Genetic Algorithms and Fine-Grained Topologies for Optimization

Xiaotong Wang
NuTech Solutions, Inc.,
28 Green Street,
Newbury, MA, 01951

Lawrence Davis
NuTech Solutions, Inc.,
28 Green Street,
Newbury, MA, 01951

Chunsheng Fu
NuTech Solutions, Inc.,
28 Green Street,
Newbury, MA, 01951

Abstract

In this paper we show how the performance of two meta-heuristic algorithms and two simple search routines varies as these algorithms are applied singly, in pairwise combinations, and in larger, finer-grained combinations. The area of application is f6 and f17, two well-known optimization benchmark problems. Our conclusion is that when these algorithms are combined in complex ways, their performance is much better than when they are used alone or in pairs, and so there is strong evidence that the current approach to optimization followed by many current practitioners with, for instance, an evolutionary algorithm succeeded by a hill-climber, could be improved on if more complex algorithm topologies were used.

1 MOTIVATION

This paper is designed to suggest that the approach currently used by many persons doing real-world optimization and doing optimization of test functions can be improved if those persons consider using combinations of optimization algorithms, rather than individual algorithms, or individual algorithms followed by a round of hill-climbing.

The conclusions of this paper will not be surprising to many readers, as they have been touched on before by other writers [e.g. Powell, Skolnick, and Tong 1991] each of whom has suggested that combinations of many algorithms can be noticeably more effective in optimization than those algorithms are in isolation or in pairs. The novelty in our paper stems from two sources. First, we will show that fine-grained algorithm topologies, in our parlance, do much better than simpler ones in optimizing. Second, we will show that for f6 and f17 (the test problems on which the results in this paper were generated) what arises from the best topologies is a type of algorithm topology different in type from those that many of us are use when we carry out optimization.

2 THE INITIAL PROBLEM

For our experiments, we used the test function f6, first suggested as an evolutionary computation benchmark by David Schaffer [1989]. This problem was used in extensive testing of evolutionary algorithms by Schaffer and his collaborators, and it was the problem that carried the weight of the expositional burden in the extended tutorial in [Davis, 1990]. It is a two-dimensional, damped sine wave that has been shown to be difficult for some types of hill-climbers and for simulated annealers. A cross-section of the f6 function is shown in Figure 1. The function is to be maximized, and so the optimal point is located at the center of the curve. The full function is generated by rotating the curve about its center point, so that it appears as a series of concentric ripples in a pond, with the highest ripple located at the very center of the pond. If one is searching for the optimal point without prior knowledge of the shape of the curve, it can be very difficult to locate. The function is extremely hilly, and since the ripple containing the optimal point is the smallest ripple in the pond, an extremely small part of the whole search space is occupied by the hill containing the optimal point.

The f6 function is formally stated as follows:

$$0.5 - \frac{(\sin\sqrt{x^2+y^2})^2 - 0.5}{(1.0 + 0.001(x^2+y^2))^2}$$

A solution to the f6 problem consists of two real-valued numbers x and y, each in the range between –100 and +100. The evaluation of such a solution is the value that the f6 functions returns when those numbers are plugged in as x and y.

3 ALGORITHMS USED

We used four algorithms in the experiments reported here:

- A random search (RS) algorithm that generates x, y pairs with uniform probability from the variable range of –100 to +100. The random search algorithm preserves the best solution it has found so far, and

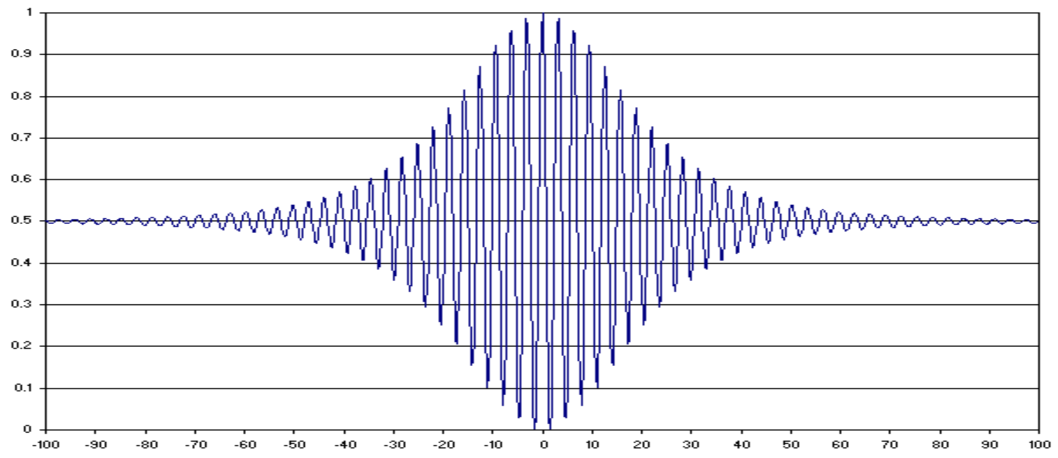

Figure 1. cross-section of f6

halts when it has carried out a predetermined number of evaluations.

- An opportunistic hill-climber (HC) that begins from a random position in the solution space. It maintains a current solution <x, y> and generates new solutions from a uniform distribution around the current x and y values. If the new solution is equal to or better than the current solution, then the new solution becomes the current solution. The hill-climber we use stops when it has carried out a predetermined number of evaluations.
- A simulated annealer (SA) that begins from a random position in the search space. The starting temperature is 2.0, and the cooling factor is 0.80. Our simulated annealer, given a number of evaluations n to work with, carries out n/20 of those evaluations at each temperature during optimization, and proceeds until it has carried out n evaluations.
- A genetic algorithm (GA) that begins with a randomly-generated set of solutions from the solution space. The genetic algorithm, given a number of evaluations n to work with, uses 10% of those as its population size, and proceeds until it has carried out n evaluations. The genetic algorithm is steady-state, allows no duplicates, represents solutions as lists of real values, and deletes the worst member of the population after a new solution is inserted.

4 INDIVIDUAL ALGORITHM PERFORMANCE

Many optimization systems, both commercial and academic, use a single algorithm or heuristic when optimizing. In cases when it is known that one will find the global optimum in the time available—when doing linear programming, for example, on a small problem—there is no need to consider alternatives, as long as one has sufficient computer resources with which to optimize. It is cases in which one is searching for good answers with limited computer resources that concern us here. What we will show below is that using any of our four algorithms in isolation to solve f6 and f17, or even using one followed by another for post-processing, is a strategy that is inferior to using fine-grained combinations of these algorithms.

Algorithm	Mean	Maximum evaluation	Minimum evaluation
HC	0.991236	1	0.986304
GA	0.990170	1	0.972698
SA	0.960624	1	0.933464
RS	0.959234	1	0.903464

Table 1. Single algorithm performance

Algorithm	Mean	algorithm	Mean
GA→HC	0.990848	HC→GA	0.990689
GA→SA	0.987373	HC→SA	0.990459
GA→RS	0.986847	HC→RS	0.990658
SA→HC	0.990585	RS→HC	0.990434
SA→GA	0.988114	RS→GA	0.988383
SA→RS	0.960609	RS→SA	0.962413

Table 2. Mean score of pairwise combination of algorithms

As Table 1 shows, our four algorithms vary widely in performance as they solve f6. Table 1 shows the mean, best, and worst scores for each of our algorithms after 3000 evaluations, as each algorithm solves f6 from a random start. The results in the table describe 500 runs of each algorithm, with a different random seed for each run. Table 1 shows us that, when using the same step size, the best single algorithm of these four for solving f6 is the hill-climber.

Let us now consider what the best pairwise combination of algorithms is. Table 2 shows the average of the best solution found for 500 runs each of each of the different seeding possibilities between our four algorithms. In each single run, one of the algorithms was run for 1500 evaluations and its best solution found was used as the seed for the second algorithm. Table 2 shows us that the best combination consists of a genetic algorithm seeding the hill-climber. The table also shows us that the performance of this combination is inferior to the performance of the hill-climber used alone. We do not show the maximum evaluation in this table since it is generally 1. The mean is the most significant value, while the minimum is less significant, so, we present the mean as the best representation of related algorithm pair performance.

It is possible that a different ratio of evaluations between these two algorithms would yield better solutions. However, this is not the case. Figure 2 shows the mean of 10 runs for each of the weight ratios between the hill-climber and the genetic algorithm, carried out at intervals of .2 and with 3000 evaluations per run. When evaluations are allotted to the genetic algorithm, performance is degraded. As the ratio rises to 3 or higher, the hill-climber is effectively carrying out search on its own, and the results are not significantly different for higher ratios.

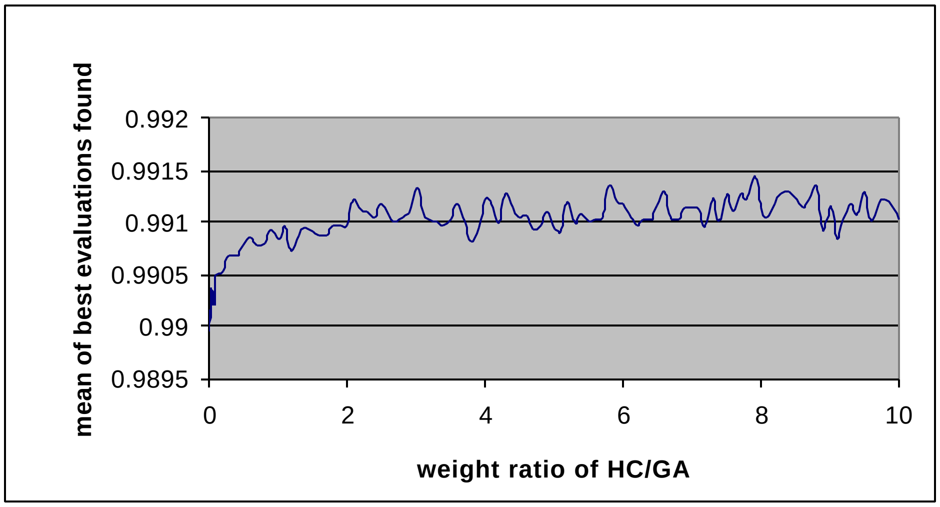

Figure 2. impact of weight ratios for F6

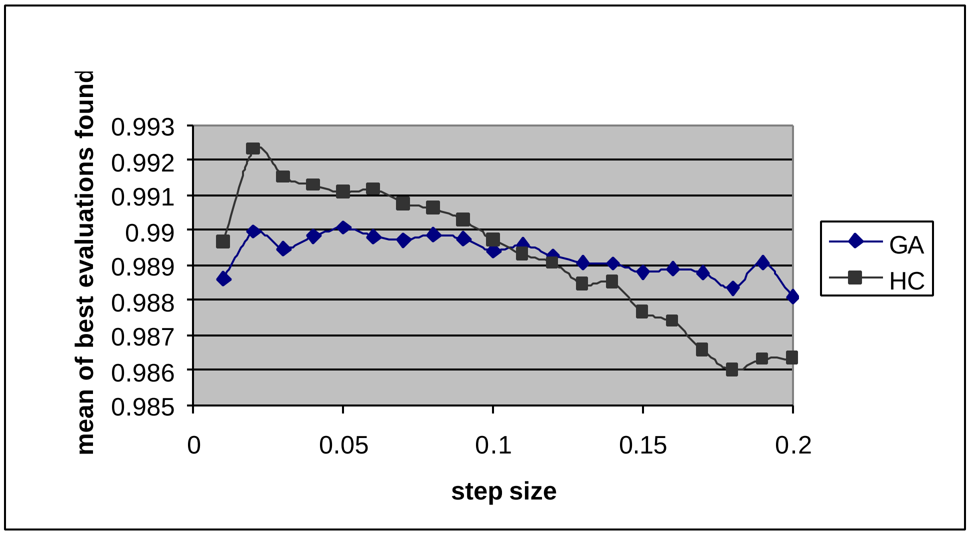

Figure 3. impact of step size for single GA and HC on f6

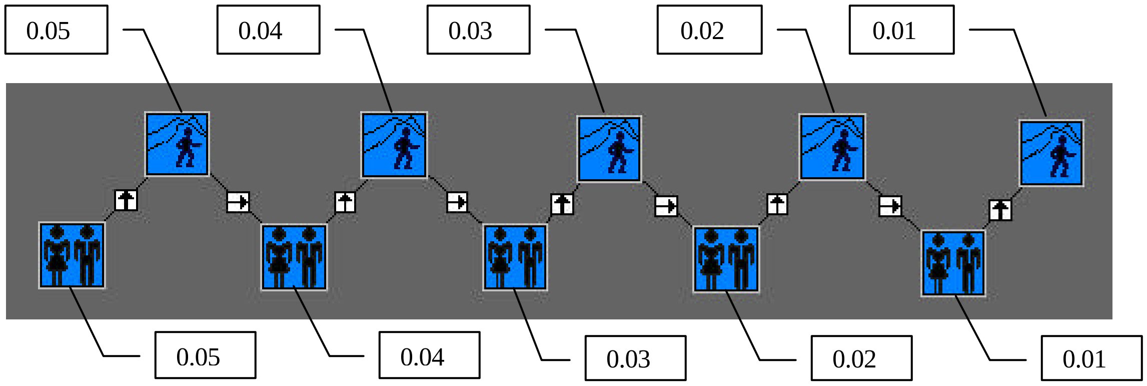

Figure 4. Step size distribution for a 5 GA $\rightarrow$ HC chain

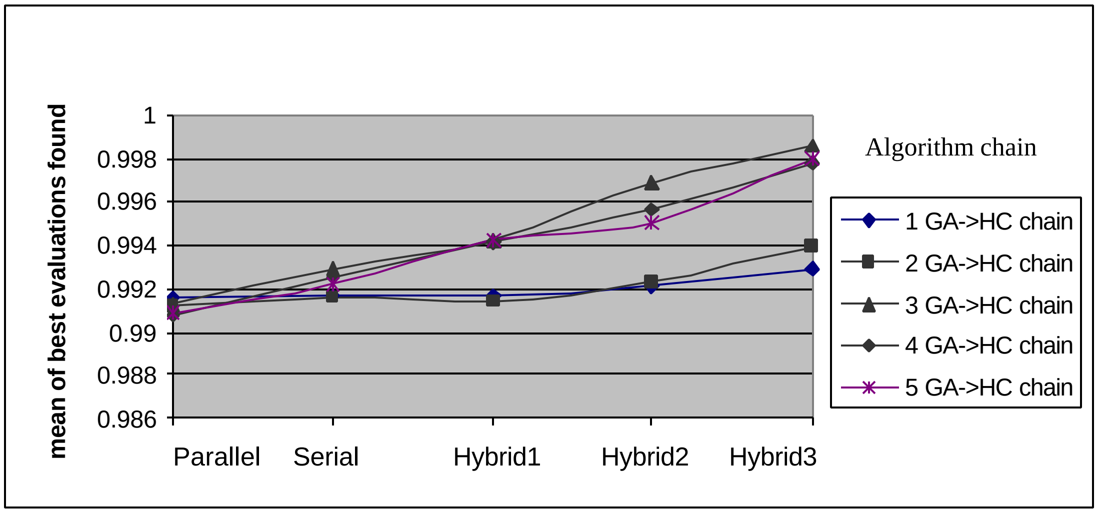

Figure 5. impact of number of algorithms and their configuration

5 THE IMPACT OF STEP SIZE

For the single-algorithm case and the two-algorithm case, the best algorithm is the hill-climber—especially interesting, when we remember that f6 is an algorithm that looks like it should frustrate hill-climbers. Why is it that the hill-climber does so well on a problem with a large number of hills? The answer is that the maximum step size used by the hill-climber in these experiments is .05, which allows the algorithm to jump from peak to peak in the search space. We set the step size at .05 for the three algorithms that use step size, having found empirically that it gave the best collective performance. Perhaps settings tailored for each algorithm will give better individual performance.

Figure 3 shows the effects of varying step size for the genetic algorithm and for hill-climber, considering their performance alone. The optimal step size for the hill-climber, considered in isolation, is actually about .02, whereas for the genetic algorithm the optimal value has less impact on performance, but lies at about .07. We see from Figure 2 that when the step size is set at .02, the hill-climber does even better than in the experiments described above, showing mean performance above .992 at that setting.

6 MORE COMPLICATED ALGORITHM TOPOLOGIES

In the remainder of this paper, we shall consider more complicated algorithm topologies. We use the term "algorithm topology" to refer to a set of algorithms linked to form a directed, acyclic graph whose links represent seeding relationships. These algorithms may be of the same type or of different types, and individual members of the topology may have parameter settings that vary from other algorithms of the same type in the topology. For example, Figure 4 shows a topology consisting of five hill-climbers (represented by an icon containing a hill-climbing figure) and five genetic algorithms (represented by an icon containing a male-female couple). This topology has seeding relationships that proceed from genetic algorithm to hill-climber to genetic algorithm, and so forth. The different hill-climbers and genetic algorithms have different settings for the step size parameter. The first algorithm to run in the topology will be the leftmost genetic algorithm, which has a step size of .05. Its best solution will seed the leftmost hill-climber, which also has a step size of .05. The hill-climber's best solution seeds the second genetic algorithm, which has a step size of .04, and so forth. The effect of running this topology will be that we will alternate between an evolutionary approach and a hill-climbing approach, always using the best solution found as a seed for the next algorithm. As we proceed through the run, the step size of the mutation operator will decrease in size until, on the final run of the genetic algorithm and the hill-climber, the step size is .01.

Figure 5 shows the results of running a number of different algorithm topologies on f6. In order to explain the results, we need first to explain what these algorithm topologies were.

Our experiments used algorithm topologies constructed from basic topology units. We used five different types of topology units in these experiments, and each was a variation on a basic GA$\rightarrow$HC chain. In our notation, "1 GA$\rightarrow$HC chain" refers to a single GA$\rightarrow$HC pair, with the genetic algorithm seeding the hill-climber. The notation "2 GA$\rightarrow$HC chain" describes two pairs of algorithms, each pair consisting of a genetic algorithm seeding a hill-climber. If this pair is run in serial fashion (Figure 6b), the output of the hill-climber in the first pair will seed the initial genetic algorithm of the second pair. If it is run in parallel (Figure 6a), then both pairs of algorithms will run without communicating, and the best solution found by either one will be returned as the solution found by the topology. To illustrate this, please refer again to Figure 4, which shows a 5 GA$\rightarrow$HC chain run in serial fashion. As shown in Figure 4, we interpolated the step size for all algorithms in the topologies in the current set of experiments from .05 to .01.

In addition to experimenting with parallel and serial topologies of the GA$\rightarrow$HC chains, we also experimented with higher-level configurations of the basic serial topologies. Figures 6c, 6d, and 6e show three types of hybrid topologies whose performance is reported in Figure 5.

Each of these three hybrids is a serial topology based upon the same components. Hybrid1, as shown in Figure 6c, is a serial topology with two similar component blocks, each of which, in this example, is a 3 GA$\rightarrow$HC chain running serially. (Note that the output of *each* hill-climber in the first block seeds the genetic algorithm that begins the second block.) Hybrid2, as shown in Figure 6d, is a serial topology with three similar component blocks, connected serially. Hybrid3, as shown in Figure 6e, is a topology containing four similar component blocks, connected serially.

It should be noted that in our experiments, each of these topologies was given the same number of evaluations to use in solving the problem. The number of evaluations in these experiments is 3,000. In the case of a 1 GA $\rightarrow$ HC chain, performance with 3,000 evaluations was shown in Table 2. In the current set of experiments, the genetic algorithm runs for 1500 evaluations and seeds the hill-climber, which also runs for 1500 evaluations, and returns its best solution. Thus, for this configuration, we have 2 algorithms sharing the 3000 evaluations equally.

It is worth noting that for the 5 GA$\rightarrow$HC chain run under the hybrid3 regime, we have 10 algorithms in each component times four components = 40 algorithms running. Since they share the same number of evaluations as the 1 GA$\rightarrow$HC chain running serially, in this case each algorithm will have only 75 evaluations.

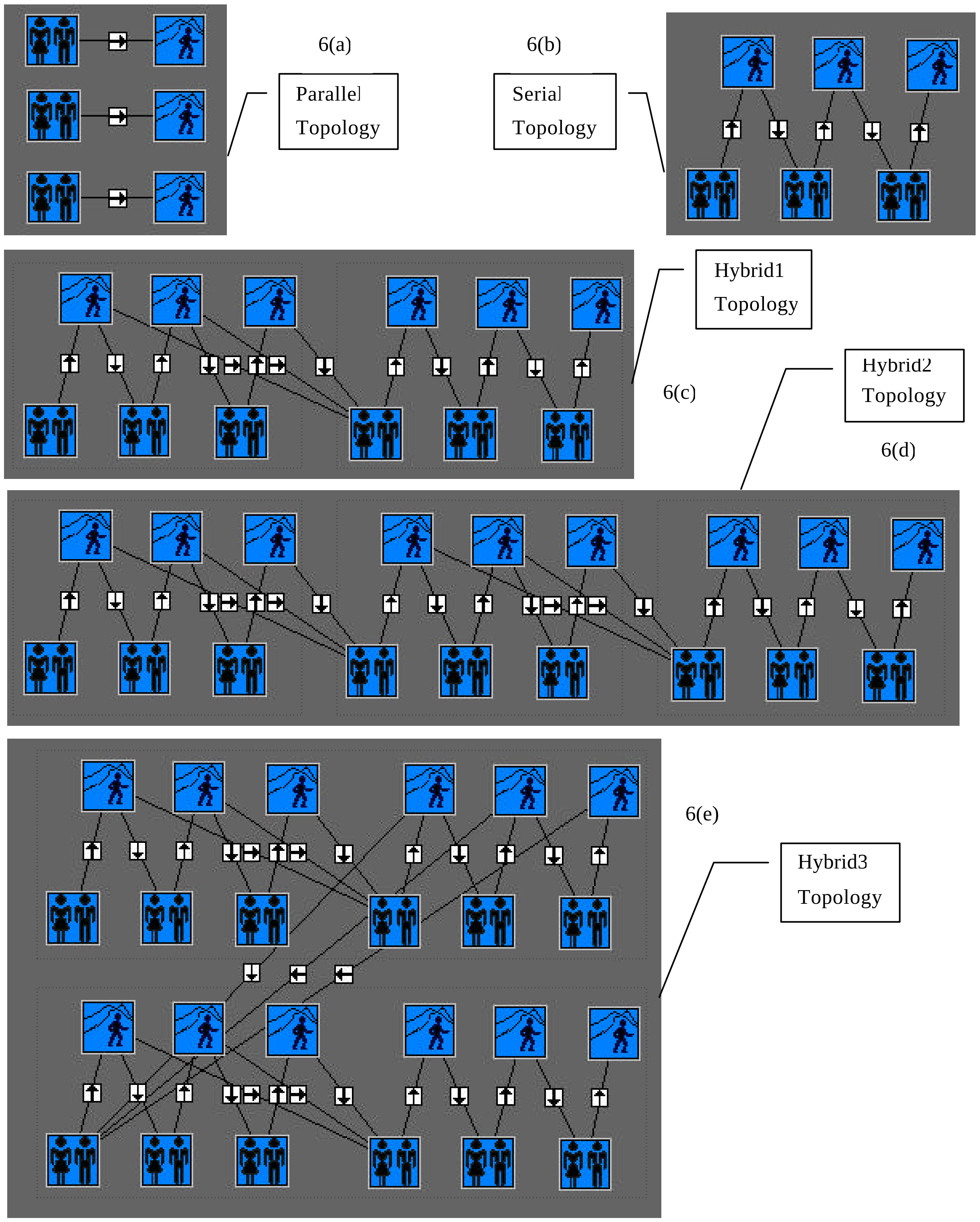

Figure 6. Topologies studied in Figure 5 using 3 GA $\rightarrow$ HC chains

The purpose of our experiments was to determine whether fine-grained topologies consisting of genetic algorithms and hill-climbers could do better than genetic algorithms and hill-climbers alone and, if so, just how fine the grain should be to get the best performance. Figure 6 shows the results. We see that the worst results were obtained by the parallel topologies, which consisted of independent GA→HC runs. The next-best results were obtained by the serial topology, which linked up those runs so that each GA→HC pair was seeded by the output of the prior pair. The next-best results were obtained by the hybrid1 topology, which duplicated the topology serial and seeded the second block of serial runs with the output of each hill-climber in the first block. The next-best was the hybrid2 topology, which used two serial blocks in serial. Finally, the hybrid3 topology used three serial blocks in serial, and did better than all other higher-level topologies.

It should be noted that each basic configuration—whether one, two, three, four, or five GA→HC pairs—did better under finer-grained topologies. It should also be noted that the best number of GA→HC pairs in the basic unit was 3. When four or five GA→HC pairs were used, results were not as good.

7 RESULTS FOR F17

In order to broaden the applicability of our results, we carried out similar studies for f17, a 30-dimensional, complicated mathematical optimization function described in (Baeck 1998). The results were similar. Using single algorithms, with parameters tuned to solve the problem, or using two algorithms in a seeding relation, optimal solutions could not be found by the best topologies in 10,000,000 evaluations. The convergence speed was so slow that we estimated it would take about 100,000,000,000 evaluations to get the optimal solutions. When we used the topology shown in Figure 7, with multiple iterations in which each run through the topology was seeded by the best result from the prior topology's run, the optimal solution was found, on average, in about 1,750,000 evaluations.

8 CONCLUSIONS

We have two principal conclusions from the results presented here. First, when the number of evaluations allowed is the same, the performance of simple topologies of our four types of optimization algorithms is vastly inferior to fine-grained, serial topologies of those algorithms. Second, we have shown that modifying the parameter values of those topologies across the optimization process leads to better results than holding them constant. We have shown that these conclusions obtain for the classical f6 function with two real-valued inputs, and for the much more difficult f17 function with 30 real-valued inputs.

Given that most current practice in evolutionary computation involves the use of simple topologies, we hope that these results will suggest refinements to our current practice that will allow us to find better solutions in the same amount of time.

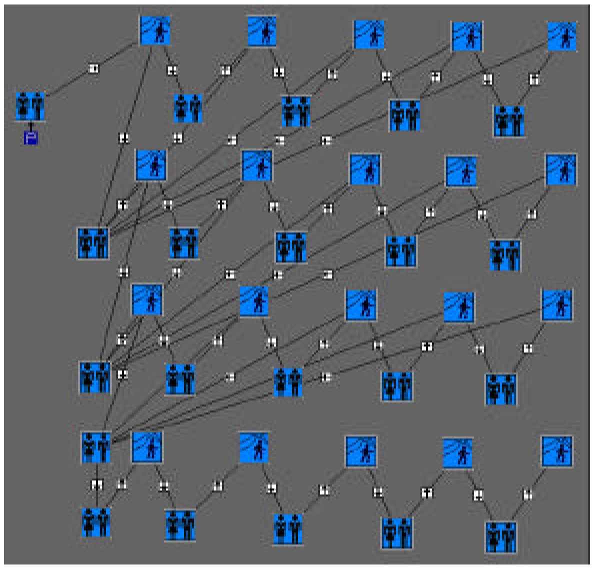
Figure 7 A best topology for f17

Note and Acknowledgement

We used HEURO (a system built by NuTech Solutions, Inc.) to display our topologies and gather data on our experimental runs. HEURO is a tool that makes it easy to create, edit, display, and evaluate the performance of algorithm topologies.

The authors would also like to thank Thomas Baeck for sharing his implementation of f17 with us.

References

T. Baeck and B. Naujoks (1998). Innovative methodologies in evolution strategies. INGENET Project Report D 2.2, Center for Applied Systems Analysis (CASA), Informatik Centrum Dortmund.

L. Davis (1990). *Handbook of Genetic Algorithms*, International Thompson Computer Press.

J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 51--60. Fitness Morgan Kauffman.

D. Powell, M. Skolnick, and S. Tong (1990). EnGENEous: A Unified Approach to Design Optimization. *Applications of Artificial Intelligence in Engineering* V (edited by J.S. Gero), Computational Mechanics Publications.