CrossNet: A Framework for Crossover with Network-based Chromosomal Representations

Forrest Stonedahl[†], William Rand[†‡], Uri Wilensky[†‡]
[†]Center for Connected Learning and Computer-Based Modeling
[‡]Northwestern Institute on Complex Systems
Northwestern University
Evanston, Illinois, USA
{forrest, wrand, uri}@northwestern.edu

ABSTRACT

We propose a new class of crossover operators for genetic algorithms (CrossNet) which use a network-based (or graph-based) chromosomal representation. We designed CrossNet with the intent of providing a framework for creating crossover operators that take advantage of domain-specific knowledge for solving problems. Specifically, GA users supply a network which defines the epistatic relationships between genes in the genotype. CrossNet-based crossover uses this information with the goal of improving linkage. We performed two experiments that compared CrossNet-based crossover with one-point and uniform crossover. The first experiment involved the density classification problem for cellular automata (CA), and the second experiment involved fitting two randomly generated hyperplane-defined functions (hdf's). Both of these exploratory experiments support the hypothesis that CrossNet-based crossover can be useful, although performance improvements were modest. We discuss the results and remain hopeful about the successful application of CrossNet to other domains. We conjecture that future work with the CrossNet framework will provide a useful new perspective for investigating linkage and chromosomal representations.

Categories and Subject Descriptors: I.2.m [Artificial Intelligence] Misc.

General Terms: Algorithms

Keywords: Genetic Algorithms, Recombination, Crossover, Linkage, Networks, Graphs

1. MOTIVATION

From the early days of genetic algorithm research, it has been known that the linkage between bits in the representation of solutions plays an important role in the GA's effectiveness at solving problems[7]. Preserving closely related beneficial groups of bits from parent to child is generally advantageous. During the creation of a new generation, if the recombination operator is overly "disruptive" with regard to a solution's structure, highly fit individuals will have more difficulty passing on their good genetic material. When faced with a real world problem, practitioners of genetic algorithms have several options for dealing with this issue:

1. They can use a standard crossover operator (e.g., one-point) which has been theoretically and empirically shown to perform well when the linked genes appear close together in the chromosomal representation. In this case, the dilemma is shifted to the best way of ordering a linear string of bits. Practitioners may find it difficult to create a representation that keeps all related bits close together.
2. They can use one of several extant genetic algorithm variations that incorporate linkage learning techniques (cf. [1]), and hope the algorithm is smart enough to uncover the underlying structure. However, linkage learning techniques often come at a substantial cost in running time.
3. They can choose to ignore the problem and trust that evolution will overcome the problems of representation. Evolution is versatile but its progress will inevitably be limited by the efficacy of the representation choice.

We are proposing a fourth option, which involves a simple and intuitive way to incorporate domain knowledge about the structure of the solution into the simple GA, with only a slight modification to the standard GA operators. The engineer can draw (or otherwise construct) a network where the genes are nodes, and two nodes are connected by a link if the two genes are related.[1] Weights may also be assigned to the links to allow the engineer to describe a more fine-grained level of gene epistasis. CrossNet-based crossover uses this network to create children by splitting parents in ways that should (probabilistically) maintain good linkage, provided that the crossover network reflects the epistatic relationships in the given problem.

At this point, the skeptical reader may be wondering if the term "CrossNet" is merely disguising a mechanism for forcing the user to do the difficult task of specifying the best linkage between the genes. Not exactly. We are not asking the user to fill in an $L \times L$ table of co-linkage entries – a task we believe they would find extremely difficult. Instead,

[1]In this paper we will use the terms "network", "node", and "link" interchangeably with the mathematical terms "graph", "vertex", and "edge".

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

we are asking them to organize the genes with more freedom than is typically permitted when using a linear chromosomal representation. This additional flexibility allows a domain expert to use as many dimensions as necessary to cluster related genes together and position unrelated genes far apart. In essence, the practitioner is tasked with describing what the epistatic interactions between genes are, and CrossNet attempts to use this information to maintain good linkage. Oftentimes, this task will be almost as easy for the user as determining a linear chromosomal structure, which they would have needed to do otherwise. We note that this technique gives up some of the biological plausibility of some crossover operators since DNA strands do have a linear structure. However, it is well-known that the interaction of proteins happens in a non-linear way, and so there may be a relationship between CrossNet and biological genotype-to-phenotype mapping.

We hypothesize that a GA using the CrossNet representation should be able to outperform a simple GA. We will investigate this hypothesis using two example domains: evolving cellular automata (CA) rules for the density classification problem (DCP), and finding maximally-valued strings for randomly created hyperplane-defined functions (hdf's). We begin by backgrounding our discussion with some related research, and then proceed to explain the CrossNet architecture more precisely. After that we will present and discuss the results from the experiments in the two domains, and we will conclude by suggesting avenues for future research.

2. RELATED WORK

Our work is partially inspired by research on modified crossover operators. There have been various permutation-based recombination operators, such as partially mapped crossover (PMX) which showed improved results on the traveling salesperson problem [5]. Another example is Falkenauer's grouping genetic algorithm, which was designed to handle grouping problems (e.g., bin packing) [4]. Further examples of modified crossover operators can be found in Chen et al.'s survey on linkage learning [1]. However, we propose CrossNet not as a specific modification of the crossover operator to accommodate a certain class of problems, but as a generalized framework supporting the creation of crossover operators custom-tailored (by the choice of network) for many classes of problems. Nonlinear chromosomal structures have also been investigated in previous work. Various chromosomal representations have been employed in the past, both to more closely mirror the structure of the problem in question and to support crossover operators that better preserve building blocks [16]. Kahng and Moon employed geographic crossover, in which bits were arranged in 2D or 3D arrays [11]. Most recently, there has been research into two chromosomal representations that specify all pair-wise genic distances, which may be viewed as a weighted complete graph structure. These include Greene's GA with "self-distancing bits" [6], and the Voronoi quantized crossover employed by Seo and Moon [15]. However, to our knowledge, we are the first to investigate the use of arbitrary network (graph) structures for chromosomal representation and crossover. Other researchers have performed crossover on networks for the sake of evolving the network (or graph) itself (e.g., neural network topologies [17], turing machines [13]). In contrast, we are not evolving the network, but using a network as a chromosomal representation.[2] For a recent comprehensive survey of work on different chromosomal structures that exploit topological linkage, see [16]. Estimation of distribution algorithm (EDA) researchers have also addressed similar questions about interactions between alleles within chromosomes by either specifying or learning a Bayesian probability network that describes relationships in the chromosome (e.g. BOA [12]). While our approach lacks the full expressive power of Bayesian modeling, it offers an advantage of simplicity in that it uses a simple GA and a straightforward graph-based representation for gene interactions. However, given past research into network structures in EDAs, the relationship between CrossNet and EDAs warrants further investigation.

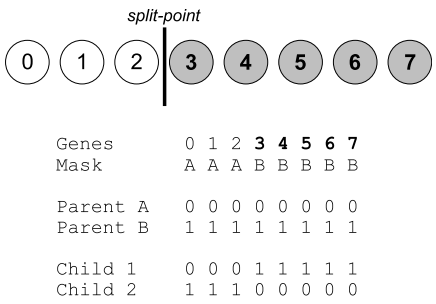


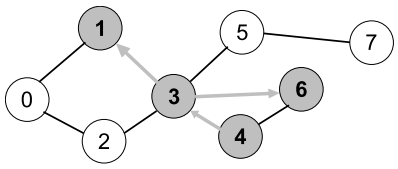
Figure 1: An example of one-point crossover on an 8-bit chromosome. For simplicity, parent A is all zeros and parent B is all ones.

3. THE CROSSNET FRAMEWORK

To understand the CrossNet framework, let us briefly review one-point crossover (Figure 1) on an 8-bit chromosome. In one-point crossover, a splitting point is chosen at random on the linear chromosome. This splitting point defines a mask that controls which genes will be taken from each parent to form the two children. The mask specifies for Child 1 which parent (A or B) will be used as the source for each gene, and the complement of the mask is used to create Child 2. Compare this to CrossNet crossover on an 8-bit network (Figure 2). The network topology would have been provided by the human practitioner, and it is currently invariant across the population.[3] The nodes of the network are divided into two sets (shown as white and gray in Figure 2) through a process that we will explain shortly. The division of the nodes in the network defines a crossover mask. Two child chromosomes are created from the two parent chromosomes using this mask. By using this "mask" mechanism, the rest of the GA can remain unchanged. There is no necessity to complicate code by using a network structure to store the genetic information for the individuals in the population. Individuals store their information as linear strings,

[2]However, the crossover network in CrossNet could be evolved (or co-evolved), as we discuss in Section 6.

[3]Individuals could easily maintain heterogeneous network structures, but a standard way of passing the network structure from parent to child would need to be devised.



Genes	0	**1**	2	**3**	**4**	5	**6**	7
Mask	A	B	A	B	B	A	B	A
Parent A	0	0	0	0	0	0	0	0
Parent B	1	1	1	1	1	1	1	1
Child 1	0	1	0	1	1	0	1	0
Child 2	1	0	1	0	0	1	0	1

Figure 2: An example of CrossNet crossover using a simple 8-bit network. The gray arrows show how nodes were chosen by spreading from node 4.

and it is only the creation of the mask during the crossover operation where the network topology matters.

Let us address the question of how the crossover network should be divided in two. Our splitting algorithm is simple. We choose the size of the splitting subset (B below) to be between zero (0) and the number of nodes (exclusive). We create B by starting with a random node, and repeatedly add nodes by selecting random edges that lead to a node not yet in B until we have reached the target size. The arrows in Figure 2 show which edges were chosen to increase the splitting set. Formally, the algorithm is specified below:

1. Input: a graph $G = (V, E)$
   (the crossover network, with vertex set V, edge set E)
2. Let set $A = V$. Let set $B = \emptyset$.
3. Choose T (the target size for set B) randomly from the integers $1 \ldots (|V| - 1)$.
4. Choose a random vertex from set A and move it to B.
5. **WHILE** $|B| < T$:
   (a) Choose an edge $e = (a, b)$ randomly from E, such that $a \in A$ and $b \in B$
      i. If G is weighted, use roulette selection.
      ii. If no such edge exists (which may occur if G is not connected), choose random $a \in A$.
   (b) Move a from set A to set B.
6. Output: Sets A and B, from which a mask is created.

There are many other splitting methods that could be employed, such as breadth-first search starting from a random node, a random walk on the nodes of the network, or choosing a cutting edge set. We think the important feature of any splitting algorithm is that it should (probabilistically) divide the graph such that nodes are more likely to be in the same set after splitting if they were closer to each other in the crossover network. Defining this property of desirability more rigorously and finding an optimal splitting algorithm for crossover networks requires further research.

To test the CrossNet framework, we developed two experiments which we will describe in Sections 4 and 5. We implemented both experiments using NetLogo [21], which provided us with parameter sweeping and data collection facilities. We connected NetLogo with a custom Java extension to efficiently handle low-level (e.g., bit string) operations, and we devoted substantial computational effort to our experiments.[4] The basic parameters of our GA were held constant for both experiments. We used a simple genetic algorithm, with crossover rate 0.7 and mutation rate $\frac{1}{2L}$ (where L is the length of the bit string). Our population size was 100 individuals, and we used tournament selection with tournament size 3 (no elitism).

4. EXPERIMENT 1: DCP

4.1 DCP setup

The density classification problem (DCP) for cellular automata has been the topic of several previous research studies, and considerable effort has been put toward evolving high fitness solutions (e.g., [20]). The DCP can be described in this way: Given an initial condition (IC) – a randomly initialized lattice – we are looking for CA rules that cause the lattice to converge to all ones if the initial density of the lattice (number of ones in the IC) is more than half, and to all zeros if the initial density is less than half. It is known to be a difficult problem – in fact, it was shown that no two-state CA rule exists that can perfectly solve the DCP [9]. We generally followed the experimental setup of Mitchell et al. [10]. More advanced techniques (e.g., coevolution [20]) have been shown to give better results, but our goal is to use the DCP as a benchmark to evaluate the effectiveness of CrossNet crossover, rather than to find particularly high performance solutions. We evolved a population of 100 rules for 100 generations on a lattice of width $w = 149$. Each generation we tested the rules against 100 new ICs chosen with uniform random density between 0.0 and 1.0 (with exactly half the ICs guaranteed have density over 0.5.) Following previous research [10] we evaluate the final state of the CA lattice after $2w$ (298) time steps. Fitness was defined as the number of ICs solved correctly, and no partial credit was awarded for partial convergence.

For one-point and uniform crossover[5], the genotype representation was the traditional lexicographic ordering of the bits of the CA rule table. Each bit corresponds to an input configuration of the lattice cell's neighborhood. A zero (0) or one (1) for that bit specifies the value of the cell in the next time step given that neighborhood configuration.[6] For our CrossNet crossover operator, we specified a 7-dimensional hypercube (hepteract) network, where each node (gene) is connected to those nodes whose configurations differ from the given node's configuration by exactly one bit. In a radius 3 CA rule, the gene specifying what to do with the input configuration 1111111 is linked to the genes specifying what to do with input configurations {0111111, 1011111, 1101111, 1110111, 1111011, 1111101, 1111110} (The simpler analogous case of radius 1 CA rules is shown in Figure 3).

In any linear ordering, it is only possible to put each gene next to (at most) two genes that control similar configura-

[4]Over 1000 hours of running time for Experiment 1.
[5]Actually, bit order is irrelevant for uniform crossover.
[6]For an introduction to 1-D cellular automata, see [10].

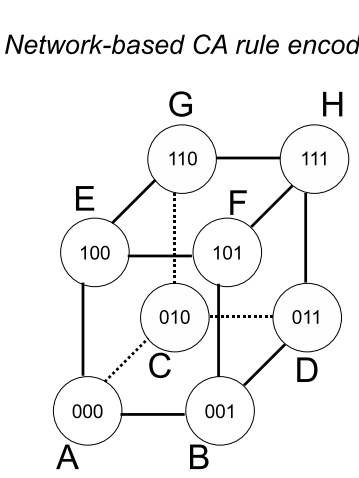


Figure 3: A cube crossover network for elementary ($radius = 1$) CA rules, compared to the standard linear representation. A–H map to the 8 possible 3-cell neighborhood configurations. For example, if $B = 1$, then the local lattice configuration 001 will cause a cell to become 1 in the next time step.

tions. The standard lexicographic ordering for CA rules does not even do this (e.g., the gene for configuration 00111111 is placed next to the gene for configuration 01000000, which differs in 7 bit locations). Our choice of network is intuitive because genes that correspond to similar local lattice configurations are placed close together. The assumption here is that genes controlling similar configurations are more likely to form good building blocks for CA rules. However, the vital question is whether we are linking the genes that work well together in the construction of a solution to the density classification problem, and it is quite possible that a better choice of crossover network exists. However, our hope was that our intuition-based network structure would be better than the (rather arbitrary) choice of a lexicographic ordering used by one-point crossover and the structureless representation used by uniform crossover.

4.2 DCP Results

For each of the types of crossover (one-point, uniform, and CrossNet), we ran 90 randomly-seeded repetitions of the experiment. The resulting best-of-generation plot is shown in Figure 4. In addition to measuring fitness, we also recorded the performance of the best individual in each generation when tested against 100 ICs drawn from an unbiased distribution, which tends to have many more ICs close to the difficult $\frac{1}{2}$ density threshold. The results on this unbiased distribution are similar to the results presented. The CrossNet-based crossover provides modest but sporadic improvement over uniform crossover. However, the improvement over one-point is not significant.

4.3 Further discussion of DCP results

In addition to the results shown here, we also ran several experiments with smaller lattice widths (99 and 49). With lattice width 99, CrossNet-based crossover (using the hypercube network) outperformed uniform, which in turn outperformed one-point. With lattice width 49, CrossNet-based crossover slightly outperformed uniform crossover in the early generations, but the performance of the three was very comparable. Since the DCP is easier to solve on smaller lattices, such as width 49, it may be that the choice of crossover operator does not have a large effect since all the operators were able to climb the fitness landscape more quickly. We also tried some experiments with a smaller population (50), and these generally showed that CrossNet had lower performance than the other types of crossover. However, this would be consistent with previous work from De Jong [2], which found that disruptiveness of crossover is actually a beneficial trait when population sizes are small.

Also, when we examined the histogram of best fitness values after the final generation, we discovered that the distribution is decidedly two-peaked. This indicates that some runs made an evolutionary break-through while others did not. For CrossNet-based crossover, 29 final fitness values were below 60, 7 were between 60 and 90, and 54 were above 90. For both uniform and one-point, 34 were below 60, 9 were between 60 and 90, and 47 were above 90. In other words, out of the 90 runs, 7 more runs made a substantial fitness breakthrough when using Crossnet-based crossover than with either uniform or one-point crossover. We find the current results on this problem to be somewhat promising, and worthy of future study. Even if the current hypercube network does not greatly increase performance, using CrossNet with a different network topology might facilitate greater improvements. Perhaps a better network could be constructed by examining the fitness landscape in the neighborhood around previously discovered high performance solutions, taking insight from the work on the DCP landscape by Verel et al.[18]

5. EXPERIMENT 2: HDF'S

5.1 Setup for hdf's

Hyperplane defined functions (hdf's) were created by Holland [8] to facilitate the study of genetic algorithms. The difference between this test suite and most other test functions is that the underlying representation of this suite is schemata [7]. By utilizing functions that reflect the way the GA searches, the GA's performance can be easily observed.

An hdf is composed of positive schemata and negative "pothole" schemata. For each schema that is matched by the genotype, the individual is rewarded (the fitness value is increased). And for each pothole that is matched, the individual is punished (the fitness value is decreased). There are elementary level schemata, which are the foundational elements, and intermediate-level schemata, which are composed of pieces of the elementary schemata.

We examined two hdf's with two levels of difficulty. These two hdf's were restricted from the general class of hdf's and corresponded to static instances of the shaky ladder hyperplane-defined functions (sl-hdf's) [14]. The first problem was a 100-bit hdf that contained ten (10) elementary schemata, seven (7) intermediate schemata, and ten (10) potholes. Elementary schemata were of length ten (10)

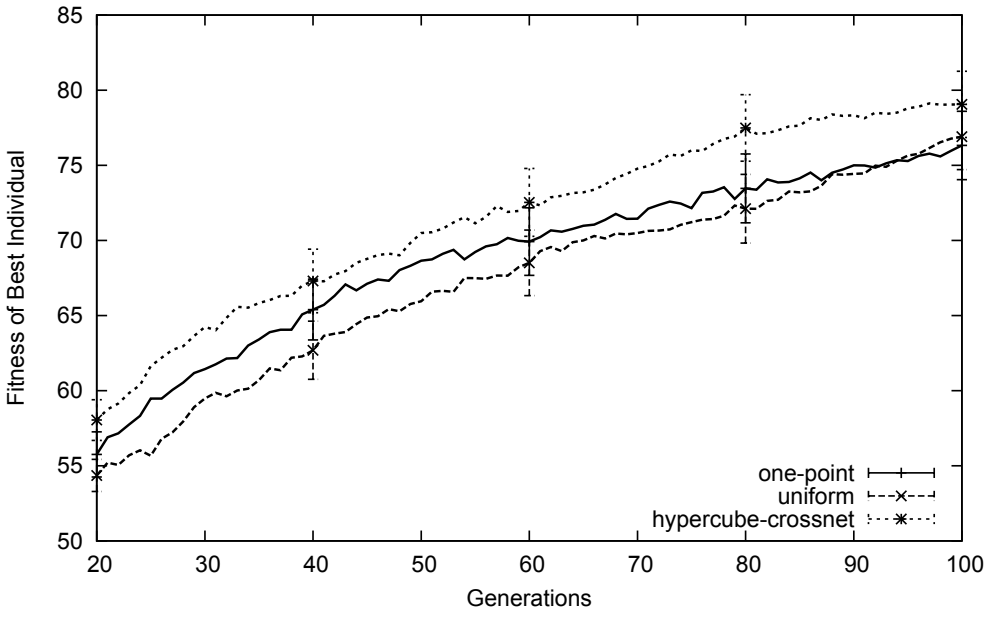


Figure 4: Best-of-generation plot for the DCP, with lattice width 149 and population 100. Points plotted are averaged across 90 runs, with standard error bars shown. *For clarity of presentation reasons, the first 20 generations are omitted.*

and order eight (8). We also used a 200-bit hdf that contained twenty (20) elementary schemata, seventeen (17) intermediate schemata, and twenty (20) potholes. Elementary schemata were of length twenty (20) and order eight (8). The 100-bit hdf is substantially easier than the 200-bit hdf. Unlike the DCP, for which the optimal solution remains an open research question, we know the perfect solutions to these hdf problems, and we know the components that make up the solution.

In fact, we used our knowledge of the elementary schemata when constructing the crossover networks for this experiment. This is equivalent to knowing which bit locations are involved in solving sub-problems of the larger problem, but not having the global knowledge of how these sub-solutions work together. Our plan was to start with a problem for which we have some understanding of the interaction between genes. We constructed our crossover network by creating cliques among the loci that form the elementary schemata. In other words, we create a link between two genes if the genes are present together in one of the elementary schemata. We also decided to test a weighted version of this crossover network, where the weight for each link is the number of elementary schemata in which both genes are present. Information about potholes and intermediate schemata were not taken into consideration when creating the network. The unweighted crossover network for the easier 100-bit hdf problem is shown in Figure 5.

5.2 Shuffled hdf's

Our initial experiments (that are not shown in this paper) examined standard hdf's and demonstrated that while CrossNet (using both weighted and unweighted clique networks) generally outperformed uniform crossover, its performance was similar to that of one-point crossover. In analyzing these experiments, we realized that hdf's have a natural linear bias, since schema are comprised of nearby bits in the linear representation. Thus, it is unsurprising that one-point crossover performed well on these problems. Since we wanted to test the CrossNet framework on a problem that is not naturally biased toward one-point crossover, we changed the standard representation of the hdf, by randomizing the ordering of loci in our two original hdf's. The results presented in the remainder of this paper use these "shuffled" hdf's.

5.3 Results for shuffled hdf's

We evolved a population of 100 individuals for 2000 generations, and we ran 180 repetitions of the experiment with different random seeds for each of the 4 types of crossover (one-point, uniform, CrossNet with the clique-network, and CrossNet with the weighted clique-network). The best-of-generation plot for the easier problem is shown in Figure 6, and the best-of-generation plot for the harder problem is shown in Figure 7.

A trend present in both problems was that in early generations uniform crossover outperformed one-point crossover and the two CrossNet variants (the clique and weighted-clique networks), but in later generations uniform crossover appeared to converge on local optima, hindering further progress.[7]

[7]For clarity of presentation, we omitted the first 400 generations in the figures. Thus we note that for the 100-bit hdf, uniform crossover outperformed the other three crossover operators by a small margin between generation 15 and generation 120. For the 200-bit hdf, uniform crossover still led after roughly 1000 generations.

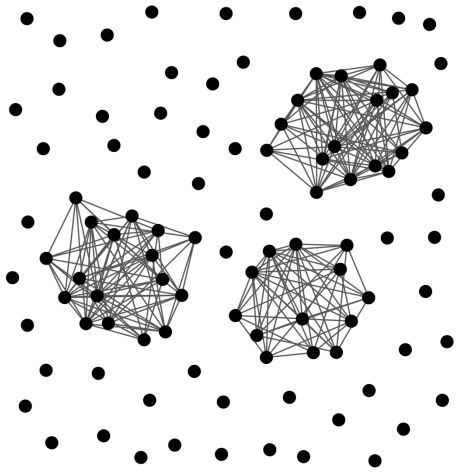


Figure 5: Crossover network for the shuffled 100-bit hdf. Isolated nodes represent genes not present in any elementary schema. Genes not present in any schema or pothole do not affect fitness.

5.4 Discussion of hdf results

For the easier (100-bit) hdf problem, the performance of both CrossNet variants (weighted and unweighted clique crossover networks) surpassed one-point crossover, which in turn surpassed uniform crossover. It seems clear in this case that the chromosomal network structures provide tighter linkage, and this facilitates the GA in solving the problem. For the harder (200-bit) hdf problem, the conclusions are less certain. In the final generations, the CrossNet variant using weighted-clique crossover gave slightly better results than non-weighted, uniform or one-point crossover variants. This suggests that for more difficult problems there may be a benefit to quantifying the level of epistatic interaction. Uniform crossover does well, particularly for fewer generations, but we hypothesize that its performance would be limited in further generations because the high rate of disruptiveness prevents it from maintaining large building blocks. The improvement in performance of uniform crossover is already decreasing at the end of these runs relative to the other variants.

It is natural to ask why the Crossover networks fared better on the easier hdf problem than on the harder one. One explanation is that the harder problem contains more levels of intermediate schemata and a greater quantity of potholes than the easier problem, and therefore the information about elementary schemata provided to the CrossNet-based operator is less valuable in the harder problem. Further experimentation with hdf's of varying levels of difficulty and with alternative network representations is necessary to substantiate this hypothesis.

One concern that we have not yet mentioned is that a crossover operator can provide linkage that is too tight. In this case, a schema with the correct alleles would be nearly indestructible, but this tight linkage would inhibit the GA from forming schemata initially. We doubt this is a problem with CrossNet, but this hypothesis could be tested by measuring the disruption rate for generated crossover masks.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed the CrossNet framework, and demonstrated how it could be used. In the DCP experiment, we provided empirical results that suggest it may be useful. In the shuffled hdf experiment, we showed that a CrossNet variant was able to outperform both uniform crossover and one-point crossover, by exploiting structural information about the problem. These two examples have provided a demonstration of the CrossNet framework, and are suggestive of further applications of network-based chromosomal representations.

Suppose there are many problems for which a CrossNet-based operator provides superior performance. What are the pragmatic drawbacks to using this mechanism?

1. You have to create an effective crossover network. This may take substantial thought and effort.

2. The resulting GA is slightly less efficient, because it introduces the mask-creation step into crossover.

Regarding the second point, the network splitting algorithm we provided for creating a mask could run in worst case $O(|V||E|)$ time. Whether this is significant depends mainly on the proportion of running time used for fitness evaluation. For the DCP, fitness evaluation completely dominated running time, and the time spent doing CrossNet crossover compared to the other crossovers was negligible. For the hdf problems, fitness evaluation was not very time consuming, and using CrossNet did substantially increase the overall GA running time. We should also mention that so far we have not attempted to optimize the CrossNet architecture. For instance, one might select masks from a cache of precomputed masks, rather than repeatedly running the network splitting algorithm.

A logical next step for future research is to examine additional applications to see whether CrossNet can substantially increase the performance of the GA in other domains, such as graph coloring or partitioning problems, which have a natural crossover network structure. The ultimate utility of the CrossNet framework likely depends on the ease of finding a suitable crossover network for a given problem. There are $2^{\frac{L(L-1)}{2}}$ possible crossover networks for a chromosome of length L. For most problems, this search space is so large that it will be impossible to find an optimal crossover network. However, it is not necessary to find an optimal network; it is sufficient to find a network that is better than any linear representation. The linear representation corresponds to one limited class of networks out of all of the possible networks, and thus may be greatly biasing the kinds of problems that the GA can easily solve. It is still an open question whether domain knowledge can routinely provide us with enough insight that we can construct superior chromosomal representations using the CrossNet framework.

Even if practitioners cannot easily construct such a network, perhaps computers can. In particular, it would be possible to evolve or co-evolve crossover networks for a problem. This technique would be tantamount to solving a linkage learning problem by reducing it to network evolution. This new approach to linkage learning is intriguing, but it gives the GA an additional problem to solve and may not be worth the effort.

The CrossNet framework may also be useful as a lens for studying existing crossover operators. As discussed in [3],

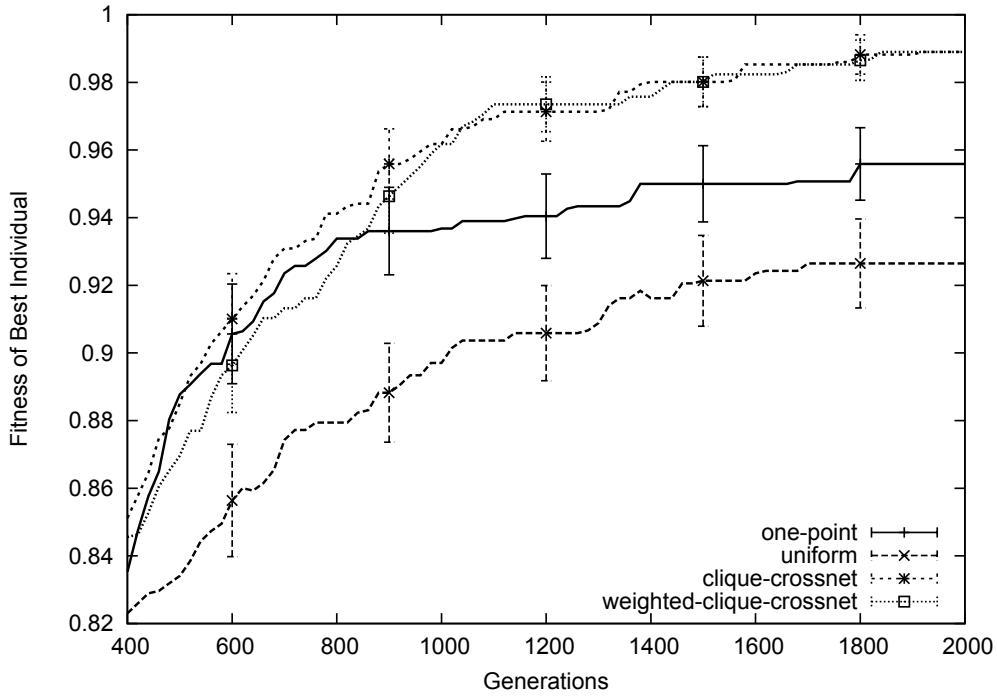


Figure 6: The 100-bit shuffled hdf problem. Data points are averaged across 180 runs, and standard error bars are shown. Fitness scores have been normalized to be between 0 and 1.

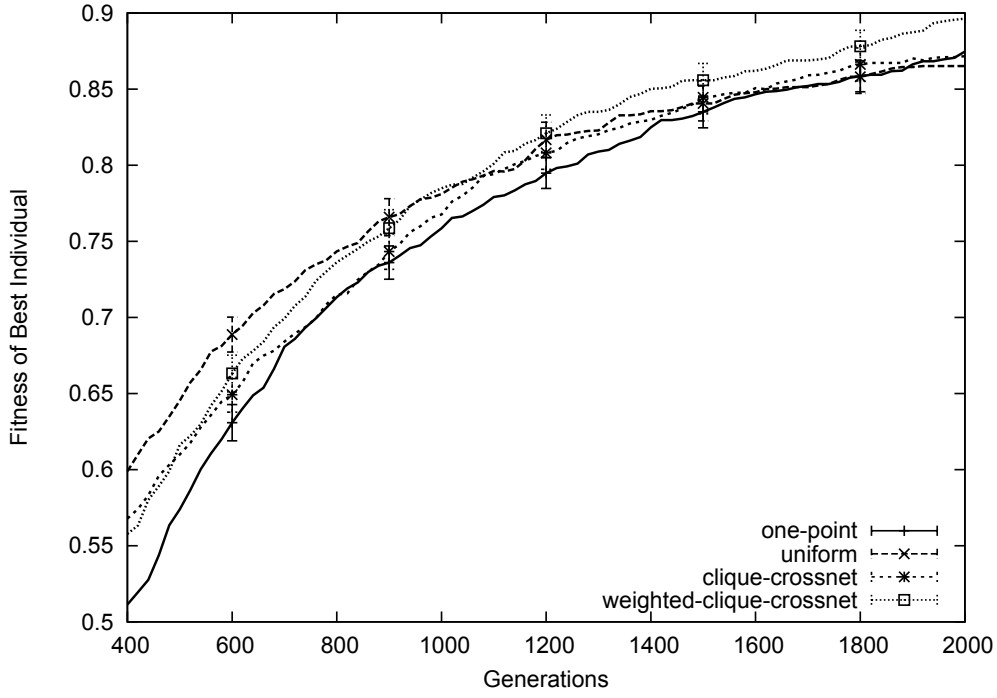


Figure 7: The 200-bit shuffled hdf problem. Data points are averaged across 180 runs, and standard error bars are shown. Fitness scores have been normalized to be between 0 and 1.

two-point crossover and uniform crossover are at opposite ends of a spectrum of disruptiveness. Two-point crossover is equivalent to CrossNet using a 1-D ring lattice crossover network, and uniform crossover is equivalent to CrossNet when using the (unweighted) complete graph. It might prove interesting to examine network structures that lie between these extremes – ring lattices with added connecting links (i.e. the "small-world" networks of [19]).

Lastly, we are also interested in the idea of hierarchical CrossNet, where the loci would be arranged in a network of networks. When evolving specifications for designing an airplane, the genes controlling wing design could be connected in one network, while the genes controlling tail design could be in another. These two networks would themselves be nodes in a larger crossover network for the design of the whole plane. Crossover could occur at multiple levels within the hierarchical network. This may facilitate the maintenance of modular structure within individuals in a GA.

The CrossNet framework provides a method by which we can easily integrate domain knowledge into the construction of crossover operators, and place many different operators within the same framework. By exploiting the representational power of networks to incorporate application-specific information into the genetic algorithm, we can facilitate the development of rich genomic representations, which have the potential to reduce the complexity of the problem space.

Acknowledgments: We thank the Northwestern Institute on Complex Systems for providing support for WR and Luis Amaral for supplying the computational resources. This work was also partially supported by NSF grant number 0713619. We also thank Sevan Ficici, Rick Riolo, and John Holland for providing feedback on some of these ideas.

7. REFERENCES

[1] Chen, Y.-P., Yu, T.-L., Sastry, K., and Goldberg, D. E. A survey of linkage learning techniques in genetic and evolutionary algorithms. Tech. Rep. 2007014, Illinois Genetic Algorithms Laboratory, Urbana, Illinois, 2007.

[2] DeJong, K., and Spears, W. An analysis of the interacting roles of population-size and crossover in genetic algorithms. In *Lecture Notes in Computer Science* (1991), vol. 496, p. 38.

[3] De Jong, K., and Spears, W. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence 5*, 1 (1992), 1–26.

[4] Falkenauer, E. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics 2*, 1 (1996), 5–30.

[5] Goldberg, D., and Lingle Jr, R. Alleles, loci, and the traveling salesman problem. *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications* (1985), 154–159.

[6] Greene, W. A. A genetic algorithm with self-distancing bits but no overt linkage. In *Proceedings of GECCO '02* (2002), pp. 367–374.

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

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

[9] Land, M., and Belew, R. No perfect two-state cellular automata for density classification exists. *Physical Review Letters 74*, 25 (1995), 5148–50.

[10] Mitchell, M., Crutchfield, J. P., and Hraber, P. T. Evolving cellular automata to perform computations: mechanisms and impediments. In *Proceedings of the Oji international seminar on complex systems : from complex dynamical systems to sciences of artificial reality* (New York, NY, USA, 1994), Elsevier North-Holland, Inc., pp. 361–391.

[11] Moon, B., Lee, Y., and Kim, C. GEORG: VLSI circuit partitioner with a new genetic algorithm framework. *Journal of Intelligent Manufacturing 9*, 5 (1998), 401–412.

[12] Pelikan, M., Goldberg, D., and Cantu-Paz, E. BOA: The Bayesian optimization algorithm. In *Proceedings of GECCO '99* (1999), vol. 1, pp. 525–532.

[13] Pereira, F. B., Machado, P., Costa, E., and Cardoso, A. Graph based crossover - a case study with the busy beaver problem. In *Proceedings of GECCO '99* (13-17 1999), vol. 2, pp. 1149–1155.

[14] Rand, W., and Riolo, R. Shaky ladders, hyperplane-defined functions and genetic algorithms: Systematic controlled observation in dynamic environments. In *Applications of Evolutionary Comp., Evoworkshops* (2005), F. Rothlauf et al., Eds., vol. 3449 of *Lecture Notes In Comp. Sci.*, Springer.

[15] Seo, D., and Moon, B. Voronoi quantized crossover for traveling salesman problem. In *Proceedings of GECCO '02* (2002), pp. 544–552.

[16] Seo, D., and Moon, B. A survey on chromosomal structures and operators for exploiting topological linkages of genes. In *Proceedings of GECCO '03* (2003), vol. 1, pp. 1357–1368.

[17] Stanley, K. O., and Miikkulainen, R. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of GECCO '02* (2002), pp. 569–577.

[18] Verel, S., Collard, P., Tomassini, M., and Vanneschi, L. Fitness landscape of the cellular automata majority problem: View from the "Olympus". *Theoretical Computer Science 378*, 1 (2007), 54–77.

[19] Watts, D., and Strogatz, S. Collective dynamics of 'small-world' networks. *Nature 393*, 6684 (1998), 409–10.

[20] Werfel, J., Mitchell, M., and Crutchfield, J. P. Resource sharing and coevolution in evolving cellular automata. *IEEE-EC 4*, 4 (Nov. 2000), 388.

[21] Wilensky, U. *NetLogo*. Center for Connected Learning and Computer-based Modeling, Northwestern University, Evanston, IL, 1999.