The Virtual Gene Genetic Algorithm

Manuel Valenzuela-Rendón

ITESM, Monterrey
Centro de Sistemas Inteligentes
C.P. 64849, Monterrey, N.L., México
valenzuela@itesm.mx
http://www-csi.mty.itesm.mx/~mvalenzu

Abstract. This paper presents the *virtual gene genetic algorithm* (vgGA) which is a generalization of traditional genetic algorithms that use binary linear chromosomes. In the vgGA, traditional one point crossover and mutation are implemented as arithmetic functions over the integers or reals that the chromosome represents. This implementation allows the generalization to virtual chromosomes of alphabets of any cardinality. Also, the sites where crossover and mutation fall can be generalized in the vgGA to values that do not necessarily correspond to positions between bits or digits of another base, thus implementing *generalized digits.* Preliminary results that indicate that the vgGA outperforms a GA with binary linear chromosomes on integer and real valued problems where the underlying structure is not binary are presented.

1 Introduction

Traditional genetic algorithms evolve populations of individuals that are represented by linear chromosomes defined in a small cardinality alphabet, usually binary [1,2]. Traditional crossover and mutation create new individuals by manipulating these bit strings. This paper shows that traditional one point crossover and mutation can be simulated as arithmetic functions over the integers represented by the binary chromosomes. In this way, a genetic algorithm for integer individuals where traditional operation are performed, not over the genotype, but rather simulated over the phenotype, can be implemented.

Binary chromosomes can be used to represent real values in many ways; one of the simplest is the use of a linear mapping [1, page 82]. If a linear mapping is used, traditional one point crossover and mutation can also be simulated as arithmetic functions over the reals represented by the binary chromosomes. Even though there is a large body of work which explores the use of real valued individuals in an evolutionary algorithm (see for example [3,4,5,6]), most of this work is oriented at creating new operators with effects that cannot be easily seen as manipulations of the bit representations of the individuals, and therefore is not directly related to the work here presented.

The basic idea of the paper is then generalized. Points where crossover or mutation can occur can be visualized not only as bit positions (or digit positions

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1457–1468, 2003.

in the case of non-binary chromosomes) but rather as the value these bits or digits represent, and therefore crossover and mutation can occur at generalized values, what we will call, at *generalized digits.*

In the rest of this paper, we will see the mathematical basis of the *virtual gene genetic algorithm* (vgGA), an algorithm that implements crossover and mutation as arithmetic functions of the phenotype of the individuals, and preliminary experiments that show that the vgGA can outperform a traditional genetic algorithm in problems with an underlying that is not binary.

2 Traditional Crossover and Mutation

Let p be an integer represented by a binary chromosome of length N, where the string of all zeros, $000\cdots0_2$, represents the integer zero, and the string of all ones, $111\cdots1_2$, represents the integer $2^N - 1$. The *lower part* of p below and including bit m can be obtained as

$$L_m(p) = p \bmod 2^m, \tag{1}$$

where

$$x \bmod y = \begin{cases} x - y\lfloor x/y \rfloor, & \text{if } y \neq 0; \\ x, & \text{if } y = 0; \end{cases} \tag{2}$$

The *higher part* of p above bit m can be obtained as

$$H_m(p) = p - L_m(p) = p - p \bmod 2^m = 2^m \lfloor p/2^m \rfloor. \tag{3}$$

By substituting an arbitrary base B for 2, the above formulas can be generalized to chromosomes in an alphabet of cardinality B.

$$L_m(p) = p \bmod B^m. \tag{4}$$

$$H_m(p) = p - p \bmod B^m = B^m \lfloor p/B^m \rfloor. \tag{5}$$

Using the lower and higher parts of an integer individual, it is possible to extract parts of a chromosome. The value represented by the digits of the higher part, which we will call the *higher part value* of p above bit m can be obtained as

$$\hat{H}_m(p) = \frac{H_m(p)}{B^m} = \lfloor p/B^m \rfloor. \tag{6}$$

The m-th digit (where the least significant digit is numbered 1, and the most significant digit is numbered N) can be obtained as

$$\text{digit}_m(p) = L_1(\hat{H}_{m-1}(p)) = \hat{H}_{m-1}(L_m(p)). \tag{7}$$

The *segment* of digits $m_1 + 1, m_1 + 2, \ldots, m_2$ can be obtained as

$$\text{segment}_{m_1,m_2}(p) = L_{\Delta m}(\hat{H}_{m_1}(p)), \tag{8}$$

where $\Delta m = m_2 - m_1$.

With the definitions of lower part and higher part, we can now express the crossover of two chromosomes over an alphabet of cardinality B as an arithmetic operation over the integers these chromosomes represent. Let p_1 and p_2 be two integers over base B, one point crossover produces two offspring h_1 and h_2 that can be expressed in the following way:

$$h_1 = \text{crossover}_m(p_1, p_2) = L_m(p_1) + H_m(p_2); \tag{9}$$

$$h_2 = \text{crossover}_m(p_2, p_1) = L_m(p_2) + H_m(p_1). \tag{10}$$

Therefore, one point crossover is simply the exchange between two integers of their lower and higher parts at a given crossover point. A simplified expression for crossover can be obtained by substituting the expressions for lower and higher part, obtaining the following:

$$h_1 = \text{crossover}_m(p_1, p_2) = p_2 + \chi_m(p_1, p_2); \tag{11}$$

$$h_2 = \text{crossover}_m(p_2, p_1) = p_1 - \chi_m(p_1, p_2); \tag{12}$$

where $\chi_m(p_1, p_2) = p_1 \bmod B^m - p_2 \bmod B^m = -\chi_m(p_2, p_1)$.

In traditional mutation for binary chromosomes, mutation of a bit is the same as complementing its value, in other words, flipping the bit from 1 to 0, or from 0 to 1. For alphabets of higher cardinality, the most natural definition of mutation of a digit is to replace it with a random value that is *not* the original value in that position. To facilitate its application when non-binary chromosomes are used, we define mutation in a slightly different manner. We will define mutation as the operation that given an integer p, removes a segment of consecutive digits, and replaces it with a random segment of the same number of digits. The mutation of the segment of digits $m_1 + 1, m_1 + 2, \ldots, m_2$ of an integer p can be expressed as

$$\text{mutation}_{m_1,m_2}(p) = L_{m_1}(p) + H_{m_2}(p) + B^{m_1} \left\lfloor B^{\Delta m} \text{rand}() \right\rfloor, \tag{13}$$

where $\Delta m = m_2 - m_1$ and rand() is a function that generates a random number in $[0, 1)$ with uniform distribution.

3 Generalized Crossover and Mutation

The concepts of lower part and higher part presented above were defined in terms of the m-th digit. The formulas include the term B^m, which is the *weight* of bit $m + 1$. A generalization of these formulas can be produced by substituting B^m with n, an integer that is not necessarily an integer power of B. Let us define *generalized lower part* and *generalized higher part* as follows:

$$L(p, n) = p \bmod n; \tag{14}$$

$$H(p, n) = p - p \bmod n = n \lfloor p/n \rfloor. \tag{15}$$

Notice that $L(p,n)$ and $H(p,n)$ refer to the generalized lower and higher parts, and that $L_m(p)$ and $H_m(p)$ refer to the lower and higher parts.

We can also find an expression for the *generalized higher part value* in the following way:

$$\hat{H}(p,n) = \frac{H(p,n)}{n} = \lfloor p/n \rfloor. \tag{16}$$

Note what n means: digit m has a weight of B^{m-1} in the value of p, i.e., if digit m has a value of d_m, it will contribute with $d_m B^{m-1}$ to the value of p. We will call *generalized digit* n of base B what is obtained when the following operation is performed:

$$\text{digit}(p,n,B) = L(\hat{H}(p,n/B),B). \tag{17}$$

This generalized digit has a weight of n/B in the value of p. To avoid the use of traditional digits, we define *generalized segment* in terms of an initial value and a segment width. The *generalized segment* of width δ starting at value n is given by the following expression:

$$\text{segment}(p,n,\delta) = L(\hat{H}(p,n),\delta), \tag{18}$$

where δ is an integer greater or equal than B. These definitions modify the meaning of parts of a chromosome to the point where it is more useful to think about chromosomes, not as strings of characters, but rather as integer values.

We can now express crossover and mutation in terms of the generalized operations defined above. The *generalized crossover* of integers p_1 and p_2 at value n results in two offspring that can be expressed as

$$h_1 = \text{crossover}(p_1,p_2,n) = L(p_1,n) + H(p_2,n); \tag{19}$$

$$h_2 = \text{crossover}(p_2,p_1,n) = L(p_2,n) + H(p_1,n). \tag{20}$$

This can also be written as the following:

$$h_1 = \text{crossover}(p_1,p_2,n) = p_2 + \chi(p_1,p_2,n); \tag{21}$$

$$h_2 = \text{crossover}(p_2,p_1,n) = p_1 - \chi(p_1,p_2,n); \tag{22}$$

where $\chi(p_1,p_2,n) = p_1 \bmod n - p_2 \bmod n = -\chi(p_2,p_1,n)$.

The *generalized mutation* for a segment of width δ starting at value n is defined as the following:

$$\text{mutation}(p,n,\delta) = L(p,n) + H(p,n\delta) + n\lfloor \delta\,\text{rand}() \rfloor. \tag{23}$$

It can be shown that traditional operators are a special case of generalized operators by substituting B^m for n. For crossover and mutation we have that

$$\text{crossover}_m(p_1,p_2) = \text{crossover}(p_1,p_2,B^m); \tag{24}$$

$$\chi_m(p_1,p_2) = \chi(p_1,p_2,B^m); \tag{25}$$

$$\text{mutation}_{m_1,m_2}(p) = \text{mutation}(p,B^m,B^{\Delta m+1}). \tag{26}$$

In the rest of this paper we will only be using the generalized expressions and therefore we will drop the word generalized.

4 Real Valued Individuals

We now proceed to adapt the formulas developed before for real valued individuals. Let r be a real number in the interval $[r_{\min}, r_{\max})$. Using a linear mapping r can be represented by a chromosome of N digits. We can see this chromosome as an integer p that can take a value in the set $\{0, 1, 2, 3, \ldots, B^N - 1\}$. The transformation between p and r is given by the following formula:

$$r = p\Delta\overline{r} + r_{\min}, \tag{27}$$

where $\Delta\overline{r} = (r_{\max} - r_{\min})/B^N$. We define *lower part, higher part,* and *segment* of a real individual r in terms of those same operations over the corresponding integer p. In this way, the lower part of r is given by

$$L(r, k, r_{\min}) = L(p, n)\Delta\overline{r} + r_{\min}; \tag{28}$$

$$= (r - r_{\min}) \bmod (k - r_{\min}) + r_{\min}. \tag{29}$$

The higher part of a real number is given by

$$H(r, k, r_{\min}) = H(p, n)\Delta\overline{r} + r_{\min}; \tag{30}$$

$$= r - (r - r_{\min}) \bmod (k - r_{\min}). \tag{31}$$

We define the crossover of two real valued individuals r_1 and r_2 as

$$h_1 = \text{crossover}(r_1, r_2, k, r_{\min}) = (L(p_1, n) + H(p_2, n))\,\Delta\overline{r} + r_{\min}; \tag{32}$$

$$h_2 = \text{crossover}(r_1, r_2, k, r_{\min}) = (L(p_2, n) + H(p_1, n))\,\Delta\overline{r} + r_{\min}; \tag{33}$$

where h_1 and h_2 are the offspring produced. Simplifying, the above can also be written in the following way:

$$h_1 = \text{crossover}(r_1, r_2, k, r_{\min}) = r_2 + \chi(r_1, r_2, k, r_{\min}); \tag{34}$$

$$h_2 = \text{crossover}(r_2, r_1, k, r_{\min}) = r_1 - \chi(r_1, r_2, k, r_{\min}); \tag{35}$$

where

$$\chi(r_1, r_2, k, r_{\min})$$

$$= (r_1 - r_{\min}) \bmod (k - r_{\min}) - (r_2 - r_{\min}) \bmod (k - r_{\min}) \tag{36}$$

$$= -\chi(r_2, r_1, k, r_{\min}). \tag{37}$$

The mutation of a real valued individual r, at value k, with a mutation width of δ is given as

$$\text{mutation}(r, k, \delta, r_{\min}) = (\text{mutation}(p, n, \delta))\,\Delta\overline{r} + r_{\min}. \tag{38}$$

Simplifying we arrive at the following:

$$\text{mutation}(r, k, \delta, r_{\min}) = L(r, k, r_{\min}) + H(r, \delta[k - r_{\min}] + r_{\min}, r_{\min}) + (k - r_{\min})\lfloor \delta\,\text{rand}() \rfloor - r_{\min}. \tag{39}$$

We can treat integer individuals as a special case of real valued individuals. and thus, the formulas presented above can be also applied to integers.

Not all values of δ produce valid results. If we want mutation to produce only individuals in the interval $[r_{\min}, r_{\max})$, the following condition must be met:

$$L(r, k, r_{\min}) + H(r, (k - r_{\min})\delta + r_{\min}, r_{\min}) \leq r + r_{\min}, \tag{40}$$

for $\delta > 1$. Substituting the expression for lower part and higher part, and simplifying the following is arrived at:

$$\delta \left\lfloor \frac{r - r_{\min}}{(k - r_{\min})\delta} \right\rfloor \leq \left\lfloor \frac{r - r_{\min}}{k - r_{\min}} \right\rfloor. \tag{41}$$

This inequality is satisfied in the general case only if δ is an integer.

5 Generating Crossover Points

In the formulas developed to perform crossover, n for integers and k for reals is a random number with a given probability function. In a traditional genetic algorithm, crossover falls between traditional digits, i.e., at integer powers of B. Crossover sites that have this distribution can be produced as $n = B^{\lfloor N \,\mathrm{rand}() \rfloor}$. A probability function that has the same form but uses generalized digits can be obtained if the crossover sites are generated by $n = \left\lfloor B^{N \,\mathrm{rand}()} \right\rfloor$.

For real valued individuals, we can find similar formulas to those developed for integers, but additionally, we have the option of having continuous distributions by dropping the use of the floor function. Table 1 summarizes possible ways to generate crossover sites. Figures 1 and 2 show the cumulative distribution function for the crossover point distributions for integer and real valued individuals mentioned above.

If traditional digits are being used, crossover cannot produce invalid results, but for generalized digits it is possible that the result is greater than $r_{\max}$. Given that for integers the sum of the offspring is equal to the sum of the parents, $p_1 + p_2 = h_1 + h_2$, we know that a condition that insures that crossover will

Table 1. Different ways to produce crossover sites

crossover points distribution	(integers) n	(reals) k
traditional	$B^{\lfloor N \,\mathrm{rand}() \rfloor}$	$B^{\lfloor N \,\mathrm{rand}() \rfloor} \Delta\overline{r} + r_{\min}$
generalized	$\left\lfloor B^{N \,\mathrm{rand}()} \right\rfloor$	$\left\lfloor B^{N \,\mathrm{rand}()} \right\rfloor \Delta\overline{r} + r_{\min}$
continuous		$B^{N \,\mathrm{rand}()} \Delta\overline{r} + r_{\min}$

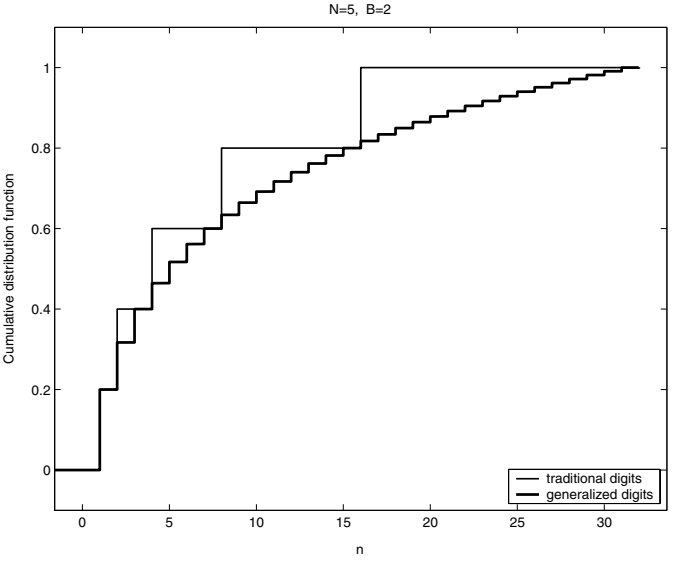

Fig. 1. Cumulative distribution functions for crossover sites for a binary chromosome of length 5 representing an integer

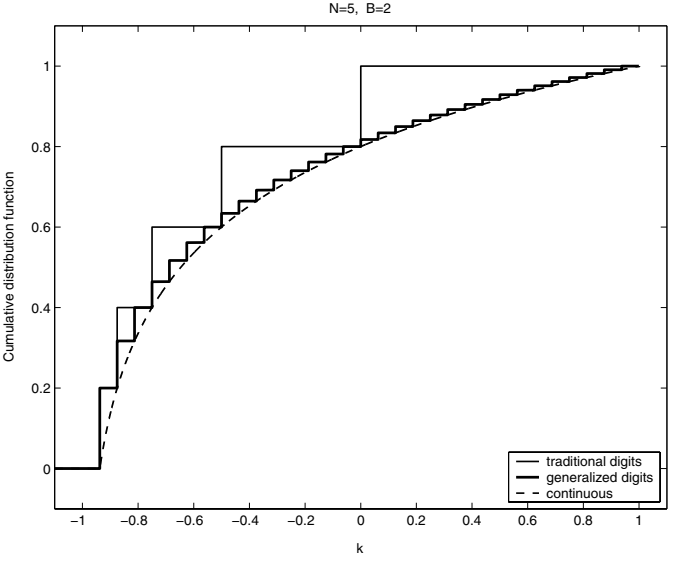

Fig. 2. Cumulative distribution functions for crossover sites for a chromosome of base 3 and length 5 that represents a real individual in $[-1, -1)$

produce valid individuals for any crossover site is the following:

$$p_1 + p_2 < B^N. \tag{42}$$

If the above condition is not met, we apply crossover with traditional digits to insure that the result will be valid. We call this the *crossover overflow correction.* The condition for reals is as follows:

$$r_1 + r_2 - r_{\min} < r_{\max}. \tag{43}$$

6 Generating Mutation Sites

It is known that the implementation efficiency of a binary genetic algorithm can be increased if mutation is controlled by means of a *mutation clock* [7]. According to the mutation clock idea, instead of generating a random number for each bit in the population to decided if it should be mutated, a random number with the proper probability distribution is generated so that it tells in how many bits the next mutation should occur. A mutation clock for the vgGA for traditional and generalized digits, and for integer and real valued individuals, was developed.

As in crossover, mutation at generalized digits could produce invalid results. Mutation removes a segment of the individual and substitutes it with a random segment in the set of all possible values. If we call γ the maximum value the segment can have so that the result is still valid, then the following equation expresses the relation between an integer p and its γ:

$$L(p, n) + H(p, \delta n) + n\gamma = B^N. \tag{44}$$

From the above, we can obtain γ as

$$\gamma = \frac{B^N - L(p, n) - H(p, n\delta)}{n}. \tag{45}$$

Now, we define mutation for integers with the *gamma correction* as

$$\text{mutation}(p, n, \delta) = L(p, n) + H(p, n\delta) + n\lfloor \gamma \,\text{rand}() \rfloor. \tag{46}$$

For real valued individuals the value of γ is given by

$$\gamma = \frac{r_{\max} + r_{\min} - L(r, k, r_{\min}) - H(r, [k - r_{\min}]\delta + r_{\min}, r_{\min})}{k - r_{\min}}. \tag{47}$$

Mutation of reals with the *gamma correction* is defined as the following:

$$\begin{aligned} \text{mutation}(k, r, \delta, r_{\min}) &= L(r, k, r_{\min}) \\ &+ H(r, [k - r_{\min}]\delta + r_{\min}, r_{\min}) + (k - r_{\min})\lfloor \gamma \,\text{rand}() \rfloor - r_{\min}. \end{aligned} \tag{48}$$

7 Experiments

One could expect the vgGA that implements generalized digits to outperform a traditional GA on problems where the underlying representation is not binary. To test this assumption, a generalization of the well known one-max problem was defined. In the consecutive one-max problem, or *c-one-max problem,* the evaluation depends on the lengths groups of consecutive digits that are equal to one when the individual is expressed in given base. Each group contributes to the fitness function with its length to a power α. For example, for $\alpha = 2$, an individual with phenotype of 412281_{10}, which can be expressed as 101143111_5, has an evaluation $1^2 + 2^2 + 3^2 = 14$ in the c-one-max problem of base 5 as shown in Fig. 3. For binary GAs, the c-one-max problem in any base that is not a multiple of 2 should be a relatively hard problem (at least harder than the problem in base 2). On the other hand, since the vgGA is not tied to a given base, its performance on this problem should be higher.

A vgGA, where individuals are vectors of integers or reals, was implemented in MATLAB, and tested with the c-one-max problem of base 2 and base 5 (this is the base of the problem and not of the individuals in the vgGA) and $\alpha = 2$. Table 2 summarizes the parameters of the vgGA used for these tests. Binary tournament selection [8] was used. Figures 4 and 5 shows the results for the c-one-max problem of base 2 and base 5, respectively. These plots are the average of the best-found-so-far of 100 runs. For the base 2 problem, traditional digits, i.e. a traditional genetic algorithm, outperform generalized digits. For the base 5 problem the results are the opposite, as expected.

A real valued version of the c-one-max problem can be obtained if the evaluation depends on the number of digits that are equal to those of a given irrational constant, expressed on a given base. The *c-pi-max problem* will be defined as the problem of finding the digits of π where the evaluation depends on the number

Table 2. Parameters of the vgGA used in all experiments

runs	100
generations	150
population size	20
N	40
p_c	1.0
p_m	0.1
B	2
δ	2

$$\underbrace{1}_{1^2} \; 0 \; \underbrace{1 \; 1}_{2^2} \; 4 \; 3 \; \underbrace{1 \; 1 \; 1}_{3^2}$$

Fig. 3. Example of c-one-max base 5 problem. The evaluation of $412281_{10} = 101143111_5$ is $1^2 + 2^2 + 3^2 = 14$

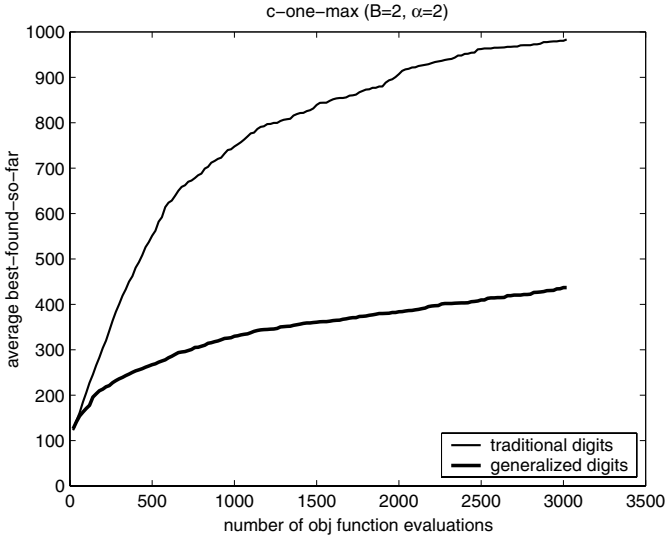

Fig. 4. Average of 100 runs of the best-found-so-far for the c-one-max problem of base 2 with $\alpha = 2$

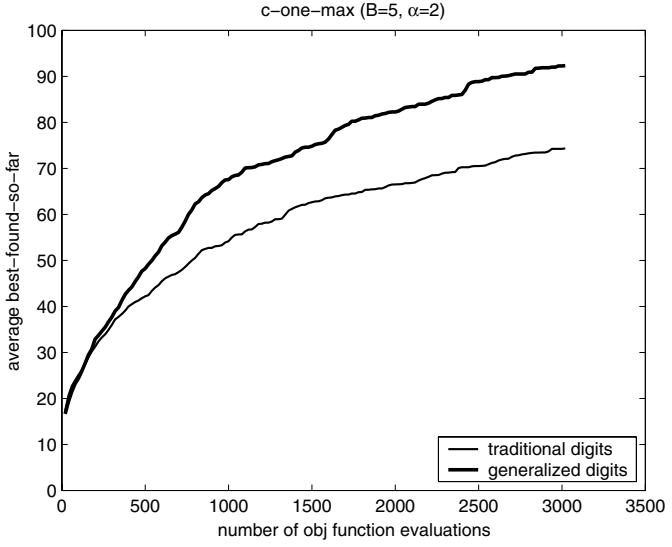

Fig. 5. Average of 100 runs of the best-found-so-far for the c-one-max problem of base 5 with $\alpha = 2$

$$\underbrace{0\ 3}_{2^2}\ .\ 0\ \underbrace{1\ 5}_{2^2}\ 0\ \underbrace{9\ 2\ 6}_{3^2}\ 0$$

Fig. 6. Example of c-pi-max base 10 problem. The evaluation of 03.01509260_{10} is $2^2 + 2^2 + 3^2 = 17$

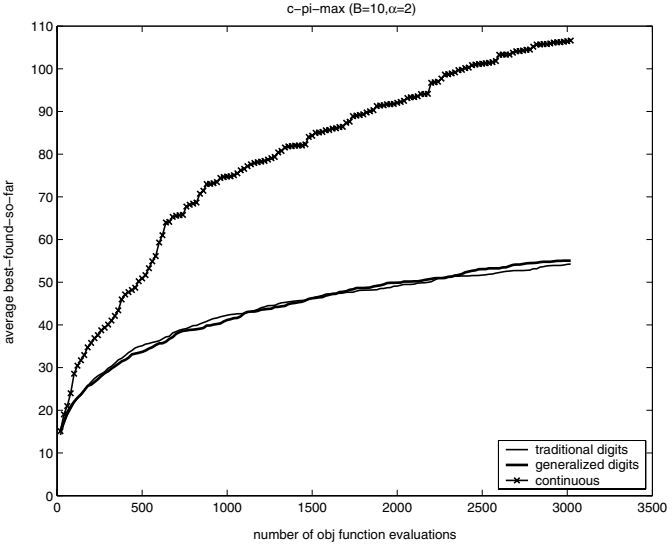

Fig. 7. Average of 100 runs of the best-found-so-far for the c-pi problem of base 5

of consecutive digits on a given base as described for the c-one-max problem. For example, an individual with phenotype of 03.01509260_{10} has an evaluation of $2^2 + 2^2 + 3^2 = 17$ for the c-pi-max problem with $\alpha = 2$, base 10, and considering two digits to the left and eight digits to the right of the decimal point as shown in Fig. 6. Figure 7 shows the results for the c-pi-max problem of base 10 with 2 integer positions and 49 decimal positions. The vgGA implements real valued individual in $[0, 5)$. As the figure shows, the vgGA that uses a continuous distribution of crossover points and mutation sites has the best performance on this problem.

8 Conclusions

This paper shows that traditional one point crossover and mutation can be mapped to arithmetic operations over integers, the formulas found can be generalized to chromosomes of any base, and also, to real valued individuals rep-

resented by a linear mapping. A *virtual gene genetic algorithm* (vgGA) which works on the phenotype and can produce the same results as a traditional binary GA, has been implemented in MATLAB.

The vgGA is a generalization of a traditional GA with binary linear chromosomes. It is a generalization because by mapping traditional crossover and mutation to operations over the phenotype, it can simulate linear chromosomes of any integer base, not necessarily binary. Additionally, the vgGA extends where crossover and mutation sites may fall, allowing the simulation of generalized digits.

The sites where crossover and mutation fall in a traditional GA can be generalized to values that do not correspond to integer powers of a given base, thus implementing *generalized digits.* Preliminary results indicate that a vgGA using generalized digits can outperform a traditional binary GA on an integer problem where the underlying structure does not depend on a binary representation. When solving a real valued problem, the vgGA can implement a continuous distribution of crossover and mutation sites. An experiment where this continuous distribution produces better results than the traditional discrete distribution was presented. The experiments presented in this paper are admittedly very limited and should be extended to other problems.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA (1989)
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975)
3. Wright, A.H.: Genetic algorithms for real parameter optimization. In Rawlins, G.J.E., ed.: Foundations of Genetic Algorithms. Morgan Kaufmann, San Mateo, CA (1991) 205–218
4. Eschelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In Whitley, L.D., ed.: Foundations of Genetic Algorithms 2. Morgan Kaufmann, San Mateo, CA (1993) 187–201
5. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 2nd. edn. Springer-Verlag, Berlin (1994)
6. Surry, P.D., Radcliffe, N.J.: Real representations. In Belew, R.K., Vose, M.D., eds.: Foundations of Genetic Algorithms 4. Morgan Kaufmann, San Mateo, CA (1997) 187–201
7. Golbderg, D.E.: Personal communication. (1990)
8. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G.J.E., ed.: Foundations of Genetic Algorithms. Morgan Kaufmann, San Mateo, CA (1991) 69–93