formally, $V_{j,k}$ is valid for $p_k \neq 0$, where as $W_{j,k}$ have an additional constraint that $p_j \neq 0$.

Now we need to define the probability of selecting a chromosome from the crowding factor group. Let r be the column vector of size N representing the fitness ranking of the individuals in the population. The entry r_j indicates the number of individuals in the population with higher fitness than chromosome j . The most fit individual is assigned a rank value of 0 and the least fit individual a rank value of $n - 1$. Here again we break ties at random. Let f be the size N vector defining the fitness values for the all the chromosomes in Ω . The entry f_j represents the fitness value $F(x_j)$, where x_j is the real value given by chromosome j . Then we can calculate r_k with the following equation:

$$r_k = \phi(k) \sum_{j=0}^{N-1} p_j \theta_1(j, k), \text{ where } \theta_1(j, k) = \begin{cases} 1 & \text{if } f_j > f_k \\ 0 & \text{otherwise} \end{cases}$$

Next, define the vector κ , whose entries κ_k indicate the number of individuals in the population with equal fitness value as chromosome k . Using the fitness vector f we can calculate the values of κ using the following equation:

$$\kappa_k = \phi(k) \sum_{j=0}^{N-1} p_j \theta_2(j, k), \text{ where } \theta_2(j, k) = \begin{cases} 1 & \text{if } f_j = f_k \\ 0 & \text{otherwise} \end{cases}$$

To calculate $P_r(j, k)$, the probability of replacing chromosome k with offspring j , we need to know all possible permutations of the crowding factor group where chromosome k has the lowest rank (highest fitness) value. Then

we add the probability of each of the crowding factor groups to obtain $P_r(j, k)$. Unlike forming the crowding groups, the probability of being selected to the crowding factor group is not random, but based in the similarity to offspring j and given by V . Given offspring j the probability of a particular crowding factor group $(I_1, I_2, \dots, I_{C_f})$, where I_k is a chromosome in the population, is given by the product

$$\Lambda(j, I_1, I_2, \dots, I_{C_f}) = \prod_{k=1}^{C_f} \frac{V_{j, I_k}}{P_{I_k}}.$$

Enumerating all possible crowding factor groups where chromosome k has the lowest rank and adding the probability of each group will give us the value of $P_r(j, k)$. For chromosomes with equal fitness rank, we need to average over all possible fitness rank values to obtain the actual value of $P_r(j, k)$. Assume that the chromosomes $(I_1, I_2, \dots, I_{r_k})$ and $(I_{r_k+1}, \dots, I_{r_k+\kappa_k-p_k})$ are the individuals in the population with higher and equal rank (not including the copies of chromosome k) respectively. Using Λ and the vectors r and κ we can calculate the probability of selection of any chromosome in the population under WAMS replacement with the following equation:

$$P_r(j, k) = \frac{1}{\kappa_k} \sum_{l=0}^{\kappa_k-p_k} \sum_{m=1}^{C_f} \binom{C_f}{m} \left(\frac{V_{j,k}}{P_k} \right)^m \left(\sum_{\ell_1=1}^{h+l} \sum_{\ell_2=1}^{h+l} \dots \sum_{\ell_{C_f-m}=1}^{h+l} \Lambda(j, I_{\ell_1}, I_{\ell_2}, \dots, I_{\ell_{C_f-m}}) \right)$$

Equation 4.5

where $h = n - r_k - 1$.

4.5.1 Example Population: WAMS Replacement

In this section we return to our sample population $p = [0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]^T$ and examine the replacement probability for all individuals in the population. The matrix Q and matrix J , shown in Figure 4.7 show the similarity ranking during replacement for all the chromosomes in the population. Observe that the matrices J and Q are similar to the matrices K and R respectively shown in Figure 4.4. This is not a coincidence, but the result of using crowding during replacement also. The main difference is that all rows in Q and J have some non-zero values because the similarity ranking during replacement is done for all chromosomes in Ω where as selection uses only the chromosomes in the population.

$$J = \begin{bmatrix} - & 1 & - & 1 & 2 & 1 & - & 1 \\ - & 1 & - & 1 & 2 & 1 & - & 1 \\ - & 2 & - & 2 & 2 & 1 & - & 1 \\ - & 2 & - & 1 & 2 & 2 & - & 1 \\ - & 2 & - & 2 & 2 & 2 & - & 2 \\ - & 1 & - & 2 & 2 & 1 & - & 2 \\ - & 1 & - & 1 & 2 & 2 & - & 2 \\ - & 1 & - & 1 & 2 & 1 & - & 1 \end{bmatrix} \quad Q = \begin{bmatrix} - & 0 & - & 1 & 2 & 4 & - & 5 \\ - & 0 & - & 1 & 2 & 4 & - & 5 \\ - & 0 & - & 0 & 2 & 4 & - & 5 \\ - & 3 & - & 0 & 1 & 3 & - & 5 \\ - & 4 & - & 2 & 0 & 2 & - & 4 \\ - & 5 & - & 3 & 1 & 0 & - & 3 \\ - & 5 & - & 4 & 2 & 0 & - & 0 \\ - & 5 & - & 4 & 2 & 1 & - & 0 \end{bmatrix}$$

Figure 4.7: Number of individuals in population with equal (J) and lower similarity ranking for all chromosomes in Ω

Using the matrix values for Q and J shown in Figure 4.7 the values for V , the crowding probability for the chromosomes in the population, can be calculated using the equation shown above. A trivial case exists when the value of the crowding group size, s , is 1. In this case all rows in V are

identical to each other and equal to u^T , the parent selection probability. In this case ($s = 1$) all individuals have the same probability of being selected for replacement. Figure 4.9 shows the case $s = 2$. Again we can observe an analogous relation between matrix V and matrix W (shown in Figure 4.5).

Figure 4.8: Graph of sample fitness function. Two maxima at 0 and 6.

To calculate the fitness ranking for the individuals in the population we need to define the fitness for all the chromosomes in Ω . Let

$$f = [4 \ 3 \ 2 \ 1 \ 2 \ 3 \ 4 \ 3]^T$$

define the fitness for all the chromosomes in Ω (Graph shown in Figure 4.8).

Using f and the equations for r and κ shown above we have

$$r = [0 \ 0 \ 0 \ 5 \ 3 \ 0 \ 0 \ 0]^T \text{ and}$$

$$\kappa = [0 \ 3 \ 0 \ 1 \ 2 \ 3 \ 0 \ 3]^T.$$

Knowing the values of the vectors r and κ , and the matrix V we can calculate $P_r(j, k)$, the probability under WAMS replacement, for any value of C_r using Equation 4.5. A trivial case occurs for $C_r = 1$. In this case the values of $P_r(j, k) = V_{j,k}$, which means that replacement is based on similarity only and

fitness plays no role. When $s = 1$ and $C_r > 1$, we have the situation where replacement is based on fitness only, and similarity plays no role. Figure 4.10 shows the values for $P_r(j, k)$ for the case where $C_r = 2$.

$$P_r = \frac{1}{36} \begin{bmatrix} - & 11 & - & 9 & 12 & 3 & - & 1 \\ - & 11 & - & 9 & 12 & 3 & - & 1 \\ - & 10 & - & 10 & 12 & 3 & - & 1 \\ - & 4 & - & 11 & 16 & 4 & - & 1 \\ - & 2 & - & 6 & 20 & 6 & - & 2 \\ - & 1 & - & 4 & 16 & 11 & - & 4 \\ - & 1 & - & 3 & 12 & 10 & - & 10 \\ - & 1 & - & 3 & 12 & 9 & - & 11 \end{bmatrix}$$

Figure 4.9: Crowding probability during replacement with $s = 2$.

In Figure 4.10 we can observe the combined fitness pressure and niching pressure that the WAMS operator applies to the chromosomes in the population. For example, examination of the entries in the third row of P_r (replacement probability of chromosomes in the population when chromosome with tag value 2 is generated after mating) reveals that chromosomes with tag values 1, 3, and 4 are more likely to be replaced by chromosome 2 (remember that tag values range from 0 to $N - 1$). That is due, in part, to the similarity (closeness) of those chromosomes to chromosome 2. Among the chromosomes (1, 3, and 4) we can also observe that even though chromosome 1 is closer to offspring 2, chromosome 4 (two copies, each having probability $240 / 36^2$) has a higher probability of replacement. This is due to the lower fitness value of chromosome 4 and the fitness pressure applied by

WAMS during replacement. Different values of s and C_r will change the niching and fitness pressure respectively of the WAMS operator. A higher value of s will increase replacement within a more localized region while a higher C_r value will increase replacement of lower fit chromosomes.

$$P_r = \frac{1}{36^2} \begin{bmatrix} - & 165 & - & 567 & 504 & 45 & - & 15 \\ - & 165 & - & 567 & 504 & 45 & - & 15 \\ - & 140 & - & 620 & 480 & 42 & - & 14 \\ - & 36 & - & 671 & 544 & 36 & - & 9 \\ - & 20 & - & 396 & 800 & 60 & - & 20 \\ - & 16 & - & 272 & 768 & 176 & - & 64 \\ - & 21 & - & 207 & 648 & 210 & - & 210 \\ - & 21 & - & 207 & 648 & 189 & - & 231 \end{bmatrix}$$

Figure 4.10: WAMS replacement probability for $C_r = 2$.

4.6 The Markov Matrix: Bringing Everything Together

In this section we use the results from the previous sections to define the transition probability between any two states after one iteration of the MNC GA. That is, given population p what is the probability that population p' results after the application of crowding selection, mating, and WAMS replacement. Summarizing the results found so far, from Equation 4.3 we have that the probability that two chromosomes j and k are selected for mating using crowding selection is given by $P_c(j, k)$. Then, the probability of generating offspring m after the mating step (using interval crossover and bit mutation) is given in Equation 4.4 and denoted by $P_m(j, k, m)$. Finally, Equation 4.5 defines $P_r(m, k)$, the probability that offspring m replaces

chromosome k under WAMS replacement. Using these results we can construct the Markov chain matrix to define the transition probability between any two states of the population after one iteration of the MNC GA.

Assuming that the resulting population after an iteration of the MNC GA only depends on the current population we can use the vector p to denote uniquely the states of the algorithm. There are a total of N^n populations, given by all possible representations of vector I , that can be represented by p in

$$h = \binom{n+N-1}{N-1}$$

unique states. For the sample problem used throughout this section we have $n = 6$ and $N = 8$ for a total of $h = 1716$ states. A big number, even for this simple problem. Let S_j , ($0 \leq j \leq h - 1$), denote state j of the population which corresponds to one of the population vectors p . The entry $S_{j,k} = p_k$, denotes the number of chromosomes with tag value k in state j .

To calculate the transition probability between any two states, say S_m and S_n , we need only to observe what occurs to the population after one iteration of the MNC GA. Given $S_m = p^{(i)}$ and $S_n = p^{(i+1)}$, the populations in the i th and $i+1$ iterations respectively, after applying crowding selection, mating, and WAMS replacement steps we have that $p^{(i+1)} = p^{(i)}$ or $p^{(i+1)} = p^{(i)} + e_j - e_k$. That is, the population remains the same or a copy of offspring j replaces a copy of chromosome k . The population will not change when either crossover does

not take place, this occurs with probability $1 - \chi$, or the offspring replaces a copy of himself, i.e., $j = k$. Therefore the transition probability between two states S_m and $S_{m'}$, $P\{S_m \rightarrow S_{m'}\}$, is non-zero only when $S_{m'} - S_m = (e_j - e_k)^T$ and $S_{m,k} > 0$, for any chromosomes j and k in Ω . The probability that offspring j replaces individual k in the population after one generation of the MNC GA is given by

$$P_g(j, k) = \sum_{i_1=0}^{N-1} \sum_{i_2=0}^{N-1} P_s(i_1, i_2) P_m(i_1, i_2, j) P_r(j, k)$$

for any chromosomes j and k in Ω . The transition probability between states m and m' is then given by

$$P\{S_m \rightarrow S_{m'}\} = \begin{cases} P_g(j, k) & \text{for } m \neq m', S_m - S_{m'} = (e_j - e_k)^T \\ (1 - \chi) + \sum_{j=0}^{N-1} P_g(j, j) & \text{for } m = m' \\ 0 & \text{otherwise} \end{cases}$$

Equation 4.6

In other words, the transition probability between a state and its reachable next states is given by the probability of generating offspring j , for any parent (i_1) and mate (i_2) in the population, times the probability that offspring j replaces chromosome k . From this equation we can also deduce that for any state S_m with i entries greater than zero (i.e., i different chromosomes in the population) there are $i * (N - 1) + 1$ possible transition states. Each chromosome i in the population can be replaced by $N - 1$

offspring in Ω different from i to form a new state or the state remains the same because crossover did not occur or chromosome i was replaced by a copy of himself. We can also observe that the probability that the population stays the same, $P(S_m \rightarrow S_m)$, is greater or equal to $(1-\chi)$. Choosing low values for the crossover probability, χ , will cause the population to stagnate.

4.6.1 Example Population: The State Transition Probabilities

In this section we show the state transition probability for a subset of reachable states from our sample population, $p = [0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]^T$, using Equation 4.6. The parameters for the equation are taken from our previous results. The values are; $C_s = 2$, $s = 2$, $C_r = 2$, $\chi = 0.9$, $\mu = 0.1$, and $\varepsilon = 2$. Of the possible, $5 * (8 - 1) + 1 = 36$, state transitions Table 4.1 shows only those states where a copy of individual with tag value 4 is replaced by any offspring with tag value distinct from 4. The last row shows the probability that the population does not change.

Table 4.1: Sample state transition probabilities in the MNC GA.

Current State	Next State	Offspring	Individual	Probability
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1]$	0	4	0.0239
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 2 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1]$	1	4	0.0253
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1]$	2	4	0.0363
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 1 \ 0 \ 2 \ 1 \ 1 \ 0 \ 1]$	3	4	0.0620
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 1 \ 0 \ 1 \ 1 \ 2 \ 0 \ 1]$	5	4	0.0940
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1]$	6	4	0.0552
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 2]$	7	4	0.0399
$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	$[0 \ 1 \ 0 \ 1 \ 2 \ 1 \ 0 \ 1]$	NA	NA	0.3308

Due to the niching pressure caused by the WAMS replacement operator we have that the next state is most likely to be one where the new offspring replaces a similar individual. From Table 4.1 we can observe that offspring with tag value 3, 5, or 6 are more likely to replace individual with tag value 4. The large probability for the case where the state does not change is due to the probability that crossover does not occur (0.1) plus the probability that any offspring replaces an individual with the same tag value.

4.7 A Closer Look at Crowding Selection

In this section we examine the properties of crowding selection more closely. Specifically, we calculate under crowding selection the bounds for the selection probability (W) of the mate as well as the expected rank and variance. Using Equation 4.1 we can calculate the lower and upper bounds of $W_{j,k}$, the probability of selecting chromosome k using crowding selection given that chromosome j is the parent. The lower bound can be obtained from the highest rank value ($R_{j,k} = n-1$) and the upper bound from the lowest rank value ($R_{j,k} = 0$). Using these values we have

$$\frac{1}{nC_s} \leq W_{j,k} \leq \frac{n^{C_s} - (n-1)^{C_s}}{nC_s}.$$

Take for example the case $n = 10$ and $C_s = 2$, we have $0.01 \leq W_{j,k} \leq 0.19$. Although there is a small probability of selecting the highest ranked individual, it is non-zero and given enough trials it will be selected.

It is not hard to show that the row sum in W is equal to one for chromosomes in the population, i.e.,

$$\sum_{k=0}^{N-1} W_{j,k} = 1 \text{ when } p_j \neq 0.$$

This result can be proven by adding Equation 4.1 over all possible rank values to verify that is equal to the total number combinations under crowding selection (n^{C_s}). Since all possible rank values are in the range $[0, n-1]$ we have that

$$\sum_{m=0}^{n-1} (n-m)^{C_s} - (n-m-1)^{C_s} = n^{C_s}.$$

Of importance to us is the expected rank of a mate as the value of C_s (crowding selection group size) changes. Knowing the expected rank, for different values of C_s , will allow us to select an appropriate value for the number of optima in the function. Using Equation 4.1 the expected rank value $E(R_{j,k})$ of the mate for any parent is

$$E(R_{j,k}) = \frac{1}{n^{C_s}} \sum_{m=0}^{n-1} m[(n-m)^{C_s} - (n-m-1)^{C_s}] = \frac{1}{n^{C_s}} \sum_{m=1}^{n-1} (n-m)^{C_s} = \frac{1}{n^{C_s}} \sum_{m=1}^{n-1} m^{C_s}.$$

In the same manner we can calculate the variance $Var(R_{j,k})$ to get

$$Var(R_{j,k}) = E(R_{j,k}^2) - E(R_{j,k})^2 = \frac{1}{n^{C_s}} \sum_{m=1}^{n-1} (2n-2m-1)m^{C_s} - \left(\frac{1}{n^{C_s}} \sum_{m=1}^{n-1} m^{C_s}\right)^2.$$

From the variance we can easily calculate the square root to obtain the standard deviation $STD(R_{j,k}) = \text{SQRT}(\text{Var}(R_{j,k}))$.

Figure 4.11: Expected rank value, variance, and standard deviation of the mate under crowding selection as the function of the crowding selection size using a population size $n = 100$.

Of importance to us is the expected rank value of the mate under crowding selection as the crowding selection size changes. It allows us to observe the niching pressure imparted during selection by the MNC GA. Figure 4.11 presents such information graphically for a population size of 100. Clearly, we can see that the expected rank value decreases as the crowding selection size increases. This indicates that during crowding selection a larger group size will more likely generate a mating pair from the same region. Given a population size we can calculate the group size that will give us the range of rank values that will promote mating among localized individuals.

Another way to look at the effect of the group size in crowding selection is by examining the probability distribution for the different rank values.

Figure 4.12 shows the probability distribution for a group size of 1, 6, 11, 16, and 21. Again, we can observe that a higher group size will increase the probability that a lower rank individual (which means more similar in our case) is selected as a mate. Similarly we can calculate for a given population size the group size that will have a certain probability for a particular rank value. These results allows us to select appropriate parameters to control how much localize mating we want for a particular problem.

Figure 4.12: Probability distribution for the rank values under crowding selection using a population of 100.

4.8 A Closer look at Worst Among Most Similar Replacement

In this section we examine the properties of WAMS replacement more closely. Specifically, we want to look at the effect of the parameters s and C_r on replacement. Recall that in WAMS replacement C_r crowding groups, each with s individuals, are formed by choosing individuals at random (with replacement) from the population. Then the most similar individual (to the offspring) in each crowding group is selected to form the crowding factor

group with C_f individuals. The least fit individual in the crowding factor group is replaced by the offspring in the population.

Creating a crowding group is similar to crowding selection. All the results obtained in Section 4.7 apply directly to the creation and selection of individuals to crowding groups. In summary, increasing the value of the group size, in this case s , decreases the expected similarity rank of the individual, thus increasing the probability of selecting individuals from the same niche. During replacement this means that offspring are more likely to replace members of the same niche when using higher values of s . Let $P_s(u)$ denote the probability that an individual in the population with similarity rank u is selected from a crowding group. Using the results from crowding selection we have

$$P_s(u) = \frac{(n-u)^s - (n-u-1)^s}{n^s}.$$

Selection from the crowding *factor* group is somewhat different. Here the individual selected is the one with the lowest fitness in a group of C_f individuals. The probability that an individual with similarity rank u is selected to the crowding factor group, denoted by $P_c(u)$, is given by

$$P_c(u) = 1 - (1 - P_s(u))^{C_f}.$$

Once selected into the crowding *factor* group the individual replaced must be the one with the lowest fitness in the group. For simplicity, let us assume

that the fitnesses of all the individuals in the population are distinct and therefore we can assign a unique fitness rank from 0 to $n-1$ to each of them. Let us also assume that the fitness rank values are assigned in ascending order of fitness, that is, a value of 0 is assigned to the individual with the lowest fitness, 1 to the second lowest fitness, and so on until the most fit individual gets a fitness rank of $n-1$. Let $P_f(v)$ denote the probability that an individual with fitness rank v is selected from a group of C_f elements selected uniformly at random with replacement. Similar to crowding selection, but using fitness rank instead of similarity rank, we have that

$$P_f(v) = \frac{(n-v)^{C_f} - (n-v-1)^{C_f}}{n^{C_f}}.$$

Assuming that the fitness rank and the similarity rank are independent variables, we can calculate the probability that an individual with fitness rank v and similarity rank u is replaced by an offspring under WAMS replacement. The replacement probability, denoted $P_r(u,v)$, is given by

$$P_r(u,v) = P_c(u)P_f(v) = (1 - (1 - P_s(u))^{C_f})P_f(v).$$

From this equation we can calculate the expected fitness rank $E(v)$ under WAMS replacement. Using the results from the previous section we have

$$E(v) = (1 - (1 - P_s(u))^{C_f}) \frac{1}{n^{C_f}} \sum_{m=1}^{n-1} m^{C_f}.$$

This equation is similar to the expected similarity rank under crowding

selection with the exception of the additional term $1 - (1 - P_s(u))^{C_f}$. The variance can also be calculated in a similar manner.

There are various things that can be pointed out from these results. Increasing the crowding *factor* group size, C_r increases the probability that lower fitness individuals are replaced in the population. The size of the crowding group determines the likelihood that the offspring replaces similar individuals. Competition between members of different niches can be increased by lowering the value of s . The risk here is that niches with lowered average fitness may not be able to maintain any individuals in them. On the other hand increasing the value of s increases competition among members of the same niche. By using appropriate values of s and C_r we can increase replacement of low-fitness individuals from the same niche allowing the MNC GA to converge to the top of every niche. It is the WAMS replacement step that applies the “survival of the fittest” metaphor to the members of the population.

4.9 Comments on the Generational MNC GA

Before starting the analysis, we reiterate that the MNC GA can be implemented in two different ways: the generational MNC GA and the steady state MNC GA. These two versions are analogous to the classical GA (or SGA) (Holland, 1975) and the steady state GA (or SSGA) (Whitley, 1988; Syswerda, 1989) respectively. In a generational MNC GA the selection step is

applied n times (the size of the population) first to create all mating pairs prior to producing any offspring. Since the MNC GA uses crowding selection this allows each individual in the population to participate in mating at least once in each generation. Later, interval crossover is applied (with probability χ) to every mating pair to produce at most n offspring, at most one from each pair. For each offspring bit mutation is applied and then inserted in the population using WAMS (worst among most similar) replacement.

In contrast, in the steady state MNC GA each step is applied sequentially. First, a single mating pair is selected using crowding selection. Second, interval crossover is applied to the pair to generate an offspring to which bit mutation is applied. Finally the resulting offspring is inserted in the population using WAMS replacement. In this case some of the individuals in the population will be replaced prior to having a chance for mating. As we will see later, the offspring replacing the individual is likely to be “similar” to the individual when WAMS replacement is applied with appropriate parameters. Here we chose to focus on the steady state MNC GA to simplify analysis. Later on we will come back to the generational counterpart and provide a similar analysis using the results already obtained.

The difference between the steady state MNC GA and it's generational counterpart is mainly during selection. During selection, the generational MNC GA forms a mating pool of n mating pairs. Each of the individuals in I ,

the population, is selected as a parent once every generation and its mate is selected using crowding selection. Therefore we have all the mating pairs formed from the same population. Then the mating pairs undergo mating to generate up to n offspring. Finally all the offspring are inserted in the population using WAMS replacement. If we define a generation in a steady state MNC GA as the result of n iterations, i.e., application of selection, mating, and replacement n times, then the j^{th} parent selected at random, say I_j , from the original population must survive in the worst case $j - 1$ replacements before it participates in mating. The worst case occurs when individual I_j is never selected as a mate by crowding selection.

In summary, the generational MNC GA creates a mating pool of size n from which up to n offspring are generated. Then each offspring is inserted in the population using WAMS replacement. Using the same notation we can describe the steady state MNC GA. In the steady state approach a mating pool of size 1 is created to generate up to 1 offspring. Then the offspring is inserted in the population using WAMS replacement. From this analogy we can observe the main differences between the approaches. Although not done here we can use many of the results from the steady state MNC GA to analyze its generational counterpart.

5. Application of the MNC GA to DNA Mapping

In this chapter we examine the performance of the MNC GA while solving a problem arising in DNA mapping. The problem consists of determining the sequence of DNA clones from restriction-fragment data. This problem is known to be NP-hard and will be posed as the problem of finding the optima of a multi-modal function. The MNC GA is tested using two data sets obtained from the Human Genome Project at the Lawrence Livermore National Laboratory. The new method holds promise in automating the sequencing computations.

Section 5.1 gives a tutorial background on the Human Genome Project and the relevance of the restriction-fragment assembly problem. Section 5.2 summarizes the experimental procedure used by one group of biologists to gather the data used in this paper. Section 5.3 summarizes some of the related work on restriction fragment map assembly. Section 5.4 addresses the representational issues including a description of the genetic operators and the fitness function used to solve this problem. Section 5.5 includes results and discussion and Section 5.6 summarizes on-going work.

Because genetic algorithms are being applied in this chapter to solve a problem in genetics, a word of caution about the terminology used is in order. Words like "chromosomes" occur while describing the genetic algorithm as

well as during the description of the DNA fragment assembly process. To the biologist, a chromosome is DNA. To the computational scientist, a chromosome is a data structure. The context will make the use clear.

5.1 Background on the DNA Restriction-Fragment Assembly Problem

The purpose of this section is to provide a simplified summary of the biological background necessary to understand some of the central issues in the DNA restriction-fragment assembly problem as it arises in the Human Genome Project.

The genetic material contained in all the chromosomes of a cell is collectively called the genome. Chromosomes are essentially DNA molecules, the ladder shaped, double-helical structures. For the purpose of this application, it suffices to say that the most important parts of this double-helix are the "steps" of the ladder. These steps, called base-pairs, denoted here by the letters A, C, G, T, can theoretically form sixteen pairs. However, only AT, TA, CG and GC pairings are allowed. That is, if the left half of the base-pair is known, the right half is uniquely determined and vice versa. It is estimated that the human genome, contained in all 23 pairs of chromosomes, is comprised of about three billion base-pairs. A hypothetical sequence of these may look like AATCTTCGGGCCT.... occupying three billion positions. Specific sub sequences of this are called genes.

The monumental task of the Human Genome Project is to (a) associate with each gene all the properties controlled by that gene, (b) associate each gene with one of the 23 chromosomes in the body, (c) identify the exact positioning of a gene on a chromosome - known as the mapping problem or the genetic-linkage problem, and (d) decipher the exact sequence of base pairs that constitute a given gene - known as the sequencing problem. Although each human is uniquely characterized by his/her genome, apparently one individual differs from another in only a small percentage (about 0.2%) of this material. For that matter, the human genome differs from the simian genome by only a few percentage points. Thus the Human Genome Project is only analyzing a sort of composite genome: 23 chromosome pairs donated by a few European and U. S. scientists. Thus, the focus is on understanding the common structure that runs through the human species, although the actual DNA used in the experiments may come from a specific individual.

Issues related to molecular biology, instrumentation and computations do play a critical role in this effort. It is impractical even to attempt to summarize the scope of this project here. A description of the technique used, at the Human Genome Center of the Lawrence Livermore National Laboratory (LLNL), for example, can be found in *Genomics*, 4, pp 129-136 (Carrano et. al., 1989). To view on-line information about the LLNL genome program see <http://www-bio.llnl.gov/bbrp/genome.html>. This Center is

mapping human chromosome 19 which is estimated to be approximately 60 million base-pairs long, a relatively small-sized chromosome.

Current technology is forcing us to limit the sequencing task to small fragments of DNA that are composed of approximately 0.5K base-pairs (Istvanick et. al., 1993). In order to divide the DNA into fragments up to this resolution level, techniques using restriction enzymes are used. The restriction enzymes act on the DNA at specific locations which are randomly distributed along the length of the chromosome. Depending on the number of different restriction enzymes used in obtaining the fragments, the data are called single-digest (one enzyme), double-digest (two enzymes), or n-digest (n enzymes) data. Most mappings are done using single- and double-digest data. Scientists use different restriction enzymes to obtain DNA fragments of the appropriate size.

In practice, scientists try to map one chromosome at a time. Many techniques generally require: (a) purifying chromosomal DNA, (b) cutting the DNA into pieces called contigs using restriction enzymes, (c) inserting contigs into DNA cloning vectors, (d) inserting cloning vectors into bacterial host cells for multiplication, (e) cutting clones into fragments, (f) analyzing the fragments, and (g) reconstructing the original order of the base-pairs on the chromosome.

5.2 Restriction Fragment Data for Chromosome 19

As the data used in this study are obtained from the Human Genome Center of LLNL (Carrano et. al., 1989), it is useful to summarize the techniques used at LLNL to gather these data.

Chromosomes are sorted via a laser flow sorter; restriction enzymes are used to shatter the sorted material into fragments. These fragments are inserted into a host vector which can then be grown in large quantities. Due to the properties of the cloning vector used at LLNL, our so-called cosmid clones average 40Kbp (forty thousand base pairs) of DNA each. It is important to note that in each such cloning step, all original order is lost.

The LLNL biologists cloned and selected 15,000 such cosmid clones for human chromosome 19. A "cosmid fingerprint" was obtained for each clone by subjecting them to a double-digest labeled with a fluorescent dye and measuring them on a modified DNA sequencing gel apparatus. These crude fingerprints were compared between all pairs (taking 2 days on a network of 40 workstations in parallel) and a set of 800 unordered islands or "contigs" were formed by an automated algorithm which also determined a near-minimal spanning path for each island. (Note that 15,000 40Kbp clones provides a nominal 10X depth of coverage for chromosome 19.) However, this set of islands remains unordered by this technique.

A wide range of other techniques, not covered here, is used to merge, order, and provide the distance between the contigs. This paper will focus on

one technique used to verify the accuracy of the contigs built via pair-wise overlap data, called EcoRI restriction fragment mapping after the name of the restriction enzyme used. A subset of the clones in a contig are chosen (i.e., the spanning path members plus other clones to attempt to get at least a 2X coverage across the contig) and digested, with fragment lengths in the range of about 0.5-20Kbp being generated, depending on the distribution of EcoRI sites within each cosmid. These maps are of great utility in locating and sequencing genes; since individual EcoRI fragments can be physically cut from gels and prepared for sequencing, the cost of sequencing a gene can often be reduced by an order of magnitude (assuming a gene can be localized to a ~4Kbp fragment, instead of having to sequence an entire 40Kbp clone).

The order of the fragments within each clone is of course unknown. At LLNL, the "EcoRI maps" are constructed rapidly by hand, since the existence of a reliable near-minimal spanning paths provides a tremendous hint towards obtaining a proper clone ordering. This in effect reduces a 2-D optimization problem to a 1-D one. Once clone order is known, fragment ordering within clones is a much simpler problem (note that fragment sizes can be measured to about 5%, and that fragments of the same size can occur in multiple unrelated positions within a single map.)

The process of initially assembling the contigs and thereby deriving the minimal spanning paths is itself quite slow and labor-intensive in terms of laboratory bench work. LLNL researchers were interested in seeing how well

a fully-automated system could do in assembling EcoRI maps if a priori contig information were not available. This is the problem discussed here: given a number of clones, in unknown order, each with a set of restriction fragments, also in unknown order, construct a maximum-likelihood map of that region, subject to the constraints that, in general, the cosmid clones should be contiguous in the resulting maps, and apart from end fragments, all fragments should overlap others in their vertical "stack" within about 5% of their lengths. If these maps could be built accurately over large regions (i.e., at least 200-500Kbp), a potential exists for reducing the overall cost of obtaining a high-resolution physical map.

Toward the goal stated above, in this chapter, we propose to establish the relative position of each cosmid clone on a contig by establishing the possible locations of the fragments in each cosmid clone in such a manner that the fragment overlap among the clones is maximized while a suitably defined total error between the overlapped fragments is minimized. There are other constraints such as the total length of the assembly be equal to the contig's original size. The problem is one of assembling cosmid clone sequences as shown in Figure 5.1. The problem is complicated further by the uncertainty in the data, the possibility of data loss (fragments of the same size are hard to distinguish during fingerprinting), and the known fact that data related to corner fragments (i.e., fragments near the fragment boundaries) is almost always unreliable. Problems of this type are known to be hard (Opatrny,

1979). For example, the case with 10 cosmid clones, there are $10! / 2$ possible clone sequences.

Figure 5.1: Physical map for island using fragments from a set of overlapping clones.

5.3 Related Work and Scope of Present Work

The DNA restriction fragment assembly, the subject matter of this chapter resembles, and is somewhat related to, the restriction-site mapping, which deals with the equivalent problem of determining the absolute location of a fragment within a cosmid clone. Here also one uses digestion data from restriction enzymes but the focus is on finding the absolute location of a fragment on a clone. Stefik (1978) used a branch and bound technique with rules to exhaustively eliminate wrong answers from the digest fragment data. This approach is sensitive to error in the data and is computationally intensive. Pearson (1982) exhaustively generated permutations of the single-

digest data to compute the error between the generated double-digest and the actual (experimental) double-digest data. This approach is faster but it is limited to small number of restriction sites also. Krawczak (1988) developed a divide and conquer technique that groups the fragments into compatible clusters and then determine the order of the fragments within each cluster. This approach can process a greater number of restriction sites. Platt and Dix (1993) used Genetic Algorithms (GAs) for restriction-site mapping using double digest data. In their work they did not consider operators suited for multi-modal search spaces and mating which preserve adjacency information.

Other techniques are available to sequence larger DNA regions. Branscomb et. al. (1990) developed a greedy algorithm to order the most probable clone sequence using overlap probabilities between the clones. The algorithm works well when a large amount of overlap between the clones exists and the fragment data has small errors. This approach is prone to getting stuck in local minima and does not use all the available data gathered at great expense. Techniques using larger clones are also being tried to order, orient, and connect the islands in the original DNA (Olson et. al., 1986; Waterman and Griggs, 1986; Stallings et. al., 1990; Fickett and Cinkosky, 1993).

Cuticchia et. al. (1992) constructed maps using simulated annealing techniques. In their work clones are ordered according to a measure of

similarity between them given by the presence or absence of specific sequences. A signature is assigned to each clone and the algorithm uses it to minimize the error between the actual length of the contig and the given length by the hypothetical clone ordering. Matching signatures are used to order the clones. In their work they only considered the relationship between consecutive clones.

Recently, Parsons et. al. (1993, 1995) applied the edge recombination crossover (Whitley et. al., 1989) in conjunction with several specialized operators and solved a 10 Kbp sequencing problem consisting of 177 fragments with no manual intervention. Previously, we too had experimented, although not reported, with a variety of permutation-based crossover operators, such as those one finds in Traveling Sales Person type problems. In that context we also found and reported that the edge recombination operator outperformed the permutation-based operators by a wide margin (Cedeño and Vemuri, 1993, 1995). In this chapter, we now describe a genetic algorithm based on the MNC GA that uses single-digest restriction data on a set of overlapping clones to find multiple solutions for the clone sequences. We use genetic operators suited for multi-modal function optimization (Cedeño and Vemuri, 1992) to determine solutions. A mating operator based on genetic edge recombination was used to preserve adjacency information and improve convergence toward multiple solutions at

the same time. Results are given for two data sets of overlapping clones from human chromosome 19.

Before delving into the details of the application of MNC algorithm to the DNA problem, it is useful to pause and justify the need for multi-modal optimization to solve this problem. We are aware of the fact that molecular biologists, unlike engineers, cannot use “sub-optimal” solutions. As any geneticist knows, one missing or misplaced base-pair could lead to the synthesis of an entirely different protein. Ideally, what a molecular biologist wants is the exact sequence, which should correspond in our mathematical formulation to the global optimum. But the search for this global optimum is intimately tied up with the selection of the optimization criterion, say the fitness function in GAs or the optimization criterion such as an energy function in classical optimization. As long as this criterion is fabricated by our methods, we will never be able to zero in on the “correct” solution using mathematical methods. The best we can do is to ease the burden on the biologist by showing her (him) possible avenues for further experimental investigation. We believe that this is the place the sub optimization plays a role and that is the reason we were searching for the k-best sub optima.

5.4 Problem Representation

It is worth repeating the cautionary word about the terminology used here. Hereafter, the word DNA is used synonymously with the word chromosome(s) found in a cell, whereas the word chromosome will be used to

describe the computational data structure used in genetic algorithms. This interesting situation arose because genetic algorithms are computational processes that imitate natural processes of selection and survival. Coincidentally, we are applying this metaphorical computational paradigm to solve a problem in genetics! So it is important to distinguish meanings of terms used in the problem domain from those used in the paradigm domain.

ALLELE NUMBER	CLONE ID	FRAGMENT SIZES (in thousands of base-pairs)
C0	5154	16.55, 4.4, 1.68, 1.07, 4.81, 8.5
C1	7442	0.79, 0.79, 2.6, 4.35, 8.24, 2.7, 6.9, 5.16
C2	21230	0.96, 1.68, 1.08, 4.77, 8.47, 1.44, 2.37, 6.29, 0.62
C3	8131	0.92, 3.73, 19.8, 4.43, 1.69, 1.25, 4.68, 5.63
C4	18993	0.96, 6.31, 5.48, 8.61, 7.29, 0.81, 0.81, 2.6, 4.36, 1.92
C5	5435	2.89, 8.24, 2.7, 6.9, 5.14, 5.14, 2.89, 1.54
C6	7255	1.04, 8.21, 2.69, 6.89, 5.12, 5.12, 2.88, 1.94, 2.42, 1.37, 3.33
C7	12282	4.52, 5.13, 5.13, 2.87, 1.94, 2.42, 1.39, 3.35, 5.41
C8	27714	6.69, 5.07, 5.41, 2.88, 1.92, 2.32, 1.4, 3.35, 5.46, 17.65, 1.0, 10.49, 0.58, 1.74
C9	10406	2.03, 1.43, 2.34, 6.28, 5.46, 8.58, 7.27

Figure 5.2: Cosmid clones with fragment data.

In this section some problem-dependent genetic operators are defined. First, the encoding of the chromosomes to describe clone sequences is examined. Second, the use of fragment sizes to define the fitness function is described. Third, the modified edge recombination mating operator is described. And last, the function that measures similarity between two clone sequences is described.

Before going into the details about the operators, it is important to show how the data for the problem is presented to the GA. Figure 5.2 shows

fragment sizes obtained from fingerprinting for a set of overlapping cosmid clones. For example, cosmid clone with the ID number 5154 which is also labeled as Allele Number C0, is known to be comprised of six fragments, containing 16550, 4400, 1680, 1070, 4810 and 8500 base-pairs, in that order. Also, cosmid clone with the ID number 8131 which is also labeled as Allele Number C3, is known to be comprised of eight fragments, containing 920, 3730, 19800, 4430, 1690, 1250, 4680 and 5630 base-pairs, in that order. By comparing these fragment sequences one can surmise that the third and fourth fragments of Allele C0 are probably the same as the fifth and sixth fragments of Allele C3 mainly because the fragment lengths are so nearly equal to each other. If this is true then Alleles C3 can be "aligned" below Allele C0 in such a manner that the fifth and sixth fragments of Allele C3 fall right below the third and fourth fragments of Allele C0 as shown in Figure 5.3. The matching of the fragment lengths is not perfect. Indeed, the mismatch at other positions is large. The goal of this problem is to maximize this type of matching while minimizing the number and degree of mismatches while keeping the total length of the assembly within reasonable limits. The data for this problem consist of the n cosmid clones with their fragment lengths and the tolerance measure e which is used to determine if two fragments are of the same size. That is, two fragments $F1$ and $F2$ are considered to be of the same size if $|F1 - F2| < e$.

Clone	Fragments
C3 8131	0.92 3.73 19.8 4.43 1.69 1.25 4.68 5.63
C0 5154	16.55 4.4 1.68 1.07 4.81 8.5
C2 21230	0.96 1.68 1.08 4.77 8.47 1.44 2.37 6.29 0.62

Figure 5.3: An example of fragment assembly.

5.4.1 Chromosome Encoding

The encoding for this problem is simple. Each allele in the chromosome has a label between 0 and $n - 1$ corresponding to one of the cosmid clones. No two alleles have the same label, and mating and mutation will preserve this constraint. In Figure 5.2, for example, an allele with the label C0 corresponds to the clone with ID 5154 and an allele with the label C9 corresponds to clone ID 10406. The clone sequence (5154, 21230, 10406, 7255, 12282, 27714, 8131, 18993, 7442, 5435), for example, is represented by the chromosome (0 2 9 6 7 8 3 4 1 5). The initial population is generated by picking, at random, values between 0 and $n - 1$ without replacement.

5.4.2 Fitness Function

To calculate the fitness of an individual, the number of fragment matches between all consecutive clones and the error between the fragments is considered. The fragment sizes are represented using integer numbers (by multiplying the number given in Figure 5.2 by 100), to accelerate computation of the fitness function. Prior to the execution of the GA, two matrices are calculated. One matrix, the match matrix M shown on the left side of Figure 5.4, contains the number of fragments that match between two clones C_i and C_j , within an error tolerance ϵ . The other matrix, the error

matrix E shown on the right side of Figure 5.4, contains the total error between the clones being matched. The error between two clones is given by the sum of the errors between all fragments that matched. For example, between clone No. 8131 (C3) and clone No. 5154 (C0) there are two pairs of fragments that match within the specified tolerance of $\epsilon = 10$. The lengths of these fragments are 169 and 168 for one pair and 443 and 440 for the second pair. Thus a 2 appears in (row 2, column 4) of the Match Matrix M. The total error between both pairs of fragments is $(169-168) + (443-440) = 4$, which is shown in (row 1, col. 4) of the Error Matrix, E.

Number of matches between clones											Total error in the matches										
CLONE	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	
C0	6	1	4	2	1	0	1	0	2	1	0	5	8	4	4	0	3	0	13	8	
C1	1	8	0	1	4	4	4	1	1	0	5	0	0	8	5	2	9	3	9	0	
C2	4	0	9	3	2	1	3	2	5	3	8	0	0	14	2	10	16	10	23	5	
C3	2	1	3	8	2	0	0	1	2	0	4	8	14	0	11	0	0	9	13	0	
C4	1	4	2	2	10	1	3	2	3	4	4	5	2	11	0	10	19	9	6	10	
C5	0	4	1	0	1	8	6	3	2	0	0	2	10	0	10	0	10	4	8	0	
C6	1	4	3	0	3	6	11	7	7	3	3	9	16	0	19	10	0	7	26	23	
C7	0	1	2	1	2	3	7	9	7	4	0	3	10	9	9	4	7	0	20	26	
C8	2	1	5	2	3	2	7	7	14	3	13	9	23	13	6	8	26	20	0	5	
C9	1	0	3	0	4	0	3	4	3	7	8	0	5	0	10	0	23	26	5	0	

Figure 5.4: The M matrix on the left whose entries are the number of fragment matches and the E matrix on the right whose entries are the total errors between any two clones with $\epsilon = 10$.

Our goal is to arrange the cosmid clones as shown in Figure 5.1 so that the lengths of the overlapping fragments match with each other as closely as possible. The necessary matching information is already gathered in the matrix M and the degree of accumulated mismatch per clone is gathered in the matrix E. However, we believe that this information alone is not sufficient to establish which two clones are “adjacent” to each other in the

arrangement shown in Figure 5.1. For example, consider how clone C0 (i.e., allele No. 5154) matches with other clones. Inspection of the match matrix M indicates that the degree of match between clone C0 and clone C2 (or, equivalently, allele No. 21230) is 4 matches. Also, fragments in clone C0 match with fragments in clone C3 as well as C8, each with 2 matches. By interpreting this to mean that C2 should be placed nearer to C0 than C3 or C8, we are ignoring information contained in the C0-C3 matches and C0-C8 matches. One more example suffices to make the point. Clone C3 should be placed closer to C2 because they match with each other the maximum number of times, namely 3, although C3 matches with three other clones, each with only 2 matches. This phenomena makes us to think that using only the number of matches between clones is not sufficient to establish the partial order between clones when they possess the same match count. We believe that part of this problem is due to false matches, between fragments of similar sizes, that may occur by chance. We tried to overcome this problem by incorporating the total error in the matches, shown in matrix E, in order to enable our GA to discriminate further between clones. Using the same example, notice that clone C3 has less total error when matched with C0 than clone C8 and therefore indicates that C0 is adjacent to C3. The following equation for fitness captures the essence of the method described so far.

$$fitness = \sum_{i=0}^{n-1} \sum_{\substack{j=-1 \\ j \neq 0}}^1 \frac{M[a_i, a_{i+j}]}{Count[a_{i+j}]} \left(1 - \frac{E[a_i, a_{i+j}]}{M[a_i, a_{i+j}] \cdot \varepsilon} \right)$$

Here a_i refers to the cosmid clone placed in the i th position of the chromosome, a_{i+j} refers to the clone to the right or left (if any) of the i th position. The first term, $M[a_i, a_{i+j}] / Count[a_{i+j}]$, give us a normalized count for the degree of match. The term $E[a_i, a_{i+j}] / M[a_i, a_{i+j}] \cdot \varepsilon$ refers to the normalized error per fragment. When this normalized error reaches unity, it means that the total error is so large that any apparent matches are worthless. With this interpretation, the second term of the equation essentially tells us the degree of confidence we can place on the normalized matches we are counting in the first term. In the above equation, n refers to the number of alleles in the chromosome which is equal to the number of clones.

By defining the fitness function as above, we are assigning a higher fitness to those clone pairs that match a higher percentage of their regions. For example, Allele 0 with 6 fragments has two matches each with Alleles 3 and 8, each having 8 and 14 fragments respectively. Since $2/8$ represents a higher percentage than $2/14$, we designed a fitness function that prefers a configuration that places Allele 3 closer to Allele 0 than Allele 8. This is achieved by dividing the number of matches by the number of fragments in the clone.

Before settling on the fitness function described above, others were considered. For example, fitness functions that just counted the number of matches between clones with no regard to normalization failed to produce the correct answer. A fitness function that just counted the number of matches and then subtracted the total error in those matches also failed to give satisfactory results. It is possible that other fitness functions may give results that are even better than what are reported here. In the future, we plan to include the number of matches as well as errors among groups of three clones or more as components of the fitness function and study its effect on performance.

5.4.3 Mating and Mutation Operators

The mating operator used in this method is based on a slight modification to the genetic edge recombination operator that was applied successfully to solve the TSP (Traveling Sales Person) problem. As in the TSP problem, the important information here is the adjacency of the alleles, although the order the alleles appear in the chromosome can be derived from the adjacency information. The idea is to recombine the links (pairs of clones) between two parents such that common links are inherited by the offspring. This operator is implemented in two steps as shown in Figure 5.5. First, those links (or traits) that are common to both the parents are identified and passed on to the offspring and the links occupy the same absolute positions in the offspring chromosome. In the example shown in Figure 5.5, the relevant link-

pairs are 7-8, 8-1, and 5-0 in the first parent and 1-8, 8-7 and 5-0 in the second parent. Notice that these links are passed on to the two offspring undisturbed. Second, those alleles that are not passed to the offspring (indicated by dashes, in Figure 5.5) are randomly assigned to the available positions while observing the constraint that no link label is repeated.

Parent 1	Common links (traits)	Offspring 1
(6 7 8 1 2 3 9 4 5 0)	(- 7 8 1 - - - - 5 0)	(3 7 8 1 4 6 2 9 5 0)
Parent 2	Common links (traits)	Offspring 2
(1 8 7 9 5 0 2 6 3 4)	(1 8 7 - 5 0 - - - -)	(1 8 7 3 5 0 6 2 4 9)

Figure 5.5: Modified genetic edge recombination for clone sequencing.

The differences between this operator and the original edge recombination operator are in the number of offspring generated and in the assignment of alleles not transferred from the parent. We generated two offspring instead of one because the location of the links in the clone sequence is important to our problem. In TSP the chromosome is circular, thus the location did not matter. We allow both parents to pass the location of the links to their offspring. To assign the other alleles we select them at random from those clones not passed by their parents. In the original operator the links are assigned from those present in any of the two parents. Alleles with fewer links are assigned first to prevent from running out of links for a given allele.

In the mating operator used here there is excessive exploration of the search space primarily due to the random filling of the unassigned slots, in the second step, while creating the offspring chromosomes. Part of this

exploration difficulty is alleviated by the fact that mates are selected using crowding selection and therefore they have common features between them. Exploration is therefore localized to a smaller region within the entire search space. On the other hand, by allowing unassigned clones to be chosen at random, we are allowing links to re-appear that might not have done so using mutation alone. Other crossover operators based on clone positions alone did not perform as well in this problem, we only give the results for the modified edge recombination operator.

Mutation is applied on an individual basis. After an offspring is generated it is mutated if the outcome from the flip of a biased coin is true. When this happens, a link from the offspring is selected at random and all alleles from that link to the last position of the chromosome are reversed. For example, the offspring (1 8 7 3 5 0 6 2 4 9) after mutation can result in (1 8 7 9 4 2 6 0 5 3) if the link between allele 7 and 3 is selected to mutate. This mutation operator is known as inversion (Holland, 1975).

The mating and mutation operators are compatible with each other in the sense that they both operate on links. The building blocks of this problem are based on the links between clones in the sequence. The GA operates on these links so that the most useful ones are passed from generation to generation.

5.4.4 Similarity Function

The similarity function counts the number of dissimilar links between two individuals. Using the parents from Figure 5.5 once again as an example,

notice that there are six dissimilar links, corresponding to the five alleles not assigned to the offspring. For concreteness, these six dissimilar links in Parent 1 chromosome are 6-7, 1-2, 2-3, 3-9, 9-4 and 4-5 and for Parent 2 are 7-9, 9-5, 0-2, 2-6, 6-3 and 3-4. This metric measures the proximity between two clone sequences by counting the different links they have and not the position of the alleles. For example, the sequences (0 1 2 3 4 5 6 7 8 9) and (9 8 7 6 5 4 3 2 1 0) have a distance of zero since all the links are the same. This metric captures the essential aspect of the problem since both solutions are equivalent in our problem.

5.5 Results

The results presented in this section were obtained on a SGI IRIS 4D computer under IRIX O.S. running the GA application written in C. The parameters for the GA are the following:

Population size:	100
Mutation probability:	0.06
Crossover probability:	1.00
Crowding selection group size (C_s):	20
Crowding factor group size (s):	10
Crowding factor (C_f):	5
Maximum number of generations to execute:	100
Tolerance ϵ :	10

These parameters were selected after various trials. In each trial, different values for each of the six parameters were tried, varying one at a time while holding the others constant. A population size of 100 was found satisfactory. Other sizes (50, 150, 200, 250, and 300) were tried. Higher sizes

did not provide new information about the problem, although they produced, on average, the solution in less number of generations. Lower sizes in some cases did not converge to the best solutions seen before in the allowed number of generations. The population size of 100 is a compromise between speed of convergence and execution time.

Mutation was set at 0.06, therefore an average of 6 individuals were mutated every generation. This low value of mutation works well in this problem. It seems to allow individuals to escape local optima without causing any major disruption on the elements of a niche. In the MNC method diversity is maintained implicitly when the population converges to different optima, reducing the need for higher mutation values. On the other hand the mating probability was set to a high value of 1.0. As described before the MNC method is basically a steady state GA and highly fit individuals have a high chance of surviving for many generations. Also, crowding selection exploits the similarity between individuals in the population by allowing localize solutions to pass common traits between them.

The group size for crowding selection was set at 20. This high value emphasizes the importance of similarity between parent and mate for this problem. Then the mating operator will likely generate an offspring within the same region as their parents. The size for the crowding factor groups was set at 10. This allowed competition between multiple optima to occur while maintaining multiple solutions. Good results were obtained with other C_c and

s values (5, 10, 15, 20) also. The crowding factor was set at 5. This value allowed the MNC method to eliminate low fitness individuals from the niches more rapidly. These parameter values allowed a diverse population to co-exist during the number of generations allowed and did not restrict competition between individuals from different niches. The tolerance value ϵ was set to 10 to minimize false matches due to chance. Higher values of ϵ increased the false matches more than true matches and therefore more possible clone sequences were found.

The MNC method took an average of 50 seconds for each run. Some of the best sequences obtained for two different data sets of overlapping clones are shown in Figure 5.6. These results were obtained from 10 different runs. The figure shows the actual sequence for the data sets and the clone sequences (with their fitness) obtained by the algorithm.

Data for set 1 is shown in Figure 5.2 and data for set 2 is shown in Section 5.7. There was nothing really special about these data sets except that they were in regions containing various gene(s) of interest to some of our collaborators. They had already mapped them manually, so we could judge the answers our GA technique provided. Finally, and perhaps most importantly, the data sets were large enough not to be toy problems. One of the data sets contained two vertical "columns" in the final map that had fragments of the same size. The existence of independent columns of the

same size is; of course, one of the things that makes this problem so tough, and interesting. Many of the runs using data set 2 generated multiple solutions with relatively close fitness values.

For data set 1 the MNC method was able to find the actual clone sequence. The best sequence as described by the fitness function did indeed match the solution. The best sequence was found in all the runs. From the other sequences found, the fitness values of the next best is 15 less than the actual sequence. Similar gaps exist between all the sequences shown in Figure 5.6 for data set 1. From the solutions we can see that the other sequences are a single mutation from the actual sequence.

Data Set 1 actual sequence and its fitness:

(8131 5154 21230 10406 18993 7442 5435 7255 12282 27714) 764

The 3 Best sequences found by the GA and their corresponding fitness:

(8131 5154 21230 10406 18993 7442 5435 7255 12282 27714) 764

(8131 5154 21230 27714 12282 7255 5435 7442 18993 10406) 749

(8131 5154 21230 10406 27714 12282 7255 5435 7442 18993) 744

Data Set 2 actual sequence and its fitness:

(12595 6722 26999 29626 29064 18301 19811 29035 17755 28828 20235) 750

The 12 Best sequences found by the GA and their corresponding fitness:

(12595 26999 6722 29626 29064 18301 28828 20235 17755 29035 19811) 757

(20235 28828 17755 29035 19811 18301 29064 29626 6722 26999 12595) 757

(12595 26999 6722 29626 29064 18301 20235 28828 17755 29035 19811) 756

(28828 20235 17755 29035 19811 18301 29064 29626 6722 26999 12595) 755

(12595 26999 6722 29626 29064 18301 28828 20235 17755 19811 29035) 754

(12595 26999 6722 29626 29064 18301 20235 28828 17755 19811 29035) 753

(12595 6722 26999 29626 29064 18301 19811 29035 17755 28828 20235) 750

(20235 28828 17755 29035 19811 18301 29064 29626 12595 6722 26999) 750

(19811 29035 17755 20235 28828 18301 29064 29626 12595 6722 26999) 750

(19811 29035 17755 20235 28828 18301 29626 29064 26999 6722 12595) 750

(19811 29035 17755 28828 20235 18301 29064 29626 26999 6722 12595) 749

(19811 29035 17755 28828 20235 18301 29064 29626 12595 6722 26999) 749

Figure 5.6: Clone sequences obtained by GA and actual sequence.

Data set 2, shown in Section 5.7, presented a more challenging problem for the MNC method. In this case the best sequence found only had clones 6722 (C7) and 26999 (C2) transposed from the actual sequence. The fitness for the actual sequence is 750, which is the sixth best score when compared with all solutions found. The actual sequence was obtained in 8 of the 10 the runs. As shown in Figure 5.6, different solutions with equal fitness values were found and maintained in the runs. Two different solutions with a distance of 2 (two different links) were found with fitness equal to 757. Four others were found with fitness equal to 750 (including the actual sequence). Another observation is that there is a difference of 8 or less in the fitness between all the sequences found. Some of the sequences are mutations of others, but there is more diversity when compared with the solutions for data set 1. The biologists later confirmed via independent techniques (i.e., hybridization probing) that the clones in data set 2 were indeed from two distinct regions in the DNA. That is, the sequence (6722 26999 12595 29626 29064 18301) and (19811 29035 17755 28828 20235) belong to separate maps in the DNA. The best solutions found by the MNC method kept both sequences apart.

Figure 5.7: Population average fitness (left) and best fitness (right) for different methods.

To see how the MNC method compares with more traditional GAs we applied the same problem (i.e., using data set 2) to the SGA and SSGA with crowding. For both the SGA and SSGA the crossover probability was set at 0.6. The crowding factor was set to 10 for the SSGA. All other parameters are the same as in the MNC method. We observed poor results by the SGA using these parameters alone. To improve the performance of the SGA the 10 best individuals in the population (known as generation gap (De Jong, 1975) were transferred from the current generation to the next. Figure 5.7 shows the population's average fitness (on the left) and the fitness of the best solution (on the right) for the MNC, SGA, and SSGA methods averaged over 10 runs.

The MNC method outperformed both the SGA and the SSGA. The population converged faster and at the same time kept multiple solutions throughout the run. Use of a generation gap allowed the SGA to retain the good individuals found in previous runs, but did not improve convergence

towards multiple solutions. The SSGA did not find the best sequence in any of the runs. The SGA found one of the two best sequences in 2 of the runs. As mentioned before the MNC found the two best sequences and others in 8 of the runs. The ability of the MNC method to exploit similarity and increased competition among localized solutions proved to be successful in this problem.

5.6 Discussion and Comments

Two points deserve further discussion. First, data containing clones with fewer than five fragments were normally sequenced erroneously by the GA. This is due to the lack of opportunity for sufficient fragment matches. Also data pertaining to corner fragments (i. e., fragments lying near the cosmid clone boundaries) is generally more prone to errors. Consequently corner segments will not match well with a high probability with their counterparts in the preceding and succeeding clones. For clones with less than five fragments, this means that on the average, at least half of the data is not useful and in some cases leads to more false matches. Clones with less than five fragments were usually placed first or last in the clone sequence by the GA.

Second, when a large number of overlaps existed between 3 or 4 clones, the GA experienced difficulty deciding the correct sequence. An example of this behavior was observed with data set 2. This phenomenon, we believe, is happening because the fitness function is only looking for matches between

the clones to the left and right without accounting for the fragments which are common to three or more clones. Permutations of these clones generally had similar fitness values. An improved fitness measure is needed to account for fragment matches between three or more clones.

Although the test data used for this work is small and could be solved using other techniques (like simulated annealing, branch and bound, etc.), it allows us to examine the performance of the MNC GA in other areas aside from function optimization. In the future we will try larger data sets to show the suitability of the approach. Additionally, we plan to extend the fitness function to incorporate overlapping among more clones and order the fragments within each clone.

Overall the MNC worked well with the data presented to it. Using the correct set of genetic operators was very important to find a MNC model that will find good solutions to the problem. Using a multi-modal approach was very useful for this problem also since it prevented premature convergence and at the same time explored the search space in a more efficient manner. Defining the operators for mating, mutation, fitness, and similarity measure to work with adjacency information between the clones rather than clone positions gave the MNC method the correct set of tools to converge towards the most probable solutions. More information must be incorporated into the fitness evaluation to distinguish even further between the best clone sequences and other similar ones.

The results on multi-modal optimization and DNA fragment assembly shows the ability of the MNC to converge to multiple solutions, maintain stable sub-populations, and succeed in complex search spaces. Exploiting similarity during selection and replacement allows a diverse population to coexist while competition for slots in the population evolve naturally. Convergence towards the best solution(s) is not affected with the increased diversity. The MNC method uses the diversity present in the population to guide the population towards multiple optima in the search space.

5.7 Fragment Data and Matrices for Data Set 2

Fragment data for set 2 of overlapping cosmid clones:

CLONE ID	FRAGMENTS (k base-pairs)
29064	15.42, 3.46, 1.50, 9.12, 4.30
19811	3.13, 7.89, 7.89, 3.02, 4.35, 14.31, 2.65, 0.64
26999	4.64, 19.69, 1.10, 1.48, 2.82, 0.77, 9.61
29626	3.46, 1.50, 9.14, 6.47, 13.48,
17755	1.26, 2.62, 6.32, 2.73, 3.54, 7.88, 7.88, 3.02, 4.35, 1.74
12595	1.48, 2.83, 0.76, 9.68, 12.75, 1.48, 6.06
20235	8.01, 12.56, 2.62, 6.32, 2.74, 3.54, 5.71
6722	2.84, 19.72, 1.16, 1.49, 2.84, 0.77, 9.69, 9.20
28828	12.45, 2.61, 6.27, 2.72, 3.52, 7.89, 3.89
18301	2.53, 13.99, 2.64, 16.88, 3.44
29035	2.45, 2.74, 3.53, 7.82, 7.82, 3.02, 4.35, 7.44

Match matrix for data set 2:

CLONE	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C0 :	5	1	1	3	2	1	1	2	1	1	2
C1 :	1	8	0	0	5	0	1	0	2	1	5
C2 :	1	0	7	1	1	4	1	6	1	0	1
C3 :	3	0	1	5	1	1	1	2	1	1	1
C4 :	2	5	1	1	10	1	4	1	5	2	6
C5 :	1	0	4	1	1	7	1	4	0	0	1
C6 :	1	1	1	1	4	1	7	1	4	2	2
C7 :	2	0	6	2	1	4	1	8	0	0	1
C8 :	1	2	1	1	5	0	4	0	7	2	3
C9 :	1	1	0	1	2	0	2	0	2	5	3
C10:	2	5	1	1	6	1	2	1	3	3	8

Error Matrix for data set 2:

CLONE	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C0 :	0	5	2	2	13	2	8	9	6	2	12
C1 :	5	0	0	0	5	0	3	0	4	1	23
C2 :	2	0	0	2	9	9	8	20	10	0	8
C3 :	2	0	2	0	8	2	8	7	6	2	7
C4 :	13	5	9	8	0	10	1	10	10	12	14
C5 :	2	0	9	2	10	0	9	4	0	0	9
C6 :	8	3	8	8	1	9	0	10	10	12	1
C7 :	9	0	20	7	10	4	10	0	0	0	10
C8 :	6	4	10	6	10	0	10	0	0	11	10
C9 :	2	1	0	2	12	0	12	0	11	0	27
C10:	12	23	8	7	14	9	1	10	10	27	0

6. Application of the MNC GA to Water Quality Control

In this chapter we apply the multi niche crowding genetic algorithm to a multi-objective optimization problem arising in water quality control. The MNC GA is capable of efficiently locating all the peaks of a multi modal function. By associating these peaks with the utility accrued from different sets of decision variables it is possible to extend the use of the MNC GA to multi-objective decision making problems. This concept is applied to address the problems arising in the context of remediation of a contaminated aquifer. The multi niche crowding genetic algorithm is used to decide the optimal location of pumping wells. The aquifer dynamics are simulated by repeatedly solving the partial differential equations describing the flow of water using the SUTRA code. Output of this simulation is used to determine the fitness of the individuals in the genetic algorithm.

6.1 Introduction

Many decision making problems of practical interest are characterized by multiple objectives. The decision maker is forced to strike a balance among many competing objectives, not all of which are quantifiable. The problem of simultaneously optimizing several objective functions has been posed and studied by many (Reid and Vemuri, 1971; Vemuri, 1974; de Neufville, 1990). What would be useful is an interactive method that presents to the analyst

the consequences of choosing one of many competing alternatives. In this paper we intend to demonstrate the suitability of a new version of a genetic algorithm to address this problem. The crux of the method lies in identifying the objectives with the peaks of a function and solving for the locations and heights of these peaks using the MNC genetic algorithm.

Specifically, we study the remediation of groundwater in contaminated aquifers. The problem is to determine the optimum placement of pumping (and recharge) wells and optimum pumping (and recharge) schedules in order to achieve a set of objectives. Posing the remediation problem as a multi-objective optimization problem and solving that using genetic algorithms has been done (Rogers and Dowla, 1994; Rogers, Dowla and Johnson, 1994; Johnson and Rogers, 1994; Horn and Nafpliotis, 1993). Here we address the same type of problem, but use the multi-niche crowding GA (Cedeño and Vemuri, 1992, 1994; Vemuri and Cedeño, 1995). The problem is posed as one of locating the peaks of a function where the height of each peak in a $(n+1)$ -dimensional space corresponds to the value of a suitably defined utility function and the n coordinates of each peak correspond to the values of the n decision variables associated with that utility function.

6.2 The Simulation Environment

To lend reality and at the same time to keep the computational burden within practical bounds we studied a hypothetical groundwater basin beneath a one square mile region. Any resemblance this may have to a real

aquifer is purely coincidental. Many simplifying assumptions about the flow of water are also made. The main objective here is to test the MNC genetic algorithm in an environment that is as realistic as possible.

6.2.1 The Aquifer

Consider a 1 mile by 1 mile square region occupied by, say, a petrochemical facility. This site, by its very nature is likely to be polluted with spills of oil or other hydrocarbons. Such pollutants seep into the ground and contaminate water in the aquifer, located, say, 90 to 180 feet below the ground. The primary concern is the containment of the pollutant from seeping outside the boundaries of the facility and polluting the drinking water supplies of the adjacent city which uses groundwater for its municipal water supply. It is essential that contaminants from the site do not diffuse and pollute the city water supply.

In one operational scenario, polluted water is first pumped out, treated, and the treated water is used to recharge the aquifer using injection wells. As pumping and injection are expensive, the total cost (fixed and running) is often used as an objective function in the optimization procedure. The total volume of water pumped could also serve as an objective function. Another possible objective would be the amount of contaminant actually removed. How does one go about designing the pumping network and pumping schedule to meet several objectives of this kind?

In our simulation we assumed that three "pump, treat and recharge" facilities already exist on the western edge of the site and one on the eastern edge. Three more "pump, treat and recharge" facilities are planned. The present task is to find an optimum location for these three new facilities. On the average each facility is assumed to cost \$2.5 million and is capable of cleaning 70 gallons of water per minute by reducing the concentration from 550 ppb to negligible quantities. Thus the scenario described is as realistic as we could make it. With proper modifications, this can be applied to a variety of other contexts.

6.2.2 The SUTRA Model

Flow of water in underground formations can be described by nonlinear partial differential equations (PDEs). Nonlinearities enter the equations because the transmissibility and storage coefficient, the two most important parameters, are dependent upon the elevation of the water table, the dependent variable in the PDEs (Vemuri and Karplus, 1969). The aquifers are non homogeneous and the flow could be two-phase. In the remediation context, solute mass transport also becomes important. The success of the management policies depend to a large extent on our ability to predict the movement of water and the contaminants over time. To meet these potential needs, we decided to use the flow and transport simulation code, SUTRA, of the U. S. Geological Survey (Voss, 1984). SUTRA (Saturated, Unsaturated TRAnsport) is a 2-D hybrid finite-element/ finite-difference model aimed at

solving the governing partial differential equations for confined areal groundwater flow and areal solute transport.

Boundary conditions imposed will depend upon the specific problem. In this hypothetical example, we assumed no-flow fault zones to the northeast and southeast. Flux boundaries were assumed along the eastern and western edges of the site. A hydraulic conductivity of about 10 feet/day is assumed.

The mesh used has a total of 2436 nodes and 2385 elements, which covers the areal extent of the aquifer, which is much bigger than the one square mile region (see Figure 6.1). Out of this, we have chosen a subregion of size 20×21 nodes, covering an area of 5200 ft x 4950 ft. Each node in this subregion is considered as a potential pump location. The distance between nodes is 260 feet, and there are 420 possible pumping locations.

Strictly speaking, the decision on where to locate the pumps should be governed by considerations such as the concentration of pollutants, pollution gradients, feasibility of drilling, and so on. In this study, we were forced to include another consideration, namely the computational burden. SUTRA is computationally expensive; it took about 6.5 minutes of elapsed time per fitness evaluation. Each fitness evaluation ran SUTRA for 10 time steps, where each step had a duration of one year. Any final resolution meant much more time and we simply could not muster the computational resources to do the job. It was felt that some gains in computational time in terms of fitness

evaluations can be made by restricting the potential pumping sites to a subregion of the total grid and by using larger time steps.

Figure 6.1 SUTRA nodes and elements in mesh

This knowledge of the flow field, obtained by solving the PDE's using SUTRA, constitutes an input to the genetic algorithm. The algorithm's goal is

to find sets of well locations in the 20 x 21 grid that best meets the objectives, subject to the constraint that no more than 10 wells are allowed. The algorithm outputs the values of the objective functions for a range of values of the decision variables. Each of the objective functions constitutes the “modes” of the multi-modal function over which the genetic algorithm conducts its search.

6.3 Interfacing with the Multi Niche Crowding Genetic Algorithm

In this section we describe the genetic operators for the aquifer problem. Specifically, the problem being solved is the determination of the optimum location of no more than 10 wells, on a 20 x 21 grid, for a total of

$$\sum_{i=1}^{10} \binom{420}{i}$$

possible configurations, so that three objectives are met. The first objective is to minimize the remediation cost which includes the cost for facilities, piping, water treatment, and day to day operation. Cost minimization can be achieved by setting up a budget and trying to find solutions which stays within or close to the budget. This approach allows the GA to treat solutions that are within the budget equally, and alleviates convergence pressure to single-well solutions. The second objective is to maximize the amount of contaminant removed from the aquifer. This is a straightforward measure and is obtained from the output of SUTRA in kilograms. The third objective

is to prevent unsafe levels of contaminant from leaving the site. The goal is to minimize the concentration of contaminant leaving the site as much as possible. This measure is also obtained from the output of SUTRA in parts per billion (ppb).

6.3.1 Chromosome Representation

A solution to the problem is represented by a dynamic-length chromosome, i.e., a chromosome whose length can change. Each chromosome is comprised of information on up to 10 wells, where the well locations appear in ascending order. The well locations are identified by node numbers in the 20 x 21 grid delineating the site in the SUTRA mesh. No duplicate well locations are allowed in the chromosome. For example, the chromosome (41 107 224 635 972) denotes a five-well pumping scenario with the wells located at the nodes 41, 107, 224, 635, and 972. Only 420 of the total 2436 nodes in the mesh are valid well locations; wells are not feasible at other nodes because those nodes fall outside the 1 square mile site. This representation is equivalent to a binary string with a bit for each possible well location in the grid. In the future we intend to include attributed nodes in the chromosome. An attributed node includes not only the node number but also other relevant information about that node such as pumping rates, pollution removed, etc. This modification should render our representation more practical. This attributed representation can accommodate constraints on well locations due to the presence of buildings or other structures.

6.3.2 Recombination

During the recombination step, the genetic operators of crossover (or, mating) and mutation are applied. After two individuals in the population are selected using crowding selection the mating operator is applied to generate one offspring (or, child). The mating operator used here is equivalent to uniform crossover (Syswerda, 1989). A child chromosome is formed by first passing to it the genes that are common to both the parents and then selecting with a 0.5 probability the unique (i. e. non common) genes in both the parents. For example, let individual A = (41 107 224 635 972) and its mate M = (22 107 635 700). The child resulting from mating A and M will have wells at nodes 107 and 635 because these well locations are common in both the parents. The other well locations in the child will come from some subset of the remaining five non-common nodes, {22, 41, 224, 700, 972}. These well locations are selected with probability 0.5. A resulting offspring could be (22 107 635 972).

Three mutation operators are considered: "add well", "delete well", and "move well". All three operators are applied independently with the same mutation probability. The offspring undergoes mutation if the flip of a biased coin is true. During an "add well" mutation a well in one of the 420 nodes in the grid is selected at random and added to the offspring while taking care to see that no duplicate nodes are allowed. The "delete well" mutation removes a well from the offspring at random, while taking care to see that only

offspring containing more than one well can undergo “delete well” mutation. Finally, the “move well” mutation changes the location of a well, selected at random, to one of the four nodes around it. The new location is either at the top, bottom, left, or right of the previous location in the grid. The “add well” and “move well” mutations allow new well locations to appear that might not be in the initial population. The “delete well” mutation prevents the individuals from growing without control. The “move well” mutation allows some fine tuning.

6.3.3 Replacement

The replacement step in the MNC (worst among most similar) selects, for replacement, the individual in the crowding factor group with the lowest fitness. In a multiobjective problem a single fitness value can only be obtained if the objectives are grouped into a single utility function with weighting values. This is not appropriate since a specific set of weight values drives the GA towards specific solutions. Here we determine the worst individual in the group by ranking the individuals for each objective and selecting the individual with the worst total ranking. This is a type of minmax strategy where an individual with a minimum fitness from among a group of individuals who are maximally similar to a child is selected for replacement by that child. This concept is pictorially shown in Figure 6.2. The individuals in the crowding factor group, indicated by A, B, C, and D, are ranked for each objective. Here a lower rank number indicates a better value

for the objective being considered. Same rank numbers are assigned to individuals with equal or similar objective values. The total ranking for each individual is then determined by the sum of all the ranking numbers. The individual with the highest total rank, in this example B, is replaced by the offspring in the population.

Using ranking as a way to determine the worst individual in a group allows good individuals for specific objectives to evolve as well as individuals that are average in two or more objectives.

Individual	Objective 1	Objective 2	Objective 3	Total Ranking
A	2	1	3	6
B	2	4	2	8
C	1	2	3	6
D	3	3	1	7

Figure 6.2: Use of ranking in Worst Among Most Similar Replacement.

6.3.4 Similarity Measure

Similarity between individuals is used, as a distance measure, during the selection and replacement steps in the MNC GA. It allows formation of species within the population naturally and maintains diversity throughout the run. For this problem, a simple measure based on the number of wells in each individual chromosome and the distance between the closest well match is used as a similarity measure. For example, to calculate the distance between individual A = (a1 a2 a3 a4) and individual B = (b1 b2 b3 b4 b5 b6) with 4 and 6 wells respectively, the following three steps are implemented. First, the squared difference in the number of wells between the individuals

is calculated. In this example, $(4 - 6)^2 = 4$. Second, the individual with the lower number of wells is selected, in this case A. Finally, for each well in A, the *closest* well in B is determined, say (a1,b4), (a2,b1), (a3,b6), and (a4,b3). The *similarity* between individuals A and B is now given by the sum of squared difference and the distance between all pairs of wells. The following equation summarizes the distance formula for this example:

$$\text{similarity} = \text{sqrDiff} + \text{dist}(a1,b4) + \text{dist}(a2,b1) + \text{dist}(a3,b6) + \text{dist}(a4,b3).$$

This equation measures the similarity of two individuals - smaller the number more similar the individuals.

6.3.5 Fitness Measure

There is no explicit fitness function defined in this problem. The fitness of each individual is implicitly given by their objective values and their rank against other individuals. Survival of an individual from generation to generation is determined by competition among small groups of individuals during replacement. Specifically, an individual will survive for many generations if its total ranking, as described in Section 6.3.3, given by the objective values is lower than those individuals in the crowding factor group. For the aquifer problem we are using three objectives; cost, contaminant removed, and the ability to meet regulatory constraints.

Cost for a set of pumping realizations is calculated using the distance of each well to an hypothetical pump and treat facility located at the center of

the 20 x 21 grid. In the future we plan to include the location of this facility as part of the chromosome definition. The total distance is used to estimate the cost for piping, which is estimated as \$70 per ft (Rogers, 1992). The cost objective is to find a set of wells which stays close to an established budget. In our runs we used a budget of \$2.5 million dollars. The squared difference between the budget and the piping cost is the value being minimized.

The second objective is straightforward. The amount of contaminant removed is taken from the estimates obtained from SUTRA runs. This value is output in kilograms and is dependent on the well locations and the amount of water being pumped. In this implementation we are using a fixed pumping rate of 0.0223 cubic feet per second for all wells. In the future we plan to include the pumping rate as part of the chromosome. The goal in formulating this objective is to maximize the amount of contaminant removed.

The third objective is to meet the regulatory limits imposed. For this, a set of monitor wells are located on the west side of the site. The amount of contaminant in parts per billion (ppb) is estimated using SUTRA. We assumed that the total amount of contaminant reaching the monitor wells should be no more than 20 ppb. This value is given by SUTRA and the goal in this problem is to minimize the amount of contaminant as much as possible.

The values calculated for each objective are used by the MNC GA only during replacement step. Each objective value is used to determine the ranking of the individual for the particular objective. The sum of the rank values is then used to select the individual to be replaced by the offspring. Any number of objectives can be included in the problem. An objective which ranks an individual according to its similarity to members of the population can also be used to increase the diversity (Winston, 1992) in the population.

6.4 Results

The results described in this section were obtained on a Sparc 2 workstation running Sun OS 4.1.3. Following is the list of parameter values used for the MNC GA:

Population size:	25
Crossover probability:	1.0
Mutation probability:	0.1
Crowding selection size:	3
Crowding factor:	3
Crowding size:	3
Number of generations:	20 and 25

In addition to restricting the potential pumping sites to a subregion of the total grid, small population sizes and a small number of generations were selected primarily to reduce the computational burden. The crowding parameters were selected after trying some combinations in the interval [2-5]. From previous experience we had learned that values between 5% and 15% of the population size are reasonable for the crowding selection size and the crowding size parameters (Vemuri and Cedeño, 1994). Crossover is

always applied to mates, hence the crossover probability is set at 1.0. The mutation probability was set at 0.1 to allow on average 10% add, move, and delete mutations every generation.

The initial population was generated at random. For each individual the number of wells in the chromosome was chosen at random from the interval [1, 10]. Then, the well locations were selected at random from the 420 nodes in the 20 x 21 grid. In all runs the initial population never contained a solution meeting the regulatory limit, assumed to be 20.0 ppb.

6.4.1 Scenario 1: Two Objective Functions

At first, the MNC GA was implemented to solve a two objective optimization problem. The first objective is to restrict the amount of contaminant leaving the site to a presumed regulatory limit of 20 ppb. The second objective is to maximize the contaminant removed from the pumped water at the treatment facilities. Although it may appear at first sight that achieving the second objective automatically results in the attainment of the first objective, it is not always necessarily so. Contaminant transport in ground water basins is very slow. If the pumping wells are located at some distance from the site boundary, it is conceivable that the effects of pumping will reach a far off location (say, the boundary of the site) only after several years. This in turn would mean that the SUTRA simulation should be conducted for a long duration of time. This is one aspect of the computational

bottleneck that prompted several simplifying assumptions in the numerical simulations.

As the regulatory constraint is a dominating constraint, it was decided that only those solutions that satisfy this constraint are of interest. Table 6.1 shows a sample of these solutions. Notice the absence of generations 6, 8, 9, 10, 11, etc. In these generations acceptable solutions were found, but they didn't improved significantly over the solutions seen so far. Among the solutions not shown in the table, there are some solutions that did a very efficient job of removing the contaminant but did not meet the regulatory limit. This apparently anomalous behavior is most likely due to the above mentioned transport delays between the pumping well locations and monitor well locations.

Table 6.1: Sample solutions with two objectives

Gen.	No. of Wells	Contaminant leaving site (in ppb)	Contaminant Removed (in kg)
4	7	19.23	4.52
5	9	1.90	25.87
7	8	3.80	35.25
12	7	1.29	35.38
13	8	13.36	75.62
15	10	0.92	51.94
20	9	0.31	46.82

The acceptable solutions are then examined for their ability to remove the contaminant. The solutions are arranged in the order of generations in which they appeared for the first time. That is, in Generation 4 (first row) there appeared an individual (a solution), for the first time, that satisfied the 20.0 ppb regulatory limit, and that individual solution is characterized by seven pumping wells, and that seven-well pumping configuration managed to remove 4.52 kg of the contaminant, while the monitor wells recorded a 19.23 ppb concentration leaking out of the site under study.

Further examination of the table reveals that all solutions meeting the regulatory limit are characterized by the presence of seven or more pumping wells. Both the 7-well solutions, from Generations 4 and 12 are consistent with each other, although the solution appearing in the 12th generation is better. The difference between the 4th and 12th generation solutions comes presumably because of the actual placement of the seven wells in question. Similar behavior is observed with respect to the two 8-well solutions and the two 9-well solutions shown in the table.

Figure 6.3 (shown at the end of this chapter) shows the solutions represented by the fourth and fifth rows in Table 6.1. The locations of the wells in the map, denoted by symbols for hand pumps, are only approximate. These solutions have four wells in common although they have different objective values. In general small changes in well locations may cause large changes in the objective values. The MNC GA alleviates this behavior by

promoting recombination between similar solutions. In both solutions the wells are mostly in the southern part of the site where the pollutant concentration is greater. Multiple wells in close proximity and wells in the north-east segment of the map can be considered useless and may be removed from the final configuration.

6.4.2 Scenario 2: Three Objective Functions

In this scenario we included the cost objective in the fitness function. The cost objective is to minimize the difference between the cost of installation and an established budget. Some positive changes were observed in the quality of solutions. The cost objective added some pressure towards smaller number of wells. Configurations with fewer wells were more common in later generations than before. At the same time the best solutions removed a greater amount of contaminant and had lower amounts of contaminant leaving the site. In general the MNC GA was able to improve the three objectives in subsequent generations.

Table 6.2 shows a set of solutions for a run of the MNC GA. Only improvements from previous generations are shown. Only solutions meeting the regulatory limit and having a higher amount of contaminant removed from previous generations are shown. The last column ranks the solutions according to their cost. Lower rank values indicate lower cost solution. As expected, improvements in one objective had a negative impact in other objectives during the initial generations. The competition by solutions having

above average ranking in particular objectives was visible again and as a result offspring with good ranking in most objectives appeared in later generations.

Table 6.2: Sample solutions with three objectives

Gen.	No. of Wells	Contaminant leaving site (in ppb)	Contaminant Removed (in kg)	Cost Rank
10	7	6.30	36.71	6
10	6	7.79	39.62	5
18	7	11.15	43.37	4
19	7	13.27	59.02	3
20	7	2.44	64.68	1
21	7	1.21	81.98	2
23	9	6.59	98.89	7

Figure 6.4 (shown at the end of this chapter) shows the placement of the wells for solutions shown on the fifth and sixth rows of Table 6.2. Both solutions have seven wells with five of them common between them. Both solutions have very low values for the amount of contaminant leaving the site. They also have high rankings in cost and contaminant removed. Prior to what we had seen using only two objectives, improvements in more than one objective was more common in later generations. Better results may be obtained if we increase the number of generations executed and the population size.

6.5 Discussion and Comments

Several issues were addressed in this chapter. First, the scope of this chapter is limited to the issue of demonstrating the ability of the GA, namely the MNC GA, in finding an acceptable solution to the multi-objective optimization problem arising in a practical context. Performance of MNC GA relative to alternate approaches is being investigated. Some preliminary results obtained when MNC GA is applied to artificially created, multi-modal test functions have already been reported [Cedeño and Vemuri, 95].

This chapter clearly demonstrated the suitability of MNC GA in meeting this restricted scope. It is shown that the MNC GA indeed has the ability to maintain different solutions satisfying multiple (and perhaps, conflicting objectives). The stratagem of replacing a single utility function to measure fitness by a rank score assigned to various objectives appears to have allowed non-inferior solutions (i.e., solutions with favorable rankings in all objectives) to evolve. Given the limits imposed by us on the population size and the maximum number of generations in the evolutionary process, the quality of solutions obtained appear to be satisfactory. The policy of maintaining diversity throughout the search (enforced by MNC GA), clearly paid off in the form of better solutions in later generations.

Second, there are several issues that remain to be addressed. For example, could an equally acceptable or even a better result be obtained by some simple search strategy? Not likely, for two reasons. First, the problem is

computationally complex. On a 20x21 grid there are 420 potential well positions. According to the formulation used in the chapter, each pumping/recharge well configuration can contain anywhere from one to ten wells. This gives us

$$\sum_{i=1}^{10} \binom{420}{i}$$

combinations of well positions to explore. This is a huge number. Secondly, most of the classical optimization methods locate the peaks of a multi-modal function one at a time, in a sequential manner. The merit of the MNC GA is in finding all the peaks simultaneously in one sweep. Perhaps this one sweep takes more effort than n successive sweeps made by a classical method, where n is the number of peaks. This question can be addressed by making an estimate of the computational effort of both approaches using the “big O” notation.

What about the influence of the SUTRA simulation code on the performance of MNC GA? The SUTRA code only solves the partial differential equation simulating the flow of water and the resulting solution is used only in evaluating the fitness function used in the MNC GA. It has a major influence on the time taken to get a solution, but it should have no influence on the performance of the GA itself.

Finally, more research is also needed on the GA front. What are the benefits of bigger population size and more generations? The issue here is practical. To adequately address this issue, we need another way of obtaining the values for the SUTRA dependent objectives to minimize the running time. We will be looking into other ways to calculate contaminant removed and regulatory limits that although may not be as precise will give us the ranking of the solutions for each objective. Additionally we can include heuristics in the mutation operators to add, move, or delete wells that will improve the overall ranking of the individual. Finally the chromosome can be extended to include other well parameters like pumping rate to evolve solutions with the complete pumping schedule. Adding well location constraints can easily be handle in the genetic operators also.

Figure 6.3: Well configurations for fourth (top) and fifth (bottom) solutions in Table 1.

Figure 6.4: Well configurations for the fifth (top) and sixth (bottom) solutions in Table 2.

7. Application of the Generational MNC GA to solve the File Design Problem

In this chapter we apply a parallelized version of the generational version of the MNC GA to solve an important problem arising in Computer Science. Consider a distributed system with multiple nodes that contain all the data for a company-wide database comprising millions of records. In order to use the resources equitably and efficiently, the data in the database must be organized so that queries for database records are balanced among all nodes. That is, all nodes must handle about the same number of queries on average. Additionally, the information in the database must be organized so that a query can be satisfied by accessing only a small number of nodes. Minimizing the number of nodes accessed on a single query reduces greatly the communication overhead on the network. This requirement asserts that records with the same attributes should be placed together, if possible, at one location. These two criteria must be balanced to obtain an optimal database configuration. This problem is known as the File Design Problem (FDP).

The File Design Problem is known to be a NP-hard. The goal is to find an assignment of database records to files that minimizes the average number of files examined over all single attribute queries. In this chapter we describe a solution to the File Design Problem using the Multi-Niche Crowding Genetic Algorithm written in SISAL (Streams and Iterations in a Single Assignment

Language, McGraw 1985), a functional language that takes advantage of parallel architectures. Using the portability and architecture independence inherent in SISAL a parallel model of the MNC GA is defined that provides increased performance without losing the convergence properties of the algorithm.

Additionally we introduce the use of heuristics in the crossover and mutation operators used during the mating step. These operators prove to be essential in finding optimal solutions to the FDP. The use of heuristics and the ability of the MNC GA to search for multiple conflicting database configurations provides us a promising hybrid approach for solving complex combinatorial problems. Results with various test cases are shown. Performance of the algorithm is shown for different computer platforms.

7.1 Introduction

The File Design Problem is an NP-hard problem; that is, the number of possible solutions increases exponentially as the problem size increases linearly. It arises in the context of database design for a distributed system.

The problem is defined as follows. Given a set of N records, each characterized by a single attribute A that takes h different values $\{a_1, a_2, \dots, a_h\}$. There are n_i records corresponding to attribute a_i , i.e., $n_1 + n_2 + \dots + n_h = N$. The assumption is made that queries for records of any given attribute are

equally likely. We also have K files of size b such that $K * b = N$. The constants K, b, n_i, N , and h are all positive integers.

The problem is to find an assignment of the N records to the K files such that the average number of files (ANF) accessed over all possible single-attribute queries is minimized. In other words, an assignment of the records to the files must be found such that (on average) queries for the records with the same attribute can be satisfied by reading from as few files as possible.

Table 7.1: Possible configurations for 12 records in 2 files of size 6. The attributes values are $\{B, C, F, V\}$ and have $\{2, 7, 1, 2\}$ records respectively.

File 1	File 2	fex(B)	fex(C)	fex(F)	fex(V)	ANF
C C C C C C	C B B V V F	1	2	1	1	1.25
C C C B V V	C C C C B F	2	2	1	1	1.50
C C C C B V	C C C B V F	2	2	1	2	1.75
C C C B B F	C C C C V V	1	2	1	1	1.25

For example, consider a database with $N = 12$ employee records characterized by their last name (here the last name refers to attribute A). Moreover, assume that all records have a last name in the set $A = \{\text{Blattner, Cedeño, Feo, Vemuri}\}$ with $n = \{2, 7, 1, 2\}$ records respectively. Here we have a total of $h = 4$ possible last names (attribute values). These records will be placed in a database consisting of $K = 2$ files of size $b = 6$. The problem now is to save the employee records in the database such that queries for records with a given last name (attribute value) access the minimum number of files

(on average). Using the first letter of each last name the first two columns in Table 7.1 shows some sample configurations for this example.

The *ANF* for a configuration is given by the formula

$$\frac{\sum_{i=1}^h f_{ex}(a_i)}{h},$$

where the function called $f_{ex}(a_i)$ returns the number of files that must be accessed to retrieve all the records with attribute a_i . From Table 7.1, the second configuration has a value of 2 for $f_{ex}(B)$ since both files contain a record with attribute value B. The first and last configurations in the table are examples of optimal solutions for this problem. Even though the *ANF* values are the same, in some contexts the last solution is better because it has a more balanced configuration. If requests for the attributes are distributed uniformly, file 1 and file 2 will be accessed 25% and 100% respectively in the first solution, were as the last solution will be accesses 50% and 75% respectively. This idea is incorporated when evaluating solutions generated by the MNC GA.

GAs have been successfully applied to the Traveling Salesman Problem (Whitley et. al., 1989), Scheduling (Syswerda and Palmucci, 1991), and the Bin Packing Problem (Falkenauer and Delchambre, 1992) to mention a few. In some cases better results were obtained when the mating operator was designed to capture the essential information in the problem. The mating

operator for the File Design Problem is based on "first fit" and "best fit" heuristics. Such heuristics group records with the same attribute together. The multimodal search space in the problem is explored in many directions by using selection and replacement operators (Cedeño and Vemuri, 1992) that encourages mating and replacement between solutions from the same extrema.

In this work we apply the generational version of the MNC GA. The intent is two fold. First, we want to show that the generational version of the MNC GA exhibits the same properties as its steady state counterpart. Second, we want to show the advantages of the parallel version of the generational MNC GA, that is straight forward to implement on a parallel architecture. SISAL was selected as the language for the parallel implementation because it is portable and easy to learn. The application can be ported to multiple platforms, including SGIs, Crays, and SUNs. Performance can then be evaluated using different number of processors. Additionally, SISAL is a deterministic functional language which guarantees the same solutions on different platforms.

There are basically three parallel GA models (Gordon et. al., 1992) exhibiting different degrees of parallelism; fine grain, distributed, and direct. In a fine grain model (Davidor, 1991; Gorges-Schleuter, 1989; and Spiessens & Manderick, 1991), each solution in the population is mapped to a processor with genetic operators applied between nearest neighbors. In a distributed

model (Mühlenbein et. al., 1991; Tanese, 1989), processors are assigned subpopulations, which converge locally and exchange genetic material among them at fixed interval. Direct models (Grefenstette, 1981), exploit the parallelism inherent in the GA operators and the GA structure while having the same properties of a sequential GA. Our SISAL implementation of the MNC GA follows the direct model while having the localized convergence exhibited in the other models.

The parallelism inherent in the generational MNC GA and in the operators is easily exploited. Performance is enhanced while maintaining the necessary computation for solving the problem. The best solution was found in all test cases with a speedup of at least 2.2 with four workers.

This chapter is organized as follows. Section 7.2 gives an overview of the MNC GA model used for this problem. Section 7.3 describes the genetic operators specific for the File Design Problem. Section 7.4 defines the experimental setup. Section 7.5 contain the results and conclusions. Finally Section 7.6 contains the comments and conclusions.

7.2 The Genetic Algorithm Model

The SISAL version of the MNC GA was designed to capture the parallelism in the model while maintaining the search for multiple solutions. In this model, multimodality is exploited by encouraging mating and replacement between solutions from the same peak. Improved performance is

obtained by creating the offspring in parallel. The offspring are then inserted into the population sequentially to preserve replacement between members of the same peak.

The solutions in the initial population are created in parallel by assigning records to files at random. There are K files with b slots each for a total of N slots. The slots are uniquely numbered with a value between 1 and N . Each record is then assigned a slot number corresponding to a unique position in a file. The constraints of the problem are easily maintained without the need for counters for each of the files. The *fitness*, a measure of “goodness” of a solution, is then calculated for each member of the population.

The algorithm is executed for a fixed number of generations. Each generation consists of creating all the offspring and inserting them into the population. Three steps are involved to create two offspring: select the parents, apply the mating operator to the parents, and calculate the fitness to the offspring. Mutation is applied by the mating operator as part of the mating process. Each offspring is inserted sequentially in the population by selecting an existing solution to die.

To create the offspring each solution in the population is selected as a parent. This allows every individual in the population to mate at least once in every generation. All the mates for the parent are selected in parallel using crowding selection. A group of C_s (crowding selection size) solutions are

chosen at random from the population and the most similar individual is selected as the mate. After selection, mating produces two offspring and their fitness are computed. The number of offspring created can be up to two times the number of solutions in the population. We create a total of n (population size) mating pairs and each pair produces 2 offspring with probability χ (crossover probability). All offspring are created in parallel with a given crossover and mutation probability.

The offspring are inserted one at a time in the population using the worst among most similar (WAMS) replacement policy. Replacement is applied sequentially. After an offspring is inserted in the population it immediately becomes a candidate for replacement and must compete with the other solutions in the population to survive. Some offspring are indeed replaced in the same generation before getting a chance to reproduce. As in selection, the replacement operator is biased toward solutions within the same extrema. Convergence is improved by the replacement operator which eliminates solutions with lower fitness.

7.3 Genetic Operators

In this section the encoding and genetic operators for the File Design Problem are described. They were designed to preserve the constraints of the problem and make them easier to implement using SISAL arrays.

7.3.1 Gene Encoding

A *gene* represents a valid solution to the problem. It consists of an array of N *alleles* corresponding to each of the records in the problem. Each allele may assume a value between 0 and $K - 1$ inclusive, indicating the file containing the record. A valid encoding is a N digit number in base K where all digits appear exactly b times. An example is shown in Figure 7.1 for the records defined in Table 7.1. To make clear that the chromosomes are not binary we select in this case $K = 3$ files of size $b = 4$.

record number:	1	2	3	4	5	6	7	8	9	10	11	12
record attribute:	B	B	C	C	C	C	C	C	C	F	V	V
gene 1:	0	0	0	0	1	1	1	1	2	2	2	2
gene 2:	0	0	1	1	2	2	2	0	0	1	1	2

Figure 7.1: Encoding for the file design problem.

7.3.2 Similarity Metric

Similarity between two solutions is measured from the number of records assigned to the same file. An example is shown in Figure 7.2 using the data from Table 7.1. As in the previous section we have 3 files of size 4. Each digit indicates the file where a record is located. In this example we have a similarity value of 8, that is 8 records have been assigned to the same file.

Records:	B	B	C	C	C	C	C	C	C	F	V	V
Configuration 1:	1	0	2	2	0	1	0	1	1	0	2	2
Configuration 2:	2	0	1	2	0	1	0	1	2	0	1	2
Similar assignments:	1	2	3	4	5	6	7	8				

Figure 7.2: Using similarity to select a mate during crowding selection.

7.3.3 Mating Operator

The crossover operator for the File Design Problem creates two offspring and was designed with two goals in mind. First, the characteristics expressed in both parents will be expressed in the offspring, thus preserving the schemata in both solutions. Second, fitness should be improved when combining two similar solutions. "Best fit" and "first fit" heuristics (described later) are used for this. Incorporating these features in the mating operator improves convergence of solutions from the same extrema. When two solutions from different extrema mate, offspring from other extrema can be created. This way the operator is not restricted to small areas in the search space.

The first step in the mating operator is to transfer similar characteristics from the parents to the offspring. This is done by transferring the records assigned to the same file in both parents to the same file in the offspring. Those records not assigned are counted for each attribute and sorted in decreasing order. One offspring is created using a best fit method based on the contents of files. In this approach unassigned records are located into files where records with the same attribute reside. The main idea is to group files with the same attribute in the same file as much as possible. The second offspring is created using a first fit method based on the empty space in the files. Here the unassigned records will be located where file space is available

for records with the same attribute. Using the configuration in Figure 7.1 an example is shown in Figure 7.3.

Given the parents in Figure 7.3, the offspring inherits only four alleles; 3 records with attribute C and 1 record with attribute B. Using the best fit method the other 4 records with attribute C are assigned to file 2 and file 0 because those files contain records with the same attribute. Using the first fit method the 4 records are assigned to file 1 because that file is the most empty and all records can be placed together. If not all records fit in one file then the remaining records are placed in the next file having the most available space. The next attribute having the highest number of unassigned records is selected and its records are assigned in a similar manner.

Offspring inherits similar alleles from parents:

Record Attribute: B B C C C C C C C F V V

Parent 1: 0 0 1 2 2 2 0 1 0 1 2 1

Parent 2: 0 1 0 1 2 2 1 2 0 0 1 2

Offspring: 0 - - - 2 2 - - 0 - - -

Unassigned records by attribute: B:1, C:4, F:1, V:2

Assignment of records in sorted order to both offspring:

	Offspring 1												Offspring 2											
	Best Fit Method												First Fit Method											
C:4	0	-	2	2	2	2	0	0	0	-	-	-	0	-	1	1	2	2	1	1	0	-	-	-
V:2	0	-	2	2	2	2	0	0	0	-	1	1	0	-	1	1	2	2	1	1	0	-	2	2
B:1	0	1	2	2	2	2	0	0	0	-	1	1	0	0	1	1	2	2	1	1	0	-	2	2
F:1	0	1	2	2	2	2	0	0	0	1	1	1	0	0	1	1	2	2	1	1	0	0	2	2

Figure 7.3: Mating operator for the File Design Problem

Mutation is applied with a fixed probability for each allele. When an allele is selected for mutation another position in the gene is selected at random and the two values are interchanged. Such mutations may introduce a new configuration in succeeding generations.

7.3.4 Fitness Function

The fitness function captures three important characteristics of an optimal solution: low *ANF*, records with the same attribute are grouped together, and records with the same attribute are spread equally among the minimum number of files needed to store them. The last two points are captured in a grouping term (*GT*) and balancing term (*BT*) respectively. The two terms are contradictory in the sense that *BT* wants to group records together, while *GT* wants to spread records equally across files. These three terms are added together, with different weight values, to get the fitness of a solution. The *BT* value is given a higher weight over the *GT* value because it promotes lower *ANF* values in the solution.

The *ANF* value is given by the formula:

$$ANF = \sum_{i=1}^h fex(a_i) / h,$$

where $fex(a_i)$ returns the number of files containing attribute a_i . From this formula we can compute an upper and lower bound to the *ANF* term. The lower bound represents a configuration where the records for all attributes

are assigned to the least number of files needed to contain them. The upper bound can be calculated from the configuration containing the records for all attributes spread among the maximum number of files possible. The lower and upper bound are called *min_anf* and *max_anf* respectively and are:

$$\min_anf = \sum_{i=1}^h \lceil n_i/b \rceil / h \leq ANF \leq \max_anf = \sum_{i=1}^h \min(n_i, K) / h.$$

Recall that n_i denotes the number of records with attribute i , h denotes the number of attributes, b denotes the size of the files, and K denotes the number of files.

To compute the *GT* value we need to know how the records of a given attribute are spread in the files. Since we want as many records as possible of the same attribute grouped together, we came up with an equation that looks at the ratio of records with the same attribute in each file. The *GT* value is computed by adding the normalized number of records squared for each attribute in every file. The higher the number of records of the same attribute in a file the higher the *GT* value. The formula for the *GT* value is:

$$GT = \sum_{i=1}^h \sum_{j=1}^K (attr(a_i, j)/n_i)^2,$$

where $attr(a_i, j)$ returns the number of records of attribute a_i in file j .

On the other hand the *BT* value wants to spread the records with the same attributes equally among the minimum number of files needed to fit the

records. The *BT* value is then computed by adding the absolute value of the difference between the number of records for each attribute and a balance configuration for the attribute. Only files containing records for the given attribute are included in the summation. The formula for this term:

$$BT = \sum_{i=1}^h \sum_{j=1}^K \left| attr(a_i, j) - n_i / \lceil n_i / b \rceil \right| \text{ when } attr(a_i, j) \neq 0.$$

Here $\lceil n_i / b \rceil$ returns the number of files needed to store the records of attribute a_i . Values of *BT* closest to zero represent more balanced configurations.

The three terms *ANF*, *BT*, and *GT* are used to define the fitness value for a solution. Since higher positive values are used to indicate a better solution the terms are normalized to return values between 0.0 and 1.0. A percentage of each term is then added to form the final fitness value as indicated by the following formula:

$$\text{fitness} = 0.70 * \frac{GT}{h} + 0.25 * \frac{\max_anf - ANF}{\max_anf - \min_anf} + 0.05 * \frac{1.0}{1.0 + BT}.$$

The fitness value for any solution is a number between 0.0 and 1.0. Solutions where the fitness value is 1.0 represent configurations where the *min_anf* value is achievable and all the records for any attribute can fit in a single file. Having the property of fitting records with the same attribute in

one file eliminates the conflict between *BT* and *GT* while obtaining a maximum value of h for *GT*.

7.4 Experimental Data

To evaluate the behavior of the algorithm six test cases, having different properties, were created. Some of the test cases contain solutions achieving the *min_anf* lower bound. In other test cases we have attributes with the number of records exceeding the file size (therefore the fitness < 1.0 and a *min_anf* configuration does not exist). In all test cases multiple attributes per file were mixed to create the different configurations. For all configurations 100 records were used. Table 7.2 summarizes all configurations created.

Table 7.2: Configuration for all test cases

Case Num.	Num. of Files	File Size	Number of Attributes	Number of Records per Attribute $n_1, n_2, n_3, n_4, n_5, \dots, n_k$	<i>min_anf</i> exist
1	5	20	10	7, 2, 3, 1, 5, 17, 18, 13, 15, 19	Yes
2	10	10	10	7, 2, 3, 1, 5, 17, 18, 13, 15, 19	Yes
3	5	20	10	7, 4, 3, 8, 6, 11, 18, 15, 10, 18	No
4	10	10	10	7, 4, 3, 8, 6, 11, 18, 15, 10, 18	No
5	5	20	21	7, 4, 3, 8, 6, 1, 8, 5, 10, 8, 1, 2, 4, 9, 5, 1, 6, 2, 3, 3, 4	Yes
6	5	20	15	7, 4, 9, 7, 7, 4, 9, 5, 9, 7, 9, 5, 6, 7, 5	No

To evaluate the performance of the implementation three different platforms were used: the SGI Iris 4D, Cray Y-MP, and Cray C90. The

execution time from one to four workers was collected for the algorithm using case 1 in Table 7.2. The MNC GA parameters used for each run are:

Population size:	100
Number of generations:	50
Mating probability:	0.95
Mutation probability:	0.01
C_s for selection:	4
C_r for replacement:	3
s for replacement:	5

These parameters were chosen after a trial and error period. They represent a good set of choices for the test data shown in Table 7.2.

7.5 Results

The generational MNC GA was very successful for the test data in Table 7.2. For all test cases, multiple optimal solutions were found and retained for many generations. In four of the six test cases at least one optimal solution was found prior to generation 6. More generations were needed for the test cases 3 and 4. These test cases have the property that the *min_anf* is not achievable and there are attribute values where the number of records is higher than the file size. In those cases, the solutions were competing between themselves for a very small improvement in fitness.

Table 7.3 shows solutions for all test cases and the generation number on which they were obtained. Each solution is represented by the file number to which each record is assign. The records with the same attribute value are

separated by commas to verify how many files are used to save them. For example, the first solution has the records for attribute 1 assigned to file 4, records with attribute 2 assigned to file 1, and so on.

Table 7.3: Solution found by the MNC GA to test cases and the number of generations needed

Case Number	Generations Needed	Best Solution Found
1	3	444444, 11, 333, 0, 22222, 333333333333333333, 111111111111111111, 44444444444444, 2222222222222222, 00000000000000000000
2	4	999999, 77, 111, 6, 88888, 33333333113311111, 222222222277777777, 5555555555999, 444444444488888, 00000000006666666666
3	25	222222, 0000, 222, 444444444, 333333, 444444444444, 000000000100001000, 3333334333333333, 2222222222, 111111111111111111
4	15	888888, 9999, 888, 55555555, 999999, 22222222227, 0000000000666666666, 111111111155667, 3333333333, 444444444477777777
5	5	000000, 3333, 222, 33333333, 222222, 0, 00000000, 11111, 22222222222, 444444444, 2, 44, 4444, 111111111, 33333, 4, 111111, 44, 333, 444, 0000
6	5	000000, 2222, 2222222222, 44444444, 2222222, 3333, 3333333333, 44444, 000040000, 3333333, 111111111, 11111, 111111, 4444444, 00000

By examining test case 4 more closely we can observe that the optimal configuration required the records for the eighth attribute (111111111155667) assigned to 4 different files. All other attribute values were assigned to 1 or 2 files only. In the same run other configurations were found where the *ANF* was the same and all the records for the eighth attribute were stored in 2 or 3 files. In such cases other attribute values were assigned to 3 or 4 files.

In general, the use of heuristics improved the convergence of the MNC GA. We tried other crossover operators, but they required many more generations to achieve similar results. At the same time the MNC GA did not allow the population to converge prematurely to a local optima. Mixing heuristics with the GA allowed us to obtain results which are as good as using the heuristics alone.

We also observed the increase in speed that can be obtained using a parallel implementation of the MNC GA. A speedup between 2.2 and 2.9 was achieved with four processors in the three different platforms. Figure 7.4 summarizes the performance from one to four processors in the different platforms.

<u>Platform</u>	<u>1 Worker</u>	<u>2 Workers</u>	<u>3 Workers</u>	<u>4 Workers</u>
Y-MP C90	12.9799	8.6748	6.4425	5.4930
Y-MP	18.6106	10.4642	8.9153	6.3648
SGI Iris	25.8900	16.5300	13.4200	11.9000

Figure 7.4: MNC GA execution time in seconds for 50 generations.

The best speedup was obtained for the Cray Y-MP platform and the worst speedup on the SGI Iris platform. The speed of the Cray is not much faster than that of the SGI when we take into account that the SGI does not have vector calculations and worse cache locality. Better speedup times can be obtained for more complex (in terms of evaluation time) fitness functions. This is because selection and mating are done in parallel whereas replacement is done sequentially. Since the fitness of a new offspring is

calculated at the end of the mating step a more complex fitness function will benefit from the parallelism.

7.6 Summary

The results obtained with the parallel version of the generational MNC GA model are encouraging. As its steady state counterpart diversity was maintained during the run and the last generation contained multiple optima. Exploiting the multimodality inherent in the File Design Problem resulted in a more balanced search over the entire space.

Creating genetic operators that use heuristics enhanced the convergence of the algorithm while at the same time allowing multiple solutions to coexist. In this work we developed a model from the problem's point of view. We enhanced the MNC GA with problem specific operators to provide a better way to search for optimal configurations. The convergence to optimal solutions was achieved in all cases while improving the performance using SISAL.

A better speedup can be achieved in more complex fitness functions or by introducing a parallel version of the WAMS replacement operator. The parallel version must retain the important properties of WAMS. Competition among solutions within the same peak is encouraged while allowing competition among the multiple peaks as well.

8. Conclusions and Suggestions for Future Research

In this dissertation we have shown the ability of the MNC GA to locate and maintain multiple niches in a multimodal landscape. Unlike other GAs that suffer from premature convergence the MNC GA is able to use implicitly the contour of the fitness landscape to evolve separate subpopulations on different niches. These subpopulations evolve naturally maintaining a healthy diversity throughout the search. It is this stability between exploration and exploitation that allows different subpopulations to coexist in the fitness landscape. This behavior is one of the most important properties of the algorithm. It allows the approach to be applied successfully to complex dynamic or static optimization problems containing multiple optima in a simple manner.

The MNC GA converges to multiple optima without the need of gradient information or shape of the niches in the fitness landscape. The genetic operators, crowding selection and worst among most similar replacement, promote competition among the members of the same niche to allow members of each niche to evolve to their full potential. At the same time it allows competition among members of different niches allowing subpopulations to grow to a size relative to its niche.

An important area for future research is the incorporation of the parameters of the genetic operators like crossover and mutation probability, crowding selection size (C_s), crowding group size (s), and crowding factor group size (C_g), in the algorithm so that they can also be adjusted dynamically with the given landscape. Additionally it would be helpful to find a way to measure the similarity among individuals without the need for the user to define a distance function for each problem. These improvements will allow the method to be applied more easily. Also it would be useful to study the effect of using fitness together with similarity for mate selection during crowding selection on the performance of the MNC GA. The same applies to WAMS replacement where we can study the effect on performance when offspring are inserted greedily into the population. That is, offspring are only inserted in the population if it has a higher fitness than the individual selected for replacement.

We also demonstrated that the MNC GA outperforms other niching methods when applied to multimodal function optimization. In these comparison tests we only counted the number of function evaluations in the various methods. In this context, one can evaluate the performance of the different methods using other metrics and more complex test cases before establishing the suitability of each approach. Other potentially useful metrics include; diversity count, niche stability, number of evaluation to reach all optima, niche average fitness, niche maximum fitness, and so on.

The applicability of the MNC GA to dynamic environments was also evident from the results presented in Section 3.10. The results provided a glimpse of the ability of the MNC GA to locate emerging optima at different locations of the landscape without impacting the overall behavior of the population. The algorithm was able to maintain other optima while searching for new emerging optima. Moreover, the subpopulation size in each niche adjusted dynamically as the niche's structure change. This is an area where the properties of the MNC GA can be fully exploited. Since the algorithm has no need for re-seeding the population nor for a high mutation rate a run can be maintained for many generations while some parameters of the fitness function are adjusted dynamically. Old results from other runs can also be used to minimize number of fitness function evaluations for problems with time consuming fitness evaluations. Other tests using practical fitness functions are needed to fully evaluate the benefits of the algorithm on dynamic landscapes.

During the analysis of the MNC GA we presented various results. First, we defined the exact probability of individuals in the population under crowding selection. These results were used to examine the expected similarity rank of the mate during crowding selection. It allowed us to observe the effect on mating for different values of the crowding selection size. Increasing the crowding selection size increases mating among members of the same niche. Higher size promotes a higher subpopulation size in the

niches but decreases diversity in the population because members from different niches do not participate in mating as often.

Second, we defined the probability of creating an offspring under interval crossover and bit mutation. This result allowed us to observe the locality induced by this crossover operator, making it suitable for niching methods that promote mating among members of the same niche. A limitation of this operator is that it assumes that a linear transformation exists between chromosome values and the real values they represent. This limits the applicability of the operator to many problems.

Third, we defined the replacement probability under WAMS replacement. Specifically we defined the probability that an individual in the population is replaced by an offspring. This allowed us to analyze the effect of different values of the crowding group size and the crowding factor group size. Increasing the crowding group size increases replacement among members of the same niche. This means that competition is concentrated mostly among members of the same niche. Increasing the crowding factor group size increases the chances of highly fit individuals to survive and that low fit individuals are replaced. Higher values will also decrease the diversity in the population since low fit individuals generally do not belong to any of the formed niches. When combining higher values for both group sizes the MNC GA is able to locate and maintain subpopulations at many more optima, including those with lower average fitness.

Finally, we took the results from crowding selection, mating, and WAMS replacement to define the Markov chain transition probability matrix for the MNC GA. We used the transition matrix to show the properties of the MNC GA on a sample problem from multimodal function optimization. From the transition states we can observe that the MNC GA assigns higher probability to those states where the population converges to multiple optima. In the future we want to expand these results to prove other properties of the MNC GA. We would like to find out the necessary conditions that will insure success of the algorithm. We also want to define metrics that will allow us to select the parameters of the genetic operators for any given problem.

The results on the application of the MNC GA to DNA fragment assembly show the ability of the method to converge to multiple solutions, maintain stable sub-populations, and succeed in complex search spaces. Using a multimodal approach was very useful for this problem also since it prevented premature convergence and at the same time explored the search space in a more efficient manner. Exploiting similarity during selection and replacement allowed a diverse population to coexist while competition for slots in the population evolve naturally. Defining the operators for mating, mutation, fitness, and similarity measure to work with adjacency information between the clones rather than clone positions gave the MNC method the correct set of tools to converge towards the most probable solutions. When compared with the canonical or steady state GA, the MNC GA maintained

more solutions and converged to the best solution(s) faster. More information must be incorporated into the fitness evaluation to distinguish even further between the best clone sequences and other similar ones. In the future we want to extend this work by improving the fitness function and applying it to the construction of the fragment sequence within each clone as well. We also want to compare our approach to other more traditional methods used for DNA mapping.

The applicability of the MNC GA to multi-objective optimization was shown in the context of aquifer management. It was shown that the MNC GA indeed has the ability to maintain different solutions satisfying multiple, and perhaps, conflicting objectives. The stratagem, during the replacement step, of replacing a single utility function to measure fitness by a rank score assigned to various objectives appears to have allowed non-inferior solutions (i. e., solutions with favorable rankings in all objectives) to evolve. The policy of maintaining diversity throughout the search (enforced by MNC GA), clearly paid off in the form of better solutions in later generations. Another area of future research is to compare our approach to other methods for multi-objective optimization and expand the method to incorporate other parameters in the genome that affect the quality of a configuration.

Finally, in the last chapter we applied the generational version of the MNC GA to the so called file design problem. Like the steady state MNC GA, the algorithm located and maintained multiple solutions throughout the run

while maintaining diversity in the population. Creating genetic operators that use heuristics enhanced the convergence of the algorithm while at the same time allowing multiple solutions to coexist. These operators enhanced the ability of the MNC GA to search for optimal configurations. The convergence to optimal solutions was achieved in all cases while improving the speed with a parallel version developed using SISAL. In the future we want to investigate in more detail the use of heuristics for genetic operators.

9. Bibliography

- Beasley, D., Bull, D. R., and Martin, R. R. (1993). A sequential technique for multimodal function optimization, *Evolutionary Computation*, volume 1, number 1, MIT Press, MA.
- Belew, R. K. and Booker, L. B., eds., (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, California.
- Branscomb, E., Slezak, T., Pae, R., Galas, D., Carrano, A. V., & Waterman, M. (1990). Optimizing restriction fragment fingerprinting methods for ordering large genomic libraries. *Genomics*, 8, 351-366.
- Brindle, A. (1981). Genetic algorithms for function optimization. Technical Report 81-2, University of Alberta, Canada, January 1981.
- Carrano, A. V., Lamerdin, J., Ashworth, L. K., Watkins, B., Branscomb, E., Slezak, T., Raff, M., De Jong, P. J., Keith, D., McBride, L., Meister, S., & Kronick, M. (1989). A high-resolution, fluorescence-based, semiautomated method for DNA fingerprinting. *Genomics*, 4, 129-136.
- Cavicchio, D. J. (1970). *Adaptive search using simulated evolution*, Ph.D. thesis, University of Michigan, Ann Arbor, MI.
- Cedeño, W. (1992). Genetic algorithms in SISAL to solve the file design problem, *Proceedings of the Second SISAL User's Conference*, San Diego CA.
- Cedeño, W. (1993). DNA restriction fragment map assembly with genetic algorithms. In J. R. Koza (ed.) *Genetics Algorithms at Stanford*. Stanford, CA:Stanford Bookstore, 40-49.
- Cedeño, W. and Vemuri, V. (1992). Dynamic multimodal function optimization using genetic algorithms, *Proceedings of the XVIII Latin-American Informatics Conference*, Las Palmas de Gran Canaria, Spain: University of Las Palmas, 292-301.
- Cedeño, W. and Vemuri, V. (1993). An investigation of DNA mapping with genetic algorithms: preliminary results, *Proceedings of the Fifth Workshop on Neural Networks*, SPIE 2204, 133-140.
- Cedeño, W., Vemuri, V., and Slezak, T. (1995). Multi-Niche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments. *Evolutionary Computation*, 2:4, 321-345.
- Cedeño, W. and Vemuri, V. (1995). Genetic algorithms in aquifer management. To appear in *MicroComputer Applications*, October 1995, Academic Press.
- Cobb, H. J. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In S. Forrest (ed.) *Proceedings of the Fifth*

- International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers San Mateo, California, 523-530.
- Cuticchia, A. J., Arnold, J., & Timberlake, W. E. (1992). The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. *Genetics* 132, 591-601.
- Dasgupta, D. & McGregor, D. R. (1992). Non-stationary function optimization using the structured genetic algorithm. In R. Manner and B. Manderick (eds.), *Parallel Problem Solving from Nature*, 2. Amsterdam: North Holland, 145-154.
- Davidor, Y. (1991). A naturally occurring niche & species phenomenon: The model and first results, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, California.
- Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold New York, NY.
- Davis, T. and Principe, J. C. (1991). A simulated annealing like convergence theory for the simple genetic algorithm. In R. K. Belew and L. B. Booker, (Eds.), *Proceedings of the fourth Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 174-181.
- De Jong, K. A. (1975). An analysis of the behaviour of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan. *Dissertation Abstracts International* 36(0), 5140B. (University Microfilms No. 76-9381).
- De Neufville, R. (1990). *Applied Systems Analysis: Engineering Planning and Technology Management*. McGraw-Hill Publishing Co.
- Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization, In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 42-50.
- Elo, S. (1994). A parallel genetic algorithm on the CM-2 for multi-modal optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, 818-822.
- Eshelman, L. J., ed. (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G.J.E. Rawlings (ed.), *Foundations of Genetic Algorithms - 1*. San Mateo, CA: Morgan Kaufmann Publishers.
- Eshelman, L. J., ed. (1995). *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers.

- Falkenauer, E. and Delchambre, A. (1992). A genetic algorithm for bin packing and line balancing. *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*.
- Fickett, J. W., & Cinkosky, M. J. (1993). A genetic algorithm for assembling chromosome physical maps. In C. R. Cantor & R. Robbins, (Eds.), *Proceedings of the Second International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*. River Edge, NJ: World Scientific, 272-285.
- Forrest, S., ed., (1993). *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, California.
- Frantz, D. R. (1972). Non-linearities in genetic adaptive search. Doctoral dissertation, University of Michigan. *Dissertation Abstracts International*, 33(11), 5240B-5241B. University Microfilms No. 73-11, 116.
- Gorges-Schleuter, M. (1989). ASPARAGOS an asynchronous parallel optimization strategy. In *Proceedings Third International Conference on Genetic Algorithms*, Morgan Kaufman.
- Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum Associates, 41-49.
- Goldberg D. E. & Smith R. E. (1987). Non-stationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum Associates
- Goldberg, D. E. (1987b). Simple genetic algorithms and the minimal deceptive problem. In Davis, L. (Ed.), *Genetic Algorithms and Simulated Annealing* (74), London: Pitman.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading MA: Addison-Wesley.
- Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms, *Foundations of Genetic Algorithms*, pages 69-93.
- Goldberg, D. E., Deb, K., & Horn, J. (1992). *Massive multimodality, deception, and genetic algorithms* (IlliGAL Technical Report No. 92005). Urbana, IL: University of Illinois at Urbana-Champaign.
- Gordon, V.S., Whitley, D., and Böhm, A.P.W. (1992). Dataflow parallelism in genetic algorithms. In *Parallel Problem Solving from Nature 2*, Elsevier Science Publishers.

- Grefenstette, J.J. (1981). Parallel adaptive algorithms for function optimization. Technical Report No. CS-81-19, Vanderbilt University, Computer Science Department.
- Grefenstette, J.J. (1992). Genetic algorithms for changing environments. In R. Manner and B. Manderick (eds.), *Parallel Problem Solving form Nature*, 2. Amsterdam: North Holland, 137-144.
- Harik, G. R. (1995). Finding multimodal solutions using restricted tournament selection. In L. J. Eshelman (ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 24-31.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*, Ann Arbor MI: The University of Michigan Press.
- Horn, J. and Nafpliotis, N. (1993). *Multi-objective optimization using the niched pareto genetic algorithm*. IlliGAL Report No. 93005, Department of Computer Science, University of Illinois, Urbana-Champaign.
- Istvanick, W., Kryder, A., Lewandowski, G., Meidânis, J., Rang, A., Wyman, S., & Joseph, D. (1993). Dynamic methods for fragment assembly in large scale genome sequencing projects. In, T. N. Mudge, V. Milutinovic, & L. Hunter (Eds.), *Proceedings of the Twenty Sixth Annual Hawaii International Conference on System Sciences: Architecture and Biotechnology Computing*. Los Alamitos, CA: IEEE Computer Society Press, 534-543.
- Johnson, V. M. and Rogers, L. L. (1994). *Location analysis in groundwater remediation using artificial neural networks and Monte Carlo simulations*. Preprint UCRL-JC-117289, Lawrence Livermore National Laboratory.
- Krawczak, M. (1988). Algorithms for the restriction-site mapping of DNA molecules. In *Proceedings of the National Academy of Sciences*. USA 85, 7298-7301.
- Liang, J., Chang, C. C., Lee, R. C. T., and Wang, J. S. (1991). Solving the file design problem with neural networks. *Tenth Annual International Phoenix Conference on Computers and Communications*.
- LLNL (1992). *Record of Decision for the Livermore National Laboratory Livermore Site*. UCRL-AR-109105.
- Mahfoud, S. W. (1992). Crowding and preselection revisited. In R. Manner & B. Manderick (Eds.), *Proceedings of Parallel Problem Solving from Nature 2*. New York, NY: Elsevier Science B. V., 27-36.
- Mahfoud, S. W. (1993). *Simple Analytical Models of Genetic Algorithms for Multimodal Function Optimization*, IlliGAL Report No. 93001,

- Dept. of Computer Science, University of Illinois, Urbana-Champaign, IL.
- Mahfoud, S. W. (1995a). *Niching Methods for Genetic Algorithms*. Doctoral dissertation / IlliGAL Rep. 95001. Urbana: University of Illinois, Illinois Genetic Algorithms Lab.
- Mahfoud, S. W. (1995b). A comparison of parallel and sequential niching methods. In L. J. Eshelman (ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 136-143.
- Manderick, B. and Spiessens, P. (1989). Fine-grained parallel genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, CA.
- Männer, R. and Manderick, B. eds. (1992). *Proceedings of Parallel Problem Solving from Nature 2*, Elsevier Science Publishers B. V..
- Maresky, J., Davidor, Y., Gitler, D., Aharoni, G., and Barak, A. (1995). Selectively destructive restart. In L. J. Eshelman (ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers.
- McGraw, J., et. al. (1985). SISAL - Streams and iterations in a single-assignment language. Language reference manual, version 1.2. Lawrence Livermore National Laboratory manual M-146 (Rev. 1), Livermore, CA.
- Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. New York, NY: Springer-Verlag.
- Mühlenbein, H. (1989). Parallel genetic algorithms, population genetics and combinatorial optimization, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, CA.
- Mühlenbein, H., Schomish, M., and Born, J. (1991). The parallel genetic algorithms as function optimizer. In *Proceedings Fourth International Conference on Genetic Algorithms*, Morgan Kaufman.
- Ng, K. P. & Wong, K. C. (1995). A new diploid scheme and dominance change mechanism for non-stationary function optimization. In L. J. Eshelman (ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers.
- Nix, A. & Vose, M. D. (1992). Modeling genetic algorithms with Markov chains, *Annals of Mathematics and Artificial Intelligence* 5, 79-88.
- Olson, M. V., Dutchik, J. W., Graham, M. Y., Brodeur, G. M., Helms, C., Frank, M., MacCollin, M., Scheinman, R., & Frank, T. (1986). Random-clone strategy for genomic restriction mapping yeast. In *Proceedings of the National Academy of Sciences*. USA 83, 7826-7830.

- Opatrny, J. (1979). The total ordering problem. *SIAM Journal of Computing*, 8:1, 111-114.
- Parsons, R. J., Forrest, S., & Burks, C. (1993). Genetic algorithms for DNA sequence assembly. In *Proceedings 1st International Conference on Intelligent Systems for Molecular Biology*. Menlo Park, CA: AAAI Press, 310-318.
- Parsons, R. J., Forrest, S., & Burks, C. (1995). Genetic algorithms, operators and DNA fragment assembly. To appear in *Machine Learning*. Boston MA: Kluwer Academic Publishers.
- Pearson, W. (1982). Automatic construction of restriction site maps. *Nucleic Acids Research* 10, 217-228.
- Platt, D. M., & Dix, T. I. (1993). Construction of restriction maps using a genetic algorithm, In T. N. Mudge, V. Milutinovic, & L. Hunter (Eds.), *Proceedings of the Twenty Sixth Annual Hawaii International Conference on System Sciences: Architecture and Biotechnology Computing*. Los Alamitos, CA: IEEE Computer Society Press, 756-762.
- Reid, R. W. and Vemuri, V. (1971). On the Noninferior Index Approach to Large-Scale Multi-Criteria systems. *Journal of Franklin Institute*, 291:241-254.
- Rogers, L. L., and Dowla, F. U. (1994). Optimal groundwater remediation using neural networks and the genetic algorithm. *Water Resources Research*, Vol. 30, No. 2, pp 458-481.
- Rogers, L. L., Dowla, F. U. , and Johnson, V. M. (1994). *Optimal field-scale groundwater remediation using neural networks and genetic algorithms*. Preprint UCRL-JC-113773, Lawrence Livermore National Laboratory.
- Rogers, L. L. (1992). Optimal groundwater remediation using artificial neural networks and the genetic algorithm. Ph.D. Thesis, Preprint UCRL-LR-114125, Lawrence Livermore National Laboratory.
- Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5:1, 96-101.
- Schaffer, J. D., ed., (1989). *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, California.
- Smith, R. E., Forrest, S., & Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1:2, 128-149.
- Spears, W. M. (1994). Simple subpopulation schemes, in *Proceedings of the 94 Evolutionary Programming Conference*, San Diego, CA.

- Spiessens, P. and Manderick, B. (1991). A massively parallel genetic algorithm - implementation and first analysis. In *Proceedings Fourth International Conference on Genetic Algorithms*, Morgan Kaufman.
- Stallings, R. L., Torney, D. C., Hildebrand, C. E., Longmire, J. L., Deaven, L. L., Jett, J. H., Doggett, N. A., & Moyzis, R. K. (1990). Physical mapping of human chromosomes by repetitive sequence fingerprinting. In *Proceedings of the National Academy of Sciences*. USA 87, 6218-6222.
- Stefik, M. (1978). Inferring DNA structures from segmentation data. *Artificial Intelligence*, 11, 85-114.
- Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 2-9.
- Syswerda, G. and Palmucci, J. (1991). The application of genetic algorithms to resource scheduling. In R. K. Belew and L. B. Booker, (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 502-508.
- Tanese, R. (1989). Distributed genetic algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers San Mateo, California.
- Vemuri, V. (1974). Multiple objective optimization in water resource systems. *Water Resources Research* 10:44-48.
- Vemuri, V. (1978). *Modeling of Complex Systems: An Introduction*. Academic Press.
- Vemuri, V. and Karplus, W. J. (1969). Identification of nonlinear parameters of ground water basins via hybrid computation. *Water Resources Research*, 5:172-185.
- Vemuri, V. and Karplus, W. J. (1981). *Digital Computer Treatment of Partial Differential Equations*. Prentice Hall, Engle Wood Cliffs.
- Vemuri, V. and Cedeño, W. (1995). Multi-niche crowding for multimodal search. In L. Chambers (Ed.), *Practical Handbook of Genetic Algorithms and Applications*, volume II, CRC Press.
- Vose, M.D. (1991). Formalizing genetic algorithms, *Technical Report CS-91-127*, Computer Science Department, University of Tennessee.
- Vose, M.D. (1992). Modeling simple genetic algorithms, *Proceedings of Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers San Mateo, CA.
- Vose, M.D. (1993). A critical examination of the schema theorem, *Technical Report CS-93-212*, Computer Science Department, University of Tennessee.

- Vose, M. D. & Liepins, G. E. (1991). Punctuated equilibria in genetic search, *Complex Systems* 5, 31-44.
- Voss, C. I. (1984). SUTRA, Saturated-Unsaturated Transport: A finite-element simulation model for saturated-unsaturated, fluid-density-dependent groundwater flow with energy transport or chemically reactive single-species solute transport. *Water Resources Investigations Report*, U. S. Geological Survey, 84-4369.
- Waterman, M. S., & Griggs, J. R. (1986). Internal graphs and maps of DNA. *Bulletin of Mathematical Biology*, 48, 189-195.
- Whitley, D. (1988). GENITOR: a different genetic algorithm. In *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*. Denver Colorado, 118-130.
- Whitley, D. (1991). Fundamental principles of deception in genetic search, in *Foundations of Genetic Algorithms*, 221.
- Whitley, D., Starkweather, T., & Fugway, D. (1989). Scheduling problems and traveling salesmen: the genetic edge recombination operator. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 133-140.
- Winston, P. H. (1992). *Artificial Intelligence*. Addison-Wesley, Reading, MA, third edition, 518-528.
- Yin, X, and Gernay, N. (1993). A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization, *Proceedings of the Artificial Neural Nets and Genetic Algorithms Conference*, Austria, Springer-Verlag, 450-457.

10. Curriculum Vitae

Walter Cedeño
446 Creekside Drive
Downingtown PA, 19335
(610) 873-2261

CAREER SUMMARY

Over ten years experience providing software solutions under different operating systems and hardware platforms using state of the art software tools and techniques. Experience using parallel processing and soft computing techniques (genetic algorithms, neural networks, etc.) to solve complex problems in science and engineering. Extensive experience applying genetic algorithms and heuristics to optimization problems. Some of the areas include DNA Mapping, Drug Design, and Database Optimization. Extensive experience in multitasking real time systems and embedded control software. Experience in client/server and distributed processing applications.

RESEARCH INTERESTS

Operations research, optimization, genetic algorithms, neural networks, fuzzy logic, DNA mapping, drug design, graph layout algorithms, distributed processing, parallel processing, and real time systems.

EDUCATION

Ph D, Computer Science: University of California, Davis, CA. September 1995.

MS, Applied Mathematics: University of Puerto Rico, Rio Piedras, PR. July 1985.

BS, Mathematics: University of Puerto Rico, Rio Piedras, PR. December 1982.

HONORS AND AWARDS

Best Student Paper Award, WNN/FNN '93, San Francisco, CA. September 1993.

Mathematics Medal, University of Puerto Rico, Rio Piedras, PR. June 1983.

High Achievements Honor, University of Puerto Rico, Rio Piedras, PR. June 1983.

Magna Cum Laude, University of Puerto Rico, Rio Piedras, PR. December 1982.

Member Natural Sciences Honor Society, University of Puerto Rico, Rio Piedras, PR. 81-82.

PROFESSIONAL EXPERIENCE

Primavera Systems Inc., Bala Cynwyd, PA, December 1994 to Present

Lead Software Engineer, Programming Dept., December 1994

Responsible for the analysis, design and implementation of the next generation of client/server project management software. Research, design and implementation of framework for drawing PERT diagrams using graph algorithms for directed acyclic graphs. Developed thread class using Visual C++ to allow background processing under Windows.

Sterling Winthrop Inc., Malvern, PA, June 1989 to November 1994

Project Manager, Research Scientific Computing, May 1993 to November 1994

Research and application of genetic algorithms to risk analysis for project prioritization. Research and application of graph algorithms and genetic algorithms to drug design. Analysis, design, and implementation of molecular modeling package using Visual C++ under Windows.

Consultant, Research Scientific Computing, June 1989 to April 1993

Research and application of genetic algorithms for multi-modal function optimization. Develop genetic algorithms library using ANSI C under DOS and UNIX for application to complex multiple optima optimization problems.

Analysis, design, and implementation of the Device Control subsystem for the Robot Workbench project using ANSI C under DOS/iRMX OS. This is a multitasking subsystem to control and monitor laboratory instruments on a robot workbench. The subsystem communicates with multiple devices connected to a HP-IB bus and RS-232 serial lines.

Lawrence Livermore National Laboratory (LLNL), Livermore, CA, September 1985 to May 1993

Computer Scientist/Math Programmer, Computing Research Group, October 1991 to May 1993

Research and development of parallel genetic algorithms using SISAL under UNIX on a four processor SGI IRIS system for application to function optimization and combinatorial problems. Research and application of genetic algorithms to the DNA restriction fragment map assembly problem. Design and implementation of X Windows interface using C under UNIX for TWINE (debugger for parallel functional language called SISAL).

Computer Scientist/Math Programmer, Technology Information Systems, March 1991 to September 1991

Design and implementation of public key encryption library using C under UNIX. The library allows mail messages to use RSA public key encryption standards and is part of the electronic commerce project.

Computer Scientist/Math Programmer, Safeguards and Security Group, September 1985 to February 1991

Analysis, design, and implementation of the Entry Control Device (ECD) embedded control subsystem using C under iRMX I Multitasking OS. The ECD is part of the CAIN II access control system. It is an intelligent controller used at LLNL to control personnel access to sensitive areas. It is an event driven system based on the 8086 CMOS CPU in a Multibus I configuration. Design and implementation of device drivers using PLM for the ECD.

***Center for Energy & Environment Research*, Rio Piedras, PR, December 1982 to September 1985**

System Manager/Programmer, Seismological Division, December 1982 to September 1985

Design and implementation of application to analyze and display seismic events on a contour map of the Caribbean using a VT 100 RETRO graphics terminal using C under VENIX (UNIX like OS for the PDP 11/23). System Manager for PC network using Novell's netware software.

***University Of Puerto Rico*, Rio Piedras, PR, August 1983 to July 1985**

Research Assistant, Department of Applied Mathematics, August 1983 to July 1985

Teach preparation for Calculus class every semester. Conduct research in the area of graph theory.

***Lawrence Livermore National Laboratory*, Livermore, CA, Summer 1983, 1984, and 1985.**

Summer Student Employee, Application Systems Division, Summer 1985

Design and implementation of application using Lisp on a Xerox 1108 computer to print different file types to a postscript printer. The application consists of an icon with a popup menu and allows the user to print Xerox press files, bitmaps, source code, and windows.

Summer Student Employee, Application Systems Division, Summer 1984

Convert the DIGLIB graphics library from FORTRAN to C on a VAX 750 running VMS. Develop program to plot experimental data on a Tektronix 4014.

Summer Student Employee, Application Systems Division, Summer 1985

Develop software modules for a data acquisition application in FORTRAN on a LSI-11 under RT-11.

PUBLICATIONS AND TECHNICAL REPORTS

Cedeño, W. and Vemuri, V. (1995). Genetic algorithms in aquifer management. To be published in the *Journal of Microcomputer Applications*.

Cedeño, W., Vemuri, V., and Slezak, T. (1995). Multi-Niche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments. *Evolutionary Computation*, 2:4, 321-345, Winter 1994.

Vemuri, V. and Cedeño, W. (1995). A new genetic algorithm for multi-objective optimization in water resource management. *IEEE International Conference on Evolutionary Computing*, Perth, Australia.

Vemuri, V. and Cedeño, W. (1995). Multi-niche crowding for multimodal search. In L. Chambers (Ed.), *Practical Handbook of Genetic Algorithms and Applications*, volume II, CRC Press.

Cedeño, W. (1993). DNA restriction fragment map assembly with genetic algorithms. In J. R. Koza (ed.) *Genetics Algorithms at Stanford*. Stanford, CA:Stanford Bookstore, 40-49.

Cedeño, W. and Vemuri, V. (1993). An investigation of DNA mapping with genetic algorithms: preliminary results, *Proceedings of the Fifth Workshop on Neural Networks*, SPIE 2204, 133-140.

Cedeño, W. (1992). Genetic algorithms in SISAL to solve the file design problem, *Proceedings of the Second SISAL User's Conference*, San Diego CA.

Cedeño, W. and Vemuri, V. (1992). Dynamic multimodal function optimization using genetic algorithms, *Proceedings of the XVIII Latin-American Informatics Conference*, Las Palmas de Gran Canaria, Spain: University of Las Palmas, 292-301.

Cedeño, W. (1991). "CAIN II Embedded Control Systems Reference Guide", Lawrence Livermore National Laboratory, Livermore, California.

Cedeño, W. & Voegeli, B. (1991). "Reference Guide to CAIN II Messages", Lawrence Livermore National Laboratory, Livermore, California.

Cedeño, W. (1990). "ECD Configuration Guide", UCRL-MA-108535, Lawrence Livermore National Laboratory, Livermore, California.

Cedeño, W. (1988). "LLNL Access Control System: Entry Control Device", The 8th Biennial Cube Symposium, Albuquerque, New Mexico.

Cedeño, W. & Jong, D. (1988). "Entry Control Device Access List Record", UCAR-10237, Lawrence Livermore National Laboratory, Livermore, California.

Cedeño, W. & Jong, D. (1988). "Entry Control Device States", UCAR-10238, Lawrence Livermore National Laboratory, Livermore, California.

Cedeño, W. (1985). "Using Symmetry to Prove Hamiltonicity in Bipartite Graphs", M.S. Thesis, University of Puerto Rico, Rio Piedras, PR.

Cedeño, W. (1984). "Exploiting Symmetry in Bipartite Graphs", 4th Interdisciplinary Science Meeting, University of Puerto Rico, Rio Piedras, PR.

PERSONAL

Active DOE Q-Clearance.

Can speak and write Spanish fluently.

GROUPS AND ORGANIZATIONS

Society of Hispanic Professional Engineers, Greater Philadelphia Chapter

Member, IEEE Standards Evolutionary Computation Working Group

Member of Genetic Programming 96 Conference Committee