Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)

Rethinking the Adaptive Capability of Accretive Evolution on Hierarchically Consistent Problems

Susan Khor
Concordia University, Montreal, CANADA
Email: slc_khor@cse.concordia.ca

***Abstract*-We consider how accretive evolution is able to evolve optimal solutions to hierarchical decomposable non-separable problems epitomized by hierarchically consistent test problems. We find that this feat is not as improbable as previously thought if a suitable phenotype which reflects the levels in the hierarchy is used as the object of a selection scheme which applies level directed selection pressure. An ideal selection scheme is first described and tested experimentally, followed by a meta-population model to evolve the ideal selection scheme. Experiments with the model revealed that evolution of the ideal selection scheme is not a pre-requisite to evolving optimal solutions. Optimal solutions were found even when the ideal selection scheme was not.**

No recombination of partial solutions is used in this paper. The findings of this paper reopen the question: what type of hierarchical structure is difficult to evolve through random mutation and selection.

I. Introduction

There has been considerable debate in the field of evolutionary computation about the utility of the recombination operator [1, 2]. These debates often take the form of a comparison between hill-climbing algorithms and genetic algorithms [3]. In his doctoral dissertation, Watson draws an analogous debate in natural evolution between step-by-step evolution or gradualism and evolution by jumps or compositional evolution, from an algorithmic point of view [4]. The centerpiece of his dissertation is the class of hierarchically consistent (HC) problems [5].

The class of hierarchically consistent problems is used to discriminate between the adaptive capacity of accretive evolution and of compositional evolution: "Certain kinds of complex systems, considered unevolvable under normal accretive change, are, in principle and under certain circumstances, easily evolvable under compositional change." [4, p.1] The terms accretive evolution and compositional evolution are explained further on in this section. Section II explains what HC problems are.

In this paper, we offer a way for finding solutions to HC problems with accretive evolution. If our proposed method succeeds, then the case for evolution – that the blind process of evolution can produce complex forms – is further strengthened. Because now there are two ways of evolving complex structures: accretive and compositional. However, this does blur the distinction between the adaptive capabilities of evolutionary algorithms which use recombination from those which do not on problems with hierarchical structure.

Our experiments in section V, confirm that if a suitable phenotype is introduced and if an appropriate selection scheme is used, evolution of optima for hierarchically consistent problems under accretive evolution is not as improbable as portrayed in [4]. Section VI presents a multi-population model to evolve the appropriate selection scheme. Experiments with this model produced unexpected results.

Both accretive evolution and compositional evolution use selective accumulation of genetic changes to evolve units. Accretive evolution and compositional evolution are differentiated on the basis of the type and source of variation: random in the accretive case; and biased and from other existing units in the compositional case. The change or new material introduced to the unit under accretive evolution "have not been pre-adapted elsewhere as a set" and is, in this sense, random; while the change or new material under compositional evolution "have been semi-independently pre-adapted in parallel in different lineages" [4, p.4].

Examples of compositional mechanisms in nature given in [4] include sexual recombination, horizontal gene transfer and endosymbiosis. Examples of artificial compositional mechanisms include genetic algorithms (GA) with tight genetic linkage and the Symbiotic Evolutionary Adaptation Model (SEAM) [6]. There are no random mutations in SEAM [4, p.235]. Examples for artificial accretive mechanisms are genetic algorithms without tight genetic linkage and the Random Mutation Hill Climbing (RMHC) algorithm [1]. The RMHC algorithm is defined in section III.

Reference [4] considers optima to HC problems *unevolvable under normal accretive change* because (i) accretive evolution as defined in [4] cannot manipulate modules effectively, (ii) the adaptive landscape of HC problems have wide fitness-saddles which grow wider as the problem size increases, (iii) there is a high-degree of ruggedness and hence many local optima in the adaptive landscape and (iv) the optimal solutions are irreducibly complex for accretive evolution.

When looking at the problem of evolving complex solutions in artificial systems, one could consider whether random mutation is capable of producing variations for selection to act upon to produce the complex solution. Alternatively, one could consider whether the selection scheme is capable of shaping the complex solution via evolution. In other words, is the selection scheme up to the task? Watson uses the first approach when he uses features of the adaptive landscape for a mutation operator to explain why accretive mechanisms have difficulty with HC problems [4]. We take the second approach and propose a selection scheme that is up to the task for HC problems.

The selection scheme we propose is biased towards conserving the adaptation of some features of a phenotype, possibly to the detriment of other features in the phenotype

and the genotype's aggregate fitness. Section IV elaborates on our approach. Note that compositional evolution also manipulates its selection scheme; it does so to maintain population diversity, which is crucial to the success of compositional evolution on HC problems [4, p.20].

The work presented in this paper is novel because to the best of our knowledge to date, there has been no other attempt to solve HC problems with RMHC. We have not found any approach that splits up the fitness of a genotype into a sequence of per-level fitness values, and compares per-level fitness values in its selection scheme. Other approaches to HC problems include hierarchical GA, competent GA, dimensional reduction, multi-objective simulated annealing, learning, neutral networks and development [7, 8, 9, 10, 11, 12]. Multi-level selection has been the subject of intense debate in biology [20] and has also received attention in artificial life [21].

The conclusion we reach in this paper is that accretive evolution can evolve solutions to HC problems provided (i) the genotype-phenotype map is known, and (ii) suitable selection pressure is applied. This conclusion is supported by experimental results summarized in Section V.

II. HIERARCHICALLY CONSISTENT PROBLEMS

A. Background

A problem is *decomposable* if its variables can be partitioned into identifiable subsets, in some intuitive way. If a problem is decomposable into a hierarchy of sub-problems or modules, it is hierarchical. If the difficulty of solving a sub-problem is the same as the difficulty of solving the whole problem, the problem is *consistent* [5].

The subsets of variables or modules in hierarchically consistent problems are interdependent. Two modules are interdependent if the optimal solution for one module depends on the solution of the other module, and vice versa. Two criteria must be satisfied to possess *modular interdependency*:
(i) for each module, the number of configurations that can lead to an optimal solution in a given context (the other modules) must be less than the total number of possible configurations but more than one; and
(ii) the ratio of inter-module dependency strength to intra-module dependency strength must be neither too high nor too low: "... the strength of inter-module dependencies should be low enough that intra-module dependencies create local optima" [4, p.139].

The implication of this is that finding optimal solutions to lower level modules does not guarantee optimality to modules higher up the hierarchy. Concepts introduced in this section are treated more thoroughly in [4, 7, 13].

Hierarchically consistent (HC) problems describe a class of test problems, which include Hierarchical If-And-Only-If (HIFF), Hierarchical Exclusive-Or (HXOR) and their shuffled versions, SHIFF and SHXOR [4, 14]. The most common version of HC problems has a block size of 1 and a binary alphabet. Reference [14] explored the impact of bias, different block sizes and different alphabet sizes on search difficulty. HIFF and HXOR are exact complements of each other thereby making the use of either function alone sufficient for our purpose. This is not the case with their biased versions [14], but we need not worry about this here.

B. Algorithm

In this paper, the genotypes are bit strings of length $N = 2^n$. We use the HIFF function defined on page 139 in [4] but modified to calculate fitness by levels.

A genotype has $\log_2 N$ levels, and $N/2^{\lambda}$ modules at level λ, with $\lambda = 1 \ldots n$. According to our algorithm, the aggregate fitness value for an optimal genotype is N-1. An optimal genotype is one with all zeroes or all ones.

The pseudo-code for our algorithm is listed in Fig. 1. Examples are worked out in Tables 1 and 2. The algorithm comprises three functions: (i) calculateHIFFLevelFitness, (ii) count_ones, and (iii) iff_function.

The calculateHIFFLevelFitness function returns the fitness for a given level, also known as *per-level fitness*. Per-level fitness is the sum of the fitness of every module at that level. Fitness of a module is calculated using the remaining two functions. The count_ones function returns the number of ones in a given bit string divided by the length of the bit string. The iff_function is the continuous version of discrete logical iff. It is $(p \times q) + (1 - p) \times (1 - q)$.

```
PROCEDURE: calculateHIFFLevelFitness
INPUT: λ, N
OUTPUT: level_fitness
BEGIN
    module_size ← 2^λ
    num_modules ← N / module_size
    FOR each module i at level λ
        IF (module_size = 2)
            p ← first bit of module i
            q ← second bit of module i
            level_fitness ← level_fitness + iff_function(p, q)
        ELSE
            bit_string1 ← first half of module i
            bit_string2 ← second half of module i
            level_fitness ← level_fitness +
                iff_function(count_ones(bit_string1),
                             count_ones(bit_string2))
    END FOR
END
```

Fig. 1 HIFF function defined on page 139 in [4] modified to calculate fitness by levels.

TABLE 1
AN EXAMPLE OF HIFF EVALUATION

Level	Number of Modules	Genotype 170								Per level fitness
		1	0	1	0	1	0	1	0	
(lowest) 1	4	0.0		0.0		0.0		0.0		0.0
2	2	0.5				0.5				1.0
(highest) 3	1	0.5								0.5
									Phenotype	⟨ 0.5, 1.0, 0.0 ⟩
									Aggregate fitness	1.5

In Table 1, modules at level 1 all have 0.0 fitness because iff_function(1, 0) yields 0.0. At level 2, there are 2 modules of 4 bits each. Count_ones("10") yields 1/2 = 0.5 since there is one '1' bit and the length of the bit string is 2.

iff_function(0.5, 0.5) yields 0.5 so fitness per module at level 2 is 0.5 and fitness for level 2 is $2 \times 0.5 = 1.0$.

TABLE 2
ANOTHER EXAMPLE OF HIFF EVALUATION

Level	Number of Modules	Genotype 111	Per level fitness
		0 1 1 0 1 1 1 1	
(lowest) 1	4	0.0 0.0 1.0 1.0	2.0
2	2	0.5 1.0	1.5
(highest) 3	1	0.5	0.5
		Phenotype	⟨ 0.5, 1.5, 2.0 ⟩
		Aggregate fitness	4.0

III. RMHC

Reference [4] used the random mutation hill climbing (RMHC) algorithm as the mechanism for accretive evolution. RMHC was introduced in [1]. The algorithm below follows [4]:

1. Create a fully-specified random bit string.
2. Choose *k* bits at random to mutate. *k* is determined by a given mutation rate. Mutation is by randomly assigning a 0 or a 1 to the bit.
3. If the mutation results in an equally fit or fitter offspring string, keep the mutation. Otherwise, ignore the mutation.
4. If the optimal string has not been found and the pre-determined number of fitness function evaluations has not been performed, go to step 2. Otherwise return the current bit string as the solution.

RMHC does not use a population, so there is no population diversity to maintain.

IV. THE PROPOSED METHOD

A. Phenotypes

The sequence of per level fitness values for a genotype forms the phenotype for the genotype, with the highest level ($\lambda = n$) fitness situated at the leftmost position of the phenotype sequence, followed by the second highest ($\lambda = n-1$) level fitness and so on. Tables 1 and 2 illustrate this.

The optimal HIFF phenotype for a problem of size $N = 2^n$ is $\langle 2^0, 2^1, \ldots, 2^i, 2^{i+1}, \ldots 2^{n-1} \rangle$. Work to calculate per level fitness is already ordinarily expanded to calculate genotype fitness, so creating phenotypes does not involve additional computation, only additional space.

Each element of a phenotype is a *feature*. The phenotype in Table 1 has 3 features. The fitness of its first feature is 0.5 out of a possible 1.0.

B. The selection scheme and the "ideal" sieve

Selection in RMHC (step3, section III) selects the genotype with the higher aggregate fitness value. We do not compare aggregate fitness values. Instead, per-level fitness values between a parent and its offspring are compared.

Our selection scheme uses a *sieve* to determine the order in which per-level fitness values of two phenotypes are compared. A sieve can be viewed as an adaptation strategy a genotype employs unconsciously and which it has no control over. Alternatively, a sieve could be seen as environmental influence, differentiating the survival importance of the features of a phenotype. 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.

Suppose the sieve for a HIFF problem with size N = 8 is ⟨2, 1, 3⟩ and the two competing phenotypes are $p_1 = \langle a, b, c \rangle$ and $p_2 = \langle x, y, z \rangle$. Then under our selection scheme, reals *b* and *y* are compared first, since they are the fitness values for the second level feature, followed by *c* and *z*, and finally *a* and *x*.

The genotype that improves any per level fitness value earlier in a comparison is selected, regardless of what happens to the fitness values of the other features or to the aggregate fitness. So going back to the previous example, if ($b > y$) or ($b = y$ and $c > z$) or ($b = y$ and $c = z$ and $a > x$), then p_1 is more fit and its genotype is selected over the competitor. If all three per-level fitness values are pairwise equal, then the genotype for p_2 is selected by default.

The "ideal" selection scheme uses the "ideal" sieve, which says to inspect features of phenotypes in level descending order, i.e. ⟨*n, n-1, n-2, ..., 1*⟩. According to this order, the adaptation of the largest module has priority, followed by its sub-modules, sub-sub modules, and recursively down to the smallest modules. Since adaptation of smaller modules in the lower levels does not guarantee concurrent adaptation of larger modules in the higher levels, features of a phenotype can be seen as selfish and competing with each other. The "ideal" sieve is ideal for the HIFF problem because it does not allow any lower level feature to gain the upper hand in this competition.

Table 3 demonstrates the "ideal" selection scheme. Genotype number 2 is selected over genotype number 3 even though its aggregate fitness (*f*) is lower because it is fitter than genotype number 3 at a higher level. For this same reason, genotype number 182 is selected over genotype number 179.

C. RMHC2

RMHC2 incorporates phenotypes and the "ideal" selection scheme into RMHC. It is the algorithm we use to evolve an optimal solution for HIFF problems. RMHC2 uses k-bit macro-mutation, where *k* is an integer within [1, K], randomly

TABLE 3
EXAMPLE OF THE IDEAL SELECTION SCHEME AT WORK.
THE IDEAL SELECTION SCHEME USES THE IDEAL SIEVE FOR THIS PROBLEM WHICH IS ⟨3, 2, 1⟩.

gnum	Genotype	HIFF Phenotype λ=3	λ=2	λ=1	*f*	Selection
2 (parent)	0000 0010	**0.75**	1.5	3.0	5.25	2
3 (offspring)	0000 0011	**0.50**	1.0	4.0	5.50	
179 (parent)	1011 0011	0.5	**0.5**	3.0	4.0	182
182 (offspring)	1011 0110	0.5	**1.0**	1.0	2.5	

chosen before each mutation event; and $K = P_m \times N$, P_m is a fixed predefined mutation rate within (0, 1) and N is the problem size or length of the genotype. A macro-mutation operator mutates a contiguous section of a genotype within two randomly picked loci [15] and is the type of mutation operator with the best chance of succeeding on HC problems [4]. The RMHC2 algorithm:

1. Start with a fully specified (every loci has a value of either 1 or 0) genotype.
2. Make a copy of the parent genotype and mutate the copy to produce the offspring genotype. Starting at a random locus *l*, mutate *k* consecutive bits, wrapping around to the start of the genotype if the end of the genotype is reached. Mutation is by random assignment of a 0 or a 1.
3. If the offspring phenotype is selected, replace the parent genotype with the offspring genotype. Otherwise, discard the offspring genotype.
4. If the optimal string has not been found and the pre-determined number of function evaluations has not been performed, go to step 2. Otherwise return the current genotype as the solution.

V. Experiments and Results

A. Experimental setup

Unless otherwise stated, the experiments use the parameter values detailed in Table 4. These values are taken from [4] with the exception of maximum evaluations per run. After a few preliminary tests, we found running to a maximum of 3,000,000 evaluations, which is the figure used in [4], to be overkill for our purpose.

TABLE 4
DEFAULT PARAMETER VALUES

Parameter	Value
Number of runs per experiment	30
Maximum evaluations per run	1,000,000
Problem size, N	128
Mutation rate, P_m	0.0625

A different random number seed is used in each run in an experiment, but all experiments used the same series of random number seeds for their runs. The number of times an optimum is found is used to measure the performance of the various algorithms under test.

B. Experiments with the "ideal" selection scheme

To test the performance RMHC2, we compared it with RMHC1. RMHC1 uses variable k-bit macro-mutation and compares aggregate fitness to decide whether to accept a changed genotype. The algorithm for RMHC1 is therefore the same as the algorithm for RMHC2 except step 3 is replaced with step 3 of RMHC.

Table 5 compares results obtained from this experiment with relevant previous results. That no RMHC1 run succeeded confirms that the difficulty of the HIFF problem under normal accretive change is preserved in our implementation of the problem. 100% of RMHC2 runs succeeded on the HIFF problem, even when the problem size is doubled to 256. This is a significant improvement over RMHC1. Therefore our first hypothesis, that using a suitable phenotype which reflects the levels in the hierarchy, coupled with a selection scheme which applies level directed selection pressure on the phenotype, significantly improves the performance of accretive evolution (RMHC), is confirmed.

TABLE 5
NUMBER OF TIMES AN OPTIMUM IS FOUND. RMHC2 STOPS WHEN IT FINDS ONE OF THE OPTIMA.

Method	Character	Times found	Evaluations
SEAM [6]	compositional	30/30 (100 %)	-
RMHC [4]	compare aggregate fitness fixed *k* random mutation	0/30 (0 %)	-
RMHC1	compare aggregate fitness variable *k* macro-mutation	0/30 (0 %)	-
RMHC2	compare phenotypes, variable *k* macro-mutation	30/30 (100 %)	2245 (avg.) 660 (std. dev.)
RMHC2	compare phenotypes, variable *k* macro-mutation N=256	30/30 (100 %)	8084 (avg.) 2325 (std. dev.)

Fig. 2 compares the progress made by a RMHC1 run with a RMHC2 run. In the RMHC1 run, lower level modules quickly make progress, but this progress does not lead to the evolution of an optimal solution. In the RMHC2 run, progress is made at all levels and an optimal solution is found.

RMHC2 is able to evolve optimal solutions for HIFF because it prevents devolution of higher level features by selecting against mutations which attempt to do so even when this means preventing adaptation or causing regression in lower level features. In a sense, RMHC2 imposes horizontal (level view) separability on a vertically (module view) non-separable problem. From the vantage point of the highest level feature, a HIFF problem becomes a hill-climbing effort. And since an optimal solution for the highest level feature or largest module is also optimal for all lower level modules, evolving an optimal solution is not an improbable pole vault up the steep side of a mountain with the "ideal" selection scheme.

Table 6 lists correlation values between fitness values at different levels and hamming distances to the closest global optimum on the HIFF problem for problem size 8 and 16. Fitness Distance Correlation (FDC) was introduced by Jones and Forrest [18], as a measure of search difficulty. Intuitively, FDC measures whether fitness increases as distance to a global optimum decreases. A search problem with a negative FDC value is categorized as straightforward or not misleading. Search difficulty for straightforward problems increases as FDC approaches 0. There are known problems with this summary statistic [19] and its application does not appear to depend on the search operator which is worrying. Nevertheless, FDC has been shown to reflect search difficulty on a number of known problems [18]. We use FDC as a first approximation of the search difficulty at different levels in a HIFF problem which may account for the success of the "ideal" selection scheme.

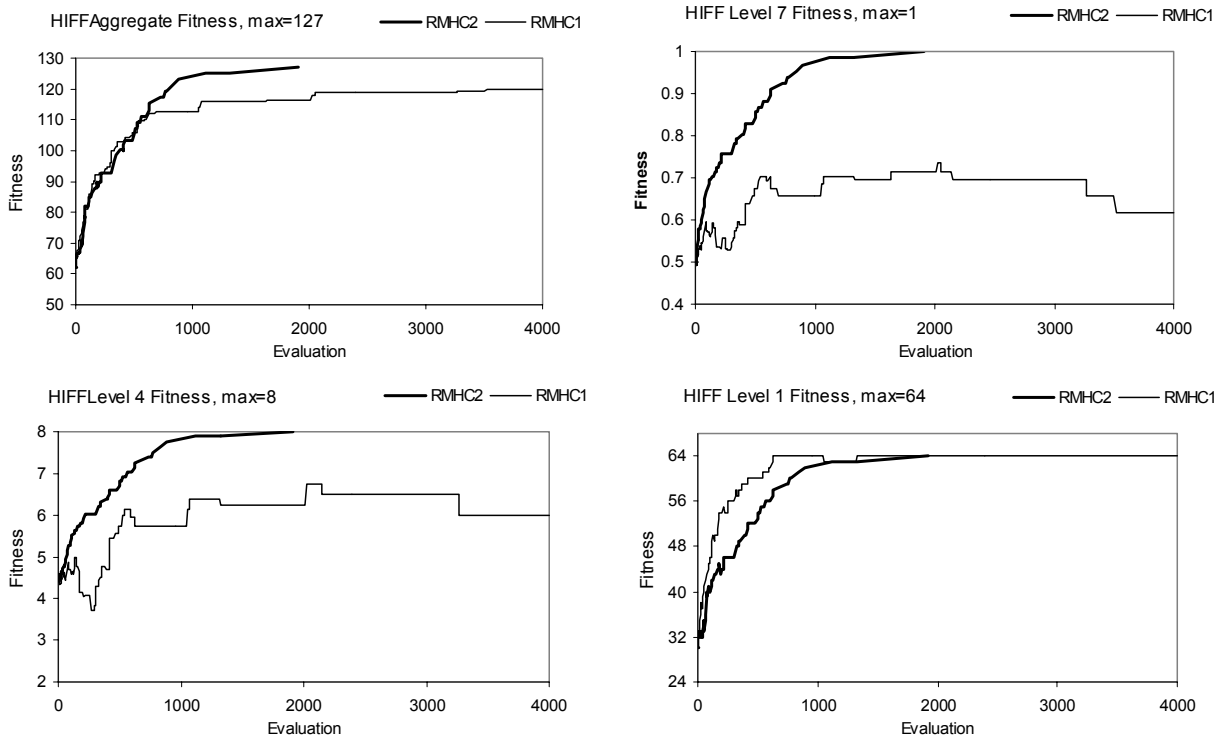

Fig. 2 N=128. The selfish interests of modules in the lower level dominate the RMHC1 run, and prevent the formation of an optimal solution. On the other hand, modules at different levels in the RMHC2 run evolve in concert and an optimal solution is formed for all levels.

The FDC values in Table 6 reveals that search difficulty on the HIFF problem is straightforward at all levels but not uniform across levels. Further, there is clear indication that search is least difficult at the highest level and becomes progressively more difficult at lower levels.

TABLE 6

Level	FDC	
	N=8	N=16
4	-	-0.6770
3	-0.6972	-0.4787
2	-0.4930	-0.3385
1	-0.3486	-0.2394
Aggregate	-0.5712	-0.4199

VI. Evolving the Sieve

A. The meta-population model

In this section, we outline our model to evolve the sieve and optimal solution concurrently. Meta-population [16] refers to a model where a population is divided into demes with minimal interaction between demes so that demes may diverge in their characteristics, and there are frequent extinction and re-colonization of demes.

A meta-population model was used to study the evolution of far-sighted traits through the suppression of short-sighted or evolutionary pathological traits [17]. Short-sighted traits are those which confer short-term benefits but lead the population to extinction in the long term.

In the HIFF problem, higher level features may be viewed as far-sighted traits and lower level features, short-sighted ones since adapting smaller modules, with no regard for the adaptation of larger modules, results in sub-optimal solutions. However, rather than suppressing traits, our task here is to rank traits.

Our model splits a population of genotypes into minimally interacting demes or subpopulations. Every subpopulation comprises one or more genotypes, occupies a cell in a two dimensional lattice with periodic boundaries, remembers the number of changes made to its genotypes and is assigned a randomly generated sieve at the start of the experiment.

There are two types of selection in our model: long-term and short-term. A *short-term selection* event happens after every mutation event and is specific to a genotype[1] within a subpopulation. Short-term selection evaluates genotypes in a subpopulation on the basis of the subpopulation's current sieve. If a mutation is accepted, that is an offspring genotype is selected over its parent, and the offspring is at least one Hamming distance away from its parent, the number of changes for the subpopulation is increased by one. Mutation in this model is k-bit random mutation, and not macro-mutation; that is the k bits for mutation are selected randomly with replacement.

[1] Genotypes within a subpopulation do not interact with each other.

A mutation event followed by a short-term selection event is an update event. A *long-term selection* event is triggered after a specified number of updates to the whole population have occurred. For example, in a 4 cell lattice with 2 genotypes per cell, each update is 8 evaluations. Suppose long-term selection is configured to occur every 10 generations, then a long-term selection event will be triggered after every 80 updates.

It is only during a long-term selection event that demes interact. In this interaction, a subpopulation compares its number of changes with that of its 8 closest neighbors (Moore neighborhood). A subpopulation is marked for extinction if its number of changes is strictly less than that of all its neighbors[2]. When all subpopulations have interacted with their neighbors, extinction and colonization of subpopulations begin.

A decimated subpopulation is replaced or colonized by a neighboring subpopulation with the highest number of changes[3]. A copy of every colonizer genotype is placed in the colonized subpopulation, so at the end of the replacement event both colonized and colonizer subpopulations have the same set of genotypes. The colonized subpopulation inherits its colonizer's number of changes value and a mutated copy of its colonizer's sieve. To avoid generation of invalid sieves, sieves are mutated by swapping two randomly picked elements of a sieve.

B. Experiments and results

Runs in this section use the parameter values set out in Tables 4 and 7.

TABLE 7
DEFAULT VALUES FOR ADDITIONAL PARAMETERS

Parameter	Value
Lattice dimensions	4×4
Subpopulation size	5
Number of generations	100

The results of the multi-population runs are summarized in Table 9. In spite of not "knowing" and not finding the "ideal" sieve, and not using macro-mutation, the multi-population runs did just as well as RMHC2. They had a 100% success rate on HIFF.

Fig. 3 traces the progress of a successful genotype in a multi-population run. Table 8 lists significant events leading up the optimal solution for this run.

The genotype[4] in Fig. 3 and Table 8 spent about two thirds of its time in cells with sieves that prioritized progress at level 7 (the highest level) and the rest of its time in cells with sieves that prioritized adaptation at level 1 (the lowest level). Although the "ideal" sieve was not found in this run, the pattern of emphasizing progress at the highest level early on in evolution appears sufficient to produce the effect of the "ideal" sieve, as evidenced from this example. It is important to note that the "ideal" sieve effect emerged without design.

We did not expect the pattern exhibited in Table 8. Instead, we expected cells with sieves that emphasized progress at lower levels to be more active early on in a run since smaller modules require fewer consecutive bits to agree and would be easier to form with random mutation. Further investigation is necessary to ascertain the frequency of this phenomenon and to identify the contributing factors.

TABLE 8
ABBREVIATED HISTORY OF EVENTS LEADING UP TO AN OPTIMAL SOLUTION

Generation	Event
0	Cell 14 Sieve at cell 14: 7, 6, 4, 3, 5, 1, 2
300 (after 24,000 evaluations)	Cell 14 colonizes cell 1 Sieve at cell 1: 7, 6, 1, 3, 5, 4, 2
700	Cell 1 colonizes cell 12 Sieve at cell 12: 1, 6, 7, 3, 5, 4, 2
800 (after 64,000 evaluations)	Cell 12 colonizes cell 8 Sieve at cell 8: 1, 2, 7, 3, 5, 4, 6
900	Cell 8 colonizes cell 7 Sieve at cell 7: 1, 6, 7, 3, 5, 4, 2 Optimum found in cell 7 after 87,398 evaluations (after 87398 / (4×4×5×100), about 1092, generations).

C. Is extinction-recolonization necessary?

Since evolution of the "ideal" sieve is not necessary, is our multi-population approach unparsimonious? To investigate this, we performed multi-population runs without long-term selection. In this set of runs, there are no extinction-recolonization events, hence no interaction between demes and no changes of sieves. This model is akin to evolving genotypes independently and in parallel with different sieves, and for this reason is called the *parallel-model*.

Results of the experiment with the parallel-model are summarized in Table 9. Fig. 3 depicts the progress made at four levels by a successful genotype in a parallel-model run. The parallel-model runs were just as successful at finding an optimal solution as the multi-population model runs, achieving 100% success rate. Further, the parallel-model runs did not take significantly more evaluations than the multi-population runs to evolve an optimum. Base on these results, our multi-population model is unnecessary.

TABLE 9
NUMBER OF TIMES AN OPTIMUM IS FOUND FOR HIFF, N=128.

Model	HIFF (P_m = 0.0625) k-bit random mutation	
	Times Found	Evaluations
Multi-population	30/30 (100 %)	77,076 (avg.) 16,723 (std. dev.)
Parallel	30/30 (100 %)	82,033 (avg.) 16,845 (std. dev.)

D. What makes a successful sieve for HIFF?

Fig. 4 compares the distribution of features in successful sieves, i.e. sieves which produced an optimum in their run; against the backdrop of all sieves that were randomly generated in 10 randomly chosen runs in the parallel-model

[2] This is un-evolutionary. A more likely scenario is for an active subpopulation, one with more changes, to seek out less active subpopulations and colonized them. Further, this colonization needs some kind of pay-off to the colonizer, perhaps space.

[3] This policy, along with others, may be modified after more detailed studies.

[4] Strictly speaking, copies of the genotype. Genotypes do not move from cell to cell, but copies of them are made.

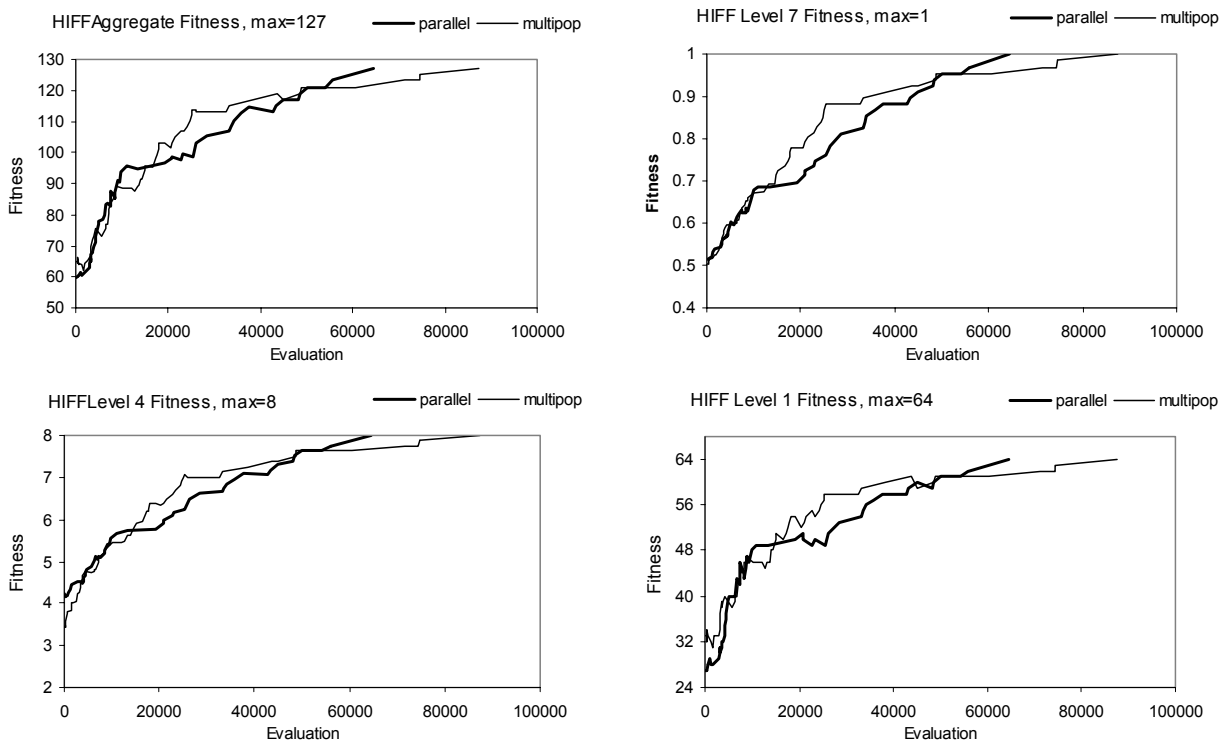

Fig. 3 N=128. In general, progress is made at all levels. Some regression occurs, indicated by a dip in the graph which reflects the non-altruistic nature of our selection scheme, progress in a feature can come at a cost to other features in the same phenotype. The multipop run goes through a series of sieves (Table 7). The parallel run uses only one sieve ⟨5, 2, 4, 1, 7, 3, 6⟩.

experiment. We will use these 160 random sieves as our reference set.

Only 3 of the 6 features were found at the first position of the successful sieves, and these features are all higher level features: 5, 6 and 7. In relation to our reference set, the probability of generating a set of 10 sieves with only features 5, 6 or 7 at the first position of the sieves is 0/10 = 0.00 and the probability of generating a set of 5 sieves with only features 5, 6 or 7 at the first position of the sieves is 1/20 = 0.05. Thus the distribution of features in the set of successful sieves is significantly different from that in the set of random sieves. Further, 9/10 sample runs had more than one sieve with features 5, 6 or 7 in the first position. Further analysis into the relative positions of features in a successful sieve could be useful.

So while our experiments confirm that an "ideal" sieve is not necessary, they also confirm that it is not the case that any sieve will be successful. From the distribution of features in successful sieves, the basic idea of prioritizing high level features over lower level ones is still operating in the parallel-model.

This conclusion is further supported by the following experiment where the first halves of the initial sieves are restricted to lower level features. When this initial condition is imposed, the multi-population runs outperformed the parallel-model runs significantly. 96% of the multi-population runs succeeded but only 36% of the parallel-model runs did. Due to the smaller problem size, N=64, it was possible for a few parallel-model runs to succeed. The multi-population runs and the parallel-model runs start with the same initial conditions as the same series of random seed numbers is used (section V).

VII. Conclusion

In this paper, we have described how to increase the adaptive capability of a random mutation hill climbing algorithm on a class of test problems known to be very difficult for accretive evolution but easy for compositional evolution, the hierarchically consistent problems. Our approach involved the introduction of phenotypes, a level-specific selection scheme and a meta-population model. We found that the evolution of the "ideal" selection scheme is not a pre-requisite to our approach on the HIFF problem. Specifically, the experiments in this paper confirm (i) that selecting on level fitness values is beneficial, and (ii) while it is not necessary to compare level fitness values in the "ideal" order, it is necessary to prioritize certain levels over others, namely higher levels over lower levels.

The problem addressed in this paper uses a continuous fitness function. The discrete version of HIFF is difficult for RMHC2 because the fitness landscape of the highest level is akin to two needles in a haystack. We are investigating how RMHC2 can be enhanced to solve the discrete HIFF problem. We are also working on a variation of the HIFF problem which has a uniform FDC value across all levels. So far we have found that this variant problem is intractable for RMHC2 but solvable under a RMHC with an altruistic selection

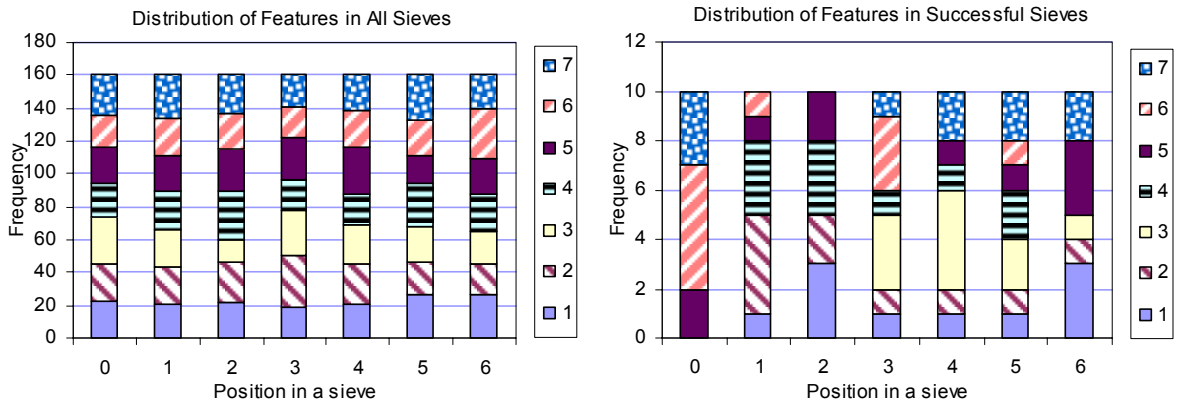

Fig. 4 Distribution of features in sieves in 10 randomly chosen parallel-model runs, N=128. One run has 16 sieves since the lattice we use is 4×4. Position 0 is the leftmost position in a sieve, position 6 is the rightmost.

scheme and under a genetic algorithm with deterministic crowding [22]. Our RMHC2 approach to the problem of evolving complex structures has some similarity with the "screening-off" process described in [23] and the question of "real" modules. We plan to investigate this further.

ACKNOWLEDGMENT

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

REFERENCES

1 S. Forrest and M. Mitchell, "Relative building-block fitness and the building-block hypothesis," in D. Whitley (editor) Foundations of Genetic Algorithms (FOGA) vol. 2, 1993, Morgan Kaufmann.

2 T. Jansen and I. Wegener, "Real royal road functions – where crossover provably is essential," *Discrete Applied Mathematics*, vol. 149, pages 111-125, 2005.

3 J. H. Holland, "Adaptation in natural and artificial systems," 1992, The MIT Press.

4 R. A. Watson, "Compositional Evolution: Interdisciplinary investigations in evolvability, modularity and symbiosis," Ph.D. Dissertation, Brandies University, 2002.

5 R. A. Watson, G. S. Hornby, and J. B. Pollack, "Modeling building-block interdependency," in A.E. Eiben, T. Bäck, M. Schoenauer and H.-P. Schweffel (editors) Parallel Problem Solving from Nature (PPSN) vol. V, 1998, pp. 97 – 106, Springer, Berlin.

6 R. A. Watson and J. B. Pollack, "A computational model of symbiotic composition in evolutionary transitions," BioSystems vol. 69, 2003, pp. 187 – 209, Elsevier.

7 E. D. de Jong, D. Thierens and R. A. Watson, "Hierarchical genetic algorithms," in X. Yao, et al. (Editors) Parallel Problem Solving from Nature (PPSN) vol. VIII, 2004, pp. 232 – 241, Springer, Berlin.

8 M. Pelikan and D. E. Goldberg, "Escaping hierarchical traps with competent genetic algorithms," in L.E Spector, et al. (editors), Genetic and Evolutionary Computation Conference (GECCO), 2001, pp. 511 – 518, Morgan Kaufmann.

9 J. Wiles, B. Tonkes and J. R. Watson, "How learning can guide evolution in hierarchical modular tasks," in J.D. Moore and K. Stenning (editors), Conference of the Cognitive Sceince Society (CogSci), 2001, pp. 1130 – 1135, Lawrence Erlbaum Associates.

10 J. D. Knowles, R. A. Watson and D. W. Corne, "Reducing local optima in single-objective problems by multi-objectivization," in Conference on Evolutionary Multi-criterion Optimization, (EMO), 2001, pp. 269 – 283, Springer-Verlag.

11 M. Ebner, M. Shackleton and R. Shipman, "How neutral networks influence evolvability," Complexity, vol. 7, 2001, pp. 19 – 33, Wiley Periodicals.

12 J. A. Walker and J. F. Miller, "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 (GECCO), 2006, pages 911 – 918.

13 H. A. Simon, "The sciences of the artificial," 1969, The MIT Press.

14 R. A. Watson and J. B. Pollack, "Hierarchically consistent test problems for genetic algorithms," in P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao and A. Zalzala (editors), Congress on Evolutionary Computation (CEC), 1999, pp. 1406 – 1413, IEEE Press.

15 T. Jones, "Evolutionary algorithms, fitness landscapes and search," PhD Dissertation, University of New Mexico, 1995, p. 57.

16 R. Levins, "Evolution in changing environments," 1968, Princeton University Press.

17 L. Altenberg, "Evolvability suppression to stabilize far-sighted adaptations," Artificial Life, vol. 11, 2005, pp. 427 – 443, The MIT Press.

18 T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in L. Eshelman (editor), 6th Interrnational Conference on Genetic Algorithms (ICGA), 1995, pp. 184 – 192, Morgan Kaufmann.

19 L. Altenberg, "Fitness distance correlation analysis: an instructive counterexample," in T. Bäck (editor) 7th International Conference on Genetic Algorithms (ICGA), 1997, pp. 57 – 64, Morgan Kaufmann.

20 L. Keller (editor), "Levels of selection in evolution," 1999, Princeton University Press.

21 T. Lenaerts, D. Groβ and R. A. Watson, "On the modeling of dynamical hierarchies: Introduction to the Workshop WDH 2002," in R. Standish, M. A. Bedau and H. A. Abbass (editors) Artificial Life VIII, 2002, The MIT Press.

22 S. Khor, "HIFF-II: A hierarchically decomposable problem with inter-level interdependency," IEEE Symposium on Artificial Life, 2007.

23 L. Altenberg, "Modularity in evolution: some low-level questions." In W. Callebaut and D. Rasskin-Gutman (Editors) Modularity: Understanding the development and evolution of natural complex systems. 2005, pp. 99-128. MIT Press.