<u>**Chapter 0**</u>

V. Rao Vemuri and Walter Cedeño
Department of Applied Science
University of California, Davis and
Lawrence Livermore National Laboratory
Livermore, CA 94550

(vemuri@icdc.llnl.gov and wcedeno@llnl.gov)

Multi-Niche Crowding for Multi-Modal Search

Abstract

Multi-Niche Crowding (MNC) is a new genetic algorithm designed for locating multiple peaks in a multimodal function. The validity of this method is demonstrated by applying it first to a variety of test functions. Then the method is applied to solve the problem of assembling the so-called restriction fragments obtained from partial digestion of DNA molecules.

0.1 Introduction

Searching for extrema in a multi-modal space is different from locating the extremum of a unimodal function. When a search technique proven to be useful for unimodal functions is applied to multi-modal functions, the method tends to converge to an optimum in the local neighborhood of the first guess. One can use methods like simulated annealing to escape from a local optimum and locate the global optimum. However, there are many applications where the location of "k best extrema" of a multi-modal function are of interest. Searching for these locations goes by the name "multi-modal optimization." In this chapter we describe a genetic algorithm (GA) suitable for multi-modal function optimization.

A GA is a mathematical search technique based on the principles of natural selection and genetic recombination [Holland, 1975]. A possible solution to a problem is referred to as an *individual.* An individual is represented by a computational data structure called a *chromosome*, which is encoded using a fixed length alphabet. *Species* are individuals with a common characteristic and *niches* are subdomains of the search space. By encouraging niching and speciation, GAs can facilitate simultaneous convergence to more than one optimum in a multi-modal search space.

Section 2 of this chapter describes multi-modal function optimization using the multi-niche crowding (MNC) model. In Section 3 the MNC method is applied to a number of test functions. In Section 4, the MNC method is used to solve the so-called DNA restriction fragment assembly problem, an important and challenging problem from biotechnology. In the last two sections of the chapter we included some discussion on the suitability of this method to the solve the restriction fragment assembly problem.

0.2 Genetic Algorithms for Multi-Modal Search

There are many versions of genetic algorithms, one differing from another in some detail. In a nutshell, all genetic algorithms have two basic steps: during the *selection step*, a decision is made as to who in the population is allowed to produce offspring, and during the *replacement step* another decision is made as to which of the members from one generation are forced to perish (or vacate a slot) in order to make room for an offspring to compete (or, occupy a slot).

The Simple GA (or SGA), which will be used as a point of departure for presenting the method discussed here, starts by randomly generating a population of *N* individuals, that is, individual solutions. These individuals are evaluated for their fitness. Individuals with higher fitness scores are selected, with replacement, to create a mating pool of size *N*. This method of selection is called *fitness proportionate reproduction* (FPR). The genetic operators of crossover and mutation are applied at this stage in a probabilistic manner which results in some individuals from the mating pool to reproduce. The assumption here is that each pair of parents produces only one pair of offspring through the crossover operation. Now the population pool contains some individuals who never got a chance to reproduce and the offspring of those who got a chance to reproduce. The procedure continues until a suitable termination condition is satisfied. All other versions of GAs differ from this basic method in some detail or another.

The steady-state GA (or SSGA) differs from SGA mainly in the replacement step, and to a lesser extent on the way the genetic operators are applied [Whitley, 1988; Syswerda, 1989]. The SSGA selects two individuals using FPR and allows them to mate to produce two offspring. This selection step is identical to the corresponding step of SGA. However, in SSGA, the offspring are inserted into the population, thus replacing two individuals, soon after they are generated whereas the SGA generates *N* offspring prior to replacing the entire population. In other words SGA uses *simultaneous* replacement strategy whereas the SSGA uses the *successive* replacement strategy. Thus SGA and SSGA are analogous, respectively, to the Jacobi and Gauss-Seidal methods of solving systems of algebraic equations.

Both SGA and SSGA suffer from the possibility of premature convergence to a local minimum, primarily due to the *selection pressure* exerted by the FPR rule. Simply assigning an exponential number of mating trials to those members of the population that exhibit above average survival traits, as the FPR rule does, is not a good strategy for a thorough exploration of complex search spaces with multiple peaks. Due to this problem as well as the *deceptiveness* exhibited by muti-modal search spaces [Goldberg et al., 1992], SGA and SSGA are not suitable for multi-modal search and optimization.

0.2.1 Multi-Modal Search: Crowding-Based Methods:

The GA model described here, called *multi-niche crowding* (or, MNC), has the ability to converge to multiple solutions at the same time by encouraging competition between individuals within the same locally optimal group (Cedeño and Vemuri, 1992, 1994). In MNC, both the selection and replacement steps of the SGA are modified with the introduction of some form of *crowding* in order to render it suitable for searching spaces characterized by multiple peaks or niches. So it is essential that the concept of crowding is briefly reviewed here.

Crowding (De Jong, 1975) is a generalization of *preselection* (Cavicchio, 1970). 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. In order to make room for these offspring, it is necessary to identify two members of the population for replacement. 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 C individuals is selected at random from the population. C, called the *crowding factor*, indicates the size of the group. A value of C = 2 or 3 appears to work well for De Jong. Second, the bit strings in the offspring chromosomes are compared with those of the C individuals in the group using Hamming distance as a measure of similarity. The group member that is most similar to the offspring is now replaced by the offspring.This procedure is repeated for the other offspring as well. This second offspring can conceivably replace its own sibling that just entered the population pool, although such a scenario is rather unlikely. In any event, crowding is essentially a successive replacement strategy. This strategy maintains the diversity in the population and postpones premature convergence. Crowding cannot maintain stable subpopulations due to the selection pressure imparted by FPR. Summarizing, in crowding offspring replace similar individuals from the population. Crowding slows down premature convergence of the traditional GA and in most cases can find the global optimum in a multi-modal search space. On the other hand it does not converge to multiple solutions and after many generations one of the peaks takes over.

In *deterministic crowding* (Mahfound, 1992) selection pressure is eliminated and preselection is introduced to obtain a GA suitable for multi-modal function optimization. This version appears to have minimal overhead, thus contributing to its efficiency. In this method selection pressure is eliminated by allowing individuals to mate at random with any other individual in the population. Pressure is applied, however, during 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 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. It is not clear if multiple solutions can be maintained for many generations using this method, although it appears that multiple solutions are sustained for more generations than when crowding is used alone.

0.2.2 Multi-Modal Search: Sharing-Based Methods

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 values of an individual are adjusted downward in accordance with the number of individuals in its neighborhood or niche. The more individuals there are in a niche, the more pressure they create on each other. The downward adjustment of fitness value is done with the help of a suitably defined *sharing function,* a function that takes into account the similarity (i. e., physical proximity) between two individuals. This approach allows multiple solutions to converge in parallel but it is very computing intensive. Deb and Goldberg [1989] applied mating restriction to sharing methods. Mating restriction only allows individuals within the same niche to mate. The method is too restrictive and the user must have an idea of the search space in order to define a suitable sharing function. Yin and Germay [1993] introduced cluster analysis to sharing to reduce its complexity and to group similar members naturally. Here the user must provided special attention to the parameters for the clustering algorithm to select the correct ones for the problem at hand.

Beasley *et al.* [1993] applied traditional GAs to multi-modal function optimization by using a fitness derating function to prevent convergence to a known local optima. In their approach the GA is applied iteratively to the problem and every solution found in previous iterations is used to derate the fitness of individuals near them. The time complexity is similar to that of sharing functions. Like in sharing the user must have an idea of the search space beforehand. Derating a large region may eliminate a possible solution.

0.2.3 Multi-Niche Crowding (MNC)

In multi-niche crowding (MNC), both the selection and replacement steps are modified with some type of crowding. The idea is to eliminate the selection pressure caused by FPR while allowing the population to maintain some diversity. This objective is achieved, in part, by encouraging mating and replacement within members of the same niche while allowing for some competition for 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 minima.

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 takes place in two steps. First, an individual A is selected for mating. This selection can be either sequential or random. Second, its mate M is selected, not from the entire population, but from a group of individuals of size c_s , picked at random from the population. The mate M thus chosen must be the one who is the most "similar" to A. The similarity metric used here is not a *genotypic metric* such as the Hamming distance, but a suitably defined *phenotypic distance metric*. Crowding Selection promotes mating between individuals from the same niche while allowing matings between individuals from different niches.

During the replacement step, MNC uses a replacement policy called *worst among the most similar.* 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_f groups are created by randomly picking *s* individuals per group from the population. These groups are called *crowding factor groups.* Second, one individual from each group that is most similar to the offspring is identified. This gives c_f individuals that are candidates for replacement by virtue of their similarity to the offspring that will replace them. From this group of most similar individuals, we pick the one with the lowest *fitness* to die and that slot is filled with the offspring.

A similar technique called *enhanced crowding* (Goldberg 1989) has been used before in classifier systems, but there the most similar individual out of a group of worst candidates is replaced. Figure 0.1 is an example of the worst among most similar replacement strategy.

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 *worst among most similar replacement* offspring are likely to replace low fitted individuals from the same niche. It can also happen that it replaces a higher fitted 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 estimulates competition between members of the same niche and between members belonging to different niches as well.

The following pseudo-code summarizes the salient features of the method:

```
1 Generate initial population of N individuals
2 For gen = 1 to MAX_GEN
3         For i = 1 to N
4                 Use crowding selection to find mate for individual i
5                 Mate and mutate
6                 Insert offspring in the population using
                        worst among most similar replacement.
```

If we use FPR in Step 4 shown above and replace the lowest fitted individuals in the population (Step 6) with the newly generated offspring, this model corresponds to a steady-state GA. In contrast with the most common generational GA, offspring are available for mating as soon as they are generated, and good individuals can survive for many generations. For the purposes of this paper, a generation is every *N*/2 mating operations, where *N* is the population size.

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 *Cs* and *Cf s*. Normally *Cf* is a value in the interval [1, 4], and *Cs* and *s* are values ranging from 1% to 15% of the population size.

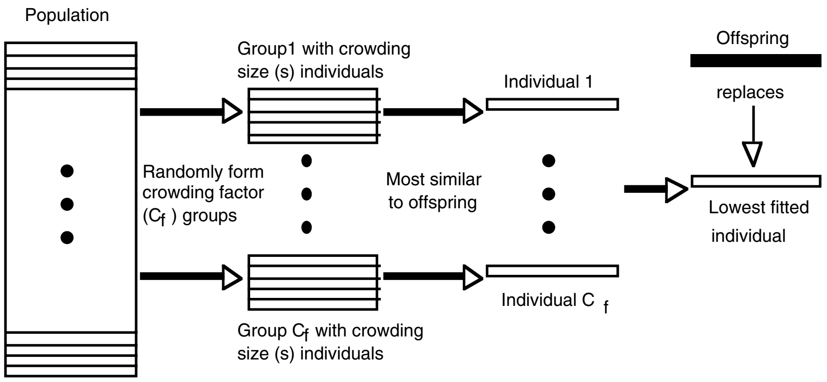

Population
Group1 with crowding size (s) individuals
Individual 1
Offspring
replaces
Randomly form crowding factor (Cf) groups
Most similar to offspring
Lowest fitted individual
Individual C f
Group Cf with crowding size (s) individuals

Figure 0.1: Schematic showing crowding factor groups created during the replacement step.

0.3 Application of MNC to Multi-Modal Test Functions

To evaluate the performance of the MNC model we used five different test functions. The first three of these functions have been used earlier by other investigators. Functions *F1(x)* and *F2(x)* defined by

$$F_1(x) = \sin^6(5.1\pi x + 0.5) \; ,$$

$$F_2(x) = \exp^{-4(\ln 2)(x-0.0667)^2/0.64} \sin^6(5.1\pi x + 0.5) \, ,$$

are shown on Figure 0.2. They correspond to the five-optima sine functions used by Goldberg in his work on sharing. Both functions were maximized using binary chromosome encoding for numbers in the interval [0, 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_2 sharing will be able to maintain a proportional number of individuals for larger number of generations. Function *F3(x,y)*, called "Shekel's foxholes" with twenty five optima, was used by De Jong in his work with crowding.

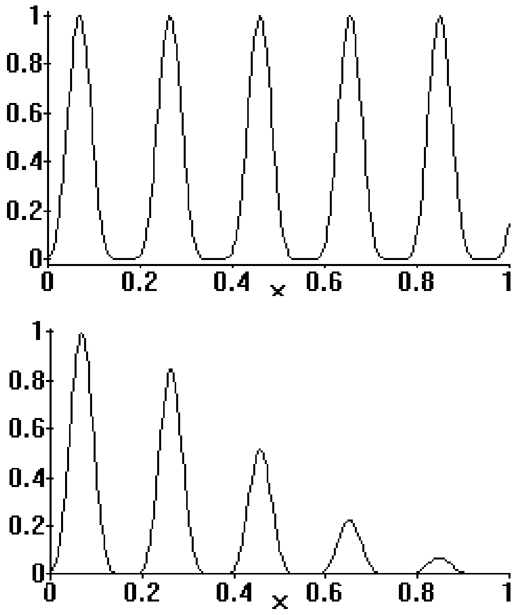


Figure 0.2: Test functions *F1(x)* (top) and *F2(x)* (bottom).

We also considered other functions not exhibiting the symmetry present in the above functions. Function *F4(x,y)*, shown on the left in Figure 0.4, contains two global optima with the same height and width but located far apart. Function *F5(x,y)*, the sample shown on the right in Figure 0.4, contains five optima with height, width, and location chosen at random in every run. Both of these functions are defined by

$$\sum_{i=1}^{p} \frac{A_i}{1+W_i((x-X_i)^2+(y-Y_i)^2)},$$

where p indicates the number of peaks in the function, (Xi, Yi) the coordinates of peak i, Ai the height of peak i, andWi determines how narrow or wide is the base of peak i.

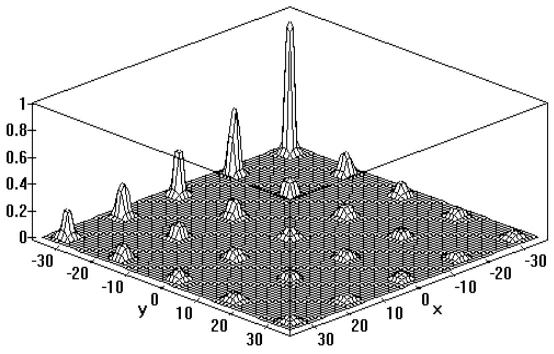
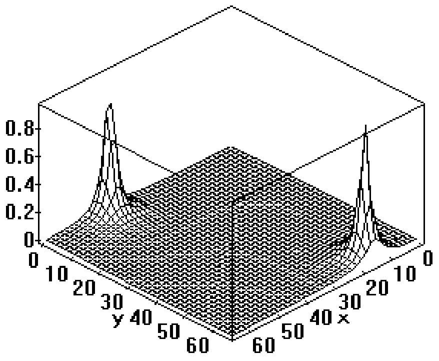


Figure 0.3: Test function *F3*(x), Shekel's foxholes.



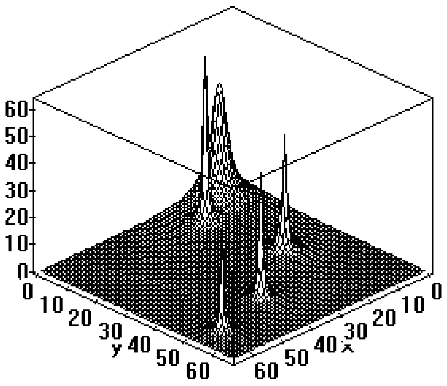


Figure 0.4: Test functions *F4* (top) and *F5* (bottom).

The simulations were done in a 486/33MHz PC. The variables *x* and *y* were encoded using two 30 bit chromosomes. To generate the initial population the search space was divided into *n* (population size) equally sized regions. One individual was chosen at random in each region. The crossover probability (*pc*) was set at 0.95 and mutation probability (*pm*) at 0.001. Similarity between two individuals was determined by the Euclidean distance between the two points. The MNC GA was executed for 100 generations in each run. Other parameters are summarized in Table 0.1.

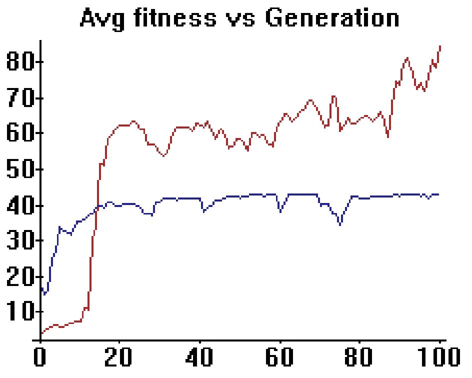
	F1 & *F2*	*F3*	*F4*	*F5*
Population size (*n*):	100	500	100	200
Number of chromosomes:	1	2	2	2
Crowding selection size (*Cs*):	15	75	5	15
Crowding factor (*Cf*):	2	2	3	3
Crowding size (*s*):	15	75	5	15

Table 0.1: Function specific parameters used in the MNC GA.

For all test cases the MNC GA was able to maintain stable subpopulations in all the higher peaks without exhibiting premature convergence. Only very small peaks in *F3* and *F5* did not have any significant number of individuals in the last generation even though some were present during the initial generations. The number of individuals at the lower peaks decreased as the fitness of the individuals in other peaks increased. This can be observed in Figure 0.5 for the two peaks located near (0,0) in function *F5*. The plot on the left has the average fitness of each peak for every generation. The plot on the right has the number of individuals in each peak for every generation. During the initial ten generations the wider peak had more individuals in the population. As the fitness of the individuals in the other peak improved the number of individuals increased. After about 20 generations the number of individuals in the skinnier peak was greater than those in the wider peak. There are of course other factors that contribute to this pattern and that needs to be investigated further.

We also observed that the number of individuals in a peak is related to more than just its height. In function *F3* 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. More tests are needed to determine other factors affecting the number of individuals in a peak.

The functions *F1*, *F2*, and *F4* were not hard for the GA. After 100 generations all five global optima in function *F1* were found including the local optimum at *x* = 1.0. All five optima were also found for function *F2*. Both the peaks at *F4* were also successfully found. The distance between the peaks did not cause any problem for the GA.



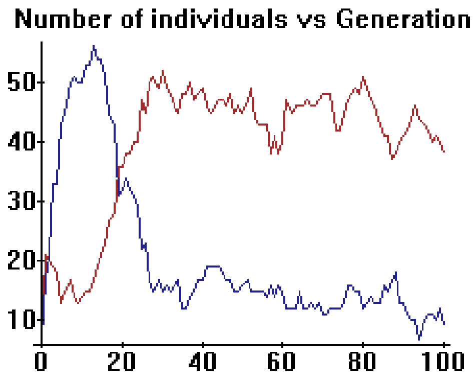


Figure 0.5: Average fitness and individuals in population for two peaks in function *F5*.

Overall, the properties exhibited by the MNC GA are very encouraging. A more rigorous analysis is necessary before statements can be made about the validity of this method in a more general context.

0.4 Application to DNA Restriction Fragment Map Assembly

The genetic material contained in the chromosomes of a cell is collectively called the genome. Chromosomes are essentially DNA molecules which are made out of four distinct types of molecules called nucleotides. Although these four nucleotides, denoted here by the letters A, C, G, T, can theoretically form sixteen pairs, only AT, TA, CG and GC pairings are allowed. It is estimated that the human genome is comprised of about three billion of these "base-pairs." The ambitious goal of the Human Genome Project is to decipher or map the exact sequence of these three billion nucleotide molecules.

0.4.1 Restriction Fragment Data

The experimental process of gathering DNA data is too intricate to describe here. However, a thumbnail sketch is sufficient to describe the broad outlines of what really takes place in the laboratory. First, the DNA molecule is cut to a manageable size using a so-called restriction enzyme. If total digestion is desired, then the enzymes are allowed to act on the restriction sites (of multiple copies) for a sufficiently long time and the result will be several identical pieces of DNA with non-overlapping base-pair sequences. However, under partial digestion, not all restriction sites experience the same rate of reaction. Furthermore, under partial digestion, the same restriction site on different copies of a DNA may behave differently. At some point, when the reaction is stopped, one finds many DNA fragments with overlapping base-pair sequences. Both complete digestion and partial digestion have roles to play in DNA sequencing studies.

Typically, the hierarchy of the fragmentation process goes somewhat like this. The DNA under study is first divided, using complete digestion into non-overlapping islands, called *contigs,* whose lengths may range between 150K to 200K base-pairs. This step is purely for the convenience of working with shorter pieces of a chromosome of manageable length. These contigs are then inserted into a type of viral DNA called a cloning vector. The type of cloning vector used in our study are called *cosmids*. These cosmid cloning vectors, containing a piece of the human DNA, are then used to infect bacterial cells at the rate of one per bacterial cell. The bacteria reproduce rapidly, and in doing so produce many copies of the piece of human DNA that was inserted into the vector. The identical copies of the contigs thus obtained are called cosmid clones. These clones are now subjected to partial digestion with restriction enzymes. This process yields several copies of cosmid clones, with overlapping base-pair segments. That is, identical base-pair segments from the original parent contig may appear in two cosmid clones, say Cosmid clone A and Cosmid clone B. Typically, each cosmid clone is approximately 40K base-pairs long. Deciding the relative position of a cosmid clone on the parent contig by inspecting these 40,000 base-pairs is a tortuous task. To render the problem more manageable, each cosmid clone is further divided into the so-called *restriction fragments* using complete digestion with a single restriction enzyme, such as EcoRI. These restriction fragments can be physically separated by size when they are placed in a porous gel. When an electrical current is applied to the gel, fragments of different sizes line up as bands and this pattern of bands is called the *fingerprint* of the cosmid clone. These fragments typically range in length from 0.5K to 15K base-pairs. A pictorial view of this process is shown in Figure 0.6. During this process information about the relative location of the islands, cosmid clones, and fragments in the original DNA is lost.

The DNA Restriction Fragment Map Assembly problem then is comprised of reassembling the cosmid clones in order to reestablish the correct sequence of base-pairs in the original contig. This reassembly is not a trivial exercise. First, there are only four base-pairs, repeated thousands of times along the length of a DNA with no predictable pattern. So when a DNA is cut down to the size of fragments, it is conceivable that one finds many fragments that are similar. Second, due to experimental difficulties, the restriction enzymes do not always succeed in cutting the fragments. Due to these reasons, assembling the contigs from fragment data continues to be a challenging problem.

To establish the relative position of each cosmid clone on a contig, we determine 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 should be equal to the contig's original size. The problem is one of assembling cosmid clone sequences as shown in Figure 0.6. 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) are 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.

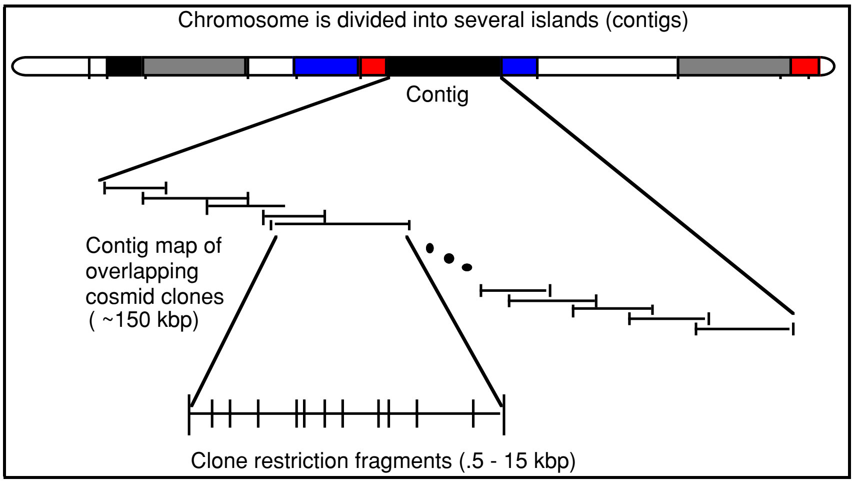

Chromosome is divided into several islands (contigs)
Contig
Contig map of overlapping cosmid clones (~150 kbp)
Clone restriction fragments (.5 - 15 kbp)

Figure 0.6: Physical map for island using fragments from a set of overlapping clones.

0.4.2 Problem Representation

Before going into the details about the use of genetic operators, it is important to show how the data for the problem is presented to the GA. Figure 0.7 shows fragment sizes obtained from fingerprinting for a set of overlapping cosmid clones.

ALLELE	CLONE									
NUMBER	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   10.49 0.58 1.74
C9    10406    2.03 1.43 2.34 6.28 5.46 8.58 7.27
```

Figure 0.7: Cosmid clones with fragment data.

For example, cosmid clone with the ID number 5154 which is also labelled 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 labelled 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 Allele 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 0.8. The matching of the fragment lengths is not perfect. Indeed, the mismatch at other positions is indeed large. The goal of this problem is to maximize this type of matching while minimizing the number and degree of mismatches and keeping the total length of the assembly within reasonable limits. The data for this problem consist of the *M* cosmid clones with their fragments 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

```
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

C8      21230    0.96 1.68  1.08 4.77 8.47 1.44 2.37 6.29 0.62
```

Figure 0.8. An example of fragment assembly.

0.4.3 Chromosome Encoding

The encoding for this problem is very simple. Each allele in the chromosome has a label between 0 and *M* - 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 0.7, 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 *M* - 1 without replacement.

	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 0.9: 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.

0.4.4 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 0.7 by 100, just to avoid dealing with decimals. All values are computed using integers 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 0.9, contains the number of fragments that match between two clones Ci and Cj, within an error tolerance *e* . The other matrix, the error matrix E shown on the right side of Figure 0.9, 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 (C0) and clone No. 5154 (C3) there are two pairs of fragments that match within the specified tolerance of e = 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.

Our goal is to arrange the cosmid clones as shown in Figure 0.6 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 0.6. 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 phenomenon makes us 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.

$$fitness = \sum_{i=0}^{M-1} \sum_{\substack{j=-1 \\ j \neq 0}}^{1} \left(\frac{Match[C_i, C_{i+j}]}{Count[C_{i+j}]} \times (1 - \frac{Error[C_i, C_{i+j}]}{Match[C_i, C_{i+j}] \times \varepsilon})\right) * 100$$

Here C_i refers to the fact that the cosmid clone Ci has been placed in the ith position of the chromosome, C_{i+j} refers to the clone to the right or left (if any) of the ith position. Match [C_i , C_{i+j}] counts the number of fragments that matched between C_i and $C_{(i+j)}$ within the specified tolerance. That is Match [C_i , C_{i+j}] is the degree of match between C_i and $C_{(i+j)}$. This quantity is divided by Count [C_{i+j}] to get a normalized count for the degree of match. The term Error [C_i , C_{i+j}] refers to the total error accumulated over all fragments that matched between C_i and $C_{(i+j)}$. By dividing this with the number of matches times the error tolerance ε, we are essentially getting 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, M is the number of alleles in the chromosome.

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 tp 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 the error among groups of three clones as factors and study its effect on performance.

0.4.5 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 0.10. 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 0.10, 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 0.10) 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		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 0.10: 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 also limited 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.

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.

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.

0.4.6 Similarity Function

The similarity function is very simple also. It counts the number of dissimilar links between two individuals. Using the parents from Figure 0.10 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.

0.5 Results and Discussion

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

Population size:	200
Mutation probability:	0.06
Crossover probability:	1.00
Crowding selection group size (c_s)	10 (5% of population size)
Crowding subpopulation size:	10 (5% of population size)
Crowding factor (c_f):	3
Maximum number of generations to execute:	100
Tolerance *e*	10

These parameters were picked 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. All reported results were averaged over five runs. A population size of 200 was found satisfactory. Other sizes (in multiples of 50) were tried, but higher sizes did not provide new information about the problem and lower sizes in some cases did not converge to the best solutions seen before in the allowed number of generations. Mutation was set at 0.06, therefore an average of 12 individuals were mutated every generation. This low value of mutation was selected to avoid eliminating the best of population frequently and allow faster convergence within each niche. On the other hand the mating probability was set to a high value of 1.0 because in our GA all individuals have a high chance of mating with a similar individual. The group size for crowding selection and crowding subpopulation was set at 5% of the population size with a crowding factor of 3. For each individual, at most 5% of the population was examined for selection and at most 15% of the population examined for replacement. These 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 E was set to 10 to minimize false matches due to chance. Higher values of E increased the false matches more than true matches and therefore more possible clone sequences were found.

Some of the best sequences obtained for two different sets of overlapping clones are shown in Figure 0.11. Data for Set 1 is shown in Figure 0.7 and data for Set 2 is shown in the Appendix. The GA took an average of 50 seconds for each run. The figure shows the actual sequence for the data sets and the clone sequences (with their fitness) obtained by MNC.

Data Set 1 actual sequence and its fitness:
```
(8131 5154 21230 10406 18993 7442 5435 7255 12282 27714) 764
```

The k Best sequences found by the GA and their corresponding fitnesses:
```
      (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
      (8131 27714 12282  7255  5435  7442 18993 10406 21230  5154) 730
      (8131  5154 21230 10406 18993 27714 12282  7255  5435  7442) 725
```

Data Set 2 actual sequence and its fitness:
```
      (12595 6722 26999 29626 29064 18301 19811 29035 17755 28828 20235) 750
```

The k Best sequences found by the GA and their corresponding fitnesses:
```
     (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 29626 29064 26999  6722 12595)    750
     (19811 29035 17755 28828 20235 18301 29064 29626 26999  6722 12595)    749
     (12595  6722 26999 29626 29064 18301 19811 29035 17755 20235 28828)    748
     (18301 28828 20235 17755 19811 29035 29064 29626  6722 26999 12595)    748
     (18301 20235 28828 17755 19811 29035 29064 29626  6722 26999 12595)    747
     (20235 28828 17755 29035 19811 18301 29064 29626  6722 12595 26999)    747
     (12595 26999  6722 29626 29064 18301 19811 17755 29035 28828 20235)    747
     (12595  6722 26999 29626 29064 18301 28828 20235 17755 19811 29035)    747
     (12595 26999  6722 29626 29064 18301 28828 20235 29035 17755 19811)    746
```

```
(12595 26999  6722 29626 29064 18301 19811 17755 29035 20235 28828)    744
```

Figure 0.11: Clone sequences obtained by GA and actual sequence.

For Data Set 1 the GA was able to find the actual clone sequence. From the other sequences found, the fitness value of the next best is 15 less than the actual sequence. Similar gaps exist between all the sequences shown in Figure 0.10 for Data Set 1. From the solutions we can see that the last four sequences are a single mutation from the actual sequence.

Data Set 2, shown in the Appendix, presented a more challenging problem for the GA. 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 fifth best score when compared with all solutions found. The actual sequence was obtained in some of the runs, but did not survive until the last generation. Another observation is that there is a difference of 13 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.

0.6 Conclusions

Two points deserve further comment. 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) are 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 all three clones. An improved fitness measure is needed to account for fragment matches between three or more clones.

Overall the GA worked well with the data presented to it. Using the correct set of genetic operators was very important to find a GA 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 GA 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.

0.7 Previous Related Work and Scope of Present Work

The DNA restriction fragment map assembly, the subject matter of this paper, 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 a small number of restriction sites also. Krawczak (1988) developed a divide and conquer technique that groups the fragments into compatible clusters and then determines 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 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.

Acknowledgments

Special thanks to Mr. Tom Slezak and Dr. Elbert Branscomb for describing the problem and providing the test data. 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 LLNL Lawrence Livermore National Laboratory and in part by a grant from the Institute of Scientific Computing Research of the Lawrence Livermore National Laboratory.

References

D. Beasley, D. R. Bull, and R. R. Martin, A Sequential Technique for Multi-modal Function Optimization, To be published in *Evolutionary Computation*, February 1993.

R. K. Belew and L. B. Booker, eds., *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, July 1991.

E. Branscomb, T. Slezak, R. Pae, D. Galas, A. V. Carrano, and M. Waterman, Optimizing Restriction Fragment Fingerprinting Methods for Ordering Large Genomic Libraries, *Genomics* 8, 351-366, 1990.

D. J. Cavicchio, Adaptive Search Using Simulated Evolution, Doctoral Dissertation, University of Michigan, Ann Arbor, MI, 1970.

W. Cedeño and V. Vemuri, Dynamic multi-modal function optimization using genetic algorithms, *Proc. of the XVIII Latin-American Informatics Conference*, Las Palmas de Gran Canaria, Spain, August 1992.

W. Cedeño and V. Vemuri, Assembly of DNA Restriction-Fragments Using Genetic Algorithms, Submitted to *Evolutionary Computation.*

W. Cedeño, Genetic algorithms in SISAL to solve the file design problem, *Proc. of the Second SISAL User's Conference*, San Diego, CA, December 1992.

A. J. Cuticchia, J. Arnold, and W. E. Timberlake, The use of simulated annealing in chromosome reconstruction experiments based on binary scoring, *Genetics* 132, 591-601, 1992.

L. Davis, (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, NY, 1991.

K. A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Doctoral dissertation, University of Michigan, *Dissertation Abstracts International* 36(10), 5140B, 1975.

K. Deb and D. E. Goldberg, An investigation of niche and species formation in genetic function optimization, *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, ed., 42-50, Morgan Kaufmann Publishers, San Mateo, CA, June 1989.

J. W. Fickett and M. J. Cinkosky, A genetic algorithm for assembling chromosome physical maps, Unpublished, 1993.

S. Forrest, (Ed.) Proc. Fifth International Conference on Genetic Algorithms, Morgan Kaufman, San Mateo, CA, Aug. 1993.

D. E. Goldberg and J. Richardson, Genetic algorithms with sharing for multimodal function optimization, *Proceedings of the Second International Conference on Genetic Algorithms*, J. J. Grefenstette, ed., 41-49, Lawrence Erlbaum Associates, Hillsdale, NJ, June 1987.

D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

D. E. Goldberg, K. Deb, and J. Horn, Massive multimodality, deception, and genetic algorithms, *Parallel Problem Solving From Nature 2*, Elsevier Science Publishers, 37-46, 1992.

J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

W. Istvanick, A. Kryder, G. Lewandoeski, J. Meidânis, A. Rang, S. Wyman, and D. Joseph, Dynamic methods for fragment assembly in large scale genome sequencing projects, *Proceedings of the Twenty Sixth Annual Hawaii International Conference on System Sciences*: *Architecture and Biotechnology Computing*, T. N. Mudge, V. Milutinovic, and L. Hunter eds. IEEE Computer Society Press, 534-543, Wailea, Hawaii, 1993.

M. Krawczak, Algorithms for the restriction-site mapping of DNA molecules. *Proc. Natl. Acad. Sci. U.S.A.*, 85, 7298-7301, 1988.

S. W. Mahfoud, Crowding and preselection revisited, *Proceedings of Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick, eds., 27-36, Elsevier Science Publishers B. V., 1992.

M. V. Olson, J. W. Dutchik, M. Y. Graham, G. M. Brodeur, C. Helms, M. Frank, M. MacCollin, R. Scheinman, and T. Frank, Random-clone strategy for genomic restriction mapping yeast, *Proc. Natl Acad Sci. U.S.A.*, 83, 7826-7830, 1986.

J. Opatrny, J., The total ordering problem, *SIAM Journal of Computing*, 8(1): 111-114, 1979.

M. D. Platt and T. I. Dix, Construction of restriction maps using a genetic algorithm, *Proceedings of the Twenty Sixth Annual Hawaii International Conference on System Sciences*: *Architecture and Biotechnology Computing*, T. N. Mudge, V. Milutinovic, and L. Hunter eds. IEEE Computer Society Press, 756-762, Wailea, Hawaii, 1993.

J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, June 1989.

R. L. Stallings, D. C. Torney, C. E. Hildebrand, J. L. Longmire, L. L. Deaven, J. H. Jett, N. A. Doggett, and R. K. Moyzis, Physical mapping of human chromosomes by repetitive sequence fingerprinting, *Proc. Natl. Acad. Sci. U.S.A.*, 87, 6218-6222, 1990.

M. Stefik, Inferring DNA structures from segmentation data, *Artificial Intelligence* , 11, 85-114, 1978.

G. Syswerda, Uniform crossover in genetic algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, June 1989.

M. S. Waterman and J. R. Griggs, Internal graphs and maps of DNA, *Bull. Math. Biol.*, 48:189-195, 1986.

D. Whitley, GENITOR: a different genetic algorithm, *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver Colorado, 1988.

D. Whitley, T. Starkweather and D. Fugway, Scheduling problems and travelling salesman: The Genetic Edge Recombination operator, *Proc. Third International Conf. on Genetic Algorithms*. Morgan Kaufmann Publishers, San Mateo, CA, June 1989.

Appendix

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		

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 between the matches

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

In the *multi-niche crowding* (Cedeño and Vemuri, 1992) 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 some diversity. 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, an individual *Ii* is selected for mating. This selection can be either sequential or random. Second, its mate *Im* is selected, not from the entire population, but from a group of individuals of size *Cs* (crowding selection size), picked at random from the population. The mate *Im* thus chosen must be the one who is the most "similar" to *Ii*. The similarity metric used here is not a genotypic metric such as the Hamming distance, but a suitably defined phenotypic distance metric.

Crowding selection promotes mating between members of the same niche while allowing individuals from different niches to mate. Unlike the mating restriction that allows only 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 in each niche.

During the replacement step, MNC uses a replacement policy called *worst among most similar*. 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, *Cf* (crowding factor) groups are created by randomly picking *s* (crowding size) individuals per group from the population. Second, one individual from each group that is most similar to the offspring is identified. This gives *Cf* individuals that are candidates for replacement by virtue of their similarity to the offspring that will replace one of them. From this group of most similar individuals, we pick the one with the lowest fitness to die and that slot is filled with the offspring. Figure 14.8 shows an example of this replacement policy.

Current technological limits are 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.