LA-UR-94-895

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

TITLE: AN INVESTIGATION OF GENETIC OPERATORS FOR CONTINUOUS PARAMETER SPACE

AUTHOR(S): Tal Grossman, CNLS/T-Division
Yuval Davidor, The Weizmann Institute of Science

SUBMITTED TO Conference: "Parallel Problem Solving From Nature" October 9-12, 1994
Israel (Jerusalem)

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO. 836 R4
ST. NO. 2629 5/81

AN INVESTIGATION OF GENETIC OPERATORS FOR CONTINUOUS PARAMETER SPACE

Tal Grossman
Theoretical Division and CNLS
MS B213
Los Alamos National Lab
Los Alamos, NM 87545, USA.

Yuval Davidor
Department of Applied Mathematics and Computer Science
The Weizmann Institute of Science
Rehovot 76100, ISRAEL.

March 9, 1994

Abstract

The sucess of a genetic optimization algorithm in continuous parameter space depends on the recombination (crossover) operators that it uses. In this paper we consider a wide spectrum of such operators within a unified framework and study their relative importance in the search process. We consider four basic types recombination operators which cover the relevant exploration potential of a continuous space: Interpolation, Extrapolation, Exchange and Mutation. Each of these basic types may have several variants. We characterize the various operators and their variants by their *spatial sampling* properties and examine their contributions to the search by applying different mixtures of the operators in several benchmark problems. The results suggest that the optimal mixture of operators may depend on the problem. But, in general, all basic types are needed for efficient optimization.

1 Background

Genetic algorithm models (GAs) that manipulate continuous parameters have been occasionally attempted throughout the GAs' history. In particular, continuous models have been attempted in real-life applications where continuous parameters constitute a natural representation of the problem domain [2, 3, 4, 9]. Despite their reported advantage over binary in certain problems, continuous GA models (CGAs) were neither used nor investigated extensively because their alleged efficiency was paradoxical to common interpretations of the schema theorem.

Traditional interpretations of GAs suggest that a parameter that can receive many values is worrisome due to the risk that many of its values will never be examined unless excessively

large population sizes and mutation rates are used. Holland's original analysis of schemata processing and implicit parallelism has been so fundamental, that it made continuous parameters counter-intuitive.

In recent years, a few theoretical works suggested why the apparently paradoxical efficiency achieved by various CGA models can be explained in terms of fundamental GAs theory [1, 6, 7, 14]. Though CGAs were used before (widely discussed in [4] and in the application to robot path planning in [2]), much of the important documented studies on CGA's were only published in 1991.

First should be noted Goldberg's theoretical work on *virtual alphabets* [7] The theory of virtual alphabets suggests that the explicit growth or decay of symbols in successive generations of a binary representation also occurs in the continuous space, only implicitly. The range of values in the continuous interval are dynamically distributed in such a way that only few segments in that range are sampled frequently. Goldberg called these attracting intervals virtual alphabet and suggested that in a continuous parameter space a CGA only searches among a limited number of the virtual alphabets, hence includes a much reduced range of possible values than what is intuitively associated with continuous parameters.

Though the above rational is an elegant explanation to the paradox why CGAs work well, it still leaves much of the mystery why continuous models supersede binary representations on many problems, and what type of reproductive/recombination operators one should use to maximize the utility of CGAs.

Several crossover mechanisms were offered so far: Wright [14] suggested that three offspring are produced from two parents p_1 and p_2 such that one offspring is the mid point between p_1 and p_2, one is $1.5p_1 - 0.5p_2$, and the third is $-0.5p_1 + 1.5p_2$. Janikow and Michalewicz suggested a mid point crossover [8] and some elaborated mutation operator. Radcliffe suggested a *flat crossover* which chooses a point a long the line connecting p_1 and p_2 with uniform probability [13]. Eshelman and Schaffer [6] suggested recently an elaboration of Radcliffe's flat crossover they called *blend crossover* which extends the line from which an offspring is selected beyond the points defined by the two parents (a kind of extrapolation).

In this paper we extend previous work on crossover mechanisms for CGA mentioned above. The basic goal of this work is not to advocate a specific version of CGA, but rather, to consider a full range of possible operators and understand their contribution to a successful optimization process. We analyze and test four reproductive operators and their variants, which constitute the basic reproductive operations possible in continuous space. These operators are characterized by their spatial sampling properties, and their relative role in the search process is tested on several problems. We suggest that with a mixture of these operators, better and more general exploratory capability than what was possible with the above mechanisms is obtained.

2 The Model and Operators

The problem to be solved by the algorithm is defined as an optimization problem of a real function $F(x)$ of a real vector x. The dimensionality of the vector x (i.e. the number of problem parameters to be optimized) is d. Each of the x_j $(j = 1..d)$ parameters can be

restricted to a given interval $I_j = [l_{minj}, l_{maxj}]$. The goal of the optimization problem is to find real vector x that corresponds to the global minimum (or maximum) of $F(x)$ over the d dimensional interval

$$I = \prod_{j=1}^{d} [l_{minj}, l_{maxj}]$$

2.1 Description of the Algorithm

In the continuous scheme, each individual is represented simply by the vector $x^i \in I$, namely each of the components x^i_j $(i = 1..n\ ,\ j = 1..d)$ is a real number from the interval I_j, where n is the size of the population.

Like the usual genetic algorithm scheme, our algorithm alternates between two major phases, selective reproduction and crossover-mutation The selection method we use here is more or less standard. Starting with an initial population of n individual vectors x^i, each of these vectors is evaluated by calculating the value $f_i = F(x^i)$, which is the objective function, or the 'fitness' of individual i. A probability p_i is then associated with each individual according to the rule (for a minimization problem):

$$p_i \propto \frac{f_{max} - f_i}{f_{max} - f_{min}}$$

where f_{max} (f_{min}) is the worst (best) value amongst the f_i over the current population.

A new population is then created by selective reproduction. Each individual of the new population will be selected from the old generation according to the probabilities p_i. This process, which favors those individuals with small f_i, produces a new set of n vectors x_i in which a subset of the old population is present (some of them may be duplicated, of course).

The next phase is the crossover-mutation phase, in which new vectors are created from those of the selected population. In this process, a set of 'genetic operators' is used in order to create the 'offspring' of pairs of 'parent vectors'.

2.2 The Reproduction Operators

The underlying assumption of the approach presented here is that the relevant information for the optimization task is simply the location of each individual (vector) in the d dimensional space, and its relative fitness. Therefore, the operators described below are constructed in such a way as to extract different elements of this geometrical information in order to create an offspring with improved performance. In what follows we describe four operators. Three of them, interpolation, extrapolation and exchange are pairwise operators. Namely, they take two parents and create an offspring which will inherit some of the information contained by the parents. The forth operator, mutation, is an unary operator which introduces a low probability perturbation of a small magnitude to members of the parent population.

2.2.1 Interpolation

The interpolation operator creates an offspring which is a weighted average of its two parents. To be more specific, the new vector x^{new}, is chosen as a random point on the line connecting the two parent vectors, x^1 and x^2:

$$x^{new} = ax^1 + (1-a)x^2.$$

The simplest choice of the parameter a is $a = 0.5$, and then the offspring is just the midpoint average of its two parents. This is the operator used in [8] and variation of that used in [14], and it will be referred to here as *mid-point interpolation*. In a more general realization of this operation, one can choose a to be a random number from the interval $[0, 1]$ like in [13]. We shall call this operator *line interpolation*.

2.2.2 Extrapolation

Taking into account the values of two parents, it is possible to find a better solution (i.e. a vector with a smaller F), by moving outward from the better of the two along the line connecting the two. This idea motivates the extrapolation operator. Like the interpolation, this operator creates the offspring as a random point on the line connecting the two parents, but this time the point will be on the interval from the better parent, say x^2, and away from the other parent x^1,

$$x^{new} = a(x^2 - x^1) + x^2.$$

Like the previous operator, there are two version of the extrapolation operator. In the *point extrapolation* a is a pre-specified parameter (usually $a < 1$). While in the *line extrapolation* version, a is a random number from the interval $[0, \textit{extrapolation range}]$, (where extrapolation-range is set externally). Constructed this way, the offspring can be 'out of bound' in one or more of its components. In such a case, each of these components is truncated to its nearest bound. This operator also prevents the genetic process from being trapped within the convex hull of the current population while exploiting meaningful genetic information.

2.2.3 Exchange

The exchange operator (also used in [8]) is similar in operation to the binary uniform crossover which mixes whole parameters and constructs an offspring from two parents by choosing for each parameter x_j^{new}, the corresponding parameter of one of the parents, x_j^1, or x_j^2 with equal probability (the equivalent binary uniform crossover).

2.2.4 Mutation

The mutation operator creates an offspring which is close to the original parent, but differs from it by a small random vector. In our implementation

$$x_j^{new} = x_j^1 + a_j m_j \quad , \quad (j = 1..d)$$

a_j is a random number in $[0.5, 0.5]$, and $m_j = C \times (l_{maxj} - l_{minj})$ where C is a (constant) mutation-range parameter. Again, an 'out of bound' offspring is truncated to the nearest bounds.

2.3 The application of the operators

After the selection process, the new population is created by taking each pair x^i and x^{i+1}, and selecting at random one of the three pairwise operators according to the probabilities: p_{int}, p_{ext} and p_{exch} (interpolation, extrapolation and exchange, respectively). When the sum $s = p_{int} + p_{ext} + p_{exch} < 1$, a fraction $1 - s$ of the vectors is copied to the next generation without a change (i.e. the parent x^i is copied to the new generation with no change). The best vector of the previous generation, is copied unchanged to the next generation as well. The last operator, mutation, is then applied with a probability p_{mut} (the mutation rate) to all the new vectors apart from the best one. The new population is now evaluated, and a new cycle of reproduction begins, and so on.

3 Crossover in Continuous Parameter Space

Like in binary representations, in which the choice of crossover depends on its schemata exploration capabilities in the context of the problem domain [5], the choice of continuous recombination operators is guided according to their exploration capabilities in the continuous space. However, in contrast to binary representations in which the creation of new search points is limited to parameter mixing, in continuous space new search points can be created through many pairwise operations not restricted to these values. To take full advantage of the continuous space, these operators should optimally explore the continuous space in the neighborhood of the parent population. The role of the four suggested operators is to perform different types of local search in a neighborhood defined by two parent vectors. In this section we present a simple geometrical picture of the *spatial sampling* characteristics of the different operators. This picture is complementary to the "virtual alphabet" [7] and the "interval schemata" [6] analysis of CGA's.

Assuming that the two parent vectors have already been selected, what are the locations that should be considered as possible future search points? If we have a pair of two good points, then a possible place for a local optimum is somewhere in between them. Such operation is obtained by the interpolation operator (Figure 1).

The continuous crossover analog of the binary crossover is the exchange operator. The exchange operator will create a new point in one of the corners of the rectangle spanned by the parents (Figure 1). This operation fits into the usual concept that a good schema (i.e. a good choice of a subset of the parameters) will have a good chance to proliferate this way in the population.

However, the use of only the interpolation and exchange operators as in [8, 13, 14] will limit the search within the d-dimensional rectangle defined by the existing population of vectors (and their exchanged versions). As suggested earlier, additional ways to explore the parameter space are needed. Indeed, random mutations can help (if the random increments are large

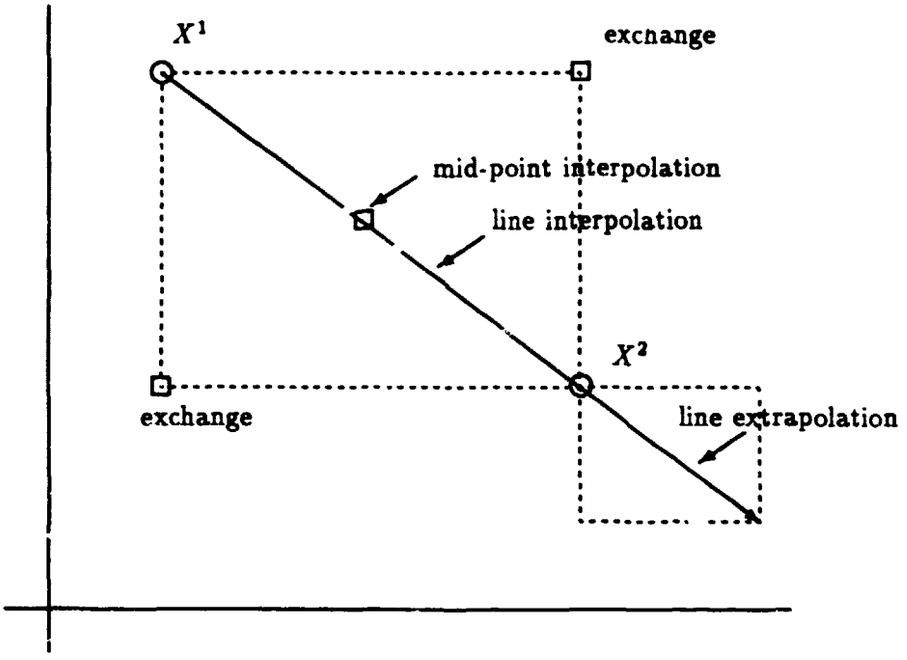

Figure 1: A schematic description of a 2-D offspring created by the interpolation, extrapolation and exchange operators, and the two operator variants: 'point' and 'line'. X^1 and X^2 are the parent vectors where X^2 having a better fitness.

enough), but the extrapolation operator described above enables the search to escape these limits in a more directed way. In particular, this operator enables the algorithm to search for optimal solutions *on the surface of the available parameter space.*

The extrapolation operator is motivated by the idea of hill climbing. Two points in the neighborhood of a minimum can be arranged in such a way that the line between them points to the general direction of the local optimum (Figure 1).

It is possible to fit the exchange and interpolation into a unified general operator which is termed here a *rectangular interpolation*. This operator will create a new vector with uniform distribution over the entire rectangle spanned by the two parents. This will be done by taking a different random number $a_j \in [0, 1]$ for each component $j = 1..d$, and then

$$x_j^{new} = a_j x_j^1 + (1 - a_j) x_j^2.$$

Similar operators, such as the arithmetical crossover suggested by Janikow and Michalewicz [11, 10], which operate on part of the components will all produce offsprings within the same rectangular region. In a similar manner we can define the *rectangular extrapolation* operator, which creates the new vector

$$x_j^{new} = a_j(x_j^2 - x_j^1) + x_j^2$$

where

$$a_j \in [0, \textit{extrapolation range}].$$

The dashed frames in Fig.1 show the regions covered by the rectangular operators in two dimensions. It is important to remember that the neighborhood defined by two parents can span a large portion of the available space. Since we choose the parents at random, they may be located at far away regions of the search space. Note also that for all the pairwise operators discussed above, the average distance of an offspring from it's parents is proportional to the distance between them. Hence the importance of the mutation operator in a case of a premature convergence of the entire population into a region of a local minimum. In this work, we do not address the problem of non trivial constrains on the search space (see [10, 11, 12]. Since we consider a rectangular search space, the simple truncation is sufficient to keep the population within the rectangular boundaries. More elaborate mechanisms (like projecting an offspring to the neasrest boundary) will be needed in cases of other types of constraints.

4 Experimental Results

In this section we study the performance of the algorithm on three test problems that were presented and used for the study of various GGA's. These are the functions f3, f6 and f14 as defined in the test suite used by Eshelman and Schaffer [6].

Our goal is to demonstrate the efficiency of the algorithm, to gain some understanding of the contribution and importance of the different operators in different situations, to find the optimal combinations of these operators and to suggest a generally good choice of the "application rate" parameters: p_{int}, p_{ext} and p_{exch}.

The first function, f14, is an optimal control problem that was used as a test function in [8, 6]. It is a high dimensional dynamic control problem, in which the control vector x should minimize the function

$$F(x) = y_0^2 + \sum_{j=1}^{d} [x_j^2 + y_j^2]$$

where $y_j = y_{j-1} + x_j$, with $y_0 = 100$, $d = 45$ and the components of x are restricted to the intervals $I_j = [-200, 200]$. The optimal solution with these parameters is $F^* \simeq 16180.4$.

For each choice of operators and parameters, 30 search runs were performed, 5000 generations each. Each of these runs was initialized with a different population of random vectors, uniformly distributed over the allowed region. Population size was 60. The performance measure which we use is the ensemble average of the best solution found in each run, and its deviation.

First we address the issue of choosing an optimal combination of pairwise operators for a given problem domain. We use f14 as an example. There is a total of nine pairwise operations obtained from two basic operators: interpolation and extrapolation, and their three possible variants: 'point', 'line', and 'rectangular'. The three variants represent the number of geometrical degrees of freedom restricting the two operators. Hence, the 'point' version fully specifies the result of the two operators to a specific point. The 'line' version allows one degree of freedom (for the offspring vector), while the rectangular version leaves d degrees of freedom per vector.

Extrapolation	Interpolation					
	point		line		rectang.	
point	16300.0	(50.0)	27300	(3000)	240000	(30000)
line	16184.5	(1.5)	16181.9	(0.4)	16186.0	(2.0)
rectang.	16190.0	(6.0)	16184.2	(1.0)	16185.2	(1.3)

Table 1: the control problem - operator choice comparison (the deviation in parentheses).

Table 1 shows the performance of each of the 9 possible choices for the interpolation and extrapolation operators (with extrapolation range 0.5, p_{int}, p_{ext} and p_{exch} = 1/3, and the mutation rate and range are 0.02 and 0.01 respectively).

Investigating Table 1 shows that for this problem the optimal choice of operators is line extrapolation with line interpolation. Another good choice of operators is the line interpolation with rectangular extrapolation. These choices not only produce close to an optimal solution, but also find it very consistently (as indicated by the small deviation). It should be emphasized though, that we have studied here only combinations in which there is just one variant of each of the two operators - interpolation and extrapolation. In a more general experiment one should be able to use any combination of operators.

Next we would like to study the influence of the rates p_{int}, p_{ext} and p_{exch} on the performance of the algorithm. Table 2 presents the performance of the algorithm with line interpolation and extrapolation, with different sets of values of those probabilities (the rest of the parameters are the same). There are three groups of parameter sets in the table. In the first, the sum $s = p_{int} + p_{ext} + p_{exch}$ is unity. In the second group s = 0.9, and in the third s = 0.8. A smaller s values usually decreases the performance, although the combinations (0.4,0.4,0.1) and (0.3,0.3,0.2) perform close to the optimal. It is evident that the 'equi-partition' between operators, i.e. $p_{int} = p_{ext} = p_{exch}$ = 1/3 is more or less the optimal choice. It is certainly clear that all the operators are necessary for good performance (for details see Table 2). In general, performance is not too sensitive to small deviations from $p_{int}, p_{ext}, p_{exch}$ equi-partition.

Of course, the optimal choice of the operators and the different parameters may depend on the specific optimization problem. To illustrate this, we present a short account of several experiments done with the two other test functions, f3 and f6.

The f3 test function (originally defined by De Jong), is a five dimensional "staircase" function. Since it is a monotonic function, it is obvious why the extrapolation operator should do well on this problem. The exchange operator is also expected to be important, as the different parameters have independent contributions to the function (no epistasis). Since in all cases the algorithm found the global minimum (which is the small step where all the variables are smaller than -5.0) within 100 generations, we use here a different measure of performance: the average number function evaluations needed to find the solution (and the deviation).

Table 3 shows the performance of each of the 9 possible choices for the interpolation and extrapolation operators, with extrapolation range 0.5, p_{int}, p_{ext} and p_{exch} = 1/3. In this problem stronger mutations were needed in order to achieve good performance, the mutation rate and range are both 0.2 . The population size is 40.

It is evident that for this problem the optimal choice of operators is line extrapolation with point interpolation. Since the point version of the extrapolation operator always produces the offspring as far as possible, this is the version that better exploits the monotonicity of this test function.

In Table 4 we present the performance for different choices of the rates. As expected, the exchange and extrapolation are the dominant operators for this problem - the best performance obtained without interpolation ! Nevertheless, the equipartition choice still offers close to optimal performance.

Yet another situation is encountered with the third function, f6. This is a two dimensional cylindrically symetric wave, with a decaying envelope. It has a global minimum at the origin and many suboptimal rings around it. It was found that the best operators in this case are the rectangular interpoiation and extrapolation. The reason is probably their ability to sample larger areas, and hence they have a better chance to find good enough points in the vicinity of the small optimal region. This is a harder problem and needed a finer tuning of the parameters in order to achieve good performance. With a population size of 50, mutation rate of 0.2, mutation range 0.05 and and equipartition of the operators, the average number of function evaluations needed to find the true minimum was 8400 (dev. 5400).

5 Summary

We studied a simple and general genetic algorithm which uses continuous parameters, and recommend to use this formal of representation and algorithm when continuous parameters are a natural representation of the solution space of a given problem. Four basic 'genetic' operators (and their variants) which suit this representation, were presented. A preliminary study of performance on a few benchmark problems has established the potential of using a mixture of genetic operators for optimization of functions in real domains.

This study also gives some intuition as to the right choice of the operators and parameters. Our recommendation for a generic CGA for an arbitrary problem is to try equal proportions of three pairwise operators: line interpolation, line extrapolation and exchange, with a small mutation rate and range (from 0.01 to 0.2), and extrapolation range 0.5.

It is clear though, that much research is still needed in order to understand the large spectrum of possible operators and their efficient application. Obvious improvement can be achieved by including adaptive schemes to set the application rate parameters, time dependent operators (like the dynamic mutation in [8]) and smart schedules for the parameters which may help in improving the search efficiency and robustness.

References

[1] Antonisse, J. (1989) A new interpretation of schema notation that overturns the binary encoding constraint, *3rd International Conference on Genetic Algorithms*, 86-91, Morgan Kaufmann.

[2] Davidor, Y. (1991) *Genetic algorithms and robotics*, World Scientific.

[3] Davidor, Y. (in press) *Free the spirit of evolutionary computation*, in Ray Paton (Ed.), *Bio-computing*, Chapman & Hall.

[4] Davis, D. L. (Ed.) (1991) *The handbook of genetic algorithms*, Morgan Kaufmann.

[5] DeJong, K. A. and Spears, W. M. (1991) An analysis of the interacting roles of population size and crossover in genetic algorithms, In Hans-Paul Schwefel and Reinhard Manner (Eds.), *Parallel Problem Solving from Nature*, Springer-Verlag.

[6] Eshelman, L.J., and Schaffer, J.D. (1992) Real-coded genetic algorithms and interval-schemata, *Foundations of Genetic Algorithms II* Whitley D. (Ed.), Morgan Kaufmann.

[7] Goldberg, D. E. (1990) The theory of virtual alphabets, In Hans-Paul Schwefel and Reinhard Manner (Eds.), *Parallel Problem Solving from Nature*, 13-22, Springer-Verlag.

[8] Janikow, C. Z. and Michalewicz, Z. (1991) An experimental comparison of binary and floating point representations in genetic algorithms, *Proceedings of the 4th International Conference on Genetic Algorithms*, 31-36, Morgan Kaufmann.

[9] Lucasius, C. B. and Kateman, G. (1989) Application of genetic algorithms in chemometrics, *proceedings of the 3rd International Conference on Genetic Algorithms*, 170-176, Morgan Kaufmann.

[10] Michalewicz Z. (1992) *Genetic Algorithms+ Data Structures = Evolution Programs*, Springer - Verlag.

[11] Michalewicz Z. and Janikow C. (1992) GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints, Communications of the ACM.

[12] Michalewicz Z., Jankowski A. and Vigneux G.A. (1990) The Constraints Problem in Genetic Algorithms, in *Methodologies of Intelligent Systems: Selected Pepers*, Emrich M.L., Phifer M.S., Huber B., Zemankova M. and Ras Z. (Eds.), ICAIT, Knoxville TN, pp.142-157.

[13] Radcliffe, N.J., (1990) Genetic neural networks on MIMD computers, Ph.D. Dissertation, Dept. of Theoretical Physics, University of Edinburgh, Edinburgh, UK.

[14] Wright, A. H. (1990) Genetic algorithms for real parameter optimization, In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms*, 205-220, Morgan Kaufmann.

p_{int}	p_{ext}	p_{exch}	Performance
0.333	0.333	0.333	16181.9
0.2	0.4	0.4	16235.4
0.4	0.2	0.4	16208.7
0.4	0.4	0.2	16182.7
0.5	0.25	0.25	16185.1
0.25	0.5	0.25	16479.5
0.25	0.25	0.5	16189.8
0.5	0.5	0.0	16720.6
0.5	0.0	0.5	30097.7
0.0	0.5	0.5	551892.8
0.3	0.3	0.3	16186.4
0.1	0.4	0.4	41345.8
0.4	0.1	0.4	16836.3
0.4	0.4	0.1	16182.3
0.4	0.4	0.0	18014.6
0.3	0.3	0.2	16182.6

Table 2: The control problem - operator rates comparison.

Extrapolation	Interpolation		
	point	line	rectang.
point	524 (207)	399 (168)	467 (213)
line	970 (407)	648 (226)	790 (323)
rectang.	792 (261)	836 (300)	772 (211)

Table 3: f3 test function - operator choice comparison. in parentheses, the deviation of the performance.

p_{int}	p_{ext}	p_{exch}	Performance
0.333	0.333	0.333	399 (168)
0.2	0.4	0.4	365 (125)
0.4	0.2	0.4	614 (203)
0.4	0.4	0.2	408 (201)
0.5	0.5	0.0	496 (309)
0.5	0.0	0.5	1403 (603)
0.0	0.5	0.5	318 (111)

Table 4: f3 test function - operator rates comparison.