Modified Uniform Crossover and Desegregation in Genetic Algorithms

Marc Andrew Pawlowsky

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Masters of Computer Science at
Concordia University
Montréal, Québec, Canada

October 1992

ISBN 0-315-84624-0

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Modified Uniform Crossover and Desegregation in Genetic Algorithms

Marc Andrew Pawlowsky

A new crossover operator called modified uniform crossover is described. A theoretical examination of modified uniform crossover, uniform crossover, and n-point shuffle crossover reveals that there is no best crossover operator in all situations. Empirical tests show that neither uniform nor modified uniform crossover is superior in all situations. A new mutation operator called desegregation is described, and is empirically shown to improve the search process.

SOMMAIRE

Un nouveau croisement opérateur qui se nomme croisement uniforme amélioré est décrit. L'examen théorique d'un croisement uniforme amélioré, d'un croisement uniforme, et du n-point d'un croisement mêlé ne permet pas d'affirmer qu'il y ait une meilleure méthode de croisement opérationnel pour toutes les situations. Les épreuves empiriques indiquent que ni le croisement uniforme, ni le croisement uniforme amélioré ne sont supérieures pour toutes les situations. Un nouvel opérateur de mutation appelé deségrégation est décrit et est démontré de façon empirique afin d'améliorer le processus de la recherche.

Chapter 1

Introduction

This work will examine shuffle crossover operators for genetic algorithms. Specifically, we will be analyzing the theoretical aspects of the operators, and making an empirical comparison between two of the more extreme operators.

We will be discussing a new crossover operator called *modified uniform crossover*. the operator has a much smaller variance and is therefore more uniform than tradition uniform crossover [1]

The main contribution of this work is that there is no one best shuffling crossover operator in all situations. This startling conclusion greatly changes the search for the best crossover operator. The choice of which crossover operator to use is now much more a heuristic choice than ever before. unfortunately we are not able to provide any guide on which crossover operator should be used.

Chapter 2 will introduce the basic concepts of Genetic Algorithms (GA's). Even if you are familiar with GA's, you should read it as the chapter presents the algorithms for the crossover operators that are used and analyzed in later chapters. A new mutation operator called *desegregation* is also described in the chapter.

Chapter 3 will present formulas for calculating the probabilities of the number

[1] Modified uniform crossover was previously published under the name improved uniform crossover by Pawlowsky and Kryzyak [15]

of genes copied and passed, as well as the chance of a schema surviving and being combined. Realizing that most readers skip formal proofs, they have been grouped together in chapter 4. The conclusion mentioned above is at the end of this chapter

Chapter 5 describes the experiments that were performed, in the empirical part of this work Chapter 6 examines the benefits of desegregation. Chapter 7 compares modified uniform crossover with normal uniform crossover.

Chapter 2

What is a Genetic Algorithm

This chapter will introduce the reader to the basic ideas and techniques in Genetic Algorithms. The ideas which are presented here will not be examined in any detail: instead, just enough material to understand future chapters and answer a few of the simpler questions will be brought up in this chapter. We recommend Goldberg[9] for a full introduction on GA's.

Genetic Algorithms are heuristic techniques modelled on nature which are used to find the global maximum values of functions with very little *a priori* information. The basic principle involves treating a possible solution vector as a chromosome, with each entry in the vector being an allele [1], and applying operators that are similar to the ones nature uses to a set of these chromosomes, called the *population*. Unlike other popular optimization techniques (e.g. Hill Climbing, Simulated Annealing), which work with one solution at a time, GA's examine more than one possible solution (chromosome) at any one time. This allows GA's to be implemented on massively parallel machines [3][13][16][26]

There are three common operators in GA's: reproduction, crossover, and mutation.

[1] For our purposes an allele is a character whose value and property is set

Parent A	⟨	1	1	1	1	1	⟩
Parent B	⟨	0	0	0	0	0	⟩
Child	⟨	1	1	0	0	0	⟩

Figure 2.1: A possible crossover.

The *reproduction* operator is responsible for selecting the members of the population that will reproduce, and which members of the population will be killed. The reproduction operator itself does not perform any searching. Instead its contribution to the search process is the selection of sucessful chromosomes that will donate genetic material via the crossover operator. As in nature, we select the fittest chromosomes from the population for reproduction and remove (kill) the weakest. The *fitness* of a chromosome is a nondecreasing function based on its *strength*, which is the evaluation of the chromosome through the function to be optimized. The chromosomes which are chosen by the reproduction operator are called *parents*.

Crossover is the main search method of GA's. To produce a new chromosome we choose two parents from the population based on their fitness. A child is then produced by taking some alleles from each parent in a random manner. An example is given in Figure 2.1.

The final common operator is *mutation*. The role of mutation is to ensure that no possible value is left out from the search by premature convergence. Convergence of a population means that the *chromosomes* (solution vectors) in the population have similar values. Mutation changes the values of the genes in the population at random, to ensure that all possible combinations of alleles can be generated by crossover. The purpose of mutation is not to search the solution space, therefore it is not applied too often. Typically mutation is applied *once* per thousand gene transfers between parents and children during crossover. Figure 2.2 gives an example population that would never explore the solution space where a 1 is in the third place if only the reproduction and crossover operators were used.

$$\begin{array}{c}
\langle\ 0\ 1\ 0\ 0\ 1\ \rangle \\
\langle\ 1\ 0\ 0\ 1\ 0\ \rangle \\
\langle\ 0\ 0\ 0\ 1\ 1\ \rangle \\
\langle\ 0\ 0\ 0\ 0\ 0\ \rangle
\end{array}$$

Figure 2.2: A population where crossover will not explore all values.

The examples given in Figures 2.1 and 2.2 use binary vectors: Holland [11] has shown that GA's work best when the search space is binary. Ackley [4] mentions in his introduction that GA's operate by trying to find the best coordinates and combining them. We can maximize the number of coordinates that are searched at any one time, and minimize the possible range of each coordinate by using binary vectors. In order to evaluate the strength of a chromosome a function ($\mathcal{G}$) is required to translate the chromosome to the domain of the evaluation function ($\mathcal{F}$). This makes the strength of the i-th chromosome (c_i) equal to $\mathcal{F}[\mathcal{G}(c_i)]$. [2]

Figure 2.3 gives a basic algorithm for GA's. The algorithm as listed introduces as many new questions as it answers. First, what domains and with what precision should the solution space be explored. This is the only *a priori* information that a GA needs. GA's cannot guarantee to find the maximum value, but they will find the area of a maximum value.[3]

As mentioned above, the fitness rating is a measurement of how many offspring a chromosome will produce. The second question arising from Figure 2.3 is how to determine the fitness rating of a chromosome. The fitness can be set in many ways [10], as long as

$$\text{fitness}(c_i) \geq 0$$

[2] In the rest of this work, chromosomes will be written without the brackets indicating that they are vectors, and without spaces to separate the elements.

[3] A spike could always exist in an areas that was not searched, for any "black box" non-exhaustive search technique. See Figure 2.4. A tradeoff must be made between the precision of the answer received (realizing that it is only an estimate) and execution time (see below). This issue has just recently been covered in a technical report by Schraudolph and Belew [24] who dynamically change the mapping function to get higher precision with shorter chromosome lengths.

```
1. Decide on the domains, and precision of the solution space that is to be explored.
2. Find a mapping scheme that maps the solution space to a binary vector. (This will give the length of the chromosome.)
3. Decide on a population size.
4. Create an initial population with random values.
5. REPEAT
    (a) Find the fitness of the population members
    (b) Choose n members of the population to become parents
    (c) Create the children from the parents using crossover
    (d) Give each gene in the children a chance to be mutated
    (e) Place the children into the population, replacing old members based on fitness.(Least fit are most likely to be replaced.)
6. UNTIL stopping conditions met.
```

Figure 2.3: Basic Genetic Algorithm

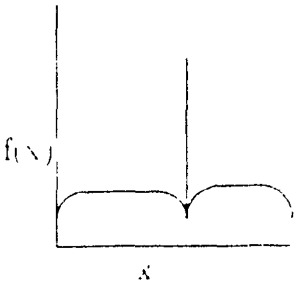


Figure 2.4: A hard function to solve
Due to the spike, this function is hard for GA's to solve

chromosome	strength	rank	fitness
c_1	22	1	2
c_2	18	2	1
c_3	18	2	1
c_4	10	3	0.5
c_5	6	4	0.25

Figure 2.5: Possible assignment of fitness by ranking.

$$\forall c_i, c_j \in \text{population}. \text{strength}(c_i) > \text{strength}(c_j) \rightarrow \text{fitness}(c_i) > \text{fitness}(c_j).$$

A common method is to use the normalization function

$$\text{fitness}(c_i) = \begin{cases} \sum_j \frac{\text{strength}(c_i)}{\text{strength}(c_j)} & \text{if strength}(c_i) \geq \beta \\ 0 & \text{otherwise} \end{cases}$$

The value of β can be chosen in many ways. Two common methods are to set β to 0, or to the mean of the population.

A second way of setting the fitness of a chromosome is to rank the population by strength, and then assign each member whose ranking is lower than the previous member a fraction of the fitness of the previous member. In Figure 2.5 the strongest members of the population have a fitness of two, and each lower ranking member of the population has half the fitness of the next highest member.

Ranking has many advantages, among them that partially ordered sets may be optimized[23] [19][4], and that it is harder for extremely strong members of the population to dominate causing premature convergence.

The fitness rating of a chromosome is the expected value of the number of children that it will be parent of. Since the fitness rating is a positive real number, a method must be found to transform it into a natural number.

An early technique for choosing the parents which would reproduce is called the "Spinning Wheel"[9] . The model is of a roulette wheel, such that each chromosome

[4] See [5] [10], [27] for the advantages of ranking

```
1. parents ← number of parents to choose from population.
2. sum ← Sum of fitness for all members of the population.
3. total ← 0 /* Partial total of sum fitness */
4. chosen ← 0 /* Number of parents chosen */
5. Make position of chromosomes in population random.
6. FOR each member $i$ of the population DO
   (a) REPEAT
       i. total ← total + (fitness of chromosome$_i$ / sum) × parents
       ii. IF chosen < total THEN
           A. chromosome$_i$ will receive another (possibly first) offspring
           B. chosen ← chosen + 1
   (b) UNTIL total > chosen
```

Figure 2.6: SUS algorithm

in the population occupies an area in proportion to its fitness level. Each time a parent is needed the wheel is spun, and the parent is chosen by where the ball lands.

A newer method is Stochastic Uniform Selection (SUS) by Baker [6]. In SUS what we do is to iterate through the population while keeping a cumulative total of the number of parents chosen and the sum of the fitness ratings. When the number of parents chosen is less than the sum of the fitness ratings, then that parent is selected to reproduce. Figure 2.6 [6] gives the algorithm in more detail. Advantages of SUS include that the number of times a parent is chosen is never more than the ceiling of the fitness, and that it is of order population size [6].

The next question to be asked of Figure 2.3 is the heart of this thesis, how is crossover performed. There are three basic techniques, plus a new method that we will be introducing called *modified uniform crossover* (u^+). There are many other published variations that will not be mentioned (see [8], [11], [12], [17], [21]). The algorithms that we are giving here are adaptations such that crossover will produce

```
1. Ensure $1 \le n \le l - 2$
2. X is a set of $n$ integers in the range $[2, l - 1]$ chosen randomly
3. choose ← random choice between 1 and 2
   /* parent to have first gene copied from */
4. FOR i=1 TO l
   (a) IF $i \in X$ THEN
       i. /* Switch parent to copy from */
       ii IF choose = 1 THEN
            A. choose ← 2
       iii. ELSE
            A. choose ← 1
   (b) Copy the i-th allele from $parent_{choose}$ to the childs i-th gene.
```

Figure 2.7: Algorithm for n-point crossover (n-pt).

one child. The basic algorithms usually produce two children, with the genes being those of the opposite parent.[5]

The first technique is known as n-point crossover (n-pt) . Given a chromosome of length l, we choose n distinct points, called crossover points (x_i), in the range 2 to $l - 1$ Starting with a parent chosen at random, copy the alleles into the child until a crossover point is reach, at which point the parent is changed. This is continued until the end of the chromosome is reached (see Figure 2.7).

An example is given in Figure 2.8 with a chromosome length of 5, two crossover points at index 1 and 2 In the example, the first (locus 1) and the third fragments (locus 4 and 5) are copied from the first parent. The second fragment (locus 3 and 4) is copied from the second parent.

The second technique is n-point shuffle crossover (sh-n) . In this technique the

[5] For a crossover that produces two children, if the i-th gene in the first child was copied from the first parent, then the i-th gene in the second child comes from the second parent Simmilarly if the j-th gene in the first child comes from the second parent, then the j-th gene in the second child comes from the first parent

Parent A	1	0	1	0	1
Parent B	0	1	1	1	0
Child	1	1	1	0	1

Figure 2.8: An example of 2-pt crossover.

Before shuffling

Parent A	$a_1 = 1$	$a_2 = 0$	$a_3 = 1$	$a_4 = 0$	$a_5 = 1$
Parent B	$b_1 = 0$	$b_2 = 1$	$b_3 = 1$	$b_4 = 1$	$b_5 = 0$

After shuffling

Parent A	$a_2 = 0$	$a_4 = 0$	$a_1 = 1$	$a_3 = 1$	$a_5 = 1$
Parent B	$b_2 = 1$	$b_4 = 1$	$b_1 = 0$	$b_3 = 1$	$b_5 = 0$

Crossover

Parent A	$a_2 = 0$	$a_4 = 0$	$a_1 = 1$	$a_3 = 1$	$a_5 = 1$
Parent B	$b_2 = 1$	$b_4 = 1$	$b_1 = 0$	$b_3 = 1$	$b_5 = 0$
Child	$a_2 = 0$	$b_4 = 1$	$b_1 = 0$	$a_3 = 1$	$a5 = 1$

Unshuffling

Child	$b_1 = 0$	$a_2 = 0$	$a_3 = 1$	$b_4 = 1$	$a_5 = 1$

Figure 2.9: Example of n-point shuffle crossover

Parent A	a_1	a_2	a_3	a_4	a_5	a_6
Parent B	b_1	b_2	b_3	b_4	b_5	b_6
Child	b_1	a_2	b_3	a_4	b_5	a_6

Figure 2.10: Example uniform crossover.

Copying of duplicate genes.

Parent A	$a_1 = 1$	$a_2 = 0$	$a_3 = 1$	$a_4 = 0$	$a_5 = 1$
Parent B	$b_1 = 0$	$b_2 = 1$	$b_3 = 1$	$b_4 = 1$	$b_5 = 0$
Child			1		

Copying of non-duplicate genes.

Parent A	$a_1 = 1$	$a_2 = 0$	$a_3 = 1$	$a_4 = 0$	$a_5 = 1$
Parent B	$b_1 = 0$	$b_2 = 1$	$b_3 = 1$	$b_4 = 1$	$b_5 = 0$
Child	$b_1 = 0$	$a_2 = 0$	1	$a_4 = 0$	$b_5 = 0$

Figure 2.11 Example modified uniform crossover.

parents are first shuffled such that they have the same genes in the same positions, but that the position of the genes inside the chromosome is random. n-pt crossover is then performed normally. Finally, the genes in the parents and the child are unshuffled.[5] (See Figure 2.9)

Uniform crossover has each parent copy $l/2$ unique genes chosen independently (see Figure 2.10).[25] Modified uniform crossover first copies the genes that have the same alleles in both parents to the child. Of the genes that have different alleles in the parents, half are copied from each parent, being chosen independently once more (see Figure 2.11) Only modified uniform crossover guarantees that if there are two alleles which are different then the crossover will be effective.[6]

The final question to be answered of Figure 2.3 is how do we know when to stop. The answer is the same as any black box method, "we do not know!" There is no guarantee that we will find the optimal solution, so we can not know when to stop.

[6] An effective crossover is one in which the child is different from both its parents

****	1**0	1010	*010
1***	101*	*0**	**1*
10**	10*0	*01*	**10
1*1*	1*10	*0*0	***0

Figure 2.12: Schema for the chromosome 1010

What is done instead is to set up an artificial stopping condition. The most common method mentioned in literature is to stop after a preset number of generations. Two other alternatives would be to stop after the population has converged to a certain level, or to stop after the populations strength has not increased after a preset number of generations.

The final subject in this chapter is "Why do GA's work?" The proof is complex and is given by Holland [11], Goldberg[9], and is re-examined by Grefenstette[10]. The essence of the proof is that we are not exploring individual chromosomes but the *schema* to which they belong.

A schema is a projection of the hyperspace that a chromosome belongs to. We will let * represent a position that is not set. The chromosome 1010 belongs to the 16 schemata[7] listed in Figure 2.12. In general a chromosome of length l belongs to 2^l different schemata.

What we are doing when we allocate chromosomes for reproduction is to allocate more search effort on the schemata that have been successful in the past (as is evidenced by the fact that they are still in the population). Since reproduction removes members which have a low fitness ratings the number of samples on "bad" schemata will be decreased in the next generation, while "good" schemata will have more samples allocated to them, thus pressuring the population and the search process to continually *improve. The purpose of crossover is to mix "good" schemata so* that they can generate better chromosomes for the next generation.

This concludes our very brief introduction to GA's. The reader now has enough

[7] The plural of schema is schemata.

material to implement the basic GA in Figure 2.3. and understand the remainder of this thesis.

Chapter 3

Theoretical Foundations

In this chapter we will discover formulas that will predict the number of genes copied by a parent to the child for shuffle (sh-n), uniform (u) and modified uniform crossover (u^+). We will introduce a new term *passed* indicating the number of genes that have the same allele in the child as in the parent, and we will derive formulas for predicting the number of alleles passed for the crossovers mentioned above. Finally we will give formulas that will predict the schema survival and combination rates for the three crossover methods.

3.1 Number of Genes Copied

A simple analysis of n-point shuffle crossover will show that as n, $1 \leq n \leq l-1$, increases the variance of the number of genes that a parent can contribute decreases. We define crossover point 0 to be just before the first locus, and the last crossover point $n+1$ to be after the last locus. n is the number of crossover points within the chromosome.

A chromosome fragment is the set of genes in a chromosome copied to the child, between two consecutive crossover points. Let a_m be the number of genes in the $2m-1$-th fragment, and b_m the number of genes in the $2m$-th fragment. The

	$\bar{a}_1 = 2$		$\bar{b}_1 = 3$			$\bar{a}_2 = 3$			$\bar{b}_2 = 2$		$\bar{a}_3 = 1$	$\bar{b}_3 = 4$			
Parent A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
Parent B	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
Child	a_1	a_2	b_3	b_4	b_5	a_6	a_7	a_8	b_9	b_{10}	a_{11}	b_{12}	b_{13}	b_{14}	b_{15}

Figure 3.1: Example of calculating $\bar{a}_m$ and $\bar{b}_m$ for 5-point crossover.

means that $\bar{a}_m$ and $\bar{b}_m$ are the number of genes from the m^{th} fragment that belongs to the first and second parent respectively. (i.e. The fragments are numbered: $\bar{a}_1, \bar{b}_1, \bar{a}_2, \bar{b}_2, \bar{a}_3, \bar{b}_3, \ldots$. See figure 3.1)

$\bar{a}_m$ is the number of genes that come after the $(2m-2)$-th crossover point and before the $2m-1$-th crossover point. Similarly $\bar{b}_m$ is the number of genes between the $2m-1$-th and $2m$-th crossover points.

After shuffling the genes we have the two following models of the child after crossover, where n is the number of crossover points, and k is a function of n:

For $n = 2k-1$

$\bar{a}_1$	$\bar{b}_1$	$\bar{a}_2$	$\bar{b}_2$	$\ldots$	$\bar{a}_k$	$\bar{b}_k$

giving $2k$ fragments.

1. $1 \leq \bar{a}_m \leq l-(2k)+1$, for $1 \leq m \leq k$

2. $1 \leq \bar{b}_m \leq l-(2k)+1$, for $1 \leq m \leq k$

3. $\sum_{m=1}^{k} \bar{a}_m = c$

4. $\sum_{m=1}^{k} \bar{b}_m = l-c$

5. $k \leq c \leq l-k$

For $n = 2(k-1) = 2k-2$

$\bar{a}_1$	$\bar{b}_1$	$\bar{a}_2$	$\bar{b}_2$	$\ldots$	$\bar{a}_{k-1}$	$\bar{b}_{k-1}$	$\bar{a}_k$

giving $2k-1$ fragments.

1. $1 \leq \bar{a}_m \leq l-(2k-1)+1 = l-2k+2$, for $1 \leq m \leq k$

2. $1 \leq \bar{b}_m \leq l-(2k-1)+1 = l-2k+2$, for $1 \leq m \leq k-1$

3. $\sum_{m=1}^{k} \bar{a}_m = c$

4. $\sum_{m=1}^{k-1} \bar{b}_m = l-c$

5. $k \leq c \leq l-(k-1) = l-k+1$

Lines *1* and *2* of the models state that each box must have at least one gene. Therefore the maximum number of genes any box can have is the length of the chromosome (l), less the number of segments, plus one for the box itself e.g place one gene in each fragment, then place the remainder in a single box.

Lines *3* and *4* indicate that the number of genes copied from the first parent is an arbitrary number c, with the second parent getting the remainder $(l-c)$

The final line, *5*, indicates that the minimum number of genes that can be copied from the first parent is equal to the number of fragments (k) that will be copied to the child. The maximum number of genes that could be copied from the parent is the number of genes less the number of fragments that will be copied from the second parent.

Note that for both cases the number of fragments is one more than the number of crossover points. This means that there are $n-k+1$ fragments that will be copied from the second parent. The two models will now be merged so that one model is used for both even and odd n.

Using $\bar{a}'_m = \bar{a}_m - 1$, and $\bar{b}'_m = \bar{b}_m - 1$ which represent the number of genes in each fragment above one, and $c' = c-k$ which represents the number of genes the first parent has copied in addition to k. The summation index on line *4* needs also to be changed to become a function of k

1. $0 \leq \bar{a}'_m \leq l-(n+1) = l-n-1$, for $1 \leq m \leq k$

2. $0 \leq b'_m \leq l-(n+1) = l-n-1$ for $1 \leq m \leq n-k+1$

3. $\sum_{m=1}^{k} \bar{a}'_m = c'$

4. $\sum_{m=1}^{n-k+1} \bar{b}'_m = (l-(n+1)) - c' = l - n - c' - 1$

5. $0 \leq c' \leq l - (n+1) = l - n - 1$

There are

$$\begin{pmatrix} c' + k - 1 \\ c' \end{pmatrix}$$

ways that the $\bar{a}'_m$'s can be selected and sum to c' [18], and

$$\begin{pmatrix} (l - n - c' - 1) + (n - k + 1) - 1 \\ l - n - c' - 1 \end{pmatrix} = \begin{pmatrix} l - k - c' - 1 \\ l - n - c' - 1 \end{pmatrix}$$

ways the j_l's can be chosen so their sum is $l - n - c' - 1$.

For example, if 3-point crossover is being performed, then there are four chromosome fragments, two of which will be copied by the first parent. The number of ways, the number of genes in each chromosome fragment copied by the first parent, can be chosen such that their sum is equal to five is calculated as follows.

$n = 3, c = 5$

$k = 2 \rightarrow c' = 3$

$$\begin{pmatrix} 3 + 2 - 1 \\ 3 \end{pmatrix} = 4$$

These are

$\bar{a}_1$	$\bar{a}_2$
1	4
2	3
3	2
4	1

The probability of choosing $c = c' + k$ genes from the first parent to be copied to the child for n-point shuffle crossover is the number of ways of choosing the a'_m's so their sum is c' multiplied by the number of ways of choosing the b'_m's so their sum equals $l - n - c' + 1$, divided by the total number of ways of choosing the a'_m's and $\bar{b}'_m$'s.

$$p_{c_1}^{sh-n}(c) = \frac{\begin{pmatrix} c' + k - 1 \\ c' \end{pmatrix} \begin{pmatrix} l - k - c' - 1 \\ l - n - c' - 1 \end{pmatrix}}{\sum_{\bar{c}'=0}^{l-n-1} \begin{pmatrix} \bar{c}' + k - 1 \\ c' \end{pmatrix} \begin{pmatrix} l - k - c' - 1 \\ l - n - c' - 1 \end{pmatrix}}$$

Substituting $c - k$ for c'

$$p_{c_1}^{sh-n}(c) = \frac{\begin{pmatrix} (c - k) + k - 1 \\ (c - k) \end{pmatrix} \begin{pmatrix} l - k - (c - k) - 1 \\ l - n - (c - k) - 1 \end{pmatrix}}{\sum_{\bar{c}'=0}^{l-n-1} \begin{pmatrix} \bar{c}' + k - 1 \\ c' \end{pmatrix} \begin{pmatrix} l - k - \bar{c}' - 1 \\ l - n - c' - 1 \end{pmatrix}}$$

$$= \frac{\begin{pmatrix} c - 1 \\ c - k \end{pmatrix} \begin{pmatrix} l - c - 1 \\ l - n - c + k - 1 \end{pmatrix}}{\sum_{\bar{c}'=0}^{l-n-1} \begin{pmatrix} \bar{c}' + k - 1 \\ c' \end{pmatrix} \begin{pmatrix} l - k - c' - 1 \\ l - n - \bar{c}' - 1 \end{pmatrix}}$$

The probability of copying c genes from the second parent is the dual of copying $l - c$ genes from the first parent [1]

$$p_{c_2}^{sh-n}(c) = p_{c_1}^{sh-n}(l - c)$$

$$= \frac{\begin{pmatrix} (l - c) - 1 \\ (l - c) - k \end{pmatrix} \begin{pmatrix} l - (l - c) - 1 \\ l - n - (l - c) + k - 1 \end{pmatrix}}{\sum_{\bar{c}'=0}^{l-n-1} \begin{pmatrix} \bar{c}' + k - 1 \\ \bar{c}' \end{pmatrix} \begin{pmatrix} l - k - \bar{c}' - 1 \\ l - n - c' - 1 \end{pmatrix}}$$

[1] Remember the total number of genes copied from both parents is l

$$= \frac{\binom{l-c-1}{l-c-k}\binom{c-1}{c-n+k-1}}{\sum_{c'=0}^{l-n-1}\binom{c'+k-1}{c'}\binom{l-k-c'-1}{l-n-c'-1}} \tag{3.1}$$

Assuming that the chromosome can be assigned to either parent with equal probability, the probability of a parent having c genes copied by shuffle crossover is

$$p_c^{sh-n}(c) = \frac{1}{2}\left(p_{c_1}^{sh-n}(c) + p_{c_2}^{sh-n}(c)\right)$$

$$= \frac{\binom{c-1}{c-k}\binom{l-c-1}{l-n-c+k-1} + \binom{l-c-1}{l-c-k}\binom{c-1}{c-n+k-1}}{2\sum_{c'=0}^{l-n-1}\binom{c'+k-1}{c'}\binom{l-k-c'-1}{l-n-c'-1}}$$

Using the identities

$$\binom{n+p}{m} = \sum_{k=0}^{\infty}\binom{n-k}{m-k}\binom{p+k-1}{k}$$

and

$$\binom{a}{b} = \binom{a}{a-b}$$

we get the probability of having a parent copy c genes by n-point shuffle crossover

$$p_c^{sh-n}(c) = \frac{\binom{c-1}{c-k}\binom{l-c-1}{n-k} + \binom{l-c-1}{k-1}\binom{c-1}{n-k}}{2\binom{l-1}{l-n-1}} \tag{3.2}$$

We will now analyse the first parent, chosen arbitrarily, for variance. Choosing the a'_m's for a given c' is the same as rolling k dice. The probabilities for choosing c' are bell-shaped. As k increases the curve becomes narrower and narrower, reducing

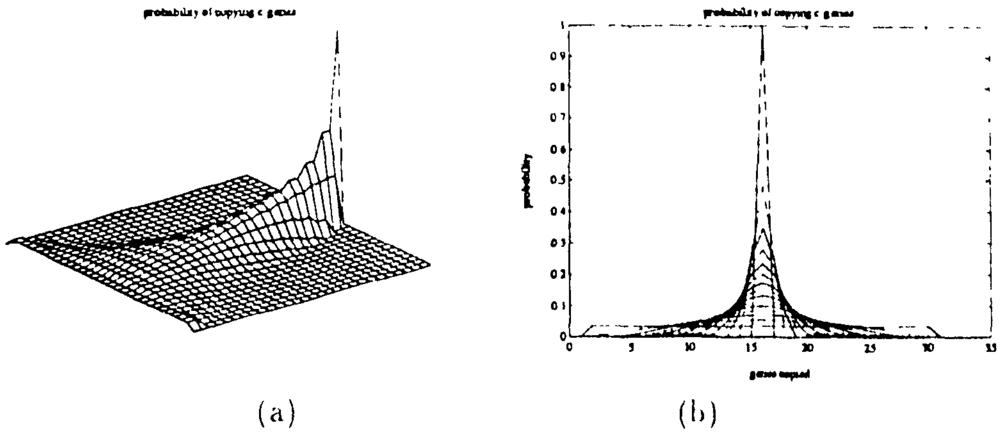


Figure 3.2: Probability of copying c genes for n-point shuffle crossover. Probability of copying c genes for n-point shuffle crossover. Where the probability is calculated using equation 3.2. Chromosome length is 30. (a) The upper left to lower right axis is the number of genes copied, and goes from 0 to 30 The lower left to upper right axis is the number of crossover points and goes from 1 to 29 The vertical axis is the probability of copying c genes with n-point shuffle crossover and has a range of 0 to 1. (b) Each curve in the plot represents a number of crossover points. The curves have higher peaks as the number of crossover points increases

the variance of the sum.[2] This can be seen in figure 3.2 which graphs equation (3.2), for $l = 30$ and $1 \le k \le l - 1$. Eshelman, Caruana, and Schaffer [8] have noticed that up to a ce tain point, the greater the number of crossover points for shuffle crossover the better the empirical results. Along with the results shown above, one would be led to believe that it would be best to keep the variance down to a minimum

As mentioned on page 2, uniform crossover produces a child by taking half of its genes from each parent. The choice of which genes are from which parent are independent, as long as no locus is chosen twice, and each parent contributes half This is analogous to $sh - (n - 1)$ crossover The probability of c genes being copied from a parent with uniform crossover is

[2] Gygax Gary Advanced Dungeon & Dragons Dungeon Masters Guide TSR Games 1979

$$
\begin{aligned}
p_c^u(c) &= 1 \text{ if } c = l/2 \\
&= 1/2 \text{ if } \lfloor l/2 \rfloor \leq c \leq \lceil l/2 \rceil \\
&= 0 \text{ otherwise}
\end{aligned}
\tag{3.3}
$$

Modified uniform (u^+) crossover produces children by first passing on all the genes that are the same in both parents to the child. Each parent then fills in half of the remaining genes. If the number of duplicate genes d $(0 \leq d \leq l)$ have the same allele, then the first parent contributes $d + \lfloor \frac{l-d}{2} \rfloor$ genes, and the second $d + \lceil \frac{l-d}{2} \rceil$ genes. The probability of a parent copying c genes is

$$
\begin{aligned}
p_c^{u^+}(c) &= 1 \text{ if } c = \frac{(l+d)}{2} \text{ and } l+d \text{ is even} \\
&= 1/2 \text{ if } \lfloor \frac{(l+d)}{2} \rfloor \leq c \leq \lceil (l+d)/2 \rceil \\
&= 0 \text{ otherwise}
\end{aligned}
\tag{3.4}
$$

Note that the duplicate genes are copied twice, once for each parent.

3.2 Alleles Passed

Syswerda [25] uses masks to analyze the survival of schemas. We will be using the same formula, to analyze the number of genes that are passed, which will lead into the analysis of schema survival, and combination.

A mask is a method of showing where the gene for a child came from. In Syswerda's work crossover produces two children. A mask is a binary string of length l. A one in position i indicates that the i-th gene was copied from the first parent into the first child, and the second parent copied its i-th gene into the second child. If the mask was a zero, then the second parent would have copied its gene

Parent A	1	1	1	1	0	0	0	0
Parent B	1	1	0	0	1	1	0	0
Child 1	1	1	1	0	0	1	0	0
Child 2	1	1	0	1	1	0	0	0
Mask	1	0	1	0	1	0	1	0

Figure 3.3: Example of crossover with masks.

into the first child: and the first parent would have copied its gene into the second child.

Figure 3.3 shows a case where the first, third, and fifth genes from the first parent where copied to the first child. The second, fourth, and sixth genes of the second parent were copied to the first child. The opposite occurs for the second child.

Further analysis will make use of the following principle:

> Schema A (a schema in the first parent) can have from zero to all of its bits masked. How many ways, given that m bits of l are masked, can schema A have i of its bits masked? The answer is the number of ways to arrange the i bits within m times the number of ways to arrange $b_A - i$ within $l - m$, or
>
> $$\binom{m}{i}\binom{l-m}{b_A - i} \tag{3.5}$$
>
> [25].

Recall that b_A is the number of bits in a schema that are fixed, i.e. either a 0 or 1. A bit in the schema is said to be masked if the bit in the schema is fixed, and if the corresponding bit in the mask is a 1

The general case is that given sets A, B such that $B \subseteq A$ the probability of receiving b items in B when picking a from A is

$$\frac{\begin{pmatrix} a \\ b \end{pmatrix}\begin{pmatrix} |A|-a \\ |B|-b \end{pmatrix}}{\sum_{b'=0}^{|B|}\begin{pmatrix} a \\ b' \end{pmatrix}\begin{pmatrix} |A|-a \\ |B|-b' \end{pmatrix}}$$

$$= \frac{\begin{pmatrix} a \\ b \end{pmatrix}\begin{pmatrix} |A|-a \\ |B|-b \end{pmatrix}}{\begin{pmatrix} |A| \\ |B| \end{pmatrix}} \tag{3.6}$$

We have defined d to be the number of genes with the same alleles in the two parents. The number of genes copied to the child from each parent is represented by the term c. A new term "passed" indicates how many alleles from a parent are in a child, regardless of which parent the allele came from.[3]

A gene is passed from the parent if it is copied from the parent, or if the gene that was copied from the other parent has the same allele.

In mathematical terms let $\mathcal{P}_A$ represent the genes passed by the first parent, $\mathcal{C}_A$ the genes that are copied by the first parent, and $\mathcal{D}$ be the set of genes that have the same alleles. Then

$$\mathcal{P}_A = \mathcal{C}_A \bigcup \mathcal{D} \tag{3.7}$$

Note that the genes which are copied, and the duplicate genes are not necessarily disjoint sets.

The number of alleles passed is indicated by the term p. (From the definition of passed genes $p \geq c$.)

Recall that the first parent copies c genes, while the second parent copies $l - c$ genes. For a crossover operator which does not check the alleles of the parents

[3] A gene that is passed, is one that can be claimed by a parent. For example if a new born baby has black hair, and both its mother and father have black hair, then the mother could claim that the baby has her hair color. This is true even if unknown to her, the gene actually came from the father

Parent A	1	0	1	0	1	1
Parent B	0	1	1	1	0	1
Child	1	0	1	1	0	1
$\mathcal{C}_{\mathcal{A}}$	X	X	X			
$\mathcal{P}_{\mathcal{A}}$	X	X	X			X
$\mathcal{C}_{\mathcal{B}}$				X	X	X
$\mathcal{P}_{\mathcal{B}}$			X	X	X	X

Figure 3.4: Example of genes passed.

(shuffle and uniform crossover). the probability of passing p genes from the first parent, is the probability of the second parent copying $p - c$ duplicate genes.

$$p_p(p) = \sum_{c=0}^{l} p_c(c) \frac{\binom{l-c}{p-c}\binom{l-(l-c)}{d-(p-c)}}{\binom{l}{d}}$$

$$= \sum_{c=0}^{l} p_c(c) \frac{\binom{l-c}{p-c}\binom{c}{p-d}}{\binom{l}{d}} \tag{3.8}$$

From the definition of modified uniform crossover given in chapter 2, $p_p^{u+} = p_c^{u+}$ since no duplicate genomes can be passed from the second parent, that were not copied from the first parent.

3.3 Schema Passed

A schema is passed from a parent if all its fixed positions occur in the child. (See figures 3.4. and 3.5) When all the fixed positions have been passed the schema is also said to have "survived".

Parent A	1	0	1	0	1	1
Parent B	0	1	1	1	0	1
Child	1	0	1	1	0	1

Schema passed from first parent			
101**1	101***	10***1	1*1**1
01**1	10****	1*1***	*01***
1****1	*0***1	**1**1	1*****
0****	**1***	*****1	******

Schema passed from second parent			
**1101	**110*	**11*0	**1*01
**101	**11**	**1*0*	***10*
*1**1	***1*1	****01	*****1
***0*	***1**	**1***	******

Figure 3.5: Example of schema passed.

To calculate the probability of passing a schema of length b_A from the first parent to the child, we use equation (3.6) to find the probability of a b_A specific genes being passed when p out of l genes are passed. (Let $|A| = l, a = p, |B| = b_A, b = b_A$.)

$$p_s(b_A) = \sum_{p=0}^{l} \frac{p_p(p) \begin{pmatrix} p \\ b_A \end{pmatrix} \begin{pmatrix} l - p \\ b_A - b_A \end{pmatrix}}{\begin{pmatrix} l \\ b_A \end{pmatrix}}$$

Simplifying the formula we get

$$p_s(b_A) = \sum_{p=0}^{l} \frac{p_p(p) \begin{pmatrix} p \\ b_A \end{pmatrix}}{\begin{pmatrix} l \\ b_A \end{pmatrix}} \tag{3.9}$$

3.4 Schemata Combining

The last formula in this chapter covers the probability of schemas combining. Two schemata are said to have combined, if the first schema is passed to the child from one parent, while the second schema is passed from the other parent. For the example given in figure 3.5 there are 256 (16 × 16) different schema combinations, one of which is 101**1 and **1101.

Using equation (3.9) we know the probability of a schema being passed. Given two schemas of length b_A and b_B, and the number of genes passed p and $l + d - p$, from the first and second parents respectively, the probability of the two schemas appearing in the child is calculated as

$$p_{sc}(b_A, b_B) = \sum_{p=0}^{l} p_t(p) \frac{\binom{p}{b_A}\binom{l+d-p}{b_B}}{\binom{l}{b_A}\binom{l}{b_B}} \qquad (3.10)$$

3.5 Summary

We have now provided formulas so shuffle, uniform, and modified uniform crossover can now be theoretically compared. These formulas are important, in that they consider any level of convergence between the two parents. In chapter 4 we will use these formulas to examine the relative theoretical strengths and weaknesses of the operators.

Chapter 4

Theoretical Analysis

This chapter will examine various theoretical advantages and disadvantages of shuffle, uniform and modified uniform crossover. We will prove that neither uniform, modified uniform, nor shuffle crossover is superior to each other for all situations. This means that the performance of a crossover operator is problem dependent.

The mathematical symbols used are the same as in the previous chapters, all new symbols will be defined when they are first used. A summary of the symbols used is provided in appendix A. It is recommended that the reader use the appendix as a reference for the many symbols used in this chapter.

Before we can reach our conclusions, we must first create a set of lemmas which will be used in deriving our final theorems.

lema 1 *The total number of genes passed from both parents is $l + d$, where l is the length of the chromosome, and d is number of alleles that occur in both parents.*

Proof: Let c be the number of genes copied from the first parent. Let d_1 and $d - d_1$ be the number of duplicate genes that were copied from the first and second parents respectively. The number of genes passed by the first parent is the number of genes copied, plus the number of duplicate alleles copied by the second parent $d - d_1$, giving a total of $c + (d - d_1)$ genes passed by the first parent.

parent 1	1	1	1	1	1
parent 2	0	0	1	1	1
child	0	1	1	1	1

$$d = 3, \quad p_1 = 4, \quad p_2 = 4, \quad l = 5$$

$$l + d = p_1 + p_2$$

$$5 + 3 = 4 + 4$$

Figure 4.1: Example of lema 1

The second parent passes the number of genes it copies $l - c$, and the number of duplicate genes copied by the first parent, giving a total of $(l - c) + d_1$ genes passed by the second parent.[1]

The total number of genes being being passed by both parents is therefore $(c + (d - d_1)) + ((l - c) + d_1) = l + d$. (See figure 4.1) □

Lema 4 uses lemmas 2 and 3 to give the possible range of the number of genes a parent can pass in shuffle crossover

lema 2 *If p_1 is the number of genes passed by the first parent in $(2k - 2)$-point shuffle crossover then* $\max(k, d) \leq p_1 \leq \min(l - k + 1 + d, l)$

Proof: From the model on page 18, the range of the number of genes copied from the first parent is $k \leq c_1 \leq l - k + 1$. Remembering that a gene is passed from a parent if it was copied from the parent, or if it has the same allele in both parents (i.e. the gene is a duplicate), the minimum number of genes passed occurs when all the duplicate genes are being copied from the parent. If the number of duplicate genes (d) is less than or equal to k, then all the duplicates could belong to the genes being copied from the first parent (figure 4.2) Therefore the minimum number of genes that could be passed is k

If $k < d \leq l - k + 1$, then the number of duplicate genes is still in the range of genes that could be copied from the first parent Then the minimum number of

[1] Note that the duplicate genes are counted as being passed twice once from each parent

PARENT A	:	a_1	0	a_3	a_4	1
PARENT B	:	b_1	0	b_3	b_4	1
CHILD	:	a_1	0	b_3	b_4	1

Figure 4.2: All duplicates in one parent.
A case where all duplicates are copied from the first parent, when $a_i \neq b_i$. The number of genes copied from the first parent, equals the number of genes passed.

genes that have to be copied to ensure that all of the duplicate genes are copied occurs when $c_1 = d$. In this case, d genes are passed from the first parent.

If $d > l - k + 1$ then there are $d - (l - k + 1) = d - l + k - 1$ more duplicates than genes that can be copied from the first parent. If all the duplicate genes are in the genes that are copied from the first parent, then the minimum number of genes that could be passed is the number of genes that can be copied plus the number of duplicate genes that can not be copied

$$(d - l + k - 1) + (l - k + 1) = d$$

Therefore the minimum number of genes that can be passed is $\max(k, d)$.

Similarly for the maximum number of genes passed. The second parent has $l - c_1$ genes copied, which means c is in the range $[k - 1, l - k]$. If $d \leq k - 1$ then all the duplicate genes could be copied from the second parent and the maximum number of genes allowed copied from the first parent. With $l - k + 1$ genes copied from the first parent, and all the duplicate genes being copied from the second parent, we would have $l - k + 1 + d$ genes passed.

If $d \geq k + 1$, it is possible for the first parent to have $l - k + 1$ genes copied. The remaining $k + 1$ genes which are copied from the second parent can consist entirely of the duplicates, which gives the maximum number of genes which could be passed is l.

Therefore the maximum number of genes that can be passed is $\min(l - k + d + 1, l)$. □

lema 3 *If p_2 is the number of the genes passed by the second parent in $(2k-2)$-point shuffle crossover, then* $\max(k-1, d) \leq p_2 \leq \min(l-k+d, l)$.

Proof: Using the same train of thought as in lema 2, we note that the number of genes that can be copied from the second parent is $k-1 \leq c_2 \leq l-k$. If $d \leq k-1$, all the duplicates can be copied from the second parent, giving $k-1$ genes passed as a minimum.

If $k-1 < d \leq l-k$, all the duplicates can be copied from the second parent. So as a minimum we have d genes passed.

For the maximum genes that could be passed, we remember that the first parent copies $c_1 = l - c_2$ genes. If $d \leq k$, then the first parent could copy all the duplicate genes. When the second parent copies $l-k$ genes, the maximum number of genes that would be passed is $l-k+d$.

When $d > k$, then the maximum number of genes that could be passed is l □

lema 4 *If p is the number of the genes passed by a parent in $(2k-2)$-point shuffle crossover, then* $\max(k-1, d) \leq p \leq \min(l-k+d, l)$

Proof Using lemmas 2 and 3, we can state the minimum and maximum bounds in terms of the bounds of the first and second parents

$$\min(\max(k, d), \max(k-1, d)) \leq p \leq \max(\min(l-k+1+d, l), \min(l-k+d, l))$$

Assuming that $d > k-1$, then $\max(k-1, d) = d$, and $\max(k, d) = d$. If $d \leq k-1$, then $\max(k-1, d) = k-1$, and $\max(k, d) = k$. Since $k-1 < \cdot$, the lower bound is $\max(k-1, d)$.

Similarly, using $d+1 \leq k$ as the dividing line, we have upper bound of $\min(l-k+d, l)$.□

Lemma 5 shows that increasing the number of crossover points in shuffle crossover, decreases the number of genes that can be passed.

lema 5 *For even number of crossover points (n) in n-point shuffle crossover, the range of the number of genes passed (p) is non-increasing as n increases.*

Proof: Let $n_1 = 2k - 2$, $n_2 = 2k$. Define the sets P_1 and P_2 as the possible values of p for n_1 and n_2 point shuffle crossover respectively. From lema 4 the elements of P_1 are defined by

$$\min(k-1, d) \leq p_1 \leq \min(l - k + d, l)$$

Similarly P_2's elements are defined by

$$\min(k, d) \leq p_2 \leq \min(l - (k+1) + d, l)$$

Therefore

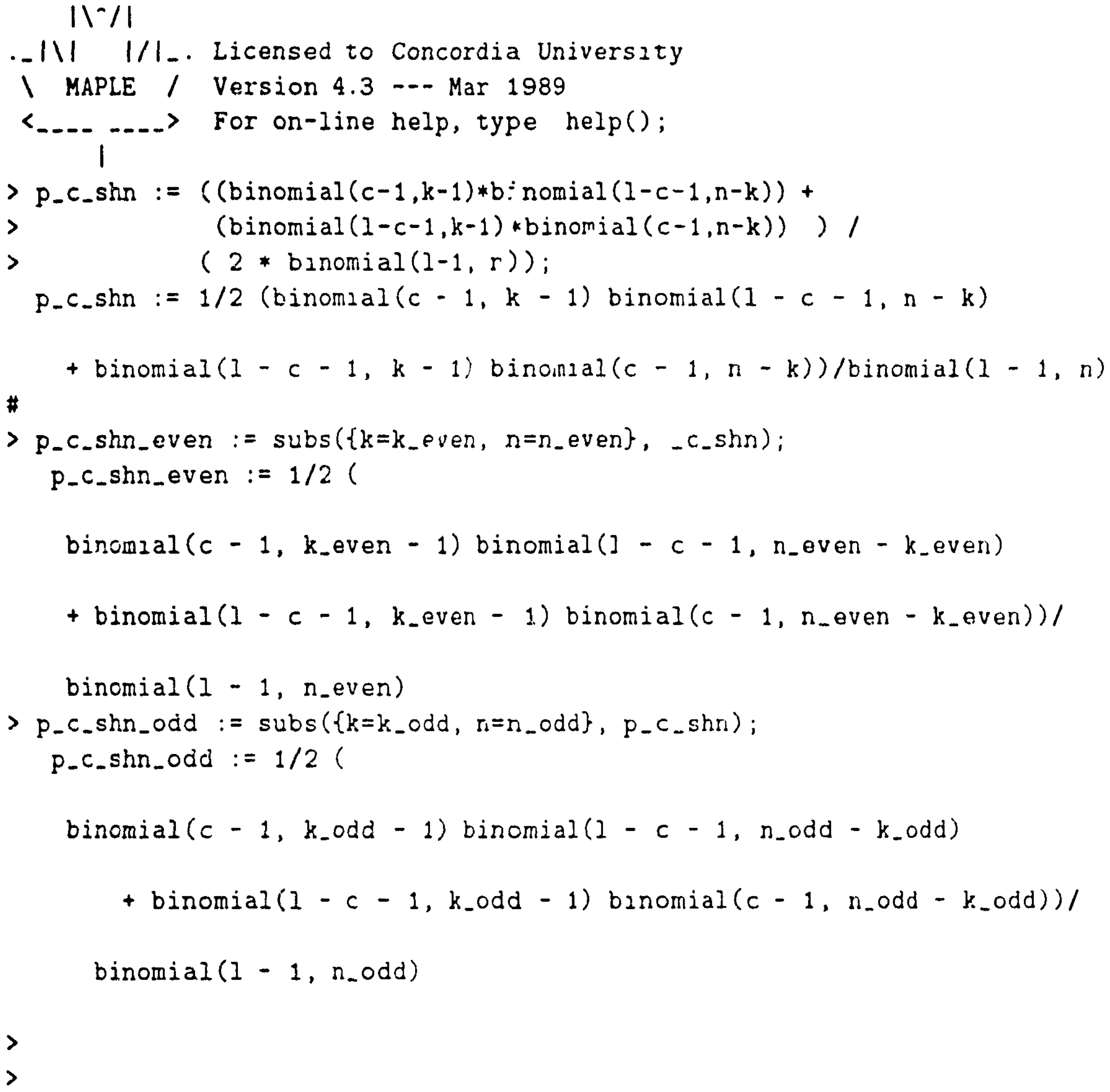
$$n_2 > n_1 \Rightarrow P_2 \subseteq P_1 \Box$$

Lemmas 6 and 7 state that we need not consider both the even and odd number of crossover points in shuffle crossover since they have the same characteristics.

Lema 6 shows that odd and even point shuffle crossover are identical. This fact is useful, since it simplifies the calculations that have to be performed when comparing shuffle crossover.

lema 6 *If n is odd and $n + 1 \leq l$ then $p_c^{sh-n}(c) = p_c^{sh-(n+1)}(c)$.*

Proof: The proof was confirmed by calculating all $l \leq 150$ for all $1 \leq n < l$. It was then programmed using Maple [7]. The output is given in figure 4.3.



```
    |\^/|
._|\|   |/|_. Licensed to Concordia University
 \  MAPLE  /  Version 4.3 --- Mar 1989
 <____ ____>  For on-line help, type  help();
      |
> p_c_shn := ((binomial(c-1,k-1)*b:nomial(l-c-1,n-k)) +
>             (binomial(l-c-1,k-1)*binomial(c-1,n-k))  ) /
>            ( 2 * binomial(l-1, r));
  p_c_shn := 1/2 (binomial(c - 1, k - 1) binomial(l - c - 1, n - k)

    + binomial(l - c - 1, k - 1) binomial(c - 1, n - k))/binomial(l - 1, n)
#
> p_c_shn_even := subs({k=k_even, n=n_even}, _c_shn);
   p_c_shn_even := 1/2 (

    binomial(c - 1, k_even - 1) binomial(l - c - 1, n_even - k_even)

    + binomial(l - c - 1, k_even - 1) binomial(c - 1, n_even - k_even))/

    binomial(l - 1, n_even)
> p_c_shn_odd := subs({k=k_odd, n=n_odd}, p_c_shn);
   p_c_shn_odd := 1/2 (

    binomial(c - 1, k_odd - 1) binomial(l - c - 1, n_odd - k_odd)

        + binomial(l - c - 1, k_odd - 1) binomial(c - 1, n_odd - k_odd))/

      binomial(l - 1, n_odd)

>
>
```

Figure 4.3: part 1. Maple program solving lema 6.

```
# PROOVE THAT p_c_shneven = p_c_shn_odd when
#
#   n_even = n_odd + 1
>
> n_even := n_odd + 1;
                              n_even := n_odd + 1
> k_even := k_odd + 1;
                              k_even := k_odd + 1
# n_even := 2 * k_even - 2;
>
> n_odd := 2 * k_odd - 1;
                              n_odd := 2 k_odd - 1
>
> p_c_shn_even;
       1/2 (binomial(c - 1, k_odd) binomial(l - c - 1, k_odd - 1)

            + binomial(l - c - 1, k_odd) binomial(c - 1, k_odd - 1))/

            binomial(l- 1, 2 k_odd)
>
> p_c_shn_odd;
         binomial(c - 1, k_odd - 1) binomial(l - c - 1, k_odd - 1)
         ----------------------------------------------------------
                       binomial(l - 1, 2 k_odd - 1)

>
```

Figure 4.3: part 2. Maple program solving lema 6.

```
# add known ranges of variables
> n_even >= 2;
                                  2 <= 2 k_odd

> n_odd >= 1;
                                1 <= 2 k_odd - 1

> n_even < l;
                                  2 k_odd < l

> n_odd < l;
                                2 k_odd - 1 < l

>
> k_even >= 2;
                                2 <= k_odd + 1

> k_odd >= 1;
                                   1 <= k_odd

>
> c >= 0;
                                     0 <= c

> c <= l;                                    c <= l
>
>
> dif := p_c_shn_even - p_c_shn_odd,
     dif := 1/2 (binomial(c - 1, k_odd) binomial(l - c - 1, k_odd - 1)

          + binomial(l - c - 1, k_odd) binomial(c - 1, k_odd - 1))/

         binomial(l - 1, 2 k_odd)

           binomial(c - 1, k_odd - 1) binomial(l - c - 1, k_odd - 1)
         - ---------------------------------------------------------
                         binomial(l - 1, 2 k_odd - 1)

>
> sol := solve(dif=0, {c});
bytes used=406720, alloc=139264, time=3.570
                                 sol := {c = c}

> sol := solve(dif=0, {l, k_odd, c}),
bytes used=807112, alloc=311296, time=6.820
                     sol := {c = _X, k_odd = k_odd, l = l}

>
> quit
bytes used=924268, alloc=352256, time=7 930
```

Figure 4.3: part 3 Maple program solving lema 6

lema 7 *If n is odd and $n+1 \leq l$ then $p_p^{sh-n}(p) = p_p^{sh-(n+1)}(p)$, $p_s^{sh-n}(b_A) = p_s^{sh-(n+1)}(b_A)$ and $p_{sc}^{sh-n}(b_A, b_B) = p_{sc}^{sh-(n+1)}(b_A, b_B)$.*

Proof: From lema 6 the $p_c^{sh-(n+1)}$ can be replaced with p_c^{sh-n} in equations (3.8). (3.9), and (3.10). Since none of the other terms contain n or k, the formulas are the same. □

The following lema shows that as we increase the number of crossover points. we reduce the number of possible combinations of schemas from the parents that can combine in the child

lema 8 *Let (b_A, b_B) be a pair of schema lengths such that their probability of combining by n-point shuffle crossover is given by the formula $p_{sc}^{sh-n}(b_A, b_B)$. Let P_{sc}^{sh-n} be the set of pairs of schema lengths whose probability of being combined is non-zero. Then*

$$P_{sc}^{sh-2k-2} \supseteq P_{sc}^{sh-2k}$$

Proof: For a schema of length b_A to be combined with another schema. it must survive in the child. For it to survive. at least b_A genes must be passed. We know from lema 5 that if $n_1 + 2 \leq n_2$. then the range of genes passed by n_1-point shuffle crossover is greater or equal to n_2-point shuffle crossover depending on the value of d.

First. assume d is small enough that the ranges of the number genes that could be passed (p) are different for n_1- and n_2-point shuffle crossover. Since every possible value of p that can be passed by n_2-point crossover. can also be passed by n_1-point crossover. $P_{sc}^{sh-2k-2} \supset P_{sc}^{sh-2k}$.

Second, assume that the number of duplicate genes (d) is large enough such that the ranges of genes that could be passed are the same for n_1 and n_2 point shuffle crossover are the same. When this happens the same sets of schemas can be combined. albeit with different probabilities for the two crossover operators. □

lema 9 *The range of genes passed* (p) *for uniform crossover is*

$$[max(\lfloor l/2 \rfloor, d), min(l, \lceil l/2 \rceil + d)]$$

Proof: The proof is similar to lemmas 2 and 3. The minimum number of genes that can be passed is the number of genes copied, which is $\lfloor l/2 \rfloor$ for uniform crossover; or the number of duplicate genes, when there are more duplicates then there are genes to to be copied. The maximum number of genes that can be passed is the length of the chromosome or the number of genes copied from one parent with the addition of all duplicates (which are copied from the other parent).□

lema 10 $\quad P^u_{sc} \supseteq P^{u+}_{sc}$

where P^u_{sc} *and* P^{u+}_{sc} *are the sets of schema lengths whoos probability of combining are non-zero for uniform and modified uniform crossover respectively.*

Proof: Same argument as for lema 8. The range of genes that can be passed by modified uniform crossover is $\lfloor \frac{l+d}{2} \rfloor \leq \lceil \frac{l+d}{2} \rceil$ From lema 9 we get the the range of genes passed for uniform crossover Since the range of modified uniform crossover is contained in the range of uniform crossover, the schema combinations are also contained.□

In lema 11, we show that there is a situation where the probability of combining two schemas is greater under modified uniform crossover than uniform crossover, for when the length of the chromosome is even. Odd cases are covered in lema 12

lema 11 $\forall l = 2i \wedge l \geq 4, \exists d, b_A, b_B, p^{u+}_{sc}(b_A, b_B) > p^u_{sc}(b_A, b_B)$

Proof: Assume $d = b_A = b_B = 2$, so that both $(l+d)/2$ and $l/2$ are even

On the RHS we start with equation (3.10)

$$p^u_{sc}(b_A, b_B) = \sum_{p=0}^{l} p^u_p(p) \frac{\begin{pmatrix} p \\ b_A \end{pmatrix} \begin{pmatrix} l+d-p \\ b_B \end{pmatrix}}{\begin{pmatrix} l \\ b_A \end{pmatrix} \begin{pmatrix} l \\ b_B \end{pmatrix}}$$

From lema 9 we can constrain the summation to eliminate the zero terms. After replacing d, b_A, and b_B we get

$$p^u_{sc}(2,2) = \sum_{p=l/2}^{min(l,l/2+2)} p^u_p(p)\frac{\binom{p}{2}\binom{l-p+2}{2}}{\binom{l}{2}\binom{l}{2}}$$

Replace $p^u_p(p)$ with its definition as stated in equation (3.8), and changing the upper bound of the summation to reflect the known minimum

$$p^u_{sc}(2,2) = \sum_{p=l/2}^{l/2+2}\left(\sum_{c=0}^{l} p^u_c(c)\frac{\binom{l-c}{p-c}\binom{c}{p-d}}{\binom{l}{d}}\right)\frac{\binom{p}{2}\binom{l-p+2}{2}}{\binom{l}{2}\binom{l}{2}}$$

Now by definition of uniform crossover, since l is even $p^u_c(c)$ is non-zero only when $c = l/2$.

$$p^u_{s}(2,2) = \sum_{p=l/2}^{l/2+2}\left(\frac{\binom{l/2}{p-l/2}\binom{l/2}{p-2}}{\binom{l}{2}}\right)\frac{\binom{p}{2}\binom{l-p+2}{2}}{\binom{l}{2}\binom{l}{2}}$$

Expanding the summation, and using the identity $\binom{a}{b} = \binom{a}{a-b}$

$$p^u_{sc}(2,2) = \binom{l/2}{2}^2\left[2\binom{l/2+2}{2}+\binom{l/2+1}{2}^2\right]\binom{l}{2}^{-3}$$

For modified uniform crossover we again start with the definition of equation (3.10)

$$p_{sc}^{u+}(b_A, b_B) = \sum_{p=0}^{l} p_p^{u+}(p) \frac{\begin{pmatrix} p \\ b_A \end{pmatrix} \begin{pmatrix} l+d-p \\ b_B \end{pmatrix}}{\begin{pmatrix} l \\ b_A \end{pmatrix} \begin{pmatrix} l \\ b_B \end{pmatrix}}$$

Substituting b_A, b_B, d.

$$p_{sc}^{u+}(2, 2) = \sum_{p=0}^{l} p_p^{u+}(p) \frac{\begin{pmatrix} p \\ 2 \end{pmatrix} \begin{pmatrix} l-p+2 \\ 2 \end{pmatrix}}{\begin{pmatrix} l \\ 2 \end{pmatrix} \begin{pmatrix} l \\ 2 \end{pmatrix}}$$

From the definition of uniform crossover, we observe that since $l+d$ is even, $p_p^{u+}(p)$ is non-zero only when $p = (l+d)/2$.

$$p_{sc}^{u+}(2, 2) = \frac{\begin{pmatrix} (l+2)/2 \\ 2 \end{pmatrix} \begin{pmatrix} (l+2)/2 \\ 2 \end{pmatrix}}{\begin{pmatrix} l \\ 2 \end{pmatrix} \begin{pmatrix} l \\ 2 \end{pmatrix}}$$

Using the identity $\begin{pmatrix} x \\ 2 \end{pmatrix} = x(x-1)/2$ when $x \geq 2$

$$p_{sc}^{u}(2, 2) = 1/16384 l^5 (l-2)^2 (l+2)(16l + 64 + l^3 + 2l^2)(l-1)$$

$$p_{sc}^{u+}(2, 2) = 1/16 \frac{(l+2)^2}{(l-1)^2}$$

Solving for $p_{sc}^{u+} \geq p_{sc}^{u}$ in terms of l (using Maple[2]) gives the solution $l \geq -2$.□

lema 12 $\forall l = 2i - 1 \wedge l \geq 5, \exists d, b_A, b_B, p_{sc}^{u+}(b_A, b_B) > p_{sc}^{u}(b_A, b_B)$

Proof:

[2] Solving $p_{sc}^{u+} = p_{sc}^{u}$ in terms of l gives solutions of -2 and complex numbers. From there it is simple to determine the $l > -2$ is the solution for the inequality

$$p^{u+}_{sc}(b_A, b_B) > p^{u}_{sc}(b_A, b_B)$$

$$\frac{\sum_{p=0}^{l} p^{u+}_{p}(p) \begin{pmatrix} p \\ b_A \end{pmatrix} \begin{pmatrix} l+d-p \\ b_B \end{pmatrix}}{\begin{pmatrix} l \\ b_A \end{pmatrix} \begin{pmatrix} l \\ b_B \end{pmatrix}} > \frac{\sum_{p=0}^{l} p^{u}_{p}(p) \begin{pmatrix} p \\ b_A \end{pmatrix} \begin{pmatrix} l+d-p \\ b_B \end{pmatrix}}{\begin{pmatrix} l \\ b_A \end{pmatrix} \begin{pmatrix} l \\ b_B \end{pmatrix}}$$

Let $b_A = b_B = d = 1$. Then from the definition of modified uniform crossover we know that if $p^{u+}_{p}(p) \neq 0$ then $p = i$, since $l + d$ is even.

$$\begin{aligned} p^{u+}_{sc}(b_A, b_B) &= \frac{\begin{pmatrix} i \\ 1 \end{pmatrix} \begin{pmatrix} (2i-1)+1-i \\ 1 \end{pmatrix}}{\begin{pmatrix} 2i-1 \\ 1 \end{pmatrix} \begin{pmatrix} 2i-1 \\ 1 \end{pmatrix}} \\ &= \frac{\begin{pmatrix} i \\ 1 \end{pmatrix} \begin{pmatrix} i \\ 1 \end{pmatrix}}{\begin{pmatrix} 2i-1 \\ 1 \end{pmatrix}^2} \\ &= i^2/(2i-1)^2 \end{aligned}$$

From equation (3.3), the probability of copying c genes for uniform crossover is $1/2$ when $i - 1 \leq c \leq i$, and zero otherwise.

From equation (3.8)

$$p^{u}_{p}(p) = \sum_{c=0}^{l} p_c(c) \frac{\begin{pmatrix} l-c \\ p-c \end{pmatrix} \begin{pmatrix} c \\ p-d \end{pmatrix}}{\begin{pmatrix} l \\ d \end{pmatrix}}$$

$$= \sum_{c=i-1}^{i} 1/2 \frac{\binom{(2i-1)-c}{p-c}\binom{c}{p-1}}{\binom{2i-1}{1}}$$

$$= 1/2 \frac{\binom{(2i-1)-(i-1)}{p-(i-1)}\binom{i-1}{p-1}}{\binom{2i-1}{1}} + 1/2 \frac{\binom{(2i-1)-i}{p-i}\binom{i}{p-1}}{\binom{2i-1}{1}}$$

$$= \frac{1}{2(2i-1)}\left[\binom{i}{p-i+1}\binom{i-1}{p-1} + \binom{i-1}{p-i}\binom{i}{p-1}\right]$$

We now use equation (3.10) again

$$p^u_{sc}(b_A, b_B) = \sum_{p=0}^{l} p^u_p(p) \frac{\binom{p}{b_A}\binom{l+d-p}{b_B}}{\binom{l}{b_A}\binom{l}{b_B}}$$

$$= \sum_{p=0}^{2i-1} p^u_p(p) \frac{\binom{p}{1}\binom{(2i-1)+1-p}{1}}{\binom{2i-1}{1}\binom{2i-1}{1}}$$

From lema 9 we can know that for $p^u_p(p)$ to be non-zero then

$$\begin{aligned} \lfloor l/2 \rfloor &\leq p \leq \lceil l/2 \rceil + 1 \\ \lfloor (2i-1)/2 \rfloor &\leq p \leq \lceil (2i-1)/2 \rceil + 1 \\ i-1 &\leq p \leq i+1 \end{aligned}$$

Restricting the range of the summation, and using the identity $\binom{a}{1} = a$, we now get

$$p^{k}_{S_i}(b_A, b_B) = \frac{1}{(2i-1)^2} \sum_{p=i-1}^{i+1} p^{k}_{P_i}(p)[p][2i-p]$$

$$= \frac{1}{(2i-1)^2}$$
$$\sum_{p=i-1}^{i+1} \frac{1}{2(2i-1)} \left[\binom{i}{p-i+1}\binom{i-1}{p-1} + \binom{i-1}{p-i}\binom{i}{p-1} \right] [p][2i-p]$$

$$= \frac{1}{2(2i-1)^3}$$
$$\sum_{p=i-1}^{i+1} \left[\binom{i}{p-i+1}\binom{i-1}{p-1} + \binom{i-1}{p-i}\binom{i}{p-1} \right] [p][2i-p]$$

$$= \frac{1}{2(2i-1)^3}$$
$$\left(\left[\binom{i}{(i-1)-i+1}\binom{i-1}{(i-1)-1} + \binom{i-1}{(i-1)-i}\binom{i}{(i-1)-1} \right] \right.$$
$$[i-1][2i-(i-1)]$$
$$+ \left[\binom{i}{(i)-i+1}\binom{i-1}{(i)-1} + \binom{i-1}{(i)-i}\binom{i}{(i)-1} \right]$$
$$[i][2i-(i)]$$
$$+ \left[\binom{i}{(i+1)-i+1}\binom{i-1}{(i+1)-1} + \binom{i-1}{(i+1)-i}\binom{i}{(i+1)-1} \right]$$
$$\left. [i+1][2i-(i+1)] \right)$$

$$= \frac{1}{2(2i-1)^3}$$
$$\left(\left[\binom{i}{0}\binom{i-1}{i-2} + \binom{i-1}{-1}\binom{i}{i-2} \right] [i-1][i+1] \right.$$

$$
\begin{aligned}
&+\left[\binom{i}{1}\binom{i-1}{i-1}+\binom{i-1}{0}\binom{i}{i-1}\right][i][i] \\
&+\left[\binom{i}{2}\binom{i-1}{i}+\binom{i-1}{1}\binom{i}{i}\right][i+1][i-1]\Bigg) \\
= &\ \frac{1}{2(2i-1)^3} \\
&([(i-1)]\,[i-1][i+1] \\
&+[(i)+(i)]\,[i][i] \\
&+[(i-1)]\,[i+1][i-1]) \\
= &\ \frac{1}{2(2i-1)^3} \\
&((i-1)(i-1)(i+1) \\
&+(2i)(i)(i) \\
&+(i-1)(i+1)(i-1)) \\
= &\ \frac{2(i-1)^2(i+1)+2i^3}{2(2i-1)^3}
\end{aligned}
$$

Solving $p_{sc}^{u+}(1,1) > p_{sc}^{u}(1,1)$ for i gives the result of $i > 1$ or $i < 1/2$. Since by definition $i > 2$, the proof is concluded. □

lema 13 $\forall n, 3 \leq n \leq l-1, 1 \leq n' \leq n-2, \exists b_A, b_B, d, 5 \leq l \leq 300, p_{sc}^{sh-n}(b_A, b_B) > p_{sc}^{sh-n'}(b_A, b_B)$.

The theory was confirmed by running a C program.[3] The upper bound of 300 was chosen, for implementation reasons not theoretical. □

The three theorems that result from the lemmas developed, show that there is no best crossover operator for all situations among n-point, uniform, and modified uniform crossover. Each of the operators have two schema lengths which they combine with a higher probability then the other operators.

[3] The program is listed in appendix B.

Theorem 1 *Neither p_{sc}^{u+} nor p_{sc}^{u} dominate each other, for all crossovers situations.*

Proof: From lemmas 11 and 12 there is a (b_A, b_B) and d such that $p_{sc}^{u+}(b_A, b_B) > p_{sc}^{u}(b_A, b_B)$. From lema 10 there is a (b_A, b_B) and d such that $p_{sc}^{u}(b_A, b_B) > p_{sc}^{u+}(b_A, b_B)$.□

Theorem 2 $\forall n, n', 3 \leq n < l, 5 \leq l \leq 300, 1 \leq n' \leq n-2$ *neither p_{sc}^{sh-n} nor $p_{sc}^{sh-n'}$ dominates for all crossover situations.*

Proof: Direct consequence of lemmas 7, 8 and 13. □

Theorem 3 *Neither p_{sc}^{u} nor p_{sc}^{sh-n} dominate each other, for all crossovers situations*

Proof. Note that shuffle crossover with $l-1$ crossover points will always copy either $\lfloor l/2 \rfloor$, or $\lceil l/2 \rceil$ genes from the first parent. Since this is exactly the same thing as uniform crossover, we can replace p_{sc}^{u} with $p_{sc}^{sh-(l-1)}$. From theorem 2 we no that there is no ideal number of crossover points.□

Chapter 5

Experiments

In chapters 6 and 7, we will describe the results of three different experiments. All the experiments use the same results from ten different problems, with six different parameter sets.

This chapter will describe the parameters used, the problems that are analyzed later on, and the methods used to evaluate each problem parameter set combination.

5.1 Parameter Sets

A parameter set consisted of: *population size*, *desegregation rate*, *mutation rate*, *crossover rate*, *decreasing fitness*, *CF*[1], *samples per CF*, and *max generations*.

The number of children created at each generation was population size × crossover rate. When this was not an integer, a random number was chosen between zero and one, and if the number was larger than the fraction part the ceiling was used, else the floor was used.

Evaluation of the fitness of members of the population was done by using ranking with the following two rules

[1] CF stands for Goldbergs [9] closeness factor

$$\text{fitness'}(c_i) = 1.0 \Leftrightarrow \neg\exists j.\ \text{strength}(c_j) > \text{strength}(c_i) \tag{5.1}$$

$$\begin{aligned} & \text{fitness'}(c_i) = \text{decreasing fitness} * \text{fitness'}(c_j) \\ \Leftrightarrow \quad & \qquad \text{strength}(c_i) < \text{strength}(c_j) \\ & \wedge \quad \neg\exists k, (\text{strength}(c_i) < \text{strength'}(c_k) < \text{strength'}(c_j)) \end{aligned} \tag{5.2}$$

The above rules give rise to the rule

$$\text{fitness'}(c_i) = \text{fitness'}(c_j) \Leftrightarrow \text{strength}(c_i) = \text{strength}(c_j)$$

where fitness'(c_i) is an estimate of how many children the i-th chromosome in the population will produce, and strength(c_i) is the evaluation of the chromosome by the function that is to be optimized. The actual fitness of the chromosomes were decided by prorating the estimates by the number of children that were to be created.

The parents where chosen using the SUS algorithm[6], with no checks to see if the same parent was being matched [2] Crossover produced only one child, compared to the normal two. It was hoped that this would increase population diversity, without changing the fundamental characteristics of a genetic algorithm.

Each child was then mutated, by having each gene subjected to change with probability mutation rate.

A member of the population was then selected for removal. The method of removal was to select *samples per CF* members from the population at random keeping the lowest scoring member. This was performed *CF* times. Of the members that were kept, the one that was the closest (using Hamming Distance) to the child was the one that was replaced.[9] All the problems where solved with a *population size* of 50, *crossover rate* of 0.6, *CF* equal to 5, and 5 *samples per CF*.

After all the children were entered into the population, desegregation was performed (see chapter 6). When an allele was found to be missing in the population, it was entered with probability desegregation rate.

[2] Allowing for the possibility of a non-effective crossover

The three parameter sets differed on the mutation and desgregation rate. Parameter set M+D had a mutation rate of 0.001 and a desegregation rate of 0.001 Parameter set D had no mutation (rate of 0.0) and a desegregation rate of 1.0. The parameter set M had a mutation rate of 0.001 with no desegregation (rate of 0.0).

All problems were searched for a maximum of 1000 generations.

5.2 The problems

The problems that were used include the standard $F1$ to $F5$ problems designed by De Jong [14], the travelling salesman problem (TSP), one max, continous one max, sparse one max as defined by Syswerda [25], and a variation of $F5$ labelled $F5'$ designed by the author.

5.2.1 F1

The problem was to optimize the function

$$f_1(x) = x_1^2 + x_2^2 + x_3^2$$

where the search space is restricted in $-5.12 \leq x \leq 5.12$ for $i = 1, 2$, and 3 with a resolution factor $\Delta x = 0.1$. (i.e. each chromosome is represented by a 10-bit binary string).

max(f) = 78.6 (in the restricted search area)

min(f) = 0

ave(f) = 26.2

[14]

5.2.2 F2

$$f_2(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

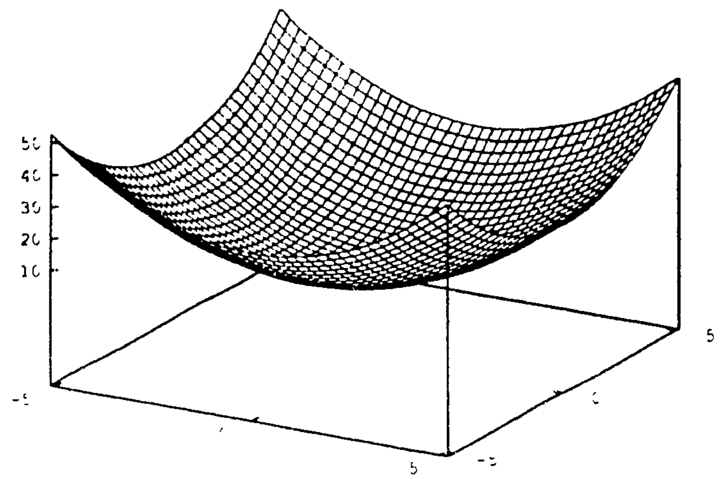


Figure 5.1: Projection of F1, with x_3 set to 0.

where the search space is restricted in $-2.048 \le x_i \le 2.048$, $i = 1, 2$ with a resolution factor of $\Delta x = 0.001$ (thus a 12-bit chromosome representation)

$\max(f) = 3905.93$ (at $x_1 = -2.048$ and $x_2 = -2.048$)

$\min(f) = 0$ (at origin)

$\text{ave}(f) = 491.05$

[14]

5.2.3 F3

$$f_3(x) = \lfloor x_1 \rfloor + \lfloor x_2 \rfloor + \lfloor x_3 \rfloor + \lfloor x_4 \rfloor + \lfloor x_5 \rfloor$$

The search space is defined by $-5.12 \le x_i \le 5.12$ for all i's with a resolution factor $\Delta x = 0.01$ (i.e. a 10-bit binary string chromosome for each x_i.)

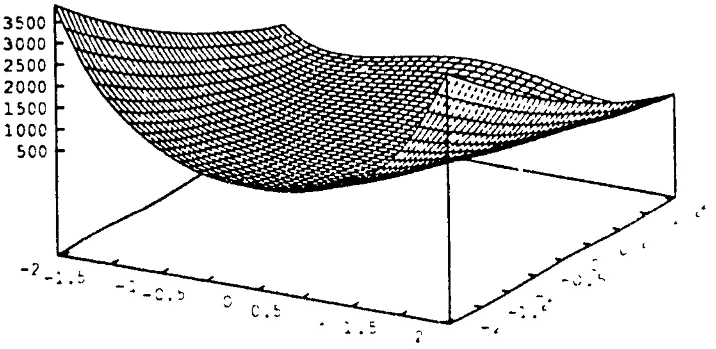


Figure 5.2: F2

max(f) = 25 (at $x_i = 5.12$ for all i's)

min(f) = 0 (at $x_i = -5.12$ for all i's)

ave(f) = -2.5

[14]

5.2.4 F4

$$f_4(x) = \sum_{i=1}^{i=30} ix_i^4 + Gauss(0,1)$$

[a summation of 30 terms with a gaussian noise of mean 0, standard deviation 1, added.]

The search space is restricted by $-1.28 \leq x_i \leq 1.28$ for all i's with a resolution factor $\Delta x_i = 0.01$ (so each x_i is represented by a 8-bit binary string.)

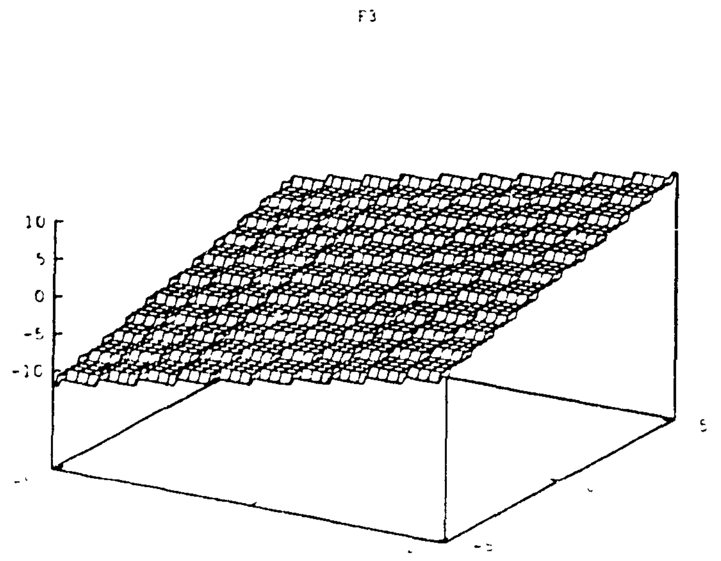


Figure 5.3: Projection of F3, with x_3, x_4, and x_5 set to 0

$\max(f) = 1248.2$ (at the boundary)

$\min(f) = 0$ (at origin)

$\mathrm{ave}(f) = 249.6$

[14]

5.2.5 F5

$$1/f_5(x) = 1/K + \sum_{j=1}^{j=25} 1/g_j(x)$$

where $g_j(x) = c_j + \sum_{i=1}^{i=2} (x_i - a_{ij})^6$

The a_{ij} used are the following:

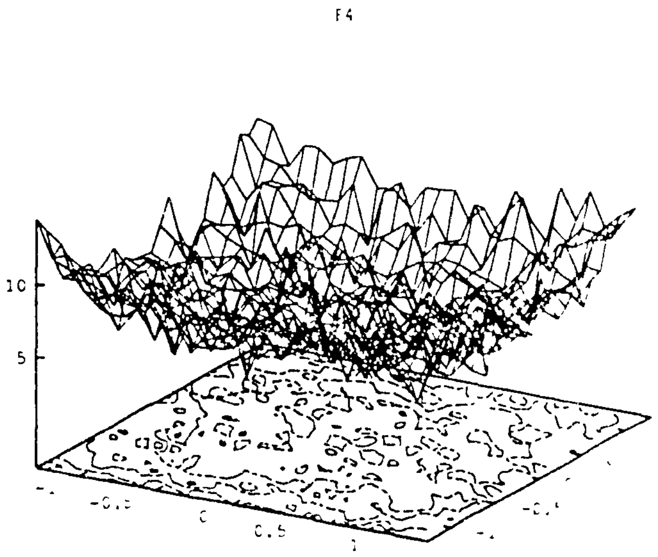


Figure 5.1 Projection of F4 with x_i ($i > 2$) set to 0

$(-32, -32)$	$(-16, -32)$	$(0, -32)$	$(16, -32)$	$(32, -32)$
$(-32, -16)$	$(-16, -16)$	$(0, -16)$	$(16, -16)$	$(32, -16)$
$(-32, 0)$	$(-16, 0)$	$(0, 0)$	$(16, 0)$	$(32, 0)$
$(-32, 16)$	$(-16, 16)$	$(0, 16)$	$(16, 16)$	$(32, 16)$
$(-32, 32)$	$(-16, 32)$	$(0, 32)$	$(16, 32)$	$(32, 32)$

[indexed by 25 j's. i.e. (a_{1j}, a_{2j}) for each vector]

with $c_j = j$ and $K = 500$

The search space is restricted by $-65.536 \leq x_i \leq 65.636$ for $i = 1, 2$ with a resolution factor $\Delta x = 0.001$ on each axis (i.e. 17-bit binary string)

$\max(f) \approx 500$

$\min(f) \approx 1$

$\mathrm{ave}(f) \approx 473$

[14]

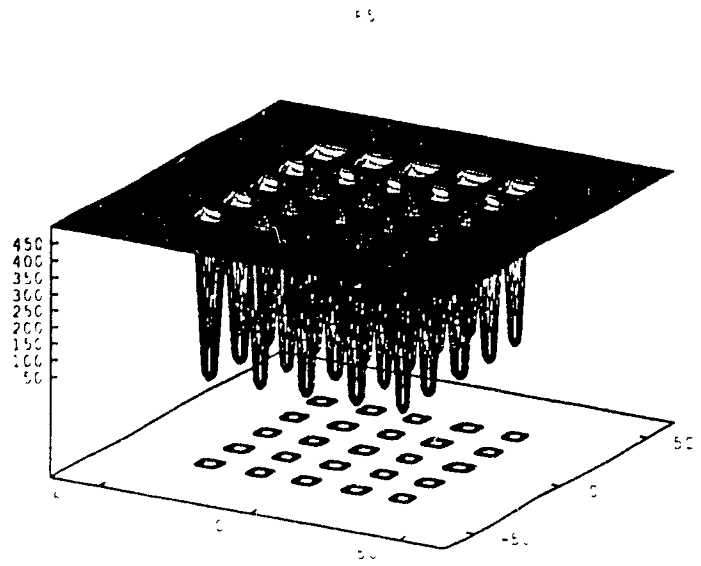


Figure 5.5: F5.

5.2.6 One Max

The function is simply the number of ones in a binary chromosome of length 30.

5.2.7 Sparse One Max

The problem was to find the chromosome where every 10th bit was set to one. The chromosome was of length 300. and the strength of the chromosome was one for every tenth bit that was set to one The other 290 bits have no effect. [25]

5.2.8 Travelling Salesman Problem (TSP)

The function is to minimize the length of a trip of 16 cities evenly spaced along a circle The tour is represented by a 96 bit chromosome. Each 8 bits indicates how close the city should be to the start of the tour. The cities are then sorted by rank and the distance for the tour is computed To make this problem a maximization problem, the negative of the distance is used as the strength.[25]

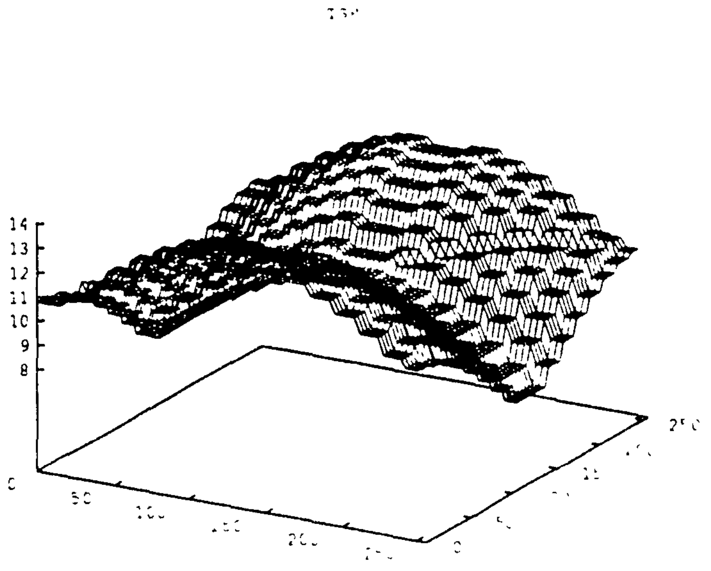


Figure 5.6: Projection of TSP Problem
Cities 1 and 9 are varying, while the other cities are ordered, with their priorities set in intervals of 16.

5.2.9 F5'

The function is $-1/F5$. The function started life as a programming error, but the results are interesting since it takes a long time for the GA to find its peaks

5.3 Performance Measures

We will be using five measurements to evaluate the effectiveness of the different crossover and parameter set combinations. All the measurements are for a generation t.

The first two measurements are the average, and maximum strength of the members of the population at generation t. These will be represented by avg_t and max_t.

The third measurement we will be using is MSI_t, which is the maximum strength found by generation t . This measurement is useful for situations where there is no cost for making an evaluation off-line, and we are interested in finding the "best" answer

The fourth measurement is the on-line strength [14]

$$\text{on-line}_t = \frac{1}{t}\sum_{i=0}^{t} \text{a}\bar{\text{v}}\text{g}_i$$

The measurement is useful for determining the value of the search process where every evaluation has a cost.

The final measurement is the off-line strength [14]

$$\text{off-line}_t = \frac{1}{t}\sum_{i=0}^{t} \text{max}_i$$

> It is appropriate in situations in which a search might be done "off-line" while the current best structure is used (and paid for) until a better one is found. This measure does not penalize search methods for exploring poor regions of the search space on the way to better solutions, as on-line does. [22]

Chapter 6

Desegregation

As mentioned in chapter 5, mutation is needed when the population converges to ensure enough diversity for the search to turn up new alternatives. The difficulty with the standard mutation algorithm is that it must have a fairly high rate of activity to be useful, since it is completely local to the chromosomes being generated by crossover. Unfortunately a high mutation rate disrupts productive chromosomes and interferes in the process of examining useful schema, by making the search process closer to a random walk.

In this chapter we will introduce a new mutation operator called *Desegregation*. The goal of desegregation is to maintain population diversity while minimizing the number of changes in the alleles of generated chromosomes.

Desegregation works by keeping track of all the new chromosomes in a generation. After all the chromosomes have been entered, the entire population is examined to see if there is an allele missing in the population.[1] If an allele does not occur then a new chromosome is chosen at random (with equal probability) and the gene whose position contains the missing allele is changed to the missing value. The algorithm is given in figure 6.1

[1] In reality the population can keep track of the number of alleles in each position by monitoring the replacement of chromosomes.

After the children have been entered into the population, and old members of the population have been killed

1. Examine all members of the population and determine if any alleles are missing
2. For each allele that is missing
   (a) Select a child at random
   (b) Change the childs genes so it contains the missing allele.

Figure 6.1 Desegregation algorithm.

Three stages of searching were observed in the experiments described in chapter 5: *wide search*, *narrow search*, and *point search*. Wide search occurs at the beginning of the process when the convergence is still increasing while the off-line and on-line strengths are increasing. Narrow search is the second phase of search, when the convergence rate has stabilized but either the off-line or on-line strengths is increasing. Point search occurs when the population has converged, while both the on line and off-line strengths are stable. The experiments have been previously described in Pawlowsky and Krzyzak [17] and are quoted below.

> During the first phase the area being searched is extremely broad, and a low mutation rate does not effect the search process in a noticeable manner since there is a good mix of values for each gene.
>
> The second phase is often absent for GA easy problems (such as F1), since a maximum value is often found before the population has converged. For problems which are difficult, premature convergence can lead to a non-optimal value being found. Desegregation is at its strongest during this phase. The genes that are changed by desegregation are extremely important, since if the population has converged on a value for

one gene, it will only search schemas with that gene's value, until a chromosome is mutated in that position. If only mutation is being used, then a high mutation rate is needed to change the gene. Unfortunately a high mutation rate will also changes genes that have not all converged on the same allele.

Desegregation on the other hand, changes only those genes that have converged. The minimal disruption of the chromosomes created by desegregation allows the search process to continue in a more efficient manner than just increasing the mutation rate.

During the third phase, the problem has been solved and any mutation will just drive the population away from convergence. We therefore expect that desegregation will lower the average population strength during the third phase, since as the population converges to the already found maximum, desegregation forces the new chromosomes away from the point [15].

The algorithm was analyzed with the experiments described in chapter 5.

As expected when desegregation is enabled, the GA is able to find the maximum value of a function earlier in the search process. The MSF and Maximum curves when desegregation is used are consistently greater than or equal to the experiments where only mutation is used. Also as expected from the reasoning (above), the parameter set without mutation outperforms the parameter sets with mutation in finding the maximum value.

Since desegregation will mutate many more genes than mutation when the population converges, it was expected and confirmed that the average population strength is lower for the parameter sets with desegregation during the third phase.

What was not expected was that during the second phase, the parameter sets that used desegregation had lower average strengths. Examining the functions that had long second phases (TSP, F5, F5'), it was noticed that the number of genes changed with desegregation was higher than the number of genes changed with mutation during the second phase.

From the results we received, it also appears that desegregation does not harm the average strength of a population for functions that have a broad maximum surface, or multiple peaks. Since these functions have many different maximum areas, and we are only trying to find any one of them, the population can diverge into the different areas. Since there was no effort made to have like members of the population selected for crossover, the parents can come from different areas in the search space, keeping the convergence rate down. The lower convergence rate lessens the need for the desegregation operator to be be active, thereby lowering the disruption caused by mutation.[15]

The behaviour is clarified by the folowing examples. Assume there is a function with two global maximums at 0000 and 1111. Let the population size be four, with two members at each maximum. Now since there are no missing alleles in the population (see figure 6.2), desegregation will not be activated, and therfore not have any harmful effects. Now if there were a single globabl maximum (1111) and the population has converged on it (figure 6.3), such that all the alleles in the population are the same, then desegregation would be active. Unfortunately desegregation would force new members of the population to contain strings with a 0 in it, which is sub-optimal. The new sub-optimal chromosomes would lower the average population strength, when entered.

1111
1111
0000
0000

Figure 6.2: Population converged on multiple maximums
Note that even though the population has converged, there are no missing alleles in the population.

1111
1111
1111
1111

Figure 6.3: Population converged on maximum value
Since the poplation has converged so there are missing allels, desegregation would be active, even though all the missing alleles lead to sub-optimal solutions.

6.1 Tables and Charts

Included at the end of this chapter are tables which compare the effectiveness of the three parameter sets described in section 5.1. The tables list the mean values of the measurement and the ranking of the parameter set for the measurement. The rankings were done by performing a Schaffe Test with a probability of 0.20. A ranking of 1 indicates that the parameter set belongs to the lowest sub group. Rankings of 2 and 3 indicate that parameter set belongs to higher ranking sub groups. A ranking of 1.5 indicates that there is no significant difference for the parameter set, and the parameter sets ranked first and second.

The X-axis for all charts is the generation number. The lines are solid for the mutation only parameter set, dashed for mutation and desegregation, and dotted for desegregation only. Since the absolute differences between the three methods is not great, the plots labeled *DIFF* have had the values of the M parameter set subtracted.

The charts labelled ON-LINE, OFF-LINE, MSF, AVG, and MAX, represent the on-line, off-line, maximum so far, average at generation, and maximum at generation scores described in section 5.3. The charts labelled CONV measure the population convergence using

$$\text{convergence} = \frac{1}{l} \sum_{i=1}^{\text{pop size}} \max(0_i, 1_i)$$

where 0_i and 1_i are the number of 0's and 1's in the i-th gene position respectively.

The charts "Mut & Deseg" have the cumulative total of the number of genes that were changed by either mutation or desegregation as their Y-axis.

F1								
UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 5 69 19	2 76 47	2 78 47	1 73 92	2 78 47	1 0 94	3 54 95
	M	2 69 64	1 75 81	1 77 93	2 75 62	1 77 93	2 0 96	2 41 20
	D	1 68 71	1 5 76 10	1 5 78 01	1 75 58	1 5 78 04	1 0 94	1 8 60
100	M+D	1 71 55	2 77 46	1 78 48	1 73 98	1 78 48	1 0 95	3 101 15
	M	2 73 32	1 77.05	1 78 74	2 77 76	1 78 74	2 0 99	2 85 15
	D	1 71 25	1 5 77 22	1 78 46	1 73 99	1 78 46	1 0 95	1 11 25
200	M+D	1 72 77	2 77 96	1 78 48	1 73 91	1 78 48	1 0 95	3 187 75
	M	2 75 47	1 77 75	1 78 47	2 77 59	1 78 47	3 0 95	2 180 55
	D	1 72 63	1 5 77 84	1 78 47	1 74 08	1 78 46	3 0 98	1 187 75
500	M+D	1 73 50	2 78 27	1 78 48	1 74 07	1 78 48	1 0 95	2 462 30
	M	2 76 82	1 78 18	1 78 47	2 77 58	1 78 47	2 0 98	2 456 05
	D	1 73 49	1 5 78 21	1 78 46	1 74 01	1 78 46	1 0 95	1 15 85
1000	M+D	1 73 76	1 78 38	1 78 48	1 73 98	1 78 48	1 0 95	2 917 95
	M	2 77 27	1 78 32	1 78 47	2 77 67	1 78 47	2 0 99	2 914 55
	D	1 73 78	1 78 34	1 78 46	1 74 08	1 78 46	1 0 95	1 24 05

Table 6.1: Results of using desegregation on F1 for uniform crossover

F1

MODIFIED UNIFORM CROSSOVER

gen	online	offline	MSF	avg	max	converge	changes
50 M+D	1 69 13	1 76 34	1 5 78 25	1 73.69	1 5 78 25	1 0.94	3 54 55
M	1 69 56	1 76 13	1 78 03	2 75 60	1 78 03	2 0.96	2 44 60
D	1 69 02	1 76 33	1 78 40	1 73 95	2 78 40	1 0 94	1 10 65
100 M+D	1 71 51	1 77 16	2 78 51	1 74 13	2 78 51	1 0 95	3 101 65
M	2 73 09	1 77 17	1 78 31	2 77 37	1 78 31	2 0 98	2 90.50
D	1 71 49	1 77 3981	2 78 49	1 74 12	2 78 49	1 0 95	1 12 10
200 M+D	1 72 7958	1 77 95	1 78 51	1 74 12	1 78 51	1 0 95	3 190 35
M	2 77 11	1 77 82	1 78 51	2 77 67	1 78 51	3 0 99	2 178 80
D	1 72 80	2 77 91	1 78 49	1 74 06	1 78 49	2 0 95	1 13 70
500 M+D	1 73 77	1 78 28	1 78 51	1 74 25	1 78 51	1 0 95	3 463 70
M	2 76 85	1 78 23	1 78 51	2 77 92	1 78 51	3 0 99	2 453 85
D	1 73 61	1 78 27	1 78 49	1 74 34	1 78 49	2 0 95	1 17 95
1000 M+D	1 73 83	1 78 39	1 78 51	1 74 03	1 78 51	1 0 95	1 917 15
M	2 77 32	1 78 37	1 78 51	2 77 97	1 78 51	1 0 99	2 909 05
D	1 73 88	1 78 38	1 78 49	1 74 14	1 78 49	1 0 95	1 24 00

Table 6 2 Results of using desegregation on F1 modified uniform crossover.

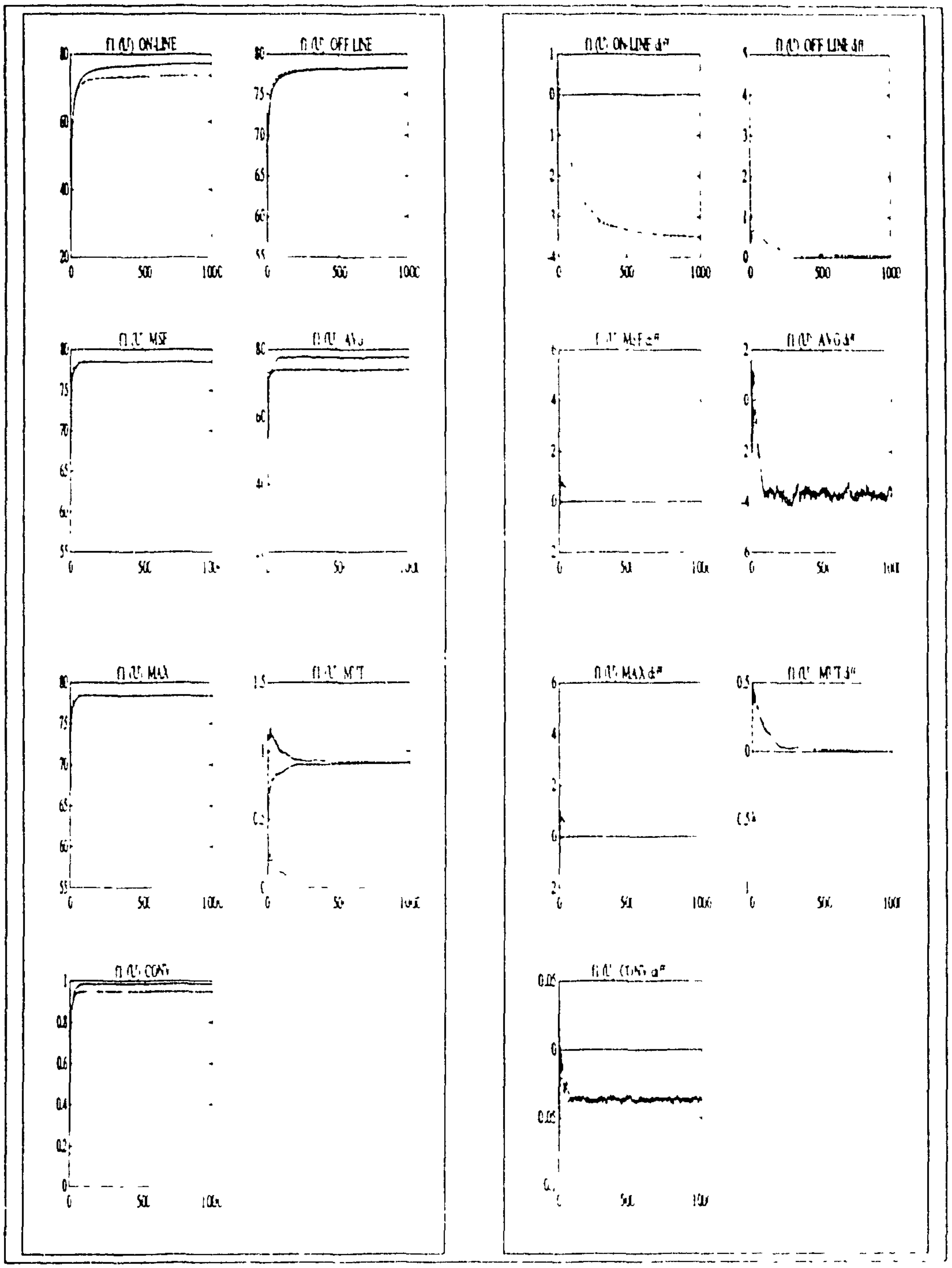


Figure 6.1 Results of F_1 for uniform crossover

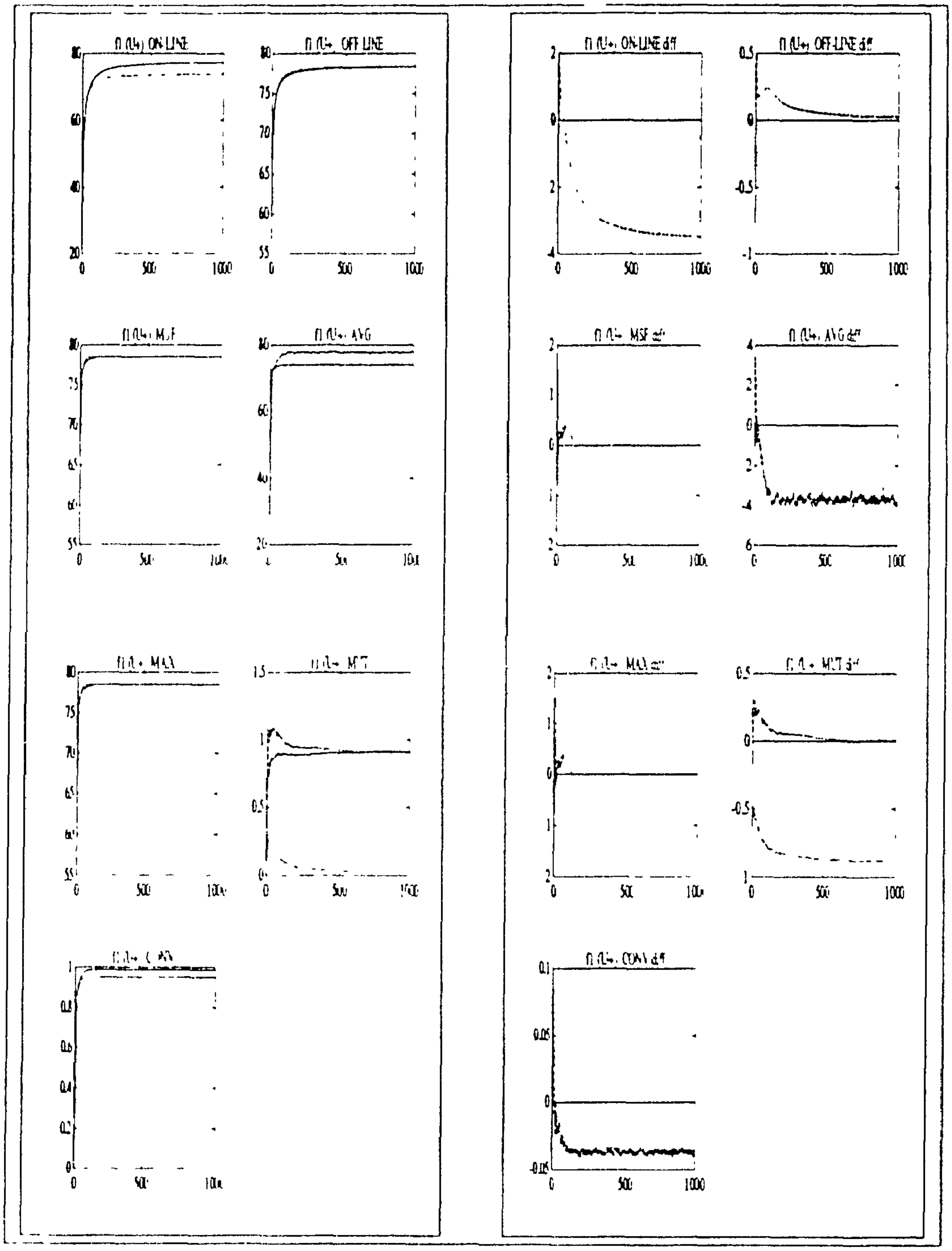


Figure 6.5: Results of F1 for modified uniform crossover

F2							
UNIFORM CROSSOVER							
gen	online	offline	MSF	avg	max	converge	changes
50 M+D	1 1454.80	1 1736.82	1 1761.20	1 1565.75	1 1761.20	1 0.93	3 43.50
M	2 1509.15	1 1736.80	1 1761.33	2 1690.71	1 1761.33	2 0.96	2 36.15
D	1 1473.16	1 1736.07	1 1759.67	1 1584.96	1 1759.67	1 0.91	1 8.05
100 M+D	1 1519.01	1 1749.08	1 1761.95	1 1602.41	1 1761.95	1 0.95	2 75.70
M	2 1613.95	1 1749.41	1 1763.77	1 1729.82	1 1763.77	2 0.98	1 71.65
D	1 1528.09	i 1748.10	1 1760.78	1 1583.80	1 1760.78	1 0.94	1 8.20
200 M+D	1 1551.80	1 1755.76	1 1762.87	1 1594.71	1 1762.87	1 0.95	2 145.95
M	2 1671.85	1 1756.57	1 1763.77	2 1731.42	1 1763.77	2 0.99	2 141.60
D	1 1557.74	1 1754.41	1 1760.78	1 1584.43	1 1760.78	1 0.95	1 10.60
500 M+D	1 1572.99	1 1760.03	1 1762.87	1 1580.75	1 1762.87	1 0.95	2 368.50
M	2 1707.49	1 1760.88	1 1763.77	2 1726.88	1 1763.77	2 0.98	2 365.70
D	1 1573.65	1 1758.35	1 1761.12	1 1573.57	1 1761.12	1 0.95	1 14.85
1000 M+D	1 1579.13	1 1761.45	1 1762.87	1 1589.48	1 1762.87	1 0.95	2 728.75
M	2 1718.43	1 1762.32	1 1763.77	2 1722.21	1 1763.77	2 0.99	2 724.85
D	1 1580.72	1 1759.74	1 1761.12	1 1586.41	1 1761.12	1 0.95	1 20.75

Table 6.3: Results of using desegregation on F2 with uniform crossover.

F2								
MODIFIED UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 1461 37	1 1739 52	1 1759.27	1 1578 17	1 1759.27	1 0 94	3 41 35
	M	2 1527 28	1 1736 66	1 1758 21	2 1667 04	1 1758 21	2 0 96	2 33 80
	D	1 1470 27	1 1735 76	1 1760 41	1 1595 55	1 1760 41	1 0 94	1 7 10
100	M+D	1 1518 22	1 1749 42	1 1760 03	1 1565 47	1 1760 03	1 0.95	3 77 95
	M	2 1615 35	1 1747 44	1 1758 53	2 1712 02	1 1758 53	2 0 98	2 66 60
	D	1 1525 35	1 1748 1995	1 1761 19	1 1574 70	1 1761 19	1 0 95	1 9 60
200	M+D	1 1546 06	1 1754 85	1 1760 96	1 1576 79	1 1760 96	1 0 95	1 152 35
	M	2 1671 29	1 1754 27	1 1761 60	2 1732 16	1 1761 60	2 0 98	2 143 10
	D	1 1550 08	1 1755 27	1 1763 12	1 1563 01	1 1763 12	1 0 95	3 9 90
500	M+D	1 1568 50	1 1758 5970	1 1761 12	1 1587 07	1 1761 12	1 0 95	2 367 80
	M	2 1705 04	1 1758 72	1 1762 35	2 1712 56	1 1762 35	2 0 98	2 3614 45
	D	1 1574 18	1 1759 97	1 1763 12	1 1578 12	1 1763 12	1 0 95	1 13 85
1000	M+D	1 1576 24	1 1759 86	1 1761 12	1 1581 45	1 1761 12	1 0 95	1 733 45
	M	3 1716 52	1 1760 62	1 1762 52	1 1740 43	1 1762 52	2 0.98	2 726 25
	D	2 1582 0958	1 1761 54	1 1732 12	1 1584 26	1 1763 12	1 0.95	1 18.45

Table 6 1 Results of desegregation on F2 with modified uniform crossover.

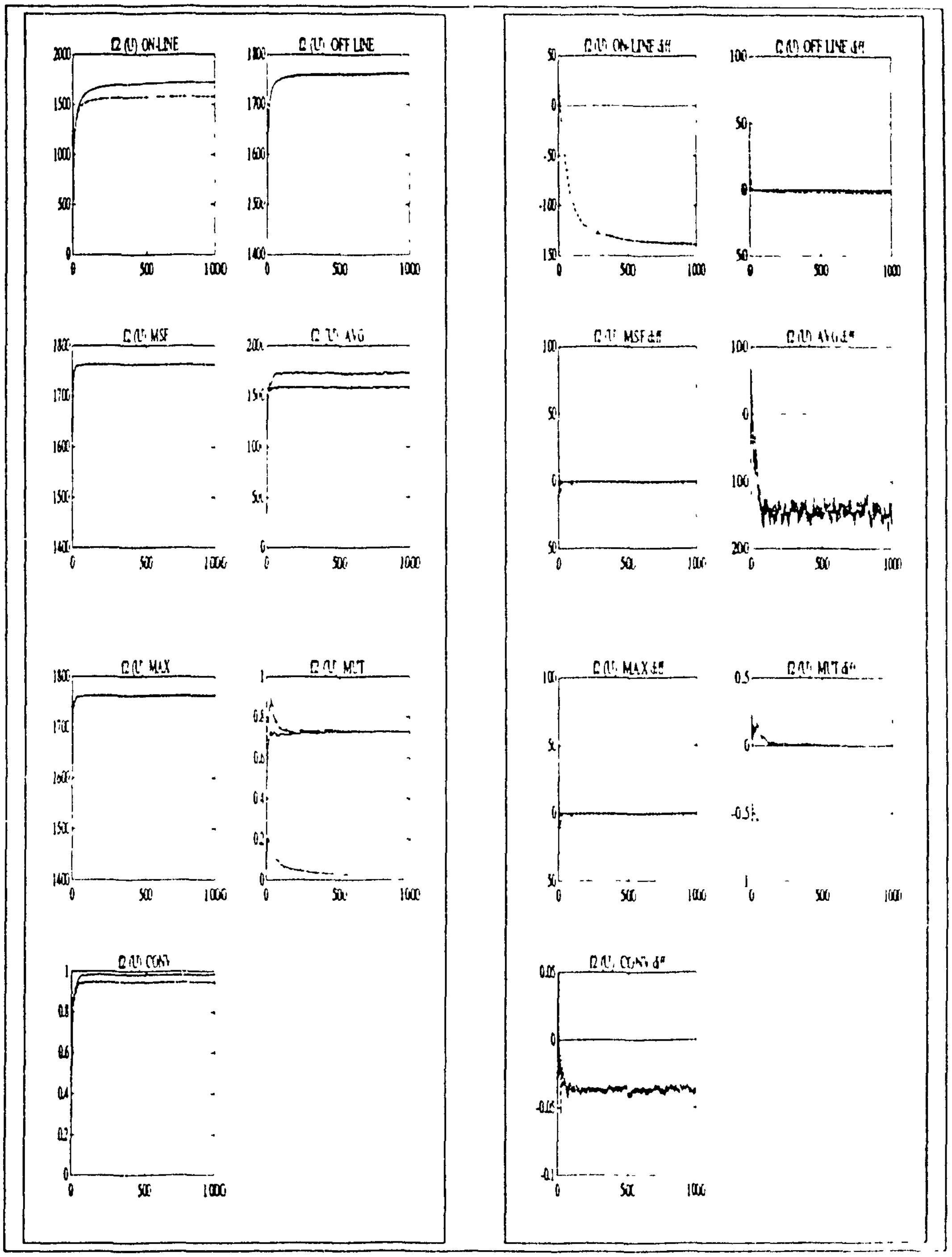


Figure 6.6: Results of F2 for uniform crossover

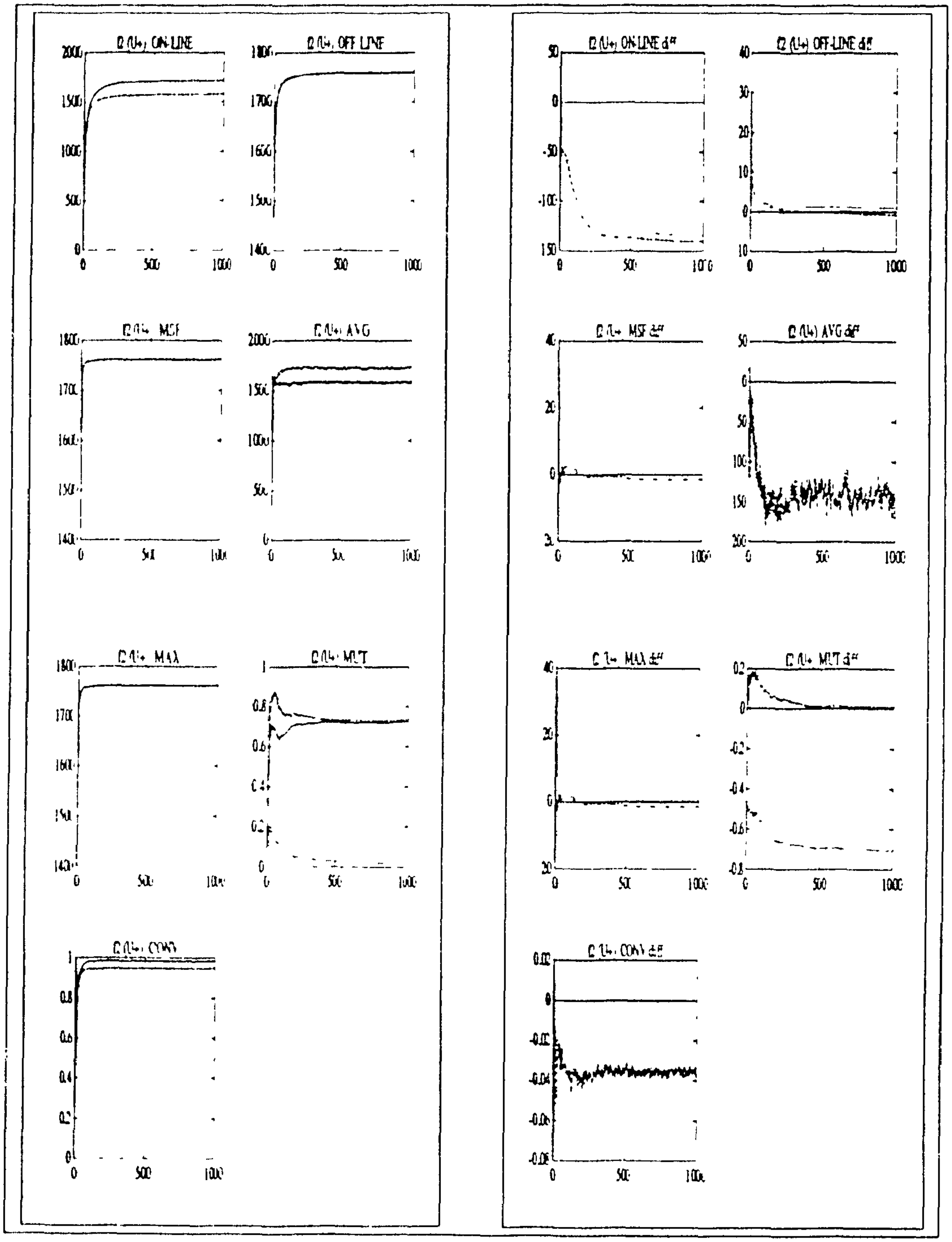


Figure 6.7: Results of F2 for modified uniform crossover.

F3								
UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 19.55	1 22.81	1.5 24.90	1 23.24	1.5 24.90	1 0.79	3 88.05
	M	2 20.55	1 22.83	1 24.70	2 24.49	1 24.70	1 0.79	2 72.65
	D	1 19.84	1 23.01	2 25.00	1 23.42	1 25.00	1 0.79	1 13.15
100	M+D	1 21.47	1 23.89	1 25.00	1 23.17	1 25.00	1 0.77	1 162.70
	M	2 22.57	1 23.81	1 24.90	2 24.76	1 24.90	2 0.78	2 111.90
	D	1 21.64	1 23.9975	1 25.00	1 23.54	1 25.00	1 0.77	1 13.00
200	M+D	1 21.35	1 23.22	1 23.75	1 22.37	1 23.75	1 0.72	2 300.15
	M	2 23.38	1 24.37	1 24.95	2 24.81	1 24.95	1 0.76	2 [illegible]
	D	1.5 22.57	1 24.4963	1 25.00	1.5 23.56	1 25.00	1 0.77	1 16.05
500	M+D	1 21.93	1 23.54	1 23.75	1 22.36	1 23.75	1 0.73	2 729.35
	M	2 24.38	1 24.74	1 25.00	2 24.8960	1 25.00	1 0.76	2 716.60
	D	1.5 23.14	1 24.7979	1 25.00	1.5 23.55	1 25.00	1 0.77	1 22.05
1000	M+D	1 22.12	1 23.64	1 23.75	1 22.30	1 23.75	1 0.73	2 1457.55
	M	1 23.39	1 23.63	1 23.75	1 23.57	1 23.75	1 0.72	2 1430.65
	D	1 23.33	1 24.8989	1 25.00	1 23.51	1 25.00	1 0.77	1 24.15

Table 6.5: Results of using desegregation on uniform crossover for function F3

F3								
MODIFIED UNIFORM CROSSOVER								
gen		online	offline	MSI	avg	max	converge	changes
50	M+D	1 19.71	1 23.01	1 25.00	1 23.47	1 25.00	1 0.73	3 87.89
	M	2 20.62	1 22.95	1 24.90	2 23.47	1 24.90	1 0.77	2 76.00
	D	1 19.85	1 23.01	1 24.95	1 24.66	1 24.95	1 0.72	1 13.45
100	M+D	1 21.57	1 23.99	1 25.00	1 23.48	1 25.00	1 0.77	3 164.21
	M	2 22.70	1 23.96	1 25.00	3 24.87	1 25.00	2 0.79	2 151.00
	D	1 21/6553	1 23.99	1 25.00	2 23.51	1 25.00	1 0.76	111.85
200	M+D	1 21.55	1 23.21	1 25.00	1 22.27	1 25.00	1 0.71	2 29.88
	M	2 23.77	1 24.48	1 25.00	2 24.87	1 25.00	1 0.77	2 299.56
	D	1.5 22.50	1 24.49	25.00	1.5 25.51	1 25.00	1 0.76	1 17.80
500	M+D	1 21.80	1 23.49	1 25.00	1 22.27	1 25.00	1 0.72	2 731.79
	M	2 24.13	1 24.79	1 25.00	2 24.85	1 25.00	1 0.76	2 753.25
	D	1.5 23.15	1 24.80	1 25.00	1.5 23.51	1 25.00	1 0.77	1 23.60
1000	M+D	1 22.07	1 23.59	1 25.00	1 22.26	1 25.00	1 0.72	2 1439.95
	M	2 21.65	1 24.89	1 25.00	2 24.89	1 25.00	1 0.76	2 1508.40
	D	1.5 23.31	1 24.8981	1 25.00	1.5 23.56	1 25.00	1 0.77	1 36.05

Table 6.6 Results of using desegregation on modified uniform crossover for function F3

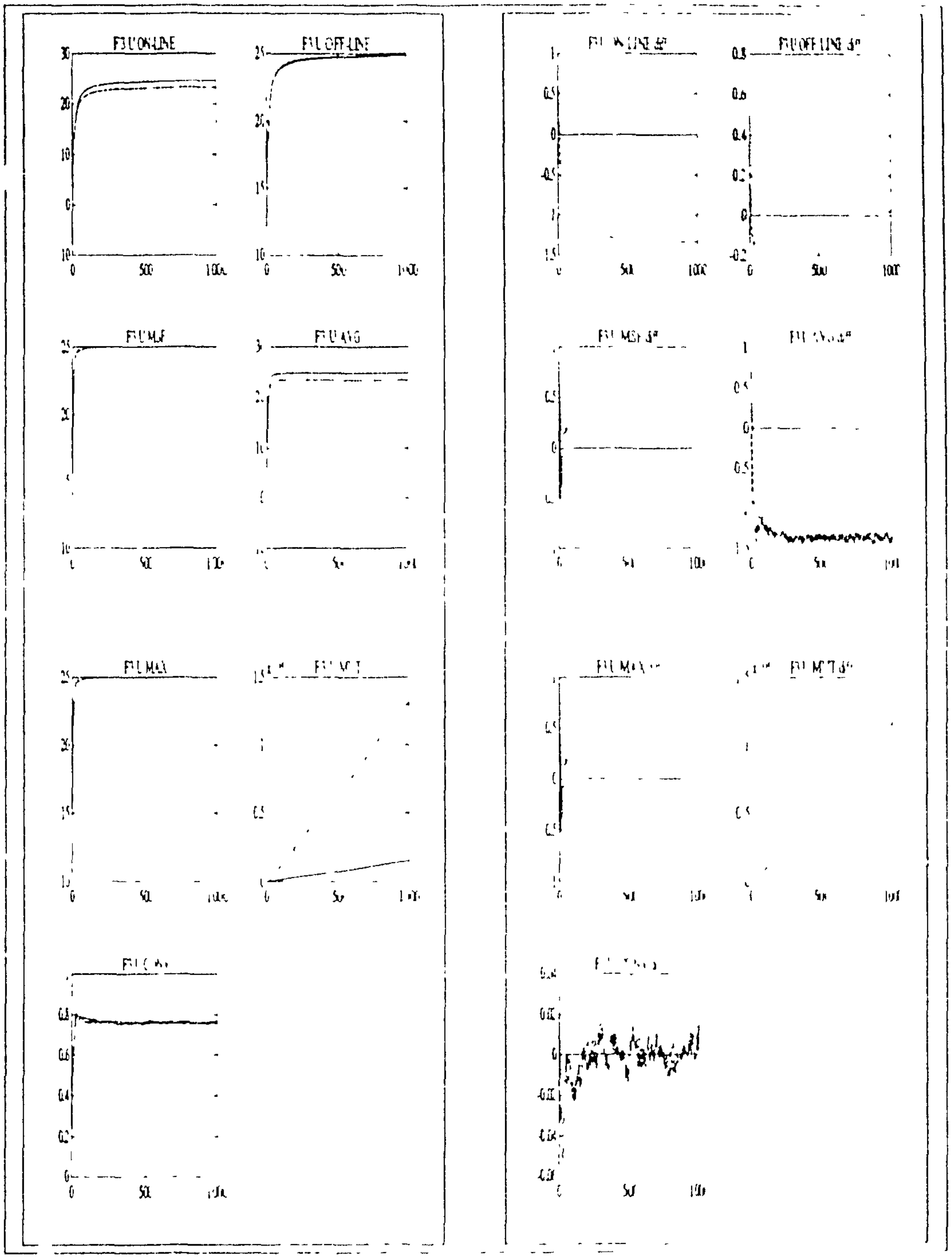


Figure 6.8 Results of F3 Uniform Crossover

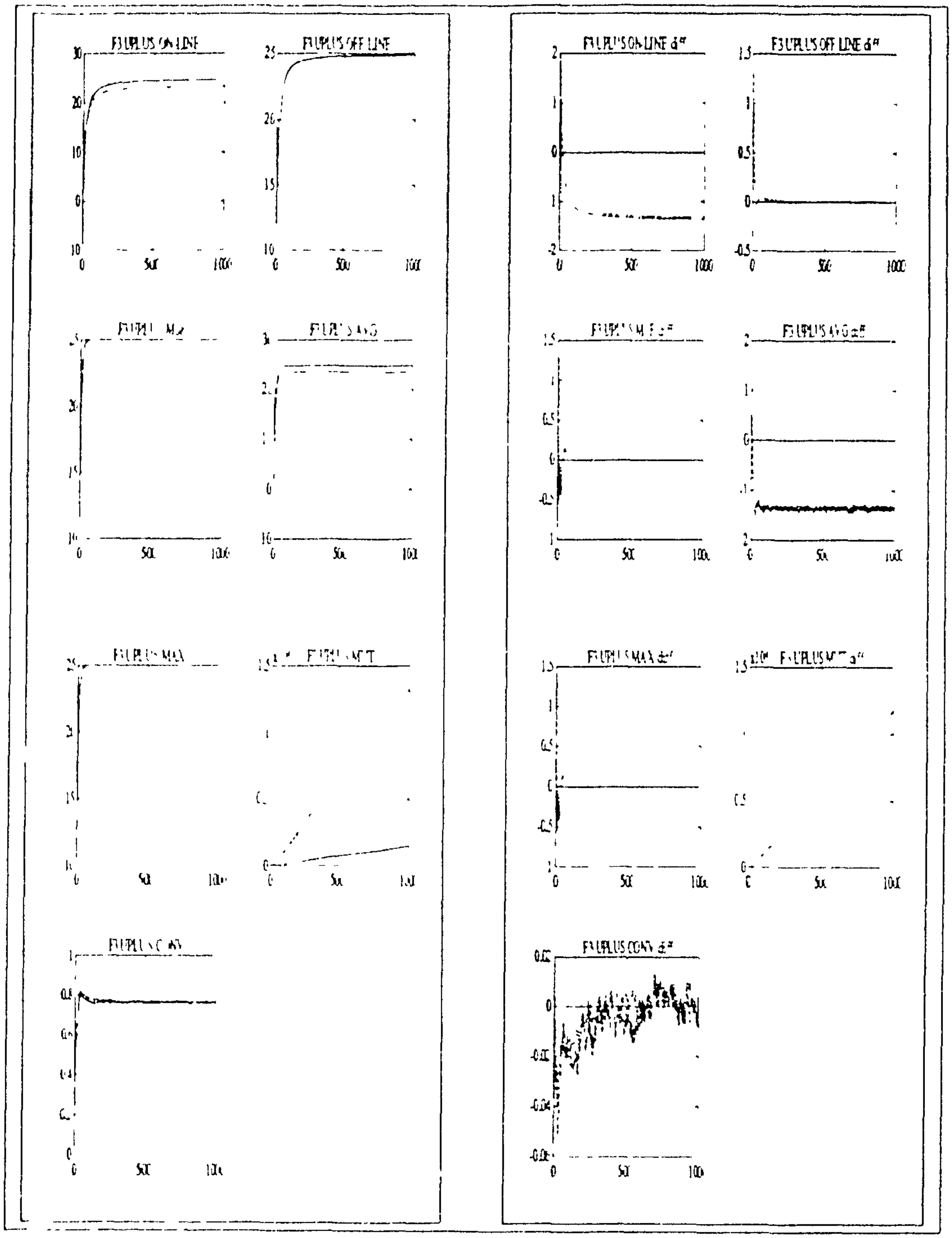


Figure 6.9 Results of F3 for modified uniform cross-over

F1								
UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1.5 26.19	1 29.12	2 33.97	1 32.18	2 33.97	1 0.90	3 2530.90
	M	2 26.73	1 29.04	2 34.28	2 33.36	2 34.28	3 0.96	1 364.55
	D	1 26.00	1 28.84	1 33.23	1 31.53	1 33.23	2 0.91	2 2027.65
100	M+D	1 29.75	2 32.10	2 35.90	2 34.17	2 35.90	1 0.94	3 6846.0
	M	2 30.72	2 32.30	3 36.56	3 35.83	3 36.56	2 0.97	1 729.75
	D	1 29.27	1 31.55	1 34.99	1 33.35	1 34.99	1 0.92	2 5929.75
200	M+D	2 32.30	2 34.33	2 37.06	2 35.39	2 37.06	1 0.94	17001.70
	M	3 33.73	3 34.80	3 37.71	3 37.29	3 37.71	2 0.98	1458.60
	D	1 31.76	1 33.74	1 36.51	1 34.86	1 36.51	1 0.92	15465.15
500	M+D	2 36.08	2 36.69	2 38.07	2 37.85	2 38.07	1 0.99	36660.15
	M	3 34.44	3 36.27	3 37.81	3 36.12	3 37.81	2 0.92	52656.90
	D	1 34.05	1 35.85	1 37.65	1 35.97	1 37.65	1 0.93	49207.85
1000	M+D	2 35.33	2 37.10	1 38.06	1 36.29	1 38.06	1 0.93	3 114131.65
	M	3 36.99	3 37.41	2 38.17	2 37.96	2 38.17	2 0.99	1 72555.36
	D	1 35.13	1 36.87	1 37.99	1 36.29	1 37.99	1 0.93	2 109186.70

Table 6.7 Results of using desegregation on uniform crossover for function F1

F4								
MODIFED UNIFORM CROSSOVER								
gen		online	offline	MSI	avg	max	converge	changes
50	M+D	1 26.17	1 29.10	2 34.20	2 32.42	2 34.20	1 0.90	3 2576.25
	M	2 29.77	1 32.12	2 35.97	3 34.19	2 35.97	2 0.91	1 6699.30
	D	1 25.72	1 28.57	1 33.07	1 31.40	1 33.07	1 0.91	2 2055.00
100	M+D	2 29.77	2 32.12	2 35.97	2 34.19	2 35.97	1 0.91	2 6699.30
	M	3 31.63	2 32.55	3 36.76	3 36.25	3 36.76	2 0.98	1 717.25
	D	1 28.98	1 31.25	1 34.73	1 33.12	1 34.73	1 0.91	2 6199.33
200	M+D	2 32.37	2 34.40	1 37.09	1 35.42	1 37.09	1 0.92	2 16916.00
	M	3 33.98	3 34.95	2 37.73	2 37.34	2 37.73	2 0.98	1 1440.30
	D	1 31.74	1 33.74	1 37.06	1 35.42	1 37.06	1 0.91	2 16538.83
500	M+D	2 34.48	2 36.30	1 37.82	1 36.12	1 37.82	1 0.92	2 52517.90
	M	3 36.21	3 36.76	2 38.07	2 37.84	2 38.07	3 0.99	1 3602.80
	D	1 34.27	1 36.08	1 37.87	1 36.19	1 37.87	2 0.93	2 51410.33
1000	M+D	2 35.35	2 37.12	1 38.02	1 36.30	1 38.02	1 0.93	3 114064.45
	M	3 37.06	3 37.45	2 38.19	2 37.95	2 38.19	2 0.99	1 7240.80
	D	1 35.25	1 37.06	1 37.98	1 36.30	1 37.98	1 0.93	2 111923.50

Table 6.8 Results of using desegregation on modified uniform crossover for function F4

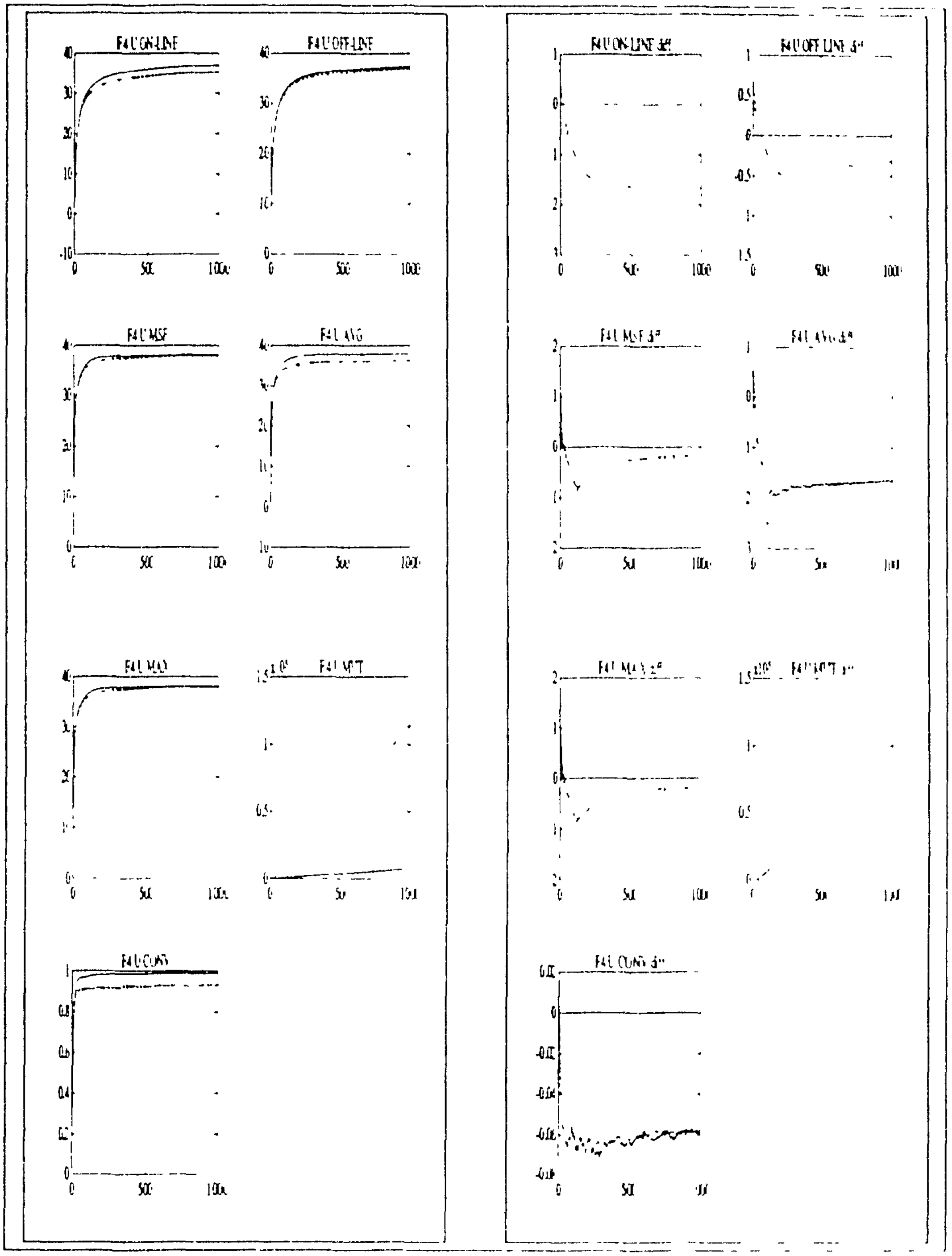


Figure 6.10. Results of F4 for uniform crossover

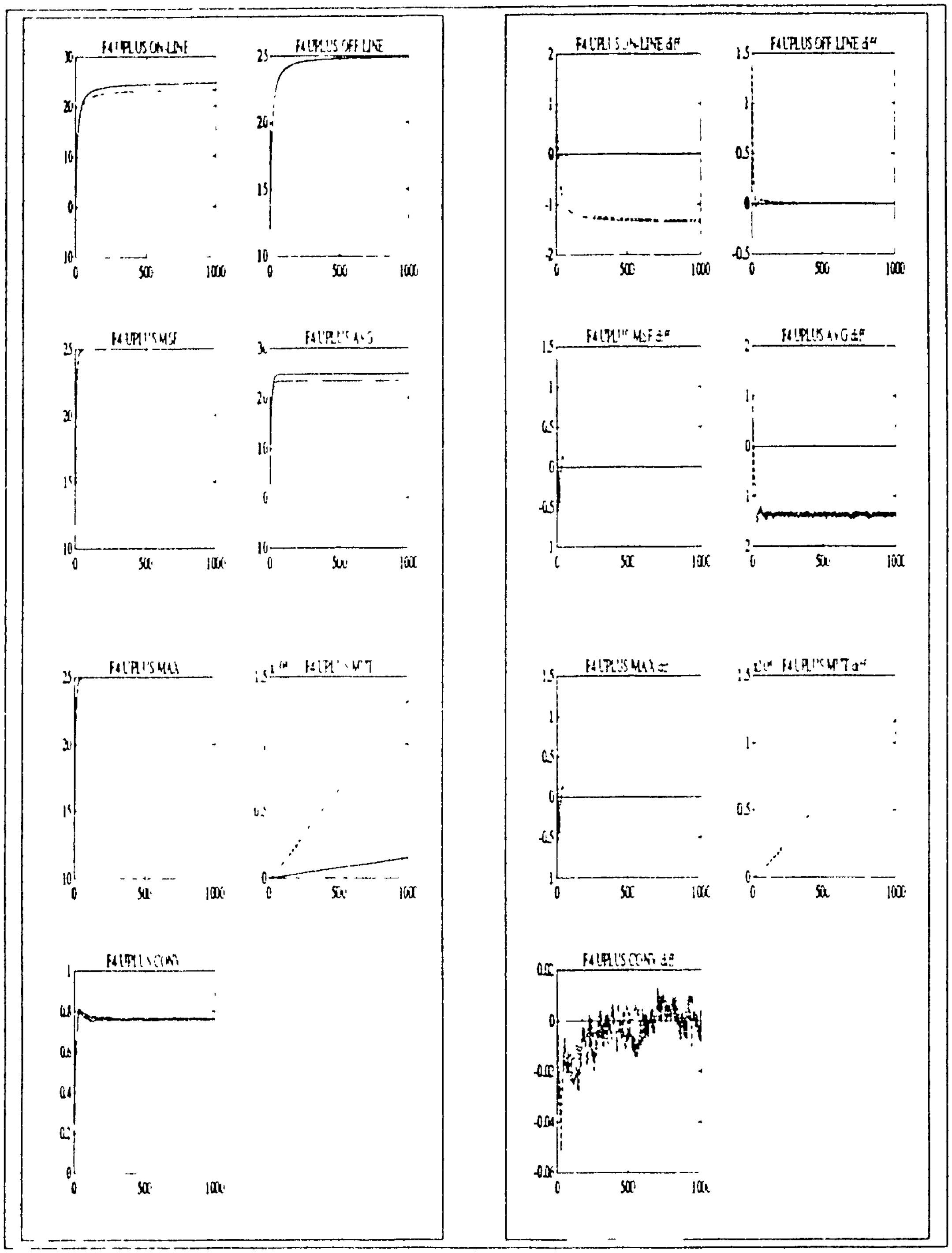


Figure 6.11 Results of F4 for modified uniform crossover.

F5							
UNIFORM CROSSOVER							
gen	online	offline	MSF	avg	max	converge	changes
50 M+D	1 499 03	1 500 00	1 500 00	1 500 00	1 499 50	1 0 50	3 52 50
M	2 499 38	1 500 00	1 500 00	1 500 00	1 500 00	1 0 18	2 47 45
D	1 5 499 23	1 500 00	1 500 00	1 500 00	1 499 50	1 0 49	1 4 25
100 M+D	1 499 41	1 500 00	1 500 00	1 499 31	1 500 00	1 0 18	2 101 25
M	2 499 69	1 500 00	1 500 00	1 500 00	1 500 00	1 0 18	2 97 20
D	1 499 42	1 500 00	1 500 00	1 499 50	1 500 00	1 0 48	1 5 00
200 M+D	1 499 48	1 500 00	1 500 00	1 500 00	1 500 00	1 0 46	2 198 50
M	2 499 84	1 500 00	1 500 00	1 500 00	1 500 00	1 5 0 47	2 196 75
D	1 499 57	1 500 00	1 500 00	1 499 52	1 500 00	2 0 18	1 6 95
500 M+D	1 499 60	1 500 00	1 500 00	1 500 00	1 500 00	1 0 47	2 490 00
M	2 499 93	1 500 00	1 500 00	1 500 00	1 500 00	1 0 18	2 183 40
D	1 499 58	1 500 00	1 500 00	1 500 00	1 500 00	1 0 46	1 11 75
1000 M+D	1 499 63	1 500 00	1 500 00	1 499 60	1 500 00	1 0 48	2 969 30
M	2 499 97	1 500 00	1 500 00	1 500 00	1 500 00	1 0 48	2 967 90
D	1 499 62	1 500 00	1 500 00	1 500 00	1 500 00	1 0 49	1 18 50

Table 6.9: Results of using desegregation on uniform crossover for function F5

F5							
MODIFIED UNIFORM CROSSOVER							
gen	online	offline	MSF	avg	max	converge	changes
50 M+D	1 499.16	1 500.00	1 500.00	1 500.00	1 500.00	1 0.48	2 50.05
M	1 499.28	1 500.00	1 500.00	1 500.00	1 500.00	1 0.48	2 48.25
D	1 498.96	1 500.00	1 500.00	1 498.94	1 500.00	1 0.48	1 4.40
100 M+D	1 499.36	1 500.00	1 500.00	1 499.50	1 500.00	1 0.47	2 95.35
M	2 499.64	1 500.00	1 500.00	2 500.00	1 500.00	1 0.47	2 94.10
D	1 499.33	1 500.00	1 500.00	1 500.00	1 500.00	1 0.48	1 4.20
200 M+D	1 499.48	1 500.00	1 500.00	1 499.51	1 500.00	1 0.49	2 193.80
M	2 499.82	1 500.00	1 500.00	1 500.00	1 500.00	1 0.47	1 192.85
D	1 499.55	1 500.00	1 500.00	1 500.00	1 500.00	1 0.48	1 5.90
500 M+D	1 499.60	1 500.00	1 500.00	1 500.00	1 500.00	0.47	2 490.40
M	2 499.93	1 500.00	1 500.00	1 500.00	1 500.00	1 0.47	2 488.60
D	1 499.61	1 500.00	1 500.00	1 500.00	1 500.00	1 0.48	1 10.85
1000 M+D	1 499.62	1 500.00	1 500.00	1 500.00	1 500.00	1 0.47	2 979.85
M	2 499.96	1 500.00	1 500.00	1 500.00	1 500.00	1 0.48	2 971.35
D	1 499.66	1 500.00	1 500.00	1 500.00	1 500.00	1 0.49	1 17.75

Table 6.10: Results of using desegregation on modified uniform crossover for function F5.

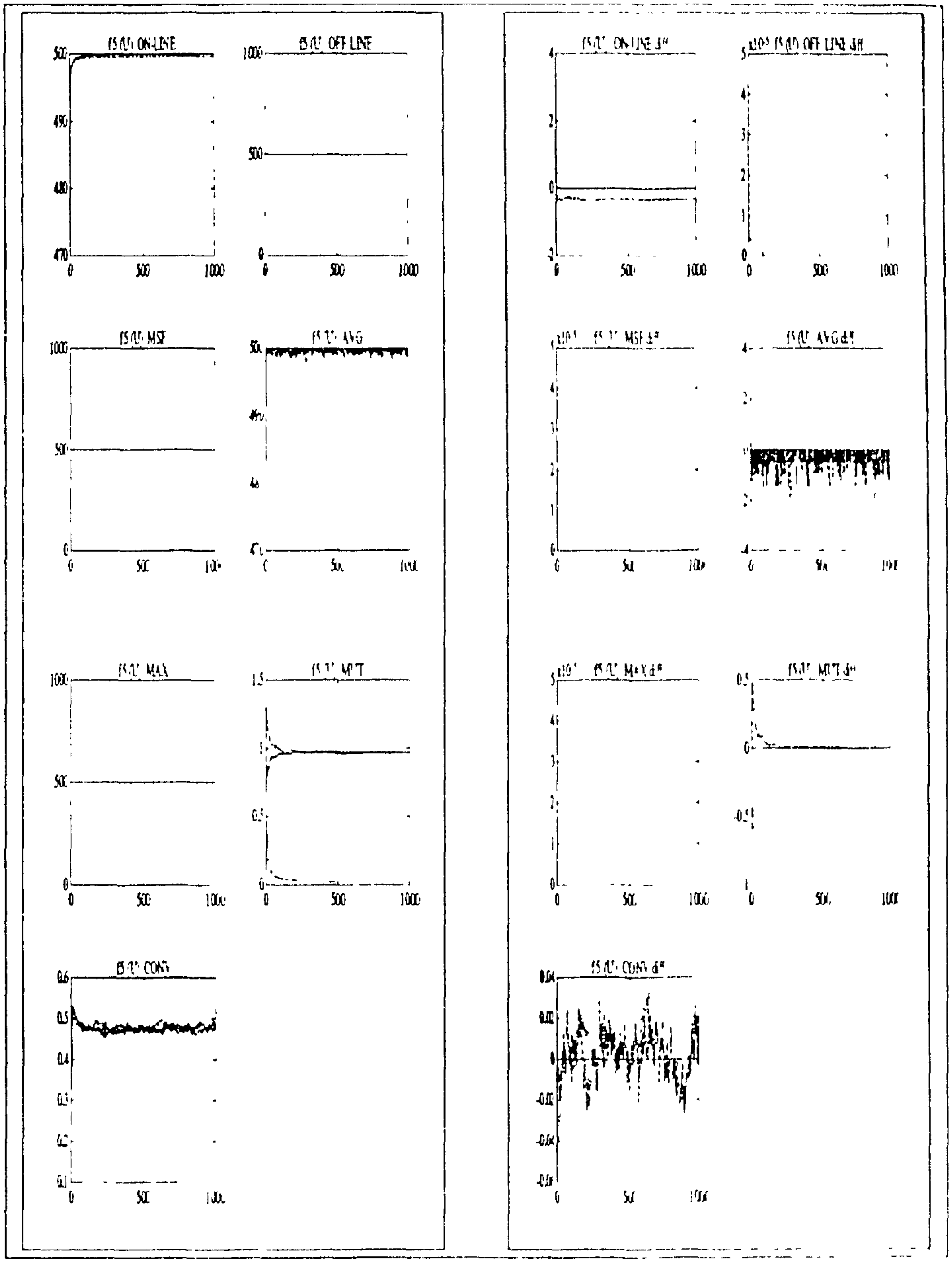


Figure 6.12: Results of F5 for uniform crossover

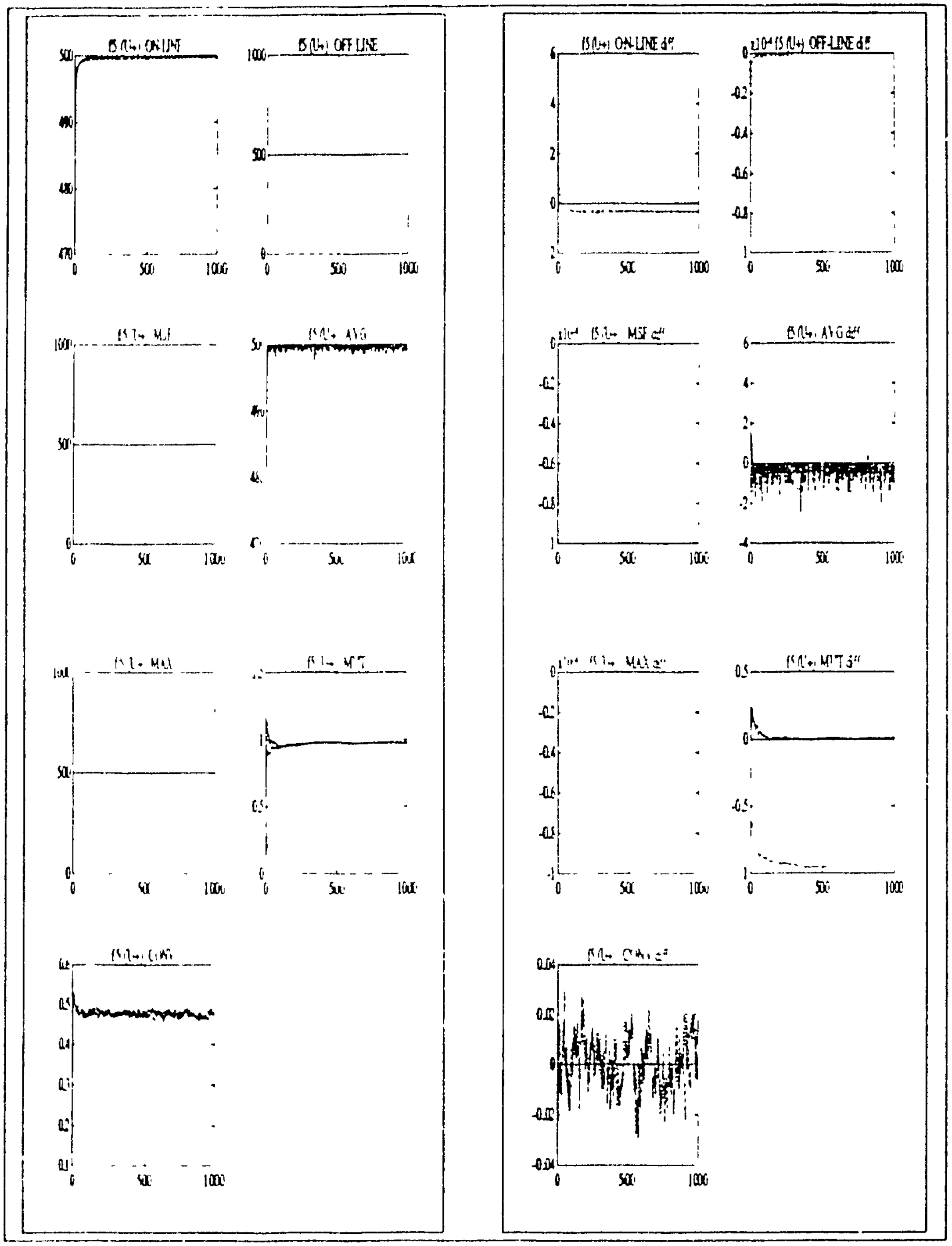


Figure 6.13. Results of F5 for modified uniform crossover.

Continous Ones								
UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 24 50	1 27 69	1 30 00	1 28 42	1 30 00	1 0 94	3 53 70
	M	2 25 02	1 27 6951	1 30 00	2 29 37	1 30 00	2 0 98	2 13 75
	D	1 24 51	1 27 69	1 30 00	1 28 40	1 30 00	1 0 94	1 8 85
100	M+D	1 26 44	1 28 83	1 30 00	1 28 12	1 30 00	1 0 94	3 98 65
	M	2 27 21	1 28 84	1 30 00	2 29 52	1 30 00	2 0 98	2 87 60
	D	1 26 14	1 28 83	1 30 00	1 28 13	1 30 00	1 0 94	1 8 55
200	M+D	1 27 42	1 29 41	1 30 00	1 28 16	1 30 00	1 0 95	3 191 35
	M	2 28 33	1 29 42	1 30 00	2 29 48	1 30 00	2 0 98	2 175 20
	D	1 27 42	1 29 41	1 30 00	1 28 39		1 0 94	1 8 90
500	M+D	1 28 02	1 29 76	1 30 00	1 28 11	1 30 00	0 94	2 462 85
	M	2 29 01	1 29 77	1 30 00	2 29 45	1 30 00	2 0 98	2 429 50
	D	1 28 02	1 29 77	1 30 00	1 28 34	1 30 00	0 94	1 14 85
1000	M+D	1 28 22	1 29 88	1 30 00	1 28 41	1 30 00	1 0 95	2 913 70
	M	2 29 23	1 29 88	1 30 00	2 29 49	1 30 00	2 0 98	2 870 45
	D	1 28 22	1 29 88	1 30 00	1 28 38	1 30 00	1 0 94	1 20 50

Table 6.11: Results of using desegregation on uniform crossover for function Continous Ones.

Continuos Ones

MODIFIED UNIFORM CROSSOVER

gen		online	offline	MSI	avg	max	converge	changes
50	M+D	1 23 76	1 26 87	1 29 05	1 27 46	1 29 05	1 0 94	3 54.30
	M	3 24 92	2 27 51	2 29 70	3 29 28	2 29 70	2 0 98	2 45.95
	D	2 24 25	2 27 45	2 29 70	2 28 10	2 29 70	1 0 94	1 7.90
100	M+D	1 25 59	1 27 95	1 29 05	1 27 44	1 29 05	1 0 94	3 98 55
	M	3 27 07	2 28 59	2 29 70	3 29 30	2 29 70	2 0 99	2 91 35
	D	2 26 17	2 28 56	2 29 70	2 28 11	2 29 70	1 0 94	1 9 25
200	M+D	1 26 52	1 28 50	1 29 05	1 27 43	1 29 05	1 0 94	3 190 40
	M	3 28 16	2 29 14	2 29 70	3 29 28	2 29 70	3 0 99	2 180 10
	D	2 27 14	2 29 13	2 29 70	2 28 13	2 29 70	2 0 94	1 10 90
500	M+D	1 27 08	1 28 83	1 29 05	1 27 43	1 29 05	1 0 94	2 461 05
	M	3 28 82	2 29 48	2 29 70	3 29 30	2 29 70	2 0 99	2 452 15
	D	2 27 74	2 29 47	2 29 70	2 28 12	2 29 70	1 0 94	1 14 50
1000	M+D	1 27 27	1 28 94	1 29 65	1 27 48	1 29 65	1 0 94	2 909 65
	M	3 29 03	2 29 59	2 29 70	3 29 25	2 29 70	3 0 98	2 905 15
	D	2 27 93	2 29 59	2 29 70	2 28 17	2 29 70	2 0 95	1 21 50

Table 6 12 Results of using desegregation on modified uniform crossover for function Continous Ones

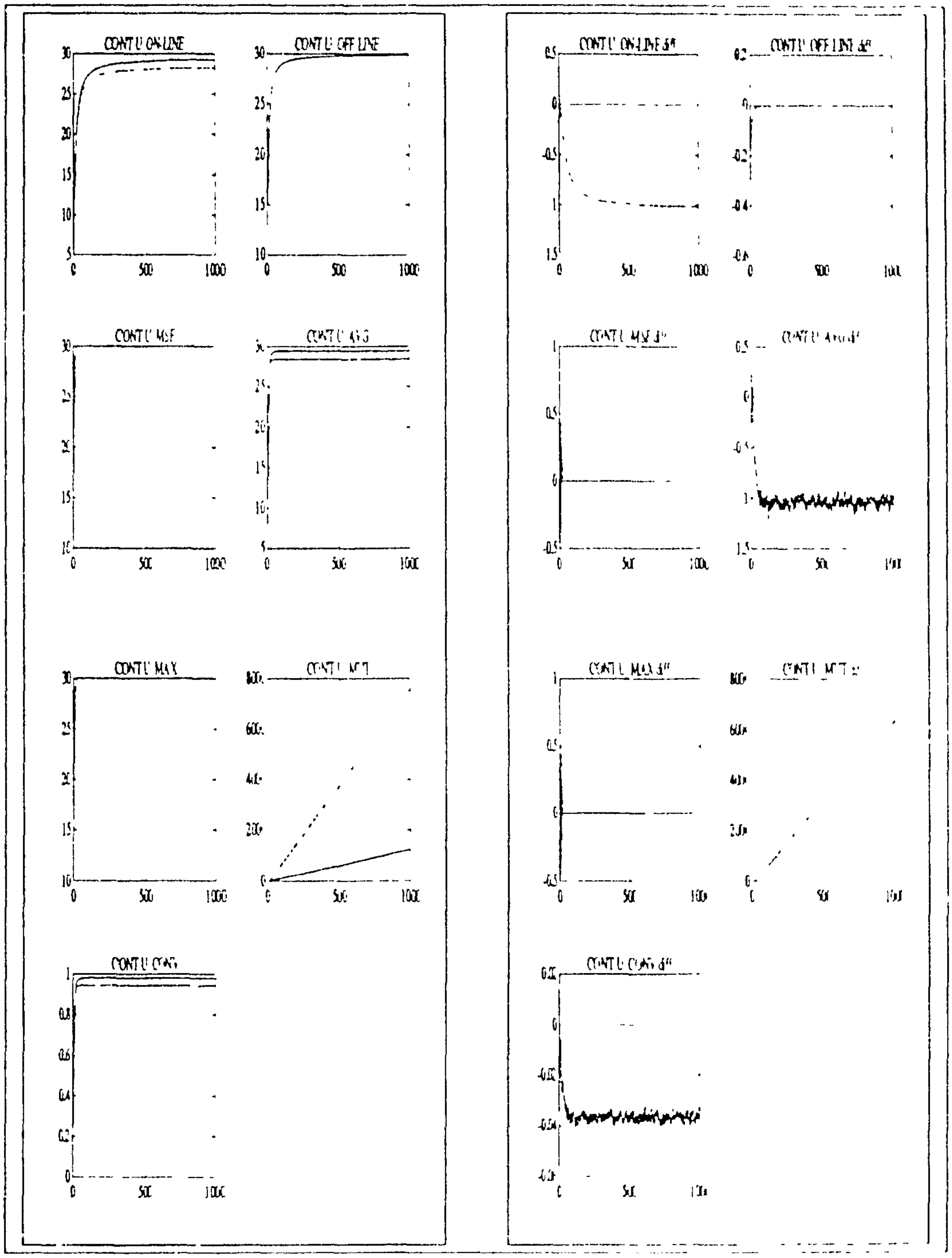


Figure 6.11 Results of Continuous Ones for uniform crossover

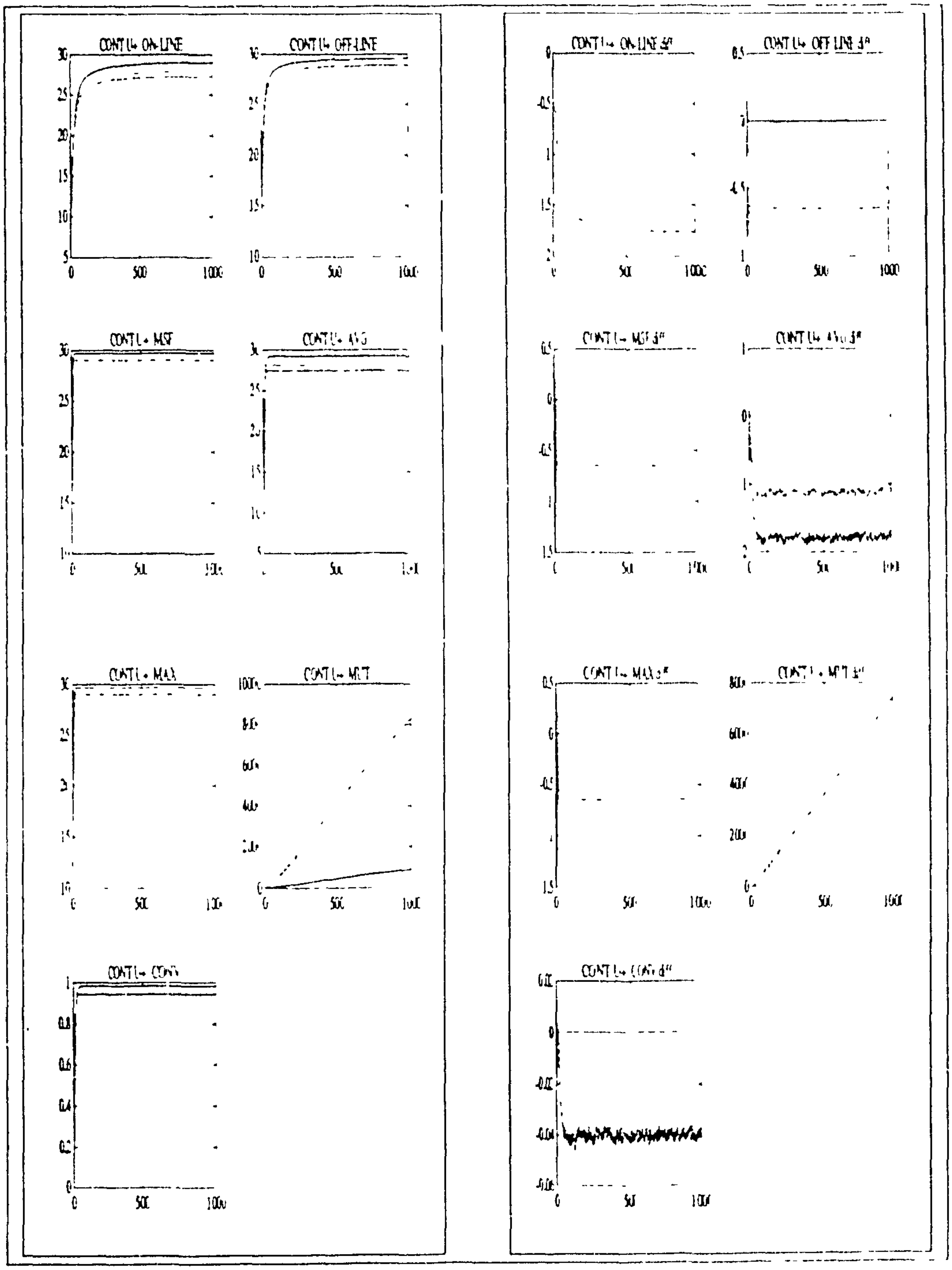


Figure 6.15. Results of Continuous Ones for modified uniform cross-over

F5'							
UNIFORM CROSSOVER							
gen	online	offline	MSF	avg	max	converge	changes
50 M+D	1 0 49	1 0.66	1 0.82	1 0 65	1 0.82	1 0 86	3 61.35
M	1 0.4994	1 0 63	1 0 74	1 0 66	1 0 74	1 0 89	2 48.45
D	1 0 53	1 0 70	1 0 8	1 0 66	1 0 83	1 0.88	1 10 30
100 M+D	1 0 58	1 0 75	1 0 87	1 0 68	1 0 87	1 0 81	3 110 25
M	1 0 62	1 0 70	1 0 89	1 0 78	1 0 81	2 0 89	2 96 80
D	1 0 60	1 0 77	1 0 86	1 0 68	1 0 86	1 0 84	1 13 15
200 M+D	1 0 64	1 0 81	1 0 88	1 0 71	1 0 88	1 0 80	3 208 00
M	1 0 71	1 0 76	1 0 81	1 0 78	1 0 81	1 0 84	2 193 80
D	1 0 65	1 0 82	1 0 87	1 0 70	1 0 87	1 0 82	1 15 30
500 M+D	1 0 68	1 0 85	1 0 88	1 0 71	1 0 88	1 0.79	3 501 70
M	1 0 76	1 0 79	1 0 81	1 0 79	1 0 81	1 0 7989	2 480 75
D	1 0 68	1 0 85	1 0 87	1 0.70	1 0 87	1 0 84	1 18 30
1000 M+D	1 0 69	1 0 87	1 0 88	1 0 71	1 0 88	1 0.79	3 984.30
M	1 0 78	1 0 7991	1 0.81	1 0 79	1 0 81	1 0.84	2 963 25
D	1 0 69	1 0 86	1 0 87	1 0 70	1 0 87	1 0.80	1 26.65

Table 6 13: Results of using desegregation on uniform crossover for function F5'.

F5'							
MODIFIED UNIFORM CROSSOVER							
gen	online	offline	MSF	avg	max	converge	changes
50 M+D	2 0 57	2 0 77	2 0 87	2 0 71	2 0 87	1 0 85	3 58 50
M	1 0 44	1 0 54	1 0 60	1 0 57	1 0 60	2 0 92	2 48 25
D	1 5 0 49	1 5 0 67	2 0 7974	1 5 0 63	2 0 7974	2 0 89	1 11 25
100 M+D	1 0 64	2 0 83	2 0 91	1 0 73	2 0 91	1 0 81	3 108 10
M	1 0 52	1 0 59	1 0 69	1 0 67	1 0 69	2 0 89	2 96 20
D	1 0 57	2 0 74	2 0 81	1 0 65	2 0 81	1 5 0 86	1 13 55
200 M+D	1 0 68	2 0 87	2 0 91	1 0 74	2 0 91	1 0 79	3 202 20
M	1 0 62	1 0 67	1 0 76	1 0 75	1 0 76	2 0 89	2 190 80
D	1 0 61	1 5 0 78	1 5 0 83	1 0 66	1 5 0 83	1 5 0 81	1 15 70
500 M+D	1 0 71	2 0 89	2 0 91	1 0 74	2 0 91	1 0 78	2 491 40
M	1 0 66	1 0 69	1 0 74	1 0 72	1 0 74	1 0 82	2 456 20
D	1 0 65	2 0 82	2 0 85	1 0 68	2 0 85	1 0 82	1 19 50
1000 M+D	1 0 72	2 0 75	2 0 91	1 5 0 73	2 0 96	1 0 77	2 978 40
M	1 007292	1 0 75	1 0 81	2 0 79	1 0 81	2 0 85	2 963 30
D	1 0 66	1 5 0.83	1 0 85	1 0 68	1 5 0 85	1 5 0 82	1 27 80

Table 6.14: Results of using desegregation on modified uniform crossover for function F5'.

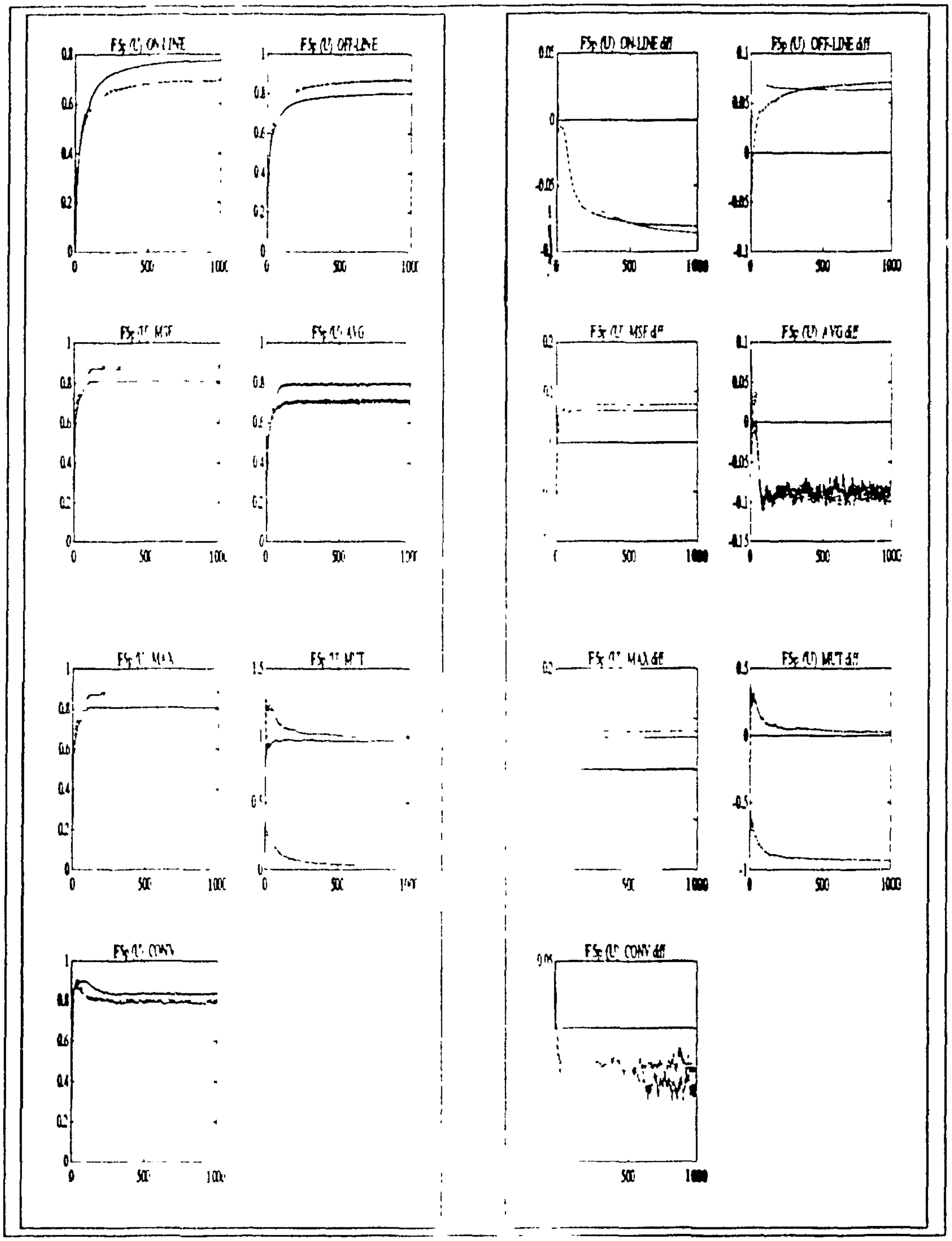


Figure 6.16: Results of [illegible] uniform crossover.

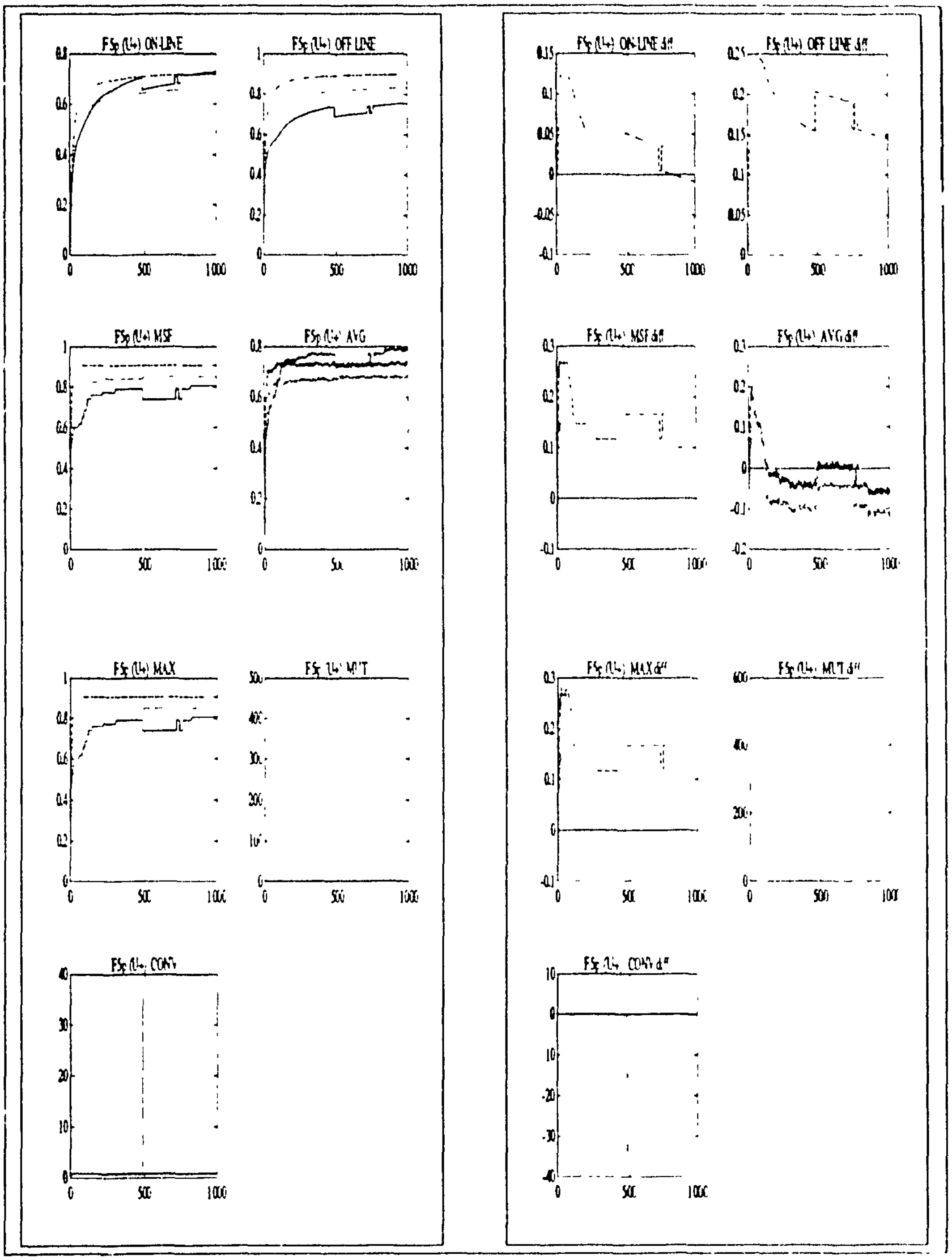


Figure 6.17: Results of F5 for modified uniform crossover

ONE MAX								
UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 26 66	1 28 68	1 30 00	1 29 05	1 30 00	1 0.94	3 265.35
	M	2 27 12	1 28 76	1 30 00	2 29 68	1 30 00	2 0 98	1 46 55
	D	1 26 80	1 28 78	1 30 00	1 29 10	1 30 00	1 0 94	2 239 95
100	M+D	1 27 86	1 29 33	1 30 00	1 29 08	1 30.00	1 0 94	3 646 50
	M	3 28 41	1 29 37	1 30 00	3 29 73	1 30.00	3 0 98	1 92 80
	D	2 27 96	1 29 38	1 30 00	2 29 14	1 30 00	2 0 94	2 600 15
200	M+D	1 28 48	1 29 66	1 30 00	1 29 10	1 30 00	1 0 94	3 1401 45
	M	3 29 07	1 29 69	1 30 00	2 29 71	1 30 00	2 0 98	2 187 05
	D	2 28 71	1 29 69	1 30 00	1 29 15	1 30 00	1 0 94	1 1318 70
500	M+D	1 28 85	[illegible] 29 87	1 30 00	1 29 11	1 30 00	1 0 94	3 3704 25
	M	3 29 47	1 29 87	1 30 00	2 29 73	1 30 00	2 0.98	1 458 55
	D	2 28 89	1 29 88	1 30 00	1 29 12	1 30 00	1 0 94	2 3449 15
1000	M+D	1 28 97	1 29 93	1 30 00	1 29 12	1 30 00	1 0.94	3 7507 05
	M	3 29 60	1 29 91	1 30 00	2 29 76	1 30 00	2 0 98	1 908 50
	D	2 29 01	1 29 94	1 30 00	1 29 15	1 30 00	1 0 94	2 6979 55

Table 6.15: Results of using desegregation on uniform crossover for function One Max.

ONE MAX								
MODIFIED UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 26 76	1 28 77	1 30 00	1 29 11	1 30 00	1 0 94	3 276 65
	M	2 27 04	1 28 72	1 30 00	2 29 74	1 30 00	1 0 98	1 46 10
	D	1 26 77	1 28 78	1 30 00	1 29 09	1 30 00	2 0 94	2 235 30
100	M+D	1 27.92	1 29 38	1 30 00	1 29 10	1 30 00	1 0 94	3 653 25
	M	2 28 37	1 29 35	1 30 00	2 29 71	30 00	2 0 98	1 93 35
	D	1 27 93	1 29 38	1 30 00	1 29 13	1 30 00	1 0 91	2 588 50
200	M+D	1 28 51	1 29 69	1 30 00	1 29 07	1 30 00	1 0 91	2 1126 85
	M	2 29 05	1 29 68	1 30 00	2 29 75	1 30 00	2 0 98	1 185 50
	D	1 28 53	1 29 69	1 30 00	1 29 12	1 30 00	1 0 94	2 1299 95
500	M+D	1 28 86	1 29 87	1 30 00	1 29 09	1 30 00	1 0 91	3 3728 00
	M	3 29 46	1 29 87	1 30 00	2 29 74	1 30 00	2 0 98	1 459 65
	D	2 28 89	1 29 88	1 30 00	1 29 14	1 30 00	1 0 91	2 3425 05
1000	M+D	1 28 98	1 29 94	1 30 00	1 29 08	1 30 00	1 0 91	3 7516 75
	M	2 29.60	1 29 93	1 30 00	3 29 73	1 30 00	2 0 98	1 918 80
	D	2 29 01	1 29 94	1 30 00	2 29 14	1 30 00	2 0 94	2 6981 55

Table 6.16: Results of using desegregation on modified uniform crossover for function One Max.

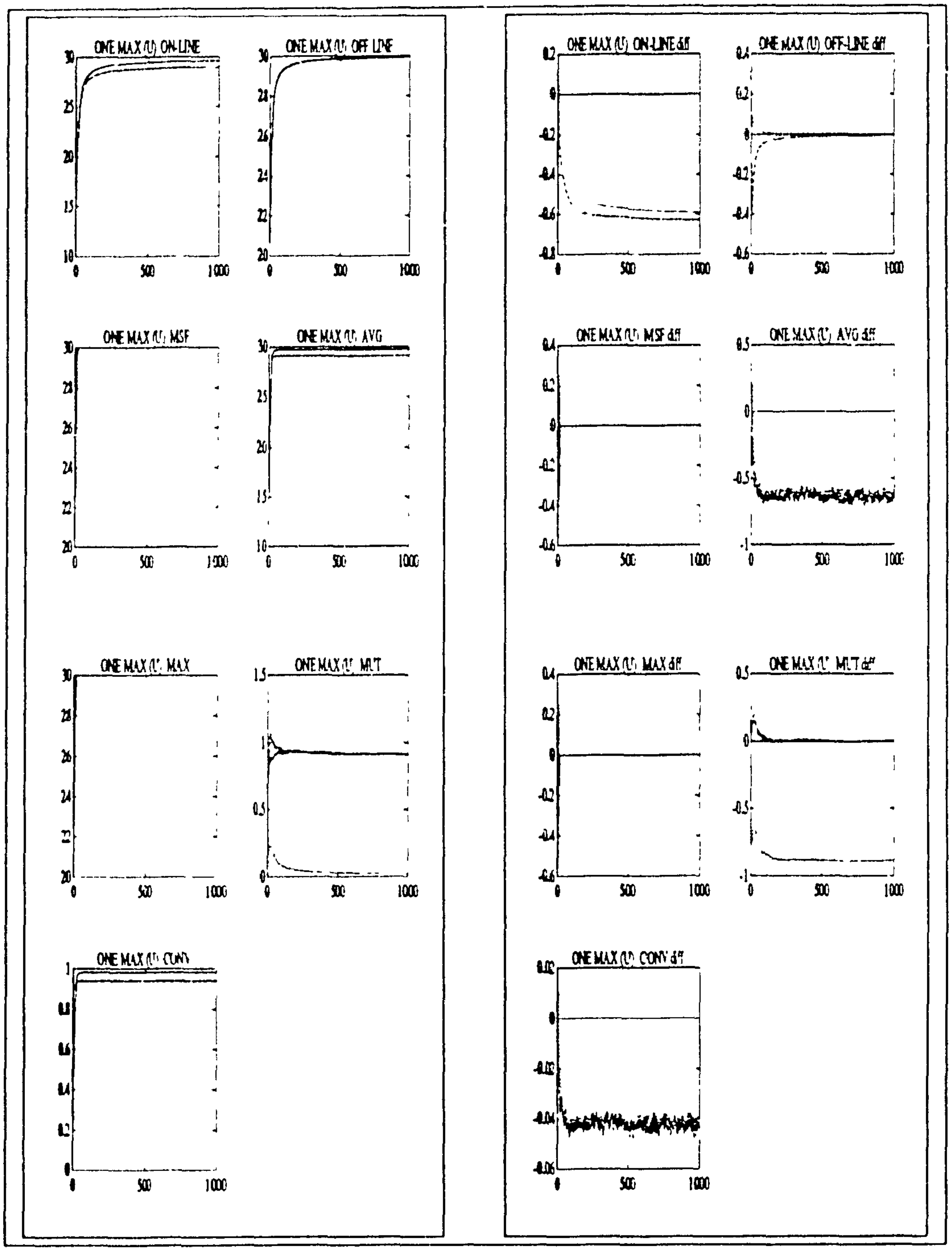


Figure 6.18: Results of One Max for uniform crossover.

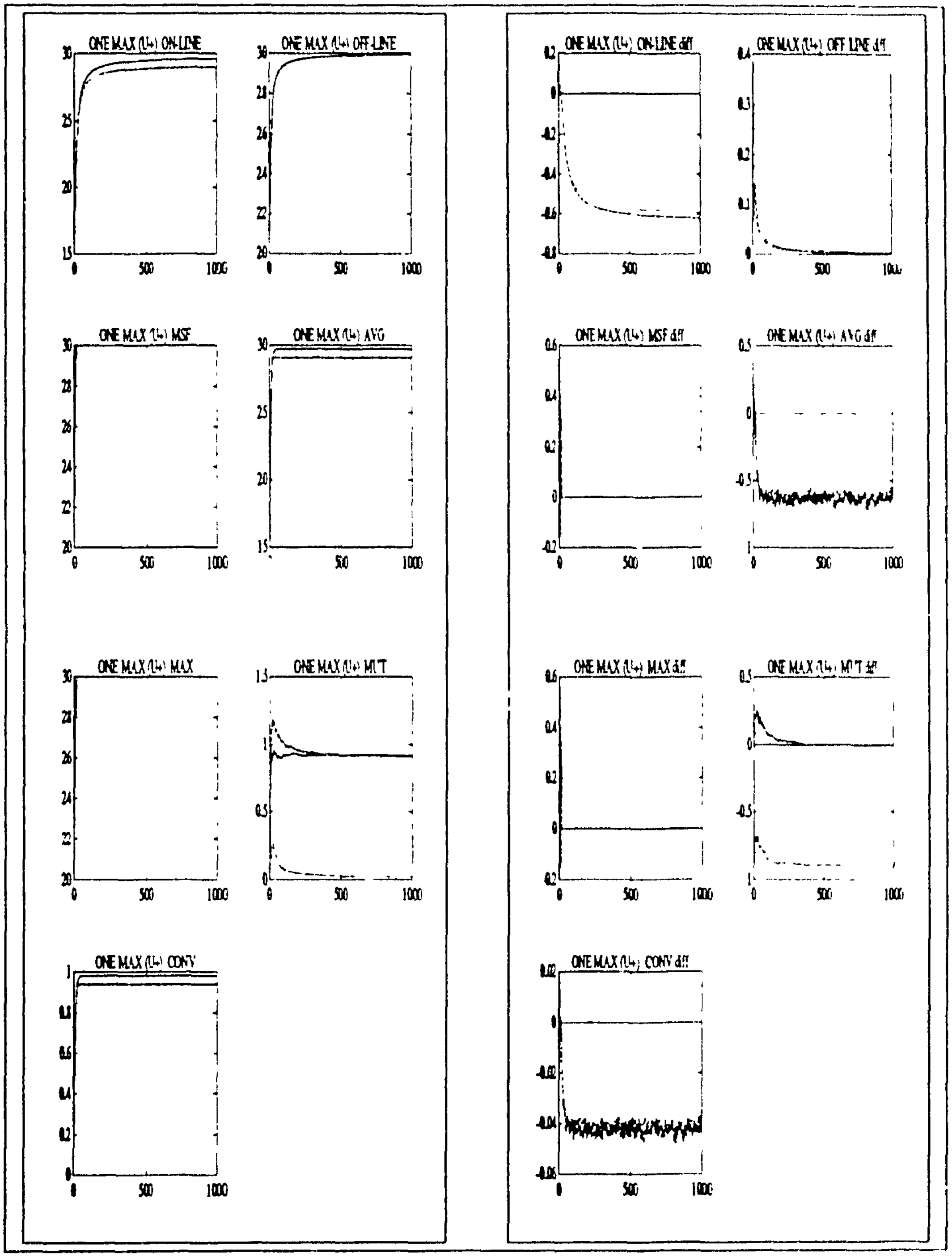


Figure 6.19: Results of One Max for modified uniform crossover.

SPARSE ONE MAX							
UNIFORM CROSSOVER							
gen	online	offline	MSF	avg	max	converge	changes
50 M+D	1 26 88	1 28 69	1 30.00	1 29.33	1 30.00	1 0.60	3 479.25
M	2 27.30	1 28 71	1 30.00	3 29 95	1 30.00	1 0.64	2 457.10
D	1 26 86	1 28 67	1 30 00	2 29 36	1 30.00	1 0.61	1 33.30
100 M+D	1 28.09	1 29 34	1 30.00	1 29.34	1 30.00	1 0.62	3 938.30
M	2 28 62	1 29 35	1 30.00	2 29.96	1 30 00	2 0 66	2 905 15
D	1 28 09	1 29 33	1 30.00	1 29.33	1 30 00	2 0 63	1 41 30
200 M+D	1 28 71	1 29.67	1 30 00	1 29.32	1 30 00	1 0 62	3 1836 65
M	2 28 28	1 29 67	1 30 00	2 29.95	1 30.00	3 0 71	2 1809.20
D	1 28 71	1 29 66	1 30 00	1 29 34	1 30.00	2 0.64	1 55.70
500 M+D	1 29.08	1 29 87	1 30.00	1 29 32	1 30 00	1 0.63	3 4568.75
M	2 '9.69	1 29 87	1 30.00	3 29 95	1 30.00	2 0.73	2 4518.95
D	1 29 09	1 29.86	1 30.00	2 29 35	1 30.00	1 0.64	1 95.10
1000 M+D	1 29 20	1 29 93	1 30 00	1 29.34	1 30 00	1 0.63	2 9096.95
M	2 29.82	1 29.93	1 30.00	2 29.95	1 30.00	3 0.73	2 9061.20
D	2 29.22	1 29.93	1 30.00	1 29.35	1 30.00	2 0.65	1 163.10

Table 6.17: Results of using desegregation on uniform crossover for function Sparse One Max.

SPARSE ONE MAX

MODIFIED UNIFORM CROSSOVER

gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 26.74	1 28 55	1 30.00	1 29 33	1 30.00	1 0 60	2 477 10
	M	2 27.27	1 28 69	1 30 00	2 29 96	1 30 00	1 0 61	2 453 65
	D	1 26.83	1 28 66	1 30 00	1 29 35	1 30 00	1 0 61	1 54 00
100	M+D	1 28 02	1 29.27	1 30.00	1 29 34	1 30 00	1 0 62	2 931 90
	M	2 28 60	1 29 3	1 30.00	2 29 95	1 30 00	2 0 67	2 906 85
	D	1 28.08	1 29 32	1 30 00	1 29 31	1 30 00	1 0 63	1 86 45
200	M+D	1 28 67	1 29 63	1 30 00	1 29 32	1 30 00	1 0 63	2 1852 85
	M	2 29 27	1 29 67	1 30 00	2 29 95	1 30 00	3 0 71	2 1806 70
	D	1 28 70	1 29 66	1 30 00	1 29 32	1 30 00	2 0 64	1 140 30
500	M+D	1 29 06	1 29.85	1 30 00	1 29 34	1 30 00	1 0 63	2 4536 85
	M	3 29 68	1 29.87	1 30 00	2 29 96	1 30 00	3 0 72	2 4536 55
	D	2 29 09	1 29.86	1 30.00	1 29 34	1 30.00	2 0 64	1 318 85
1000	M+D	1 29.19	1 29.93	1 30 00	1 29 35	1 30 00	1 0.63	2 9054 95
	M	3 29 82	1 29 93	1 30.00	2 29 95	1 30.00	2 0.72	2 9062 50
	D	2 29 21	1 29.93	1 30.00	1 29 35	1 30 00	1 0.64	1 606 30

Table 6.18: Results of using desegregation on modified uniform crossover for function Sparse One Max.

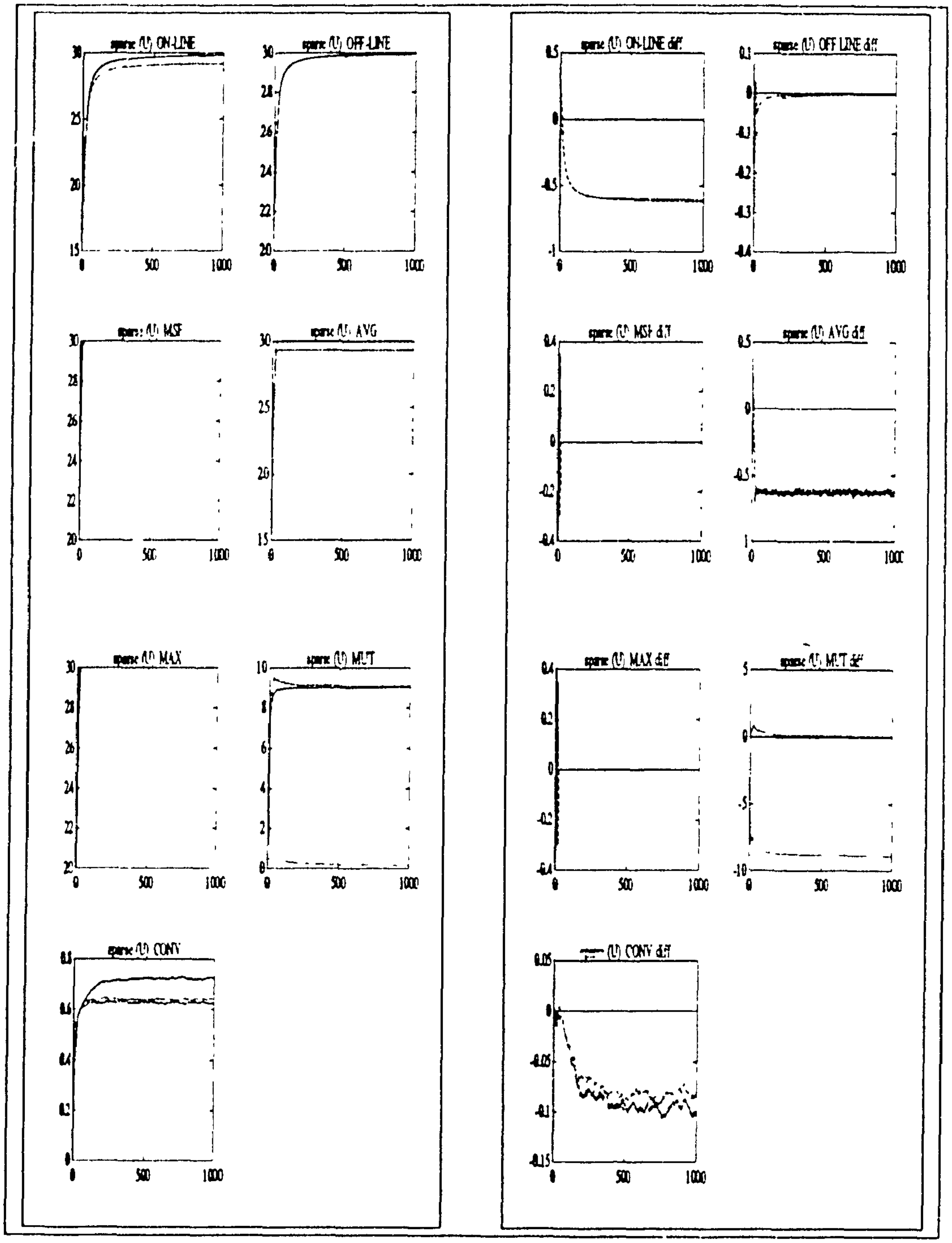


Figure 6.20: Results of Sparse One Max for uniform crossover.

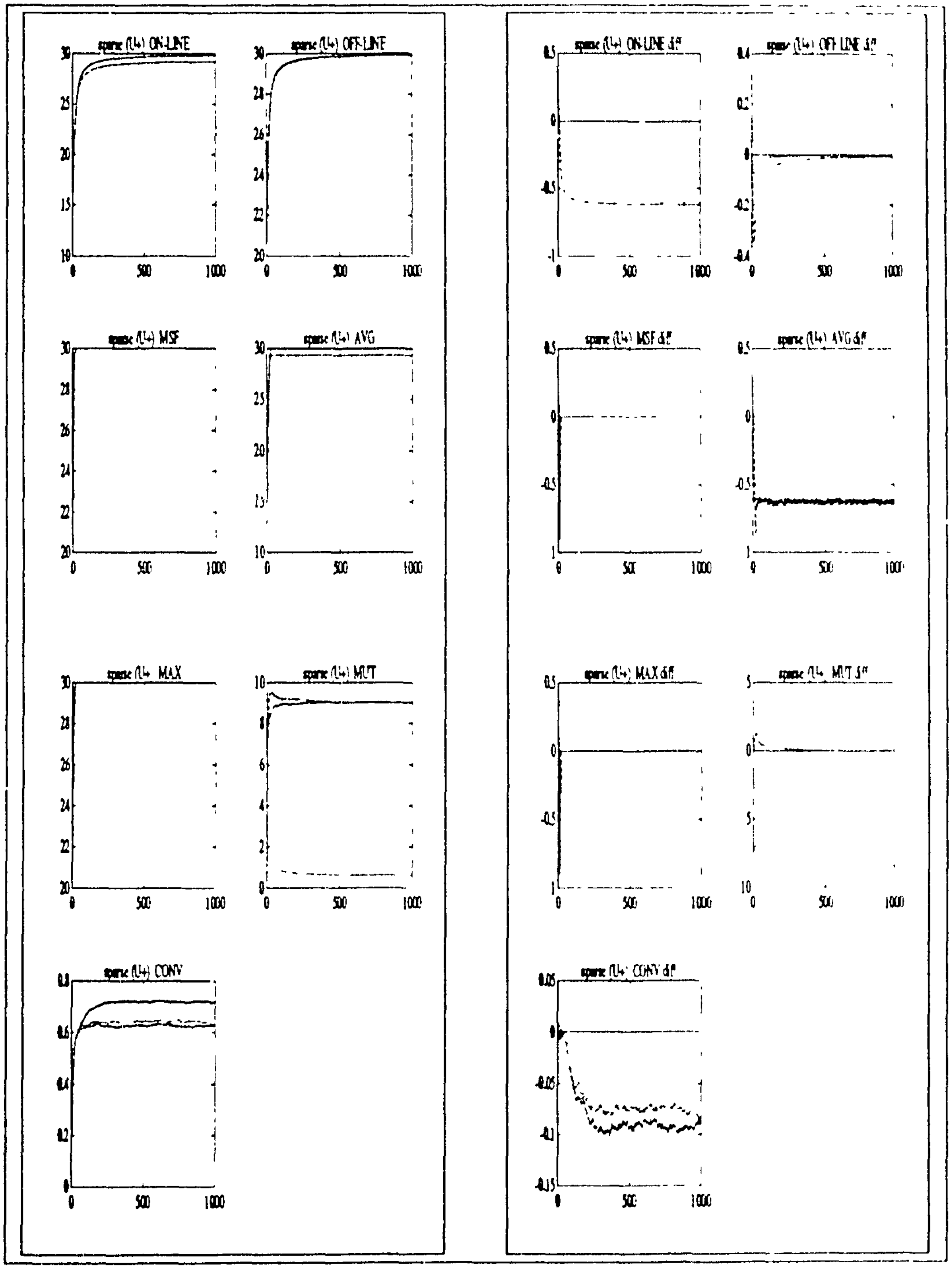


Figure 6.21: Results of Sparse One Max for modified uniform crossover

TRAVLLLING SALESMAN PROBLEM								
UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 -14.12	1 -11.37	2 -9.18	1 -11.33	2 -9.89	1 0.89	1 163.95
	M	1 -13.90	1 -11.89	1 -10.68	1 -11.60	1 -19.68	2 0.92	2 144.80
	D	1 -14.44	1 -11.81	1.5 -9.89	1 -11.99	1.5 -9.89	1 0.89	1 25.00
100	M+D	1 -12.56	2 -10.05	2 -8.35	1.5 -10.68	2 -8.35	1 0.87	3 316.40
	M	1 -12.37	1 -11.07	1 -9.9970	1 -10.30	1 -9.9970	2 0.90	2 288.15
	D	1 -12.96	1.5 -10.58	1.5 -9.03	1 -11.27	-1.5 -9.03	1 0.88	1 35.90
200	M+D	1 -11.41	2 -8.94	2 -7.53	1 -9.99	2 -7.53	1.5 0.86	3 606.80
	M	1 -11.24	1 -10.46	1 -9.64	1 -9.94	1 -9.64	1 0.85	2 583.10
	D	1 -11.8958	1 -9.57	2 -8.02	1 -10.2963	2 -8.02	2 0.87	1 42.10
500	M+D	1 -10.38	2 -7.93	2 -7.07	1 -9.49	2 -7.07	2 0.84	3 1477.55
	M	1 -10.02	1 -9.56	1 -8.4975	2 -8.69	1 -8.4975	1 0.82	2 1452.05
	D	1 -10.58	2 -8.22	2 -6.79	1.5 -9.18	1 -6.79	2 0.84	1 60.45
1000	M+D	1 -9.7980	2 -7.36	2 -6.57	1 -8.99	2 -6.57	2 0.83	2 2934.90
	M	2 -8.76	1 -8.44	1 -7.72	2 -7.92	1 -7.72	1 0.76	2 2773.75
	D	1 -9.75	2 -7.39	2 -6.46	1 -8.82	2 -6.46	2 0.83	1 78.35

Table 6.19: Results of using desegregation on uniform crossover for function TSP.

TRAVELLING SALESMAN PROBLEM								
MODIFIED UNIFORM CROSSOVER								
gen		online	offline	MSF	avg	max	converge	changes
50	M+D	1 -14.36	1 -11.72	1 -10.08	1 -12.00	1 -10.08	1 0.89	3 171.75
	M	1 -13.89	1 -11.78	1 -10.20	2 -10.88	1 -10.20	2 0.93	2 144.55
	D	1 -14.19	1 -11.57	1 -9.83	1 -11.97	1 -9.83	1 0.88	1 29.75
100	M+D	1 -13.12	1 -10.72	1.5 -9.36	1 -11.64	1.5 -9.36	1 0.86	3 327.10
	M	2 -12.19	1 -10.97	1 -9.97	2 -10.33	1 -9.97	2 0.88	2 292.35
	D	1.5 -12.8962	1 -10.47	2 -8.87	1.5 -11.17	2 -8.87	1 0.87	1 39.70
200	M+D	1 -12.11	1.5 -9.73	1 -8.19	1 -10.67	2 -8.19	1 0.85	3 617.25
	M	2 -11.07	1 -10.32	1 -9.41	2 -9.64	1 -9.41	1 0.86	2 583.85
	D	1.5 -11.75	2 -9.36	1 -7.84	1.5 -10.23	2 7.81	1 0.85	1 46.30
500	M+D	1 -10.75	2 -8.34	2 -7.09	1 -9.49	2 -7.09	2 0.84	3 1491.05
	M	1 -9.98	1 -9.54	1 -8.86	1 -9.13	1 -8.86	1 0.82	2 1459.45
	D	1 -10.38	2 -7.97	2 -6.68	1 -9.09	2 -6.68	2 0.84	1 58.25
1000	M+D	1 -10.04	2 -7.69	2 -6.70	1 -9.14	2 -6.70	2 0.84	2 2935.20
	M	1 -9.72	1 -9.0956	1 -8.49	1 -8.69	1 -8.49	1 0.81	2 2909.30
	D	1 -9.68	2 -7.25	2 -6.46	1 -8.93	2 -6.46	2 0.84	1 78.25

Table 6.20: Results of using desegregation on modified uniform crossover for function TSP.

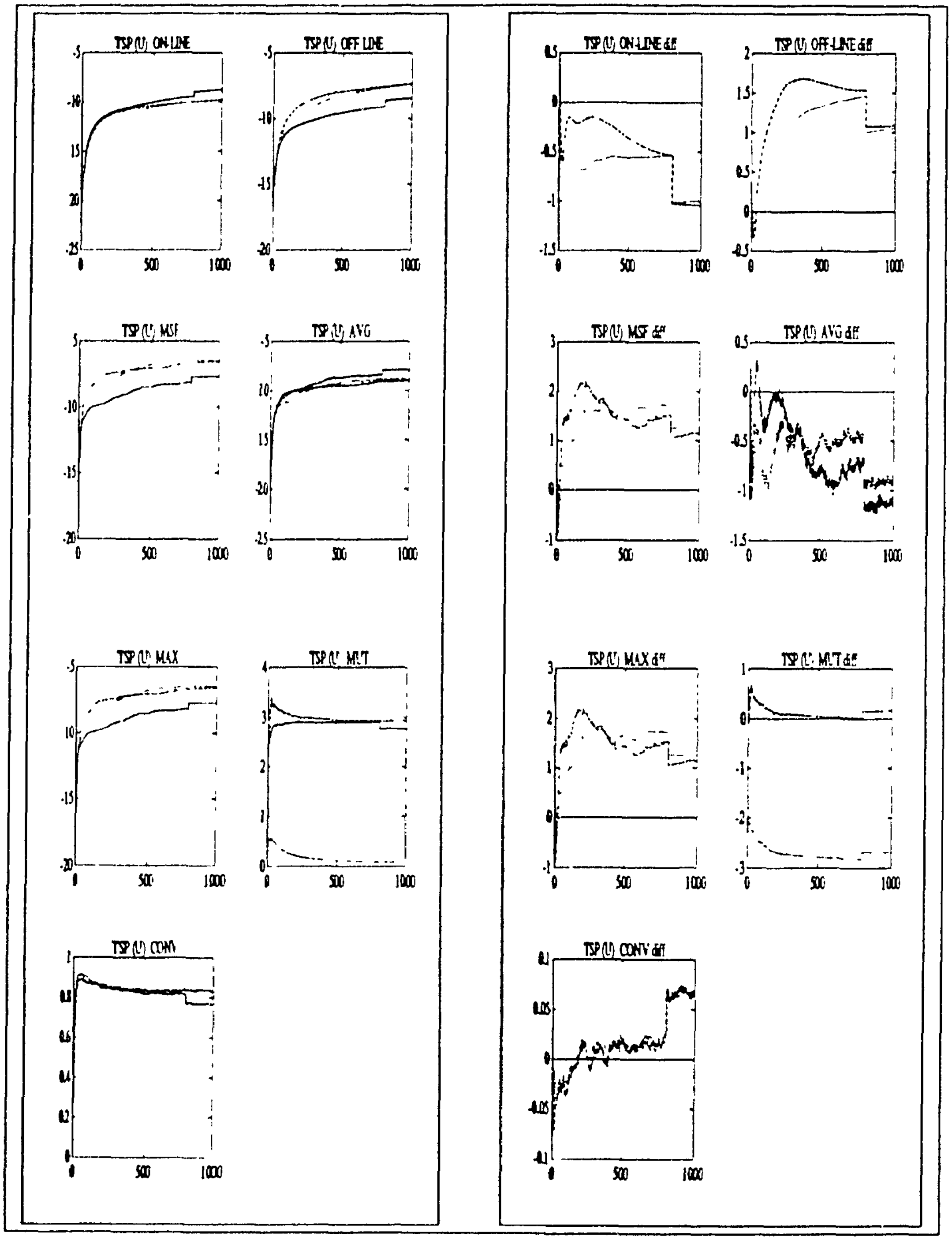


Figure 6.22: Results of Travelling Salesman Problem for uniform crossover.

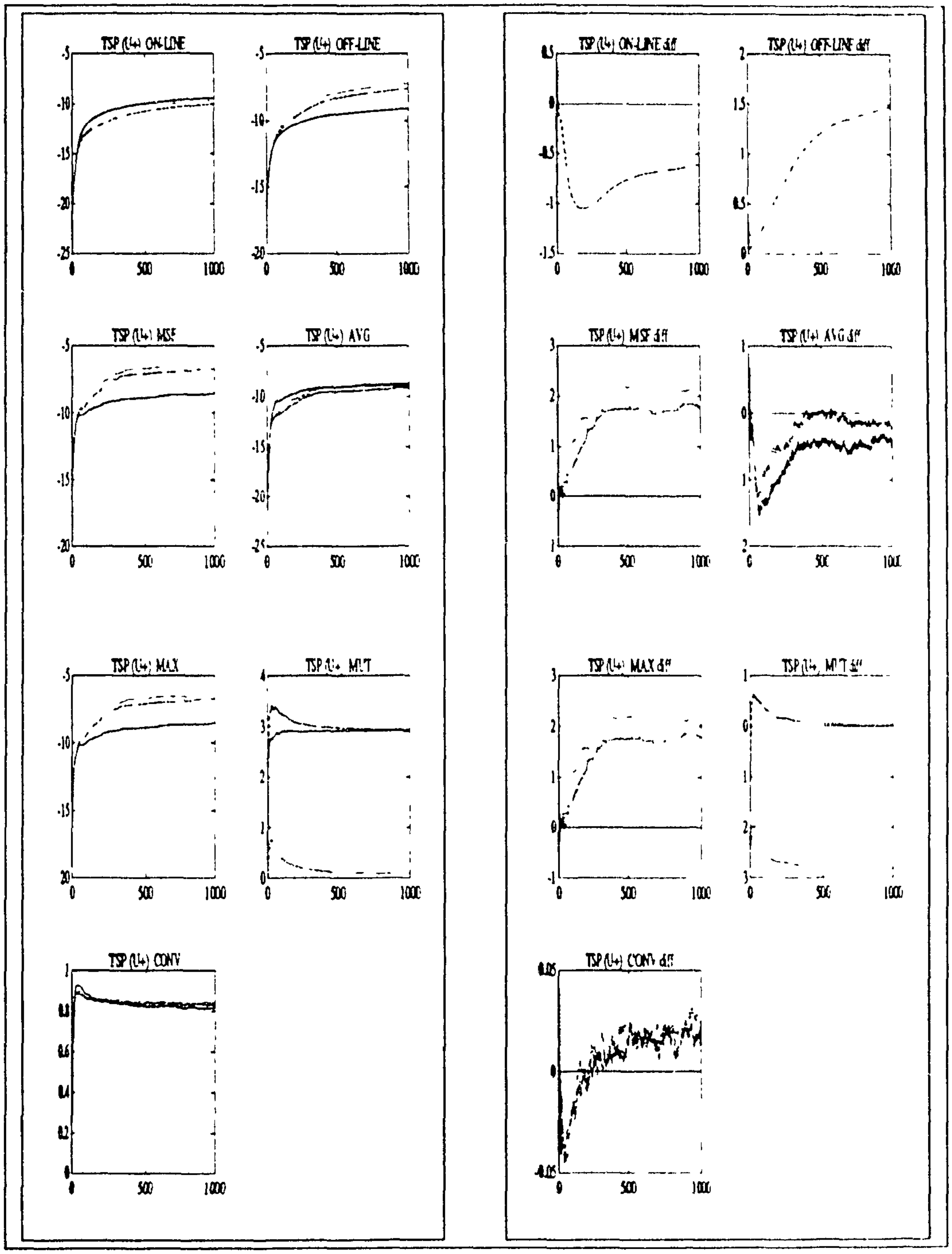


Figure 6.23: Results of Travelling Salesman Problem for modified uniform crossover

Chapter 7

Uniform versus Modified Uniform Crossover

This chapter will discuss the results of the empirical experiments described in chapter 5. The expected results as explained in theorem 1 are that for some problem sets uniform crossover will perform better, and for others modified uniform crossover will have better results.

The analysis is performed by doing a t-test on each function parameter set combination. The results of the t-test will also be analyzed using the same three search stages described in chapter 6.

Tables 7.1 to 7.10 list the operator that is higher, and the probability that the value for the two operators are identical. A dash for the operator indicates that the two operators were equal. A dash for the probability indicates that at least one of the operators had identical values for all trials.

The graphs presented in this graph are similar to the graphs presented in chapter 6. The solid curves represent the uniform crossover operator, while the dotted curves represent the modified uniform crossover operator. The "diff" graphs use uniform crossover as the base.

As in section 6.1, the charts labelled ON-LINE, OFF-LINE, MSF, AVG, and

MAX. represent the on-line. off-line maximum so far, average at generation, and maximum at generation scores described in section 5.3. The charts labelled CONV measure the population convergence using

$$\text{convergence} = \frac{1}{l} \sum_{i=1}^{\text{pop size}} \max(0_i, 1_i)$$

where 0_i and 1_i are the number of 0's and 1's in the i-th gene position respectively.

The charts "Mut & Deseg" have the cumulative total of the number of genes that were changed by either mutation or desegregation as their Y-axis.

From examining the tables it is clear that neither uniform crossover nor modified uniform crossover is a much better operator.

Among the results that are glaringly different is that modified uniform crossover performs much better for on-line values for One Max with parameter set M+D. while uniform crossover performs better for Sparse One Max. Since the two problems are very similar. other then the extra bits which have no value in Sparse One Max, the results were unexpected.

The reader should note that the results of the crossover operator could vary depending on which parameter set was used. This is clear in problem F1 at generation 50, for the M.S.F. rating. where uniform crossover is superior for the mutation and desegregation parameter set, yet modified uniform crossover is better for the desegregation parameter set. Other interesting points are generation 50 for problem F4 where the on-line and off-line values are stronger for uniform crossover for parameter set M+D, and weaker for parameter set D.

Note that for function F5, uniform crossover is better than modified uniform crossover for the desegregation parameter set.

Whether the online, offline,or MSF score is to be used for rating a crossover operator is also important. This is shown in generation 50 of problem F3. for the desegregation parameter set. where modified uniform crossover has better on-line and off-line scores. but uniform crossover has better MSF scores.

From the examination of the results of the experiments it seems clear that there is no clear winner between uniform crossover, and modified uniform crossover. As mentioned above the rating of which operator performs superiorly is dependant on both the function to be optimized, the parameter set, and which scores are to be used in the evaluation.

F1									
	M+D			M			D		
gen	on-line	off-line	M.S F	on-line	off-line	M S F	on-line	off-line	M S F
50	u .888	u .654	u .189	u .771	u 229	u 729	u+ 361	u+ 412	u+ 071
100	u .832	u .677	u+ 390	u 467	u+ .498	u 258	u+ 214	u+ 259	u+ 251
200	u+ .831	u .808	u+ .390	u .761	u+ 423	u+ 143	u+ 107	u+ 198	u+ 328
500	u+ .250	u 858	u+ .390	u+ 704	u+ 233	u+ 143	u+ 020	u+ 145	u+ 328
1000	u+ 075	u+ 614	u+ .390	u+ .260	u+ 150	u+ 143	u+ .006	u+ 170	u+ 328

Table 7.1: Superior Crossover Operator (F1)

F2									
	M+D			M			D		
gen	on-line	off-line	M S F	on-line	off-line	M.S.F	on-line	off-line	M.S.F
50	u+ 684	u+ .638	u 198	u+ .217	u .983	u .407	u .831	u .967	u+ .309
100	u 927	u+ .919	u 219	u+ 895	u 662	u .136	u .711	u+ .981	u+ .793
200	u 304	u .661	u .216	u 932	u .419	u .214	u .107	u+ .724	u+ .147
500	u 135	u .387	u .253	u .455	u .268	u .413	u+ .802	u+ .320	u+ .206
1000	u 162	u 310	u .253	u .477	u .327	u .465	u+ .405	u+ .240	u+ 206

Table 7.2: Superior Crossover Operator (F2)

F3									
	M+D			M			D		
gen	on-line	off-line	M.S.F	on-line	off-line	M.S.F	on-line	off-line	M.S.F
50	u+ 0.333	u+ 0.612	u+ 0.937	u+ 0.234	u+ 0.411	u 0.973	u+ 0.163	u+ 0.121	u 0.330
100	u+ 0.233	u+ 0.222	u+ 0.883	u+ 0.214	u+ 0.153	u 0.921	- -	u+ 0.163	- -
200	u 0.998	u+ 0.149	u+ 0.751	u 0.997	u+ 0.123	u 0.921	u 0.971	u+ 0.330	-
500	u 0.982	u+ 0.134	u+ 0.672	u 0.981	u+ 0.126	u 0.921	u 0.971	- -	- -
1000	u 0.977	u+ 0.320	u+ 0.569	u 0.976	u+ 0.321	u 0.921	u 0.971	u+ 0.330	- -

Table 7.3: Superior Crossover Operator (F3)

F4									
	M+D			M			D		
gen	on-line	off-line	M S.F	on-line	off-line	M S.F	on-line	off-line	M.S.F
50	u 711	u .755	u .234	u .410	u .398	u .217	u+ .611	u+ 649	u+ 805
100	u .640	u 689	u .746	u .138	u .278	u .216	u+ .542	u+ .560	u+ .615
200	u 417	u 453	u .501	u .080	u .278	u .765	u+ .949	u+ .997	u .188
500	u .343	u 381	u 370	u 049	u .200	u .544	u .312	u .294	u .011
1000	u 253	u 282	u ,063	u .039	u .142	u .021	u 272	u .246	u+ 698

Table 7.4: Superior Crossover Operator (F4)

F5									
	M+D			M			D		
gen	on-line	off-line	M S F	on-line	off-line	M S F	on-line	off-line	M S F
25	u+ .547			u .428			u 160		
50	u+ 370			u 428			u 199		
75	u+ .993			u+ 429			u+ .106		
100	u+ .625			u+ .428			u+ .372		
125	u+ 919			u 427			u 393		
150	u .662			u .428			u 486		
200	u+ .984			u 427			u 818		
500	u .966			u .572			u+ .147		
1000	u .708			u 560			u+ 227		

All trials started at generation 0 and kept the maximum value of 500.

Table 7.5: Superior Crossover Operator (F5)

CONTINOUS ONES									
	M+D			M			D		
gen	on-line	off-line	M S.F	on-line	off-line	M S F	on-line	off-line	M S F
50	u 0.000	u 0.556	u 0.097	u 0.000	u 0.281	u 0.090	u 0.000	u 0.010	u 0.010
100	u 0.000	u 0.300	u 0.024	u 0.000	u 0.061	u 0.023	u 0.000	u 0.010	u 0.010
200	u 0.000	u 0.161	u 0.011	u 0.000	u 0.022	u 0.013	u 0.000	u 0.010	u 0.010
500	u 0.000	u 0.102	u 0.010	u 0.000	u 0.013	u 0.011	u 0.000	u 0.010	u 0.010
1000	u 0.000	u 0.093	u 0.010	u 0.000	u 0.011	u .0.010	u 0.000	u 0.010	u 0.010

Table 7.6: Superior Crossover Operator (CONTINOUS ONES)

F5'									
	M+D			M			D		
gen	on-line	off-line	M S F	on-line	off-line	M S F	on-line	off-line	M S F
50	u+ 0.180	u 0.441	u 0.486	u+ 0.136	u 0.345	u 0.653	u+ 0.116	u 0.111	u 0.596
100	u+ 0.217	u 0.210	u 0.497	u+ 0.151	u 0.170	u 0.549	u+ 0.360	u 0.110	u 0.406
200	u+ 0.260	u 0.232	u 0.404	u+ 0.210	u 0.217	u 0.405	u+ 0.506	u 0.173	u 0.293
500	u+ 0.373	u 0.172	u 0.390	u+ 0.335	u 0.177	u 0.396	u+ 0.506	u 0.352	u 0.551
1000	u+ 0.439	u 0.450	u 0.451	u+ 0.406	u 0.467	u 0.453	u+ 0.506	u 0.967	u 0.551

Table 7.7: Superior Crossover Operator (F5')

One Max									
	M+D			M			D		
gen	on-line	off-line	M S F	on-line	off-line	M S F	on-line	off-line	M S F
25	u+ 216	u+ 206		u 460	u 579		u 657	u+ 990	
50	u+ 197	u+ 206		u 377	u 579		u 650	u+ 990	
75	u+ 161	u+ 206		u 411	u 579		u 590	u+ 990	
100	u+ 151	u+ 206		u 336	u 579		u 587	u+ 990	
125	u+ 146	u+ 206		u 319	u 579		u 566	u+ 989	
150	u+ 130	u+ 206		u .341	u 579		u 599	u+ 990	
200	u+ 127	u+ 206		u 355	u .578		u 711	u+ .989	
500	u+ 244	u+ 205		u 566	u 579		u 441	u+ 988	
1000	u+ 201	u+ 207		u 377	u 579		u .541	u+ 990	

Table 7.8: Superior Crossover Operator (One Max)

Sparse One Max									
	M+D			M			D		
gen	on-line	off-line	M S F	on-line	off-line	M S F	on line	off line	M S F
25	u 170	u 195		u 737	u 893		u 812	u 920	
50	u 168	u 195		u 733	u 893		u 824	u 920	
75	u 175	u 195		u 709	u 893		u 816	u 820	
100	u .175	u 195		u 708	u 893		u 829	u 920	
125	u 175	u 195		u 698	u 893		u 833	u 920	
150	u .177	u .196		u 701	u 892		u 827	u 921	
200	u 163	u 195		u 664	u 893		u 794	u 920	
500	u 205	u 195		u 600	u .893		u 775	u 920	
1000	u 223	u 195		u 609	u .893		u 698	u 920	

Table 7.9: Superior Crossover Operator (Sparse)

	TSP								
	M+D			M			D		
gen	on-line	off-line	M S F	on-line	off-line	M S F	on-line	off-line	M.S F
25	u+ 791	u+ 803	u+ 149	u 177	u 601	u .971	u+ .274	u+ 302	u+ 882
50	u 296	u 148	u+ 216	u+ 855	u+ 611	u+ 818	u+ 423	u+ 420	u+ .981
75	u 077	u 045	u+ 232	u .383	u .535	u 962	u 685	u .641	u+ .970
100	u 028	u 019	u+ 251	u+ .428	u+ 627	u+ 971	u+ .835	u+ 725	u+ 940
125	u 011	u 011	u+ 263	u+ .468	u+ .635	u+ .932	u+ 813	u+ 678	u+ 862
150	u 015	u 012	u 619	u+ 476	u+ 613	u+ .935	u 704	u .567	u .799
200	u 012	u 010	u .721	u+ 177	u+ 568	u+ .933	u+ 621	u+ 490	u+ .924
500	u 182	u 164	u 989	u+ 940	u+ 989	u 841	u+ .394	u+ 283	u+ 944
1000	u 279	u 250	u 928	u 478	u 473	u 853	u+ 656	u+ 398	u+ .998

Table 7.10: Superior Crossover Operator (TSP)

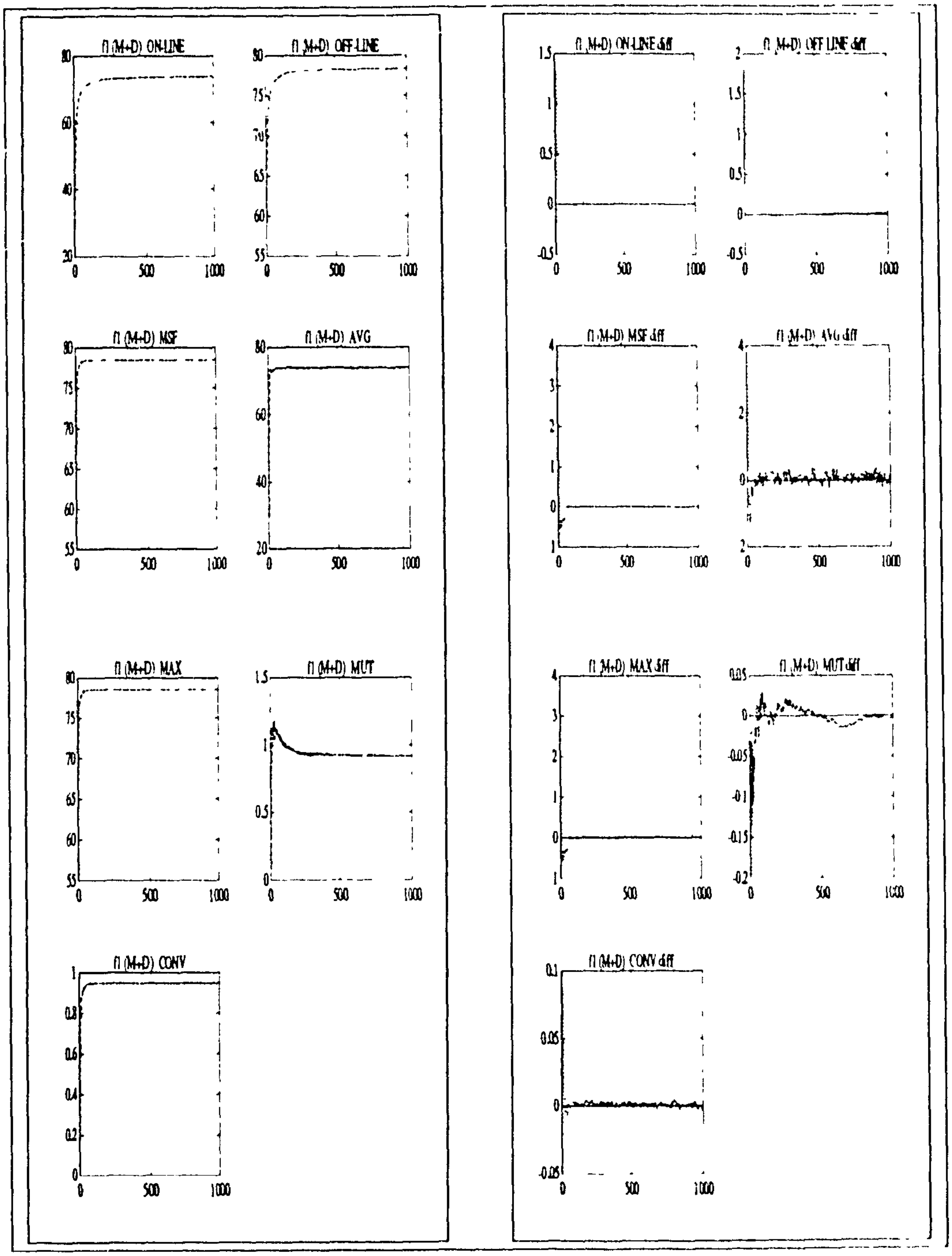


Figure 7.1: Results of F1 for mutation and desegregation.

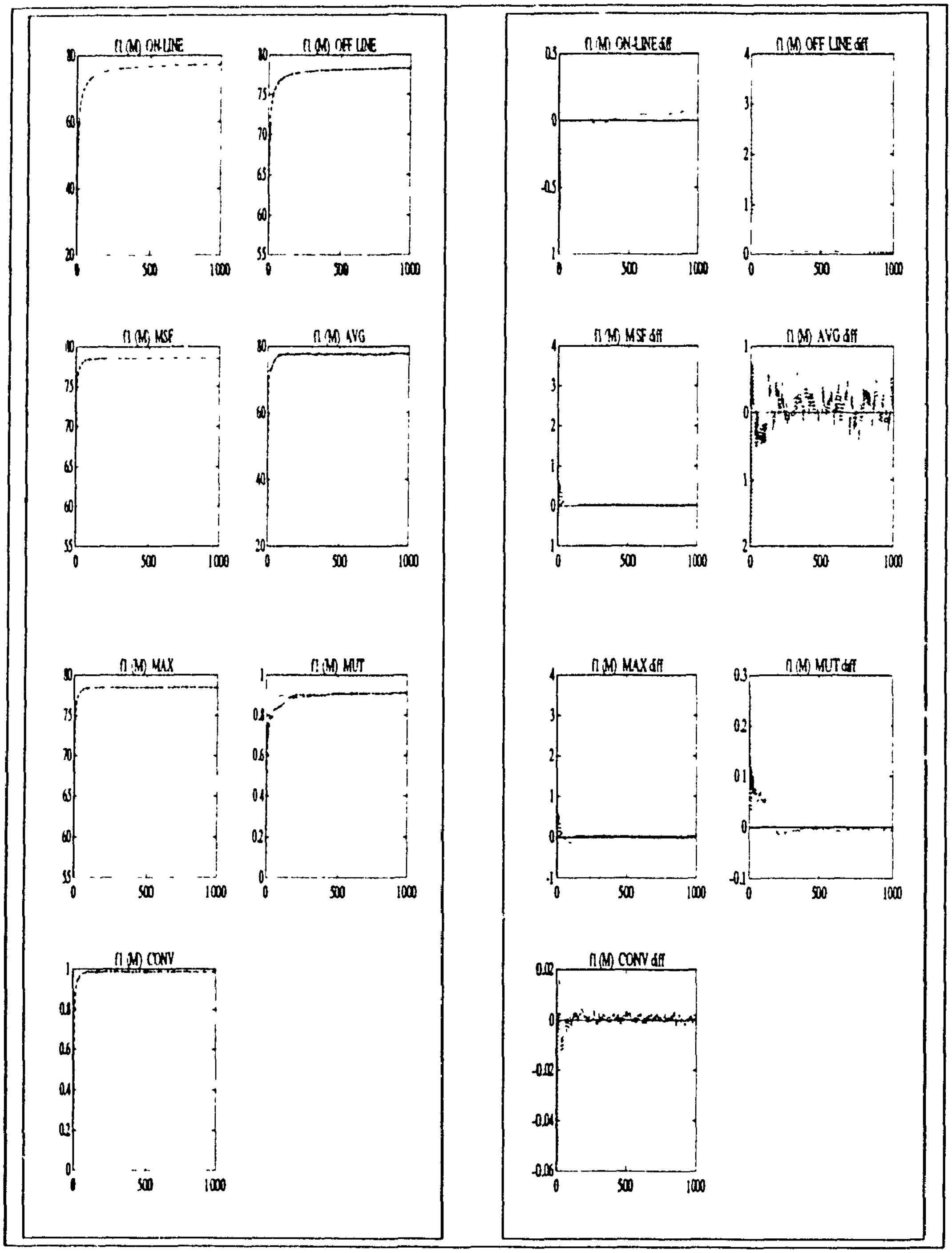


Figure 7.2: Results of F1 for mutation.

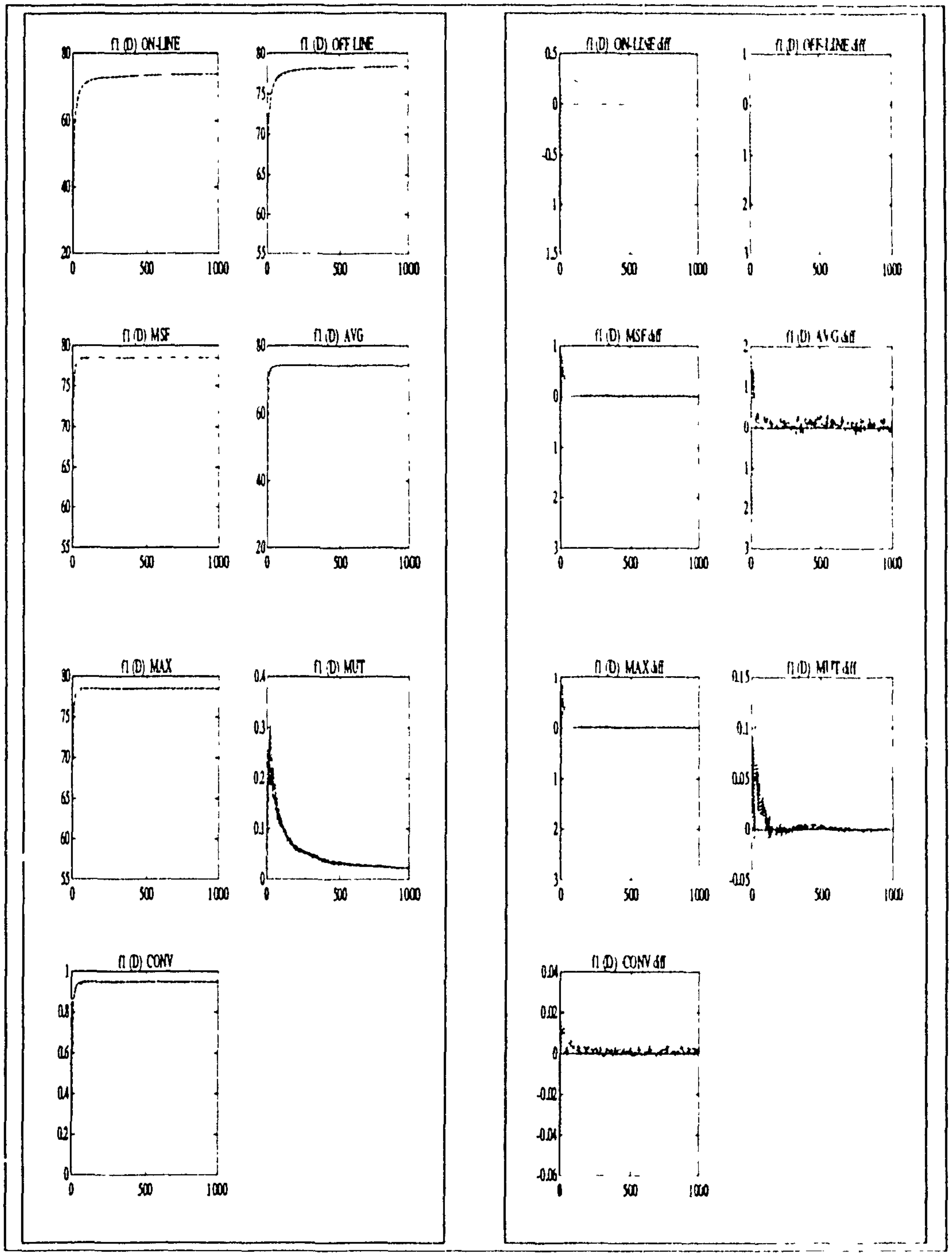


Figure 7.3: Results of F1 for desegregation.

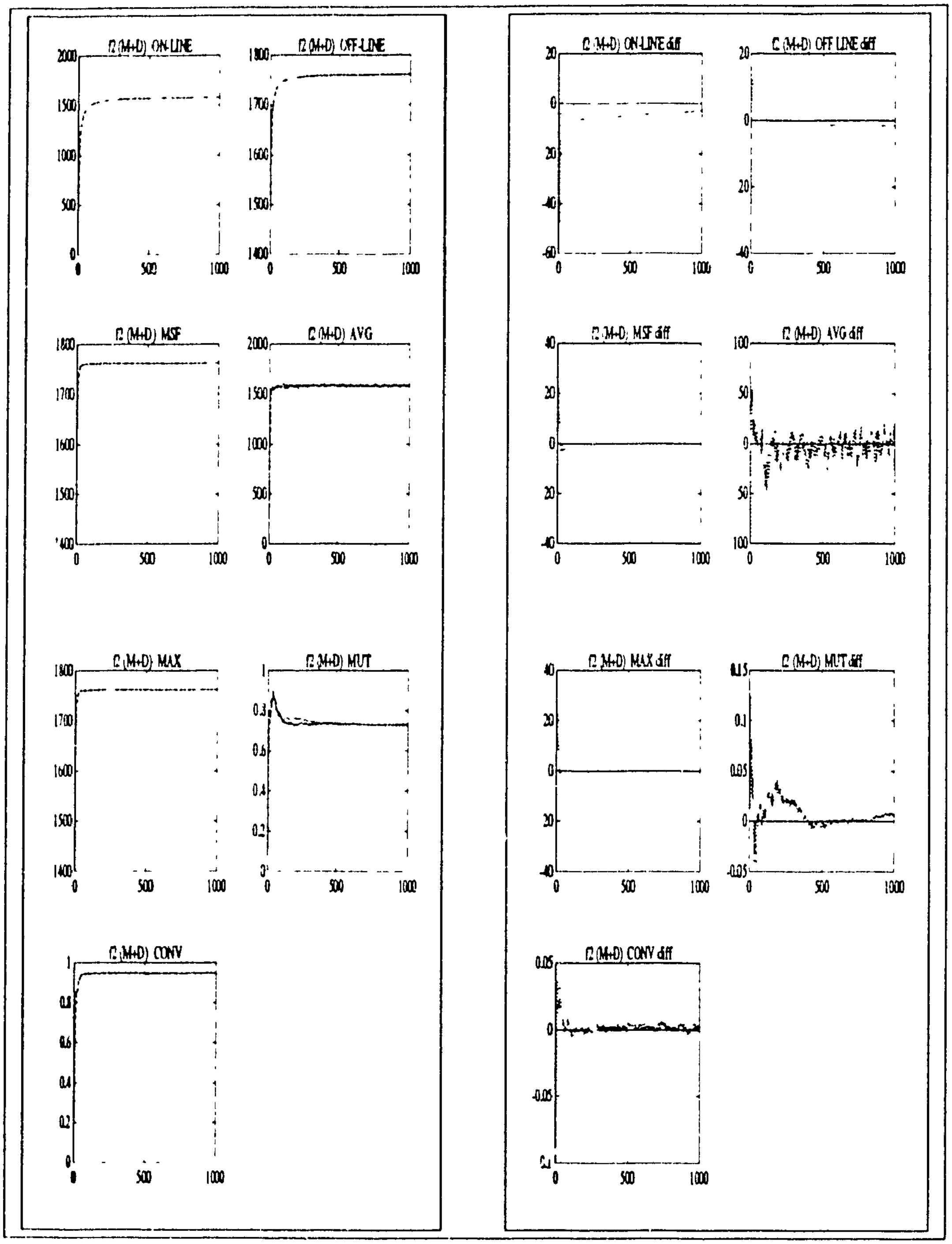


Figure 7.1: Results of F2 for mutation and desegregation.

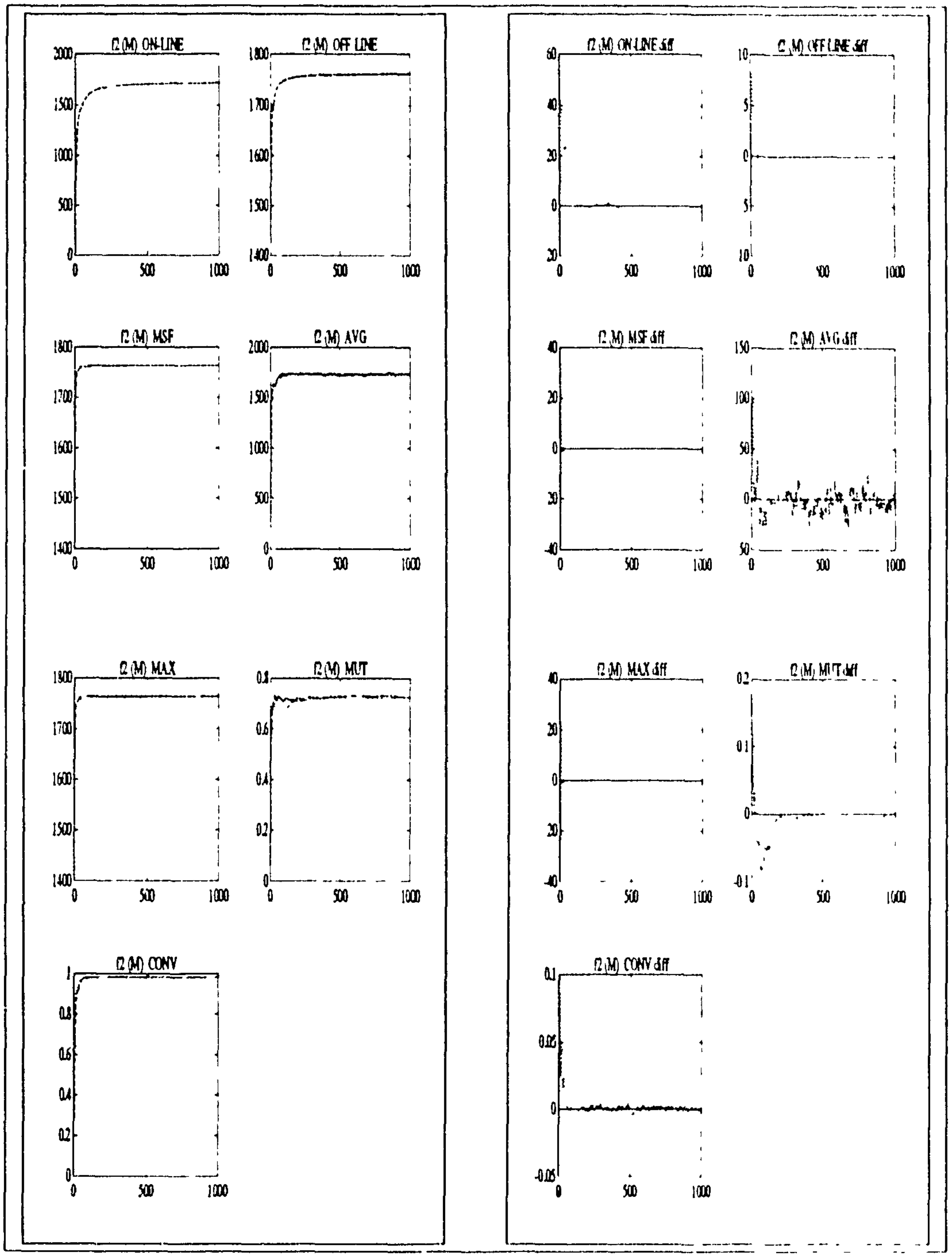


Figure 7.5: Results of F2 for mutation.

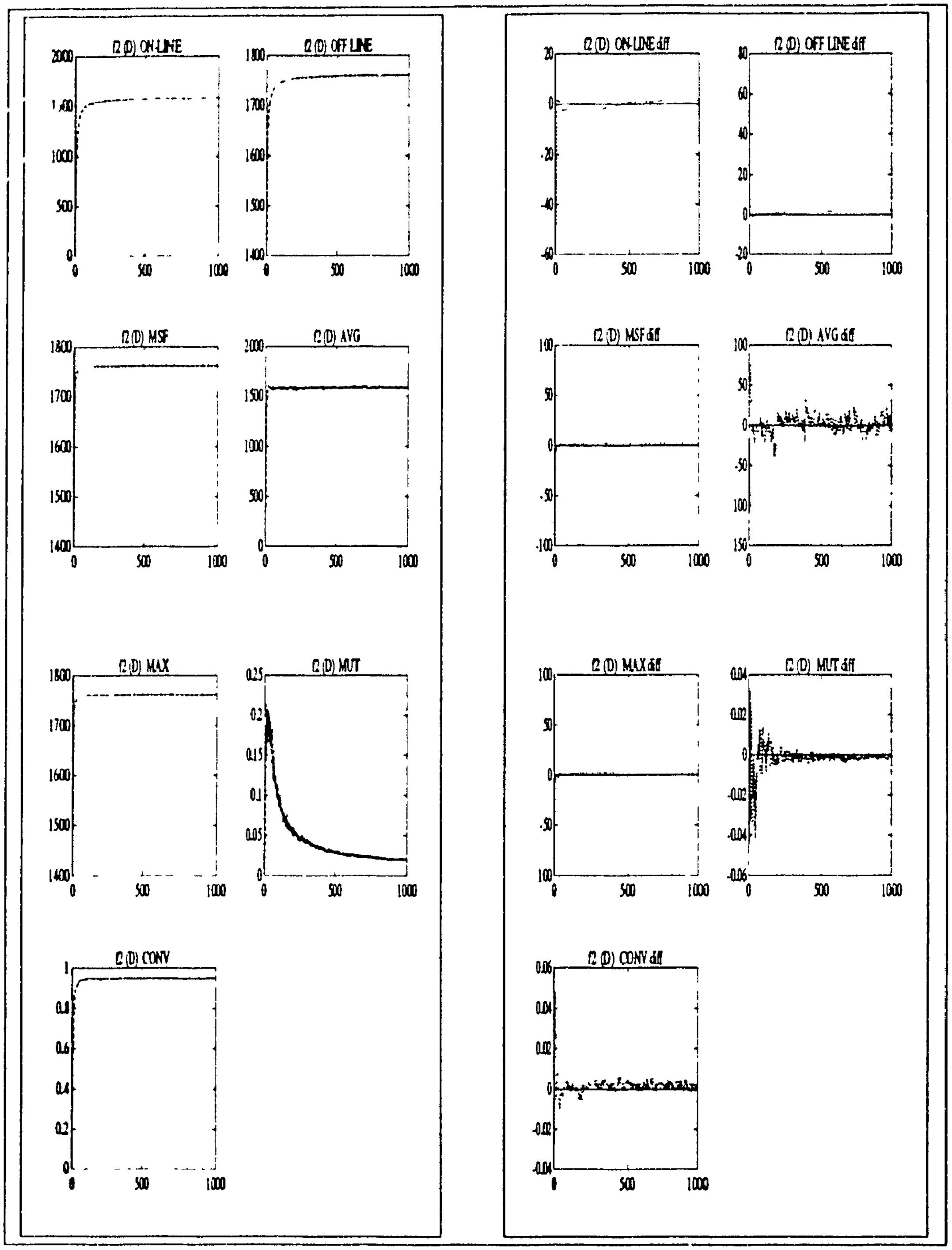


Figure 7.6: Results of F2 for desegregation.

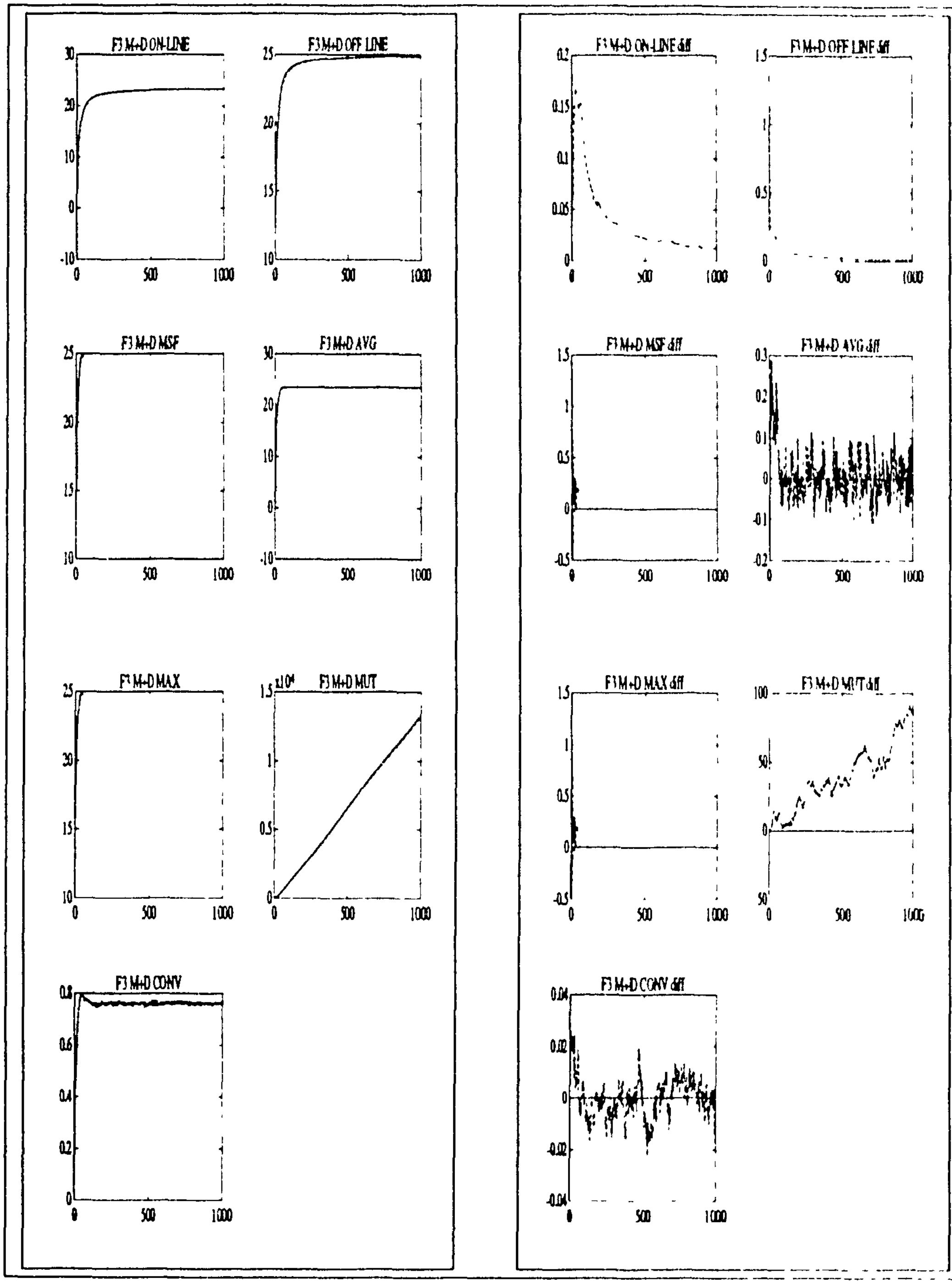


Figure 7.7: Results of F3 for mutation and desegregation.

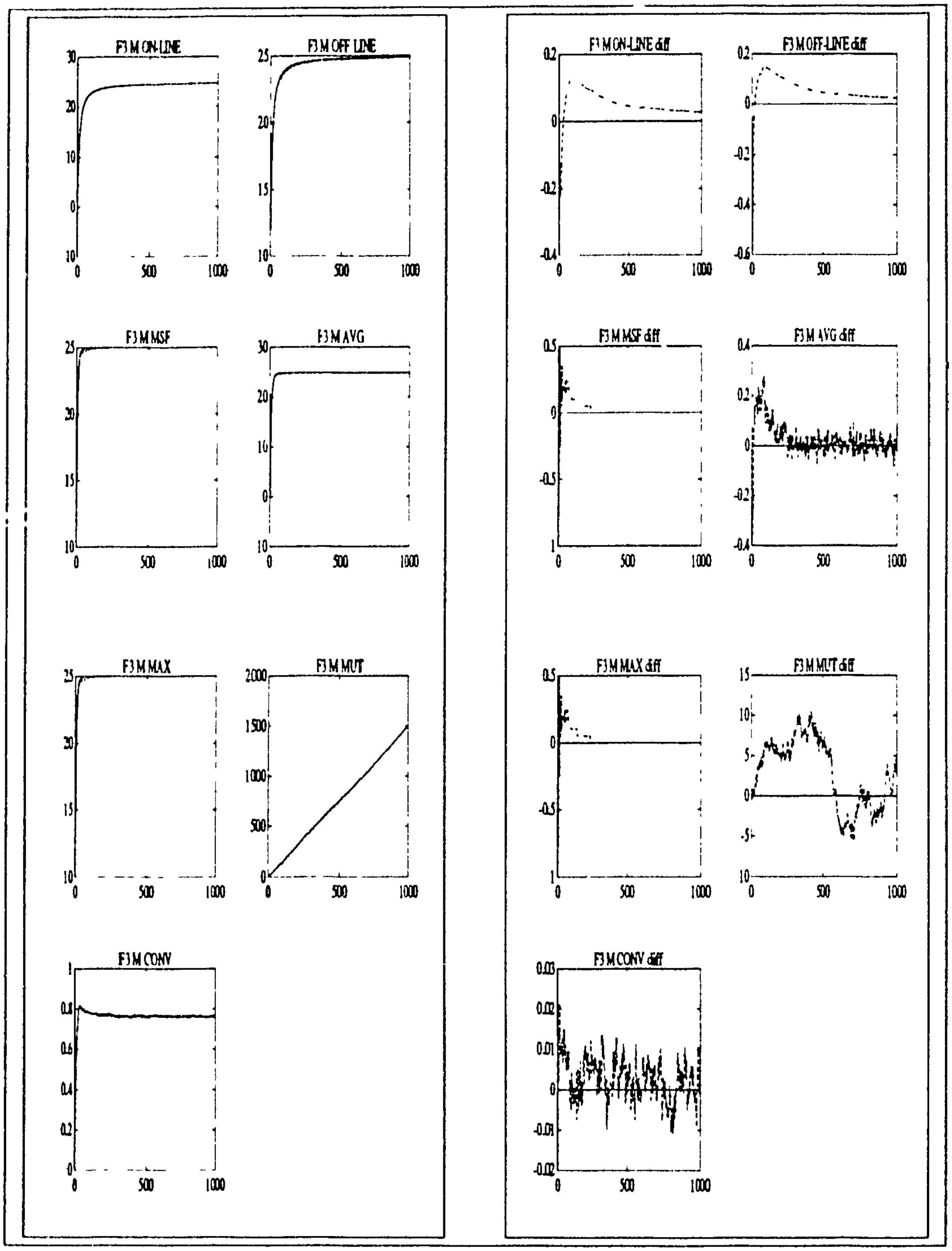


Figure 7.8: Results of F3 for mutation.

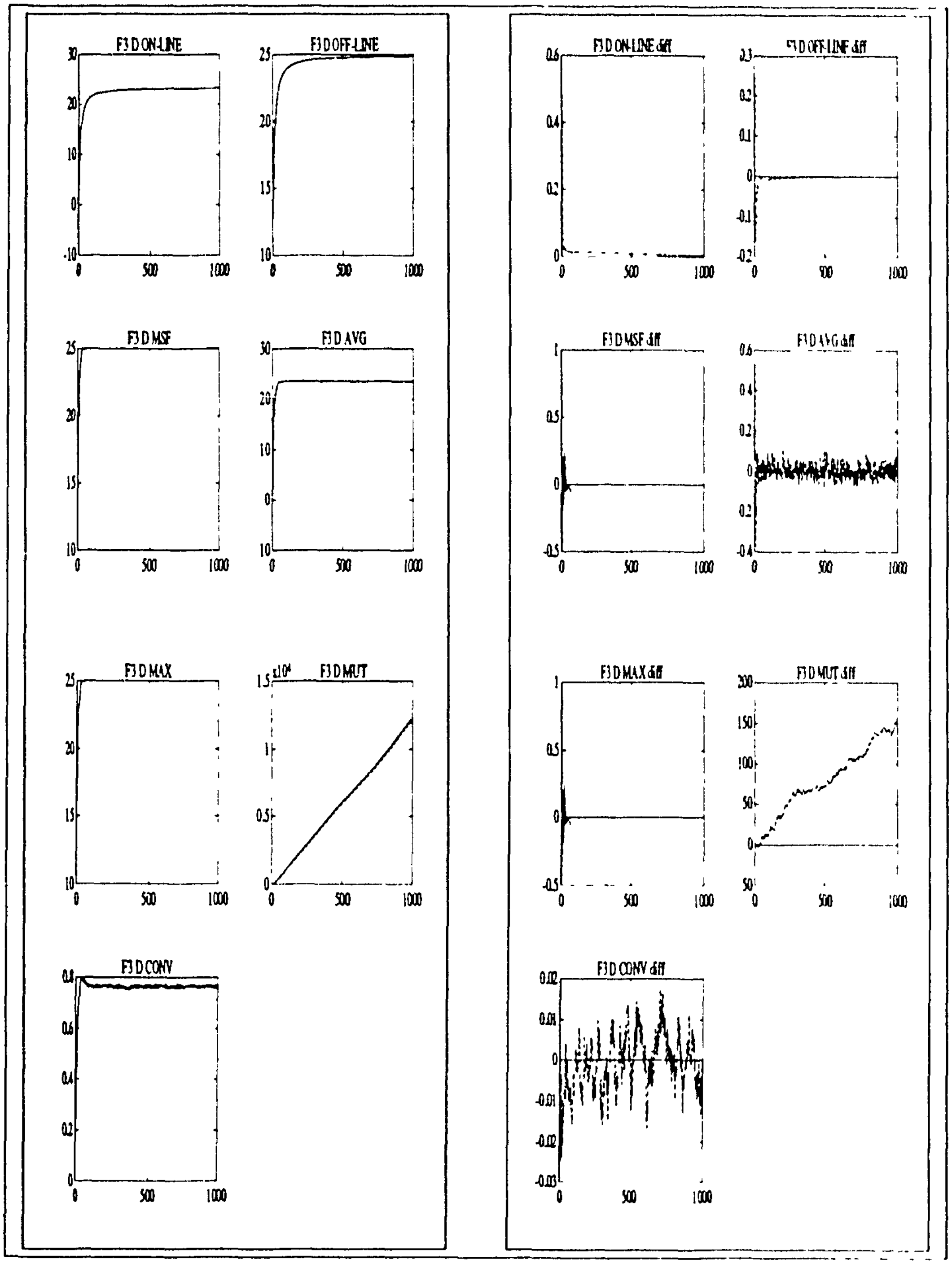


Figure 7.9: Results of F3 for desegregation.

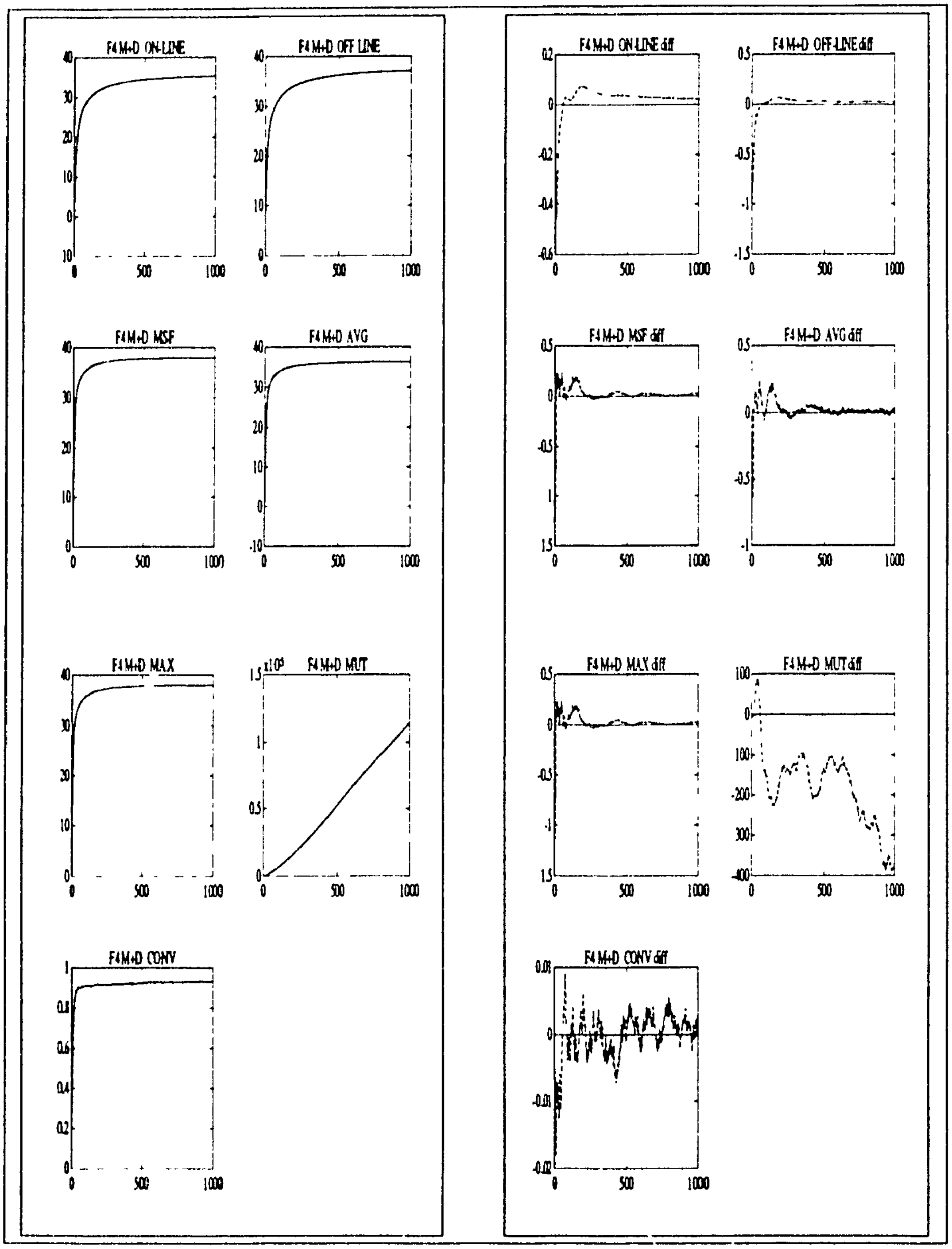


Figure 7.10: Results of F4 for mutation and desegregation.

xover/ps/f4mutsco.ps

xover/ps/f4mutdif.ps

Figure 7.11: Results of F4 for mutation.

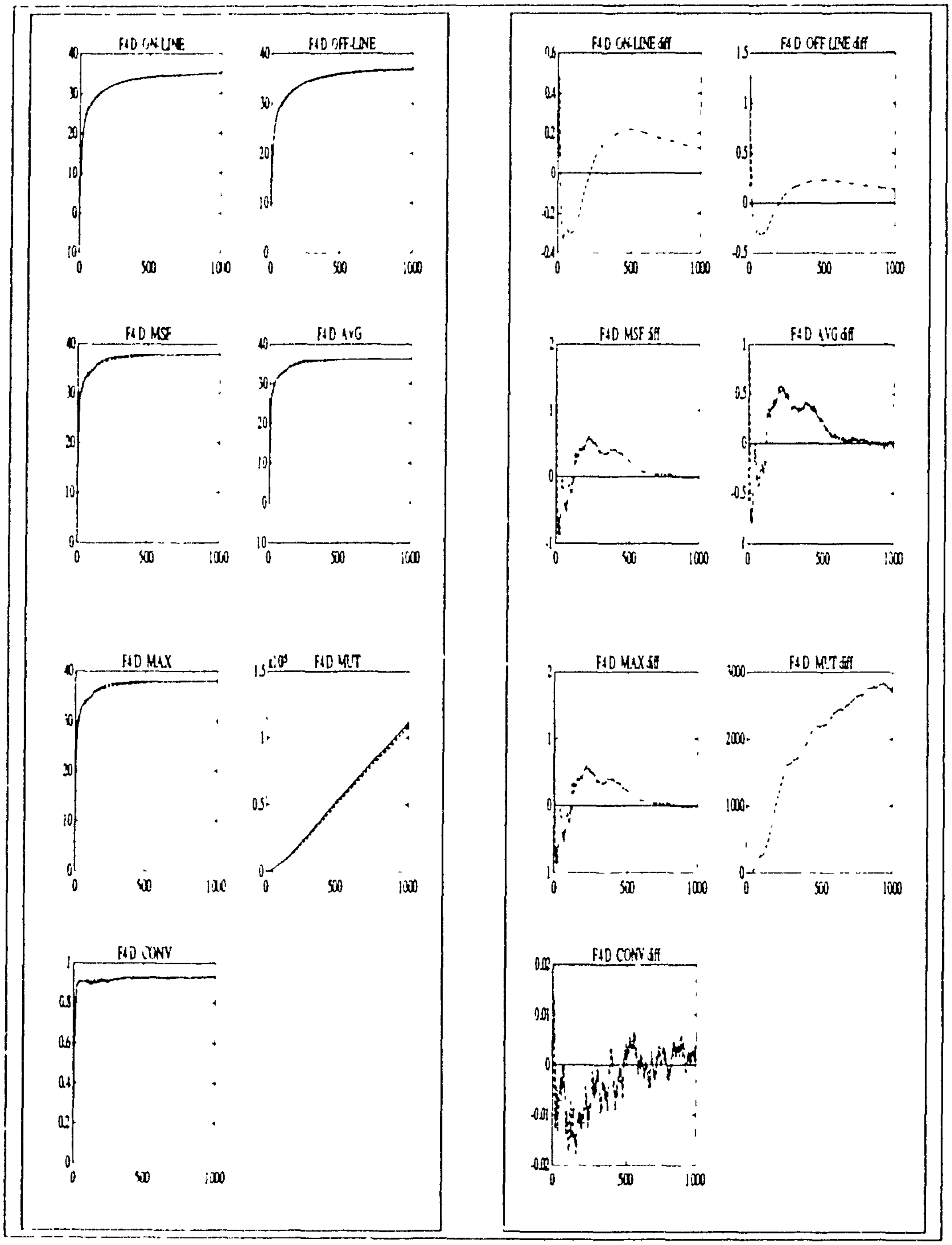


Figure 7.12: Results of F4 for desegregation.

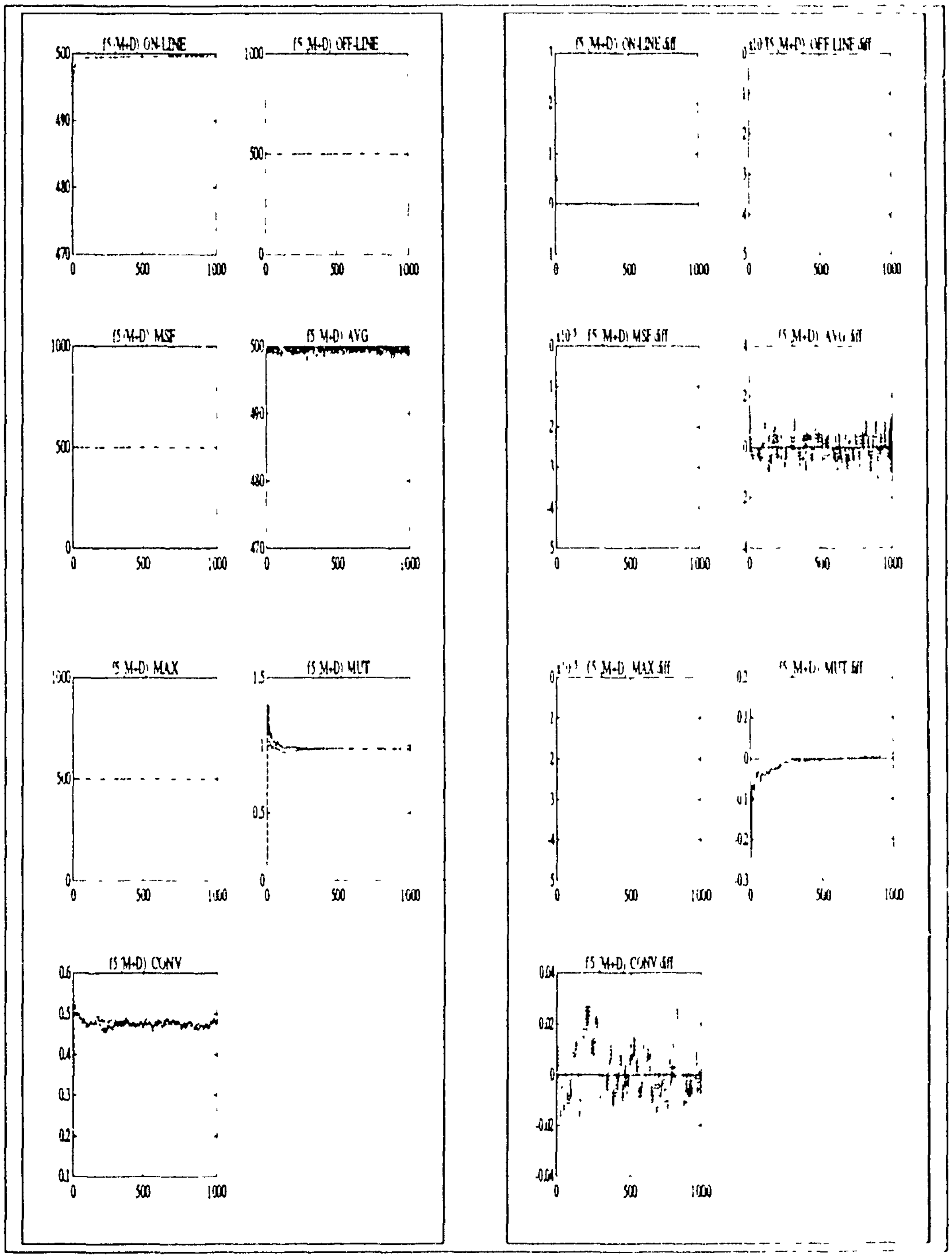


Figure 7.13: Results of F5 for mutation and desegregation

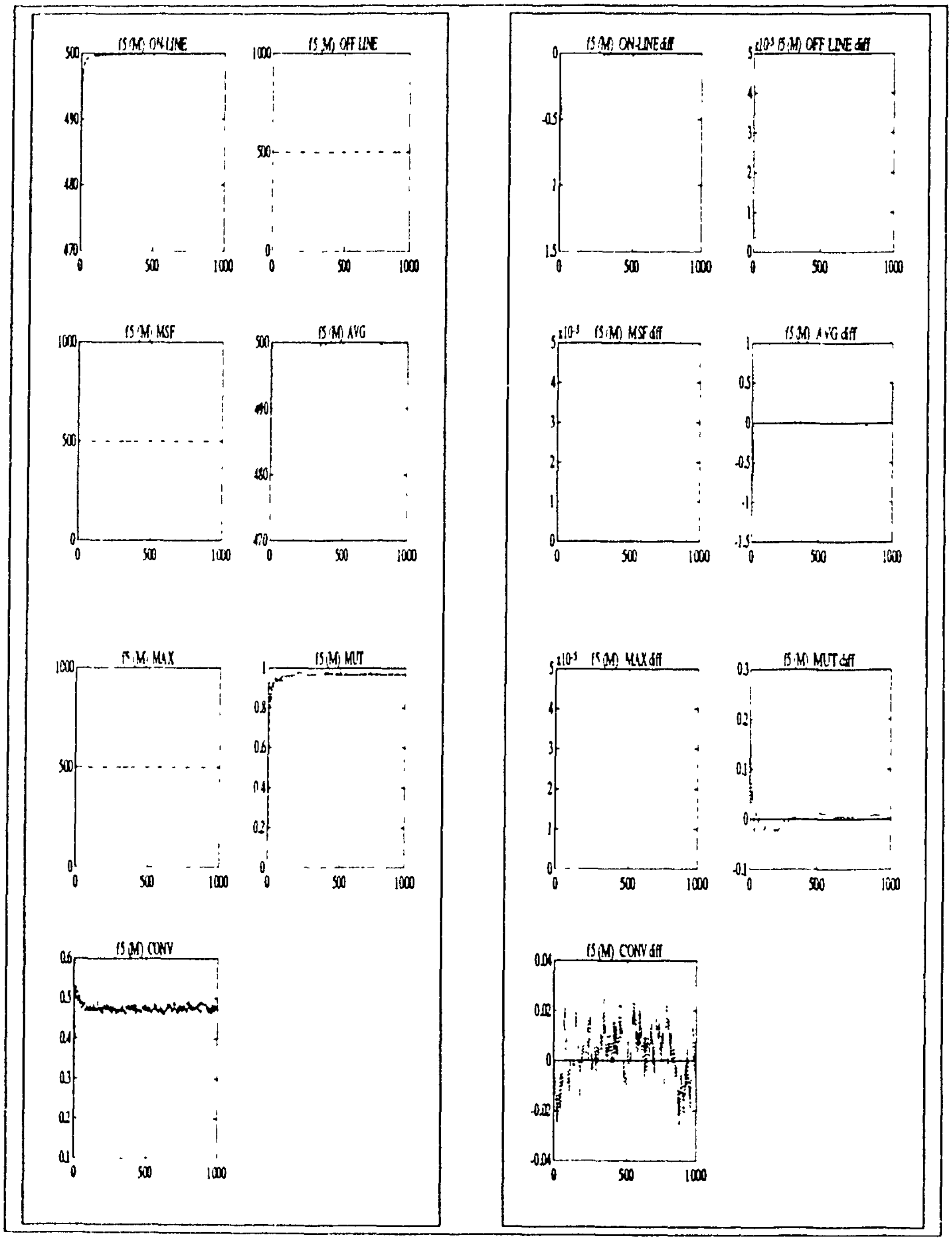


Figure 7.14: Results of F5 for mutation.

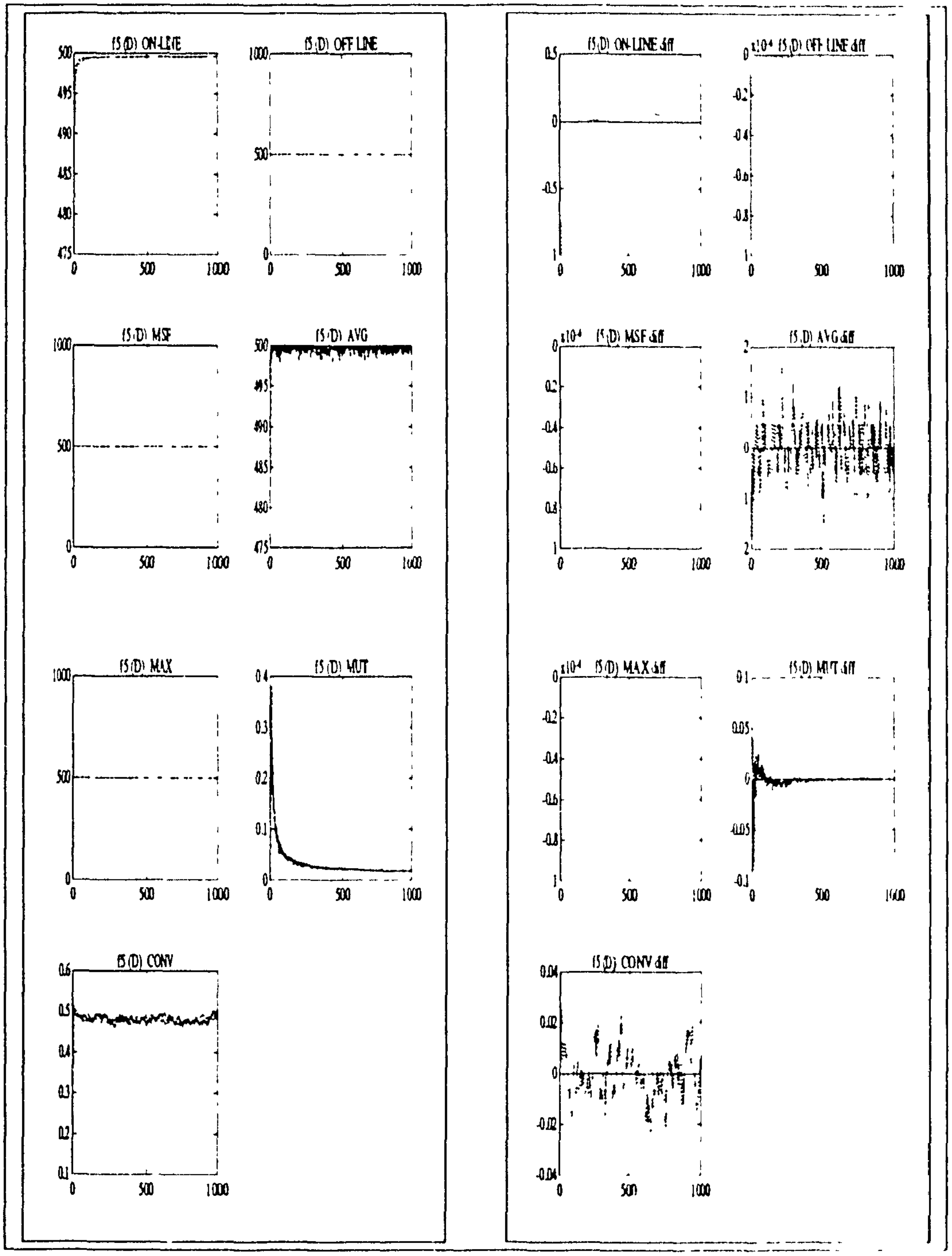


Figure 7.15: Results of F5 for desegregation.

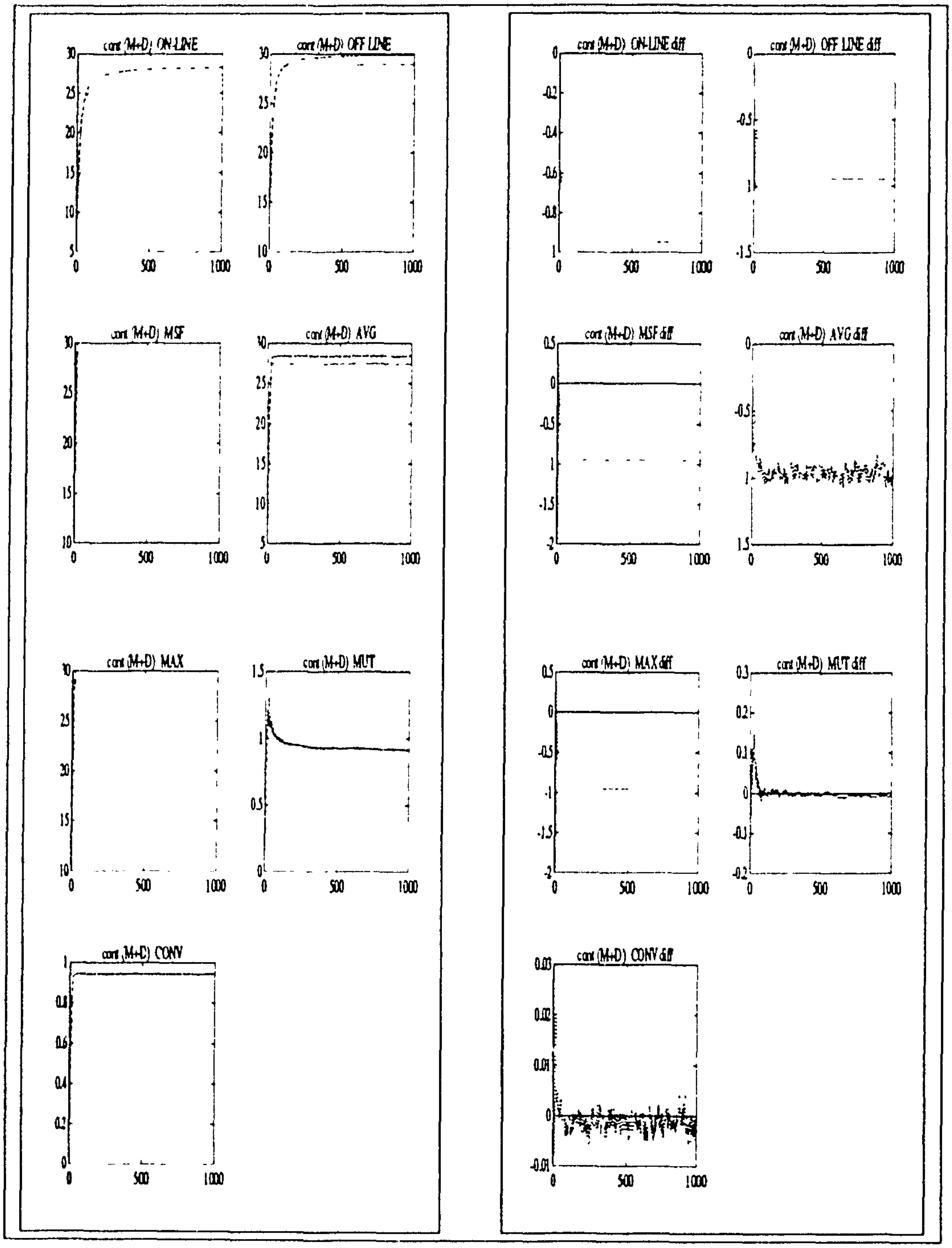


Figure 7.16: Results of Continous One Max for mutation and desegregation.

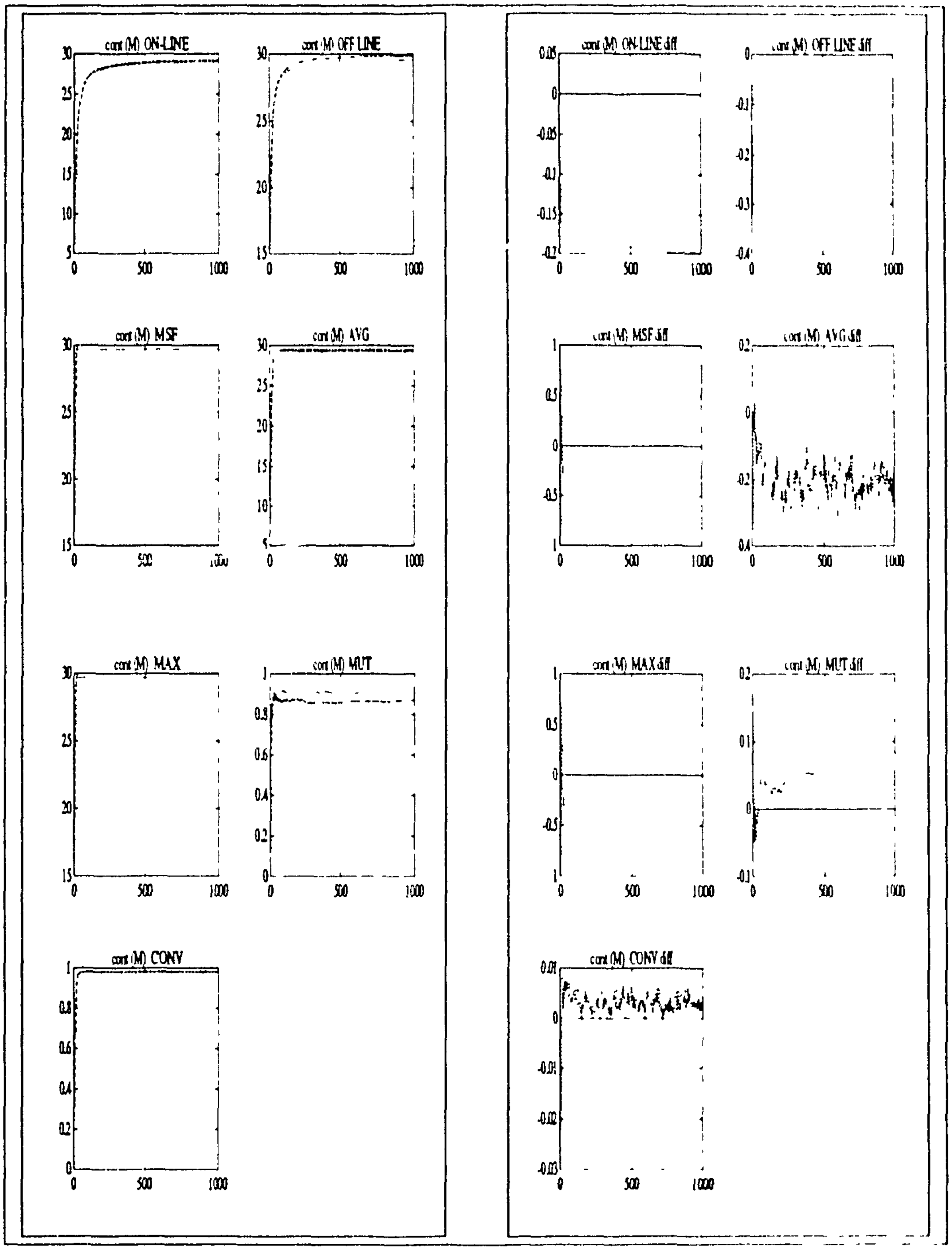


Figure 7.17: Results of Continous One Max for mutation

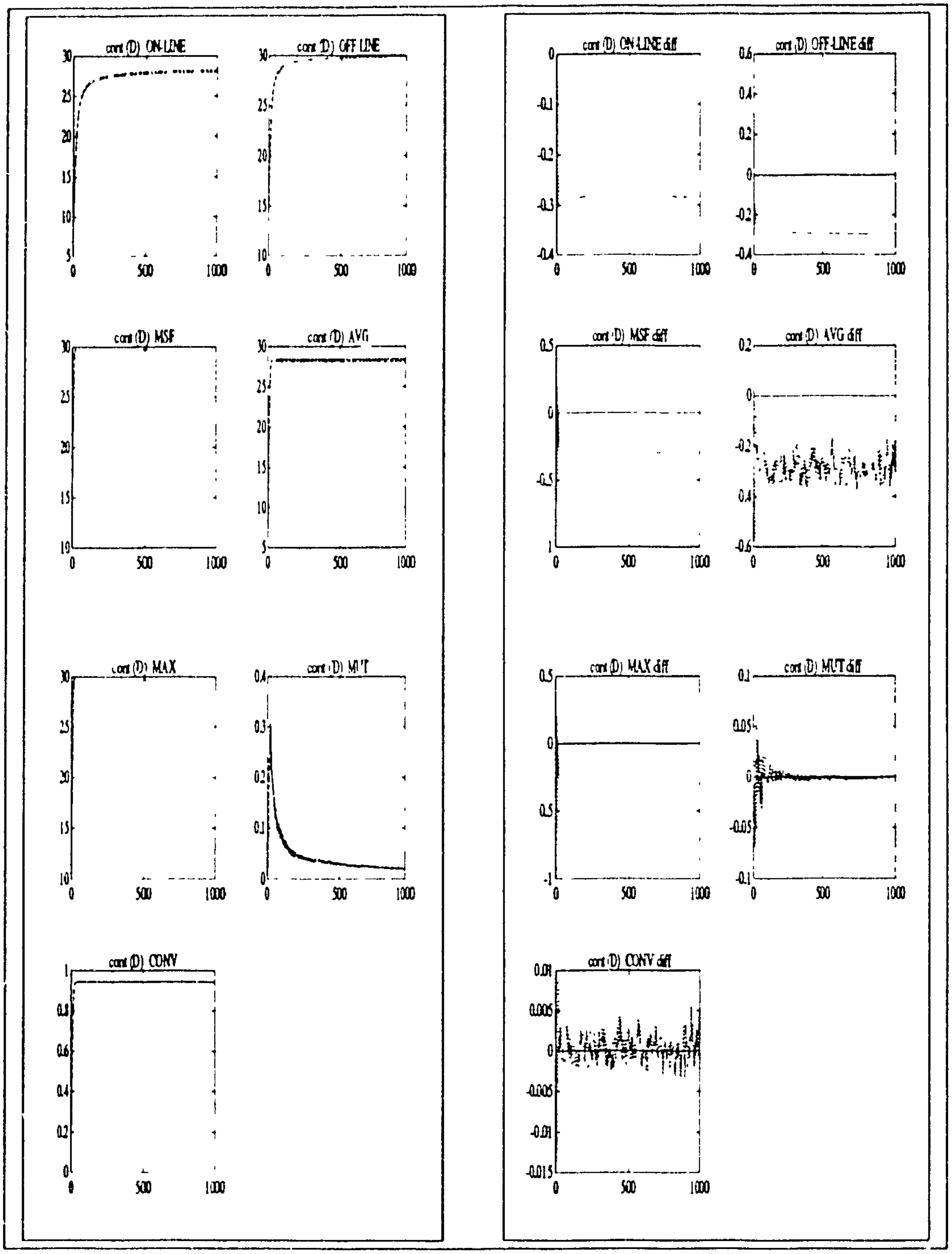


Figure 7.18: Results of Continouse One Max for desegregation.

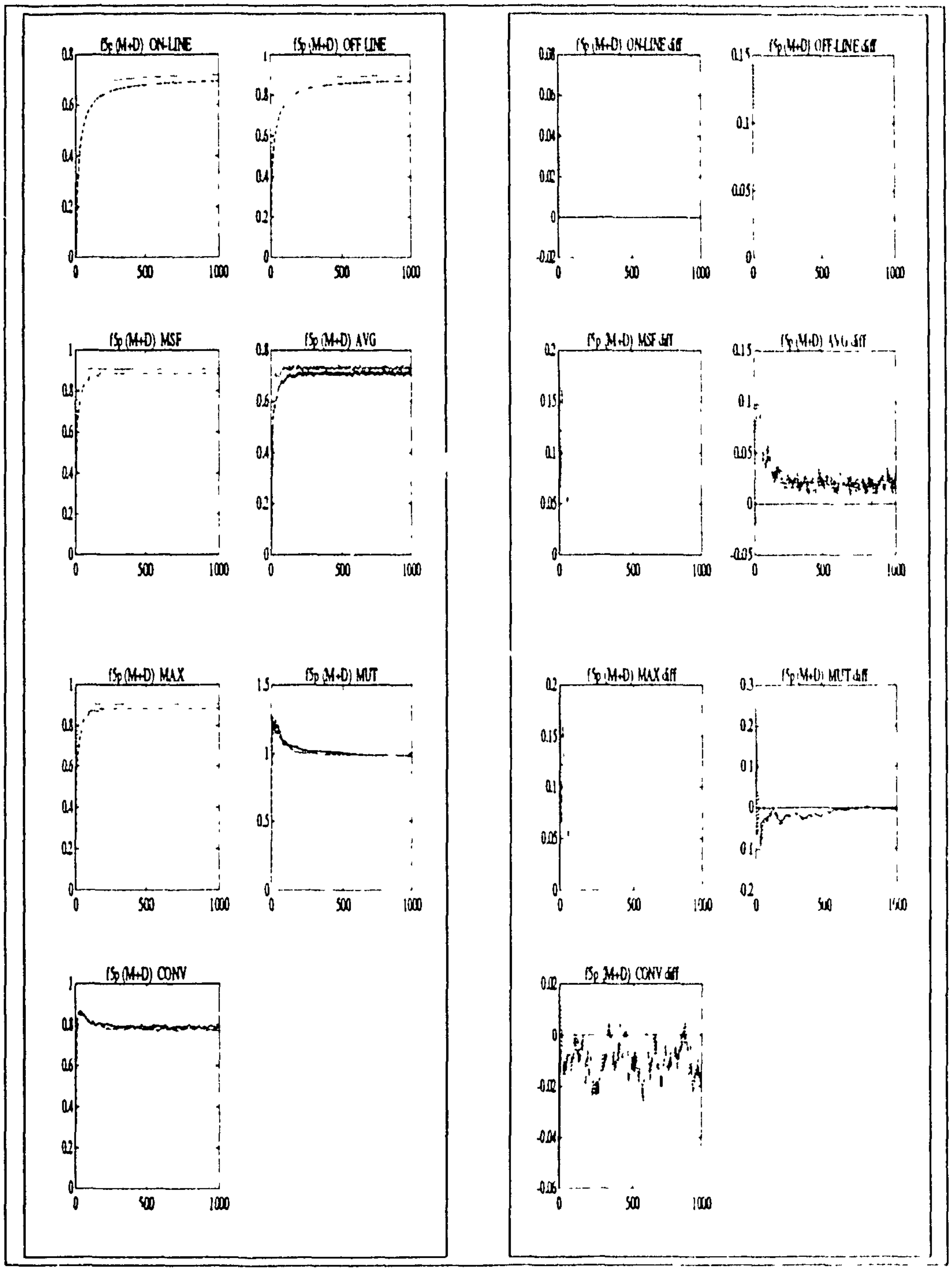


Figure 7.19: Results of F5' for mutation and desegregation

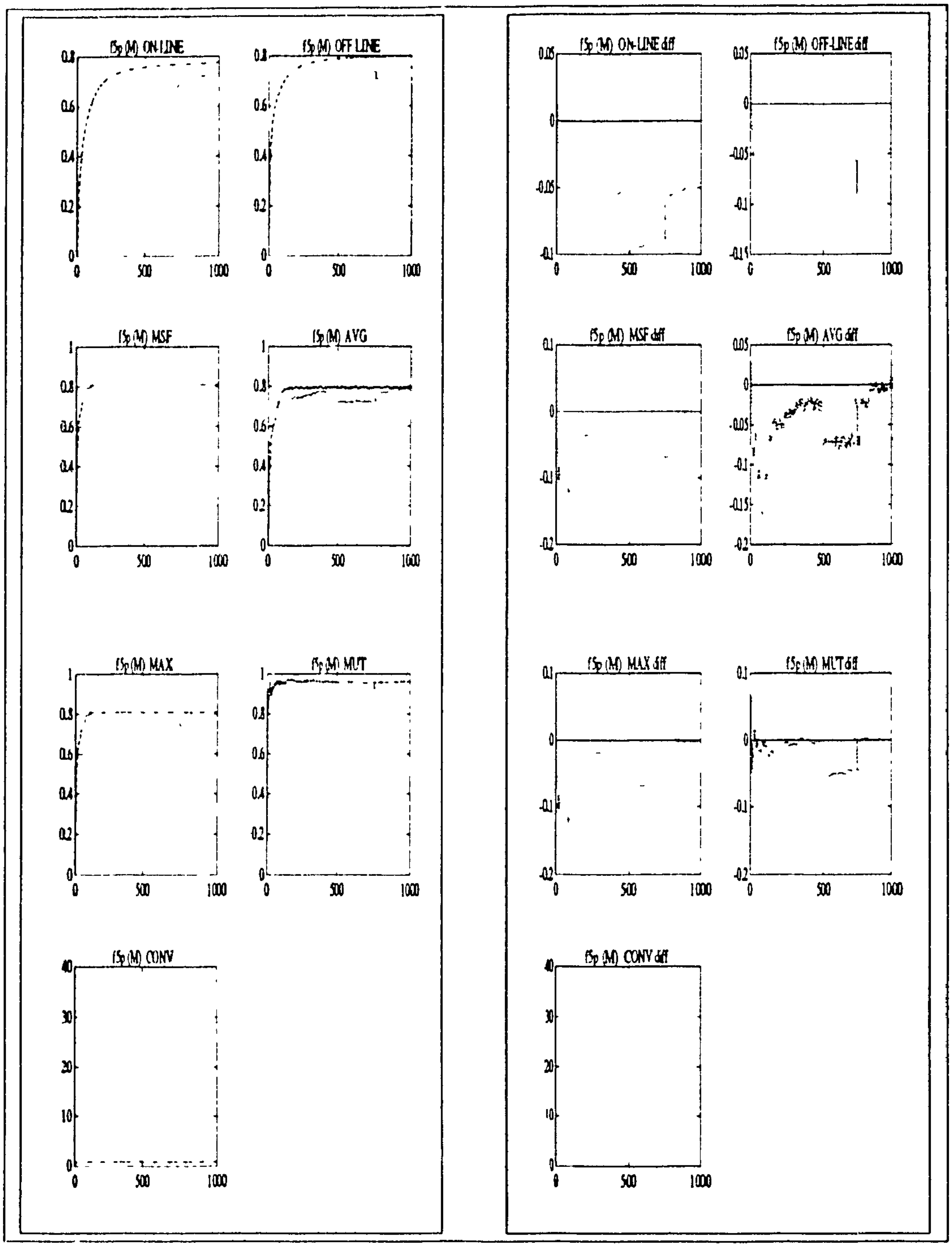


Figure 7.20: Results of F5' for mutation.

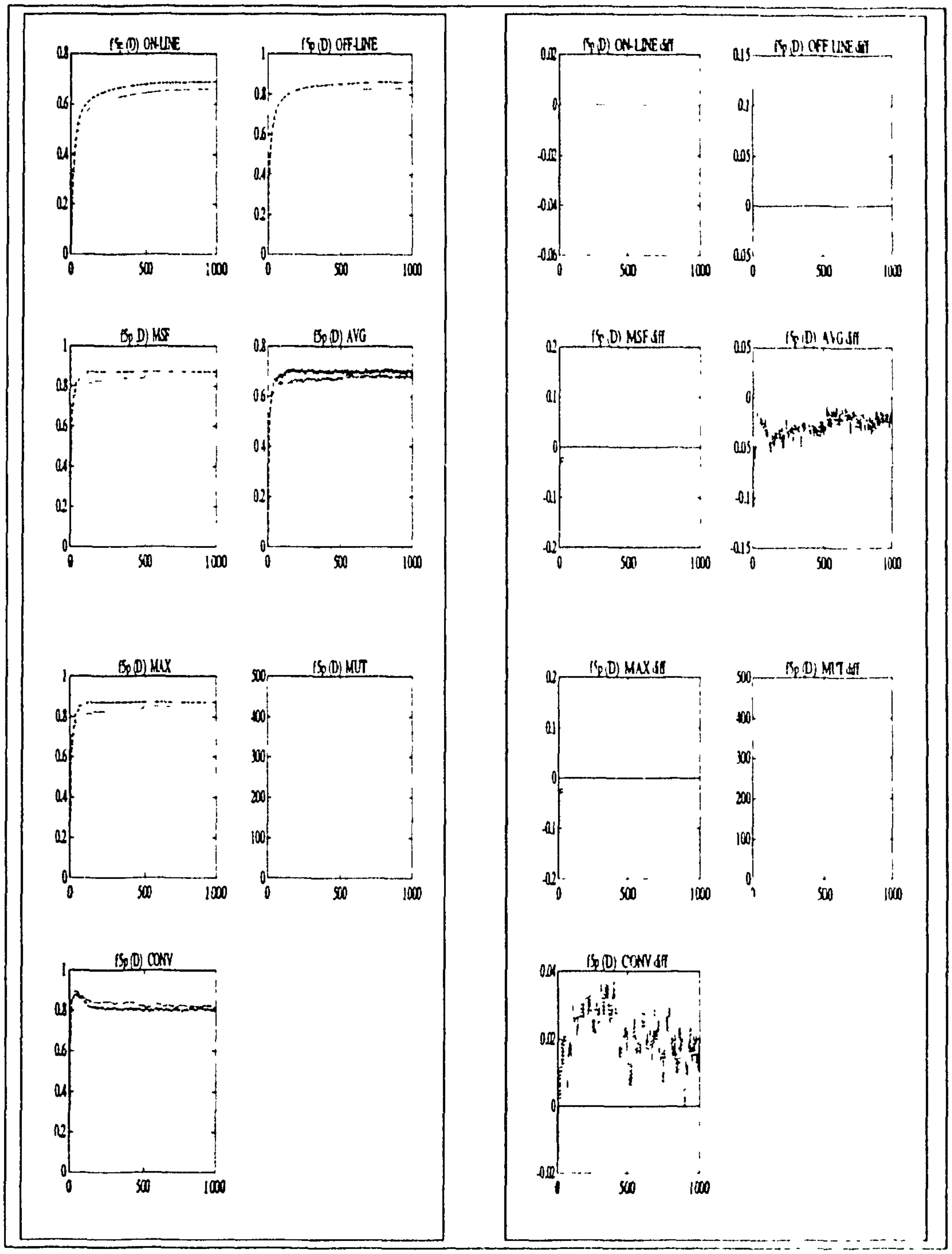


Figure 7.21: Results of F5' for desegregation.

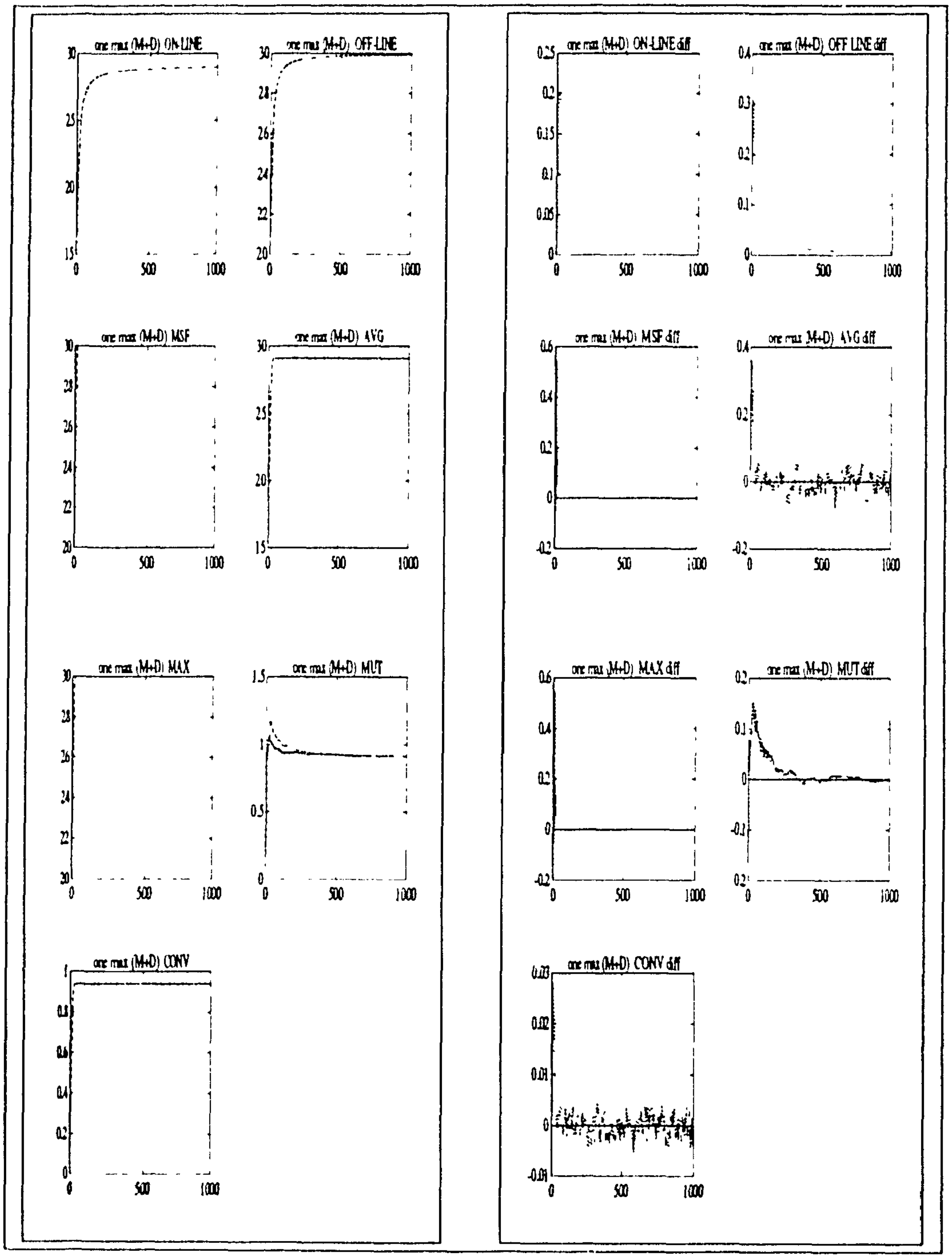


Figure 7.22: Results of One Max for mutation and desegregation.

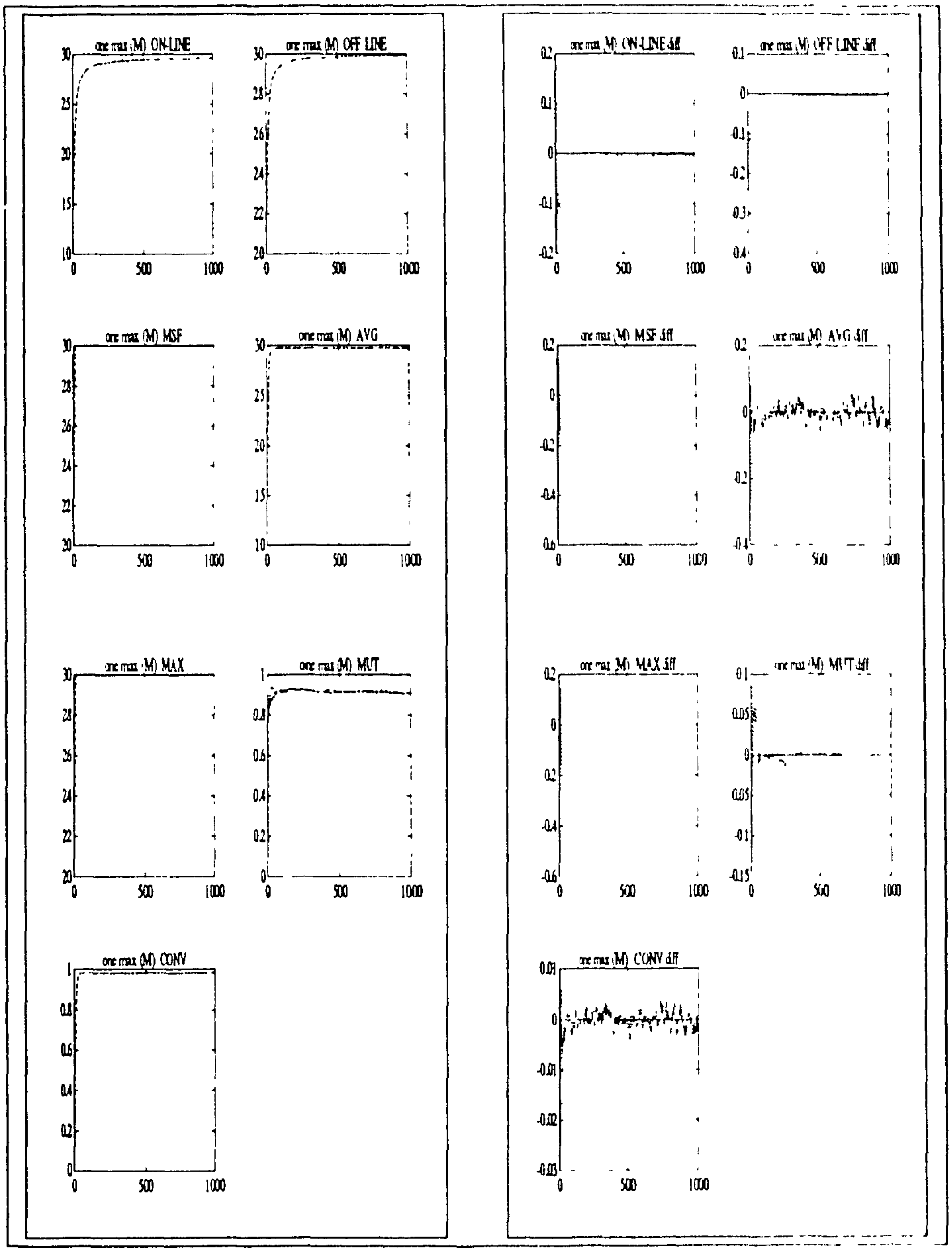


Figure 7.23: Results of One Max for mutation.

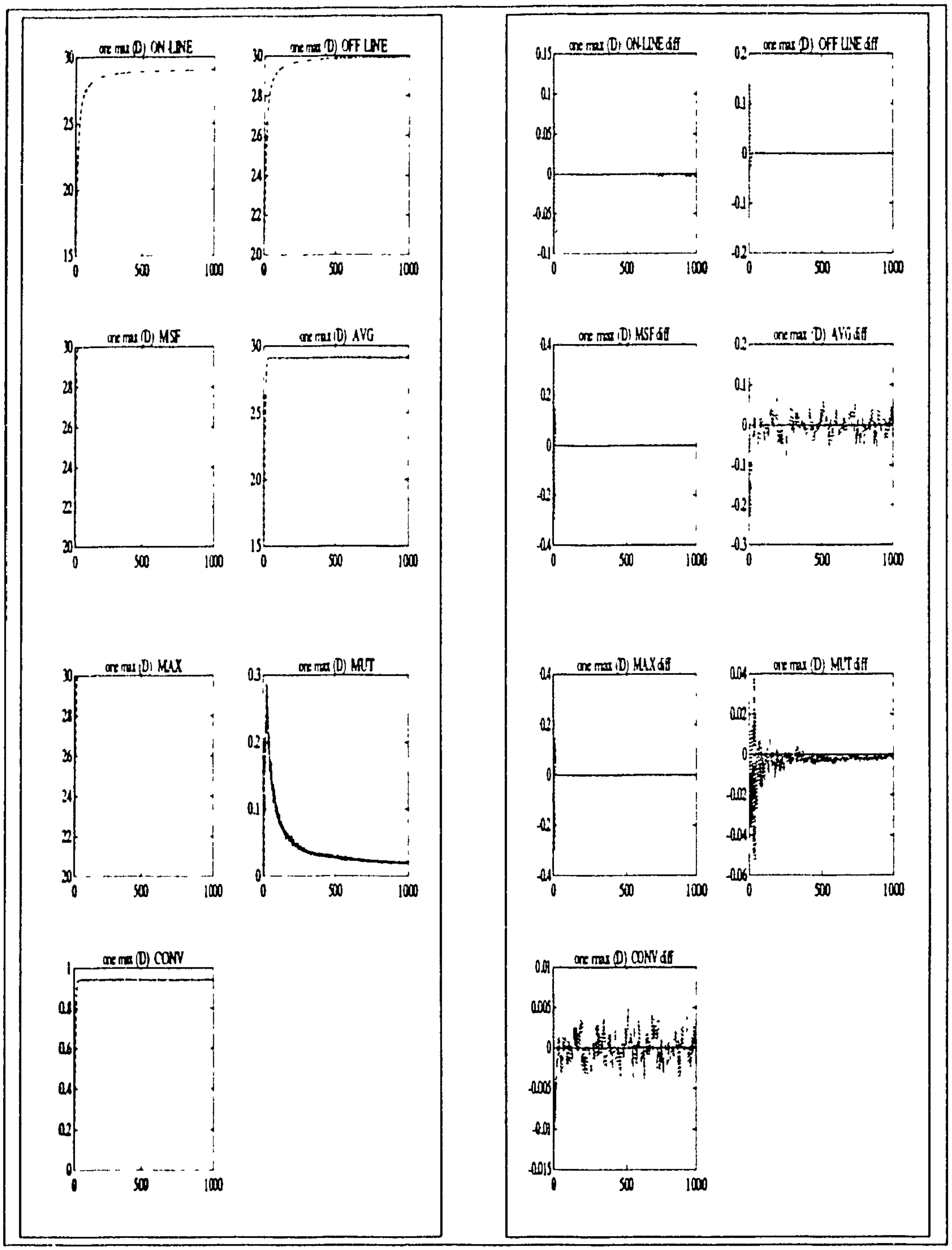


Figure 7.21: Results of One Max for desegregation.

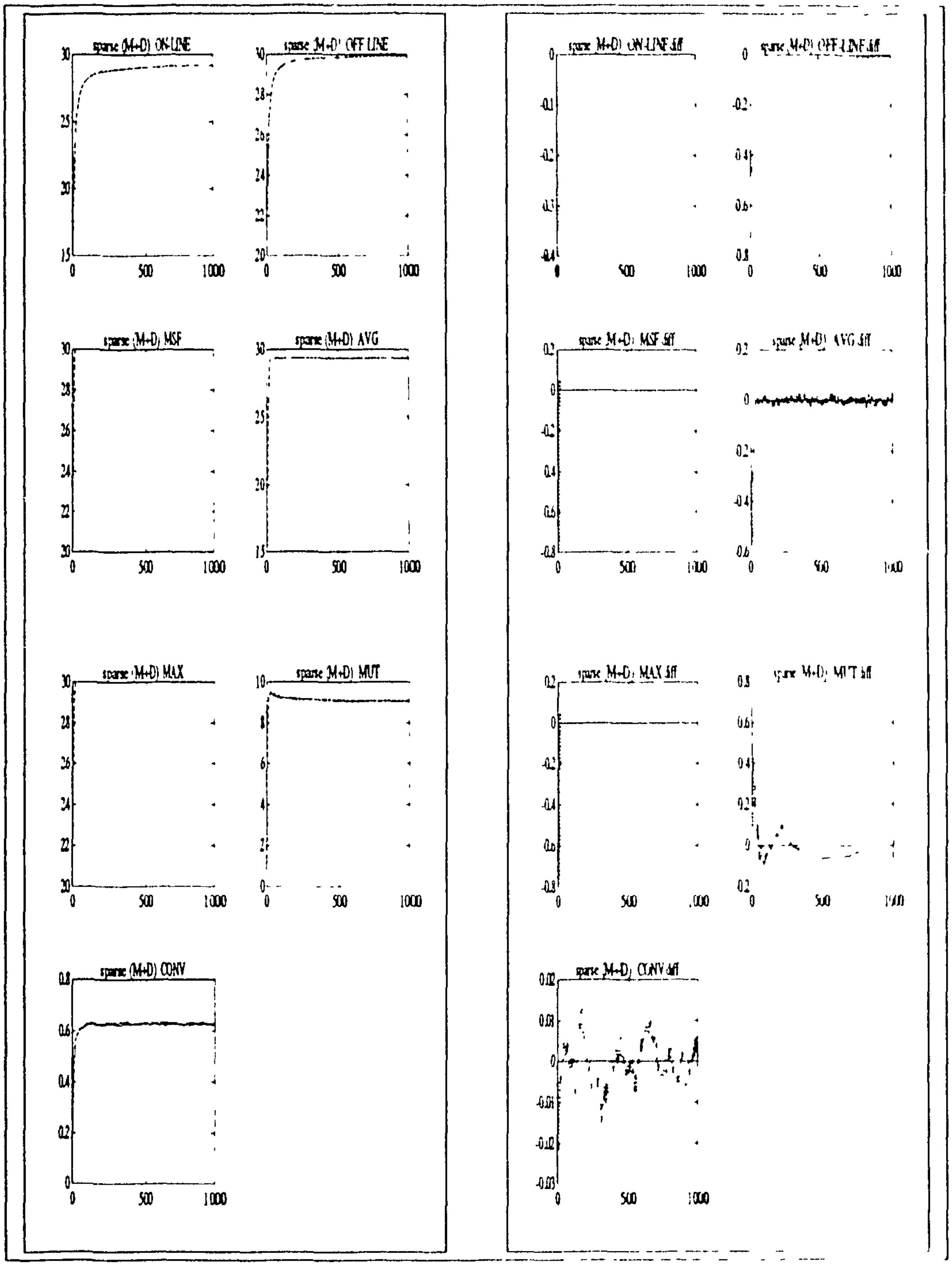


Figure 7.25: Results of Sparse One Max for mutation and desegregation.

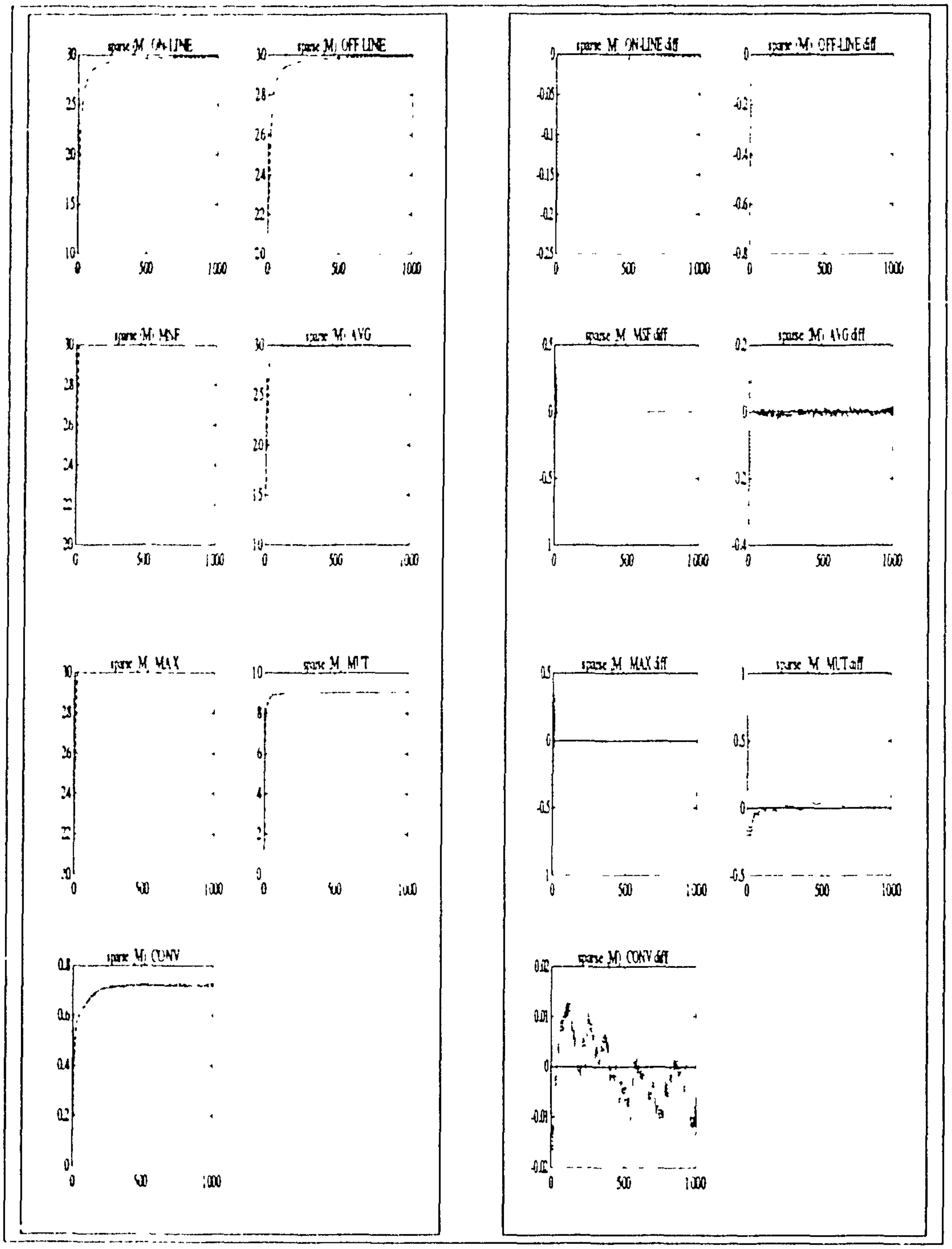


Figure 7.26: Results of Sparse One Max for mutation.

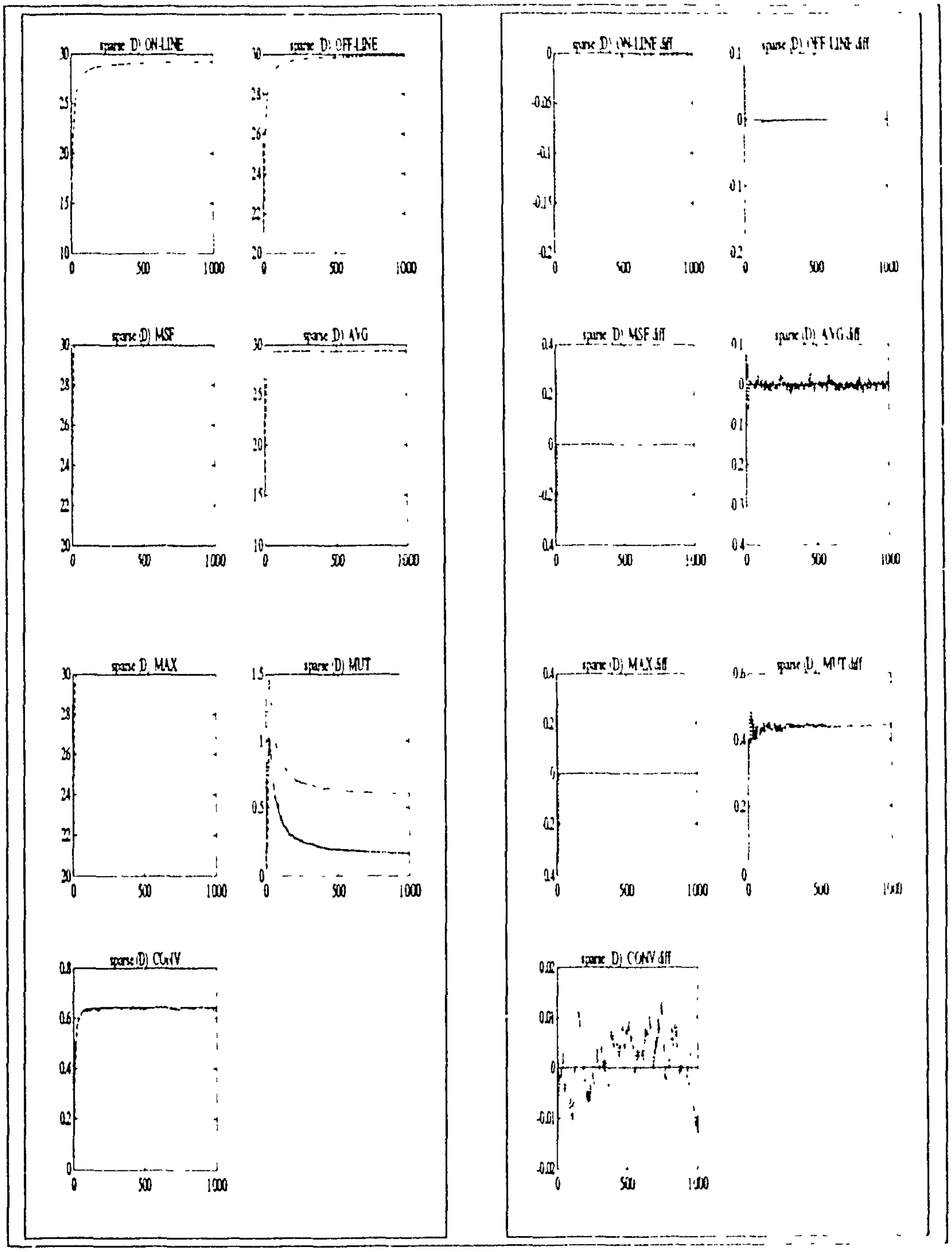


Figure 7.27: Results of Sparse One Max for desegregation.

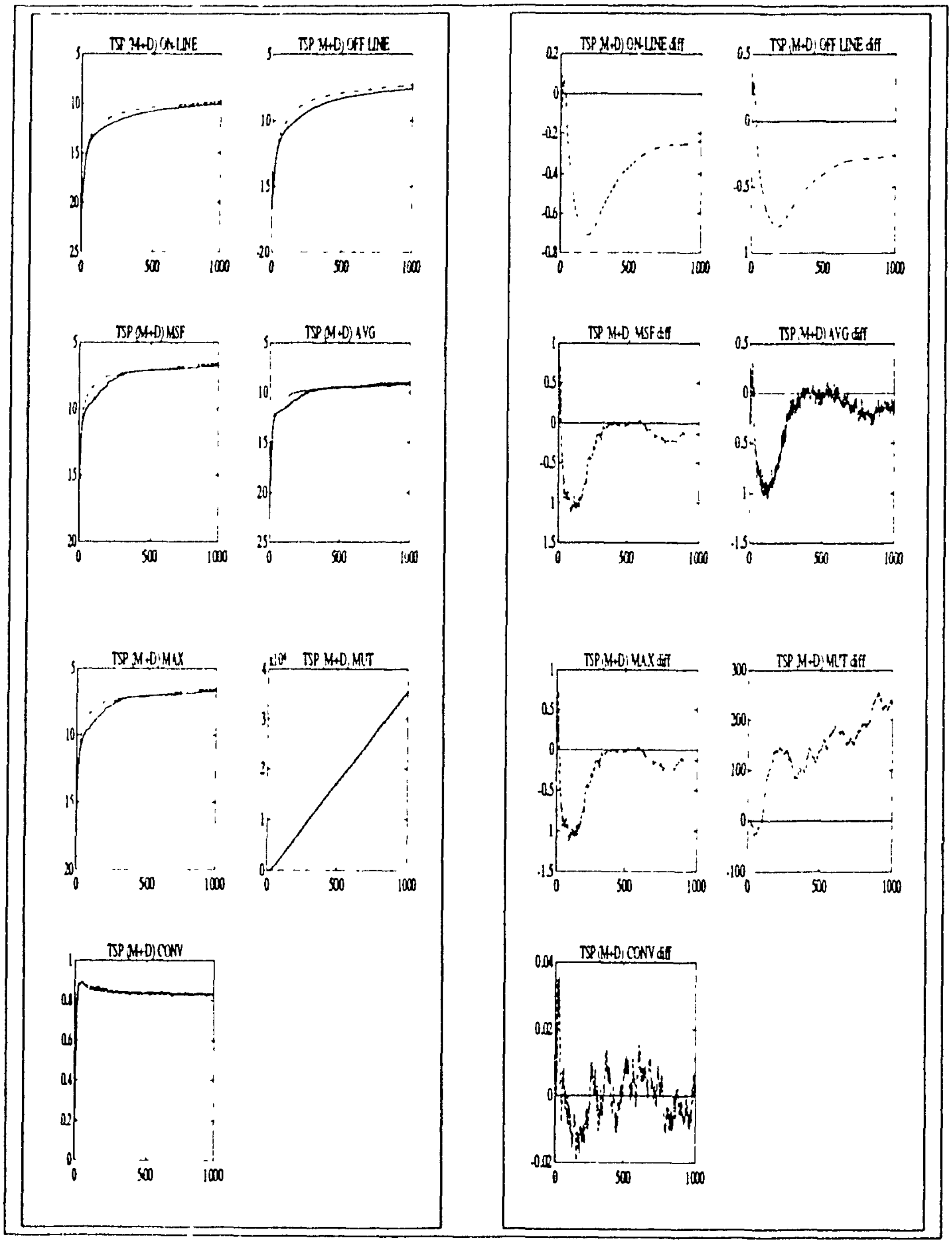


Figure 7.28. Results of Travelling Salesman Problem for mutation and desegregation.

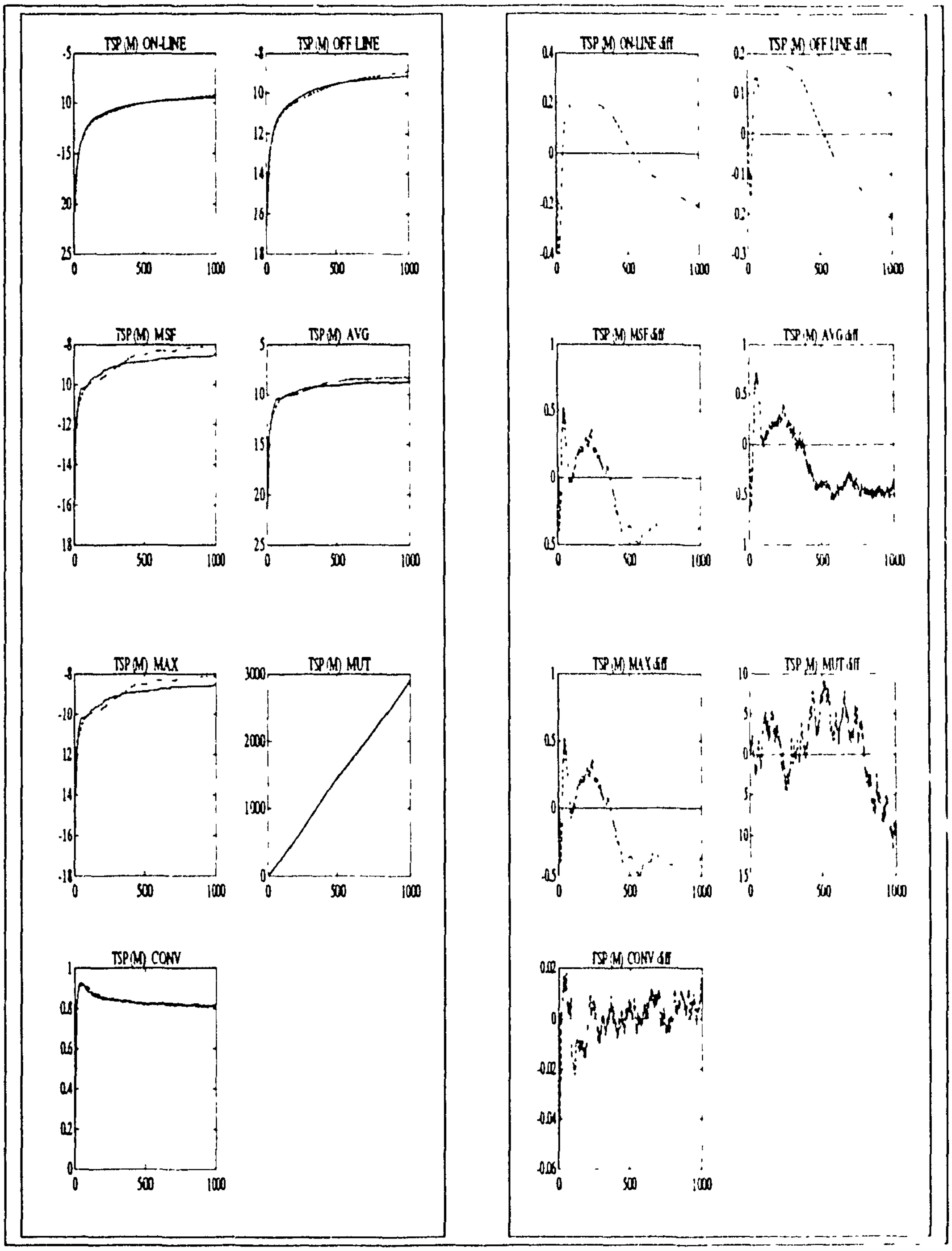


Figure 7.29: Results of Travelling Salesman Problem for mutation.

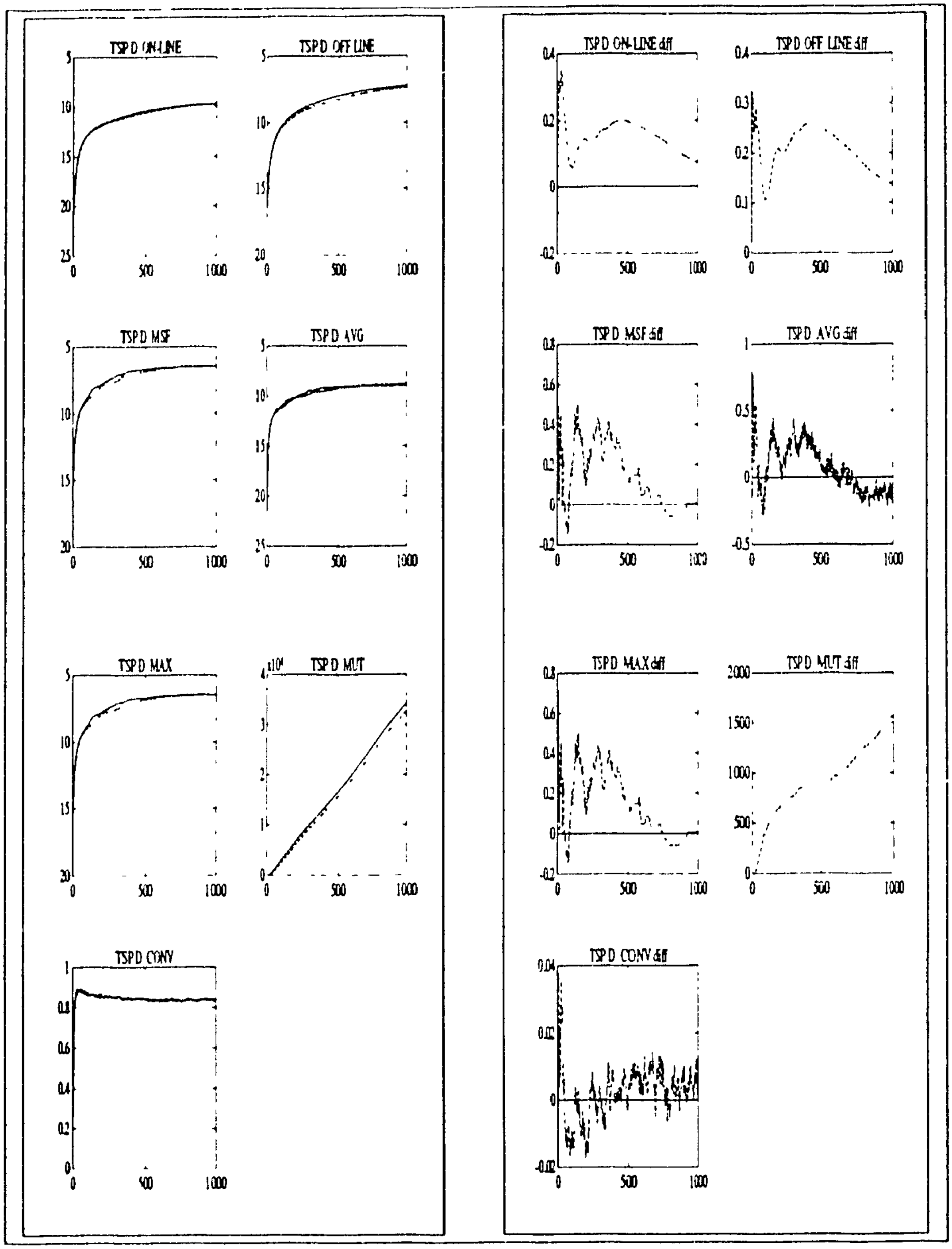


Figure 7.30: Results of Travelling Salesman Problem for desegregation.

Chapter 8

Future Research

In this thesis, two different areas of research were covered. First position independent crossovers were examined in an theoretical manner. While we were able to show that there is no one best crossover operator between uniform, modified uniform, and n-point shuffle crossover for combining schemas of all lengths, we did not give any heuristic for choosing which crossover operator should be used.

An idea that could be explored would be to map the crossover method into the chromosome. If there is a good crossover method for the problem, then its encoding should survive along with the "good" chromosomes it generates. This idea is similar to punctuated crossover.[21]

If would make the theory as presented more complete, if the Maple program which demonstrates lema 6, and the C program which demonstrates lema 13 could be proven. In particular it would be nice to remove the upper bound on the length of the chromosome for the shuffle crossover examination, that was necessitated by the exhaustive proof in lema 13.

The theory should also be re-examined to see whether the results hold, if crossover produces two children instead of one, as examined in this work.

The theorems presented in this work only deal with three specific crossover operators, it would be highly desirable to determine whether or not there is a function p_c^1 or p_c^2 that combines all schema lengths better than any other function, were $p_c^1(c)$ is the probability of copying c genes without examining the duplicate genes, $p_c^2(c)$ is the probability of copying c genes while examing the duplicates.

Second desegregation, which deals with mutation was examined empirically. I believe that it would be very difficult to make a theoretical comparison of the effects desegregation on population strength in much more detail then was given in chapter 6. So this leaves the worthiness of desegregation to be empirically verified in various different settings. Of particular interest would be to see the effects of desegregation where crossover produces two children, instead of one.

For those interested in massively parallel machines, the algorithm needs to be modified, so there does not have to be a central count of the alleles that are present in the population center.

Desegregation also has to be modified for populations that have varying chromosome lengths.[1] The algorithm also has to be modified for chromosomes that are not encoded based on the transformation to a vector for evaluation.

One possible improvement to desegregation would be to develop a method of realizing that the population is searching a very small space, and have the desegregation rate decrease. This would reduce the problem of having the average population strength being lower when desegregation is being used on problems with only one maximum. One possible implementation would be to make the desegregation rate of function of the measured population convergence.

The biggest problem not dealt with in this thesis is knowing when to stop the search process. While the problem was examined in the research leading to this thesis, no conclusive method could be found. Among the promising alternatives, is to examine the average and maximum population strengths with respect to time. If during a preset number of generations, the correlation between the strengths and the

[1] See *Don't Worry, Be Messy* , by David E Goldberg, Kalyanmoy Deb, Bradley Korb, in the Proceedings of the Forth Internation Caonference on Genetic Algorithms

generation number is approximately zero, then in a lot of cases the search process will not benefit from further search.

One of the problems that was encountered while performing experiments, was to find difficult functions to be optimized. Goldberg is currently working on this problem, by engineering difficult functions. It would be desirable if the GA community settkes on a more diverse set of functions to be examined then the De Jong [11] functions $F1$-$F5$. I believe that the Syswerda [25] encoding of the TSP problem, while not an efficient represention of the Travelling Salesman Problem, is a very difficult function. It also appears that function $F5'$ is also a very difficult function to be solved by GA's.

Appendix A

Mathematical Symbols

The table below lists the mathematical symbols used through out the work.

$\mathcal{O}(x)$	The order of the computational process is x
$(a,b]$	The open, and closed interval.
a_i	The chromosome at locus i from the first parent.
b_i	The chromosome at locus i from the second parent.
a_i	The i-th chromosomal segment that is copied from the first parent.
b_i	The i-th chromosomal segment that is copied from the second parent.
b_A	The length of a schema in the first parent.
b_B	The length of a schema in the second parent.
c	The number of genes copied during crossover.
c_i	A chromosome in the population occupying position i
strength(c_i)	The strength of chromosome c_i

d	The number of genes with the same alleles between the two parents.
k	A measurement indicating the number of chromosome segments that will be copied from the first parent during n point shuffle crossover.
l	The length of a chromosome.
n	The number of crossover points in shuffle crossover.
p	The number of genes passed during crossover.
$p_c(c)$	probability of copying c genes during crossover.
$p_c^{sh-n}(c)$	probability of copying c genes during n-point shuffle crossover.
$p_c(c)^{u+}$	probability of copying c genes during modified uniform crossover.
$p_c(c)^{u}$	probability of copying c genes during uniform crossover.
$p_p(p)$	probability of passing p genes during crossover.
$p_p^{sh-n}(p)$	probability of passing p genes during n-point shuffle crossover.
$p_p^{u+}(p)$	probability of passing p genes during modified uniform crossover
$p_p^{u}(p)$	probability of passing p genes during uniform crossover.
$p_s(b_A)$	The probability of passing a schema of length b_A during crossover.
$p_{sc}(b_A, b_B)$	The probability of passing a schema of length b_A from the first parent and a schema of length b_B from the second parent during crossover.

Appendix B

C program

The C program below, was used for demonstrating lema 13 in chapter 4. The program set the values of $b_A = 1$, $b_B = 1$, and $d = 0$ to ease the implementation details. The maximum bound on the length of the chromosome (l) was set to 300 since numeric overflows would occur at higher values.

A chromosome length of 300, is extermely large for published problems, so the upper bound is more a theoretical limit than a pratical one.

```
#include <math.h>
#include <stdio.h>

#define integer    double
#define zero      ((double) 0)
#define one        ((double) 1)
#define two        ((double) 2)
#define five       ((double) 5)
#define three_hundred ((double) 300)
```

```
/* ERROR EXIT CONDITION CODES */
#define set_size_negative    1
#define choose_negative      2
#define factorial_undefined  3


integer factorial(integer x)
  /*   x!   */
  {integer ans, i;

   if (x < 0) {_exit(factorial_undefined);};

   ans = one;
   for (i=two; i <= x; i = i + one) {
     ans = ans * i;}

   return(ans);}

\begin{verbatim}
double binomial(integer set_size,
          integer choose)
  /*     / set_size \
         |          |
         \ choose   /
  */
 {integer nominator, denominator;
  double ans;
  if (choose > set_size) {return(zero);}

  if (set_size < zero)
    {printf("E-SET_SIZE_NEGATIVE- set_size=%f\n", set_size);
     _exit(set_size_negative);}
  if (choose <  zero)
    {printf("E-CHOOSE_NEGATIVE- choose=%f\n", choose);
     _exit(choose_negative);}

  if (choose == set_size) {return(one);}
  if (choose == zero) {return(one);}

  ans = ((double) factorial(set_size)):
  ans = ans / ((double) factorial(choose));
  ans = ans / ((double) factorial(set_size - choose));

  return(ans); }

double psc(integer l,
           integer n)
  {integer k, denominator, nominator, p;
   double ans;

   k = (integer) floor((double)((n + one) / two));

   denominator = 2 * binomial(l-one, l-n-1) * l * l;
   nominator = 0;
   for (p=one; p <= l; p = p + one) {
     nominator = nominator +
       p * (l-p) * (binomial(p-one,k-one)*binomial(l-p-one,n-k) +
                    binomial(l-p-one,k-one)*binomial(p-one,n-k));}

   ans = ((double) nominator) / ((double)denominator);
   return(ans);}


int main(int argc, char *argv[]) {

  double last, prob;
  integer l,n;

  for (l=five; l <= three_hundred; l = l + one) {
    last = ((double) 0.0);
    for (n=one; n < l; n = n + two) {
      prob = psc(l, n);
      printf("%4d %4d ", ((int) l), ((int) n));
      if (prob > last) {
        printf("%8.6f>   %8.6f\n", prob,last);}
      else if (prob < last) {
        printf("%8.6f   <%8.6f\n", prob,last);}
      else {
        printf("%8.6f =  %8.6f\n", prob,last);}
      last = prob;}
    printf("\n");}
```

```
return(0),}
```

Bibliography

[1] *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum, 1985.

[2] *Proceedings of the Second International Conference On Genetic Algorithms*, 1987.

[3] D.H. Ackley. A connectionist algorithm for genetic search. In 1GA [1], pages 121–135.

[4] D.H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, 1987.

[5] J.E. Baker. Adaptive selection methods for genetic algorithms. In 1GA [1], pages 101–111.

[6] J.E. Baker. Reducing bias and efficency in the selection algorithm. In 2GA [2], pages 14–21.

[7] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, Micheal B. Managon, and Stephen M Watt. *WATCOM MAPLE (Version 4.3 for VMS)*. Waterloo, Ontario, 1989.

[8] Larry J. Eshelman, Richard A. Caruana, and J. David Schaffer. Biases in the crossover landscape. In Schaffer [20], pages 10–19.

[9] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[10] John J. Grefenstette and James E. Baker. How genetic algorithms work: A critical look at implicit parallelism. In Schaffer [20], pages 20–27.

[11] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[12] J.H. Holland, K.J.Holyoak, R.E. Nisbett, and P.R. Thagard. Classifier systems, q-morphisms, and induction. 1988.

[13] Prasanna Jog and Dirk Van Gucht. Parallelisation of probabilistic sequential search algorithms. In 2GA [2], pages 170–176.

[14] K.A. De Jong. *An Analysys of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, The University of Michigan, Ann Arbor, Michigan, USA, 1975.

[15] Marc Andrew Pawlowsky and Adam Krzyzak. Desegregation in genetic algorithms. In Denis Laurendeau, Paul Fortier, Dominic Grenier, and Xavier Maldague, editors, *Canadian Conference on Electrical and Computer Engineering*, pages 55.3.1–55.3.4. Canadian Society for Electrical and Computer Engineering, 25–27 September 1991.

[16] Chrisila B. Pettey, Michael R. Leuze, and John J. Grefenstette. A parallel genetic algorithm. In 2GA [2], pages 155–161.

[17] George G. Robertson and Rick L. Riolo. A tale of two classifier systems. 1988.

[18] Kenneth H. Rosen. *Discrete mathematics and its aplications*. Birkhäuser mathematics series. The Random House, New York, first edition, 1987.

[19] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In 1GA [1], pages 93–100.

[20] J. David Schaffer, editor. *Proceedings of the Third International Conference On Genetic Algorithms*, San Mateo, California, June 4–7 1989. Morgan Kaufmann Publishers, Inc.

[21] J. David Schaffer and Morishima A. An adaptive crossover distribution mechanism for genetic algorithms. In 2GA [2], pages 36–40. was SCHAFFER.87

[22] J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer [20], pages 51–60.

[23] J. David Schaffer and J.J. Grefenstette. Multi-objective learning via genetic algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 593–595, 1985.

[24] Nicol N. Schraudolph and Richard K. Belew. Dynamic parameter encoding for genetic algorithms. CSE Technical Report 90-175, Cognitive Computer Science Research Group, Computer Science & Engr. Dept., University of California, San Diego, July 1990.

[25] Gilbert Syswerda. Uniform crossover in genetic algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference On Genetic Algorithms*, pages 2–9, San Mateo, California, June 4–7 1989. Morgan Kaufmann Publishers, Inc.

[26] Reiko Tanese. Parallel genetic algorithm for a hypercube. In 2GA [2], pages 177–183.

[27] Darrell Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In Schaffer [20], pages 116–121.