siGis-26-2003

February 25, 2004

Multi–Chromosomal Representations and Chromosome Shuffling in Evolutionary Algorithms

2003 Congress on Evolutionary Computation
December 8–12, 2003, Canberra A.C.T., Australia

Helmut A. Mayer
helmut@cosy.sbg.ac.at

Markus Spitzlinger
mspitzl@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg

Correspondence to:

Helmut Mayer
Universität Salzburg
Institut für Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
AUSTRIA

Telephone: +43-662-8044-6315
FAX: +43-662-8044-611

Multi–Chromosomal Representations and Chromosome Shuffling in Evolutionary Algorithms

Helmut A. Mayer
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
helmut@cosy.sbg.ac.at

Markus Spitzlinger
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
mspitzl@cosy.sbg.ac.at

Abstract- We present experiments investigating the use of multi–chromosomal representations in evolutionary algorithms. Specifically, the conventional representation of parameters on a single chromosome is compared to a genotype encoding with multiple chromosomes on a set of test functions. In this context we present chromosome shuffling, a genetic operator recombining complete chromosomes based on biological evidence. The hypothesis that the multi–chromosomal representation ameliorates the transmission of good sub–solutions to the population is tested on functions of varying degree of complexity.

1 Introduction

In this paper we investigate the potential benefits of multi–chromosomal representations for genotypes processed by an *Evolutionary Algorithm* (EA) (Fogel, 1995; Schwefel, 1995). With the growing complexity of EA applications the search spaces sampled by the solutions encoded in a genotype become increasingly large. A human measure to counteract the problem of growing complexity is the strategy of *Divide–and–Conquer*, which arguably already plays an important role in EAs, as promoted by the *Building Block Hypothesis* (BBH) (Goldberg, 1989). However, with increasing complexity the building blocks identified by the BBH may be too small, calling for an additional layer of building blocks encoding complete sub–solutions of a problem. We could argue that nature has evolved such a layer by packing genes (small building blocks) into much larger chromosomes (big building blocks). Evidently, this is a gross oversimplification as it is known that cooperating wild–type genes can be found on different chromosomes in nature (Lodish et al., 1995).

The biological motivation for the use of multiple chromosomes comes from *Meiosis*, a complicated cell division process involving sexual reproduction. A maternal and a paternal set of chromosomes (humans have 23 arranged in *Diploid Sets*, i.e., each chromosome occurs in two homologous variants) is combined into one cell. Homologous chromosomes of father and mother align in a phase called *Synapsis*, and parts of the genetic code can be exchanged by crossing over at certain sites. Usually, 1 to 8 crossover points can be identified on one chromosome (Lodish et al., 1995). This process is the model for the recombination operator in EAs; however, a very interesting step occurs after crossover. The two recombined chromosomes (actually, four, because of diploidy) separate randomly to different areas of the cell. Thus, there is an additional shuffling of genetic material at the level of chromosomes. In humans, this process allows for $2^{23} \approx 8.4 \times 10^6$ different chromosome combinations.

In the realm of artificial evolution we term this process *Chromosome Shuffling* and model it simply by exchanging the chromosomes of the two parents with a certain probability. As is the case in nature, only homologous chromosomes (describing the same variables of the solution with identical representation) are shuffled, which allows the co–existence of different chromosomes within an individual. E.g., an artificial neural network could be encoded by a bit chromosome (connections between neurons), an integer chromosome (neuron parameters, e.g., type of activation function), and a real chromosome (connection weights). However, in this work we are concerned with multiple chromosomes having identical structure (real values encoded as bitstrings).

From above statements potential benefits of multi–chromosomal representations can be identified. First, a complex problem can be decomposed into sub–problems being encoded in corresponding chromosomes with specific representations. Complete solutions of a sub–problem can be exchanged by means of chromosome shuffling without the disruptive effects of crossover. Second, each chromosome can be encoded using a representation adapted to the specific sub–problem. Third, chromosome shuffling could induce improved robustness of solutions, as a “specialist” chromosome that contributes to good solutions in a few individuals only will be quickly weeded out by evolution. A more general solution of the sub–problem being successful in a great variety of individuals will have increased chances of survival.

1.1 Related Work

Pierrot and Hinterding (1997) presented an investigation of the use of multi–chromosomes to solve an allocation problem by means of an EA. 500 goods are to be produced on three machines, where each machine has specific fixed and variable costs. The fixed cost is incurred only, if the machine is utilized for the production of goods. The variable costs are defined per good produced on a specific machine. The main idea is to specify the usage of machines on one chromosome, while the variables on the second chromosome encode the number of goods to be produced on the corresponding machine. It has been found that a multi–

chromosomal representation has the potential to improve solutions, but it should be noted that no attempt has been made to adjust mutation and crossover rates in the experiments with single and multiple chromosomes. As a consequence, the different mutation and crossover rates have been identified as main source of the improvements of the solution encoded on multiple chromosomes (Pierrot and Hinterding, 1997).

Ronald and colleagues (1997) performed experiments with differently encoded chromosomes in a genotype (*Mixed Encoding*). The authors modified the *Tavelling Sales Person* (TSP) problem by a number of modes of transport with different costs available in each city. Change of the transport mode is associated with a penalty (additional cost). While the tour of cities is encoded by a conventional permutation chromosome, the transport modes are described by an integer chromosome. The best tour found in the modified TSP problem differed from the best solution for the unmodified problem, but unfortunately, in (Ronald et al., 1997) no comparisons were made to assess the quality of the solution generated by the multi–chromosomal representation.

An important development in EAs is the evolution of the genetic representation (Wu, 1996; Mayer, 1998). Clearly, mechanisms evolving the location and linkage of building blocks (*Linkage Learning*) (Kargupta and Park, 2001) could be adapted to additionally evolve the number of chromosomes. However, in this work we a priori define the number of chromsomes and the genetic encoding on the specific chromosomes, the latter being common practice employing single chromosome genotypes. Nevertheless, chromosome evolution including the location of genes on the chromsomes is an interesting area of research, especially, when considering the non–linkage of cooperating genes in nature as referred to above.

2 Chromosome Shuffling

If we want to model the biological multi–chromosomal recombination process in an EA, the algorithm has to be enriched by an additional genetic operator, which we would like to call *Chromosome Shuffling*. The details of the recombination of multi–chromosomal genotypes based on crossover and shuffling are sketched in Figure 1.

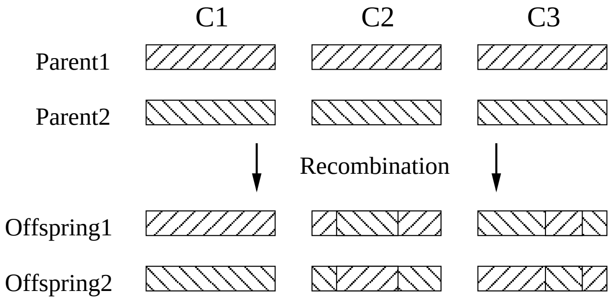

Figure 1: Recombination of two parents with three chromosomes involving 2-point crossover and chromosome shuffling.

In this example the genotype of each of the two parents is composed of three chromosomes. The conventional crossover operation (2–point crossover in Figure 1) can be used with the restriction that it operates only on single chromosomes. As usual the occurrence of crossover is governed by the crossover probability p_c. The shuffling of chromosomes is simply achieved by exchanging homologous chromosomes of the parents. For this process we introduce the *Shuffle Rate* (or shuffling probability) p_s with which each parental chromosome pair is exchanged. In Figure 1 chromosome $C1$ is directly passed from the parents to the offspring, $C2$ undergoes crossover only, and $C3$ experiences both, crossover and shuffling. Note that chromosome shuffling is analogous to *Uniform Crossover* exchanging homologous genes (or bases) with a certain probability. Occurring on different levels of genetic representation chromosome shuffling preserves meaningful sub–solutions, while uniform crossover is not aware of building blocks it may exchange (or destroy).

In order to compare multi–chromosomal representations with the standard single chromosome encoding we have to consider that a unique crossover rate for both representations results in different numbers of crossover operations (when employing n–point crossover). In an effort to adjust the crossover rate to different representations we calculate the probability p_{NOR} that an individual is not altered by recombination (crossover and shuffling), i.e., it “survives” the recombination process. In the most general case of multiple chromosomes with shuffling and crossover the survival probability is

$$p_{nor} = [p_s^c + (1 - p_s)^c](1 - p_c)^c, \qquad (1)$$

where p_s is the shuffle rate, p_c is the crossover rate, and c is the number of chromosomes. The first term takes into account the probabilities of shuffling exchanging all or no chromosomes, while the second term describes the probability of crossover not to occur. When comparing two genotypes with a different number of chromosomes, the adjusted parameters are derived by equaling p_{nor} (Equation 1). When fixing c and p_c for both representations, p_s may take on illegal values, hence, in this and similar cases p_c has to be set to different values for the genotypes to be compared.

3 Experimental Setup

For an empirical evaluation of the influence of multi–chromosomal representations on the success of artificial evolution we utilize a set of well–known test functions. The selection of the specific functions has been guided by hypotheses on the benefits (or drawbacks) of utilizing multi–chromosomal representations. The number of chromosomes (always encoded as bitstrings) is commanded by the number of parameters of the respective function. We compare the multi–chromosomal encoding of solutions to the optimization problems with the standard single chromosome representation by measuring the time (number of generations) to reach the optimum.

3.1 The Royal Road Function (F_{RR})

The *Royal Road* functions (F_{RR}) have been designed to study building block interactions in a *Genetic Algorithm* (GA). These functions are characterized as having a pre-defined optimal solution, pre-defined building blocks, and a hierarchical building block structure (Mitchell et al., 1992). In other words F_{RR} have been tailored to the expected needs of a GA, where the short basic (lowest level) building blocks should be easy to discover and could be quickly recombined to higher level building blocks (Table 1).

s_1	= 11111111**;	c_1 = 8
s_2	= ********11111111**;	c_2 = 8
s_3	= ****************11111111**;	c_3 = 8
s_4	= ************************11111111********************************;	c_4 = 8
s_5	= ********************************11111111************************;	c_5 = 8
s_6	= **11111111****************;	c_6 = 8
s_7	= **11111111********;	c_7 = 8
s_8	= **11111111;	c_8 = 8
s_9	= 1111111111111111**;	c_9 = 16
s_{10}	= ****************1111111111111111********************************;	c_{10} = 16
s_{11}	= ********************************1111111111111111****************;	c_{11} = 16
s_{12}	= **1111111111111111;	c_{12} = 16
s_{13}	= 11111111111111111111111111111111********************************;	c_{13} = 32
s_{14}	= ********************************11111111111111111111111111111111;	c_{14} = 32
s_{15}	= 1111111111111111111111111111111111111111111111111111111111111111;	c_{15} = 64

Table 1: A Royal Road Function (from (Mitchell et al., 1992)).

The Royal Road fitness function is given by:

$$F_{RR}(x) = \sum_{s \in S} c_s \sigma_s(x) \quad \sigma_s(x) = \left\{ \begin{array}{cc} 1 & \text{if } x \text{ instance of } s \\ 0 & \text{otherwise,} \end{array} \right. \tag{2}$$

where x is the encoded parameter value, and c_s is a value assigned to the schema s (here c_s = $order(s)$). With above fitness function and a chromosome length of 64 the optimal $F = 256$.

If F_{RR} are tailor–made for a GA with a single chromosome genotype, they should be even more amenable to a multi–chromosomal representation, where each building block s is encoded on a chromosome (yielding eight chromosomes for above function (Table 1)). Specifically, chromosome shuffling should quickly spread the building blocks in the population once they are discovered.

3.2 The Rosenbrock Function (F_2)

Rosenbrock's function (Digalakis and Margaritis, 2002), (Whitley et al., 1995) is a unimodal function with two variables given by

$$F_2 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \tag{3}$$

It has a very narrow ridge, whose tip is sharply pronounced running around a parabola. The parameter values are limited to the interval $-2.048 \leq x_i \leq 2.048$. The function has its minimal value $F_2 = 0.0$ at $x_i = 1.0$.

Here, we expect the nonlinear interaction of the two variables not to favor the multi–chromosomal representation, as shuffling of a single variable (chromosome) contributing to a good solution in an individual does not necessarily improve another solution.

3.3 The Schwefel Function (F_7)

Schwefel's function (Gordon and Whitley, 1993; Whitley et al., 1995) (labeled F_6 in (Digalakis and Margaritis, 2002) is a multimodal function with a large number of local minima given by

$$F_7 = 10V + \sum_{i=1}^{10} (-x_i \sin(\sqrt{\mid x_i \mid})), \; V = 418.982764. \tag{4}$$

A peculiarity of F_7 is the large distance of the second–best to the global minimum making it hard for many algorithms once going into the wrong direction. The value V has to be ore–adapted to the numerical resolution enabled by the encoding. In our experiments $V = 418.982764$, the ten variables are limited to the interval $-512 \leq x_i \leq 511$. Encoding each variable with 10 bases (bits) the optimal value $F_7 = 0.0$ at $x_i = 421$.

Though, F_7 is a complicated function, there are no nonlinear interactions between any two variables. As a consequence, minimization of each variable contributes to improvement of the fitness function, which makes F_7 a promising candidate for chromosome shuffling to increase the EA's convergence rate.

3.4 The Goldstein–Price Function (F_{GP})

The Goldstein–Price function (Desai and Patil, 1996) is a highly nonlinear function of two variables as defined by

$$F_{GP} = \begin{array}{l} (1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) * \\ (30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)) \end{array} \tag{5}$$

The parameter values are limited to the interval $-2.0 \leq x_i \leq 2.0$. the function's minimal value $F_{GP} = 3.0$ at $x_1 = 0.0$ and $x_2 = -1.0$.

We included F_{GP} under the assumption that the strong nonlinear interactions will even make matters worse for the multi–chromosomal representation, as the chances that shuffling can successfully spread single variables should be lower than with F_2.

3.5 EA Parameters

Each variable of the functions F_2, F_7, and F_{GP} is encoded using a chromosome length $l = 10$. The mutation rates have been either taken from the literature, or have been determined by a small number of pre–experiments ($F_{RR}/p_m = 0.005$ (Mitchell et al., 1992), $F_2/p_m = 0.1 = \frac{2}{cl}$, $F_7/p_m = 0.03 = \frac{3}{cl}$, and $F_{GP}/p_m = 0.2 = \frac{4}{cl}$).

Population size = 128 (400 with F_7), Crossover = 2–point, Crossover rate = adjusted, Shuffle rate = adjusted, Selection method = Binary tournament without replacement, Generation gap = 1.0.

The evolutionary process has been stopped, when we found the optimal solution for F_{RR}, or came within a distance of 0.001 to the optimum for the other functions. In case of F_7 we halted evolution after a maximum of 2000 generations.

In the experiments (Section 4) we investigate the following variants of representation and operators: single chromosome with crossover S, multiple chromosomes with crossover M_C, multiple chromosomes with shuffling M_S, and multiple chromosomes with crossover and shuffling M_{CS}.

The crossover and shuffle rates for multi–chromosomal representations have been adjusted based on the survival probability p_{nor} (Equation 1). As often the adjusted crossover and shuffle rates are rather small we ran some more experiments a with standard crossover and shuffle rates, the latter also set to a biologically plausible rate of $p_s = 0.5$.

Each experiment is repeated 200 times using a different random seed for each run generating the start population with equal probabilities for the alleles 0 and 1.

4 Experimental Results

Table 2 shows the number of generations to find the optimum for F_{RR} comparing single and multiple chrosomome representations.

Genotype (p_C, p_S)	Mean	StdDev	Median
$S(0.7)$	**199.63**	127.34	176
$M_C(0.1397, 0.0)$	326.16	151.30	304
$M_{S1}(0.0, 0.1397)$	311.19	180.76	280
$M_{CS1}(0.07, 0.075)$	235.23	144.55	205
$M_{S2}(0.0, 0.5)$	285.01	204.31	227
$M_{CS2}(0.7, 0.5)$	**115.20**	65.06	97

Table 2: Statistical parameters on the number of generations to find the optimum for F_{RR} (averaged on 200 runs).

It can be seen that the best result is achieved by the single chromosome representation S, when compared to the adjusted multi–chromosomal representations. Clearly, recombination relying on crossover or shuffling only is not capable of improving the result of the single chromosome genotype. The local crossover operator certainly helps to find building blocks, but they cannot be transmitted efficiently to the population. On the other hand shuffling on its own could spread building blocks quickly, but mutation without being assisted by crossover needs more time to discover building blocks. The latter is supported by the second block of (unadjusted) experiments, where M_{S2} despite a much larger shuffle rate cannot beat M_{CS1} utilizing crossover and shuffling, however, at probably too small rates. “Standard” values for crossover and shuffling improve the result considerably, which could be expected for F_{RR}. Note also that all results in Table 2 improve the best mean value of 590 generations in (Mitchell et al., 1992). As we could identify only two differences in the algorithms (1–point crossover and proportional selection with sigma scaling in (Mitchell et al., 1992)), we believe that binary tournament selection exerting less selection pressure is the main source of improvment.

The results for F_2 are presented in Table 3.

Genotype (p_C, p_S)	Mean	StdDev	Median
$S_1(0.5)$	**104.11**	102.43	72
$M_C(0.2929, 0.0)$	109.97	106.78	76
$M_S(0.0, 0.5)$	125.82	148.39	78
$M_{CS1}(0.2, 0.125)$	105.40	105.13	73
$M_{CS2}(0.5, 0.5)$	121.83	116.50	86
$S_2(0.7)$	122.75	106.82	97
$S_3(0.5)$	**101.05**	85.31	70

Table 3: Statistical parameters on the number of generations to find the optimum for F_2 (averaged on 200 runs).

As expected the multi–chromosomal representation does not improve the results for this function, however, the adjusted versions generate results very similar to S_1. An exception is M_S supporting the hypothesis that pure shuffling of parameters is inefficient in the presence of *Epistasis*. Intrinsically, crossover applied to a single chromosome representation has shuffling capabilities, however, this is not true for the special case of F_2 (two variables) and 2–point crossover. Thus, we speculated that 1–point crossover (with shuffling capability over two variables) used with S_3 could improve performance, but the small difference is likely to be of statistical nature.

The results for F_7 are presented in Table 4.

Genotype (p_C, p_S)	Mean	StdDev	Median
$S_1(0.7)$	**278.33**	552.64	93
$M_C(0.1134, 0.0)$	1652.41	716.27	2000
$M_{S1}(0.0, 0.1134)$	527.34	759.60	110
$M_{CS1}(0.05, 0.06677)$	808.27	898.85	124
$M_{S2}(0.0, 0.5)$	485.72	744.66	106
$M_{CS2}(0.7, 0.5)$	**288.92**	571.85	98
$M_{CS3}(0.05, 0.1134)$	329.57	607.27	96

Table 4: Statistical parameters on the number of generations to find the optimum for F_7 (averaged on 200 runs).

Again, the single chromosome representation leads the field, but some interesting trends can be observed. The adjusted M_{S1} is performing much better than M_C, which supports our initial hypothesis that F_7 is amenable to shuffling due to the missing interactions between variables. The adjusted values for M_{CS2} might be too small, as suggested by the much better results for M_{CS3} with only a slightly increased shuffle rate. The multi–chromosomal representation with “standard values” comes very close to S_1.

The results for F_{GP} are shown in Table 5.

So far the results have been roughly in accordance with our hypotheses, and when looking at the results for F_2 (Table 3, we would not expect that the even more complex F_{GP} can be optimized efficiently by multi–chromosomal representations. However, the results favor multiple chromosomes in this case exhibiting an interesting trend. The best results are achieved by using crossover or shuffling exclusively. Also, our assumption that 1–point crossover having shuffling capability in the case of two variables (which is not true for 2–point crossover) may improve performance

Genotype (p_C, p_S)	Mean	StdDev	Median
$S_1(0.5)$	5740.64	5897.06	3958
$M_{C1}(0.2929, 0.0)$	**4844.89**	4752.97	3628
$M_S(0.0, 0.5)$	4846.08	4724.53	3408
$M_{CS1}(0.2, 0.125)$	5512.07	5616.94	3530
$M_{CS2}(0.5, 0.5)$	5336.40	5883.66	3626
$S_2(0.7)$	5316.91	5348.06	3436
$S_3(0.5)$	5092.61	4510.33	3729
$M_{C2}(0.5, 0.0)$	**4677.84**	4187.79	3366

Table 5: Statistical parameters on the number of generations to find the optimum for F_{GP} (averaged on 200 runs).

is confirmed by the results for S_3 (1–point crossover). At the moment we are working on collecting more data of individual F_{GP} runs so as to find possible explanations for this behavior.

5 Summary

We have presented experiments with evolutionary optimization of test functions investigating the possible benefits of multi–chromosomal representations in evolutionary algorithms. For recombination of genotypes with multiple chromosomes the additional genetic operator chromosome shuffling being based on biological evidence has been introduced. It has been found that multi–chromosomal representations expectedly perform well when facing problems, where solutions of a problem can be linearly decomposed into partial solutions. Unfortunately, this is not the type of problems that occur in the real world, however, we could also report on encouraging results with a highly nonlinear test function, where the multi–chromosomal representation showed to be superior to standard genotypes with a single chromosome, when comparing the time (number of generations) spent to find the optimum. First results of ongoing work utilizing multi–chromosomal representations for the evolution of fuzzy controllers indicate that the generalization capabilities of the evolved controllers can be improved, a property, which can be directly linked to the survival of "robust" chromosomes. In extension to linkage learning methods further work will certainly include attempts to find mechanisms allowing the evolution of the number of chromosomes, the type of encoding, and the location of parameters encoded on each chromosome.

Bibliography

Desai, R. and Patil, R. (1996). SALO: Combining simulated annealing and local optimization for efficient global optimization.

Digalakis, J. and Margaritis, K. (2002). An experimental study of benchmarking functions for evolutionary algorithms. *International Journal of Computer Mathematics*, 79(4):403–416.

Fogel, D. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, New York.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.

Gordon, V. S. and Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183, San Mateo, CA. Morgan Kaufman.

Kargupta, H. and Park, B.-H. (2001). Gene Expression and Fast Construction of Distributed Evolutionary Representation. *Evolutionary Computation*, 9(1):43–69.

Lodish, H., Baltimore, D., Berk, A., Zipursky, S. L., Matsudaira, P., and Darnell, J. (1995). *Molecular Cell Biology*. Scientific American Books, 3rd edition. ISBN 07167-2380-8.

Mayer, H. A. (1998). ptGAs–Genetic Algorithms Evolving Noncoding Segments by Means of Promoter/Terminator Sequences. *Evolutionary Computation*, 6(4):361–386.

Mitchell, M., Forrest, S., and Holland, J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and GA performance. In Varela, F. J. and Bourgine, P., editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991*, pages 245–254, Paris. A Bradford book, The MIT Press.

Pierrot, H. J. and Hinterding, R. (1997). Using multichromosomes to solve a simple mixed integer problem. In *Australian Joint Conference on Artificial Intelligence*, pages 137–146.

Ronald, S., Kirkby, S., and Eklund, P. (1997). Multi–chromosome Mixed Encodings for Heterogeneous Problems. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 37–42. IEEE Press.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Sixth–Generation Computer Technology Series. Wiley, New York.

Whitley, D., Mathias, K., Rana, S., and Dzubera, J. (1995). Building better test functions. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 239–246, San Francisco, CA. Morgan Kaufmann.

Wu, A. S. (1996). *Non–Coding DNA and Floating Building Blocks for the Genetic Algorithm*. PhD thesis, University of Michigan.