The Hierarchical Fair Competition (HFC) Framework for Sustainable Evolutionary Algorithms

Jianjun Hu hujianju@msu.edu
Department of Computer Science and Engineering,
Erik Goodman goodman@egr.msu.edu
Kisung Seo ksseo@egr.msu.edu
Zhun Fan fanzhun@egr.msu.edu
Department of Electrical and Computer Engineering,

Rondal Rosenberg rosenber@egr.msu.edu
Department of Mechanical Engineering,
Michigan State University, East Lansing, MI, 48823, USA

Abstract

Many current Evolutionary Algorithms (EAs) suffer from a tendency to converge prematurely or stagnate without progress for complex problems. This may be due to the loss of or failure to discover certain valuable genetic material or the loss of the capability to discover new genetic material before convergence has limited the algorithm's ability to search widely. In this paper, the Hierarchical Fair Competition (HFC) model, including several variants, is proposed as a generic framework for sustainable evolutionary search by transforming the convergent nature of the current EA framework into a non-convergent search process. That is, the structure of HFC does not allow the convergence of the population to the vicinity of any set of optimal or locally optimal solutions. The sustainable search capability of HFC is achieved by ensuring a continuous supply and the incorporation of genetic material in a hierarchical manner, and by culturing and maintaining, but continually renewing, populations of individuals of intermediate fitness levels. HFC employs an assembly-line structure in which subpopulations are hierarchically organized into different fitness levels, reducing the selection pressure within each subpopulation while maintaining the global selection pressure to help ensure the exploitation of the good genetic material found. Three EAs based on the HFC principle are tested - two on the even-10-parity genetic programming benchmark problem and a real-world analog circuit synthesis problem, and another on the HIFF genetic algorithm (GA) benchmark problem. The significant gain in robustness, scalability and efficiency by HFC, with little additional computing effort, and its tolerance of small population sizes, demonstrates its effectiveness on these problems and shows promise of its potential for improving other existing EAs for difficult problems. A paradigm shift from that of most EAs is proposed: rather than trying to escape from local optima or delay convergence at a local optimum, HFC allows the emergence of new optima continually in a bottom-up manner, maintaining low local selection pressure at all fitness levels, while fostering exploitation of high-fitness individuals through promotion to higher levels.

Keywords

Sustainable evolutionary algorithms, building blocks, premature convergence, diversity, fair competition, hierarchical problem solving, genetic programming

1 Introduction

Evolutionary Algorithms (EAs) have been applied to more and more challenging problems, including evolvable hardware (Yao and Higuchi, 1998), evolutionary robotics (Harvey et al., 1997), electrical circuit and control system synthesis (Koza et al., 2000), and many other engineering design problems. One of the major issues with these complex real-world problems is the scalability problem, as discussed in (Koza, 1994; Vassilev and Miller, 2000). Roughly speaking, an algorithm is not scalable if the computing effort to find an optimal (or acceptably good) solution grows beyond economic practicality as the size of the problem becomes large enough to represent important real-world situations. This problem is usually related to the premature convergence phenomenon of EAs: the loss, before an optimal solution is discovered, of the capability to make fitness progress, often attributed to a loss of population diversity (typical in genetic algorithms) or to a loss of the capability to generate useful variation because of the convergence of a portion of the genome (as often occurs at the root of the trees in classical genetic programming). For hard problems, tuning parameters or adjusting ad hoc operators usually does not provide scalability. However, many real-world problems are characterized by the existence of inherent structural regularity or even building blocks of various sizes in the solutions. Their successful solution by evolutionary means often amounts to an incremental form of evolution with gradual and steady progress, or with a succession of improvements, as described by (Holland, 2000). Evolutionary algorithms with this kind of steady-state innovation capability (Goldberg, 2002) are called continuing or sustainable evolutionary algorithms.

Sustainable EAs are closely related to two major issues in EA research, namely, premature convergence and scalability. Currently, there are three main families of strategies for fostering sustainable search in EAs.

The first one is centered around the concept of diversity. It is based on the fact that at convergence, the diversity of the EA population is low. Genetic operators such as crossover and mutation, when coupled with selection and applied to a nearly-converged population, tend to produce solutions that resemble those already in the population, or the population would not have converged to that point in the first place. However, while maintenance of diversity is a necessary condition for avoiding of premature convergence, it is not a sufficient one. Many methods of maintaining high diversity are not effective in maintaining high-quality search – for example, using a very high mutation rate or introducing many randomly initialized individuals to a highly-evolved population is typically not very effective, but are sometimes employed. Many other more effective techniques are proposed to maintain or increase the diversity of a population, including niching methods such as fitness sharing and crowding, restricted mating, island or multi-population models, pygmies and civil servants (Ryan, 1996), species-conserving GA (Li et al., 2002), etc. Diversity can also be explicitly maintained by allowing different subpopulations to use different representations or resolutions, as in the injection island GA (iiGA) (Lin et al., 1994). Good surveys are provided in (Mahfoud, 1995; Darwen, 1996; Ryan, 1996).

The second approach toward sustainable evolutionary search is conducted in the context of incremental evolution, where a simplified version (or subproblems) of a problem is subjected to evolution, the results of which will be used as starting point for a more complex version of the problem. Harvey (1992) proposed the Species Adaptation GA (SAGA), which uses genetically fairly converged populations during the search. SAGA is suitable for incremental evolution in which the problem itself changes over time, such as dynamic optimization problems and evolutionary robotics.

It usually depends on a variable-length genotype representation and neutral network to fight against premature convergence. This kind of incremental evolution with variable genotypes, when coupled with speciation, has achieved significant results in evolving neural networks (Stanley and Miikkulainen, 2002). A similar, but more complex, evolutionary approach has also been used to improve scalability for evolvable hardware, evolving increasing levels of user-defined subsystems (Torresen, 2002) with a simple divide-and-conquer approach.

The last, but most important, line of study on sustainable EA design is centered around the concept of building block supply, growth and mixing (Goldberg, 2002). Messy GAs and other competent GAs (Goldberg, 2002) are characterized by the identification and recombination of building blocks, and have mainly been applied to binary-encoded GAs. The pervasiveness of building blocks (or subcomponents) in the physical and biological world strongly suggests that competent EAs for complex real-world problems will rely heavily on the capability to maintain population diversity, to continue exploration, and to exploit extant building blocks (Holland, 2000). The success of competent GAs (Goldberg, 2002) for difficult problems has confirmed this, but it is only the first step toward exploiting the building block concept. However, the concept of good, co-adapted genetic material as used here does not require that it be tightly linked, as in the classical building block definition as a short, low-order, high-fitness schema. Good genetic material may also be of higher order, and loci need not be adjacent on the chromosome to constitute good genetic material for further evolution, as is amply demonstrated in evolution strategies work.

Another important observation of the physical and biological world is that there exist successive levels of building blocks from nuclei to atoms to molecules to polymers; from organelles to cells to tissues to organs to organisms to ecosystems. The maintenance and evolution of all levels of hierarchical subsystems or building blocks is essential to the efficiency of nature in forming complex systems. Simon (1973) showed that hierarchical systems evolve much more rapidly from elementary constituents than will non-hierarchical systems containing the same numbers of elements. This kind of hierarchical strategy for problem solving has been explored first in genetic programming with ADFs (Koza, 1994), module acquisition (Angeline and Pollack, 1994) and Adaptive Representation (AR) (Rosca and Ballard, 1994), and later in a genetic algorithm, hBOA, the hierarchical extension of BOA (Pelikan and Goldberg, 2001). These approaches are somehow representation specific and rely on the explicit identification of building blocks in the forms of ADFs, modules, or merged variables. A similar hierarchical strategy is also used in coevolution for subcomponent discovery and assembly (Potter and De Jong, 2000). There are also some manual hierarchical divide-and-conquer strategies with multiple subpopulations (Aickelin et al., 2001; Hsu et al., 2002; Torresen, 2002; Eby et al., 1999) .

However, there is another strategy for exploiting the structure of the problem space to conduct efficient evolutionary search. Evolutionary algorithms such as evolution strategies do not rely on identifying building blocks, but rather employ what we shall here call stepping stones to guide their evolutionary progress. A stepping stone is defined as an individual of a given fitness level that is judged, relative to its parent(s) or others of the fitness level of its parent(s), as worth keeping. This does not depend upon the existence or explicit identification of building blocks or other properties or schemata of the problem that are important to producing high-fitness solutions. While evolution strategies use only a single level of stepping stones, the Hierarchical Fair Competition (HFC) method described here will use a hierarchy of stepping stones stratified by fit-

ness levels.

Despite these efforts, the very convergent nature of the existing EA framework still keeps us from achieving scalability for many complex problems. EA practitioners still struggle between large population sizes with a single epoch and small population sizes with multiple epochs (Goldberg, 1999; Luke, 2001) . The building-block-discovering mechanisms like ADF need to be coupled with a sustainable search procedure that can make sustainable progress and have little risk of premature convergence.

This paper introduces a generic framework, called Hierarchical Fair Competition (HFC), for sustainable evolutionary algorithms. This mechanism was devised by considering the prerequisite of sustainable evolution, including diversity maintenance, incremental development, and assembly of good genetic material at all fitness levels. Because it has the capability to ensure sustained diversity, HFC is also believed to be applicable to EAs such as evolution strategies (ES), which, like HFC, do not depend on the existence of building blocks and do not require a crossover operator, although that applicability has not been shown in this paper.

The next section describes two observations and two assumptions about existing EAs, and discusses their effects on the performance of current EAs for difficult problems. Next, two requirements for sustainable evolution met by the HFC evolutionary model are described, and some related techniques in this context are discussed. In Section 3, the HFC model is presented, including its original metaphors and its three components. Two adaptive variants of HFC are introduced that allow HFC to adapt to the search space of the problem. Section 4 then presents the experimental results of HFC for a genetic programming benchmark problem and a real-world system synthesis problem. Section 5 introduces a new robust GA named QHFC, obtained by combining HFC and deterministic crowding, and it demonstrates how HFC can endow a classical convergent GA with a robust and sustainable search capability. After some discussion and analysis of the HFC model in Section 6, the paper concludes with proposals for future directions of research.

2 Background

2.1 Two Observations and Two Assumptions in Typical EA Frameworks

Current EA techniques carry with them implicit assumptions about evolutionary search. There are two important observations and two implicit assumptions about most current EAs that can prove detrimental to their performance on many difficult problems, but which HFC can help to alleviate.

2.1.1 Observation I: The average absolute fitness of the population increases continually

One prominent observation about the typical EA framework is that as the evolutionary process goes on, the average absolute fitness of the population gets higher and higher. The population then seems to gradually lose its capability for exploration, until convergence to the global optimum or premature convergence (to some undesired region of the search space) finally occurs. Many algorithms use rank-based or tournament-type selection methods to enable them to continue to distinguish between better and worse individuals in the population, but that does not solve this problem, as explained below. People associate this loss of exploratory capability with the loss of diversity and try to increase the diversity of the population as a remedy. But actually, loss of diversity is just a symptom of this phenomenon - diversity per se is not sufficient: what is required is to maintain useful diversity. The more direct reason that premature convergence

occurs is that with increasing average fitness of the population, only those new individuals with competitively high fitness tend to be allowed to survive. However, new "explorer" individuals in fairly different regions of the search space usually have low fitness, until enough local exploration and exploitation of their beneficial characteristics occur. Even for those with competitive fitness, their sparse distribution in distant regions still exposes them to the risk of being lost as the result of insufficient sampling or genetic drift. This explains why simply increasing the mutation rate or inserting random individuals does not work well: a highly evolved individual with co-adapted components, when mutated strongly or recombined with a random individual, almost always leads to inferior offspring. Random individuals or extensively mutated individuals usually have low fitness and can not survive long enough to get their new genetic material exploited as their loci have not yet been able to co-adapt to form high-fitness schemata. This means that the higher the average fitness of the population, the less it has the capability to make and utilize large genetic modification, while small modification can only perturb the high-fitness individuals and it cannot move them out of local optima. This gradual loss of the capability to incorporate new genetic material is one of the key factors leading to premature convergence.

Some kind of mechanism is needed to reduce this effect of increasing the average fitness of the population (or to reduce the domination of the population by the earliest-discovered individuals of the highest fitness), which make it hard for new individuals (with less well co-adapted genetic material and usually with lower fitness) to survive. The Species Conserving GA (Li et al., 2002) uses similarity-based niching to explicitly keep those exploratory "not highly fit but different enough species seed" individuals. Fitness sharing and crowding use a similar strategy to keep "not highly fit but different enough" exploratory individuals by limiting the number of individuals at specific local search regions called niches. However, such techniques suffer from a property of the search spaces of many difficult problems, as described next.

2.1.2 Observation II: Many difficult problems have enormous and highly multimodal search spaces

Many difficult search problems (and in fact, many of those at which the HFC methods described here are aimed) are made difficult because their search spaces are enormous, with high dimensionality and high multimodality. (There are other types of difficult search problems, of course – for example, needle-in-a-haystack problems – that these methods may not address any better than other methods will. But for the purposes of this paper, let us consider first problems made hard because of high dimensionality, large cardinality of many of the dimensions, and strong multimodality). Distance or similarity-based niching, multi-population and search space division methods are widely used to maintain a desired level of population diversity. Crowding and deterministic crowding (Mahfoud, 1992) work by replacing similar individuals to allow space for different individuals. Fitness sharing uses carrying capacity of niches to keep diverse individuals. The island parallel model uses multiple subpopulations in the hope that each accommodates different candidate solutions, with only infrequent migration to aid exploitation. SCGA (Li et al., 2002) uses similarity-based species protection to promote diversity. Tsutsui and Ghost (1998) use search space division to keep diversified populations. All these techniques work by spreading or pushing individuals horizontally (rather than "vertically" along the fitness dimension in Figure 1) to different areas in the search space (Figure 1) – for example, fitness sharing systematically prevents individuals from clustering at a few high-fitness peaks. They strive to main-

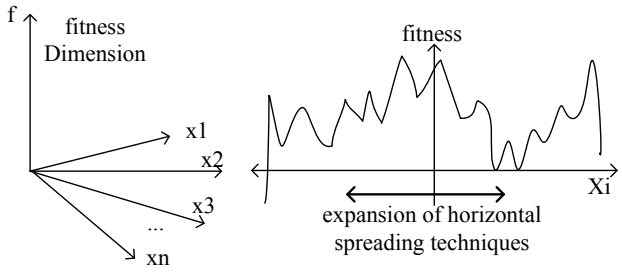


Figure 1: Enormity of the search space and local optima and the expansion in horizontal spreading EAs

tain genotypic or phenotypic diversity. But they work well only with sufficiently large population sizes or sufficient numbers of subpopulations. Koza et al. (1999) worked with population sizes of 350,000 or even larger to achieve satisfactory results, confirming this capability, but at significant cost. However, as is shown in Section 5, increasing population size is not a scalable solution for difficult problems and may induce unnecessarily high computational cost.

It is clear that the horizontal genotype or phenotype spreading techniques mentioned above do not solve the scalability problem for addressing difficult problems, as they perform poorly as the size and multimodality of the search space increases. As illustrated in Figure 1, even a one-dimensional search problem can be highly multimodal. For problems of high dimensionality (many engineering problems have 50-100 or more independent design variables) or for problems in which the number of local optima grows exponentially with problem size, the search space and number of local peaks (or attractors) becomes too large to allocate sufficient population size to accommodate representatives of most of them. Even though there are some techniques to reduce the number of effective niches to some extent (Goldberg et al., 1992), the issue of requiring a huge population size is still not resolved. This difficulty is caused by some deep design assumptions of the current EA framework about how an EA should work.

2.1.3 Assumption I: Problems can be reasonably addressed using an EA conducting a single-thrust search

As the motivation of using EAs is to find solutions with high fitness, EAs are usually taken as a single-thrust search process: starting from random populations, some lower-level building blocks are discovered and assembled into higher level building blocks (or, as in the case of evolution strategies, some decision variables evolve decreasing magnitudes for their perturbations) and the best fitness and average fitness of the population increase until stagnating or reaching specified stopping criteria. For example, messy GAs are to be sized such that the initial population is sufficient to ensure an adequate supply of raw building blocks (Goldberg et al., 1993). Population-sizing models for GAs are derived (Goldberg, 1989; Goldberg et al., 1992) based on takeover times of building blocks in an (implicitly) single thrust evolution. Impractically large population sizes would be needed to make fitness sharing work for massively multi-modal problems (Goldberg et al., 1992). Huge population sizes are used to ensure sufficient diversity in multi-population parallel GP (Koza et al., 1999), still based on the single-thrust EA model.

For difficult problems with a huge number of local optima and a complex search space, it is inappropriate to assume that all building blocks or potentially important combinations of genetic material are present in the initialization stage, and even if they were, that they would be able to survive long enough to be exploited, due to unbalanced sampling resulting from their different degrees of salience (Goldberg (1999) treats the building block aspect). Some type of "assembly line" sustainable EA search process is needed. In such a process, a random individual generator continuously feeds new individuals into the EA population. While this mechanism would be completely ineffective in an ordinary GA (because the new individuals are grossly uncompetitive in the populations they enter and make poor mates for highly evolved solutions), this kind of continuous, probabilistically complete supply of building blocks can partially relieve messy GAs and other EA techniques from the burden of unfeasibly large population size requirements for difficult problems. That is, no particular minimum population size is required at any one instant. However, simply introducing random individuals continuously into a population experiencing increasing average fitness is not sufficient to achieve the desired result, as explained in Section 2.1.1. HFC provides a hierarchical method to implement this sustainable EA search process. This is clearly confirmed in Section 5, where an HFC-enabled simple GA is able to solve reliably a 256-bit non-shuffled Hierarchical-IF-and-only-iF (HIFF) problem (Watson and Pollack, 1999) with a population size of 100, while a simple GA with deterministic crowding with or without reinitialization cannot solve it reliably even with a population size of 4000.

2.1.4 Assumption II: Premature convergence can be adequately handled by maintaining the genotypic and/or phenotypic diversity of a population of a reasonable size throughout the run

Loss of genotypic diversity among intermediate-fitness individuals typically leads to loss of further exploratory power. The entities converging could be structures near the root of a GP tree, a group of loci in a GA, or a set of decision variables in ES. EAs work by accumulating a succession of improvements. High fitness individuals are usually evolved in this incremental way, in which lower-level building blocks are assembled into higher-level building blocks, or the structures of moderate-fitness individuals become fixed. The higher the fitness of the individual, the more tightly those loci become co-adapted to each other. In the traditional single-thrust EA framework, the evolutionary process cannot guarantee that all such intermediate-level structures grow at similar speeds, and as a result, only some of them get a chance to be sampled and recombined, while others are lost. At the same time, the "critical initial generations" effect says, "unless a schema (or its components) grows at the outset, its chances for success later are quite poor" (Goldberg, 2002). This is because single-thrust EAs do not tend to preserve the stepping stones that were used to make the initial climb up the fitness landscape, nor to maintain a "place" where those stepping stones may be regenerated. Thus, they lack the capability to incorporate new genetic material continually. All the individuals are marching toward the fitness frontier or the highest-fitness regions of the search space discovered to date. So essentially, for single-thrust EAs, the search process converges rapidly at the outset with regard to the majority of the intermediate-fitness structures present in the population. The diversity-preservation mechanisms soon concentrate on preserving genotypic diversity among the highest-fitness individuals, neglecting the individuals that constituted the "path" to the top. The premature convergence of the best individuals is then to a large extent caused by the premature

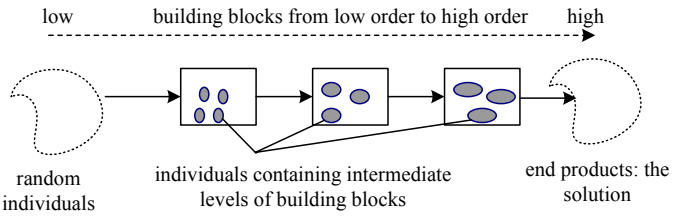


Figure 2: The assembly line structure of the sustainable EA model

loss of useful lower-fitness individuals.

The importance of maintaining and continuously generating new "stepping stones" suggests a new model for an EA (Figure 2). This model arranges processing units in a series to form an "assembly line". Randomly generated individuals are fed into one end unit of this assembly line, in which individuals with different desirable characteristics random "chunks" are assembled into individuals with better-than-random fitness. The output of this first processing unit, the first-level stepping stones, is continually fed to the next processing unit for assembly of higher-performance stepping stones, and so on. The HFC model provides exactly such an assembly line processing structure for continuously supplying and assembling intermediate levels of stepping stones.

2.2 Two Requirements for Sustainable Evolution Met by the HFC Model

Given an understanding of how the above two observations and two assumptions about traditional EAs reflect impediments to their performance, the HFC model has been devised to meet two requirements described below that are believed to be important to assure sustainable search capability for an EA.

2.2.1 Requirement I: The Continuous Supply and Incorporation of Low-Level Genetic Material

A sustainable EA must assure that a continual supply of low-level novel genetic material is available for assembly or mutation to discover (or rediscover) individuals of intermediate fitness levels, in order to sustain the search indefinitely.

In HFC, the individuals of the whole population are organized into a hierarchy of fitness levels. A generator of random individuals continuously feeds raw genetic material into an assembly line of processing units. This process does not rely on a huge initial population to ensure an adequate supply of the raw material, as in (Goldberg, 1989). Even if during some initial generations, some beneficial or critical low-level structures are not available or not selected, they can appear later with the continuing inflow of random individuals from the generator, and can survive and be incorporated into higher-fitness individuals that have a good chance of surviving to influence the sustainable search.

Previous attempts to introduce random individuals into nearly converged populations to partially reinitialize them (for example, Goldberg, 1989; Whitley et al., 1991) , or to increase the mutation rate dramatically when the population begins to converge (for example, (Cobb and Grefenstette, 1993)) have difficulty incorporating lost building blocks into highly evolved individuals, as described above in Section 2.1.1, especially in the case of genetic programming. What does HFC do to change this situation? In a converged population with many high-fitness individuals, the individuals are usually

composed of highly coupled or co-adapted subcomponents. When these individuals undergo crossover with a random individual or are mutated strongly, the operations almost always destroy the well-evolved closely coupled relationship, reducing the fitness dramatically. So it is difficult for the highly converged (evolved) population to jump out of local optima. This can also be explained based on Holland's hypothesis that, "Neutrality decreases with each successive level" (Holland, 2000), which means that for high-fitness populations, mutation doesn't work as effectively as it does with low fitness populations. Therefore, it is better to introduce random individuals or use high mutation rates only in subpopulations that do not already contain high-fitness individuals with highly co-adapted subcomponents. Accordingly, in HFC, the newly generated random individuals are always incorporated into the low-fitness-level subpopulations, so they can survive within the fair competition environment with low selection pressure and be naturally utilized, with their beneficial genetic material moving up the assembly line only when refined to enable competitive survival at a higher fitness level.

When one examines the fitness progress curve of most GP (and also GA) experiments, the most salient observation is that the largest fitness gains typically occur in the very early stages. The evolution in the later stages appears to be more like a refining process. In tree-based genetic programming, the initial stages of evolution usually establish the general framework of the topology of the GP trees of an individual. The nodes near the tree root converge relatively quickly. Actually, the higher the fitness, the more constraints the established topology puts on possible later modifications, and the less likely they are to become major innovations. It is not wise to effectively terminate this phase after a very limited assembly process, as the fitness of individuals rises. The phenomenon described here is very similar to Waddington's canalization of development (Waddington, 1942). Since highly evolved individuals have less and less probability of changing their basic frameworks, it is preferable to maintain, somehow, repositories of representative intermediate individuals, from which further innovation may occur.

This analysis suggests introducing a paradigm shift in the strategy for avoiding premature convergence. Instead of trying to help a highly converged population escape local attractors, or to retard its convergence, a better strategy is to allow the continual emergence of individuals that may populate new attractors, in a bottom-up manner, as is seen in HFC. This process is kept unbiased by the high-fitness regions already found in the search process.

2.2.2 Requirement II: The Culturing and Protection of Intermediate-Fitness Levels of Individuals

Intermediate-fitness individuals (stepping stones) at a progression of advancing fitness levels must be maintained and protected against elimination by high-fitness individuals discovered early, thereby sustaining them sufficiently to allow for their exploitation.

Goldberg (2002) stresses the importance of ensuring building block growth via recombination for design of competent GAs. However, based on the analysis of Observation II, it is clear that competent GAs and other EAs may suffer from insufficient population size and premature loss of intermediate building blocks, for sufficiently difficult problems. While the HFC structure, unlike that of competent GAs, does not require the existence of identifiable building blocks, for problems that have them, the HFC model, with its assembly line structure of subpopulations in successive fitness levels, provides a mechanism for continuously culturing lower-level building blocks into

higher-level building blocks, so long as different fitness levels implicitly correspond to different levels of building blocks embedded in the individuals belonging to those levels. Of course, this is possible only when the genetic operators somehow are able to combine building blocks rather than disrupting building blocks as is the case for two-point crossover in the shuffled HIFF problem (Watson and Pollack, 1999). Fitness is already used as the signal for differentiating good building blocks from bad ones or building blocks of different orders in messy GA schemes (Goldberg, 2002). This is justified by the fact that high-fitness individuals usually have more strongly coupled components (like real variables, binary genes, etc.). So these groups of strongly coupled components can be regarded as higher-level building blocks. This kind of implicit building block concept - i.e., not needing to explicitly identify the building blocks, but maintaining them appropriately, nonetheless - distinguishes HFC from the competent GAs (Goldberg, 2002) and from ADF in GP (Koza, 1994).

To reduce the "founder effect" (Holland, 2000) - that early-discovered building blocks interfere with finding and nurturing other competing building blocks - some kind of protection mechanism must be used. This idea can be explored jointly with the concept of diversity maintenance. Niching usually uses a distance measure to spread and keep genotypically or phenotypically diverse individuals. Elitism is another mechanism sometimes used to protect good building blocks. The Species Conserving GA explicitly preserves "different and good species seeds." The Cohort GA (Holland, 2000) uses delayed-but-guaranteed reproductions to reduce the loss of good schemata. HFC can incorporate any or all of these techniques at each level. However, HFC provides an additional mechanism to keep multiple diverse high-fitness individuals, along with their intermediate genetic material. First, for each level, there is a continual inflow of new individuals from lower levels. In this sense, HFC can be looked on as a hierarchical version of elitism, in which subpopulations at each level are the "Halls of Fame" of all lower-level subpopulations. This kind of hierarchical elitism is especially important since the fractions of crossovers and mutations that produce better offspring decrease with increasing fitness. It ensures that the products of these increasingly rarely successful crossover and mutation operations get preserved and exploited. From a diversity point of view, the whole population is highly diversified, since we have individuals ranging from the random to the highly evolved. In fact, the ratio of high- to moderate- to low-fitness individuals may be adjusted arbitrarily in HFC to control the relative rates of exploration and exploitation. This additional diversity maintenance capability is especially useful where distance criteria for niching are not available or are hard to compute. It also allows stronger selection pressure at each level without the risk of premature convergence of the population as a whole. While this requirement for protection of multiple fitness levels of individuals has been described above in the building block framework of Holland and Goldberg, it does not depend on the existence of classical building blocks to be used effectively, unlike the competent GA mechanism for explicit identification of building blocks.

2.3 Compatibility of HFC with Other Techniques for Increasing Scalability of Evolutionary Algorithms

Goldberg (2002) describes two types of approaches to improve the scalability of an EA. The first is the explicit building block discovery and exploitation mechanisms, as used, for example, in his messy GA and in Koza's automatically defined function (ADF) (Koza, 1994), which may transform intractable problems into tractable ones. The second type includes the family of parallel GA models, including both coarse- and fine-

grained variants, the reduction of the evaluation effort through the use of more rapidly calculated approximations to the fitness function or fitness inheritance, and the hybridization approach. As a generic framework for sustainable search, HFC is compatible with all of these approaches, and may be applied in addition to any of them. HFC is a natural extension of the island (coarse-grain) model for parallel EAs, although it may also be implemented as a specialized selection scheme for breeding and survival within a single population (Hu et al., 2003). It is also clear that the sustainable search enabled by the assembly line structure of HFC, when coupled at each level with existing techniques like fitness sharing, species conserving, and elitism, can yield yet more efficient and effective EAs, as is demonstrated in Section 5 and in more studies to be reported in the future.

3 The HFC Model and its Adaptive Variants

3.1 The Metaphor: Fair Competition Principle from Societal and Economic Systems

The HFC model was initially inspired by competition mechanisms observed in biological, societal and economic systems. Competition is widespread in these systems, and selection is sometimes very strong, but diversity remains large. In the biological world, environmental niches are organized in different levels, occupied by all types of organisms with different sizes. As Bonner pointed out, "Size is a universal property of all organisms and size niches remain constant over geological time" (Bonner, 2000). This means that competition in natural evolution is essentially organized into different levels and bacteria don't compete directly with elephants. In human society, competitions are also 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. Young individuals with potentially outstanding capabilities in specialized areas do not have to face immediate competition with the most highly developed individuals in the population, but can rise rapidly as they become more able to compete, to play an important role in future advances. 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. We use the educational system to illustrate this principle in more detail.

In the educational 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 (Figure 3) (in most 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 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 up before joining more intense levels of competition. If some individuals are "lost" in these fair competitions, they were selected against while competing fairly only against their peers. If we take the academic level as a fitness level, it means that only individuals with similar fitness can compete.

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) can join any level of

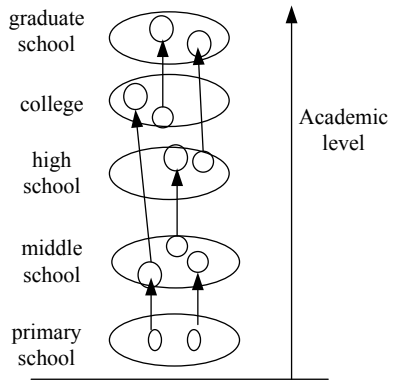


Figure 3: In the educational system, low-level students compete to get admission to higher-level schools. Competition is designed to exist only among individuals of similar ability levels

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 "over-achieving" individuals. It maintains relatively low selection pressure at each level while maintaining strong global selection pressure.

Similar "fair competition" is also enforced in the economic world, where a variety of regulations and laws (e.g., antitrust laws) are set up by the government or international organizations to ensure fair competition and exclude domination. The system governing athletic competitions of various sorts is similar and well known. These strategies are necessary to promote healthy competition and allow new start-ups to have a chance to mature.

3.2 The Components of the HFC Model

Inspired by the fair competition principle and the hierarchical organization of competition within different levels in biological and societal systems, the Hierarchical Fair Competition model (HFC) is proposed for use in genetic algorithms, genetic programming, and other forms of evolutionary computation. The HFC model is composed of three interdependent components.

3.2.1 The Hierarchical Organization of Subpopulations to Establish a Fitness Gradient

In the HFC model (Figure 4), multiple subpopulations are organized in a hierarchical way, in which each subpopulation belongs to a specific fitness level that accommodates immigrating individuals within a specified range of fitness, and that forces emigration of individuals with fitness above that range. The entire range of possible fitnesses is spanned by the union of the fitness ranges of all levels. Each level has an admission buffer that has an admission threshold determined either initially (fixed) or adaptively. The admission buffer is used to collect qualified candidates, synchronously or asynchronously, from the subpopulations of lower levels. Each level also has an export fitness threshold, defined by the admission threshold of the next higher fitness level.

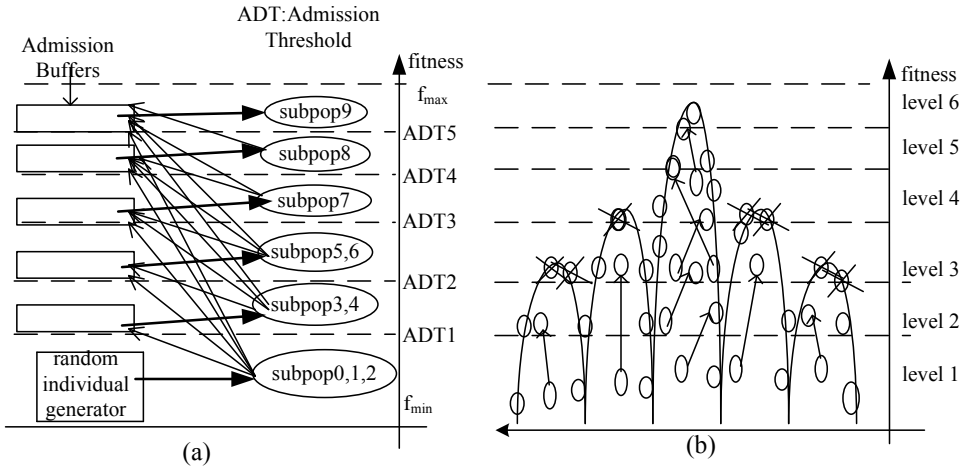


Figure 4: (a) In the HFC model, subpopulations are organized in a hierarchy by ascending fitness level. Each level (with one or more subpopulations) accommodates individuals within a certain fitness range determined by its admission threshold (b) The HFC model extends the search horizontally in the search space and vertically in the fitness dimension and kills bad individuals at appropriate times while allowing promising young individuals to grow up continuously

Only individuals whose fitnesses are above the admission threshold and below the export threshold of the fitness level of a given subpopulation are allowed to enter, or stay, respectively, in that subpopulation. Otherwise, they are exported to a subpopulation of the appropriate higher fitness level.

A problem that can occur at any fitness level is that the children produced via mutation or crossover of individuals at a given fitness level have fitnesses below its admission threshold. This could allow the average fitness of individuals at that level to degrade below the admission threshold. This degradation problem can be dealt with in any of several ways:

1. these low-fitness offspring can be allowed to remain in the level – reminiscent of the occasional backward step allowed in simulated annealing – with the assumption that the selection mechanism and immigration replacement policy will act strongly enough that the average fitness remains satisfactorily above the admission level, or

2. the low-fitness offspring could be "down-migrated" to a lower-level population, but this is seen as undesirable – it is not these low-fitness individuals that are likely to contain the building blocks we strive to preserve and recombine; or

3. offspring generated with fitnesses below the admission threshold could be discarded, and the operation generating them repeated (perhaps for a fixed maximum number of times). This would tend to help combine building blocks into higher-level ones; or

4. any offspring generated from lower fitness than their parent (or parents, in the case of crossover) could be discarded (and the parents kept, instead), or

5. offspring could be generated from a chosen parent (or pair, for crossover) until an individual with fitness at least as high as the parent (or the mean of the two parents) is generated (subject to a maximum number of trials).

It may appear that option 3) is the most desirable, and would not lead to systematic degradation of the fitness level of any population. However, some experiments with QHFC, discussed in Section 5, showed that policy 1) is computationally more efficient, so long as the lower levels can continue exporting individuals to higher levels. This kind of "fuzzy" segregation of fitness levels can also increase the diversity of the population. The experiments reported below used policy 1), so did not discard any offspring or demote them to lower-fitness subpopulations.

A useful extension to HFC (which was used here for static HFC experiments) is to introduce one or more "floating" 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 fitness hierarchy. In this scheme, it is reasonable not to start the evolution process in higher-level subpopulations until some minimum number of immigrants above the admission threshold have entered them.

3.2.2 Random Individual Generator: the Source of Genetic Material

As illustrated in Figure 4, at the bottom fitness level, there exists a random individual generator that feeds "raw" genetic material (in the form of individuals) continuously into the bottom processing level. It is beneficial if this generator is made to be unbiased such that it is able to supply a complete set of all possible low-level building blocks, unless there is prior knowledge about the search space that should be used to bias the generator. Of course, new building blocks may be discovered by recombination and other genetic operations. The inflow of random individuals relieves HFC from depending on a large initial population size to provide and preserve enough different genetic material at the outset to allow a thorough search of the problem space in a single "epoch."

3.2.3 The Migration Policy from Lower to Higher Fitness Levels

The exchange of individuals can be conducted synchronously at a certain interval or asynchronously, as in many parallel models. At each moment of exchange, or, if desired, as each new individual in a subpopulation is evaluated, any individual whose fitness qualifies it for a higher level is exported to the admission buffer of whatever higher level has a fitness range that accommodates the individual. After the exporting processes of all levels finish, the subpopulations of each level import an appropriate number of qualified candidates from their admission buffers to replace some worst individuals (unless some other local diversity maintenance policy is used to select candidates for replacement). If subpopulations at the base level find any open spaces left over from exporting, they fill those spots with random individuals. If subpopulations of higher levels find empty spaces after importing individuals from their admission buffers or if their admission buffers are empty, they can either mutate current members or select two members and do crossover to generate the needed number of new individuals, or they can import individuals in the subpopulation from the level below, even though they do not meet the admission criterion. (Note: the bottom level subpopulations import from the random individual generator). In the experiments reported here, crossover is used to make up any shortfall of individuals at any level except the bottom level.

In the HFC methods reported here, migration of individuals was allowed in only one direction, from lower-fitness subpopulations to higher-fitness subpopulations, but migration is not confined to only the immediately higher level. In addition, any "degraded" individuals generated in a population of a given fitness level were not "exiled" to a lower level, even though they may not have met the admission criterion for migrating into the level-that is, method 1) above was used. This may allow for promising jumps with short-term fitness degradation in the fitness landscape. It can also be justified in that, despite the possible decrease of the fitness, many useful intermediate-level building blocks may still remain. In the work reported here, the admission buffer is volatile. That is, at each migration time, these buffers are filled with candidates and are emptied after the exporting process is complete at each migration interval. However, it is expected that, at some point, admission buffers may be used as the main persistent repository of good genetic material, while the subpopulations serve as the assembly workshops, similar to the way "external populations" are used in some modern multi-objective evolutionary algorithms such as PAES (Knowles and Corne, 2000).

The algorithmic procedure of HFC is similar to a simple multi-population GA except the above controlling mechanisms, namely, the setting and adjusting of admission thresholds, the migration policy, and the incorporation of random individuals into the bottom levels. For simplicity, the detailed algorithm pseudo code is omitted here.

3.2.4 Setting the Parameters of the HFC Model

Several decisions must be made to configure the HFC model. First, the number of fitness levels and their admission thresholds should be determined according to the fitness range of the problem. For difficult problems, more levels are preferred to allow more precise control of the assembly line for good genetic material. The export threshold of the bottom level can be set, for example, as the average fitness of random individuals of the first generation. Other admission thresholds can be set by dividing the fitness range evenly (used in the static HFC algorithm) or with some adaptive adjustment as in the adaptive HFC to be discussed below. As it is often easier to make large fitness jumps in a lower-level subpopulation, they may often be usefully assigned larger fitness ranges, leaving the high-level populations more closely spaced. But, of course, that depends on the properties of the fitness landscape of the problem. 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).

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, the number of fitness levels into which the entire fitness range will be divided is decided in advance, as are the fitness thresholds and all other GA parameters. In the adaptive HFC model, the number of levels, number of subpopulations, size of each subpopulation, and admission and export fitness thresholds of each can be determined automatically and adapted as the evolution proceeds. The benefit of the adaptive HFC model (in Section 3.3) 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. However, even a "coarse" setting of the parameters in a static HFC model has demonstrated major improvements in search efficiency on some difficult benchmark problems in both GP (Section 4) and GA (Section 5).

Subpopulations at each level can have the same or different sizes and settings for

other evolutionary parameters. It may be important to allow a majority of the total population to explore the fitness frontier more aggressively, at least at later portions of the search. This can also be achieved by allowing a relatively large subpopulation size for the floating subpopulation, which always belongs to the highest activated fitness level. In addition, it is also preferable to use higher selection pressures (larger tourney sizes, for example) in higher-fitness-level subpopulations to ensure more efficient exploitation - remember that our HFC mechanism will reduce the tendency to convergence that this otherwise produces. The point is that it should be sufficiently large to keep multiple distinct strains of individuals and thus promote the efficient mixing of building blocks or other types of beneficial genetic material. For each level, the population can be divided into several subpopulations to promote diversity. The size and fitness range of each subpopulation can be set. Commonly used diversity maintenance methods may also be incorporated within each subpopulation if desired.

The migration interval can be set according to the difficulty of the problem. If it is very difficult to make progress in fitness, it may be better to make this interval longer, in order to allow more mixing before introducing new genetic material from lower levels.

3.3 Adaptive HFC models

In the static HFC model described above, we need to determine the number of subpopulations, the number of fitness levels, the relationship of subpopulations to fitness levels, and the admission threshold of each fitness level. All the admission thresholds are determined based on some initial examination of the fitness landscape of the problem, such as the range of the fitness. Adaptation mechanisms can relieve us from this prerequisite expertise in the problem space and the associated need to determine the subpopulation topology. Here, two variants of the adaptation scheme are described. One allocates the subpopulations (and computational effort) adaptively to the fitness levels as the fitness frontier advances and corresponding levels are activated. Another variant adaptively adjusts the admission thresholds to follow the progress of the fitness frontier.

3.3.1 Adaptive Allocation of Subpopulations to Fitness Levels (HFC-ATP)

In the static HFC model, the allocation of subpopulations to fitness levels is configured before the evolution is started. A shortcoming of this method is that in the initial generations, the high level subpopulations won't actually contain any qualified individuals, and so, according to this method, are not activated for evolution. Although the floating subpopulation (Figure 5a) can be used to enhance the search effort on the fitness frontier, the computational capability of higher-level subpopulations is largely wasted before their activation. This reduces the effective population size. One possible solution to this difficulty is to allow two-way migration. Then individuals of all levels could be evaluated and unqualified low-fitness individuals in high-level subpopulations could be moved to lower levels; however, this is still likely to yield ineffective search at the higher-level subpopulations, as at most a few distinct individuals at those levels survive and remain in those subpopulations early in the evolution process. Another solution presented here (Figure 5b) is to adaptively allocate subpopulations to fitness levels. In the beginning, all subpopulations are allocated to the bottom level. Later, once certain higher levels get some qualified individuals, then all intermediate levels are activated and the subpopulations on those activated levels are filled by admitting qualified individuals or importing lower level individuals. This "HFC-ATP" algorithm works like a rubber band. At the initial stage, it is quite compressed, but

gradually, the rubber band stretches to accommodate individuals with a larger range of fitnesses. In this paper, subpopulations are allocated evenly to all activated levels. Details of this algorithm are described in (Hu et al., 2002).

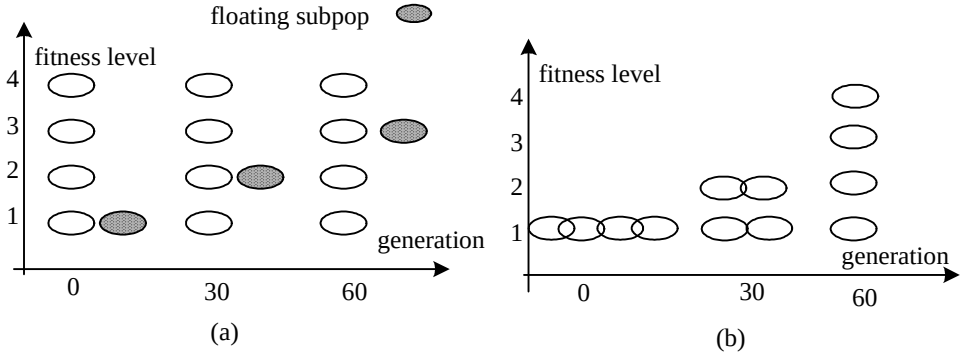


Figure 5: Floating subpopulation (left) & adaptive allocation of subpopulations to fitness levels (right).

3.3.2 Adaptive Setting of Admission Thresholds (HFC-ADM)

One of the 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 sometimes hard to define appropriate admission thresholds initially. Considering that admission thresholds in HFC are only used to stratify the population to avoid unfair competition, the behavior of the search is generally not extremely sensitive to the precise 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) spans the entire range of possible fitnesses. The adaptive admission threshold mechanism is composed of a calibration stage and update cycles. During the calibration stage, the export threshold of the base fitness level is set as the average (or some percentile) fitness of the whole population at the end of *nCalibGen* generations. This base level is used to export higher-fitness "normal" individuals to higher levels for further exploitation. Immediately at the end of the calibration process, the standard deviation σ_f, the max fitness f_{max} , and the average fitness f_μ of individuals of all levels are calculated. Then the fitness range of each level can be calculated by the following formula:

Admission threshold of base level $f^0_{adm} = -\infty$ (1)

Admission threshold of the first level $f^1_{adm} = f_\mu$ (2)

Admission threshold of the highest fitness level $f^{N_l-1}_{adm} = f_{\max} - \sigma_f$ (3)

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

$f^i_{adm} = f_\mu + L_i \times (f_{\max} - \sigma_f - f_\mu)/(N_l - 2)\ \ i = 2, ..., N_l - 2$ (4)

where N_l is the number of fitness levels of HFC. Here in (4), the basic idea is to start from maximum fitness value, first set $[f_{\max},\ f_{\max} - \sigma]$ as the fitness range of the top level, set $[-\infty,\ f_\mu]$ as the fitness range of base level, and then allocate the fitness range of $[f_\mu,\ f_{\max} - \sigma - f_\mu]$ equally to the other $N_l - 2$ levels.

However, it is clear that as the evolutionary search goes on, higher-fitness individuals are continually discovered that ruin the stratification developed by the admission thresholds determined at the initial calibration stage. Therefore a dynamic ad-

Table 1: Shared Parameters for Even-10-parity problem

max_evaluations: 300,000	tournament selection size: 7
Init_method : half_and_half	Init_depth : 4-7
max_nodes : 300	max_depth: 10
pCrossover: 0.95	pReproduction: 0.05
pMutation: 0.0	reproduction with best selection (elitism)

mission threshold updating mechanism is needed. After each generation, the maximal fitness,f_{max} , and the fitness standard deviation of the top-level subpopulations,σ_f , are recomputed to reset the admission thresholds of all the fitness levels except the base level and the first level, by (3) - (4). To maintain some momentum and to avoid dramatic variation of the best fitness, we use the *AdaptationRate* to decide by how much to change the current admission thresholds:

$$f_{adm}^{i,new} \leftarrow (1 - AdaptationRate) f_{adm}^{i} + AdaptationRate * f_{adm}^{i,\exp} \quad (5)$$

where is the updated admission thresholds for level i, f_{adm}^{i} is the previous old admission threshold of level i, $f_{adm}^{i,exp}$ is the expected new admission threshold for level i. The idea is that it is better to maintain the smoothness of the admission update with respect to the increase of f_{max}.

4 Performance Evaluations of HFC in GP

This section presents experimental results that demonstrate the sustainable search capability of the HFC model on two GP problems in terms of solution quality and evaluations needed. A robust HFC algorithm for GA and its experimental results will be presented separately in Section 5. Since many interesting real-world problems, such as neural network synthesis and electrical circuit design, go beyond numerical optimization, two genetic programming problems were chosen, for their variable length representations and the necessity for incremental evolution. The highly discrete and rugged landscape of the program space adds additional difficulty to these problems. One of the test problems is the even-n-parity problem, which is well established as a benchmark problem for genetic programming (Poli and Page, 2000). This problem is characterized by the existence of perfect solutions. The other is a real-world electrical circuit synthesis problem, namely, an eigenvalue placement problem, in which no perfect solution is generally known. Testing with these types of problems, it is hoped, will yield more insight into the HFC model than evaluation on much simpler benchmark problems.

4.1 The Even-10-Parity Benchmark Problem

As a Boolean function induction problem, the task of the even-n-parity problem is to evolve a function with n binary inputs and one binary output such that the output of the function is 1 (true) if and only if an even number of the inputs is evaluated to be true. The difficulty of this problem depends on the function set exploited and the order (n) of the target function. Since even-n-parity problems with the standard function set OR, AND, NOR, NAND provide little gradient information for incremental search and are deemed an inappropriate benchmark problem for GP (Poli and Page, 2000), we prefer to use the following function set OR, AND, NOT, XOR in this paper. The inclusion of the XOR function provides a means for GP to make a succession of improvements.

Table 2: Parameter Setting for HFC techniques: HFC and HFC-ADM use fixed subpopulation sizes for each fitness level. HFC-ATP dynamically allocates subpopulations to fitness levels. (see Figure 5)

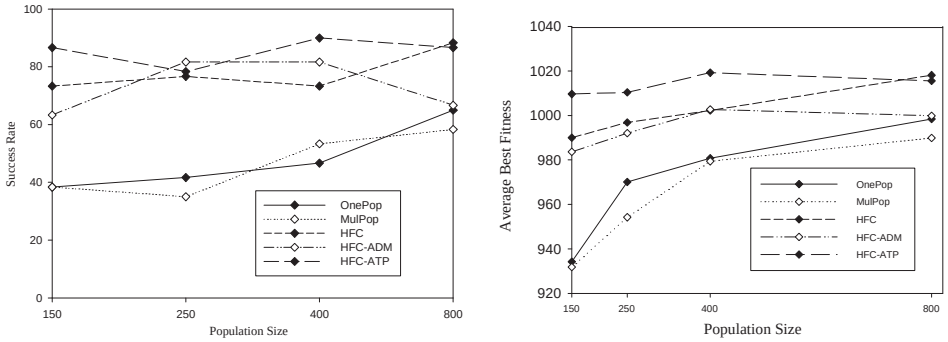
	HFC	HFC-ADM	HFC-ATP
	Exchange frequency: 5 Fitness levels: 9		
	Admission thresholds from level 1 to 9: -100, 550, 600, 650, 700, 750, 800, 850, 900	Threshold adjustment Interval: 5 nCalibrateGen : 5 AdaptationRate: 0.95	Admission thresholds from level 1 to 9: -100, 550, 600, 650, 700, 750, 800, 850, 900
	Subpop i belongs to fitness level i for $i < 10$, subpop 10 is the floating subpop.		
Popsize=150	Subpop size for subpop 1 to subpop 9: 10 For floating subpop10: 60		Subpop size for subpops 1 to 10: 10,10,10,10,10,15,15,20,20,30
Popsize=250	Subpop size for subpops 1 to 6: 15 for subpops 7 to 9: 20 for floating subpop10: 100		Subpop size for subpops 1 to 10: 15,15,15,20,20,25,25,25,40,50
Popsize=400	Subpop size for subpops 1 to 9: 20, 20, 20, 20, 30, 30, 30, 40, 40 for floating subpop10: 150		Subpop size for subpops 1 to 10: 20,20,30,30,35,35,40,40, 50, 100
Popsize=800	Subpop size for subpop 1 to subpop 9: 30, 30, 30, 30, 40, 40, 60, 160 for floating subpop10: 400		Subpop size for subpop 1 to subpop 10: 40,40,40, 60, 70,70, 10,80,100,200

4.2 Experimental Settings

In this experiment, five algorithms are evaluated on the even-10-parity problem, all using the function set mentioned above. The algorithms are labeled single population (OnePop), multi-population (MulPop), HFC with floating subpopulation (HFC), HFC with adaptive admission threshold determination mechanism (HFC-ADM), and HFC with adaptive allocation of subpopulations (HFC-ATP). Four (total) population sizes (150, 250, 400, 800) are tested for all five algorithms, each with 60 runs so that the results are statistically significant. The shared parameters for all the experiments are summarized in Table 1. Since there is a random individual generator at the bottom level to provide new genetic material, mutation was not used here. Note that a certain degree of elitism is enforced for OnePop and MulPop through the reproduction operator with best selection operator. This removes the possibility that HFC provides superior performance only by virtue of elitism. For the MulPop algorithm, the whole population is evenly divided into 10 sub-populations, arranged in a ring topology, as is commonly done. The migration interval is set as 5 generations. The number of individuals (K) to be migrated is about 1%, (K = 2, 3, 5, and 8 for population sizes of 150, 250, 400, and 800, respectively). The migration strategy for MulPop is to select the K best individuals from the donor subpopulation and copy them (without removal from the donor) to replace the K worst individuals in the receiving subpopulation. The parameters specific to the HFC technique and its adaptive variants are summarized in Table 2.

The determination of fitness admission thresholds for HFC and HFC-ATP was very straightforward. The maximum fitness of the even-10-parity problem is 1024, and the average fitness of the first 5 generations is typically around 550, so the fitness range of [550, 1024] was evenly allocated to all fitness levels. As will be shown by the performance of HFC-ADM with adaptive admission threshold determination, HFC techniques are not generally very sensitive to these parameters. The allocation of subpopulation sizes was decided based on a rough balance between exploring the fitness frontier and maintaining a supply of intermediate-fitness building blocks or stepping stones. Generally, more individuals were allocated to higher fitness levels. For HFC and HFC-ADM, the floating subpopulation size should be big enough for effective ex-

ploration at the highest currently activated fitness level. All of the parameters in Table 2 were set before these runs were made, without tuning them for each specific problem instance.



(a) Comparison of success rates after 300,000 evaluations: HFC techniques consistently outperform standard OnePop and MulPop GP and are essentially invariant with respect to population size.

(b) Comparison of average best-of-run fitness. With given population size and evaluations, HFC techniques consistently achieve better average best fitness, reflecting their robustness of search.

Figure 6: Comparison of success rates after 300,000 evaluations and Comparison of average best-of-run fitness.

4.2.1 Results

The average best raw fitness of run for each algorithm was tabulated for a given number of evaluations performed. In the case of the even-10-parity problem, since the perfect solution is known, the success rate of each algorithm within 300,000 evaluations was also measured. It is impressive that, according to Figure 6(a), all three HFC techniques consistently outperformed single population and simple multi-population GP for all the population sizes. The success rates almost doubled for population sizes 150 and 250 and they were also much higher for population sizes 400 and 800. Considering the little additional computing effort of HFC to organize the individuals, this significant improvement might be surprising. The streamlined supply of intermediate building blocks provided by the assembly line of multiple fitness levels provided a mechanism for sustainable search without getting stuck at local optima. The sustainable framework also greatly reduced the requirement for large population sizes in GP. The results in Figure 6 showed that the HFC techniques are very insensitive to the total population size, once a minimum size requirement is met. On the other hand, traditional EAs depend strongly on large population sizes to allow them to find good results before convergence occurs. This is just as predicted by the analysis in Section 2.1.2. In this respect, HFC changes the convergent nature of the existing EA framework into sustainable search.

To compare search efficiency, the average number of evaluations used by each method to find the 20 earliest-found perfect solutions was computed, as shown in Figure 7. (The minimal number of perfect solutions found by all the methods was 20). For this relatively easy even-parity problem, it shows that traditional EAs may get good results more quickly but taking a higher risk of premature convergence, while HFCs do not incur much of a penalty in terms of speed (especially for HFC-ATP, in this case) in finding the optimal solution and they exhibit much more robustness in finding optimal

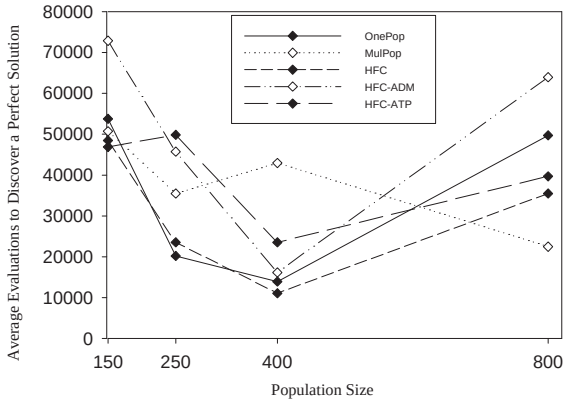


Figure 7: Average evaluations to discover a perfect solution: within the limit of 300,000 evaluations, the average numbers of evaluations of HFC techniques to discover perfect solutions were comparable to those of OnePop and MulPop, although HFC allocates some subpopulations only for supply of intermediate building blocks. This comparison was made by averaging the evaluation counts only of runs that did find perfect solutions, and HFC achieved much higher success rates in finding the perfect solutions (Figure 6). If high evaluation counts were included for unsuccessful runs, HFC methods would look much more favorable.

solutions (Figure 7). For difficult problems, the ”hasty” conventional EAs will have much less chance to make such quick progress.

4.3 Analog Circuit Synthesis for Eigenvalue Placement by Genetic Programming

The use of test problems that arise in the real world poses some difficulty as it sometimes requires other investigators to do a significant amount of work in preparing to run the same problem. However, it is important to study benchmarks that include the complex and rugged fitness landscapes that many real-world problems exhibit. A problem that is often encountered in the design of dynamic systems was chosen for study here-the eigenvalues placement problem. It is a classical ”inverse” design problem, a particular form of analog circuit synthesis problem. In this problem, an analog circuit is represented by a bond graph model (Seo et al., 2002), which includes inductors (I), resistors (R), capacitors (C), and Sources of Effort (SE) (corresponding to voltage sources). The task was to synthesize such a circuit, including its topology and the sizing of its components, such that the eigenvalues of its characteristic equation approximated a pre-specified set of eigenvalues as closely as possible. This is an inverse problem, and its difficulty lies first in the highly rugged and discrete topology search space and then in the highly multi-modal parameter search space of each circuit topology. The high degree of nonlinear epistasis adds additional complexity.

A developmental genetic programming approach for analog circuit synthesis (Koza, 1999) is applied to this synthesis problem (Seo et al., 2002). In this approach, a program tree composed of topology- and component-parameter-establishing functions and terminals is evolved, based on a pre-defined embryo circuit that establishes how solution quality is to be measured. The result of this development process is a fi-

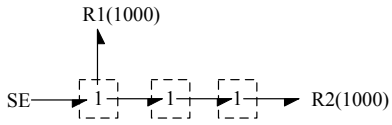


Figure 8: Embryo bond graph for the eigenvalue synthesis problem.

Table 3: Parameter setting for 8-eigenvalue synthesis problem (modified from even-10 parity problem)

8 eigenvalue targets	$-0.1 \pm 5.0j, -1 \pm 2j, -2 \pm j, -3 \pm 0.7j$
total population size	500
max_nodes	400
max_depth	12
Admission thresholds of 9 fitness levels for HFC and HFC-ATP	-100, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90
Subpopulation sizes for subpops 1 to 10 for HFC and HFC-ADM	25,25,25,25, 30,30,30,30,30,250

nal circuit. Function and terminal sets for this problem have been devised, as reported in (Seo et al., 2002). The fitness function is defined as follows: for each GP individual, a circuit is generated with the developmental process. Then its characteristic matrix is established and its eigenvalues are computed. Each solution eigenvalue is associated with the nearest eigenvalues of the target set, and the sum of distance errors between them is calculated, then divided by the order of the target set. A hyperbolic scaling of fitness is then performed as follows, converting the goal from minimization to maximization:

$$Fitness(Eigenvalue) = 0.5 + {}^{1}\Big/{}_{(2 + \sum Error/Order)} \quad (1)$$

Since the minimal error is 0, the highest possible fitness is 1.0.

4.3.1 Experimental Setting

The embryo circuit used in the experiments is illustrated in Figure 8. It has three modifiable sites for further development. The 8-eigenvalue target is set in Table 3. The parameter settings for the five algorithms are the same as for the even-10-parity problem in Section 4.1 except for the changes noted in Table 3. For each experiment, 40 runs were conducted, with 300,000 evaluations in each.

4.3.2 Results

With this complex real-world synthesis problem, the sustainable search capability of the HFC model is expected to enable obtaining much better results without the premature convergence usually found in experiments with standard GP techniques. The best-of-run distance errors from the target eigenvalues are averaged for 40 runs. The standard deviation of the best distance errors is also calculated (Table 4). The simulation results strongly support the claims of the preceding sections. The HFC-ATP algorithm again achieves almost half the average distance error with much smaller variance. This kind of robustness is enabled by the sustainable search capability of HFC.

To evaluate the relative performance of the five approaches, multiple two-tail t-tests are applied. The result is summarized in Table 5. It shows that the difference between the standard GP and the HFC techniques is significant at the 1.2% level.

Table 4: Comparison of Average best-of-run distance error for 8-eigenvalue synthesis problem

Algorithm	OnePop	MulPop	HFC	HFC-ADM	HFC-ATP
Mean Distance Error	0.598	0.64	0.458	0.407	0.278
Std.Dev.	0.287	0.363	0.19	0.228	0.15

Table 5: The t-test results for different approaches to an 8-eigenvalue placement problem: the difference between the standard GP and the HFC techniques is significant at the 1.2% level, while the difference between OnePop and MulPop is not significant.

t-test	MulPop	HFC	HFC-ADM	HFC-ATP
OnePop	0.68	0.012	0.0015	5.8E-09
MulPop		0.0069	0.0010	3.8E-07
HFC			0.28	1.2E-05
HFC-ADM				0.0040

Another criterion useful to compare the sustainable search capability with other methods is to compute the average evaluation number at which the last progress is made in runs of a fixed number (300,000) of evaluation steps. The time from this last progress step to evaluation 300,000, when that time becomes large, likely reflects the "stagnation" of the search, and is expected to increase (i.e., the last progress step is earlier) when premature convergence occurs. Table 6 shows that the HFC technique continues to make progress through evaluation numbers much closer to 300,000, which means that steady progress is achieved throughout the run. In fact, as the step at which last progress is made approaches 300,000, evidence that the run has stagnated at all disappears. On the other hand, the standard GP approaches have stagnated (made no progress) for 70,000 to 100,000 evaluations before evaluation 300,000, on average.

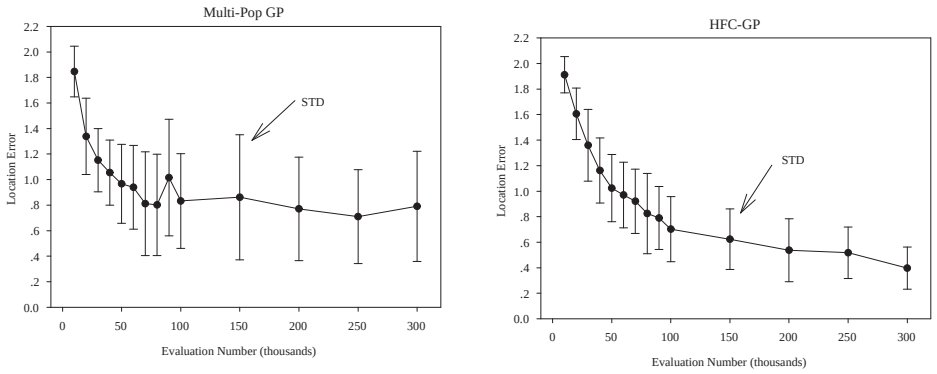
Using the same parameter settings for a more difficult 10-eigenvalue problem $\{-0.1 \pm 5.0j, -1 \pm 2j, -2 \pm j, -3 \pm 0.7j, -4 \pm 0.4j\}$, we calculate the evaluation numbers of last progress for 16 runs of MulPop and HFC-ATP, each run with 400,000 evaluations. The difference is even more striking - 263,400 for MulPop with standard deviation 106,639, but 392,320 with HFC-ATP with standard deviation 12,534. This result clearly demonstrates that for this more difficult problem, HFC techniques can achieve much more robust search and continue to make steady progress.

The robustness and the sustainable search capability are also examined by investigating the relationship between the average of the best eigenvalue location errors over 40 runs and the maximum number of evaluations, ranging from 10,000 to 300,000. Here HFC-ATP is compared only to MulPop. Experiments for each maximum evaluation limit are run with different random seeds. The experimental parameters are the same as in the experiment described above except that for HFC, the subpopulation sizes for

Table 6: Comparison of stagnation times for the 8-eigenvalue placement problem: in the HFC model, last progress is made much closer to the evaluation limit of 300,000, and it displays a much smaller tendency to converge.

	OnePop	MulPop	HFC	HFC-ADM	HFC-ATP
Mean step at which last progress is made	232,300	204,300	262,500	283,500	283,100
Standard Deviation	64,300	70,700	55,200	18,500	34,900

subpops 1 to 10 are 30,30,40,40,50,50,50,50,60,100. For MulPop, the migration frequency is every 5 generations, with 10% migration of individuals at that interval, and with a copying-type migration (migrants appear in new population, but are not removed from the donor population). The results are presented in Figure 9. The robustness of HFC is demonstrated by its much lower standard deviation of location errors. The sustainable search capability is shown by its continuing progress given more evaluations, while for MulPop, after a certain threshold number of evaluations, no progress is made in any reasonable number of additional evaluations.



(a) Multi-population GP (b) HFC-GP

Figure 9: Comparison of the average best-of-run location errors vs. number of evaluations for multi-population-GP and HFC-GP. Error of 0 is the optimal value. HFC achieves much more robust search with continuous progress, reflected by its much smaller standard deviations. Multi-population-GP is less reliable, displaying large standard deviations of the location error. It also has the tendency that beyond 300 K evaluations, there will not be much progress, while for HFC-GP, sustainable progress appears.

5 A Sustainable Genetic Algorithm: the Quick HFC (QHFC)

As a generic framework for sustainable evolutionary computation, HFC is representation-independent and does not rely on the building block hypothesis as stated in Holland's GA theory (2000). Actually, there is no established schema theory or even consensus on the definition of building blocks in the GP community (O'Reilly & Oppacher, 1995; Rosca, 1997). The sustainable search capability enabled by HFC on the GP problems above with variable-length, highly nonlinear interaction and coupling of program chunks demonstrates the effectiveness of HFC to ensure sustainable search. The question that naturally arises is whether HFC will work well in the (typically) simpler GA domain-particularly with a binary representation, where extensive theoretical studies and effective techniques exist and clear comparisons can be made. For example, how does HFC compare with other diversity-maintenance techniques in GA? Is it possible to combine HFC with other methods to produce a better sustainable search algorithm?

This section describes how the hierarchical fair competition principle can be used to design a surprisingly simple, effective and robust genetic algorithm, named Quick

HFC (QHFC), for binary GA problems. This is achieved by combining the ideas of the HFC and AHFC algorithms described in Section 3 with a simple but effective diversity-maintenance technique: the deterministic crowding (DC) algorithm (Mahfoud, 1992). A difficult standard GA test problem, HIFF, is used to show how the often-used DC diversity-maintenance technique fails, how its performance depends strongly on the population size and how HFC can be used to make it more robust in terms of both population size and search capability. The performance of QHFC is compared with a standard GA enhanced by deterministic crowding (DCGA) and DCGA with a multi-partial-reinitialization method (DCGA+MR), illustrating that the HFC principle can transform the convergent DCGA into a much more robust and capable search algorithm. It turns out that QHFC has made it possible to use much smaller population sizes and many fewer evaluations to solve rather large instances of the HIFF problem.

5.1 Designing a Sustainable Genetic Algorithm Based on the HFC principle

A straightforward implementation of the HFC principle for a binary genetic algorithm problem simply requires transforming HFC-GP and AHFC-GP in Section 3 into HFC-GA or AHFC-GA by replacing the tree representation with a binary representation. However, preliminary experiments showed that these two algorithms don't work in that form for binary GA problems. Close examination reveals the reason and provides further insight into the HFC principle for sustainable evolution. It is well known that one of the significant differences between GA and GP is that the loss of diversity in a binary GA with crossover and small mutation rates is much faster than that in GP. Crossover of two identical binary chromosomes produces two identical offspring, while in GP, crossover of two identical trees usually produces distinct offspring. As a result, the levels of HFC-GA converge very quickly for binary problems because of the fixed allocation of breeding probabilities to levels used in HFC-GA, thereby violating the HFC principle that the lower level should be able to ensure sustainable export of individuals (thus maintaining diversity) into higher levels; otherwise the higher level will just "die out," deprived of its supply of new and viable genetic material, as traditional GA and GP do.

Now the question is reduced to how to ensure sustained and useful diversity for all intermediate levels. One solution is to allocate much more breeding opportunity to lower levels. But this will reduce the relative effort of exploitation of higher levels. A better strategy is to do more breeding in lower levels to improve their diversity only when the higher levels detect that lower levels have converged to some extent, and then allocate them more breeding opportunities to let them catch up. This requires an effective mechanism to detect the convergence of a subpopulation. There are many mechanisms to measure the diversity of the population (Wineberg & Oppacher, 2003) and some are used in guiding the balance of exploration and exploitation (Ursem, 2002). Most of these methods require the calculation of genotypic distances between all possible pairs in the population. Although this calculation can be reduced to O(n) time complexity (Wineberg & Oppacher, 2003), it is hard to set the diversity thresholds for all levels in HFC, and diversity measured in this sense may not be the useful diversity needed for good evolution (remembering that adding random individuals to intermediate-fitness populations, for example, increases diversity but hardly improves search performance). Based on this analysis, a new, practical mechanism is introduced into the original HFC framework to support the continuing potency of each level in the HFC framework. Potency here is defined as the capability of a fitness level in HFC to produce offspring with fitness high enough for export to higher HFC levels. This

mechanism for maintaining the potency of each level except the top level works as follows: starting from the level below the top level, breeding is conducted successively in each level, using steady-state breeding methods, while tracking the number of offspring produced that are eligible for promotion (migration to the next-higher fitness level). If a given number of promotable offspring are not produced within a specified number of evaluations at a given level, then a "catch-up" procedure is conducted: a specified fraction of that level's individuals is replaced by individuals taken from (i.e., removed from) the next lower level, and popsize genetic operations and evaluations are performed. Then, in turn, the openings created at the next lower level are immediately filled with individuals removed from the level below that, etc., until, at the lowest level, the openings are filled by new randomly generated individuals (however, further genetic operations and evaluations are not performed as part of this "ripple down" filling of openings). This "double loop" procedure assures that each level, before it next breeds, has either recently produced individuals worthy of promotion to the next level or has received new individuals from the next lower level, thus ensuring its potency to export higher-level individuals.

This mechanism for sustaining the potency of search does not require evaluating any measure of the distance among genotypes or phenotypes, and could also be applied to GP and other sorts of problems. However, the particular QHFC applied here to a GA, a "good" distance function was available, so was used in order to demonstrate how QHFC can transform a "classical" GA with deterministic crowding into a more robust and efficient algorithm. The other difference introduced, in order to facilitate the QHFC application, was to use a generational GA at the top level and a steady-state GA at the remaining levels for breeding.

The main QHFC algorithm is summarized in Figure 10 along with its subroutines in Figure 11. Compared with HFC-GP and AHFC-GP, QHFC has many fewer parameters to specify, and the admission thresholds are automatically adjusted.

5.2 GA Test problem and the Genetic Algorithms compared

We test QHFC with a well-known standard test problem in GA: the hierarchical if-and-only-if (HIFF) problem (Watson et al., 1998), an instance of hierarchically decomposable functions (HDFs). In this problem, a binary string of length 2^K is to be evolved, where K is the number of levels in the hierarchy. The fitness of a binary string is defined by the recursive function shown below

where B is a block of bits $(b_1, \ldots, b_n)$, $|B|$ is the length of the block (n), b_i is the i-th element in block, B_L and B_R are the left and right half substrings.

Using HIFF as the test problem is motivated by the fact that diversity maintenance is critical to prevent premature convergence and to solve it successfully (Watson & Pollack,1999). It is not a separable problem because of the strong non-linear interactions of its building blocks at all levels. Several approaches have been applied to this problem including deterministic crowding, fitness sharing (Watson & Pollack,1999, 2000) and several "competent GAs" with linkage learning mechanisms (Pelikan & Goldberg, 2001), providing a good basis for comparison.

In the experimental study here, QHFC is compared with the generational deterministic crowding GA (DCGA) and its multi-partial-reinitialization version (DCGA+MR) (see Figure 12). DCGA (without multi-partial-reinitialization) is implemented simply by setting *percentRefill*=0 in Figure 12.

QHFC-GA Algorithm

Parameters

Total population size: $|P_t|$, bit mutation rate: p_m
L: number of subpopulations (levels) of QHFC
γ: size factor parameter, the ratio of higher level archive size w.r.t next lower level archive size $|P_{k-1}| = |P_k|.\gamma$
breedTopFreq: number of generations to breed top level between potency testing of lower levels (via breeding)
detectExportNo: number of individuals from a level that must be promoted for the level to be considered potent catchupGen: maximum evaluations in any but top level,normalized by level's popsize, for potency test
percentRefill: percentage of this level's popsize to import from next lower level when there is no progress in the top level,or when lower levels fail potency test (do not furnish detectExportNo qualified immigrants within specified number of evaluations)
noprogressGen: maximum number of generations without any fitness progress in top level before triggering importing of percentRefill individuals from next lower level

Main Procedure

1.Initialization
  randomly initialize and evaluate the HFC subpopulations
  calculate the average fitness of the whole population and set it as the admission fitness of the bottom level, $f_{\min}$, which is fixed thereafter
  remove individuals with fitness less than $f_{\min}$, and equally distribute the rest of the individuals among the levels, according to fitness, thereby determining the admission threshold of each level
  generate random individuals to fill the openings in each archive
2. While termination_condition is false
  breed the top level for *breedTopFreq* generations using generational deterministic crowding and applying mutation after each crossover
  if no progress on best fitness of the whole population for *noprogressGen* generations,
    call **import_from_below**, but ensuring the best individual is not replaced
  if average fitness of top level $> 2 * f_{adm}^{L-1} - f_{adm}^{L-2}$, adjust admission thresholds by evenly allocating fitness range to each level:
    $f_{adm}^k = f_{\min} + k(f_{\max} - f_{\min})/L$, for $k = 0$ to $L-1$
    where f_{adm}^k is the admission fitness of level k,
    $f_{\max}$ is the maximum fitness of the whole population
  //potency testing
  for each level from $L-2$ to 0
    call **do_potency_testing**
    if not succeed
     call import_from_below to replace (at random) percentRefill percentage of the current level
     breed one generation at this level
    endif
  end for
end while

Figure 10: QHFC Algorithm

5.3 Experimental Results

The following experiments compared the performance of QHFC, DCGA, DCGA with multi-partial-reinitialization for the 128-bit and 256-bit non-shuffled HIFF problems (the 64-bit HIFF problem was not addressed because it is so easy that any conclusion from it may be misleading). In all three algorithms compared, simple two-point

```
Procedure  do_potency_testing (l)
l is the level for potency testing

catchup_evaluation ⟵ 0
exportedIndividual ⟵ 0
while catchup_evaluation < catchupGen*  and exportedIndividual< detectExportNo
  randomly pick two individuals from level
  crossover, mutate, and evaluate
  if fitness of offspring >f_adm^(l+1),
    promote it (them) to level (replacing randomly any but the best individual
    or other individuals just promoted) and call  import_from_below to
    replace its (their) closest parent(s)
    exportedIndividual ⟵ exportedIndividual +1
  else
    do deterministic crowding with the 4-member family
  endif
end while
if  fail to promote detectExportNo individuals
  return not success
else
  return success
Procedure end

Procedure import_from_below(l, nImport, victimList)
l: the level into which to import new individuals from next lower level
nImport: the number of individuals to import from next lower level
victimList: a list of indices  of individuals which will be replaced
  by the imported new individuals

if l = 0
  randomly generate nImport new individuals and import into (lowest) level l
else
  randomly choose nImport individuals from level l−1 to replace individuals in
  victimList. If victimList is empty, randomly choose victim individual from
  current level. Put the indices of the new immigrant individuals from level
  l−1 into the newVictimList of level l−1, whose openings will eventually be
  filled with individuals from level l−2 (this assures the replacement of
  individuals removed from level l−1).
  call  import_from_below ( l−1, nImport, newVictimList)
Procedure end
```

Figure 11: Subroutines of QHFC Algorithm

crossover with mutation was used, where mutation is applied on offspring immediately after crossover. All methods used a bit mutation rate of 0.0075 for both 128-bit HIFF and 256-bit HIFF. For the DCGA-MR, the *percentRefill*=0.25, *maxNoProgressGen*=10. For QHFC, in all the experiments, with different population sizes and different problem sizes, the following set of parameters was used.

L: 5	γ: 0.7	*breedTopFreq*: 2	*detectExportNo*: 2
percentRefill: 0.25	*catchupGen*: 20	*noprogressGen*: 10	

All three algorithms were experimented using a series of population sizes from 100 to 4000, all with maximum number of evaluation 1,000,000 and each experiment with 30 runs. The result is illustrated in Figure 13 and Figure 14.

Figure 13 clearly shows that for the simpler 128-bit HIFF problem in which diversity maintenance is not a big issue, using a simple GA with deterministic crowding could solve the problems quite well given a sufficient population size. However, for real-world problems, correctly estimating the required population size is not easy and

Figure 12: Deterministic Crowding GA with/out multi-partial-reinitializations

DCGA(-MR) Main Procedure
percentRefill: the percentage of the population that will be replaced by random individuals.
maxNoProgressGen: maximum number of generations with no progress on the maximum fitness of the population before we partially reinitialize the population

1.Intialize the population randomly
2.Repeat until stopping condition
randomly group individuals into pairs
do crossover and mutation for each pair of individuals p1, p2 to produce two offspring c1, c2
according to **pairing rule**, each offspring is paired with and compete against one parent.
if the fitness of the offspring is better than that of its corresponding parent, the parent gets replaced
check if maximum fitness has not been updated after *maxNoProgressGen* generations, we replace *percentRefill* percent of the population with randomly generated individuals.

Pairing rule: if $H(p1,c1)+H(p2,c2) < H(p1,c2)+H(p2, c1)$ then pair p1 with c1, and p2 with c2, otherwise pair p1 with c2 and p2 with c1, where H. returns the Hamming distance between two individuals.

population sizing theory can be unreliable and difficult to apply (Goldberg, 2003). Also, the idea of using large populations to maintain diversity is not a scalable approach, as illustrated in the 256-bit HIFF problem, where a GA with deterministic crowding fails in half 15 of 30 runs even with a population size of 4000, while QHFC can still achieve 25 successful runs out of 30 with a population size 100. This robustness is derived from the sustainable diversity promoting mechanisms in HFC, where genetic material from lower levels provides the non-exhaustible diversity. This is a key feature that distinguishes HFC from other diversity maintenance techniques such as niching, restricted mating, island or multi-population models, pygmies and civil servants, and species-conserving GA, as mentioned in Section 1. As regards the well-known partial-reinitialization approach, Figure 14 shows that DCGA+MR performed the worst among the three methods, simply by wasting evaluations on hybrids of high-fitness individuals with random individuals, where high-fitness individuals with strongly coupled components cannot typically be improved by inserting junk alleles.

The advantage of HFC is not limited to robustness of search even with small population sizes; it also solves many problems with fewer evaluations, as illustrated in Figure 14. For simple problems like 128-bit HIFF, a GA with deterministic crowding wins by using fewer evaluations. This is achieved by allocating all breeding opportunities to high-fitness individuals maintained in the population, while QHFC loses by allocating a portion of its evaluations to lower levels to ensure sustainable diversity. However, one can see that this preventive measurement does not incur a very large penalty in terms of number of evaluations. The strategy of allocating a fraction of the evaluations to lower fitness levels is rewarded by increased robustness and the capability to solve more difficult problems, as shown for the 256-bit HIFF problems in Figure 14. With a population size of 4000, a traditional DCGA with/without multi-partial-reinitialization found the optimal solutions in fewer than 15 of 30 runs (Figure 13). The successful runs were obtained using three times the number of evaluations used by QHFC with a pop-

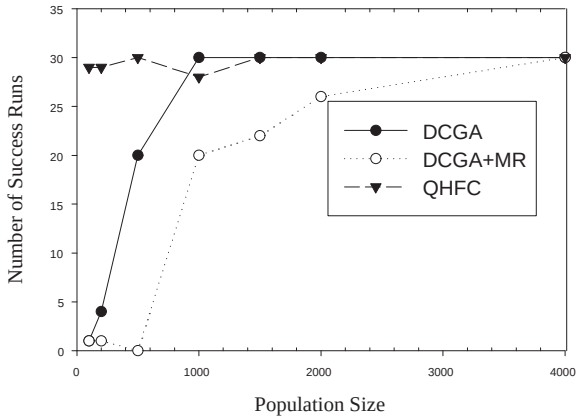


Figure 13: Comparison of QHFC with DCGA with/without multiple partial reinitializations in terms of robustness of the search capability and robustness in terms of population size. For the simple 128-bit HIFF problem, DCGA can achieve robust search if the population size is sufficiently large (¿=1000). QHFC, instead, can do very robust search from population sizes 100 to 4000. For the difficult 256-bit HIFF problem, a population size of 4000 for DCGA can only find the solution in half the 30 runs, while QHFC can solve it very robustly while still using population sizes from 100 to 4000. DCGA+MR apparently suffers from the disturbance of imported random individuals and works worse in both case than DCGA.

ulation size of 2000, which also succeeded in all 30 runs. The influence of population size in QHFC is interesting. With an extremely small population size of 100, QHFC can still achieve quite robust search and find the optimal solution in 25 runs out of 30, but with a much larger variation in the number of evaluations. With a larger population size (1000-2000), QHFC can find the optimal solutions within more consistent numbers of evolutions. When population size is set larger than necessary, QHFC takes more evaluations to find the optimal solutions than with smaller population sizes, but still many fewer than DCGA needs. This increased average number of evaluations to find optimal solutions with overly large population size is caused by spending too many evaluations in lower levels, since the same set of parameters was used for all experiments. One important lesson to draw here is that QHFC is quite scalable for the HIFF problem, in that the required population size and the number of evaluations needed do not vary much for population sizes from 100 to 4000. QHFC is also seen to be quite scalable in terms of problem size, with about 160,000 evaluations needed for 128-bit HIFF and about 234,000 for 256-bit HIFF.

Results on HIFF problems have also been reported in (Watson & Pollack, 1999), where a GA with domain-knowledge-based fitness sharing and a population size of 1000 were used. It showed that even this fitness sharing method with considerable domain knowledge fails on the 128-bit and 256-bit problems, while QHFC with population size 100 and 200 consistently and reliably solved the 128-bit and 256-bit HIFF problems. The results of QHFC were also compared with those of the somewhat complicated hierarchical Bayesian Optimization Algorithm (Pelikan & Goldberg, 2001, 2003),

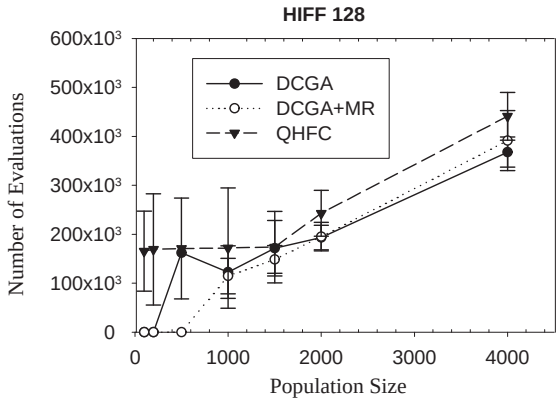


Figure 14: Comparing QHFC with DCGA with/without partial reinitializations in terms of average number of evaluations needed to find the optimal solutions. For the simple 128-bit HIFF, DCGA wins with slightly fewer evaluations, but HIFF wins by being able to use much smaller population sizes. For the 256-bit HIFF, QHFC wins in terms of both robustness of search (finding the optimal solutions in 25-29 runs out of 30 for the population sizes of 100 or higher) and number of evaluations (with population size 2000, QHFC succeeded 29 times, using only about one-third the evaluations of DCGA (with a population size 4000 and with only half the runs successful).

one of the competent GAs with explicit linkage learning mechanisms. The optimal result of hBOA on the 128-bit HIFF was about 26,000 evaluations, with a standard deviation of about 12,000, and on the 256-bit HIFF, was about 88,000 evaluations, with a standard deviation of about 10,000. QHFC, without any linkage learning mechanism, achieved competitive results with much smaller population sizes (down to 100), reliably. For the 128-bit HIFF, QHFC used about 160,000 evaluations, and used about 234,000 for 256-bit HIFF (averaged over 30 runs). Clearly QHFC, as a linkage-blind approach, uses many more evaluations. But about half of 30 QHFC runs succeeded within 85,000 evaluations for 128-bit HIFF and within 153,000 evaluations for 256-bit HIFF. The discrepancy in number of evaluations needed compared with hBOA is less with larger HIFF problems - the QHFC appears to be scaling with a lower slope than the hBOA on this problem. And it is hard to do a precise comparison: the results reported in (Pelikan & Goldberg, 2001) are the best results after empirically tuning the population sizes of hBOA to minimize the number of evaluations for each problem instance, whereas the same set of parameters was used for all QHFC experiments here, independent of the population size and problem size. QHFC may, in fact, be tunable to require fewer evaluations than reported here. QHFC has also been tried on some other hierarchical deceptive functions (to be reported elsewhere), again obtaining very good results.

However, one fact to be pointed out is that the performance of QHFC here does not apply to shuffled HIFF problems, while hBOA can solve both shuffled and non-shuffled HIFF reliably. As a shuffled HIFF problem is not solvable by GAs with a simple crossover operator (Watson & Pollack, 2000), i.e., the type used with QHFC here, the conclusion is that QHFC may greatly improve the robustness and scalability of various EAs, so long as the operators used are appropriate to the given problem. Based

on the scalability and robustness of QHFC on the hierarchical HIFF problem with tight linkage, it seems that QHFC can achieve a reasonable level of hierarchical problem solving capability. Since the simple crossover operator is the bottleneck for QHFC to solve shuffled HIFF problems, one interesting idea is to combine QHFC with BOA and to see if this hybrid can compete with hBOA (Pelikan & Goldberg, 2003). To do that, one only needs to replace the crossover operator in QHFC here with the Bayesian model building method in BOA. The combination of QHFC with hBOA also appears extremely promising.

6 Analysis and Discussion of the HFC model

6.1 Examining HFC in Terms of Diversity Maintenance and Incremental Evolution

Although HFC does not manage diversity per se as an approach to ensuring sustainable evolution, HFC essentially provides several effective ways that do maintain diversity. First, the continuous introduction of random individuals into the lowest level subpopulations ensures the supply of diversified genetic material, which can then be exploited in the "fair competition environment" of low-fitness-level subpopulations. Thus, HFC implements the equivalent of an effective multi-start or re-initialization mechanism on a continual basis. HFC also maintains diversity by decreasing the risk of dominance by the earliest-discovered high-fitness individuals. At each fitness level, HFC ensures that the competition of individuals is fair. Only individuals with comparable fitnesses are allowed to stay in a certain fitness level, and new competitive immigrants from lower levels are continuously incorporated. If a new offspring turns out to be extraordinarily fit, it will immediately emigrate to a yet-higher fitness level. So, essentially, the local selection pressure at each specific level is low while the global selection pressure is maintained by the stratified structure of fitness levels.

The structure of HFC is especially suited for incremental evolution, where the solutions usually need a steady-state developmental process for refinement. In HFC, this process is realized by the climb of developing individuals from lower fitness levels to higher fitness levels. Along the way, they compete only with developmentally similar individuals, assuring them sufficient time to "mature."

6.2 HFC's Stepping Stone Supply vs. Explicit Linkage Learning or Modular Approach

HFC works by ensuring continuous availability of "stepping stones" - individuals at a hierarchy of fitness levels and accompanying levels of coadaptation of genetic material, populating the range from randomly generated individuals to the best yet discovered. It assures the descendants of newly generated individuals time for "maturation" in the presence of other individuals of similar fitness, throughout the fitness hierarchy, avoiding strong selection pressure or competition with more fit individuals. HFC does not require the existence of building blocks to work well, but takes advantage of them implicitly if they are present. This is in contrast with the explicit building block exploitation in messy GA and other linkage-learning GAs. It is also different from the half-blind building block exploitation in the ADF mechanism of GP. However, the apparent effectiveness of HFC justifies its role as a generic framework for sustainable evolutionary search, which is not guaranteed by linkage learning and other modular approaches alone. Coupling the sustainable search capability of HFC with the explicit building-block-discovery capability of linkage learning is expected to achieve a synergistic effect for suitable problems as discussed in Section 5.

6.3 HFC, Restarting, and the Short-Run-or-Long-Run Dilemma

HFC provides a good solution to the short-run-or-long-run dilemma (Luke, 2001; Goldberg, 1999). With a given number of total evaluations available, it is hard to decide whether it is better to run multiple (independent or serial) epochs with small population sizes or to run a single epoch with a large population size. Running with large population sizes may lead to convergence too quickly due to unbalanced scaling of building blocks (Goldberg, 1999) while small population sizes may not provide sufficient building blocks. Use of a rejuvenating operator still suffers from lack of intermediate building blocks, as discussed in Section 2.2.1. However, HFC, which is not very sensitive to the population size, makes it possible to use much smaller population sizes, while preserving the capability for sustainable search. It is a natural mixing of implicit parallel and serial processing (Goldberg, 1999). Compared with multi-epoch EAs or other restarting techniques, HFC saves a huge amount of computational effort by reusing building blocks.

6.4 HFC and Other Schemes with Hierarchical Organization of Subpopulations

Some other EAs also employ a hierarchical organization of subpopulations. The Pyramid GA (Aickelin, 2000) used lower-level subpopulations to optimize single aspects (of the objective function) of the problem while higher levels imported individuals from lower levels for further recombination. Hsu et al. (2002) used similar ideas (LLGP) for agent evolution. The injection island GA (or iiGA) (Lin et al., 1994) used a hierarchical organization to implement multi-resolution or other heterogeneous, layered search. As a sustainable search framework, HFC is conceptually different from these approaches, which still suffer from the convergent nature of the traditional EA framework. In HFC, the hierarchical organization of subpopulations by fitness provides a mechanism for maintaining the intermediate stepping stones necessary for sustainable search.

6.5 The HFC Model as a Generic Framework for Sustainable Evolutionary Search

While HFC is attractive for the performance improvement it offers, it is more strongly intended as defining a generic framework for continuing or sustainable evolutionary search. It is compatible with most existing techniques, such as Species Preserving GA (Li et al., 2002), fitness sharing and crowding, linkage learning, etc. It is a macro-level structure or framework in which existing techniques can be applied at each fitness level as needed. Moreover, the essential idea of the HFC model is the capability to maintain multiple levels of intermediate-fitness stepping stones to allow sustainable search. In this sense, the explicit hierarchical organization may not always be necessary, although stratification of breeding by fitness level is another useful feature enabled by that organization. The current HFC model may become less efficient in problems where the genetic operators used are not suitable and thus there is no gradient information to guide the exploration for building blocks. For problems requiring assembly of widely-spread schemata (i.e., exhibiting strong epistasis at sites widely distributed on the chromosome), such as the shuffled HIFF problem of (Watson & Pollack, 2000), HFC with a traditional GA may not be enough for effective search. Coupling HFC with linkage learning techniques may prove to be necessary in such cases, because ordinary crossover is too disruptive to the important schemata.

7 Conclusions and Future Work

7.1 Conclusions

In this paper, the Hierarchical Fair Competition (HFC) model is proposed as a generic framework for a sustainable evolutionary search for the solution to complex problems. This sustainable search capability is achieved by ensuring sustained access to "stepping stones" of intermediate fitness levels in a sequence of fitness-stratified breeding pools. For problems exhibiting building block structure, these pools act as repositories containing a succession of increasingly large and fit assemblies of such building blocks. Understanding the dynamics of HFC provides a new insight into the premature convergence of EAs as well as the structure of an EA itself. Instead of trying to escape local attractors from relatively highly converged/evolved populations containing individuals having strongly coupled components, a better strategy is to ensure sustainable search by allowing the emergence of new optima in a bottom-up manner.

The convergent nature of the extant EA framework is examined in terms of the effect of the continuous increase of the average fitness of the population. This inherent property makes the horizontal-spreading diversity maintenance techniques, such as niching, inefficient in the case of complex, highly multimodal problems, as is common in the program spaces of GP. An argument is made that it is important that, at any time, the population contain some number of "stepping stones" - intermediate-fitness individuals, continually updated by synthesis from lower-level stepping stones, and managed so as to assure that they are capable of producing higher-fitness individuals.

Inspired by the fair competition principle in societal and economic systems, the HFC model consists of three components: subpopulations structured hierarchically according to fitness, a random individual generator which is active throughout the evolutionary process, and a mandatory, universal migration policy from lower fitness levels to higher fitness levels. The assembly-line structure of HFC enables it to search the problem space with steady-state progress by maintaining a hierarchy of stepping stones, spanning the space from randomly generated individuals to the most fit individuals discovered. HFC is representation independent and thus applicable to binary-coded GAs, integer- or real-valued GAs, tree-based genetic programming, etc. This model shows good characteristics in terms of diversity maintenance, incremental evolution and building block exploitation.

HFC and two variants were evaluated with a standard GP benchmark problem (even-10-parity) and a complex real-world analog circuit synthesis problem. Another easy-to-use HFC algorithm, QHFC, was introduced and applied to a difficult benchmark GA problem - HIFF. It solves non-shuffled HIFF with many fewer evaluations and much smaller population sizes as compared to deterministic crowding, contrasting sharply with prevailing GA population sizing theory (Harik et. al., 1999), which usually assumes a single-thrust GA model as discussed in Section 2.1.3. In all of these experiments, HFC, with its sustainable search capability, achieved much better robustness and found better solutions in terms of average best fitness, and using fewer evaluations, than other GA and GP techniques attempting to avoid premature convergence. The t-test for the analog circuit synthesis problems and the detailed comparison of QHFC and DCGA with or without multi-partial-reinitialization demonstrated the significant performance difference between HFC and a typical EA framework.

As a generic framework, HFC constitutes a useful technique that improves existing EAs. In addition to the QHFC first reported here (combining deterministic crowding and HFC), a multi-objective extension of HFC has also been studied (Hu, et. al., 2003).

Additionally, a single-population-based "continuous" (CHFC) structure has also been proposed and tested, and appears very promising (Hu, Goodman et al., 2003).

7.2 Future work

There are several important improvements needed in the HFC sustainable EA model. The first is to introduce some control mechanisms to manage a persistent type of admission buffer to assure that immigrant stepping stones are sufficiently exploited. While diversity-promoting schemes are used in QHFC to avoid convergence of each fitness level, making use of a genotypic distance measure, it may be possible to eliminate the distance function, which would make QHFC immediately useful for genetic programming and many other problems in which a distance function is problematic to define. Another important task is to couple linkage learning techniques with HFC to attack large complex problems with arbitrary linkage patterns. Combination of HFC with hBOA appears promising. As HFC is a representation-independent solution to the premature convergence problem, hybridization of HFC with evolution strategies may also achieve synergistic effects and make it possible to find solutions to more challenging problems.

Acknowledgment

The first author would like to thank his previous advisor, Professor Shuchun Wang of Beijing University of Aeronautics & Astronautics of China for encouraging him to use societal and biological models in AI research. We are also grateful to Dr. Riccardo Poli for the sub-machine-code for the even parity problem. This work was supported by the National Science Foundation under contract DMI 0084934.

References

Aickenlin, U. and Dowsland, K. (2000). Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3(3):139–153.

Angeline, P. J. and Pollack, J. B. (1994). Coevolving high-level representations. In *Proceedings of Artificial Life III*. Addison-Wesley.

Bonner, J. T. (2000). *First Signals : The Evolution of Multicellular Development*. Princeton University Press.

Cobb, H. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In *Proc. Fifth Int. Conf. on Genetic Algorithms*, pages 523–530.

Darwen, P. (1996). *Co-evolutionary Learning by Automatic Modularisation with Speciation*. PhD thesis, University of New South Wales.

Eby, D., Averill, R. C., Goodman, E., and Punch, W. (1999). Optimal design of flywheels using an injection island genetic algorithm. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, 13:389 –402.

Goldberg, D. E. (1989). Sizing populations for serial and parallel genetic algorithms. In *Proc. Third Internat. Conf. on Genetic Algorithms*, pages 70–79.

Goldberg, D. E. (1999). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. Technical report, University of Illinois at Urbana-Champaign.

Goldberg, D. E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA.

Goldberg, D. E., Deb, K., and Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. In Manderick, R. M. . B., editor, *Parallel Problem Solving From Nature II*, pages 37–46. Amsterdam: North-Holland.

Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In *Fifth Int. Conf. on Genetic Algorithms*.

Harvey, I. (1992). Species adaptation genetic algorithms: A basis for a continuing saga. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354, Cambridge, MA. MIT Press/Bradford Books.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224.

Holland, J. H. (2000). Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):373–391.

Hsu, W. H. and Gustafson, S. M. (2002). Genetic programming and multi-agent layered learning by reinforcements. In Langdon, W. B. e. a., editor, *Proceeding of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Hu, J., Goodman, E. D., and Seo, K. (2003). Continuous hierarchical fair competition model for sustainable innovation in genetic programming. In Riolo, R. and Worzel, W., editors, *Genetic Programming Theory and Practice*, chapter 6. Kluwer Publishers, Boston, MA.

Hu, J., Goodman, E. D., Seo, K., and Pei, M. (2002). Adaptive hierarchical fair competition (ahfc) model for parallel evolutionary algorithms. In *Proc. 2002 Genetic and Evolutionary Computation Conference*. New York.

Koza, J. H. (1994). *Genetic Programming II*. MIT Press, Cambridge, Massachusetts.

Koza, J. H., Bennett III, B. N., Andre, D., , and Keane, M. A. (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. MIT Press, Cambridge, Massachusetts.

Koza, J. R., Keane, M. A., Yu, J., Bennett III, F., and Mydlowec, W. (2000). Automatic creation of human-competitive programs and controllers by means of genetic programming. *Journal of Genetic Programming And Evolvable Machines*, 1:121–164.

Li, J. P., Balazs, M., Parks, G. T., , and Clarkson, P. J. (2002). A species-conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation,*, 10(3):207–234.

Lin, S.-C., Goodman, E. D., and Punch, W. (1994). Coarse-grain parallel genetic algorithms: Categorization and new approach. In *IEEE Conf. on Parallel and Distribributed Processing*, volume 11.

Luke, S. (2001). When short runs beat long runs. In et al., L. S., editor, *Proc. 2001 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Mahfoud, S. (1992). Crowding and preselection revisited. In *Parallel Problem Solving from Nature*, pages 27–36. North-Holland.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

Pelikan, M. and Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. In et. al., I. L. S., editor, *Proc. GECCO–2001 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann.

Poli, R. and Page, J. (2000). Solving high-order boolean parity problems with smooth uniform crossover, sub-machine-code gp and demes. *Genetic programming and evolvable machines*, 1:37–56.

Potter, M. and De Jong, K. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

Rosca, J. and Ballard, D. (1994). Hierarchical self-organization in genetic programming. In *11th Internat. Conf. on Machine Learning*. Morgan Kaufmann.

Ryan, C. (1996). *Reducing premature convergence in evolutionary algorithms.* PhD thesis, National University of Ireland.

Seo, K., Hu, J., Fan, Z., Goodman, E., and Rosenberg, R. (2002). Automated design approaches for multi-domain dynamic systems using bond graphs and genetic programming. *The International Journal of Computers, Systems and Signals*, 3:55–70.

Simon, H. (1973). The organization of complex systems. In Pattee, H. H., editor, *Hierarchy Theory - The Challenge of Complex Systems*. George Braziller Publisher, New York.

Stanley, K. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127.

Torresen, J. (2002). A scalable approach to evolvable hardware. *Genetic Programming and Evolvable Machines*, 3:259–282.

Tsutsui, S. and Ghosh, A. (1998). Search space division in gas using phenotypic squares estimates. *Information Sciences*, 109(1-4):119–133.

Vassilev, V. and Miller, J. (2000). Scalability problems of digital circuit evolution. In *Proc. 2nd NASA/DOD Workshop on Evolvable Hardware*. Los Alamitos, California.

Waddington, C. (1942). The canalization of development and the inheritance of acquired characters. *Nature*, 150:563–565.

Watson, R. and Pollack, J. (1999). Hierarchically-consistent test problems for genetic algorithms. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proc. 1999 Congress on Evolutionary Computation*.

Whitley, D., Mathias, K., and Fitzhorn, P. (1991). Delta-coding: An iterative search strategy for genetic algorithms. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 77–84.

Yao, X. and Higuchi, T. (1998). Promises and challenges of evolvable hardware. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 28(4):55–78.