Hill Climbing on Discrete HIFF: Exploring the role of DNA Transposition in Long-term Artificial Evolution

Susan Khor
Concordia University, Canada H3G 2W1
slc_khor@cse.concordia.ca

ABSTRACT

We show how a random mutation hill climber that does multi-level selection utilizes transposition to escape local optima on the discrete Hierarchical-If-And-Only-If (HIFF) problem. Although transposition is often deleterious to an individual, we outline two population models where recently transposed individuals can survive. In these models, transposed individuals survive selection through cooperation with other individuals. In the multi-population model, individuals were allowed a maturation stage to realize their potential fitness. In the genetic algorithm model, transposition helped maintain genetic diversity even within small populations. However, the results for transposition on the discrete Hierarchical-Exclusive-Or (HXOR) problem were less positive. Unlike HIFF, HXOR does not benefit from random drift. This led us to hypothesize that two conditions necessary for transposition to enhance evolvability are (i) the presence of local optima and (ii) susceptibility to random drift. This hypothesis is supported by further experiments. The findings of this paper suggest that epistasis and large mutations can sustain artificial evolution in the long-term by providing a way for individuals and populations to escape evolutionary dead ends. Paradoxically, epistasis creates local optima and holds a key to its resolution, while deleterious mutations such as transposition enhance evolvability. However, not all large mutations are equal.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods and Search

General Terms

Algorithms, Design

Keywords

hill climbing, crossing fitness saddles, hierarchical test problems, evolvability

1. INTRODUCTION

A local optimum for a hill climber is an evolutionary cul-de-sac brought on by a myopic view of the adaptive landscape and a short-term gain mentality. In general, there are two ways to handle dead-ends of any kind: avoidance and back-tracking. We explore these strategies on the discrete Hierarchical-If-And-Only-If (HIFF) problem [19] which has an exponential number of local optima for hill climbers and is not trivial for genetic algorithms either with population diversity being a critical factor [18].

The interaction strength between variables in the HIFF problem are such that lower level modules find their optimum more quickly than higher level modules. This difference in adaptation speed between levels makes HIFF nearly decomposable [13]. Further, although there is no conflict between optimal modules at different levels, assembling optimal modules does not necessarily produce larger optimal modules. Hence the poor performance of the random mutation hill climber (RMHC) [5] on the HIFF problem [18].

Reference [10] proposed an approach to solve the continuous HIFF problem with an enhanced random mutation hill climber called RMHC2. RMHC2 does multi level selection. That is, instead of comparing aggregate fitness values, phenotypes are created from the per-level fitness values of HIFF genotypes and selection is by comparison of phenotypes. In the ideal case, this comparison is done in top-down order so that priority is given to adaptation of higher level modules. A strict top-down order is not necessary and it is possible to find a global optimum while evolving both genotypes and the order of comparison. These results also apply to continuous Hierarchical-Exclusive-Or (HXOR). Section 2.2 elaborates on the RMHC2 algorithm.

However, we found that RMHC2 is not successful on discrete HIFF because there are fewer stepping stones in discrete HIFF (section 2.3). As such, discrete HIFF has wider fitness saddles for RMHC2. In the extreme case, the fitness landscape for the highest level of a discrete HIFF problem is two needles in a haystack.

In this paper, we propose, RMHC2-T, a further enhancement of RMHC2, to solve the discrete HIFF problem. RMHC2-T uses a nature-inspired mechanism called *transposition* to perturb a search from approaching a local maximum and thus avoid the widest fitness saddles. The general idea is: during a transposition event, a segment of a genotype, a *transposon*, is moved to another location in the same genotype. Due to interaction between parts of a genotype, the cut-and-paste action of a transposition triggers a series of adaptive moves. Thus a genotype heading along a dead ended evolutionary pathway is diverted toward a hopefully more promising path. Section 3 explains our implementation of the transposition operation and section 4 gives the algorithm for RMHC2-T.

Because transposition is a fairly large mutation, its effect is most likely deleterious. But, as our experiments in section 5 confirm, due to the epistatic nature and modularity of the HIFF problem, this temporary loss of *actual* fitness can be made up for by a gain in *potential* fitness, that is the ability to reach a higher fitness point than the one just descended from, after some number of evolutionary steps. By no means is this a straightforward process. The HIFF problem has two global optima. Descending from one local peak can inadvertently nudge the genotype towards the global optima that is before transposition much further away

Hamming distance wise – in other words, in the wrong direction. Further, there is the problem of evolutionary plausibility. Can a transposed genotype survive in the competitive environment of a population? To reap the benefits of a transposition, a genotype has to survive long enough to realize its potential fitness. We address this question in section 6.

RMHC2-T allows its genotype to descend into the valley of lower fitness points. However, this descent is not arbitrary. There are other ways to permute a genotype, such as by random bit shuffle or inversion [7], but they did not perform as well as transposition in our experiments (section 5).

RMHC2-T is less successful on the discrete HXOR problem. Due to its more complicated string pattern, HXOR does not benefit from random drift (section 7).

The experiments with RMHC2-T lead us to the following hypothesis: that transposition enhances evolvability on problem-algorithm pairs with local optima and that can exploit random drift. We test this notion on a broader range of formal problems in section 8.

From a computational standpoint (as opposed to a biological one), Holland's translocation operator [7] is similar to conservative transposition in that both operators relocate parts of a whole within the whole. However Holland proposed translocation in the context of multi-chromosomal genotypes and segregation. Translocation moves genes between chromosomes of a genotype. The little experimentation on the utility of translocation in genetic algorithms did not produce positive conclusive results [6, p.180].

Epistasis and viability selection are often culprits of local optima in individual and in population search. Large mutations in individuals are often deleterious in the short term. However, the findings in this paper suggest that when combined under the right conditions, they can enhance evolvability in the long term and produce objects with many interacting parts.

2. BACKGROUND

2.1 Hierarchical Decomposable Problems

The HIFF and HXOR problems are instances of the class of hierarchically decomposable problems introduced by Watson et al. [19]. The HIFF problem in particular was used to distinguish between gradual and compositional evolution and to demonstrate the importance of modularity [18]. A number of approaches have been proposed to solve the HIFF problem and variants of the hierarchically decomposable problems [4, 11, 12, 14, 16]. However, beside RMHC2, we have not found one which is based solely on random mutation of the genotype and selection.

In this section, we describe HIFF genotypes. This description also applies to HXOR genotypes unless stated otherwise. The genotypes are bit strings of length $N = 2^n$. A genotype has $\log_2 N$ levels, and $N/2^\lambda$ modules at level λ, with $\lambda = 1\ldots n$. The algorithm we use rewards one fitness point to each module that satisfies the constraints of the problem. For a HIFF problem, the constraint is similarity. For a HXOR problem, the constraint is dissimilarity. Hence, by this algorithm, the aggregate fitness value of an optimal genotype is N-1. An optimal HIFF genotype is one with all zeroes or all ones. An optimal HXOR genotype is maximally dissimilar at every level. For example, 1001 0110 and 0110 1001 are the optimal genotypes for HXOR with 8 variables. Appendix A (section 12) describes a procedure to produce optimal HXOR genotypes.

Reference [10] gives the algorithm to calculate continuous HIFF. This algorithm is similar to the HIFF function defined in [18] but is modified to calculate fitness by levels. Essentially, fitness of a continuous HIFF module at a level is given by $(p \times q) + (1 - p) \times (1 - q)$ where p is the proportion of one bits in the first half of a module and q is the proportion of one bits in the second half of a module (see Table 1C for an example). We present the algorithm to calculate discrete HIFF level fitness in Figure 1. It works by first "shrinking" the genotype to the right level and then counting the number of non-null matches in the shrunken genotype. To calculate HXOR-D level fitness, substitute the condition (e1 is the same as e2) with (e1 is not the same as e2).

```
PROCEDURE: calculateHIFFDLevelFitness
INPUT: λ, genotype
OUTPUT: level_fitness
BEGIN
    g2 ← genotype
    FOR EACH level i from 1 to λ
        num_modules ← size of g2 / 2
        empty g1
        FOR EACH module j from 0 to num_modules – 1
            msp ← j * 2          // msp is module start position
            e1 ← element in g2 at msp
            e2 ← element in g2 at msp + 1
            IF (e1 is not null and e2 is not null and
                                              e1 is the same as e2)
                add e1 to g1
            ELSE
                add null to g1
        END FOR
        g2 ← g1
    END FOR
    len ← size of g2
    level_fitness ← 0.0
    FOR EACH i from 0 to len-1
        IF (element in g2 at i is not null)
            level_fitness ← level_fitness + 1.0
    END FOR
END
```

Figure. 1 Pseudo-code to calculate level λ discrete HIFF fitness for a genotype

Tables 1A, 1B and 1C illustrate the calculation of discrete HIFF (HIFF-D), discrete HXOR (HXOR-D) and continuous HIFF (HIFF-C) per-level fitness and aggregate fitness values for genotype number 193 or bit string 1100 0001, respectively. To distinguish between fitness values and bits in these tables, real numbers is the domain of fitness values for the discrete and the continuous functions. The HIFF-D fitness of level 1 is 3.0 because at each level, a 1 or 0 bit is given 1.0 fitness points and a null (-) is worth 0.0 fitness points. Similarly for HXOR-D. The HIFF-C fitness of level 1 is 1.0 + 1.0 + 1.0 + 0.0 = 3.0.

Table 1A. Discrete HIFF fitness.

	Number of	Genotype 193								HIFF-D
Level	Modules	1	1	0	0	0	0	0	1	Per level fitness
(lowest) 1	4	1		0		0		-		3.0
2	2	-				-				0.0
(highest) 3	1	-								0.0
									Phenotype	⟨ 0.0, 0.0, 3.0 ⟩
									Aggregate fitness	3.0

Table 1B. Discrete HXOR fitness.

Level	Number of Modules	Genotype 193								HXOR-D Per level fitness
		1	1	0	0	0	0	0	1	
(lowest) 1	4	-		-		-		1		1.0
2	2	-				-				0.0
(highest) 3	1	-								0.0
Phenotype										$\langle$ 0.0, 0.0, 1.0 $\rangle$
Aggregate fitness										1.0

Table 1C. Continuous HIFF fitness.

Level	Number of Modules	Genotype 193								HIFF-C Per level fitness
		1	1	0	0	0	0	0	1	
(lowest) 1	4	1.0		1.0		1.0		0.0		3.0
2	2	0.0				0.5				0.5
(highest) 3	1	0.5								0.5
Phenotype										$\langle$ 0.5, 0.5, 3.0 $\rangle$
Aggregate fitness										4.0

2.2 RMHC and RMHC2 algorithms

RMHC2 is derived from RMHC [5]. RMHC complements a fixed number of *k* bits at a time and keeps an offspring if it is as fit as or fitter than its parent. RMHC2 complements a variable number of *k* bits at a time and has two additional components: (i) phenotypes and (ii) a multi-level selection scheme. Table 1A in section 2.1 gives an example of a HIFF phenotype. The per level fitness values of a genotype is sequenced to form the phenotype for the genotype, with the highest level (λ = n) fitness value situated at the leftmost position in a phenotype, followed by the second highest (λ = n-1) level fitness value and so on. The optimal HIFF phenotype for a problem with N = 2^n variables is $\langle 2^0, 2^1, \ldots, 2^i, 2^{i+1}, \ldots 2^{n-1}\rangle$. The same applies for HXOR.

RMHC2 uses a multi-level selection scheme to compare phenotypes. This selection scheme has a *sieve* component which defines the order in which the elements of a phenotype are compared. In evolutionary terms, the elements of a phenotype can be viewed as features and a sieve, as the relative importance of each feature to the survival of a genotype. Different features are prized in different environments and sieves reflect these relationships. A sieve is represented as an array of positive integers. In a problem with 3 levels, any 3-permutation of the set {1, 2, 3} is a valid sieve. There are *n*! possible sieves for a problem with *n* levels. Importance of a feature decreases from left to right in a sieve. The adaptation of more important features are prioritized over the adaptation of less important features.

Suppose the sieve for a HIFF problem with size N = 8 is $\langle 2, 1, 3\rangle$ and the two competing phenotypes are $p_1 = \langle a, b, c\rangle$ and $p_2 = \langle x, y, z\rangle$. Then the first comparison is made between the second level features *b* and *y*. This is followed by comparisons between *c* and *z*, and between *a* and *x*. In a maximization problem, if ($b > y$) or ($b = y$ and $c > z$) or ($b = y$ and $c = z$ and $a > x$), then the genotype for p_1 is selected. Otherwise, the genotype for *p2* is selected. If all three per-level fitness values are pairwise equal, then the genotype for p_2 is selected by default. The "ideal" sieve is $\langle n, n\text{-}1, n\text{-}2, \ldots, 1\rangle$ and does not allow any lower level feature to adapt at the expense (devolution) of a higher level feature. Unless stated otherwise, RMHC2 uses the "ideal" sieve in this paper. Table 2 compares the RMHC and RMHC2 selection schemes on two pairs of genotypes.

2.3 Discrete versus Continuous HIFF

The discrete HIFF function differs from its continuous counterpart in one significant way for RMHC2. The levels in discrete HIFF

Table 2. RMHC versus RMHC2 selection. By default, RMHC2 uses the "ideal" sieve which is $\langle 3, 2, 1\rangle$ when N=8.

Genotype	HIFF-D Phenotype			Aggregate Fitness	Selection	
	λ=3	λ=2	λ=1		RMHC	RMHC2
245 (parent) 1111 0101	0	**1**	2	3	188	245
188 (offspring) 1011 1100	0	0	3	3		
188 (parent) 1011 1100	0	0	3	3	245	245
245 (offspring) 1111 0101	0	**1**	2	3		

are closed in the sense that a module at level λ is not privy to the intermediate state of its two constituent modules at level λ-1. A discrete HIFF module at level λ "knows" about the state of its two constituent modules at level λ-1 only when they happen to form an optimal level λ module. Prior to the formation of an optimal module at level λ, the bits within the two nested modules at level λ-1 are free to mutate according to the constraints at level λ -1 and below. Hence, higher level modules in discrete HIFF cannot bias the adaptation of lower level modules.

On the other hand, a module at level λ in continuous HIFF "knows" about the intermediate state of its two constituent modules at level λ-1. In continuous HIFF, fitness of a module at a level is given by $(p \times q) + (1 - p) \times (1 - q)$ where p is the proportion of one bits in the first half of a module and q is the proportion of one bits in the second half of a module. So level fitness in continuous HIFF is sensitive to changes in a genotype at the bit level and higher level modules use this information in RMHC2 to not only preserve progress but also direct adaptation of lower level modules. Whether a level is higher or lower depends on context, a level is higher to its lower levels and is lower to its higher levels. In short, continuous HIFF has a higher degree of top-down control than discrete HIFF.

To see the significance of top-down control to evolution of HIFF genotypes, consider a module at level λ with majority *m*. The majority value, *m*, of a module is the number of the more frequently occurring bit. For example, a module of length 4 with majority 3 is any of the 8 modules with either 3 zero bits and 1 one bit or 3 one bits and 1 zero bit. Modules with the same majority value need not have the same phenotype because the arrangement of the zero and one bits is significant for HIFF. $s/2 \leq m \leq s$ where *s* is module size. The HIFF problem is essentially the problem of maximizing *m* for all modules in a genotype.

In discrete HIFF, a level λ module will have a fitness value of 0 until $m = s$. While $m < s$, the sub-optimal nested modules at level λ-1 will adapt to their optimality. In this process, *m* may increase or decrease. When all modules at level λ-1 are optimal and $m < s$, the only adaptive move for RMHC2 is the one that increases *m* to *s*. Any other mutation will decrease the fitness at level λ-1 without a corresponding increase in fitness at level λ or above. Hence, there is no pressure on the modules at level λ-1 to move away from their equilibrium state. Further, depending on the size of the lower level modules and the size of the mutation step, the move to $m = s$ at level λ may not be possible. In continuous HIFF, fitness of a module changes with changes in *m*. What is important for reaching global optimality is that fitness of a module at level λ or above increases (decreases) with an increase (decrease) in *m*. Hence, while there are sequences of genotypes with increasingly fit phenotypes in both discrete HIFF and continuous HIFF for RMHC2, the Hamming distance between consecutive genotypes in a continuous HIFF sequence is shorter.

The above exposition tells us that there is more uncertainty associated with increasing the majority value of a module in discrete HIFF than continuous HIFF, for RMHC2. More uncertainty means that RMHC2 on discrete HIFF is likely to take longer than RMHC2 on continuous HIFF. But this is not the only consequence. Direction by higher levels is useful in the HIFF problem because it provides the broader view or context of the problem that lower levels lack. Without a broader view, a HIFF module is prone to adapt to a local maximum and the phenomenon of fitness saddles witnessed for RMHC on the HIFF problem [18] resurfaces for RMHC2 on the discrete HIFF problem. Fitness saddle problems cannot be solved by increasing the number of evaluations or time allowed for the search. A different search strategy is required.

The following example illustrates the actualization of a fitness saddle. The width of a fitness saddle can be measured by the Hamming distance between a point at its sub-optimal peak and its nearest equally high or higher (EHH) point. The RMHC2 uses 1-bit flip mutation and the current genotype is genotype 241 or 1111 0001. Table 3 enumerates the fitness values for genotypes under discussion. On HIFF-D, RMHC2 can move to genotype 243 or 240. There is only one EHH point between 243 and the optimum (255). That point is 252. But 252 is 4 Hamming distance units away from 243, and therefore unreachable by 1-bit mutation. RMHC2 is unable to cross this fitness saddle. On HIFF-C, RMHC2 cannot move to 240 but can take several routes to 255. Suppose it also moves to 243. There are two EHH points, 247 and 251, both within reach, and 255 is reachable from either of these two points.

Table 3. Fitness saddle example.

		HIFF-D Fitness				HIFF-C Fitness			
gnum	Genotype	3	2	1	A	3	2	1	A
0	0000 0000	1	2	4	7				
195	1100 0011	0	0	4	4				
203	1100 1011	0	0	3	3				
221	1101 1011	0	0	2	2				
240	1111 0000	0	2	4	6	0.0	2.0	4.0	6.0
241	**1111 0001**	**0**	**1**	**3**	**4**	**0.25**	**1.5**	**3.0**	**4.75**
242	1111 0010	0	1	3	4	0.25	1.5	3.0	4.75
243	1111 0011	0	1	4	5	0.5	1.0	4.0	5.5
244	1111 0100	0	1	3	4	0.25	1.5	3.0	4.75
245	1111 0101	0	1	2	3	0.5	1.5	2.0	4.0
246	1111 0110	0	1	2	3	0.5	1.5	2.0	4.0
247	1111 0111	0	1	3	4	0.75	1.5	3.0	5.25
248	1111 1000	0	1	3	4	0.25	1.5	3.0	4.75
249	1111 1001	0	1	2	3	0.5	1.5	2.0	4.0
250	1111 1010	0	1	2	3	0.5	1.5	2.0	4.0
251	1111 1011	0	1	3	4	0.75	1.5	3.0	5.25
252	1111 1100	0	1	4	5	0.5	1.0	4.0	5.5
253	1111 1101	0	1	3	4	0.75	1.5	3.0	5.25
254	1111 1110	0	1	3	4	0.75	1.5	3.0	5.25
255	1111 1111	1	2	4	7	1.0	2.0	4.0	7.0

Table 4 compares the Fitness Distance Correlation (FDC) [8] statistic for discrete (D) and continuous (C) HIFF[1]. In this table, the FDC statistic measures the correlation between fitness values at a level and hamming distances to the closest global optimum. FDC values closer to 0 indicate increased search difficulty while FDC values closer to -1 indicate decreased search difficulty. For HIFF-C, search is easier at higher levels. In contrast, the FDC values for HIFF-D (N > 4) first move away from 0 then come back towards 0.

[1] FDC may not be a completely reliable measure for level search difficulty. We have found a counter example where per-level FDC values approach 0 at higher levels, but the problem is easily solved by "ideal" RMHC2. Further investigation is underway.

Table 4. FDC values

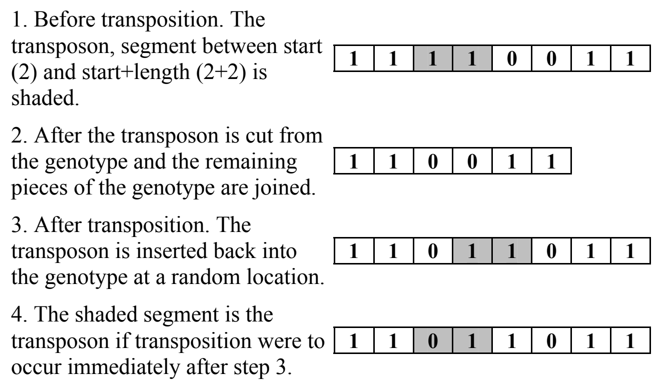
Level	HIFF-D		HIFF-C	
	N=8	N=16	N=8	N=16
4	-	-0.0287	-	-0.6770
3	-0.2877	-0.2463	-0.6972	-0.4787
2	-0.5403	-0.3789	-0.4930	-0.3385
1	-0.3486	-0.2394	-0.3486	-0.2394
Aggregate	-0.4690	-0.3281	-0.5712	-0.4199

3. DNA TRANSPOSITION

The structure and content of the natural genome is not fixed and is not altered solely by nucleotide substitution during recombination and mutation. The genome is plastic, to allow an assortment of gene expression mechanisms. In this paper, we focus on the structural aspect of genome plasticity, that is the physical rearrangement of genes in a genotype without increasing or decreasing the number of genes in the genotype. In nature, one such mechanism is *conservative DNA-only transposition* [1].

There are two broad types of transposition: retro-transposition and DNA only transposition. In retro-transposition a stretch of DNA undergoing transposition, the *transposon*, is copied and the copy is inserted back into the DNA at another location. DNA only transposition may be conservative or replicative but we only consider the conservative case in this paper. Conservative DNA only transposition is a cut and paste operation: a transposable element is removed from the DNA and inserted back into the DNA at another location. A conservative DNA only transposition does not change the composition of a genotype, that is the number of copies of genes in a genotype is not altered. The target site for a transposon is less specific than its donor site. At the donor site, the boundaries of a DNA only transposon is bracketed by a pair of DNA sequences called inverted repeats which indicates to the transposase enzyme where to cut and rejoin the DNA. The exact purpose of transposition is not known although it is recognized as an important source of genetic diversity and it plays an important role in the immune system.

Figure 2 illustrates our implementation of the transposition mechanism. The *start* position denotes the starting position of a transposon in a genotype. The *length* specifies the size of a transposon. The start and length variables are integers randomly


1. Before transposition. The transposon, segment between start (2) and start+length (2+2) is shaded.

2. After the transposon is cut from the genotype and the remaining pieces of the genotype are joined.

3. After transposition. The transposon is inserted back into the genotype at a random location.

4. The shaded segment is the transposon if transposition were to occur immediately after step 3.

Figure 2. Steps 1 to 3 illustrate a transposition.

chosen from [0, half of the genome length] and remain fixed for the life of a genotype and is inherited un-mutated by the progeny of a genotype. Having identified the transposon, this mobile segment is removed from the genotype and the remaining two segments of the genotype are joined. A random position in the now smaller genotype is chosen and the transposon is inserted into that position, expanding the genotype to its original size.

4. THE RMHC2-T ALGORITHM

RMHC2-T is RMHC2 with transposition. The RMHC2-T algorithm is:

1. Start with a fully specified (every loci has a value of either 1 or 0) genotype.
2. Initialize *start* and *length* variables per section 3.
3. If a T event is triggered, perform T on the genotype, calculate the new phenotype, and go to step 7. Otherwise go to step 4. A T event is triggered when the random integer selected from [1, Pt] matches a pre-defined integer within the range. Pt is a parameter of the RMHC2-T algorithm.
4. Make a copy of the genotype. Generate *k* and complement *k* random bits in the clone genotype. *k* is a random integer from [1, $P_m \times N$]. P_m is a real number from (0, 1] and a parameter of the RMHC2-T algorithm.
5. Compare original and clone phenotypes using the multi-level selection scheme. Except for the multi-population model (section 6.1), RMHC2-T always uses the “ideal” sieve (section 2.2) in this paper.
6. If the clone phenotype is fitter, replace the original genotype with the clone genotype. Otherwise, discard the clone genotype.
7. If an optimal genotype has not been found and the pre-determined number of function evaluations has not been performed, go to step 3. Otherwise return the current genotype as the solution.

Note that in RMHC2-T, transposition is performed directly on the original genotype and circumvents the multi-level selection step. Transposed genotypes are exposed to the sifting mechanism of selection in populations (section 6).

To see the effect of transposition on RMHC2, we refer back to the fitness saddle problem in section 2.3 (Table 3). To progress from genotype 243 towards 255, an effective transposition would be one that splits the pair of zeroes since only 1-bit mutation moves are possible. RMHC2 does not allow 243 to move directly to 251. Neither would RMHC since the aggregate fitness of 251 is lower than 243. Suppose 243 is transposed to 221. The genotype suffers a fitness loss from this transposition. However, now it is possible for RMHC2 to move to 251 and from there to 255. This is what we mean by losing actual fitness but gaining potential fitness. However, it is also possible that once at 221, RMHC2 moves to 203 and then to 195. Now there are just as many zeroes as ones in the genotype and the search has moved further away from 255. The multi-level selection scheme of RMHC2 helps control this waffling but does not eliminate it. Transposition and RMHC2 is more efficient than transposition and RMHC1 (section 5). RMHC1 is RMHC2 without the multi-level selection scheme.

5. EXPERIMENTS

Hypotheses for the experiments in this section are:

(i) That transposition helps RMHC2 solve the discrete HIFF problem.

(ii) That RMHC2-T is more efficient than RMHC1-T. RMHC1-T is RMHC2-T without multi-level selection, that is aggregate fitness values are compared instead of phenotypes.

(iii) That transposition is a distinct kind of permutation. We compare transposition with random bit shuffling and with bit inversion.

Preliminary tests confirm that increasing mutation rate, using macro-mutation where the bits to mutate are consecutively located, using RMHC2 but with a selection scheme that only accepts mutations that do not decrease the fitness value of any level, and combinations thereof are not solutions. Table 5 lists the parameter values used in the experiments.

Table 5. RMHC2-T experiments.

Parameter	Value	
Problem size, N	64	128
Number of runs	30	30
Maximum evaluations	500,000	1,500,000
Mutation rate (Pm)	0.0625	0.03125
Transposition (Pt)	1000	1000

Table 6 summarizes the results of the experiments. On the 64-variable problem, RMHC2 did not find a global optimum in any run while RMHC2-T found a global optimum in all 30 runs. This confirms the first hypothesis: that transposition helps RMHC2 solve the discrete HIFF problem.

RMHC1-T was just as successful at finding a global optimum as RMHC2-T on the 64 variable discrete HIFF problem. However, RMHC1-T took significantly more evaluations than RMHC2-T. The probability that there is no difference between the average number of evaluations for RMHC2-T and for RMHC1-T using a 1-tailed T-test is less than 6% with 58 degrees of freedom. This confirms the second hypothesis.

Table 6. Results for HIFF-D.

	N=64			N=128
	RMHC2	RMHC2-T	RMHC1-T	RMHC2-T
Times found	0/30	30/30	30/30	30/30
Avg. evaluations (std. dev.)	-	105,590 (91,398)	148,589 (117,574)	510,772 (416,920)
Median evals	-	65,353	89,156	389,593

We ran RMHC2-T with inversion and random bit shuffling in place of transposition. That is, in step 3 of RMHC2-T (section 4), instead of performing a transposition at T, an inversion or a random bit shuffle is performed. Inversion inverts the bits between start and start + length. For example, 1011 1011 with the middle 4 bits inverted is 1001 1111. A random bit shuffle rearranges the entire genotype. The inversion runs failed on the 64-variable problem, but the random shuffle runs succeeded. So, we tested random shuffle on the 128-variable problem. On this larger problem, random shuffle was not successful while RMHC2-T has a 100% success rate. Thus the hypothesis that transposition is a distinct kind of permutation is confirmed on sufficiently large problems, based on search performance.

However, even on the 64 variable HIFF problem, difference between the quality of random bit shuffle and of transposition permutation is evident. Figure 3A shows the zig-zag pattern that a sample RMHC2-T run makes as it progresses through its search. A sample RMHC2-S run also makes a zig-zag pattern (Figure 3B), but the magnitude of its fluctuations are larger than those in RMHC2-T. Further, there is a discernable leaning to the right (higher fitness) in RMHC2-T. There is little to no zig-zag pattern for RMHC2 (Figure 3A) and RMHC2-I (Figure 3B). Their evolution come to a stand-still. Transposition performs better because although it permutes the genotype it also preserves the physical linkage between bits in a transposon. Random shuffling is disruptive and too permissive of waffling. Inversion is not disruptive enough.

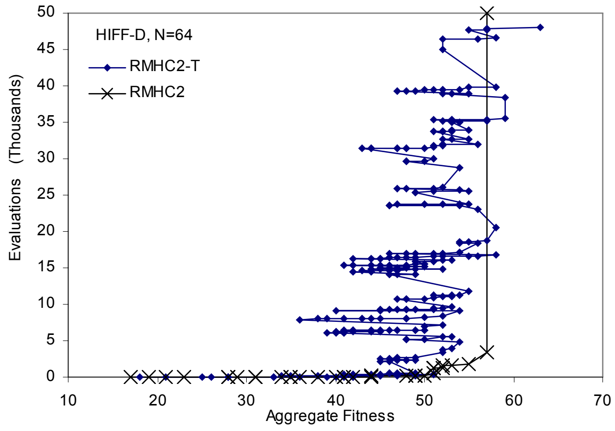


Figure 3A. RMHC2 and RMHC2-T sample runs.

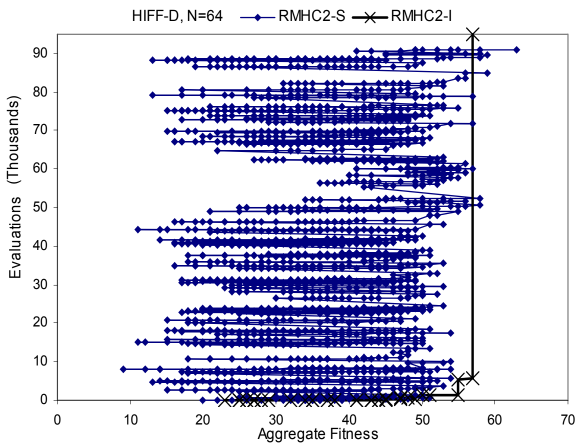


Figure 3B. RMHC2-S and RMHC2-I sample runs.

6. TRANSPOSITION IN A POPULATION

In section 1, we mentioned that with continuous HIFF and RMHC2, it was possible to evolve both genotypes and the sieve simultaneously and find a global optimum. Does this apply to discrete HIFF and RMHC2-T? Further, how would a recently transposed genotype survive selection in a population? In this section, we outline two population models to answer these questions. The first model is a multi-population model [2] and addresses both questions. This model is similar to the one used for continuous HIFF and RMHC2 [10] but the criterion for comparing demes is changed and the transposition operation is added. The second model is a transposition and crossover-only genetic algorithm with fitness-proportionate selection and generational replacement (CGA-T). CGA-T addresses the second question.

6.1 The multi-population model

In this model, subpopulations of genotypes are placed in a two-dimensional grid with periodic boundaries. Genotypes within a subpopulation do not interact with each other and subpopulations have minimal interaction with each other. There is no exchange of genetic material between genotypes. Each genotype has its own start and length values. Each cell in the grid has its own sieve, randomly generated at first. Genotypes evolve using RMHC2-T with the current sieve in their respective cells. After some number of generations, subpopulations compare their fitness with that of their neighbours. The fitness of a subpopulation is the average aggregate fitness of its genotypes. The most fit subpopulation replaces the least fit subpopulation in a Moore neighbourhood. During this synchronous extinction-recolonization event, the genotypes and their start and length values, and a variant of the sieve of the colonizer cell replaces those in the colonized cell. A sieve is varied by swapping two randomly chosen elements.

Table 7 lists the parameter values for our multi-population experiment. The balance between transposition rate and number of generations is important. After a transposition, a genotype needs time to realize its potential fitness. If this maturation time is too short or transposition occurs too frequently, the genotype will most likely pull the average fitness of its deme down and be in danger of extinction.

Table 7. Parameter values

Parameter	Multi-population	CGA
Maximum evaluations	3,000,000	500,000
Number of runs	30	30
Problem size, N	128	128
Population size	-	128
Population size per deme	3	-
Number of demes	4 x 4	-
Mutation rate (Pm)	0.03125	-
Crossover rate (Px)	-	1.0
Transposition (Pt)	500	2
Number of generations	500	-

The multi-population model found a global optimum 28 out of 30 times (93%) and used an average of 1,648,875 with a standard deviation of 451,230 and a median of 1,608,231 evaluations.

6.2 The CGA model

CGA is a single-point crossover-only genetic algorithm with fitness-proportionate selection and generational replacement. Selection in CGA is base on aggregate fitness. Each crossover event produces two offspring for the next generation. In CGA-T, separate T events may be triggered for either or both offspring, in which case the target offspring genotype is transposed. T is triggered in the same manner as in RMHC2-T (section 4).

Table 7 lists the parameter values for our CGA experiments on discrete HIFF. CGA-T was 100% successful and used an average of 29,237 evaluations with standard deviation 12,515 and median 25,072 while none of the CGA runs were successful. The variances of CGA populations quickly decreased to zero, while those of CGA-T remained above zero. Maintaining population diversity is one of the key components of a successful genetic algorithm application. Previous techniques used on the HIFF problem such as deterministic crowding required large populations in the order of thousands to succeed [15]. The population size for CGA-T is small in comparison.

From the experiments in this section, the answer to the first question is yes. The answer to the second question is cooperation. In the multi-population model, individuals in a deme help each other to survive a replacement event but do not exchange genetic material with one another. In the genetic algorithm model, individuals cooperate by exchanging their genetic material via single-point crossover.

7. RANDOM DRIFT

In sections 5 and 6 we have observed the beneficial effects of transposition on the discrete HIFF problem in individual and population base search. Does transposition also benefit the discrete HXOR problem? The experiments in this section are designed to answer this question.

We ran RMHC2-T with the parameter configuration in Table 6 (section 5) on discrete HXOR with 64 variables. None of these runs were successful. Figure 4 depicts one sample run. Some progress is made towards higher fitness values early in the run but further along, there is no discernable tilt to the right in Figure 4, unlike Figure 3A.

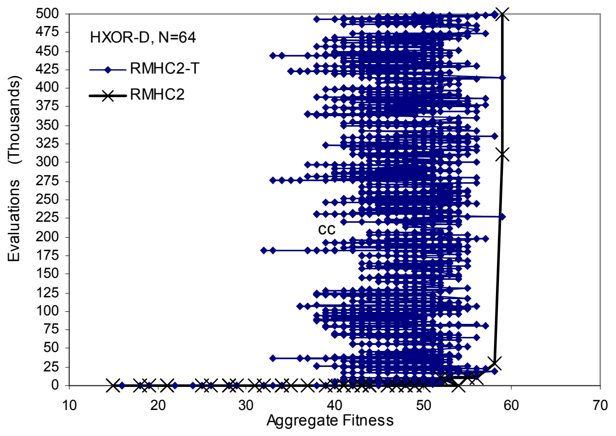


Figure 4. RMHC2 and RMHC2-T sample runs.

Results for CGA-T on discrete HXOR using the configuration in Table 7 were also dismal. Like HIFF, transposition helped a CGA population maintain diversity. But unlike HIFF, population diversity did not result in an optimal genotype being found.

We attribute these failures to the inability of HXOR to exploit random drift. In the HIFF problem, the arbitrary accumulation of ones (zeroes), will eventually lead to a global optimum. In the HXOR problem, it is not enough to accumulate equal number of ones and zeroes, which is easily done, but the ones and zeroes need to be placed in a non-trivial order to form a global optimum. The asymmetry between HIFF and HXOR due to difference in their optimal genotype strings has been witnessed elsewhere [17]. Appendix A gives an algorithm to produce optimal HXOR genotypes. This description is not as simple as append ones or zeroes to create optimal HIFF genotypes, and gives some inkling that genotypes satisfying the constraints of the HXOR problem are more complex than optimal HIFF genotypes.

8. A HYPOTHESIS

The experiments so far suggest the following hypothesis: that transposition is beneficial on problem-algorithm pairs with local optima and random drift. In sections 8.1 and 8.2, we test transposition on problem-algorithm pairs with random drift but with little to no local optima. In section 8.3, the hypothesis is tested on non-hierarchical decomposable problems.

8.1 RMHC2-T and HIFF-C

RMHC2 performs well on HIFF-C (Table 8) indicating an absence of problematic local optima for RMHC2 on HIFF-C. However, increasing the frequency of transposition slows down RMHC2, as predicted by our hypothesis. The probability that RMHC2-T with a Pt value of 50 is just as efficient as RMHC2 is less than 8% using the 1-tailed T-test with 58 degrees of freedom. The mutation rate used in this experiment is 0.0625, N =128 and the maximum number of evaluations is 10,000.

Table 8. Results of RMHC2 and RMHC2-T on HIFF-C.

	RMHC2	RMHC2-T(Pt=500)	RMHC2-T(Pt=50)
Times found	30/30	30/30	30/30
Avg. evaluations (std. dev.)	3096 (1148)	3326 (1226)	3532 (1206)

8.2 GIGA and HIFF-D

The Gene-Invariant Genetic Algorithm (GIGA) [3] preserves the initial distribution of bits throughout a run. So there is no loss of genetic diversity. In our GIGA, crossover is two-point, population size is 64, maximum evaluations is 100,000, the population is kept sorted and parent selection is by rotating through the population from fittest to least fit with the least fit end connected to the fittest end and mating every consecutive pair of genotypes. Each crossover produces a pair of offspring. Each offspring may be transposed. An offspring pair replaces their parents if the fitter offspring is fitter than the fitter parent (elitism).

Table 9. Results of GIGA and GIGA-T on HIFF-D, N=128.

	GIGA	GIGA-T (Pt=50)	GIGA-T (Pt=5)
Times found	30/30 (100%)	28/30 (93%)	10/30 (33 %)
Avg. evaluations (std. dev.) when found	59,602 (12,532)	66,140 (16,669)	73,981 (16,507)

From the results in Table 9, there are no problematic local optima for our GIGA on HIFF-D. When transposition is added, performance of GIGA deteriorates, as predicted by our hypothesis. This is because if only one offspring in a pair is transposed and the offspring pair is fitter than the parent pair, the initial distribution of bits will be disturbed. Hence the GA is not Gene-Invariant anymore and problem with diversity maintenance creeps in.

8.3 Other Test Problems

We performed other experiments similar to the ones in this paper on two Royal Road functions and on the One-Max problem. The Royal Road functions are R_1 [5] where no bonus is given for combination of schema, and R_X where we substitute the all-ones optimal string with a HXOR optimal string. There is no epistasis and no random drift effect in R_X individuals. Unlike HIFF and HXOR, transposition does not change the actual or potential fitness of a One-Max genotype. But transposition most likely reduces the actual fitness of a Royal Road genotype. This reduction means that more bits are free to mutate in any direction. Thus transposition also reduces potential Royal Road genotype fitness.

As expected, transposition was not beneficial to RMHC on the Royal Road functions. Transposition was beneficial to CGA on R_1, but not on R_X. Also as expected, transposition helped CGA on One-Max. However, transposition did not impair the performance of RMHC on One-Max. Thus it is possible for

Table 10. When does transposition help an evolutionary algorithm? Summary of findings.

Problem	Random drift	Local Optima for RMHC*	RMHC*	Local Optima for CGA	CGA	Local Optima For GIGA	GIGA
HIFF-D	Yes	Yes	Yes	Yes	Yes	No	No
HXOR-D	No	Yes	No	Yes	No	-	-
HIFF-C	Yes	No	No	-	-	-	-
Royal-Road (R_1)	Yes	No	No	Yes	Yes	-	-
Royal-Road (R_X)	No	No	No	Yes	No	-	-
One-Max	Yes	No	Neutral	Yes	Yes	-	-

transposition to have a neutral effect. In general, the hypothesis is supported by these additional experiments.

9. CONCLUSION

Using an operation found in the natural genome, we have shown how a hill climber (RMHC2-T) and a genetic algorithm (CGA-T) can escape evolutionary dead ends. Table 10 summarizes the results of our experiments. Transposition had a positive effect on both individual and population base search techniques in the presence of random drift and epistasis. Epistasis creates local optima for RMHC individuals. The concept of epistasis can be extended to interaction between individuals, as is done in the NKC model [9]. Although the epistatic relationship between individuals in CGA is transient, local optima for a population can be created through inter-individual interactions of selection, recombination and replacement. These results support the notion that epistasis and genome plasticity have long-term evolutionary benefits under the right conditions.

10. ACKNOWLEDGEMENTS

This work is supported by the Faculty of Engineering and Computer Science, Concordia University and NSERC. Thanks to Peter Grogono and the anonymous reviewers for their comments.

11. REFERENCES

1 Alberts, B. (Editor) *Molecular Biology of the Cell*. Fourth edition, 2002. Garland Press. Available online at NCBI.
2 Altenberg, L. Evolvability suppression to stabilize far-sighted adaptations. *Artificial Life*, vol. 11, 2005, pages 427- 443. The MIT Press.
3 Culberson, J. C. Mutation-crossover isomorphisms and the construction of discriminating functions. *Evolutionary Computation* vol. 2, 1994, pages 279-311.
4 de Jong, E. D., Thierens, D. and Watson, R. A. Hierarchical genetic algorithms. In X. Yao, et al. (Editors) *Parallel Problem Solving from Nature* (PPSN) vol. VIII, 2004, pages 232 – 241, Springer, Berlin.
5 Forrest, S. and Mitchell, M. Relative building-block fitness and the building block hypothesis. In D. Whitley (editor) *Foundations of Genetic Algorithms* (FOGA) vol. 2, 1993, Morgan Kaufmann.
6 Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989, Addison-Wesley.
7 Holland, J. H. *Adaptation in Natural and Artificial Systems.* MIT Press edition, 1992.
8 Jones, T. and Forrest, S. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. Eshelman (Editor), 6th *International Conference on Genetic Algorithms* (ICGA), 1995, pp. 184 – 192, Morgan Kaufmann.
9 Kauffman, S. A. *The Origins of Order*. Oxford University Press, 1993.
10 Khor, S. Rethinking the adaptive capability of accretive evolution on hierarchically consistent problems. *IEEE Symposium on Artificial Life*, 2007.
11 Knowles, J. D., Watson, R. A. and Corne, D. W. Reducing local optima in single-objective problems by multi-objectivization. In *Conference on Evolutionary Multi-criterion Optimization*, 2001, Lawrence Erlbaum Associates.
12 Pelikan, M. and Goldberg, D. E. Escaping hierarchical traps with competent genetic algorithms. In L.E Spector, et al. (editors), *Genetic and Evolutionary Computation Conference*, 2001, pages 511 – 518, Morgan Kaufmann.
13 Simon, H. A. *The Sciences of the Artificial*. 1969, The MIT Press.
14 Walker, J. A. and Miller, J. F. Embedded Cartesian Genetic Programming and the Lawnmover and Hierarchical-if-and-only-if problems. In M. Keijzer et al. (Editors) *Genetic and Evolutionary Computation Conference*, 2006.
15 Watson, R. A. Analysis of recombinative algorithms on a non-separable building-block problem. In W. N. Martin and W. M. Spears (Editors) *Foundations of Genetic Algorithms* (FOGA), vol. 6, 2001, pages 69-90. Morgan Kaufman.
16 Watson, R. A. and Pollack, J. B. A computational model of symbiotic composition in evolutionary transitions. *BioSystems*, vol. 69, 2003, pages 187 – 209, Elsevier.
17 Watson, R. A. and Pollack, J. B. Hierarchically consistent test problems for genetic algorithms. In P. J. Angeline, et al. (Editors), *Congress on Evolutionary Computation* (CEC), 1999, pages 1406-1413. IEEE Press.
18 Watson, R. A. *Compositional Evolution – The impact of sex, symbiosis and modularity on the gradualist framework of evolution.* MIT Press, 2006.
19 Watson, R. A., Hornby, G. S. and Pollack, J. B.. Modeling building-block interdependency. In A.E. Eiben, et al. (Editors) *Parallel Problem Solving from Nature* (PPSN) 1998, vol. V, pages 97-106, Springer, Berlin.

12. APPENDIX A

An optimal HXOR genotype of length $N = 2^n$, $n > 0$, can be produced with the algorithm in Figure 5. A start_bit of 0 and num_levels of 2 produces optimal_genotype 1001.

```
PROCEDURE: createHXOROptimum
INPUT: num_levels, sb      //sb is start_bit, can be either 0 or 1
OUTPUT: optimal_genotype
BEGIN
    to empty optimal_genotype, append sb, then append (1-sb)
    FOR EACH i from 1 to num_levels-1
        FOR EACH element e in optimal_genotype
            IF (e is 0)   insert 1 to optimal_genotype before e
            ELSE    insert 0 to optimal_genotype before e
        END FOR
    END FOR
END
```

Figure 5. Pseudo-code to create optimal HXOR genotypes.