Adaptive Hierarchical Fair Competition (AHFC) Model for Parallel Evolutionary Algorithms

Jianjun Hu* hujianju@cse.msu.edu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

Erik D. Goodman* goodman@egr.msu.edu

Kisung Seo* ksseo@egr.msu.edu

Min Pei* pei@egr.msu.edu

*Genetic Algorithms Research and Applications Group
Michigan State University
2857 W. Jolly Rd., Okemos, MI 48864

Abstract

The HFC model for parallel evolutionary computation is inspired by the stratified competition often seen in society and biology. Subpopulations are stratified by fitness. Individuals move from low-fitness to higher-fitness subpopulations if and only if they exceed the fitness-based admission threshold of the receiving subpopulation, but not of a higher one. The HFC model implements several critical features of a competent parallel evolutionary computation model, simultaneously and naturally, allowing rapid exploitation while impeding premature convergence. The AHFC model is an adaptive version of HFC, extending it by allowing the admission thresholds of fitness levels to be determined dynamically by the evolution process itself. The effectiveness of the Adaptive HFC model is compared with the HFC model on a genetic programming-based evolutionary synthesis example.

1 INTRODUCTION

Parallel evolutionary algorithms (PEA's) have gained increasing attention in many large-scale application problems including graph-partitioning problems, set partitioning problems, and many commercial efforts in analog circuit synthesis at Analog Design Automation Co. (Liang, 2001), Neolinear Inc (Ochotta, 1996; Krasnicki, 1999) and Genetic Programming Inc. (Andre, 1996). Parallel evolutionary computation models can be largely categorized into three classes (Cantu-Paz, 1998; Nowostawski, 1999): (1) global single- population master-slave models (2) single-population fine grained models, and (3) multi-population coarse-grained (or island) models. As cluster computing and networked PC's have become available in many companies, multi-population parallel models (sometimes combined with master-slave models) have become increasingly popular. Parallel evolutionary algorithms have major advantages over single-population models, including parallel evaluation and rapid exploration with decreased risk of premature convergence. However, current parallel EA's are still not competent vis-a-vis scalability, either with respect to increasing degree of difficulty of the problem or to speedup with an increasing number of processors. It is clear that a competent parallel evolutionary algorithm should have the capability to:

(1) quickly exploit high-fitness individuals as they are discovered. One of these mechanisms is Elitism, which is effective in preserving good individuals, as has been demonstrated in several of the most successful evolutionary multi-objective optimization algorithms, such as NSGAII and SPEAII (Zitzler, 2000).

(2) keep multiple high-fitness individuals simultaneously to facilitate exploration in multiple search areas or directions

(3) maintain diversity of the population to avoid premature convergence

(4) be scalable with respect to increasing number of processors

(5) adapt its parameters for autonomous evolutionary computation.

Multi-population PEA's can be classified into homogeneous models and heterogeneous models. Sprave (1999) proposed a unified model of population structures in PEAs, but his model doesn't concern with the heterogeneity of the sub-populations. In homogeneous parallel EA models, each subpopulation is regarded as playing the same role in evolution. Homogeneous PEA's often lack efficient mechanisms to exploit the newly discovered high-fitness individuals. Although they may keep several high fitness individuals in different demes, they suffer from the fact that high-fitness individuals may easily dominate all subpopulations by means of the exchange ("migration") process. Heterogeneous parallel EA's are typically more resistant to this phenomenon. For example, the injection island GA (iiGA) (Lin, 1994; Eby, 1999) uses a hierarchical structure, typically stratifying subpopulations according to the level of resolution of the representation, allowing control of the tradeoff between low-resolution exploration and high-resolution exploitation. The iiGA has also been used with different fitness functions in various subpopulations, even if they used the same problem representation. Aickelin (1999) also proposed such a PEA, which he called a pyramidal EA, in which the hierarchical structure of the subpopulations is defined by a hierarchy of fitness functions.

In a recent paper (Hu and Goodman, 2002), we proposed the Hierarchical Fair Competition (HFC) model for parallel evolutionary computation. The HFC model is inspired by the observation of a strategy employed in some societal and biological systems to maintain different high-fitness individuals in a whole population. HFC turns out to have the features of a competent PEA cited above except the adaptability of (6). In this paper, we introduce an adaptive version of the HFC model, in which the admission thresholds are automatically determined and adjusted in the evolutionary process. In Section 2, the metaphor and the HFC model are described relative to the above features. In Section 3, an adaptive mechanism for determining the parameters of the HFC model is presented, along with the algorithm. We apply the AHFC model to a genetic programming problem and compare it with the static HFC model in Section 4. The conclusions and discussion are provided in Section 5.

2 THE HIERARCHICAL FAIR COMPETION MODEL (HFC) FOR PARALLEL EVOLUTION

2.1 MOTIVATION AND BACKGROUND OF HFC

The HFC model originates from an effort to combat the premature convergence phenomenon in traditional genetic algorithms and genetic programming. In a traditional GA, as the evolutionary process goes on, the average fitness of the population gets higher and higher, so that new individuals tend to survive only if they have similarly high fitness. New "explorer" individuals in fairly different regions of the search space usually have low fitness, until some local exploration and exploitation of their beneficial characteristics has occurred. So a standard EA tends to concentrate more and more of its search effort near one or more early-discovered peaks, and to get "stuck" near these attractors (or local optima). It is clear that in a standard EA, there exists a severely unfair competition. That is, selection pressure makes high-fitness individuals reproduce quickly and thus supplant other individuals with lower fitness, some of which may lie in the vicinity of the global optimum, when if their neighborhood were explored more thoroughly, much higher-fitness individuals would be found. This fact holds true even when we find search points near a global optimum, as long as they are not close enough to have high fitness relative to those near other, earlier-explored local optima. This "unfair" competition contributes a lot to the slow search progress of many EA's when confronted with difficult, high- dimensionality, multi-modal problems. To address this unfair competition problem, we need allow young but promising individuals (*i.e.,* those in relatively newly-found regions, which may ultimately give rise to high-fitness offspring, but which are currently not of high fitness) to "grow up" and, at an appropriate time, join in the cruel competition process and be kept for further exploitation or be killed (as appropriate) when they are demonstrated with some confidence to be bad. At the same time, we hope to maintain the already-discovered high-fitness individuals and select from them even more promising individuals for exploitation without killing younger individuals. Following the tradition of getting inspiration from biology, we find that in some societal and biological systems, there exists an efficient mechanism that can maintain *and foster* potentially-high-fitness individuals (or, more accurately, potential progenitors of high-fitness individuals) efficiently. This is the hierarchical fair competition (HFC) principle as discussed below.

2.2 THE METAPHOR OF HFC: HIERARCHICAL FAIR COMPETITION IN SOCIETAL AND BIOLOGICAL SYSTEMS

Competition is widespread in societal and biological systems, but diversity remains large. After close examination, we find there is a fundamental principle underlying many types of competition in both societal and biological systems: the Fair Competition Principle.

2.2.1 The Fair Competition Principle in Societal Systems

In human society, competitions are often organized into a hierarchy of levels. None of them will allow unfair competition – for example, a young child will not normally compete with college students in a math competition. We use the educational system to illustrate this principle in more detail.

In the education system of China and many other developing countries, primary school students compete to get admission to middle schools and middle school students compete for spots in high schools. High school students compete to go to college and college students compete to go to graduate school (Fig. 1) (in many Western countries, this competition starts at a later level, but is eventually present, nonetheless). In this hierarchically structured competition, at each level, only individuals of roughly equivalent ability will participate

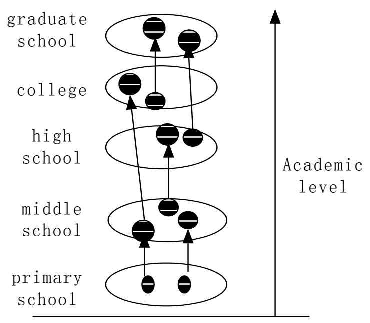


Figure 1: In education system, low level students compete to get admission to higher level schools.

in any competition; *i.e.,* in such societal systems, only fair competition is allowed. This hierarchical competition system is an efficient mechanism to protect young, potentially promising individuals from unfair competition, by allowing them to survive, learn, and grow before joining more intense levels of competition. Individuals that "lose" in these fair competitions were selected against while competing fairly only against their peers. Students compete fairly against others in their grade level because they are usually of similar absolute fitness levels, having been exposed to similar amounts of education and experience.

An interesting phenomenon sometimes found in societal competitions is the "child prodigy." A ten-year-old child may have some extraordinary academic ability. These prodigies may skip across several educational levels and begin to take college classes at a young age. An individual with sufficient ability (fitness) is allowed to join any level of competition. This also suggests that in subpopulation migration, we should migrate individuals according to their fitness levels, rather than according to "time in grade."

With such a fair competition mechanism that exports high-fitness individuals to higher-level competitions, societal systems reduce the prevalence of unfair competition and the unhealthy dominance or disruption that might otherwise be caused by "early-achieving" individuals.

2.2.2 The Fair Competition Principle in Biological Systems

It is somewhat surprising that in "cruel" biological/ ecological systems, the fair competition principle also holds in many cases. For example, there are mechanisms that reduce unmatched or unfair competition between young animals and mature ones. Among mammals, young individuals often compete with their siblings under the supervision of parents, but not directly with other mature individuals, since their parents protect them against other adults. When the young grow up enough, they leave their parents and join the competition with other mature individuals. Evolution has found the mechanisms of parental care and sibling competition to be useful in protecting the young and allowing them to grow up and develop their full potentials. Fair competition seems to be beneficial to the evolution of many species.

2.3 THE HFC MODEL

Inspired by the fair competition principle and the

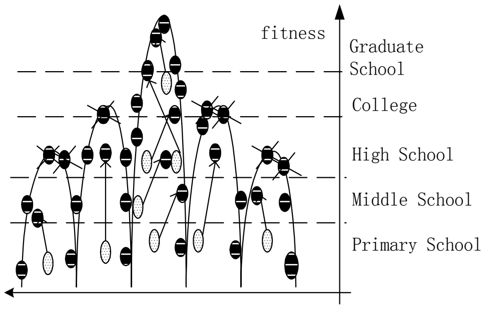


Figure 2: HFC model extends the search horizontally in search space and vertically in fitness dimension and kills bad individuals at appropriate times while allowing promising young individuals grow up continuously

hierarchical organization of competition within subpopulations in societal systems, we propose the Hierarchical Fair Competition parallel model (HFC), for genetic algorithms, genetic programming, and other forms of evolutionary computation.

In this model (Fig 3), multiple subpopulations are organized in a hierarchy, in which each subpopulation can only accommodate individuals within a specified range of fitness. The entire range of possible fitnesses is spanned by the union of the subpopulations' ranges. Conceptually, each subpopulation has an admission buffer that has an ***admission threshold*** determined either initially (fixed) or adaptively. The admission buffer is used to collect

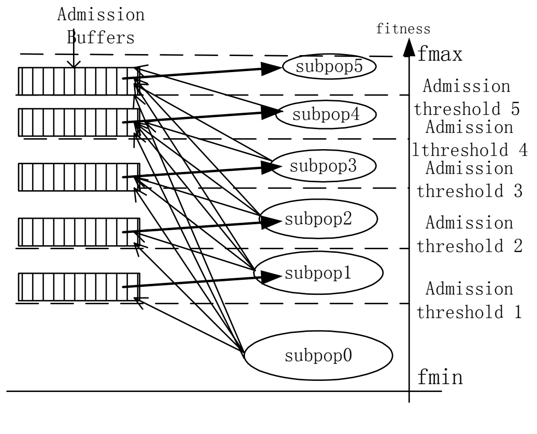


Figure 3: In HFC model, subpopulations are organized in a hierarchy with ascending fitness level. Each subpopulation accomodates individuals within a certaiin fitness range determined by the admission thresholds

qualified candidates, synchronously or asynchronously, from other subpopulations. Each subpopulation also has an ***export threshold*** (fitness level), defined by the admission threshold of the next higher-level subpopulation. Only individuals whose fitnesses are between the subpopulation's admission threshold and export threshold are allowed to stay in that subpopulation. Otherwise, they are exported to the appropriate higher-level subpopulation. Exchange of individuals is allowed only in one direction, from lower-fitness subpopulations to higher-fitness subpopulations, but migration is not confined to only the immediately higher level.

Each subpopulation can have the same or different sizes, operators, and other parameters. However, considering that there are often more low-fitness peaks than high-fitness peaks, we tend to allocate larger population sizes or more subpopulations to lower fitness levels, to provide extensive exploration; and we tend to use higher selection pressures in higher-fitness-level subpopulations to ensure efficient exploitation. As it is often easier to make a big fitness jump in a lower level subpopulation, we often end up using larger fitness ranges for low-level subpopulations, and smaller ranges for high-level subpopulations (Fig. 2), but, of course, that depends on the properties of the fitness landscape being explored. The critical point is that the whole range of possible fitnesses must be spanned by the union of the ranges of all levels of subpopulations. Of course, the highest-level subpopulation(s) need no export threshold (unbounded above) and the lowest-level subpopulation(s) need no admission threshold (unbounded below).

Exchange of individuals can be conducted synchronously after a certain interval, or asynchronously, as in many parallel models. At each moment of exchange, each individual in each subpopulation is examined, and if it is outside the fitness range for its subpopulation, it is exported to the admission buffer of a subpopulation with an appropriate fitness range. When a new candidate is inserted into an admission buffer, it can be inserted into a random position or inserted by sorting (or a null buffer may be used, inserting migrants directly into the receiving subpopulation, using some replacement rule). After export, each subpopulation imports the appropriate number of qualified candidates from its admission buffer into its pool. Subpopulations (especially at the base level) fill any spaces still open after emptying their admission buffers by generating new individuals at random to fill the spaces left by the exported individuals.

The number of levels in the hierarchy or number of subpopulations (if each level has only one subpopulation) can be determined initially or adaptively. In the static HFC model, we must manually decide into how many levels the fitness range will be divided, the fitness thresholds, and all other GA parameters. In a dynamic HFC model, we can dynamically change the number of levels, number of subpopulations, size of each subpopulation, and admission and export fitness thresholds. As will be seen below, a benefit of the adaptive HFC model (an example of a dynamic HFC) is that it can adaptively allocate search effort according to the characteristics of the search space of the problem to be solved, thereby searching more efficiently (initial research on various methods for adaptation of thresholds is in preparation for reporting elsewhere). However, even "coarse" setting of the parameters in a static HFC model has yielded major improvement in search efficiency over current EA's on example problems.

Another useful extension to HFC used here is to introduce one or more *sliding* subpopulations, with dynamic admission thresholds that are continually reset to the admission threshold of the level in which the current best individual has been found. Thus, these subpopulations provide additional search in the vicinity of the advancing frontier in the hierarchy.

2.3.1 HFC as a competent parallel model for parallel evolutionary computation

(1) While low-fitness individuals can persist long enough to allow thorough exploration, as soon as they produce high-fitness offspring, the offspring can advance to higher-fitness levels immediately for further exploitation, to compete and be recombined with other high-fitness individuals.

(2) The HFC model maintains a large number of high-fitness individuals in high-fitness-level subpopulations without threatening lower-fitness (but perhaps promising) individuals. Thus possibly promising new search locales can persist long enough to be appropriately exploited.

(3) HFC provides another mechanism for maintaining diversity. First, the diversity of the population is ensured by the stratification in the fitness space. Second, continuous introduction of random individuals into the lowest-level subpopulations and the promotion of their high-fitness offspring to upper-level subpopulations can be regarded as the introduction of entropy and randomness into the overall evolutionary system. Actually, looking from low-fitness levels to higher-fitness levels, we observe increasing order in the population. The HFC evolution is thus a self-organizing process in which the highest order is achieved at the top fitness level. This mechanism reduces the chance of HFC becoming "stuck" at local optima and helps it explore new search areas. HFC thus implements implicitly a multi-start or re-initialization mechanism on a continual basis.

(4) The HFC model quickly captures superior offspring and moves them to a place where they are free to compete with, and be recombined with, each other. This produces an effect similar to the elitism often used in multi-objective evolutionary computation, such as NSGAII or SPEAII (Zitzler *et al.,* 2000), in which superior individuals are also kept separately.

At that level, we can control the intensity of selection to determine the tradeoff between exploitation of those high-fitness individuals and exploration in their neighborhoods.

(5) HFC has a good scalability to more processing hosts. As more processors are available, they can be distributed to different fitness levels – either to low-level subpopulations for more extensive exploration, or to the higher-level ones for intensive exploitation of high-fitness individuals.

One of the major difficulties in the HFC evolutionary algorithm is the determination of the admission thresholds for a given problem. As the fitness landscape is often unknown before evolutionary search, it is hard to define these admission thresholds initially. Considering that admission thresholds in HFC are only used to segregate the whole population to avoid unfair competition, the behavior of the search is generally not extremely sensitive to the values of these admission thresholds, so that it is not necessary to set them to exactly optimal values. The only requirement for these thresholds is that the union of the fitness level ranges (which is determined by these admission thresholds) span the entire range of possible fitnesses. Based on this analysis, we propose an automatic admission thresholds determination mechanism for HFC model.

3 THE ADAPTIVE HFC MODEL

In the static HFC model, we need to determine the number of subpopulations, the number of fitness levels, the relationship of subpopulations to fitness levels and the admission thresholds of each fitness level. All the admission thresholds are determined based on some initial exploration of the fitness landscape of the problem, such as the range of the fitness or distribution of early-discovered peaks. The threshold adaptation mechanism proposed here enables us to be relieved from this prerequisite expertise in the problem space. All we must decide is the number of admission levels (N_l).

Since in HFC, random individuals are continuously inserted into subpopulations of the base fitness level, the export threshold of the base fitness level can be set as the average fitness of the whole population after several (*nCalibGen*) generations. In AHFC, this is called the **calibration stage**, which determines the level of the fitness value of frequently encountered ("normal") individuals with respect to random individuals. So the base level is used to export normal individuals to higher levels for further exploitation. At the end of the calibration process, the standard deviation σ_f and the max fitness $f_{\max}$ of individuals at the highest level, the average fitness f_μ of individuals at the base level are calculated. Then the fitness range of each level can be calculated by the following formula:

Admission threshold of base level= $-\infty$ (1)

Admission threshold of the first level = f_μ (2)

Admission threshold of the highest fitness level =

$$f_{\max} - \sigma_f \qquad (3)$$

Admission thresholds of other fitness levels L_i, are determined by:

$$f_\mu + L_i \times (f_{\max} - \sigma_f - f_\mu)/(N_l - 2) \quad i = 1,\ldots,N_l - 1 \qquad (4)$$

Table 1: Adaptive Heterogeneous HFC Algorithm for Parallel EA's

```
1. Initialization
        Determine $\vec{P}_{EA}$ : parameters for standard multi-population EA. (We assume here using one set of parameters for all subpopulations)
        $N_l$ :Number of levels of the hierarchy
        $nCalibGen$: calibration generations
        $nExch$: generations between admission process exchanges
        $gen$ = 1: current generation
2. Do
        if $gen < nCalibGenth$ (in calibration stage)
                run EA without exchange
        else if $gen = nCalibGenth$ (calibration stage ends)
                determine the admission thresholds for each level by formulas (1) - (4)
        else if $gen \% \ nExch = 0$
                Do for each subpopulation from lowest level to highest level {
                        Examine fitness of each individual and export to corresponding subpopulation at higher level for which fitness range accommodates this exported individual (replacing worst individual in target subpopulation)}
                end do
        else if $gen \ \% \ nGenUpdate = 0$
        update admission thresholds of all but the base level by (3), (4)
        $gen$ ++
until the stopping criterion is satisfied.
        return the highest-fitness individual(s) from the highest-level subpopulation
End
```

However, it is clear that as the evolutionary search goes on, higher-fitness individuals are continuously discovered that ruin the segregation by the above admission thresholds determined at the initial calibration stage. So a dynamic **admission threshold updating** mechanism is proposed here. After each *nGenUpdate* generations, the maximal fitness, $f_{\max}$, and the fitness standard deviation of the top level sub-populations, σ_f, are recomputed to determine the admission threshold of all the fitness levels except the base level and the first level, by (3) - (4).

To enable efficient search, the mapping relationship of sub-populations to all levels also needs to be adapted dynamically. It is obvious that at initial stage, as all individuals are randomly generated, these individuals usually have low fitness. So most of the subpopulations should belong to the base level. As higher-level individuals discovered, more subpopulations should be allocated to higher levels to exploit high-fitness individuals. The following scheme is used in this paper: firstly, all subpopulations are allocated to base level. After the calibration stage, subpopulations are then evenly allocated to each level. Extra subpopulations can be allocated to higher levels (if aggressive exploitation is desired) or to lower level (if intensive exploration is desired).

This AHFC algorithm works like a string. At the initial stage, it is quite compressed, but gradually, the string stretches to accommodate individuals with a larger range of fitness. The whole algorithm of AHFC is given in Table 1. For simplicity, we give the pseudo code only for the adaptive HFC model with synchronous exchanges (no buffers).

4 EXPERIMENTS

The adaptive HFC model for Genetic Programming (HFC-GP) has been applied to a real-world analog circuit synthesis problem that was first pursued using GP with a static HFC (Hu, 2002). In this problem, an analog circuit is represented by a bond graph model (Seo, 2001; Fan, 2001) and is composed of inductors (I), resistors (R), capacitors (C), transformers (TF), gyrators (GY), and Sources of Effort (SE). Our task is to synthesize a circuit, including its topology and sizing of components, to achieve specified behavior. The objective is to evolve an analog circuit with response properties characterized by a pre-specified set of eigenvalues. By increasing the number of eigenvalues specified, we can define a series of synthesis problems of increasing difficulty, in which premature convergence problems become more and more significant when traditional GP methods are used.

Circuit synthesis by GP is a well-studied problem that generally demands large computational power to achieve good results. Since both topology and the parameters of a circuit affect its performance, it is easy to get stuck in the evolution process.

4.1.1 Experiments on an Analog Circuit Synthesis Problem

Four circuits with increasing difficulty are to be synthesized, with eigenvalue sets as specified in Table 2. Circuits were evolved with single-population GP, multiple-population GP, HFC-GP, and AHFC-GP. The GP parameter for the single-population GP is shown in cell (1,2) of Table 3. The GP parameters for the multi-population GP were the same as for the single-population GP, except that the total population is divided into subpopulations with sizes shown in cell (2, 2) of Table 3. A one-way ring migration topology was used.

The parameters for the HFC-GP were the same as for the multi-population GP, except that the ring migration was replaced by the HFC scheme. The fitness admission thresholds were set based on our prior experience with such eigenvalue problems. In this problem, we defined a fitness admission threshold for each subpopulation (one subpopulation per level, in this case) as shown in cell (2, 3) of Table 3. Subpopulation 15 was used as a "sliding" subpopulation to aggressively explore the fitness frontier. The parameters of AHFC-GP were nearly identical to those of the HFC, except that we don't need to determine the admission thresholds of each level.

The performances of the four approaches were assessed on four problems with increasing difficulty. Each experiment was run ten times, with the average of the results reported in Fig.4, where the four GP methods are indicated by

OnePop: Single-population GP
MulPop: multi-population GP (ring topology)
HFC-GP: HFC model for GP
AHFC-GP: Adaptive HFC model for GP

Table 2: Target Eigenvalues

Problem 1:	6-eigenvalue problem
	$-2\pm3.3i,\ -7.5\pm4.5i,\ -3.5\pm12.0i$
Problem 2:	8-eigenvalue problem
	$-2\pm3.3i,\ -7.5\pm4.5i,\ -3.5\pm12.0i,\ -3.4\pm12.0i$
Problem 3:	10-eigenvalue problem
	$-2\pm3.3i,\ -7.5\pm4.5i,\ -3.5\pm12.0i,\ -3.4\pm12.0i,\ -10.0\pm8.0i$
Problem 4:	12-eigenvalue problem
	$-2\pm3.3i,\ -7.5\pm4.5i,\ -3.5\pm12.0i,$ $-3.4\pm12.0i,\ -10.0\pm8.0i,\ -1.5\pm3.0i$

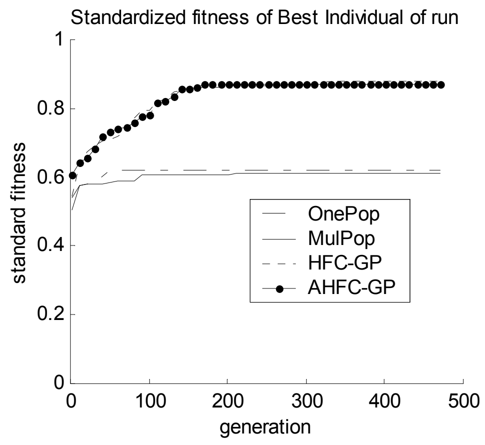
Table 3: Parameter Settings for GP

Parameters of Single Population GP	Popsize: 2000
	init.method = half_and_half
	init.depth = 3-6
	max_nodes = 800
	max_depth = 13
	crossover rate = 0.9
	mutation rate = 0.1
	max_generation = 1000

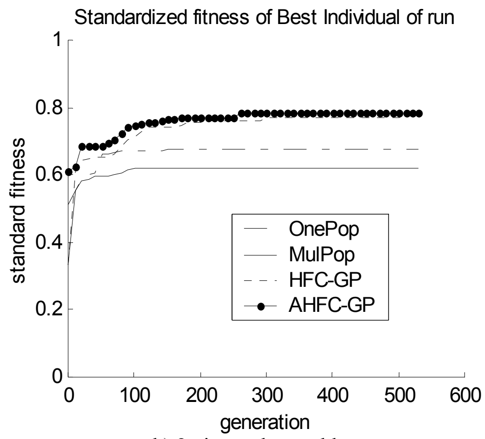
Additional Parameters of Multi-Population GP	Number of subpopulations = 15; Size of subpop 2 to 14 = 100 size of subpop 1 = 300 size of subpop 15 = 400 migration interval = 10 generations migration strategy: migrate (copy) 10 best individuals to the next subpopulation in the ring to replace its 10 worst individuals
Additional Parameters of HFC-GP	admission_fitnesses of: subpop 1 = -100000.0 subpop 2 to 14: 0.65, 0.68, 0.72, 0.75, 0.78, 0.80, 0.83, 0.85, 0.87, 0.9, 0.92, 0.95 subpop 15 = varying

Additional Parameters of AHFC-GP	NCalibGen = 10
	NExch = 10
	N_l =8

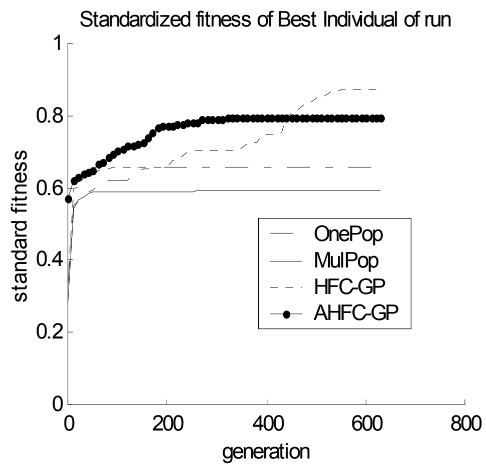
From Figure 4, it is impressive to see that in all four problems, both AHFC and HFC performed dramatically better than the other algorithms vis-à-vis best of run, and the improvement was more dramatic on the more difficult problems. The superior performance at the initial generations may have resulted from the rapid exploitation of superior individuals, in a single subpopulation, in comparison to the ring parallel GA. Yet convergence in the HFC and AHFC was much slower than in the single- and multi-population GP runs. In fact, we observe relatively steady improvement during the runs for this set of problems. For the easier problems, the (dynamic) AHFC actually out-performed the (static) HFC slightly, in spite of the rich experience on this class of problems that was used for setting the HFC thresholds. The fact that the HFC ultimately surpassed the AHFC on the two harder problems indicates that there is room for improvement of the AHFC scheme used here. However, the fact that it is competitive with human-determined static values based on prior experience shows that it is a step in a beneficial direction.



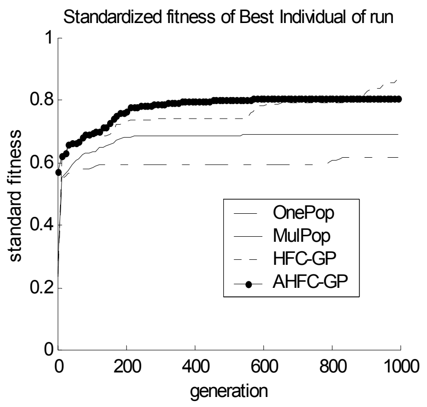
a) 6-eigenvalue problem



b) 8-eigenvalue problem



c) 10-eigenvalue problem



d) 12-eigenvalue problem

Figure 4. Fitness of Best Individual to Date vs. Generation: dashdot (OnePop), solid (MulPop), dashed (HFC)

5 CONCLUSIONS AND FUTURE WORK

Based on our analysis of the role of admission thresholds used in the HFC model and our experiments on a series of difficult, highly epistatic real-world problems, it has been demonstrated that the adaptive HFC model can work nearly as well as the original HFC model, and even better in some cases, without any prerequisite knowledge of the fitness landscape of the problem. The dynamic allocation of the subpopulations to fitness levels also improves the search efficiency. These adaptation mechanisms make our algorithm to be easily plugged into new problems without much parameter tuning. Our experiments demonstrated the effectiveness of the HFC and AHFC models in improving significantly both the search speed and the quality of the best solutions found compared with standard EAs.

This paper represents a first step toward autonomous parallel evolutionary computation based on the HFC model. The second step is the automation of the adaptive distribution of the computing resource among levels. We expect that the number of subpopulations, the number of fitness levels, the distribution of subpopulations to each level, along with the admission thresholds, can all be determined adaptively, in which case we would have an autonomous parallel evolutionary computation model in which the communication topology and migration scheme are all decided by the evolutionary process itself, according to the characteristics of the problem at hand.

In this paper, we implemented the synchronous version of the AHFC model and simulated parallel genetic programming on a single PC. More consideration about the communication cost and asynchronous adaptation mechanism of the AHFC model in the case of a large population is needed. The scalability of the AHFC model with respect to more processors also needs to be proved with experiments on real parallel cluster computing facilities, which is on the top of our task list.

Acknowledgment

The authors acknowledge the assistance of Prof. Ronald C. Rosenberg and Zhun. Fan in defining and exploring the example problem presented here. This work was supported by the National Science Foundation under contract DMI 0084934.

References

J.M. Liang; T. McConaghy; A. Kochlan; T. Pham; G. Hertz. "Intelligent Systems for Analog Circuit Design Automation: A Survey," *Proceedings World Multiconference on Systemics, Cybernetics and Informatics,* Vol. IX. SCI 2001/ ISAS 2001, Orlando, Florida (USA), 2001.

E. S. Ochotta, R. A. Rutenbar, L. R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/ OBLX," *IEEE Trans. CAD*, Vol. 15, pp. 273-294, Mar., 1996.

M. Krasnicki, R. Phelps, R. Rutenbar, and R. Carley. "Maelstrom: Efficient simulation-based synthesis for custom analog cells," *Proceedings of the 1999 ACM/IEEE Design Automation Conference*, 1999.

D. Andre and J. R. Koza. "Parallel genetic programming on a network of transputers," in Angeline, Peter J. and Kinnear, Kenneth E., Jr. (eds.), *Advances in Genetic Programming,* MIT Press, Cambridge, MA, 1996.

E. Cantu-Paz, "A survey of parallel genetic algorithms," *Calculateurs Parallels*, 10(2), Paris, Hermes. 1998.

J. Sprave. A unified model of non-panmictic population structures in evolutionary algorithms. In P. J. Angeline and V. W. Porto, editors, *Proc. 1999 Congress on Evolutionary Computation (CEC'99)*, vol 2, p. 1384-1391, Washington D.C., 1999. IEEE Press, Piscataway NJ.

N. Mariusz. and R. Poli, "Review and Taxonomy of Parallel Genetic Algorithms," Technical Report, University of Birmingham, School of Computer Science, No. CSRP-99-11, May 1999.

S.C. Lin, E. Goodman, and W. Punch, "Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach," *IEEE Conf. on Parallel and Distrib. Processing*, Nov., 1994.

D. Eby, R. C. Averill, E. Goodman, and W. Punch, "Optimal Design of Flywheels Using an Injection Island Genetic Algorithm," *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, 13, p. 389-402, 1999.

U. Aickelin, "A Pyramidal Evolutionary Algorithm with Different Inter-Agent Partnering Strategies for Scheduling Problems," *Genetic and Evolutionary Computation Conference Late-Breaking Papers*, E. Goodman, ed., ISGEC Press, San Francisco, pp. 1-8. 2001.

J.J. Hu, E.D. Goodman, "The Hierarchical Fair Competition (HFC) Model for Parallel Evolutionary Algorithms," *Proceedings of the 2002 Congress on Evolutionary Computation: CEC2002*, (forthcoming), IEEE, Honolulu, Hawaii, 2002.

K. Seo, E. Goodman, and R. Rosenberg, "First Steps toward Automated Design of Mechatronic Systems Using Bond Graphs and Genetic Programming," *Proc. Genetic and Evolutionary Computation Conf. - 2001*, July 7-11, Morgan Kaufmann, San Francisco, p. 189, 2001.

Z. Fan, J.J. Hu, K. Seo, E. Goodman, R. Rosenberg, and B. Zhang, "Bond Graph Representation and GP for Automated Analog Filter Design," *Genetic and Evolutionary Computation Conference Late Breaking Papers*, pp. 81-86, 2001.

E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolution Algorithms: Empirical Results," *Evolutionary Computation*, 8(2), pp. 173-195, 2000.