2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Symmetric Comparator Pairs In The Initialization Of Genetic Algorithm Populations For Sorting Networks

Lee Graham, Franz Oppacher
Carleton University
1125 Colonel By Drive
Ottawa, Ontario, Canada, K1S 5B6
lee@stellaralchemy.com oppacher@scs.carleton.ca

Abstract– **This paper concerns a well-known combinatorial optimization problem called sorting networks, which has been of some interest to the computer science and engineering community for many decades. A simple method for the initialization of genetic algorithm (GA) populations for the sorting network problem is presented in which networks are constructed with a structural symmetry, making use of symmetric comparator pairs. This method, and several variations thereof, are motivated and compared experimentally with a more standard initialization method. The benefits of symmetric comparator pairs on sorting networks, both before and after GA evolution, are demonstrated. Some interesting but unexplained observations regarding the distribution of such networks are presented as well.**

I. Introduction

The sorting network problem has a long and interesting history in computer science, the highlights of which are described in Knuth [1], including the oft-recounted chronology of a string of sorting network design records broken in rapid succession in the 1960's. In recent years, the application of evolutionary algorithms to this problem has turned out a number of human-competitive results, such as those of Juillé [2], Hillis [3], and others.

Sorting networks are an example of what are called oblivious sorting algorithms, in which the steps taken do not depend of the results of previous steps, a fixed number of which occur in a fixed order. They are well suited for implementation in hardware. The fundamental components in a sorting network are the "comparators". Each comparator has two input lines and two output lines. Two values enter via the input lines; the comparator then compares (and may exchange) the pair of values, which emerge in sorted order on the two output lines.

By chaining together the right sequence of comparators operating on N total input lines (indexed 0 to N-1), any list of N items can be sorted. The Zero-One Principle guarantees that if a sorting network can sort all possible binary input lists (each input line receives either a zero or a one) then it can sort any arbitrary input list. Thus, a method of checking the validity of such a sorting network is to try it on all 2^N binary lists of size N, and verify that the output is correctly sorted in each case. The total number of such binary lists that are properly sorted by a candidate network (hereinafter, the network "score") is an obvious choice for use as a measure of network quality.

A small 8-input sorting network is depicted in Fig. 1. The data travel from left to right in the network, across the horizontal input/output lines, while the comparators in the network (vertical lines) swap values that are out of order.

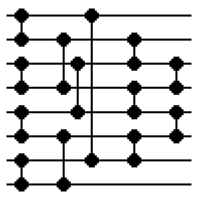

Fig. 1. An example of a sorting network.

A group of comparators that operate independently of one another (hereinafter, a network "level") can perform their comparison-exchange operations simultaneously. The network in Fig. 1, for example, consists of four such levels, and can therefore execute a sort in four time units. Sorting network designs can be optimized for speed (minimization of level count), or size (minimization of comparator count). Many current record-holding network designs for both time- and space-optimization are documented in Knuth [1].

Since the 1960's, much research has gone into this challenging combinatorial optimization problem. Recent noteworthy examples, from the field of evolutionary computing and elsewhere, include [2,3,4,5,6,7,8,9].

In this paper we explore the effects on network quality of several higher-level layout constructs in the generation of random candidate networks. We also look at how the use of such constructs affects the performance of a simple GA when used in population initialization. Sections II and III describe these higher-level constructs. Section IV outlines the motivation for their creation. Sections V and VI provide experimental results that support their usefulness. Section VII reports an interesting and unexpected observation, left as an open puzzle. Finally, section VIII concludes with a summary of the findings and their implications.

II. Symmetric Comparator Pairs

The term **symmetric comparator pair** refers to either of the following:

- Two comparators in an N-input sorting network, which are independent of one another and for which the index of the top input line of each is equal to N - j - 1, where j is the index of the bottom input line of the other (each is the vertical mirror image of the other).
- A single comparator in an N-input network, which is its own vertical mirror image (top input line index = N - bottom input line index - 1).

This definition renders the word “pair” somewhat of a misnomer; nevertheless, we will use the term to refer to both possibilities. Three examples of symmetric comparator pairs in 16-input networks are shown in Fig. 2.

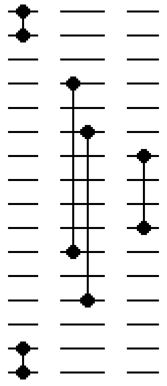

Fig. 2. Three examples of symmetric comparator "pairs".

When generating candidate sorting networks, such as would be done during GA population initialization, top and bottom comparator input line indices can be chosen independently of other comparators in the network. The term **asymmetric** will be used to describe such a network. Accordingly, we will use the term **symmetric** to denote networks generated in such a manner as to ensure that each comparator is part of a symmetric comparator pair.

The following two hypotheses are tested in the experiments described below (sections V and VI):

1. Symmetric networks, on average, score higher than asymmetric networks.
2. This improvement in score will translate to improved GA performance if symmetric networks are generated during population initialization.

III. Force-Filled Level (FFL) Networks

A variation on symmetric networks, which we will call **FFL symmetric**, takes into account a further aspect of higher-level network structure. A network level will be said to be force-filled if it contains *k* comparators confined to the innermost 2*k* lines of the network. An FFL network is one in which the first $L > 0$ levels are force-filled, and in which the number of comparators in each of those *L* levels is non-increasing, from left to right.

FFL networks with N inputs can be divided into single-tier and multi-tier networks – single-tier being those for which all force-filled levels contain N/2 comparators (the maximum possible for any level in the network), and multi-tier being those in which some force-filled levels contain less than N/2 comparators. Each tier, then, contains a consecutive group of force-filled levels of equal size.

For simplicity, we will denote and distinguish such FFL network configurations by listing the number of levels in each tier, separated by underscores. Thus, in a three-tiered 4_2_2FFL network, the first four levels (tier 1) contain N/2 comparators each, the fifth and sixth levels (tier 2) contain N/2 – 1 comparators each, and the seventh and eighth levels (tier 3) contain N/2 – 2 comparators each. Any subsequent comparators (if any) have unspecified placement.

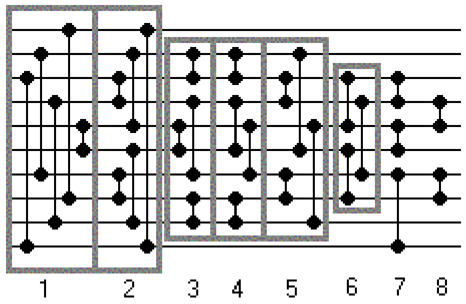

Fig. 3. An example of a 10-input 2_3_1FFL symmetric network. The eight levels of the network are numbered.

An FFL network will be said to be FFL symmetric if each of its force-filled levels consists entirely of symmetric comparator pairs. Fig. 3 shows an example of a three-tiered 2_3_1FFL symmetric network, with each tier and force-filled level indicated by an enclosing rectangle.

The experiments described in sections V and VI will attempt to determine whether or not randomly generated FFL symmetric networks tend to have higher scores than randomly generated symmetric and asymmetric networks, and whether or not such improvements translate to an increase in GA success rates when FFL symmetric networks are used in population initialization.

IV. Motivation

The motivation for examining the effects and possible benefits of symmetric and FFL symmetric networks stems from observations of the structure of many sorting networks presented in the technical literature. The authors noticed that a great many such networks, which are among the most efficient known (in terms of comparator count and/or number of levels), are composed, in large part, of symmetric comparator pairs, as described in section II, and contain force-filled levels of non-increasing comparator counts – the multi-tier FFL symmetric configuration described in section III. Although there are exceptions, it seems reasonable to explore the question of whether or not such a pattern might provide useful insights into the nature of the sorting network problem.

We present three 16-input sorting network examples from the technical literature to illustrate the pattern described above. The first (Fig. 4) is the record-setting network from Green [10], discovered in 1969. It has held the minimum comparator count record ever since, with 60 comparators. It is composed *entirely* of symmetric comparator pairs, and is a two-tier 4_1FFL symmetric network.

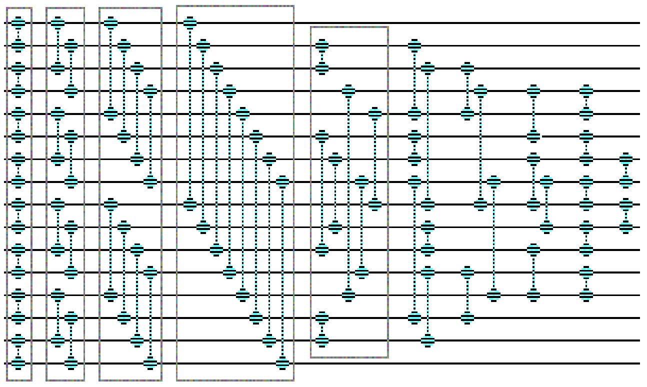

Fig. 4. Green's [10] record-setting 16-input network is 4_1FFL, and all 60 comparators are symmetrically “paired”.

The second example (Fig. 5) is that of Hillis [3]. The latter half of this network was generated using a co-evolutionary algorithm, with the first 32 comparators taken directly from Green’s solution. It is composed almost entirely of symmetric comparator pairs, and is a 4_1FFL network, though not 4_1FFL symmetric.

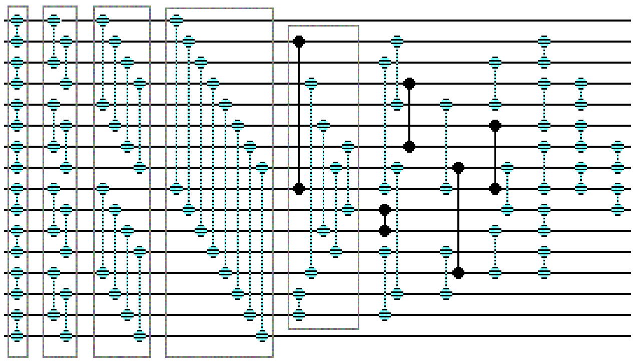

Fig. 5. Hillis' [3] 16-input network is 4_1FFL. Comparators not symmetrically “paired” are bolded.

The final example (Fig. 6) is from Juillé [2]. Like Hillis’ solution, this network is also the product of an evolutionary algorithm. Although it exhibits less symmetry than the previous two, more than half of its comparators are symmetrically “paired”, and it too is a 4_1FFL network, though not 4_1FFL symmetric. Many other published optimal and/or recording-holding networks show the same patterns, such as those shown in Knuth [1].

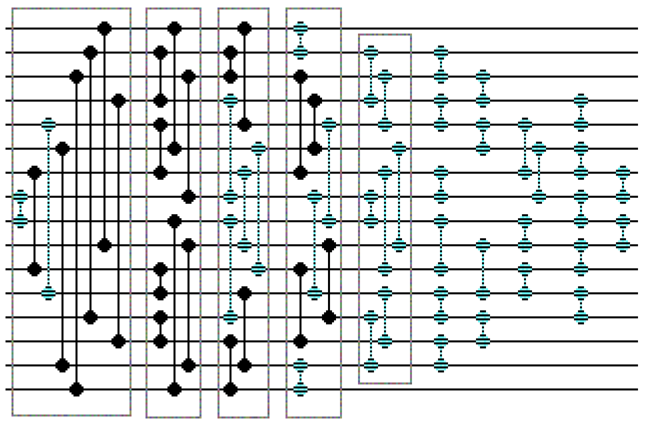

Fig. 6. Juillé's [2] 16-input network is 4_1FFL. Comparators not symmetrically "paired" are bolded.

V. Score Distributions for Random Networks

The first set of experiments performed by the authors examined the score distributions of randomly generated networks with asymmetric, symmetric, and FFL symmetric configurations. For the 10-input sorting network problem, and for each chosen configuration, two million random networks were generated and evaluated. The resulting score frequency distributions and their means were calculated and plotted for analysis.

The first two frequency distributions generated were those for asymmetric networks (wherein each comparator is generated independently), and symmetric networks (wherein each comparator is generated to be symmetrically “paired”). The resulting frequency distributions show a clear lensing effect (Fig. 7). Although the distribution mean for the symmetric configuration is slightly higher than that of the asymmetric configuration (Fig. 11), this spreading or lensing is of much greater significance than the mean when it comes to GA population initialization. The GA is, in part, a filter, and should benefit from higher-quality networks represented by the chubby tail at the right of the distribution.

The second group of distributions generated were those for five different single-tier FFL symmetric configurations – 1FFL, 2FFL, 3FFL, 4FFL, and 5FFL (Fig. 8). Each of these distributions shows progressively less spreading than the simple symmetric configuration, while the means continue to increase (Fig. 11 shows means for 1FFL, 2FFL, and 5FFL only). Each also remains above the symmetric curve’s right tail, if only slightly.

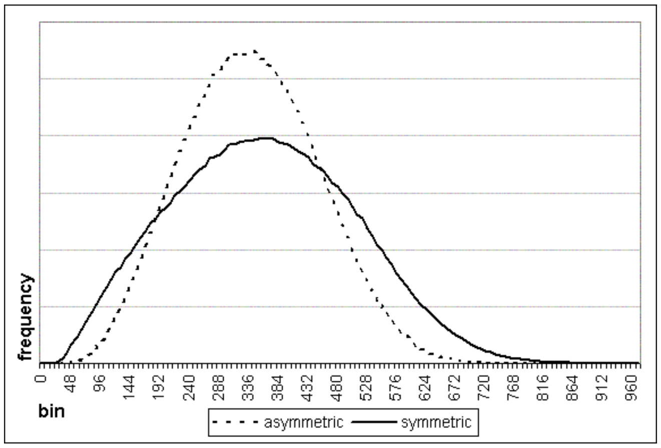

Fig. 7. Score distributions for asymmetric and symmetric random networks (bin size 8). Each distribution based on two million networks.

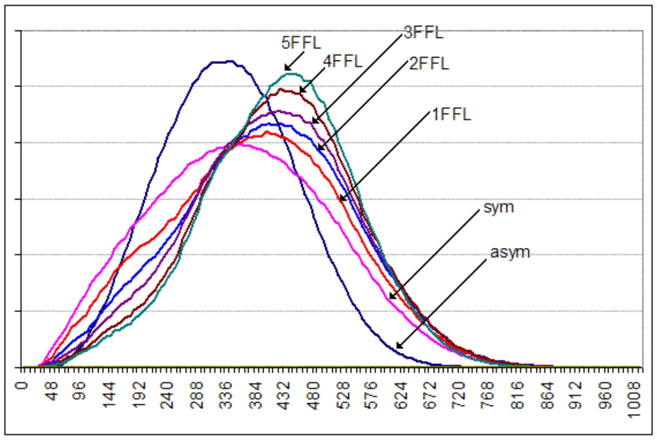

Fig. 8. Score distributions for single-tier FFL symmetric random networks (bin size 8). Asymmetric and symmetric distributions included for comparison. Each distribution based on two million networks.

With the sole exception of 1_6FFL symmetric, the multi-tier FFL symmetric distributions show further improvements still. Nearly all sit high above the right tail of the symmetric distribution curve (Fig. 9). A corresponding increase in means (again with the exception of 1_6FFL symmetric) is shown in Fig. 11, with 3_3FFL symmetric coming out on top. Fig. 10 takes a closer look at the right tails of these distribution curves to reveal that the 3_3FFL symmetric configuration also sits above the others at the high end of the range.

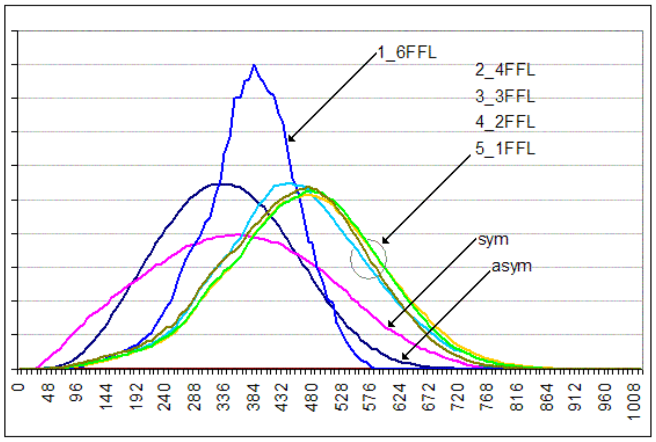

Fig. 9. Score distributions for two-tier FFL symmetric random networks (bin size 8). Asymmetric and symmetric distributions included for comparison. Each distribution based on two million networks.

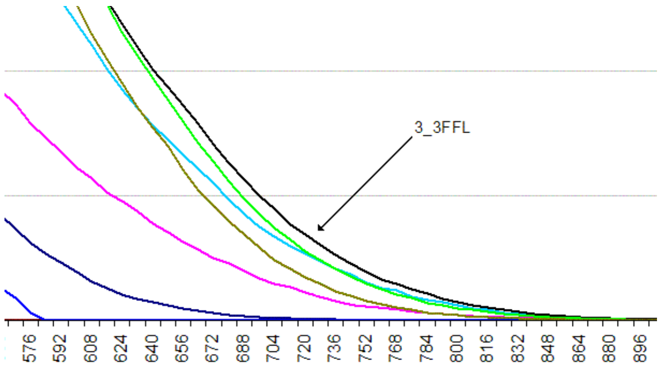

Fig. 10. Zoomed view of score distributions from Fig. 9 showing that the 3_3FFL symmetric random network distribution has the chubbiest tail.

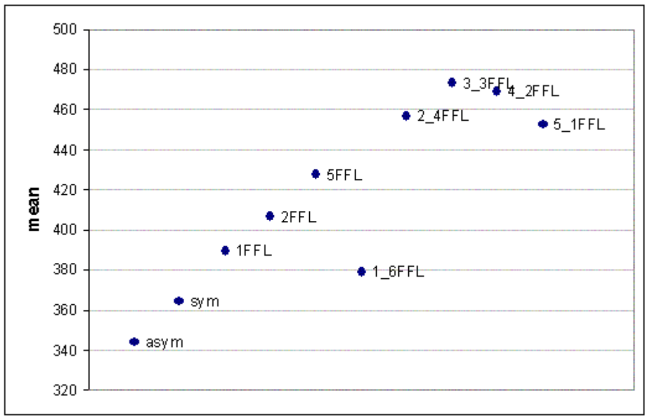

Fig. 11. Means for asymmetric, symmetric, and FFL symmetric random network distributions. Each value based on two million networks.

VI. The Effect on GA Success Rates

To determine whether the benefits shown in section V carry through in GA evolution, each network configuration, in turn, was applied during population initialization of a simple GA. The GA used a population size of 500, a halting condition of 200 generations, size 8 tournament selection, a crossover rate of 10% (crossover operator: 30% uniform, 70% single-point), and a per-individual mutation rate of 40% (mutation operator: indirect replacement – a comparator is chosen at random and deleted, while a new one is inserted at a random location). Duplicate individuals were forbidden, and detected via a hash value. These GA settings are the end result of parameter tuning conducted previously by the authors [11]. The particular symmetric and asymmetric network configurations apply only to population initialization. Thereafter, mutation and crossover are free to disrupt symmetry and/or high-level FFL layout of the networks.

For each configuration, the GA was run 5,600 times independently and the success rate was calculated, with a perfect network score of 1024/1024 constituting success.

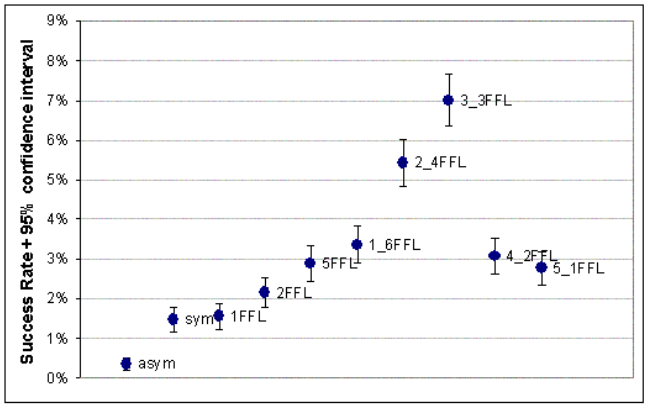

Fig. 12. GA success rates for asymmetric, symmetric, and FFL symmetric random network population initialization techniques. Each value based on 5,600 independent runs.

The graph in Fig. 12 shows the outcome of the trials. The overall pattern closely matches that of the random network distribution means in Fig. 11, however the relatively small differences in the means for multi-tier FFL symmetric configurations translate to considerable improvements in GA success rate, especially for the 3_3FFL symmetric layout.

VII. Score Distributions and Parity

The authors provide the following interesting observations regarding the score distributions of random symmetric networks, with no accompanying explanation. No such explanation has yet been uncovered by the authors, and we leave it as open puzzle for the keen and analytically-minded reader. These observations were hit upon quite unexpectedly when random network distributions from section V were plotted with bins of size one.

The size one bins reveal that certain score distributions are composed of two distinct distribution curves: those with even scores, and those with odd scores. This parity-related division seems to exist only for the symmetric and FFL symmetric network layouts. Fig. 13 shows that the even and odd parity score distributions for randomly generated asymmetric networks are nearly identical, whereas those for symmetric random networks are clearly different (Fig. 14).

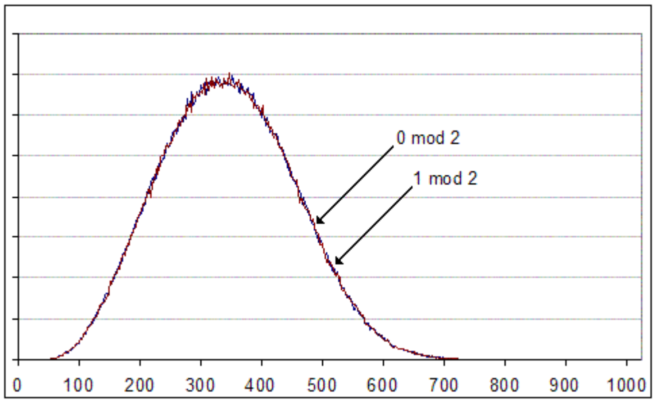

Fig. 13. Even and odd parity score distributions for asymmetric random networks (bin size 1). Distribution based on two million networks.

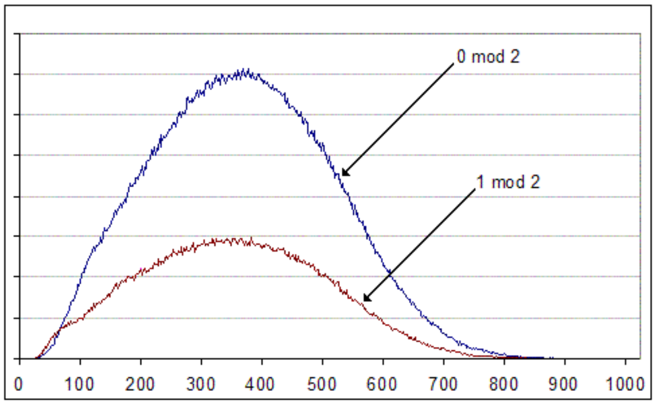

Fig. 14. Even and odd parity score distributions for symmetric random networks (bin size 1). Distribution based on two million networks.

Single- and multi-tier FFL symmetric score distributions also show a parity-related division, but unlike the two-way split of symmetric networks, the divisions extend down to modulo four, showing three distinct curves: two for even parity scores, and one for odd parity scores (Fig. 15). It is unclear exactly why such divisions should exist.

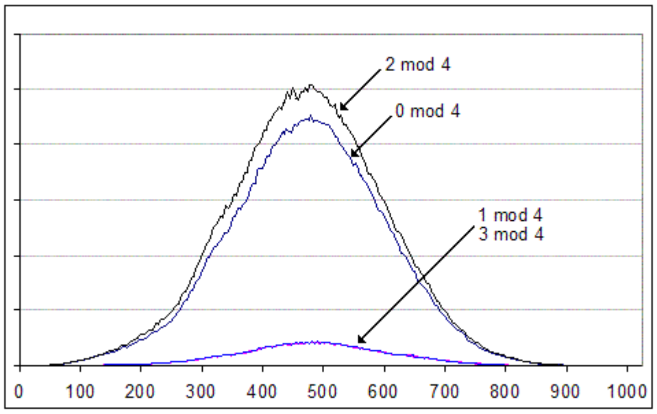

Fig. 15. Distributions of scores congruent to 0, 1, 2, and 3 modulo 4 for 3_3FFL symmetric random networks (bin size 1). Distribution based on two million networks.

VIII. Conclusions

The two hypotheses stated in section II appear to be supported by the experimental and observational evidence presented in sections V and VI. The increase in mean scores for symmetric and FFL symmetric network configurations seem to indicate that the higher-level network constructs of symmetric comparator pairs and force-filled levels are, all things being equal, beneficial with respect to network scores for randomly generated networks. The GA experiments from section VI appear to confirm that these higher-level constructs' benefits carry through in GA evolution, resulting in higher success rates for an equal amount of computational effort – a result strongly hinted at by the chubby-tailed score distributions. The authors believe these results may be of use to those interested in the sorting network problem since the symmetric network configurations described herein are easily implemented and provide a straightforward means of producing random networks with scores from a distribution with high mean and wide range. From a more theoretical perspective, these experiments highlight the importance of population initialization techniques in GAs, and the impact such techniques can have on performance.

References

[1] Knuth, D. (1998) *The Art of Computer Programming, Volume 3: Sorting and Searching (2nd edition).* Addison Wesley.

[2] Juillé, H. (1995) *Evolution of Non-deterministic Incremental Algorithms as a New Approach for Search in State Spaces.* In Proceedings of ICGA-95. Morgan Kaufmann, pp. 351-358.

[3] Hillis, D. (1991) *Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure.* In Proceedings of Artificial Life II. Westview Press.

[4] Choi, S.S., & Moon, B.R. (2001) *A New Approach to the Sorting Network Problem Evolving Parallel Layers.* In Proceedings of GECCO-2001. Morgan Kaufmann, pp. 258-265.

[5] Choi, S.S., & Moon, B.R. (2002) *Isomorphism, Normalization and a Genetic Algorithm for Sorting Networks.* In Proceedings of GECCO-2002. Morgan Kaufmann, pp. 327-334.

[6] Choi, S.S., & Moon, B.R. (2002) *More Effective Genetic Search for the Sorting Network Problem.* In Proceedings of GECCO-2002. Morgan Kaufmann, pp. 335-342.

[7] Harrison, M.L., & Foster, J.A. (2004) *Co-evolving Faults to Improve the Fault Tolerance of Sorting Networks.* In Proceedings of EuroGP 2004. Springer-Verlag.

[8] Lukás, S. (2004) *Evolving Constructors for Infinitely Growing Sorting Networks and Medians.* In SOFSEM: Theory and Practice of Computer Science. Springer, pp. 314-323.

[9] Piotrów, M. (2004) *Depth Optimal Sorting Networks Resistant to k Passive Faults.* SIAM Journal on Computing, Volume 33, Number 6, pp. 1484-1512.

[10] Green, M.W. (1973) In Sorting and Searching, Volume 3: The Art of Computer Programming, edited by D. Knuth. Reading, MA: Addison-Wesley.

[11] Graham, L., Masum, H., & Oppacher, F. (2005) *Statistical Analysis of Heuristics for Evolving Sorting Networks.* In Proceedings of the 7th Annual Genetic and Evolutionary Computation Conference (GECCO 2005). ACM Press, pp. 1265-1270.