A Case for Codons in Evolutionary Algorithms

Joshua Gilbert and Maggie Eppstein

Dept. of Computer Science, University of Vermont,
Burlington VT, 05405 USA.
`{jgilbert,eppstein}@emba.uvm.edu`

Abstract. A new method is developed for representation and encoding in population-based evolutionary algorithms. The method is inspired by the biological genetic code and utilizes a many-to-one, codon-based, genotype-to-phenotype translation scheme. A genetic algorithm was implemented with this codon-based representation using three different codon translation tables, each with different characteristics. A standard genetic algorithm is compared to the codon-based genetic algorithms on two difficult search problems; a dynamic knapsack problem and a static problem involving many suboptima. Results on these two problems indicate that the codon-based representation may promote rapid adaptation to changing environments and the ability to find global minima in highly non-convex problems.

1 Introduction

The design and implementation of evolutionary algorithms as a means of computational optimization have been inspired by biological evolution through natural selection. It is generally recognized that there are several important aspects that contribute to the performance of a genetic algorithm, including the underlying data representation, method of selection, genetic operators (e.g., mutation and recombination), fitness evaluation, population structure, and the nature of the problem being solved [1]. While the choices of these aspects are not independent of each other, the purpose of this work is to explore a new method for data representation, motivated by the genetic code. Herein we use the term "genotype" to mean the data upon which the genetic operators are applied, and the term "phenotype" to mean the data that is evaluated in the fitness function.

1.1 Genotypic and Phenotypic Data Representation in Evolutionary Algorithms

Many optimization problems are of the form $F(\mathbf{p})=$ *fitness*: $\mathbb{P}^{np} \rightarrow \mathbb{R}$, where we attempt to find the optimal phenotype vector $\mathbf{p}$, of length np, that optimizes the fitness function F, possibly subject to some additional constraints. In discrete search problems, the phenotype alphabet 6 may be the set of integers $\mathbb{I}$, or some subset thereof. In continuous optimization problems, the phenotype alphabet $\mathbb{P}$ may be the set of real numbers $\mathbb{R}$, or some subset thereof. In evolutionary algorithms, the phenotype $\mathbf{p} \in \mathbb{P}^{np}$ may be represented by a genotype $\mathbf{g} \in \mathbb{G}^{ng}$, of length ng, such that

there is an encoding $\gamma(\mathbf{g})=\mathbf{p}$: $\mathbb{G}^{ng} \rightarrow \mathbb{P}^{np}$. For example, in "simple" genetic algorithms the genotype is represented as a binary string, where each ng/np successive bits are translated to the phenotypic alphabet using a standard binary encoding or Gray encoding function [1, 2], although recently there has been a trend towards use of strings of integers or real numbers directly for both genotype and phenotype (i.e., γ is trivially the identity function), as has been the case all along in evolution strategies [1]. Conceptually, there is nothing preventing the implementation of evolution strategies with a nontrivial encoding function γ between genotype and phenotype. Regardless of whether there is a trivial or non-trivial encoding function, the key point here is that the mapping from genotype to phenotype is almost always one-to-one.

There are some interesting exceptions to this generalization, however, and where redundant encodings have been used they have often been shown to offer advantages. In "messy GAs" [3] the adjacencies between alleles in the variable-length genotype are explicitly stored with index numbers; overdetermined strings are disambiguated using a conflict-resolution mechanism, and are thus an implementation of a dynamically changing many-to-one mapping. However, the emphasis in messy GAs, which have been shown to perform well on deceptive problems, has been on their ability to overcome linkage problems via the explicit adjacency representation rather than on the potential benefits offered by the redundancy. Diploidy and dominance [4] offer another way to effect a dynamic many-to-one mapping between genome and phenome, and have been shown to improve performance in some dynamic optimization problems. Similarly, gene "regulation" can be used to turn various genes on and off and thereby maintain hidden diversity, enabling rapid adaptation to dynamic changes in the fitness landscape [5].

Given that evolutionary computation algorithms were inspired by biological evolution, we find it somewhat surprising that we could find few references in the literature to implementations that adopted a more biologically motivated encoding between genotype and phenotype. Banzhaf and colleagues [6-8] are a rare exception to this, and have shown that they can improve results in genetic programming by adopting a partially redundant genotype-to-phenotype encoding, loosely modeled on the genetic code, to map binary strings to symbols in a genetic programming application. We briefly discuss biological encoding in the next section, and then address how we have adapted this for use in evolutionary algorithms.

1.2 Genotypic and Phenotypic Data Representation in Biological Systems.

In DNA, there is a 4-letter alphabet at the genotype level $\mathbb{G} = \{a, g, c, t\}$, representing the four nucleotide bases (adenine, guanine, cytosine, and thymine). A string of DNA bases is translated into a protein, or string of amino acids, using a codon triplet translation scheme (i.e. each sequence of 3 bases within the coding portions of genes is translated into 1 amino acid, or terminates the amino acid sequence). Mutation and recombination occur at the genotype level on the DNA strings, whereas the translated sequences of amino acids are the proteins that comprise the basis of the phenotype upon which selection acts. Since there are $4^3=64$ possible codon values, but only 20 amino acids (plus the stop codon), this is a redundant coding scheme (64 to 21) with some distinct codons mapping to the same amino acid, as shown in Table 1.

Table 1. The "universal" codon translation table from codon triplets to the 20 amino acids (shown by their standard 3-letter abbreviations) and the stop codon [9]. Here, the three dimensions of the table have been compressed to two

2nd codon position

1st codon position		T	C	A	G	3rd codon position
		Phe	Ser	Tyr	Cys	T
		Phe	Ser	Tyr	Cys	C
		Leu	Ser	Stop	Stop	A
	T	Leu	Ser	Stop	Trp	G
		Leu	Pro	His	Arg	T
		Leu	Pro	His	Arg	C
		Leu	Pro	Gln	Arg	A
	C	Leu	Pro	Gln	Arg	G
		Ile	Thr	Asn	Ser	T
		Ile	Thr	Asn	Ser	C
		Ile	Thr	Lys	Arg	A
	A	Met	Thr	Lys	Arg	G
		Val	Ala	Asp	Gly	T
		Val	Ala	Asp	Gly	C
		Val	Ala	Glu	Gly	A
	G	Val	Ala	Glu	Gly	G

A quick examination of Table 1 reveals that the organization of the genetic code is non-random, with most of the redundancy occurring in the 3rd codon position. Crick [10] has proposed, in his "wobble hypothesis", that this may be due to biochemical limitations that cause decoding non-specificity by transfer RNA. However, statistical arguments have also been used to show that the code is "optimal" in that it minimizes errors, since erroneous codons either code for the same amino acid or to ones with similar hydrophobicity, therefore minimizing the phenotypic impact of single nucleotide polymorphisms [11]. More recently, Freeland [12] proposed that the redundant genetic code may actually act to maximize the rate of adaptive evolution to changing environments. Doubtless there are biochemical constraints that have strongly influenced why life as we know it has evolved with a base alphabet of 4, a codon size of 3, and this particular redundant codon mapping to the 21 possible phenotypic outcomes. Nonetheless there is, biochemically speaking, room for at least some variability. For example, there is a growing body of information regarding the existence of non-standard genetic codes in vertebrate mitochondria as well as in nuclear DNA in a wide range of organisms [13]. Even the size of the basic alphabet is potentially variable [14-15]. Given the highly conserved nature of the redundant codon-based genetic code, despite its demonstrated evolvability, we are motivated to explore the potential advantages such an encoding may have to offer in evolutionary search.

Certainly, the implementation of a codon-based representation affects the possible schemata [1] that can be manifested. Over the past decade, the potential information theoretic benefits of a codon-based representation have begun to receive some attention in the literature [6-8, 12, 16]. Using a Fourier analysis, Kargupta [16] demonstrates that some, but not all, codon-like translation schemes can enable

polynomial time approximations of some exponentially hard problems. However, his analysis is so far restricted to binary alphabets, and does not address the more general problem of how to optimally construct a codon translation scheme for arbitrary phenotypic alphabets and arbitrary problems. Freeland [12] argues that the structure of the redundant genetic code maximizes the probability that mutations will be favorable in dynamic fitness landscapes. The empirical studies by Banzhaf's group [7-8] found that a binary-to-symbolic codon mapping scheme in a genetic program improved search performance over a traditional genetic program in which genotype and phenotype were the same. These studies are intriguing and indicate the need for more comprehensive research in this area. In this manuscript we report on preliminary studies in codon-based representations in genetic algorithms.

2   Methods

We restricted our test problems to those with phenotype alphabet size of 21 distinct values, as in the biological phenotype. In what we will refer to as the "non-codon GA", the alleles in each chromosome consisted of 21 distinct integers. In the "codon GA" we continued with the biological metaphor and restricted the alleles in each genotypic chromosome to an alphabet size of 4 and a codon size of 3 (each three adjacent alleles were logically associated into a codon triplet). Thus, the codon-based chromosomes had three times as many entries as the non-codon based chromosomes, although the alphabet size was smaller (i.e., with packing, memory requirements could be made similar). In both cases, mutation and recombination occurred on these chromosomes of genotypic alleles. Mutation step size for each allele in the genotype, for the experiments reported here, followed a negative exponential distribution, with a range of mutation step size of 0..1 for the codon-GA genotype alphabet {1..4} and a range of 0..2 for the non-codon GA genotype alphabet {0..20} (we determined these values through experimentation with a variety of mutation step sizes and selected the values that generated the best performance for each type of GA on our test problems). Single-point crossover was employed for recombination, and elitism was implemented such that the best individual from each generation was included in the population for the next generation.

In the non-codon GA, the genotype doubled as the phenotype; i.e., the modified chromosomes were directly input into the objective function for fitness calculation (Figure 1a). In the codon GA, each codon (sequence of three alleles) was first mapped to one of the 21 possible phenotype values via a redundant codon translation table to yield the phenotype (sequence of values each in range 0..20); the phenotype was then input to the objective function for fitness calculation (Figure 1b).

Selection was performed using stochastic universal sampling [1]. For each problem, we ran 200 replicates (i.e., we used 200 distinct starting seeds for the random number generator) of the non-codon GA and the codon GA with three distinct codon translation tables (see Section 2.2), where each replicate had a population size of 300 individuals (chromosomes). For the results shown here, each replicate was allowed to run for 100 generations. Monte Carlo replicates of the non-codon GA and each codon GA were paired, in that the same set of 200 starting seeds for the random number generator was used for the non-codon and codon versions, and the paired individuals started with the same fitness. This latter was accomplished by first

generating random populations of individuals for the non-codon GA, then stochastically decoding them. I.e., using the codon table in reverse, we stochastically selected one of the matching codon triplet values for each corresponding phenotypic value, in order to ensure initial genotypes that yielded the same initial phenotypes as in the non-codon GA.
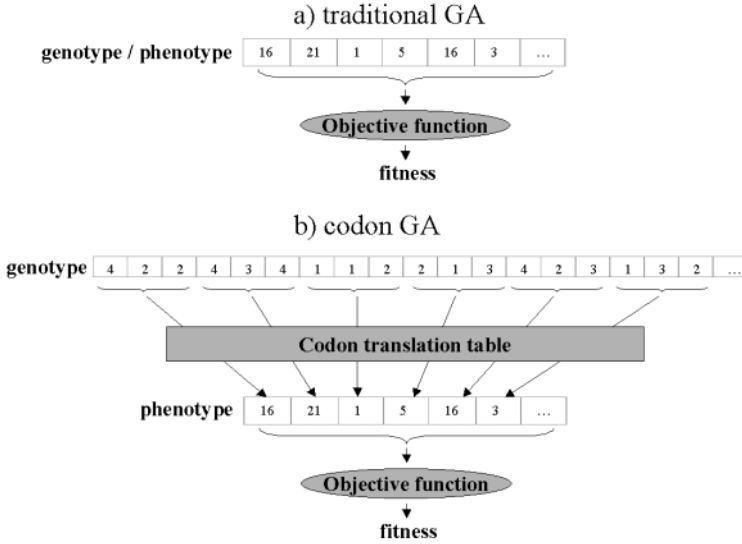

Fig. 1. Schematic of our implementation of data representation in a) a traditional non-codon genetic algorithm, and b) genetic algorithms using codon translation tables to separate phenotype from genotype

2.2 Creation of Codon Translation Tables

We defined two indices to characterize codon tables. The "similarity index" is a measure of how similar the codon triplets are within the various semantic "groups", where a semantic group is defined as the set of codon values that translate into the same phenotypic value. In a codon triplet C, two distinct codon values (C^i and C^j) can share allele values at 0, 1, or 2 positions; we use the number of shared allelic values, at each of the 3 codon allele positions k, between codons in a group as a measure of similarity within the group. The sum (S_g) of the maximum possible number of shared alleles between any two pairs of codons in a group g with a given number of members n_g can be calculated based on the size of the genetic allele alphabet (e.g., 4), and the size of a codon (e.g., 3), and is used to normalize the index. The similarity index (Sim_t) of a codon mapping table t with m groups (e.g., 21) is then defined as follows:

$$Sim_t = \frac{\sum\limits_{g=1..m}\left(\sum\limits_{i=1:n_g-1}\sum\limits_{j=i+1:n_g}\sum\limits_{k=1:3}C_k^i \equiv C_k^j\right)}{\sum\limits_{g=1..m}S_g}, \text{ where } C_k^i \equiv C_k^j \text{ means } \begin{cases}1, \text{ if } C_k^i = C_k^j \\ 0, \text{ otherwise}\end{cases} \quad (1)$$

Similarity indices thus range from a maximum of 1.0 (each codon group is as similar as possible) to a minimum of 0.0 (all pairs of codons in each codon group share no common alleles).

We also define an "adjacency index" for codon mapping tables. A phenotypic value is considered adjacent to another phenotypic value if it can be reached by a single base mutation in the genotype. The adjacency index (Adj_t) of a codon table t with m groups is defined as the sum, over all groups g, of the number of unique phenotypic values (not including the phentoype for the group itself) that can be reached by any single allele change in any codon value in the group. Higher adjacency indices thus imply that more phenotypic values are adjacent (i.e. they define the average number of modes in the multi-modal mutation step size distribution, not counting the mode at zero). Unlike the similarity index, the adjacency index is non-normalized.
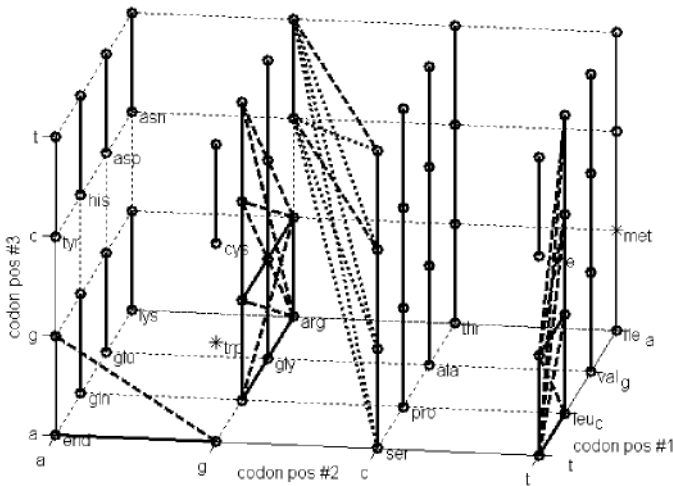

Fig. 2. A graphical representation of the universal biological codon translation table shown in Table 1. Each group of codon triplets that translate into the same phenotypic value are connected by lines depicting the number of base values (out of the three positions) that are the same between any two codon values in the group; solid lines mean they share two base in common, dashed lines mean they share one base in common, and dotted lines mean they share no bases

We can depict the biological codon table shown in Table 1 as a 3-D graph, with potential allelic values for each of the three codon positions plotted on a different axis (Figure 2). Here, we have used lines to connect codon values (nodes in the graph) for every two pairs of codons in each of the same semantic group; solid lines denote 2

shared alleles, dashed lines denote 1 shared allele, and dotted lines denote 0 shared alleles. The biological codon table has a high similarity index $Sim_{biological}$=0.92 (most of the codons in each group share the same first two allelic values, as reflected by all the vertical lines), but a relatively low adjacency index $Adj_{biological}$=8.10, suggesting that perhaps evolution has acted to maximize similarity but minimize adjacency.

In the biological codon table, semantic group sizes range from 1 to 6, reflecting the unequal frequencies of the various amino acids and the stop codon. However, if we are to adapt the codon translation idea to unbiased evolutionary algorithms designed to solve arbitrary black box optimization problems, then we desire equal *a priori* probabilities of all possible phenotypic values (i.e., all semantic groups should be as close to the same size as possible). For these initial studies, we selected problems with the same alphabet sizes as in the biological problem. Thus, within the constraints of mapping the 64 possible codon values onto 21 phenotypic values, we chose to create 20 codon group sizes of 3 and one codon group size of 4. We created 3 distinct mapping tables to try with our codon GA, each with different characteristics of similarity and adjacency. The first table, HL, has High (maximum) similarity and Low adjacency, similar to the characteristics of the biological codon table (Figure 3a). The second table, HM, has High (maximum) similarity and Medium high adjacency (Figure 3b). The third table, LH, has Low similarity (the minimum possible, given that there is one codon group size of 4) and High adjacency (Figure 3c). The similarity and adjacency indices of these three tables, and of the biological table, are given in Table 2. In this study, we have not yet accounted for the phenotypic nearness of adjacent phenotypes in the codon translation table; phenotypes were arbitrarily assigned to codon groups.
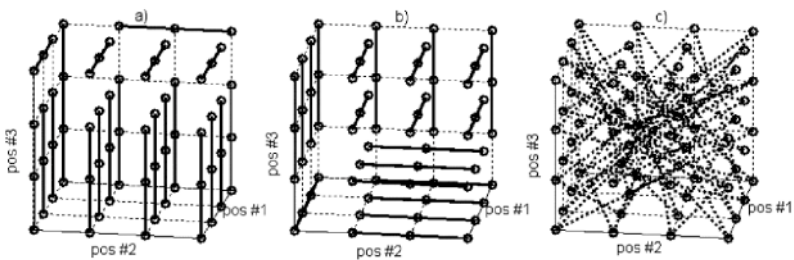

Fig. 3. Graphical depictions of the 3 codon translation tables used in this study. a) HL, b) HM, and c) LH. Meanings of linetypes are as described in Figure 2

2.3 Test Problems

In these preliminary studies, we tested the non-codon GA and the codon GA (using each of the three codon translation tables) on two arbitrarily selected difficult test problems: 1) Dynamic Knapsack, and 2) Indecisive.

2.3.1 Dynamic Knapsack

The objective in the "Dynamic Knapsack" problem is to fill a knapsack with items, so as to maximize the sum of the values of the items in the knapsack, without violating a

weight constraint. (This was converted to a minimization problem by inverting the sum of values; a penalty term was used to enforce the weight constraint.) We arbitrarily specified that there were 5 distinct object types with randomly assigned real values and weights. We allowed between 0 and 20 of each item type placed in the knapsack. The phenotype for each individual was thus a sequence of 5 integers, each in the range 0..20.   The problem was made dynamic by alternating the weight constraint between 100 and 400 each 5 generations. The true optimal solution was determined for each random set of values and weights, and for each weight constraint, by exhaustive search, for comparison to evolved solutions.

Table 2. Similarity and adjacency indices of the biological codon table and the three artificial codon tables used in these experiments

Codon Table	Sim	Adj
Biological	0.92	8.10
HL	1.00	7.05
HM	1.00	11.05
LH	0.01	15.52

2.3.2 Indecisive

The phenotype for the "Indecisive" problem is again a sequence of integer values, each in the range 0..20. The results shown here are for sequences of length 10. The objective function is computed as follows: 1) count the frequency of occurrence of each value, 0 through 20. 2) Subtract 1 from all frequency counts *except for* the count for one arbitrarily chosen (but fixed) number. 3) Take the maximum of the resulting modified counts. 4) Invert this value to turn this into a minimization problem. Note that Indecisive is a difficult search problem in that it has a multi-modal search space with many attractive local suboptima, but only one global optimum, all located at maximum Hamming distances from one another.

3   Results

3.1 Dynamic Knapsack

The results for Dynamic Knapsack, on one set of representative random object weights and values, are plotted in Figures 4 and 5. In all cases the GAs exhibited rapid adaptation to each of the individual weight constraints over each series of 5 generations for which the same weight constraint applied. Since all solutions found that satisfied the weight constraint of 100 were also feasible solutions at the weight constraint of 400, the transition from weight constraint of 100 to 400 was relatively easily accommodated. However, since the converse is not true, the transition from weight constraint of 400 to 100 caused a large increase in objective function value and this "overshoot" was notably worse for the non-codon GA than for the codon GA, and continued to get worse with each cycle (Figure 4). On the other hand, overshoot for all three codon GAs was much lower than for the non-codon GA (note the log scale in Figure 4), and tended to stay constant or even reduce (e.g., when using the

codon translation table LH) over successive cycles. Because the scale differences in these results make comparisons difficult to see in Figure 4, we present a different view of the data in Figure 5. In Figure 5a, we show the objective function values at the end of each period with weight constraint of 100; here, it is apparent that the non-codon GA is not able to recover quickly enough from the increasing overshoot in order to adapt well to the periodic weight constraint. A close-up view of this same data is plotted in Figure 5b, to enable discrimination between the codon GA using the three tables. All three codon tables gave good results and in contrast to the non-codon GA they all continued to improve their adaptation to the weight constraint of 100 over the periodic cycles. It is of interest that codon table HL had the highest overshoot (i.e., greatest variability in solution quality) (Figure 4) but evolved the lowest objective function values at the ends of each 5 successive generations at weight constraint 100. In Figure 5c we show the objective function values at the end of each period with weight constraint of 400; in this case, all the GAs did well and the performance over successive periodic cycles was fairly constant, with the non-codon GA showing slightly better adaptation to the higher weight constraint than the codon GAs.
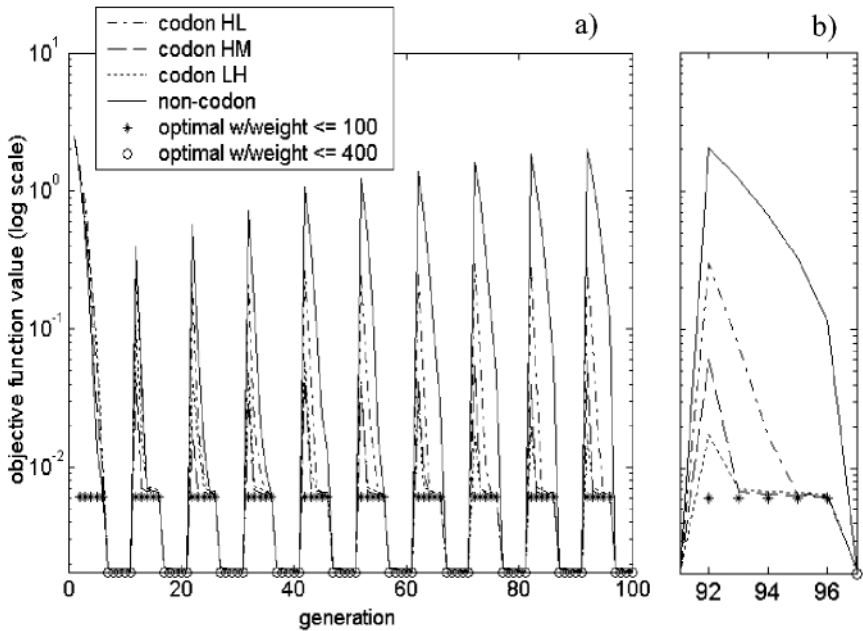

Fig. 4. a) Average over 200 replicates of the objective function values for the best individual in a population of 300 individuals for both codon and non-codon GAs on the Dynamic Knapsack problem. The symbols indicated the step function representing the true global optima under the two different weight constraints. Note the log scale. b) Close up of the last cycle with weight constraint 100.

3.2 Indecisive

The results on Indecisive are shown in Figure 6, where there is a dramatic difference in performance between the non-codon GA, which never found the global optimum within 100 generations, and the codon GAs, which always found the global optimum
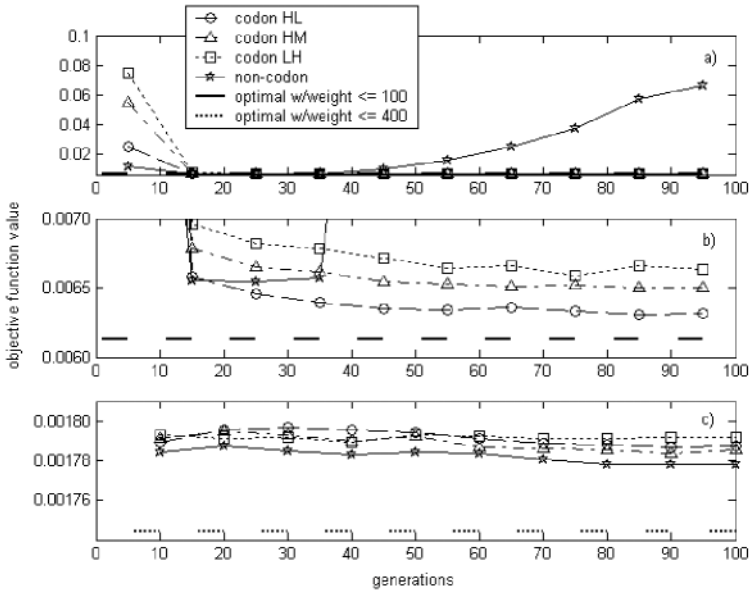

Fig. 5. Three views of the data plotted in Figure 4. a) The objective function value at the end of each interval with a weight constraint of 100. b) Close-up of data plotted in a. c) The objective function value at the end of each interval with a weight constraint of 400

rapidly. In this problem, the codon GA using codon table HL consistently found the global optimum the most rapidly.

4   Conclusions

Our preliminary results imply that, at least for some problems, a redundant codon-based representation positively affects the ability of a genetic algorithm to a) rapidly adapt to dynamic changes in the fitness function and b) find a global optimum in a difficult multi-modal search space. The optimal choice of how to organize the codon translation table for a cyclical fitness landscape appears to depend on whether it is more important to minimize the variability of solution quality (low similarity, high adjacency) or the evolution of the best solution quality over time (high similarity, low adjacency). In the static problem we tested, the codon translation table with high
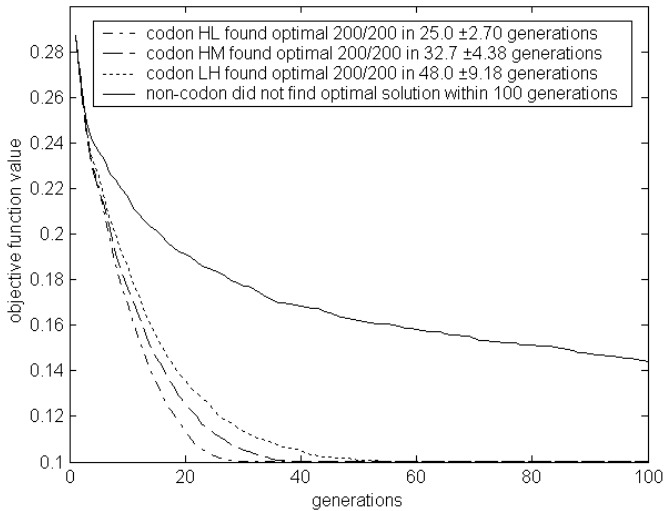
Fig. 6. Average, over 200 replicates, of the objective function values for the best individual in a population of 300 individuals, for both codon and non-codon GAs on the Indecisive problem. Values in the legend indicate the number of times the global optimum was found out of the 200 replicates, and the number of generations required to find it is shown as mean ± standard deviation

similarity and low adjacency yielded both lowest average values and variances of number of generations to find the single global minimum. Interestingly, high similarity and low adjacency are the characteristics that have evolved in the biological codon table. Much more work in this area is warranted to further understand the mechanisms by which redudant codon-based representations enhance search, which evolutionary algorithms and application problems can benefit from this approach, and how to best select the fundamental parameters such as the size of the codon, the size of the genotypic alphabet at each codon position, and the internal organization of the codon translation table. In addition, it will be important to explore whether the effects of redudant codon translation schemes are complementary, synergystic, or alternatives to the benefits offered by other types of redundant data reprentations in evolutionary algorithms.

References

1. Bäck T. Fogel, D.B., and Michalewicz, T. (eds): Evolutionary Computation 1: Basic Algorithms and Operators, Inst. of Physics Publ., Bristol, UK (2001)

2. Rothlauf, F.: Binary representations of integers and the performance of selectorcombinative genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco, California (2002) 695

3. Goldberg, D.E., Deb, K., and Korb, B.: Don't worry, be messy. In: R.K. Belew and L.B Booker, (eds.): Proc. 4th Intl. Conf. On Genetic Algorithms. Morgan Kaufman, San Mateo, CA (1991) 24–30

4. Goldberg, D.E., and Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J.J. Grefenstette, (ed.): 2nd Intl. Conf. On Genetic Algorithms. Lawrence Erlbaum Assoc. (1987) 59–68

5. Dasgupta, D.: Incorporating redundancy and gene activation mechanisms in genetic search for adapting to non-stationary environments. In: L. Chambers, (ed.), Practical Handbook of Genetic Algorithms: New Frontiers (Vol II). CRC Press, (1995) 303–316

6. Banzhaf, W.: Genotype-Phenotype-Mapping and Neutral Variation – A case study in Genetic Programming. In: Y. Davidor, H.-P. Schwefel and R. Männer, (eds.) Proc. Parallel Problem Solving from Nature III, Lecture Notes in Computer Science (1994) 322–332

7. Keller, R., and Banzhaf, W.: Genetic Programming using Genotype-Phenotype Mapping from Linear Genomes into Linear Phenotypes. In: J. Koza, D. Goldberg, D. Fogel, R. Riolo, (eds.) Proc. 1st annual Conf. on Genetic Programming (GP-96). MIT Press, Cambridge, MA (1996) 116–122

8. Banzhaf, W.: The Evolution of Genetic Code in Genetic Programming. In: W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela and R. Smith (eds.) Proc. of first GECCO conference, Morgan Kaufmann, San Francisco. (1999) 1077–1082

9. Creighton, T.E.: Proteins: Structures and Molecular Properties, 2nd Ed., W.H. Freeman and Co., New York (1993)

10. Crick, F.H.C.: Codon-anticodon pairing: the wobble hypothesis. *J. Mol. Biol.*, **19** (1966) 548–555

11. Freeland, S.J. and Hurst, L.D.: The genetic code is one in a million. J. Mol. Evol., **47** (1998) 238–248

12. Freeland, S.J.: The Darwinian genetic code: an adaptation for adapting? Genetic Programming and Evolvable Machines, **3** (2002) 113–127

13. Knight, R.D., Freeland, S.J., and Landweber, L.F.: Rewiring the keyboard: evolability of the genetic code. Nature Reviews: Genetics, **2** (2001) 49–58

14. Szathmáry, E.: What is the optimum size for the genetic alphabet? Proc. Natl. Acad. Sci., USA, **89** (1992) 2614–2618

15. Piccirilli, J.A., Karuch, T., Moroney, S.E., and Benner, S.A.: Enzymatic incorporation of a new base pair into DNA and RNA extends the genetic alphabet. Nature **343** (1990) 33–37

16. Kargupta, H.: A striking property of genetic code-like transformations. Complex Systems, **13** (2001) 1–32