The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination

Larry J. Eshelman

Philips Laboratories,
North American Philips Corporation,
345 Scarborough Road,
Briarcliff Manor, New York 10510

Abstract

This paper describes and analyzes CHC, a nontraditional genetic algorithm which combines a conservative selection strategy that always preserves the best individuals found so far with a radical (highly disruptive) recombination operator that produces offspring that are maximally different from both parents. The traditional reasons for preferring a recombination operator with a low probability of disrupting schemata may not hold when such a conservative selection strategy is used. On the contrary, certain highly disruptive crossover operators provide more effective search. Empirical evidence is provided to support these claims.

Keywords: cross-generational competition, elitist selection, implicit parallelism uniform crossover, incest prevention, restarts.

1 Introduction

Any search algorithm that operates via a reproduction-recombination cycle on a population of structures can be called a genetic algorithm (GA) in the broad sense of the term. Since Holland's seminal work (1975), however, in order for any such algorithm to qualify as a genetic algorithm in a more restricted sense, it must be

shown that its search behavior displays what Holland calls *implicit parallelism*. Thus, it is incumbent upon anyone who claims that some new population-based search algorithm is a genetic algorithm in the more restricted sense to make a case for its implicit parallelism.

In this paper I argue that CHC, a nontraditional genetic algorithm, does indeed display implicit parallelism. Furthermore, I argue that some of the features that on the surface seem to disqualify it as a true GA not only do not so disqualify it, but in fact make it more powerful than the traditional GA.

I shall use the following outline of a genetic algorithm as a point of reference to describe how CHC differs from a traditional GA (based on Grefenstette and Baker, 1989, but generalized):

```
procedure GA
begin
   t = 0;
   initialize P(t);
   evaluate structures in P(t);
   while termination condition not satisfied do
   begin
      t = t + 1;
      select_r C(t) from P(t-1);
      recombine structures in C(t) forming C'(t);
      evaluate structures in C'(t);
      select_s P(t) from C'(t) and P(t-1);
   end
end.
```

By a *traditional GA* I mean a GA for which the following is assumed: (1) The initialization of the population P(0) (of fixed size M) is random. (2) The selection for reproduction ($select_r$) is biased toward selecting the better performing structures. (3) The selection for survival ($select_s$) is unbiased, typically replacing the entire parent population P(t-1) with the child population C'(t) generated from P(t-1). (4) The recombination operator is either one- or two-point crossover. (5) A low rate of mutation is used in the recombination stage to maintain population diversity.

CHC differs from a traditional GA on all but the first of these points. First, it is driven by survival-selection (*elitist selection*) rather than reproduction-selection. In other words, the bias in favor of the better performing structures occurs in survival-selection rather than reproduction-selection. Second, a new bias is introduced against mating individuals who are similar (*incest prevention*). Third, the recombination operator used by CHC is a variant of uniform crossover (*HUX*), a highly disruptive form of crossover. Finally, mutation is not performed at the recombination stage. Rather, diversity is maintained (or more precisely, reintroduced) by partial population randomization whenever convergence is detected (*restarts*).

In the remainder of this paper I describe CHC in detail, contrast it with the traditional GA, give a theoretical justification for the differences, and provide some empirical comparisons. Sections 2 and 3 describe and analyze the essential features of CHC — a conservative selection algorithm working in conjunction with a radical (i.e., highly disruptive) recombination operator. Sections 4 and 5 describe mechanisms for

avoiding premature convergence and maintaining diversity. Sections 6 and 7 present strong empirical evidence that CHC outperforms the traditional GA. Section 8 shows how the philosophy of CHC can be extended to permutation problems. Section 9 offers some concluding remarks. An appendix at the end of this paper provides pseudocode for CHC.

2 Elitist Selection

CHC replaces "reproduction with emphasis" with "survival of the fittest". More precisely, during selection for reproduction, instead of biasing the selection of candidates C(t) for reproduction in favor of the better performing members of the parent population P(t-1), each member of P(t-1) is copied to C(t), and randomly paired for reproduction. (In other words, C(t) is identical to P(t-1) except that the order of the structures has been shuffled.) During survival-selection, on the other hand, instead of replacing the old parent population P(t-1) with the child population C'(t) to form P(t), the newly created children must compete with the members of the the parent population P(t-1) for survival — i.e., competition is *cross-generational*. More specifically, the members of P(t-1) and C'(t) are merged and ranked according to fitness, and P(t) is created by selecting the best M (where M is the population size) members of the merged population. (In cases where a member of P(t-1) and a member of C'(t) have the same fitness, the member of P(t-1) is ranked higher.) I shall call this procedure of retaining the best ranked members of the merged parent and child populations *population-elitist selection* since it guarantees that the best M individuals seen so far shall always survive.

Several other GA's use fitness-biased survival-selection — Whitley's GENITOR (1989), Syswerda's Steady State GA (SSSGA) (1989), and Ackley's Iterated Genetic Search (IGS) (1987). CHC differs from all three of these algorithms in that the competition for survival is cross-generational — a child only replaces a member of the parent population if it is better. Furthermore, unlike these three algorithms, CHC operates in generational cycles with many matings rather than one mating per cycle. CHC's sole reliance upon survival-selection for its bias in favor of the better performing individuals also distinguishes it from GENITOR and SSSGA, but not IGS. Finally, CHC's deterministic, rank-based method of doing selection distinguishes it from SSSGA and IGS, but not GENITOR.

In the remainder of this section I shall address the question whether a GA using only population-elitist selection (i.e., relying solely upon cross-generational, deterministic survival-selection for its bias in favor of the better performing individuals) will produce exponential sampling of the better performing schemata (building blocks defined by combinations of bit-values), and thus exhibit implicit parallelism. The answer to this question depends upon what is meant by implicit parallelism. It will be argued that although CHC does not exhibit implicit parallelism as it was originally defined by Holland, it does exhibit a weaker version of implicit parallelism.

First, it is easy to show that elitist selection does not exhibit strong implicit parallelism, where implicit parallelism is understood to mean: *If the observed performance of schema H_i is consistently higher than the observed performance of H_j, the market share of H_i (fraction represented in the population) grows at an exponentially greater rate than the market share of H_j* (Grefenstette and Baker, 1989). The reason that elitist selection fails to demonstrate strong implicit parallelism is that

at most one additional copy of an individual is made in each generation. This means that for any schema H_j that has a high average observed performance but also a high performance variance, CHC cannot compensate in one generation for the performance of the poor individuals by making extra copies of the good individuals. If the poor performing individuals containing H_j are replaced, but only one additional copy is made of the high performing individuals, the net result may be fewer instances of schema H_j even though its average performance was relatively high compared to the population average.

Elitist selection, however, does permit CHC to exhibit Grefenstette's and Baker's (1989) weak version of implicit parallelism: *If schema H_j completely dominates schema H_i, then the market share of H_j grows at least as fast as that of H_i in any generation.* (Schema H_j is said to *completely dominate* schema H_i if every possible representative of H_j is at least as good as every possible representative of H_i.)

In order to see how elitist selection satisfies the conditions for weak implicit parallelism, consider what happens to a single individual. Ignoring for the moment the disruptive effects of recombination, at most two copies of any member of the parent population can survive into the next generation. Furthermore, an individual can never have more copies survive than another individual if it is worse than that other individual. (The converse does not follow: if it is better, it does not necessarily produce more copies.) Thus, the market share of those schemata that completely dominate other schemata will grow at least as fast as the market share of the latter (again ignoring the effects of disruption due to recombination).

To illustrate how this can lead to exponential growth of the number of instances of the better schemata let us first consider a degenerate case in which the recombination operator is the identity operator and simply returns the two parents. In this case C'(t) will not be changed by recombination and will be the same as P(t-1). After survival-selection, however, the best M/2 individuals in P(t-1) will have duplicate copies in P(t), replacing the worst M/2 individuals in P(t-1). Thus, in $\log_2 M$ generations the best individual will have taken over the population. Even though no individual can produce more than a single offspring per generation, good individuals, since they are ranked higher, survive at least as long as, and usually longer than, lower ranked individuals, and, therefore, tend to produce more offspring. Since these offspring, in turn, produce surviving offspring, exponential growth will result. Those individuals that are in the top half of the population will be growing at the same rate (doubling in number each generation). In the long run, however, some of these individuals will be squeezed out of the top half of the population, and the better individuals will increase their market share at the expense of the lower ranked individuals.

Now let us reintroduce recombination into the analysis. (I shall assume that the recombination operator produces two new strings by recombining material from two parents.) Suppose schema H_j completely dominates schema H_i, and instances of both H_j and H_i appear in the parent population. Remember that this means that each instance of H_j performs at least as well as any instance of H_i. This is enough to guarantee that under elitist selection the individuals containing H_j are safe from being replaced by those offspring containing H_i. Further, if the recombination operator sometimes preserves instances of H_j, they will eventually increase their market share at the expense of instances of H_i (unless every sample of H_j has the same performance value as every sample of H_i). In other words, in so far as individuals

containing H_j tend to be better than individuals containing H_i, the market share of H_j will grow exponentially for a longer period of time.

Note that in the above account nothing was assumed about using a recombination operator with a low probability of disrupting schemata. The interesting thing about elitist selection is that it is not necessary to make such an assumption in order to insure exponential growth of the market share of the better schemata. In traditional selection (proportional or linear rank selection) good performers get more copies than bad performers. Elitist selection achieves the same thing by preserving good performers for more generations and thus over time generating more copies. The effect of a disruptive crossover operator, however, is quite different for elitist selection than for traditional selection. Consider, for example, the extreme case where the crossover operator is so disruptive that often all the children are worse than the worst member of the parent population. This would be disastrous for a traditional GA. On the other hand, if elitist selection is used, the parent population will survive intact. Elitist selection is much more robust than traditional selection because it compensates for those occasions in which it generates many inferior children by replacing fewer of the parents.

The essential requirement for any recombination operator used in conjunction with elitist selection is that it not be strongly biased against the better schemata. It does not matter if the operator is highly disruptive provided the offspring of the relatively poor individuals do not have a better chance of surviving a mating than the offspring of the better individuals. Given this proviso it follows that if schema H_j completely dominates schema H_i, then the market share of H_j will be expected to grow at least as fast as H_i in any generation. Furthermore, to the extent that there are instances of H_j that are strictly better than instances of H_i the exponential growth of the market share of H_j can be expected to eventually overtake that of H_i.

Since elitist selection's claim to even weak implicit parallelism depends upon the proviso that the recombination operator is not biased against the better schemata, we must turn our attention to whether there are any highly disruptive recombination operators for which this proviso holds. But before taking up this question, we need to address an even more fundamental question: Why should anyone want to use a highly disruptive recombination operator? I shall address both questions in the next section.

3 Uniform Crossover

It is important to keep in mind that what we are looking for in a GA is an operator that provides productive recombinations and not simply preservation of schemata. There is a tradeoff between effective recombination and preservation. A recombination operator that always crossed over a single differing bit, for example, would create new individuals while being minimally disruptive, but it would not be a very useful operator. More generally, this tradeoff can be seen by examining the formula for the minimum number of schemata preserved via crossover (i.e., when the parents are complementary on all loci):

$$2^x + 2^{(L-x)}$$

where L is the string length and x is the number of bits crossed over. Note that the number of schemata guaranteed to be preserved is greatest when the operator does no recombining, i.e., when $x = 0$ or L. On the other hand, the number of schemata

guaranteed to be preserved is the lowest when $x = L/2$. Thus, if we are mainly interested in preservation, we would prefer that x be low. But what if our concern is with recombination?

The intuitive idea behind recombination is that by combining features from two good parents we may produce even better children. What we want to do is copy high valued schemata from both parents, simultaneously instantiating them in the same child. Clearly the more bits that we copy from the first parent the more schemata we copy and thus the more likely we are to copy, without disruption, high valued schemata. On the other hand, the more bits we copy over from the first parent the fewer we can copy from the second parent, thus increasing the likelihood of disrupting high valued schemata from the second parent. Consequently, a crossover operator that crosses over half the bits (or even better, half the differing bits) will be most likely to combine valuable schemata simply for the reason that the maximum number of schemata is combined from each parent.

Combining many schemata from both parents, however, is just half the story. Every low order schema is defined at a proper subset of the defining positions of at least one higher order schema.[1] Although this is a truism, it has important consequences for search. If the best performing individual in the population is not the global optimum, then certain instances of lower order schemata that are critical to its good performance are combined with other lower order schemata that are hindering performance. The job of recombination is to test these associations by vigorously trying a variety of recombinations. As an illustration of a crossover operator that fails in this regard, consider the one-point crossover operator, H1X, which always cuts the strings at the mid-point into two equal segments. One would not expect this to be a very effective crossover operator for a search algorithm. There is one set of schemata that are always disrupted and another set that are always preserved. Although in any crossover operation H1X will recombine a maximum number of schemata, on subsequent crossovers it will always operate on the same schemata. The case is similar, but less extreme, with a two-point crossover operator, H2X, that crosses over half the material — i.e., one point is chosen at random anywhere on the string, and the second point is chosen so as to be $L/2$ loci from the first point. H2X has the same low probability ($2/L$) of disrupting two adjacent bits as traditional two-point crossover (2X). (It is assumed in the following discussion that the parents differ on the defining loci.) As the defining length is increased, however, the probability of disrupting order-2 schemata increases more rapidly for H2X than for 2X. If the defining length is $L/2$, the probability of disruption for 2X is 0.5 whereas for H2X it is 1.0 (assuming an even number of bits in the string). In fact, any schema (a) whose defining length is greater than or equal to $L/2$ and (b) which has no "gap" of adjacent undefined loci greater than or equal to $L/2$ will always be disrupted by H2X, whereas there will always be some probability that 2X will not disrupt such a schema, provided there are some undefined loci. (It should be kept in mind, however, that the only reason 2X has such a low probability of disruption is that it often does not do much recombination. For example, the probability that it will exchange exactly one bit is 2/L.) More generally, certain schemata tend to get passed through repeated applications of H2X

[1] The order of a schema is the number of defining loci. Its defining length is the difference between the first and last defined loci.

without being disturbed whereas other schemata rarely or never survive intact. This is the result of the strong positional bias of H2X that privileges certain schemata, those that have a short defining length, at the expense of other schemata. In order for a valuable schema to be recombined with schemata from another individual, the valuable schema has to be preserved. Likewise, in order for the performance of a valuable schema to be distinguished, it must be pried apart from other schemata. Privileging some schemata at the expense of others hinders this process. (See Schaffer, Eshelman and Offutt (1991) for a more detailed discussion of this point.)

In summary, neither preserving schemata by not doing much recombination (e.g., 2X) nor doing a lot of recombination but always of the same schemata (e.g., H2X) is a very satisfactory alternative. In the light of these considerations, let us now examine the actual recombination operator used by CHC — a variant of uniform crossover. Uniform crossover (UX) exchanges bits rather than segments. For each position in the string the bits from the two parents are exchanged with fixed probability p (typically 0.5). CHC uses a modified version of UX, HUX, that crosses over exactly half the non-matching alleles (what Booker (1987) calls the reduced surrogate), where the bits to be exchanged are chosen at random without replacement. HUX guarantees that the children are always the maximum Hamming distance from their two parents. The flip side of HUX's disruptiveness is that it maximizes the chance of two good schemata, one from each parent, being combined in a child since half the material from each parent is chosen. Furthermore, all schemata of the same order have an equal chance of being disrupted or preserved.

We are now ready to address the question raised at the end of the previous section. Granted that elitist selection will allocate trials exponentially to the better schemata, provided the disruptiveness is not strongly biased against the better schemata, what reason is there to believe that HUX will not be strongly biased against the better schemata? In so far as better individuals are better because the performance reflects the discovery of some good high order schema, it might seem that it would be much harder to propagate this high order schema than other lower order schemata that are not so good. Furthermore, if this critical schema does not completely dominate its competitors, there is the possibility, in spite of the preservative powers of elitist selection, of losing this critical schema from the parent population — e.g., if other parents can introduce better individuals in the parent population without this schema, even though they might be better yet if they contained this critical schema.

Admittedly, under some circumstances HUX will be biased against the better schemata. It should be kept in mind, however, that the same qualification applies to the traditional GA. In fact, any recombination operator will be biased in favor of certain types of building blocks and against others. Just as the traditional GA works with building blocks with short defining lengths, CHC works with low order building blocks. HUX is especially good at prying apart large (high order) building blocks into small (low order) building blocks and recombining them. Under some circumstances this feature may be a weakness, just as two-point crossover's bias in favor of short defining length building blocks will, under certain circumstances, be a weakness.

Granted that no recombination operator can under all circumstances satisfy the proviso that it not be biased against the better schemata, it might still be objected that the circumstances under which HUX and, more generally, uniform crossover fail are much more serious. After all, the probability of preserving a building block whose defining

length is 10 using two-point crossover in a string of 100 bits is of a different order of magnitude than preserving a building block whose order is 10 using uniform crossover. It might seem that uniform crossover cannot be much more successful than mutation on problems that require the manipulation of schemata of high order. The probability of preserving any schema using uniform crossover (assuming that two children are produced and that the mate contains the compliment of the specified schema) is only twice as great as randomly generating this schema.

It should be kept in mind, however, that although HUX is highly disruptive, it is, unlike mutation, a true recombination operator. It cannot introduce new alleles, but simply recombines schemata contained in the parents. Consequently, loci that are converged cannot be disturbed by uniform crossover. In other words, the more converged the two parents are, the less disruptive uniform crossover is. Furthermore, convergence is not some *deus ex machina* brought in at the last minute to rescue uniform crossover. There is no way a GA can allocate more trials to the schemata of better performing individuals without converging on the loci defining those schemata. Convergence can be a sign of progress as well as a sign of stagnation.

HUX and, more generally, uniform crossover can only be made to look as bad as random search by assuming a worst-case analysis. It is true that if there is one instance of an order-10 schema in the population, and all other individuals in the population are converged on its complimentary bits, then the probability of preserving it is $1/2^9$ (if two children are produced). Why, however, assume this worst case? If, for example, there is a only one instance of a schema H_k of order k, in a population of size M, then the probability that a child will be created by uniform crossover that preserves this schema is:

$$\sum_{i=0}^{i=k-1} \frac{Instances(H_k,i)}{(M-1)\,2^{(k-i)-1}}$$

where $Instances(H_k,i)$ is the number of individuals in the population that share the same value for exactly i bits of the k bits defining schema H_k. $Instances(H_k,i)/(M-1)$, then, is the probability of choosing a mate that matches on i of the k defining bits of H_k. The worst case occurs whenever for *all* $i > 0$, $Instances(H_k,i) = 0$. On the other hand, if $Instances(H_k,i)$ is relatively large for i near k, then crossover is likely to preserve schema H_k.

In order to better understand how likely CHC is to propagate any high order schema H_k we need to consider three cases. In the first case, there are lower order building blocks that contribute to the performance of the individuals independently of finding the high order building block. In this case, there is likely to be some convergence in the population making it easier to find and propagate the critical high order schema H_k. In the second case, all the low order schemata that might serve as building blocks for H_k have similar values to their competitors, and so are no help in finding H_k. On the other hand, there is no pressure for the bits in the loci defining H_k to converge on any particular set of values. Values for these bits will tend to wander randomly. This means that the probability of propagating H_k is not the worst case of $1/2^{(k-1)}$ but closer to $1/2^{(k/2-1)}$. The reason is that given a randomly initialized population and no pressure for convergence, half the bits on average will match those in H_k. Finally, the worst case occurs when the problem is a deceptive problem (Goldberg, 1987), and there is pressure for the lower order building blocks to converge

on values that are compliments of the bit-values defining H_k. In this case, if the order of H_k is high, the schema will be unlikely to propagate ($1/2^{(k-1)}$). Furthermore, if there are several such high order deceptive schemata instantiated in different individuals, then it will be highly unlikely that they will ever be successfully combined intact in a single offspring.

Even in this latter case the traditional GA does not necessarily have an advantage over CHC. The traditional GA can do relatively well on such deceptive problems only if two additional conditions hold. First, the GA must use a population size that is large enough to make it likely that these high order deceptive schemata will occur in the initial, randomly generated population. Secondly, the high order deceptive schemata should have short defining lengths, and, more importantly, their defining loci should not be interspersed. The first condition can easily be met provided one is willing to pay the cost — large populations require more evaluations to converge to a solution. The trouble is that typically one does not know whether or not the problem is a high order deceptive problem. If it turns out not to be such a problem, then a much smaller population is likely to be more efficient for search. The second condition, however, is the critical one. What reason is there for assuming that the problem will be so well behaved that the deceptive schemata will tend not to overlap? It is easy to contrive problems where this assumption is met, but there is no reason to believe that real problems are going to be so benign. In fact, GAs are usually advertised as being most useful for large, complex, poorly understood problems, where there is little or no a priori knowledge about how the bits interact. Under such conditions the traditional GA will have no advantage over CHC with regard to deceptive schemata.

4 Avoiding Incest

The exponential growth of instances of good schemata is of little value if it leads to premature convergence. One of the effects of crossing over half the differing bits between the parents is that the danger of premature convergence is lessened. Even if at each generation the most recent descendant mates with one of its original ancestors (the same one each time), it will take $\log_2 h$ generations to converge (within one bit) to the original ancestor where h is the Hamming distance between the original parents. In the case of two-point crossover, on the other hand, each of the two children will differ from its nearest (measured by Hamming distance) parent by an amount ranging from one bit to no more than half the length of the string L. Thus, the longest time it can take to converge within one bit of its ancestor is $\log_2 h$ generations and the shortest is one generation. [2] Of course, a child is unlikely to be repeatedly mated with one of its remote ancestors, but since better individuals have more descendents, it is fairly likely that a individual will be mated with one of its near relatives. In so far as this leads to crossing over of individuals that share a lot of alleles, exploration via recombination quickly degenerates. Although always crossing over half the differences (using HUX) slows this process sometimes individuals are paired that have few differences. If one or both children survive this mating, it will be even more likely that such an event will occur the next generation.

[2] If, instead of always choosing the child that is nearest (in terms of Hamming distance) to the parent that is its original ancestor, a child is chosen at random, then convergence may take longer than $\log_2 h$ generations, but the expected value will still be less than $\log_2 h$.

4

CHC has an additional mechanism to slow the pace of convergence — a mechanism for helping avoid *incest*. During the reproduction step, each member of the parent population is randomly chosen without replacement and paired for mating. Before mating, however, the Hamming distance between potential parents is calculated, and if half that distance (the Hamming distance of the expected children from their parents) does not exceed a difference threshold, they are not mated and are deleted from the child population. (The difference threshold is set at the beginning of the run to L/4 — half the expected Hamming distance between two randomly generated strings.) Thus, typically only a fraction of the population is mated to produce new offspring in any generation. Whenever no children are accepted into the parent population (either because no potential mates were mated or because none of the children were better than the worst member of the parent population), the difference threshold is decremented. The effect of this mechanism is that only the more diverse potential parents are mated, but the diversity required by the difference threshold automatically decreases as the population naturally converges. The number of survivors for each generation stays remarkably constant throughout the search because when CHC is having difficulty making progress, the difference threshold drops faster than the average Hamming distance, so that more individuals are evaluated. Conversely, when CHC is finding it easy to generate children that survive, the difference threshold drops at a slower rate, and the number of matings falls. (See Eshelman and Schaffer (1991) for a more detailed discussion of the benefits of incest prevention.)

5 Restarts

Nothing has been said so far concerning mutation. This is because in the reproduction-recombination cycle CHC does not use any mutation. The use of HUX and incest prevention in conjunction with a population size large enough to preserve a number of diverse structures (e.g., 50) enables CHC to delay premature convergence, and thus do quite well without any mutation. But these various mechanisms cannot guarantee that no allele will prematurely converge. Some sort of mutation is needed.

Mutation, however, is less effective in CHC than in the traditional GA. Since CHC is already very good at maintaining diversity, mutation contributes very little early on in the search. On the other hand, late in the search, when the population is nearly converged, mutation, combined with elitist selection, is not very effective in reintroducing diversity. At this stage of the search mutation will rarely produce an individual who is better than the worse individual in the population, and consequently, very few new individuals will be accepted into the population. In contrast to CHC, a traditional GA, by replacing the parent population each generation, insures that new variations will constantly be introduced.[3]

CHC's way out of this impasse is to introduce mutation only when the population has converged or search has stagnated (i.e., the difference threshold has dropped to zero and there have been several generations without any new offspring accepted into the parent population). More specifically, whenever the reproduction-recombination cycle

[3] This might explain why the Evolution Strategy work in Germany (Hoffmeister and Bäck, 1990), which initially used a form of cross-generational competition, later abandoned it, opting for a selection strategy more closely resembling that used by the traditional GA.

achieves its termination condition, the population is reinitialized (diverged) and the cycle is repeated. The reinitialization, however, is only partial. The population is reinitialized by using the best individual found so far as a template for creating a new population. Each new individual is created by flipping a fixed proportion (e.g., 35%) of the template's bits chosen at random without replacement. One instance of the best individual is added unchanged to the new population. This insures that the next search cannot converge to a worse solution than the previous search. This outer loop, consisting of reinitialization followed by genetic search, is iterated until its termination condition is met (either reaching a fixed number of reinitializations, or repeated failure to find any better structures than the retained structure).[4]

It should be pointed out that for CHC there is no danger that the best individual found so far, even though included in the reinitialized population, will rapidly take over the population. CHC's incest preventing mechanism (the dropping difference threshold), in combination with elitist selection and disruptive recombination prevents this. This does not mean, however, that saving the best individual has no effect on search. Crossover will still recombine the schemata represented in this surviving best individual with those of the other individuals in the population, but mating is prevented whenever two individuals are paired that are relatively more similar than the population as a whole as reflected by the difference threshold.

An advantage of partial reinitializations over chronic mutation is that CHC can do quite well on a large range of problems using the same parameter settings. Restarts provide many of the benefits of a large population without the cost of slower search. On easy problems the optimal solution is found in the first initialization cycle, whereas on hard problems the optimal solution is found only after repeated restarts.

6 Empirical Results

CHC's performance has been compared with that of a traditional GA on a number of functions. The most extensive comparisons are based on a test suite of 10 functions, F1-F10, previously studied by Schaffer, et al. (1989). They tested a traditional GA[5] on functions F1-F10 with 840 different parameter settings including 6 population sizes, 10 crossover rates, 7 mutation rates, and two crossover operators (one- and two-point). Each search was run for 10000 evaluations and repeated 10 times. The test suite consisted of the five De Jong functions (1975) as well as five other multi-modal functions, including two sine-wave-based functions, a FIR digital filter optimization task, a 30-city traveling salesman problem (with a sort order representation), and a 64-node graph partition task.

In order to identify how well a GA can perform on each of these 10 functions, I chose for each function the 5 parameter sets that worked best (i.e., found the best solutions

[4] Ackley's (1987) IGS algorithm also includes an outer, reinitialization loop, but IGS still retains mutation in the inner loop, and fully randomizes the population when it is reinitialized.

[5] The GA in Schaffer's study used Baker's (1987) stochastic universal sampling, dynamic linear fitness scaling (scaled to the worst individual) (Grefenstette and Baker, 1989), the individual elitist strategy (Grefenstette, 1986) (not to be confused with the population-elitist selection discussed in this paper), a generation gap of 1.0 (i.e., the entire parent population is replaced by the children), and Gray coding for the numeric parameters (Caruana and Schaffer, 1988).

'6 in the least number of evaluations) from among the 840 parameter sets tested and conducted runoffs. In this second series of experiments, the search was continued until the global optimum was found or until a maximum of 50000 evaluations was reached. (Because F4 is a noisy function, it was treated somewhat differently. The search was halted whenever the performance value before adding noise was less than two standard deviations of the noise function from the optimum (i.e., F4(x) < 2).) Each search was repeated 50 times. CHC was also tested on F1-F10 under the same conditions except that a single set of parameters was used for all the functions (a population size of 50 and a divergence rate of 35%). Note that since a single parameter set was used for CHC whereas the parameter settings for the GA were selected independently for each function, there is a methodological bias in favor of the traditional GA.

Table 1: Mean Number of Evaluations to Find the Global Optimum

Func	Traditional GA			CHC		
	#opt[a]	mean[b]	sem[c]	#opt	mean	sem
F1	50	805	48	50	1089	25
F2	50	9201	703	50	9065	591
F3	50	1270	100	50	1169	27
F4	50	2228	135	50	1948	97
F5	50	1719	96	50	1396	38
F6	37	9272	1291	50	6496	725
F7	50	8688	738	50	3634	291
F8	11	30986	3712	50	7279	725
F9	1	24402	NA	29	24866	2404
F10	2	5520	1087	33	10217	1107

[a] The number of searches out of 50 that the algorithm succeeded in finding the optimum value.

[b] Mean number of evaluations to find the optimum in those searches where it did find the optimum.

[c] Standard error of the mean.

Table 1 compares the performance of CHC with the best GA for each of the functions. In spite of the methodological bias in favor of the GA, CHC does better on nine of the ten functions. (An algorithm performs better on a function if it finds the global optimum more often, or it finds the global optimum the same number of times as its competitor but in fewer evaluations.) For five of the six functions in which both algorithms found the optimum in all 50 searches, CHC, on average, found the optimum in fewer evaluations, and for the remaining four functions, CHC found the optimum more often than the GA. Furthermore, the only function on which the traditional GA does significantly better than CHC is F1, a smooth, unimodal function that is the easiest for most traditional optimization techniques. On the other hand, CHC does significantly better on all the multi-modal functions (F5-F10).

Since the traditional GA was rarely able to find the optimum on the last three functions, perhaps it is only fair to consider how well CHC does when the number of evaluations is limited to 10000 as was the case in the study done by Schaffer, et al.

The GA was able to find the optimum once for F8 and twice for F10 within 10000 evaluations whereas CHC was able to find the optimum 42 times for F8 and 20 times for F10. In the case of F9, a 30-city traveling salesman problem, neither algorithm was very successful at finding the optimum (i.e., best known tour of 421) within 10000 evaluations — CHC found it three times and the GA not at all. If we look, however, at the mean of the shortest tours found over 50 runs, it is obvious that CHC is doing significantly better. The mean performance for CHC was 461 after 10000 evaluations and 429 after 50000 evaluations, whereas for the GA it was 538 after 10000 evaluations and 482 after 50000 evaluations. (The standard error for all four means was less than 5.)

Finally, something should be said about the role played by restarts. The divergence rate used in these experiments (35%) is a rate that works well over a wide range of functions; however, it is not necessarily the best rate for functions F1-F10. In fact, performance can be significantly improved on F10 (so that it will find the optimum in all 50 runs) without significantly affecting performance on the other functions, by using a divergence rate of 35%, but completely randomizing the population whenever there is a total of three restarts that do not result in improvements. In should also be pointed out that CHC rarely had to resort to restarts when searching the easier functions, F1, F3-F5. The functions that required the most restarts were the most difficult functions, F9 and F10 — an average of 6.2 restarts per run for F9 and 9.6 per run for F10. This is not to imply that it is simply the use of restarts that accounts for CHC's better performance on these more difficult functions. For both F9 and F10, CHC's performance was significantly better at the point of the first restart (7798 and 4302 mean evaluations, respectively) than the GA had achieved after 50000 evaluations.

7 Deceptive Problems

Although the results of the previous section indicate that CHC is a robust search algorithm, it should be noted that there are two identifiable classes of functions that are consistently more difficult for CHC to optimize than the traditional GA: (1) functions that are easy for a robust hillclimber (e.g., a stochastic hillclimber) and (2) tightly ordered (benign) deceptive functions. These are functions for which CHC's usual assets are liabilities. CHC's mechanisms for slowing convergence, especially incest prevention, enable it to perform a coarse-to-fine search of the space. The traditional GA's tendency to quickly converge on a promising area of the search space can be to its advantage, however, if no additional insights are to be gained from knowledge of the problem's macrostructure.

In the case of tightly ordered deceptive functions — functions in which the bits of the deceptive subproblems are adjacent — the traditional GA's positional bias — favoring schemata of short defining length — gives it an advantage over CHC, provided the GA uses a very low mutation rate (e.g., zero) and a large population size (e.g., 500). As was argued in section 3, however, there is no reason to believe that real problems are going to be so benign. After all, GAs are supposed to be most useful for poorly understood search spaces where there is little or no a priori knowledge about which bits interact, and so little likelihood that the problem will be represented so that bits constituting deceptive subproblems will be tightly ordered.

In order to get a better handle on CHC's ability to deal with deceptiveness, CHC was run on Goldberg's order-3 deceptive problem (Goldberg, Korb and Deb, 1989) and Liepins' order-n deceptive problems for n of 3, 4 and 5 (Liepins and Vose, 1991). All four problems consisted of 10 deceptive subproblems. Liepins' fully deceptive function, f, for a string of length n is defined as follows:

$$f(x) = \begin{cases} 1 - \frac{o(x)}{2n}, & \text{if } o(x) = 0; \\ 1, & \text{if } o(x) = n; \\ 1 - \frac{1+o(x)}{n}, & \text{otherwise}, \end{cases}$$

where $o(x)$ is the number of 1's in the string. Goldberg's fully deceptive order-3 function is defined in Table 2.

Table 2: Goldberg's order-3 deceptive problem

f(000) = 28		
f(001) = 26	f(010) = 22	f(100) = 14
f(011) = 0	f(101) = 0	f(110) = 0
f(111) = 30		

Table 3 shows CHC's performance for the four functions based on 50 replications of each search. The first line shows the results using the "standard" parameter settings — population size of 50 and divergence rate of 0.35. Results for two other divergence rates (but the same population size) are also shown.

Table 3: CHC's Performance on Deceptive Functions: Percent of Subproblems Solved in 50000 Evaluations

div rate	Goldberg's order-3	Liepins' order-3	Liepins' order-4	Liepins' order-5
0.35	85.8	100	79.6	19.2
0.25	99.2	100	92.2	25.6
0.15	100	100	99.6	17.4

Goldberg (1989) reports that for his order-3 problem a traditional GA was able to find the optimum in about 40000 trials if the subproblems are tightly ordered, but converges to the local optimum (all 0's) if the subproblems are loosely ordered (the first 3-bit subproblem is located at positions 1, 11, and 21, the second subproblem is located at positions 2, 12 and 22, etc.). His messy GA was able to find the optimum for the loosely ordered problem in about 40000 evaluations. Since CHC's crossover operator is not position biased, its performance is not affected by the ordering. CHC was able to find the optimum in 20960 function evaluations (mean of 50 runs, with a minimum of 9933, a maximum of 36297 evaluations, and a standard error of 980) when the divergence rate was 0.15. This makes CHC a worthy competitor to Goldberg's messy GA (at least for low-order, loosely ordered deceptive problems). For the Liepins' order-3 problem the mean number of evaluations to find the optimum were 7196, 6207, and 6170, respectively, for the three divergence rates of 0.35, 0.25, and 0.15. It should also be pointed out that if we don't restrict CHC to using HUX but allow CHC to use two-point, reduced surrogate crossover and a divergence rate of 0.5, CHC is able to find the global optimum for Goldberg's order-3 deceptive problem in 3162 evaluations and for Liepins order-3, -4 and -5 deceptive problem in 2143, 4929, 11230 evaluations, respectively (means for 50 runs).

8 Extending CHC to Permutation Problems

This section illustrates how it is possible to extend the philosophy of CHC to a GA that manipulates permutations of integers to solve traveling salesman problems. The basic algorithm is the same as before except that it uses different operators for mutation (when it does a restart) and crossover, and each individual is improved by a hillclimber before it is evaluated, thus limiting the search to the space of local minima. The operators are similar to those used by Muhlenbein's ASPARAGOS/PGA (Gorges-Schleuter, 1989, Muhlenbein, 1990). The crossover operator creates a single child by preserving the edges that the parents have in common and then randomly assigning the remaining edges in order to generate a legal tour. Whenever the population converges, the population is partially randomized for a restart by using the best individual found so far as a template and creating new individuals by repeatedly swapping edges until a specific fraction of the edges (e.g., 30%) differ from those of the template. Whenever an individual is created for the initial population, by mutation for a restart, or by crossover, a hillclimber is used to improve the tour. The hillclimber is a variant of Lin's two-opt (1973). Subtours are systematically inverted by swapping two edges. Whenever an improvement is found, the tour is updated, and the hillclimber continues until it can make no more improvements. If the individual is created by crossover or mutation (if a restart), the hillclimber is constrained to swapping the new edges — i.e., the edges that were created randomly by crossover or by mutation in the case of a restart. There is a tremendous improvement in efficiency by restricting the edges that can be swapped by the hillclimber. This is especially apparent when used in conjunction with crossover — the time devoted to hillclimbing drops exponentially as the population converges and the parents have more edges in common.

Incest prevention works the same as it does for the binary version of CHC, except that instead of counting the number of bits that the two parents have in common, the number of edges that they share is counted. If the number of common edges is not above the incest threshold, the potential parents are not mated. Only one child is created per mating. No changes were made in CHC's selection procedure.

The algorithm was tested on Padberg's 532-city traveling salesman problem (1987). (Most standard problems of less than 100 cities are not much of a challenge.) Table 4 reports the performance for 10 runs. Each run was allowed 4 restarts with a divergence rate of 30%. The population size was 50.

Table 4: 532-City-TSP Tour Lengths

mean	sem	best	worst	#runs <27800	#runs <27750
27747	11.8	27710	27849	9	9

It should be noted that the best value (27710) found by CHC is within 0.1% of the optimum (27686). CHC's performance on this problem is comparable to that reported for ASPARAGOS — a mean of 27770 for ten replications with the best run having a tour of 27715 (Gorges-Schleuter, 1989). More recently, Muhlenbein has reported that his parallel GA has been able to find a tour of 27702 after 3 hours of search using 64 processors (1990). This result outclasses any previous GA on a large TSP problem. Although CHC's performance is not quite as good as that reported by Muhlenbein, it should be pointed out that CHC was able to find a tour of 27710 in under 3 hours (the average run time was 2.5 hours per replication) on a *single* processor — a Sun SPARCstation 1+.

9 Conclusion

CHC outperforms the traditional GA as a function optimizer over a wide range of functions. Given a fixed number of evaluations or the goal of finding the optimum in a minimum number of trials, a GA that can maintain population diversity while using a small population size (e.g., 50) will have an advantage, since large population sizes mean fewer generations and, thus, less search. CHC is able to maintain population diversity without sacrificing implicit parallelism by combining a highly disruptive crossover operator with a conservative selection procedure that preserves any progress made.

The evolution of CHC has been a story in which, once elitist selection was in place, every change that increased the recombination power of crossover, and consequently its disruptiveness, also improved CHC's performance across a wide variety of functions. The switch from traditional crossover to UX, and then to HUX, and finally the addition of the downward adjusting difference threshold to prevent "incest" have all led to dramatic improvements. Moving mutation from inside to outside the reproduction-recombination loop also significantly improved performance, especially on the more difficult problems, and was instrumental in allowing CHC to perform well on a number of functions using the same set of parameter values.

Acknowledgements

Dave Schaffer, Dan Offutt, and Rich Caruana have contributed many useful ideas throughout the evolution of CHC. I would also like to thank Dave Schaffer, Dan Offutt and Benjamin Zhu for their helpful comments on various drafts of this paper.

References

D. H. Ackley. (1987) *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Kluwer Academic Publishers.

J. E. Baker. (1987) Reducing Bias and Inefficiency in the Selection Algorithm. *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, 14-21. Hillsdale, NJ: Lawrence Erlbaum Associates.*

L. Booker. (1987) Improving Search in Genetic Algorithms. In L. Davis (ed.), Genetic Algorithms and Simulated Annealing, 61-73. San Mateo, CA: Morgan Kaufmann.

R. A. Caruana and J. D. Schaffer. (1988) Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. *Proceedings of the 5th International Conference on Machine Learning*, 153-161. San Mateo, CA: Morgan Kaufmann.

K. A. De Jong. (1975) Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI.

L. J. Eshelman and J. D. Schaffer. (1991) Preventing Premature Convergence in Genetic Algorithms by Preventing Incest. *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann.

D. E. Goldberg. (1987) Simple Genetic Algorithms and the Minimal, Deceptive Problem. In L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*, 74-88. San Mateo, CA: Morgan Kaufmann.

D. E. Goldberg, B. Korb and K. Deb. (1989) Messy Genetic Algorithms: Motivation, Analysis, and First Results. TCGA Report No. 89003. To appear in *Complex Systems*.

M. Gorges-Schleuter. (1989) ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy. *Proceedings of the Third International Conference on Genetic Algorithms*, 422-427. San Mateo, CA: Morgan Kaufmann.

J. J. Grefenstette. (1986) Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man & Cybernetics* SMC-16 (1): 122-128.

J. J. Grefenstette and J.E. Baker. (1989) How Genetic Algorithms Work: A Critical Look at Implicit Parallelism. *Proceedings of the Third International Conference on Genetic Algorithms*, 20-27. San Mateo, CA: Morgan Kaufmann.

F. Hoffmeister and T. Bäck. (1990) Genetic Algorithms and Evolution Strategies: Similarities and Differences. *Proceedings of the First International Workshop on Parallel Problem Solving from Nature*. Dortmund, Germany: University of Dortmund.

J. H. Holland. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, University of Michigan Press.

G. E. Liepins and M. D. Vose. (1991) Representational Issues in Genetic Optimization. *Journal of Experimental and Theoretical AI* (May).

S. Lin and B. W. Kernighan. (1973) An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research* 21: 498-516.

H. Muhlenbein. (1990) Parallel Genetic Algorithms and Combinatorial Optimization. Symposium on Parallel Optimization. Madison, WS.

W. Padberg and G. Rinaldi. (1987) Optimization of a 532-City Symmetric TSP *Operation Research Letters* 6: 1-7.

J. D. Schaffer, R. A. Caruana, L. J. Eshelman and R. Das. (1989) A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. *Proceedings of the Third International Conference on Genetic Algorithms*, 51-60. San Mateo, CA: Morgan Kaufmann.

J. D. Schaffer, L. J. Eshelman and D. Offutt. (1991) Spurious Correlations and Premature Convergence in Genetic Algorithms. In G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms and Classifier Systems*. San Mateo, CA: Morgan Kaufmann.

G. Syswerda. (1989) Uniform Crossover in Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, 2-9. San Mateo, CA: Morgan Kaufmann.

D. Whitley. (1989) The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. *Proceedings of the Third International Conference on Genetic Algorithms*, 116-121. San Mateo, CA: Morgan Kaufmann.

Appendix: Pseudocode for CHC

```
procedure CHC
begin
   t = 0;
   d = L/4;
   initialize P(t);
   evaluate structures in P(t);
   while termination condition not satisfied do
   begin
      t = t + 1;
      select_r C(t) from P(t-1);
      recombine structures in C(t) forming C'(t)
      evaluate structures in C'(t);
      select_s P(t) from C'(t) and P(t-1);
      if P(t) equals P(t-1)
         d--;
      if d < 0
      begin
         diverge P(t);
         d = r × (1.0 - r) × L;
      end
   end
end
```



```
procedure select_r
begin
   copy all members of P(t-1) to C(t) in random order;
end.

procedure select_s
begin
   form P(t) from P(t-1)
      by replacing the worst members of P(t-1)
      with the best members of C'(t)
   until no remaining member of C'(t)
      is any better than any remaining member of P(t-1);
end.

procedure recombine
begin
   for each of the M/2 pairs of structures in C(t)
   begin
      determine the Hamming_distance
      if (Hamming_distance/2) > d
         swap half the differing bits at random;
      else
         delete the pair of structures from C(t);
   end
end.

procedure diverge
begin
   replace P(t) with M copies of the best member of P(t-1);
   for all but one member of P(t)
   begin
      flip r × L bits at random;
      evaluate structure;
   end
end.

variables
   M  population size
   L  string length
   t  generation
   d  difference threshold
   r  divergence rate
```