Chapter 13

Graph-based Evolutionary Algorithms

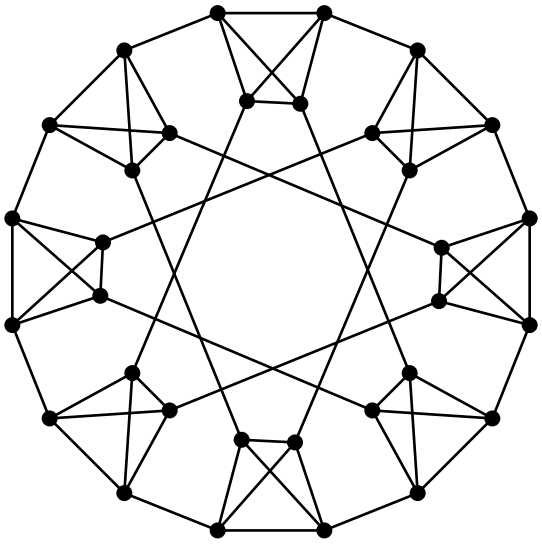


Figure 13.1: An example of a combinatorial graph

In this chapter, we will use combinatorial graphs to add geographic structure to the populations being evolved with evolutionary algorithms. Many experiments from previous chapters will be referred to and expanded. If you are unfamiliar with combinatorial graphs, you should read Appendix C. An example of a combinatorial graph is given in Figure 13.1. We will place individuals on the vertices of the graph and only permit replacement and mating to take place between connected vertices.

This is a generalization that gives a feature, present in biology, to evolutionary algorithms. Consider a natural population of rabbits. No matter how awesome the genetics

of a given rabbit, it can only breed with other rabbits nearby. Placing the structures of our evolving population into a geography and permitting breeding only with those nearby limits the spread of information in the form of superior genes. Single tournament selection is already available as a method of limiting the number of children of a high fitness individual, but, as we will see, using a graphical population structure gives us far more flexibility than varying tournament size does. Even tournament selection has a positive probability of any good gene breeding and replacing another.

You may wonder why we wish to limit the spread of good genes. The answer lies in our eternal quest to avoid local optima or premature convergence to a sub-optimal solution. Limiting the spread of a good structure without utterly preventing it permits the parts of the population "far" from the good structure to explore independently selected parts of the search space. Where a standard evolutionary algorithm losses population diversity fairly rapidly and ends up exploring a single sector of the fitness landscape of a problem quite soon, an evolutionary algorithm with an imposed geography can continue to explore different parts of the fitness space in different geographic regions.

Combinatorial graphs are described in Appendix C, and you should read the summary contained there, if you are not already familiar with them. Our primary questions in this chapter are:

1 *Does placing a combinatorial graph structure on a population ever change performance?*

2 *If so, what sorts of graph structures help which problems?*

3 *How do we document the degree to which a given graph structure helps?*

Throughout the chapter, you should think about the character of the example problems in relation to the diversity preservation (or other effects) induced by the use of a graph as a geographic population structure. Recall the broad classes of problems that exist: unimodal and multi-modal, optimization as opposed to co-evolution. The long term goal that underlies this chapter is to obtain a theory, or at least a sense, of how the fitness landscapes of problems interact with graphical population structures.

13.1 Basic Definitions and Tools

We will impose a graphical structure on an evolutionary algorithm by placing a single population member at each vertex of the graph and permitting reproduction and mating only between neighbors in the graph. (Note that this is not the only way one could use a graph to impose a graphical geography on an evolutionary algorithm.) Our model of evolution will need a way of selecting a gene from the population to be a parent, a way of selecting one of its neighbors to be a co-parent, and a way of placing the children.

Definition 13.1 *The* **local mating rule** *is the technique for picking a neighbor in the graph with which to undergo crossover and for placing children on the graph. It is the graph-based evolutionary algorithm's version of a model of evolution.*

There are a large number of possible models of evolution. Not only are there many possible local mating rules, but, also, there are many methods for picking the vertex that defines the local neighborhood. We will only explore a few of these models. Following Chapter 2, we will define the methods for picking the parent, locally picking the co-parent, and of placing children. A local mating rule will consist of matching up three such methods.

Definition 13.2 *The second parent, picked from among the neighbors of the first parent, is termed the* **co-parent***.*

The parent may be picked by roulette, rank, random, or systematic selection operating on the entire population. The first three of these methods have the exact same meaning as in Chapter 2. *Systematic* selection orders the vertices in the graph and then traverses them in order, applying a local mating rule at each vertex. Any selection method may have *deferred* or *immediate* replacement. Deferred replacement does not place any children until co-parents have been picked for each vertex in the graph, matings performed, and children generated. The children are held in a buffer until a population updating takes place. Deferred replacement yields a generational graph-based algorithm. *Immediate* replacement places children after each application of the local mating rule and is akin to a steady state version of the algorithm.

Co-parents may be selected (from the neighbors of the parent) by roulette, rank, random, or absolute fitness. The first three terms again have the exact same meaning as in Chapter 2. Absolute fitness replacement selects the best neighbor of the parent as the co-parent.

Replacement will involve one of: the parent, parent and co-parent, the neighbors of the parent including or not including the parent. *Absolute replacement* replaces both parent and co-parent with one of the children generated. *Absolute parental replacement* replaces only the parent with one of the children selected at random. *Elite parental replacement* replaces the parent with one of the children selected at random, *if* the child is at least as fit as the parent. *Elite replacement* places the best two of parent, co-parent, and children into the slots formerly occupied by the parent and co-parent. *Random neighborhood replacement* picks a vertex in the neighborhood of the parent (including the parent) and places one child selected at random there. *Elite neighborhood replacement* picks a vertex in the neighborhood of the parent (including the parent) and places one child selected at random there, if it is at least as good as the current occupant of the vertex. *Elite double neighborhood replacement* picks two neighbors at random and replaces each with a child selected at random, if the child is better.

Neutral Behavior of Graphs

Before we run evolutionary algorithms on graphs, we will develop some diagnostics of the behavior of a graph. These diagnostics will be approximations of biodiversity and of useful mating of the population on the graph. By useful mating we mean mating involving crossover with creatures of a type not encountered before. Crossover between similar creatures is close to wasted effort.

Definition 13.3 *If we have identifiable types* $\{1, \ldots, k\}$ *in a population of* N *creatures with* n_i *creatures of type* i*, then the* **entropy** *of the population is*

$$E = -\sum_{i=0}^{k} \frac{n_i}{N} \cdot Log_2\left(\frac{n_i}{N}\right).$$

If you have studied information theory, you will recognize the entropy defined above as the Shannon entropy of the "probability" of encountering a creature of a given type when sampling from the population. Entropy will be our surrogate for biodiversity. It has a number of properties that make it a good choice as a diversity measure. First of all, it increases as the number of types of creatures increases. This makes sense - more types, more biodiversity. The second good property is that, if the number of types are fixed, then entropy increases as the population is more evenly divided among the types. Imagine a cornfield with one foxtail and one dandelion. Now, imagine a field evenly divided between foxtails, dandelions, and mustard plants. Both fields have 3 types of plants in them but the second field is far more diverse. The third desirable property of entropy is that it is independent of the total number of creatures, and, so, permits comparison between populations of different sizes.

Definition 13.4 *The* **edge** *of a population on a graph is the fraction of edges with different types of creatures at their ends.*

Evolutionary algorithms generate new solutions to a problem in three ways. The initialization of the population is a source of new solutions though, if initialization is random, a crummy one. Mutation generates variations of existing solutions. Crossover blends solutions. When it works well, crossover is a source of large innovations. An unavoidable problem with standard evolutionary algorithms is that they lose diversity rapidly; soon, crossover is mostly between creatures of the same approximate "type." The result is that most crossover is wasted effort. The edge of a graph is the fraction of potential crossovers that could be innovative. Figure 13.2 shows the edge and entropy for 5 graphs over the course of 1,000,000 mating events in a neutral selection experiment.

Definition 13.5 *A* **neutral selection** *experiment for a graph* G *with* k *vertices is performed as follows. The vertices of the graph are labeled with the numbers* 1 *through* k *in some order.*

A large number of mating events are performed in which a vertex is chosen at random, and then the label on one of its neighbors, chosen at random, is copied over its own label. At fixed intervals, the labels are treated as a population and the entropy and edge are computed.

Since neutral selection experiments are stochastic, typically one must average over a large number of them to get a smooth result. We will explore this stochasticity in the first experiment of this chapter.

Experiment 13.1 *For the 9-hypercube, H_9, perform 5 neutral selection experiments with 1,000,000 mating events and a sampling interval of 1000 mating events. The graph H_9 is described in Appendix C. Graph the edge and entropy for each of the 1000 samples taken and for each of the 5 experiments separately. Report the number of the sampling event on which the entropy drops to zero (one label remaining), if it does. Compare your plots with the average, over 100 experiments, given in Figure 13.2. Comment on the degree to which the plots vary in your write up and compare in class with the results of other students.*

Looking at the tracks for entropy and edge from Experiment 13.1, we see that there is a good deal of variability in the behavior of the evolution of individual populations on a graph. Looking at Figure 13.2, we also see that there is substantial variation in the behavior using different graphs.

Definition 13.6 *An* **invariant** *of a graph is a feature of the graph that does not change when the way the graph is presented changes, without changing the fundamental nature of the graph.*

Name	degree	diameter	edges
C_{512}	2	256	512
$P_{256,1}$	3	129	768
$T_{128,4}$	4	66	1024
H_9	9	9	2304
K_{512}	511	1	130,816

Table 13.1: Some graph invariants for the graphs whose neutral selection results appear in Figure 13.2

Experiment 13.2 *Write or obtain software that can gather and graph data as in Figure 13.2. The graphs ($K_{512}, H_9, T_{4,128}, P_{256,1}$, and C_{512}) used in this experiment are described in Appendix C. Perform a neutral selection experiment using 1,000,000 mating events with the*

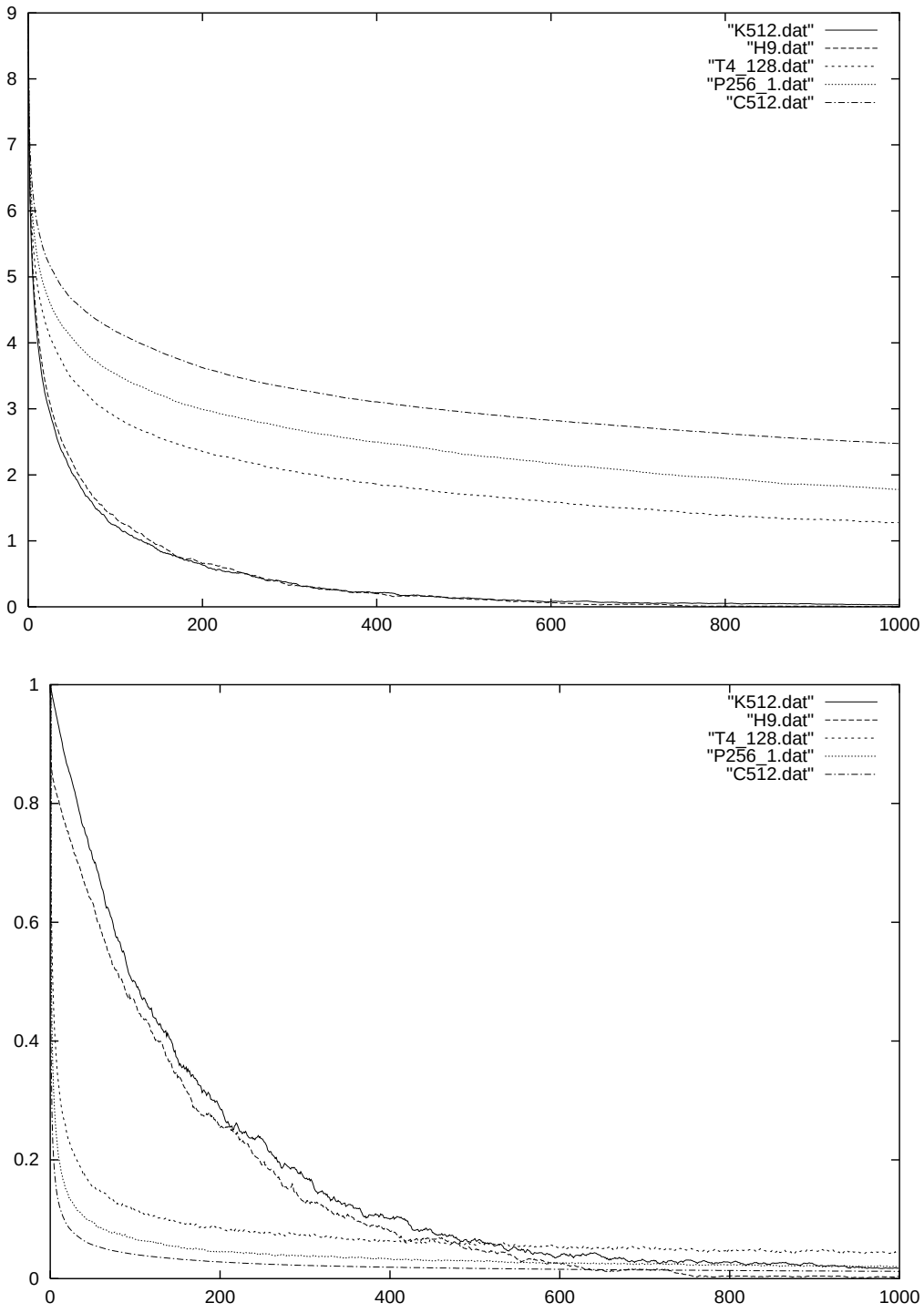


Figure 13.2: Graphs showing the entropy and edge in a neutral selection experiment for the complete graph K_{512}, the 9-dimensional hypercube H_9, the 4×128 torus $T_{4,128}$, the generalized Petersen graph $P_{256,1}$, and the 512-cycle C_{512} (The plots are averaged over 100 experiments for each graph. Each experiment performs 1,000,000 mating events, sampling the entropy and edge each 1000 mating events.)

edge and entropy values sampled every 1000 mating events and averaged over 100 replications of the experiment. Test the software by reproducing Figure 13.2.

Now, rerun the software on the graphs $P_{256,1}$, $P_{256,3}$, $P_{256,7}$, and $P_{256,15}$. Graph the entropy and edge for these 4 Petersen graphs together with C_{512} and K_{512}. This experiment checks to see if edge and entropy vary when degree is held constant (3) and also compares them to graphs with extreme behaviors. In your write up, comment on the degree to which the edge and entropy vary and on their dependence on the degree of the graph.

Experiment 13.2 leaves the number of edges the same while changing their connectivity to provide a wide range of diameters (excluding C_{512} and K_{512} which serve as controls). The next experiment will generate graphs with the same degree and a relatively small range of diameters. This will permit us to check if there is variability in neutral selection behavior that arises from sources other than diameter.

Experiment 13.3 *Write or obtain software that can generate random regular graphs of the sort described in Definition C.22. Starting with the Petersen graph $P_{256,1}$ and making 3500 edge moves for each instance, generate at least 4 (consult your instructor) random 3-regular graphs. Repeat Experiment 13.2 for these graphs, including the C_{512} and K_{512} controls. In addition, compute the diameter of these graphs and compare diameters across all the experiments performed.*

For the remainder of the chapter, you should keep in mind what you have learned about neutral selection. Try to answer the question: what, if any, value does it have for predicting the behavior of graph-based evolutionary algorithms?

Problems

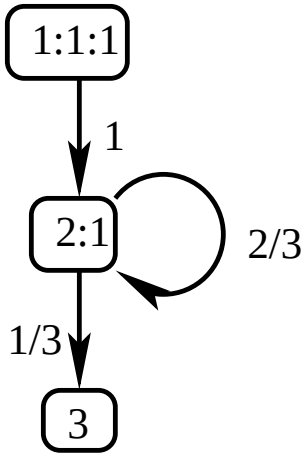
Problem 13.1 *List all the local mating rules that can be constructed from the parts given in this section. Put a check mark by any that don't make sense and give a one sentence explanation of why they don't make sense.*

Problem 13.2 *For a single population in a neutral selection experiment are entropy and edge, as a function of time measured in mating events, monotone decreasing? Prove your answer.*

Problem 13.3 *Examine the neutral selection data presented in Figure 13.2. For both the entropy and edge tracks, decide which of the invariants in Table 13.1 is most predictive of the entropy behavior and the edge behavior.*

Problem 13.4 Essay. *Describe the interaction between edge and entropy. If edge is high, is entropy decreasing faster on average than if edge is low? At what level of entropy, on average, is it likely for edge to go up temporarily? Can you say anything else?*

Problem 13.5 *Find a sequence of mate choices for a neutral selection experiment on K_5 for which the entropy stays above 1.5 forever. Do you think this will ever happen? Why?*



Problem 13.6 *Consider a neutral selection experiment on K_3. It starts with distinct labels on all 3 vertices. After the first mating event, there are 2 labels that are the same and 1 that is different. The next mating event has $\frac{1}{3}$ chance of making all the labels the same and $\frac{2}{3}$ chance of leaving two labels the same and one different. Once all the labels are the same, the system is stuck there. This behavior, encompassing all possible histories, can be summarized in the Markov chain diagram shown above. First convince yourself that the probabilities given are correct for K_3. Draw the equivalent diagram for K_4, using the states [1:1:1:1], [1:1:2], [1:3], [2:2], and [4].*

Problem 13.7 *Suppose we are performing neutral selection on C_n, the n-cycle. Describe all possible collections of labels that can occur during the course of the experiment.*

Problem 13.8 *Suppose that n is even and that we are running a neutral selection experiment. For K_n, $P_{\frac{n}{2},2}$, and C_n, consider the sets of all vertices that have the same label. Do these collections of vertices have to be connected in the graph? If the answer isn't obvious, you could run simulations. The answer should be quite obvious for K_n.*

Problem 13.9 *Extend Table 13.1 by copying the current entries and adding the information for the Petersen graphs used in Experiment 13.2.*

Problem 13.10 *Two graphs are* isomorphic*, if you can exactly match up their vertices in a fashion that happens to also exactly match up their edges. Notice that, unless two graphs have the same number of vertices and edges, they cannot be isomorphic. For the graphs $P_{16,n}$ with $1 \leq n \leq 15$, find out how many "different" graphs there are. (Two graphs are "the same," if they are isomorphic.)*

If you are unfamiliar with graph theory, this is a very difficult problem; so, here are four hints. First, consider equivalence of numbers (*mod* 16). *If you go around the inner circle of the Petersen graph by jumps of size one to the left or right, you still draw the same edges, and, so, pick up some obvious isomorphisms with which to start. Second, remember that isomorphism is transitive. Third, isomorphism preserves all substructures. If we have a close cycle of length 4 in one graph and fail to have one in another, then they cannot be isomorphic. Fourth, notice that isomorphism preserves diameter.*

Problem 13.11 *Compute the diameters of* $P_{256,k}$ *for* $1 \leq k \leq 255$. *What is the maximum, the minimum, and how do these numbers reflect on the choice of the 4 Petersen graphs in Experiment 13.2?*

Problem 13.12 *Generate 1000 graphs of the sort used in Experiment 13.3. For the diameters of these graphs, make a histogram of the distribution, compute the mean, standard deviation, minimum value, and maximum value. Finally, compare these with the results of Problem 13.11. Are the Petersen graphs representative of cubic graphs?*

Problem 13.13 *Suppose that we adopt an extremely simple notion of connectivity: the fraction of edges a graph has relative to the complete graph. Thus,* C_5 *has a connectivity of* 0.5, *while* K_5 *has a connectivity of* 1.0. *Answer the following questions.*

(i) What is the connectivity, to 3 significant figures, of the 5 graphs used to generate Figure 13.2?

(ii) Approximately, how many graphs are there with connectivity intermediate between H_9 *and* K_{512}*?*

(iii) In what part of the range of connectivities from 0 to 1 is most of the variation in neutral selection behavior concentrated?

Problem 13.14 *Suppose you have a random graph with edge probability* $\alpha = 0.5$ *(for all possible edges, flip a fair coin to see if the edge is present). Compute, as a function of the number* n *of vertices in the graph, the probability it will have diameter 1.*

Problem 13.15 Essay. *Modify the code you used in Problem 13.12 to also report if the graph is connected. What fraction of graphs were connected? Now, perform the experiment again starting with* C_{512} *instead of* $P_{256,1}$ *and find what fraction of the graphs are connected. Explain the results.*

13.2 Simple Representations

In this section, we will examine the effects of imposing a geography using graphs on the simplest of evolutionary algorithms, those with data structures of fixed-sized strings or vectors of real numbers. The initial work will be a comparison of the 1-max Problem and one of its variations, called the k-max Problem.

Definition 13.7 *The k-max fitness function $F_{k\text{-}max}$ maps strings of length l over a k-member alphabet to the count of their most common character. Thus, $F_{k\text{-}max}(0110100100110) = 7$, while $F_{k\text{-}max}(ABBCCCDDDDEEE) = 4$.*

The advantage of the k-max Problem is that it is constructively polymodal. This permits us to compare the 1-max Problem, a completely unimodal problem, with a number of constructively polynomial problems with a very similar character. Let us start by doing an experiment that baselines the behavior of graph-based evolutionary algorithms on the 1-max Problem.

Experiment 13.4 *For the 5 graphs K_{512}, H_9, $T_{4,128}$, $P_{256,3}$, and C_{512}, using random selection of the parent, roulette selection of the co-parent, and immediate elite replacement of the parent by the better child, run a graph-based evolutionary algorithm on the 1-max Problem over the binary alphabet with length 16. Use two point crossover and single point mutation. In addition, run a baseline evolutionary algorithm on the 1-max Problem using single tournament selection with size 4.*

For each of the 6 evolutionary algorithms, save time-to-solution (cutting off algorithms at 1,000,000 mating events) for 100 replications of each algorithm. Give a 95% confidence interval for the mean time-to-solution for each algorithm. Discuss which graphs are superior or inferior for the 1-max Problem and compare with the single tournament selection baseline.

The 1-max Problem has two distinct evolutionary phases. In the first, crossover mixes and matches blocks of 1s and can help quite a lot. In the second, a superior genotype has taken over and we have 0s in some positions throughout the population, forcing progress to rely on mutation. In this latter, longer phase, the best thing you can do is to copy the current best structure as fast as possible. The dynamics of this second phase of the 1-max Problem suggest that more connected graphs should be superior.

Experiment 13.5 *Repeat Experiment 13.4 for the 2-max Problem and for the 4-max Problem. The fitness function changes to report the largest number of any one type of character, first over the binary alphabet then over the quaternary alphabet. Compare results for 1-max, 2-max, and 4-max.*

The Royal Road function (see Section 2.5) is the standard "hard" string evolver problem. Let's use it to explore the effects of varying the mutation rate (which we already know is important) with the effects of changing the graph used.

Experiment 13.6 *Repeat Experiment 13.4 for the classic Royal Road problem ($l = 64$ and $b = 8$) or for $l = 36$, $b = 6$, if the run time is too long on the classic problem. Do 3 groups of 6 collections of 100 runs, using each of the mutation operators: one point mutation, probabilistic mutation with rate one, and probabilistic mutation with rate two. Compare the effects of changing graphs with the effects of changing mutation operators.*

There is no problem with running Experiments 13.4, 13.5, and 13.6 as steady state algorithms. If we have a Tartarus type problem in which we are sampling from among many available fitness cases, then we require a generational algorithm. The following experiment thus uses deferred replacement.

Experiment 13.7 *Create a generational graph-based evolutionary algorithm (one using deferred replacement) to evolve string controllers for the Tartarus problem. We will use variable-length strings and the gene doubling and gene halving operators described in Section 10.1. The initial population and variation operators are as in Experiment 10.5. Test fitness on 100 random 6×6 Tartarus boards with 6 boxes. Use the same graphs that were used in Experiment 13.4. The algorithm will visit each vertex systematically as a parent. Select the co-parent by roulette selection and use absolute replacement.*

Baseline the experiment with an evolutionary algorithm using size 4 tournament selection. Perform 100 runs of length 200 generations and compare average and best results of all 6 sets of runs. If possible, apply knowledge about what strings are good from Chapter 10 to perform a qualitative assessment of the algorithm.

The main purpose of Experiment 13.7 is to give you an example that needs to be a generational rather than a steady state algorithm. Another issue that is worth examining is that of population size.

Experiment 13.8 *Redo Experiment 13.6 with the following list of graphs: $P_{2^n,3}$, $n = 4, 6, 8,$ and 10. Run a steady state algorithm and measure time-to-solution in mating events. Use the data you already have from $n = 8$ to establish a time after which it would be reasonable to give up. Compare across population sizes and, if time permits, fill in more points to document a trend.*

We now turn to the problem of real function optimization with graph-based algorithms, beginning with the numerical equivalent of the 1-max Problem, the fake bell curve.

Experiment 13.9 *For the 5 graphs K_{512}, H_9, $T_{4,128}$, $P_{256,3}$, and C_{512}, using roulette selection of the parent, rank selection of the co-parent, and elite replacement of the parent by the better child, run a graph-based evolutionary algorithm to maximize the function:*

$$f(x_1, x_2, \ldots, x_8) = \frac{1}{1 + \sum_{i=1}^{8}(x - i)^2}.$$

Use one point crossover and Gaussian mutation with standard deviation $\sigma = 0.1$. Place the initial population in a hypercube from $(0, 0, 0, 0, 0, 0, 0, 0)$ to $(9, 9, 9, 9, 9, 9, 9, 9)$ with points placed uniformly at random. The function given is a shifted version of the fake bell curve in 8 dimensions.

Run a baseline evolutionary algorithm using single tournament selection with tournament size 4, as well. For each of the 6 evolutionary algorithms, save time-to-solution (cutting off algorithms at 1,000,000 mating events) for 100 replications of each algorithm. Take a functional value of 0.999 or more to be a correct solution. Give a 95% confidence interval for the mean time-to-solution for each algorithm. Discuss which graphs are superior or inferior and compare with the single tournament selection baseline.

Even in 8 dimensions, random initialization of 512 structures will yield some fairly good solutions in the initial population. It would be nice to document if mutational diversity can build up good solutions where none existed before and then later recombine good pieces from different parts of a diverse population. We will perform an experiment in this direction by starting with a uniformly awful population.

Experiment 13.10 *Perform Experiment 13.9 again, but this time initializing all creatures to the point $(0, 0, 0, 0, 0, 0, 0, 0)$. Compare with the previous experiment. Did the ordering of the graphs's performances change?*

$$f(x, y) = \frac{3.2}{1 + (40x - 44)^2 + (40y - 44)^2} + \frac{3.0}{1 + (3x - 5.4)^4 + (3y - 5.4)^4} \tag{13.1}$$

For the next experiment, we will use Function 13.1, constructed to have two optima, one very near $(1.1, 1.1)$ and the other very near $(1.8, 1.8)$. The former is the global optimum, while the latter is broader and, hence, much easier to find.

Experiment 13.11 *Using the graphs K_{32}, H_5, $T_{4,8}$, $P_{16,3}$, and C_{32}, write or obtain software for a graph-based evolutionary algorithm to maximize Equation 13.1. Use roulette selection of the parent and rank selection of the co-parent, absolutely replacing the parent with the first child. Use mixing of x and y coordinates as the crossover operator and use crossover 50% of the time. Use single point mutation with a Gaussian mutation with variance $\sigma = 0.1$ 100%*

of the time. For each graph, run a steady state algorithm until a population member first comes within $d = 0.001$ *of either* $(1.1, 1.1)$ *or* $(1.8, 1.8)$*. Discuss which graphs are better at finding the true optimum at* $(1.1, 1.1)$*. If this experiment is done by multiple people, compare or pool the results for random graphs.*

Experiment 13.12 *Repeat Experiment 13.11, but change the local mating rule to (i) random selection of the parent and rank selection of the co-parent, and then to (ii) systematic selection of the parent and rank selection of the co-parent. For the systematic selection, simply take the vertices in order. Compare the local mating rules and document the impact (or lack of impact).*

Experiment 13.11 tests the relative ability of graphs to enable evolutionary search for optima. Function 13.1, graphed in Figure 13.3, has two optima. The local optimum is broad, flat, and has a large area about it. The global optimum is much sharper and smaller. Let's move on to a complex and very highly polymodal fitness function - the self avoiding walks from Section 2.6.

Experiment 13.13 *Modify the graph-based string evolver to use the coverage fitness function for walks on a* 5×5 *grid, given in Definition 2.16. Use two point crossover and two point mutation. Compute the number of failures to find an answer in less than 250,000 mating events for each of the following graphs:* K_{256}*,* H_8*,* $T_{4,64}$*,* $T_{8,32}$*,* $T_{16,16}$*,* $P_{128,1}$*,* $P_{128,3}$*,* $P_{128,5}$*, and* C_{256}*. Give a 95% confidence interval on the probability of failure for each of the graphs. Also run a size 4 tournament selection algorithm without a graph, as a baseline. Are there significant differences?*

The k-max Problem has several optima with large basins of attraction and no local optima. The coverage fitness for walks has thousands of global optima and tens of thousands of local optima. The basins of attraction are quite small. The value of diversity preservation should be much greater for the coverage fitness function than for the k-max fitness function. Be sure to address this issue when writing up your experiments.

Problems

Problem 13.16 *Clearly, the optima of the k-max Problem are the strings with all characters the same, and, so, the problem has k optima. Suppose, for each optimum, we define the* basin of attraction *for that optimum to be strings that go to that optimum under repeated application of helpful mutation (mutating a character not contributing to fitness to one that does). For* $k = 2, 5$ *and* $l = 12, 13$*, compute the size of the basin of attraction for each optimum (they are all the same) and the number of strings that are not in any basin of attraction.*

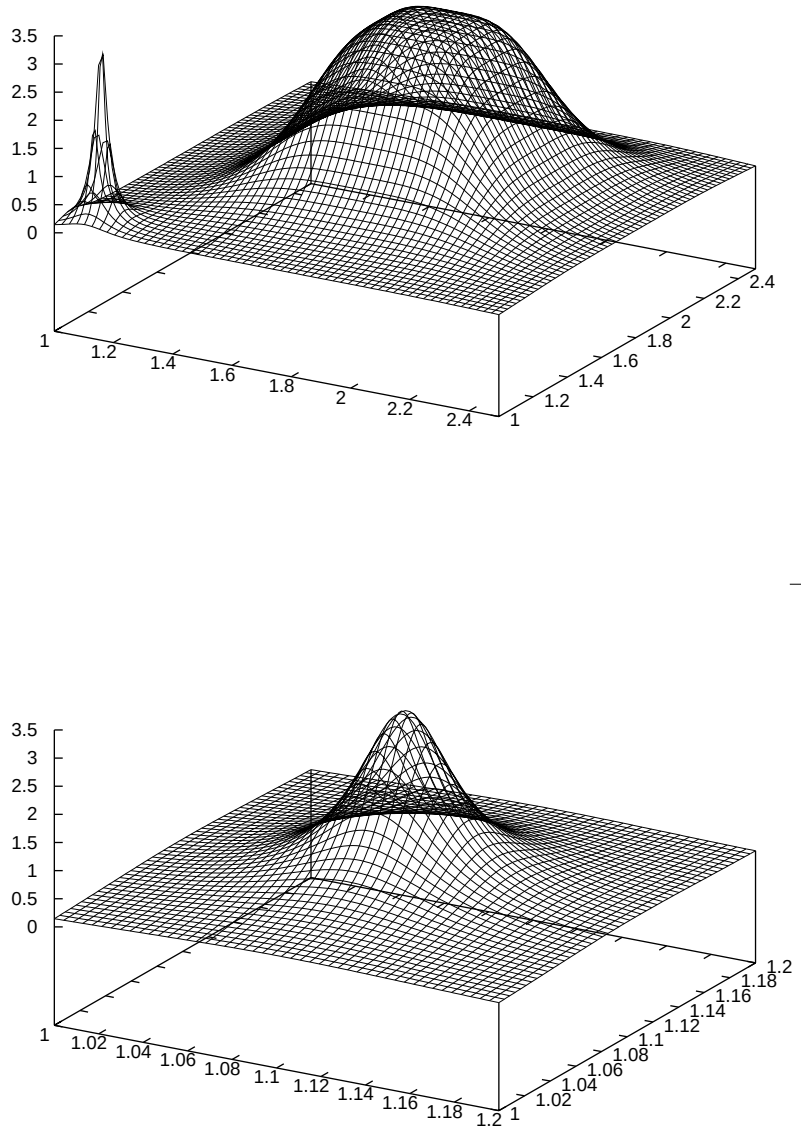


Figure 13.3: Function 13.1 showing both optima and a closeup of the true optimum

Problem 13.17 *The optima of the k-max Problem are all global optima; they all have the same fitness value, and it is the highest possible. Come up with a simple modification of the k-max Problem that makes the optima have different heights.*

Problem 13.18 *Explain why running a graph-based evolutionary algorithm for Tartarus with evaluation on 100 randomly chosen boards should not be done as a steady state algorithm.*

Problem 13.19 *Suppose that you have a problem for which you know that repeated mutation of a single structure is not as good as running an evolutionary algorithm with some population size $n > 1$. If you are running a steady state algorithm and measuring time-to-solution (or some other goal) in mating events then prove there is a population size such that increasing the population beyond that size is not valuable.*

Problem 13.20 *In Experiment 13.11, the graphs use a much smaller population than in other experiments in the section. Why? Explain in a few sentences.*

Problem 13.21 *For Function 13.1, perform the following computations. Place a 200×200 grid on the square area with corners at (0,0) and (3,3). For each grid cell start at a point in the center of the grid cell. Use a gradient follower with a small step size to go uphill until you reach a point near one of the two optima. This means that you repeatedly compute the gradient and then move a small distance in that direction, e.g. 0.003 (1/1000th the side length of the search grid). The grids that move to a given optimum are in its* gradient basin of attraction*. What is the relative size of the gradient basins of attraction of the two optima? For a discussion of the gradient, see Appendix B.*

Problem 13.22 *Explain how to generalize Function 13.1 to n dimensions.*

Problem 13.23 Essay. *Would you expect the population diversity of a graph-based evolutionary algorithm to be greater or smaller than the population diversity of a standard evolutionary algorithm.*

Problem 13.24 Short Essay. *A thought that occurs quite naturally to readers of this book is to evolve graphs based on their ability to help solve a problem. Discuss this idea with attention to (i) how to represent graphs and (ii) time complexity of the fitness evaluation for graphs.*

Problem 13.25 Short Essay. *Reread Experiments 13.9 and 13.10. Now, suppose you are attacking a problem you do not understand with graph-based algorithms. One danger is that you will place your initial population in a very small portion of the search space. Does Experiment 13.10 give us a tool for estimating the cost of such misplacement? Answer the question for both string and real number representations (remember the reach of Gaussian mutation).*

Problem 13.26 Essay. *Assume the setup used in Experiment 13.6, but with far more graphs. Suppose that you order the graphs according to mean time-to-solution. Changing the mutation operator changes mean time-to-solution: must it preserve the order of the graphs? In your essay, try to use the notion of fitness landscape and the interaction of that fitness landscape with the gene flow on the graph.*

Problem 13.27 Essay. *In this section, we have been testing the effect of changing graphs on the behavior of a graph-based evolutionary algorithm. Can graphs themselves be used as probes for that type of problem? If not, why? If so, how?*

Problem 13.28 *For the experiments you have performed in this section, order the graphs within each experiment by the rule: $G > H$, if the performance of G on the problem is significantly better than the performance of H. Give these partial orders; if you have performed more than one experiment, comment on the differences between the orders.*

13.3 More Complex Representations

Simple representations with linear chromosomes, such as those used in Section 13.2, have different evolutionary dynamics from more complex representations, like those used in Chapters 6-10 and Chapter 12. In this section, we will examine the behavior of some of these systems in the context of GBEAs (graph-based evolutionary algorithms).

In the remainder of this chapter, we will try to control for degree versus topology in the graphs we use by using random regular graphs. We've used these graphs before in Experiment 13.3. They are described in Appendix C, but we will briefly describe them again here. The technique for producing random regular graphs is not a difficult one, and it generalizes to many other classed of random object generation problems.

The task is to generate a random member of a class of objects. The technique is to find a random transformation that makes a small modification in the object(so that the modified object is still in the class), very like a mutation. As with mutations, we need the change operation to have the property that any object can be turned into any other eventually. We start with any member of the class and make a very large number of modifications, in effect randomly walking it through the object configuration space. This results in a "random" object. Since we want to generate random regular graphs with the same degree as the ones in our other experiments, we use the graphs from the other experiments as starting points.

The change operation used to generate random regular graphs is the edge swap, illustrated in Figure 13.4 and performed in the following manner. Two edges of the graph are located that have the property that they are the only two edges with both ends in the set of 4 vertices that comprise their ends. Those two edges are deleted, and two other edges between the 4 vertices are added. This transformation preserves the degree of the graph while modifying its connectivity.

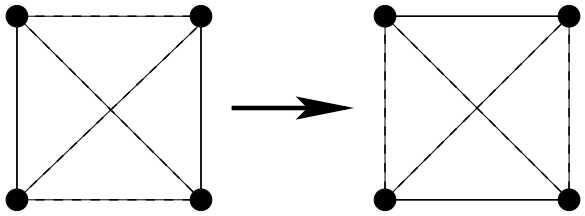


Figure 13.4: The edge swap operation (Solid lines denote present edges; dotted lines denote absent edges.)

One point should be made about using random regular graphs. There are an incredible number of different random regular graphs for each degree and number of vertices, as long as there are at least a few tens of vertices. This means that the random regular graph generation procedure is sampling from some distribution on a space of graphs. So what? So, it's important to generate the random regular graphs *before* performing multiple experimental runs and to remember that you are performing experiments on *instances* of a family of graphs. When pooling class results, you may notice large variations in behavior based on which random graphs were used. There are some really good and some really bad random regular graphs out there.

Experiment 13.14 *Review Experiment 6.4, in which we used a lexical partner to enhance performance of a finite state automaton on a string prediction task. Redo this experiment as a baseline, and then, using the same fitness function and representation for finite state automata, perform the experiment as a graph-based algorithm. Use the following graphs:* K_{120}, $P_{60,1}$, $P_{60,7}$, $T_{8,15}$, C_{120}, *and 3 instances of random regular graphs derived from* $P_{60,1}$ *and* $T_{8,15}$ *using twice as many edge swaps as there are edges in a given graph. Use random selection for the parent and roulette selection for the co-parent, and use automatic, immediate replacement of the parent with the better of the two children produced. Comment on the impact of the graphs used. Is there much variation between graphs of the same degree? Does the use of graphs increase or decrease the impact of the lexical partner function?*

Experiment 13.14 covers a lot of territory. The graphs may enhance the performance of the lexical partner function or retard it; this effect may not be uniform across graphs. The choice of graph in a GBEA is a very complex sort of "knob" - more complex than the mutation rate or population size - but still a parameter of the algorithm that can be tuned. More complex issues, such as the impact of graphs on co-evolution, we defer to the future.

Definition 13.8 *A* **partial order** *is a binary relation* $\leq$ *on a set* S *with the following 3 properties:*

(i) for all $a \in S$, $a \leq a$,

(ii) for all $a, b \in S$, $a \leq b$ *and* $b \leq a$ *implies* $a = b$, *and*

(iii) for all $a, b, c \in S$, $a \leq b$ *and* $b \leq c$ *implies* $a \leq c$.

These 3 properties are called the **reflexive**, **anti-symmetric**, *and* **transitive** *properties, respectively. Divisibility is a partial ordering of the positive integers.*

Definition 13.9 *The* **performance partial order** *of a set of graphs on a problem for a given GBEA is a partial ordering of graphs in which* $G \leq H$, *if the time-to-solution using* G *is significantly less than* H. *In this case, "significantly" implies a statistical test such as disjoint confidence intervals for time-to-solution.*

In theory, crossover is putting pieces together, while mutation is tweaking existing pieces and, at some rate, generating new pieces. When evolving ordered structures (Chapter 7), the nature of the pieces is less clear than it is in a problem with a simple linear gene. Let's check the impact of GBEAs on a couple of ordered gene problems.

Experiment 13.15 *Modify the software used in Experiment 7.7 to run as a GBEA and also compare the standard and random key encodings. Use the same graphs and graph algorithm settings as in Experiment 13.14. For each representation, give the performance partial order for 95% confidence intervals on time-to-solution. What is the impact of the graphs on maximizing the order of permutations?*

And now on to the Traveling Salesman problem. This problem has the potential for segments of a given tour to be "building blocks" that are mixed and matched. This, in turn, creates room for graphs to usefully restrict information flow as good segments are located.

Experiment 13.16 *Redo Experiment 7.9 as a GBEA; use only the random key encoding. Use the same graphs and graph algorithm settings as in Experiment 13.14. If your instructor thinks it's a good idea, run more cases of the Traveling Salesman problem from those given in Chapter 7. Give the performance partial order on 95% confidence intervals for time-to-solution. What is the impact of the graphs on the given examples of the Traveling Salesman problem?*

Since graphs restrict information flow, population seeding may interact with GBEAs to yield novel behavior. Review Algorithms 7.1 and 7.2 in Chapter 7.

Experiment 13.17 *Redo Experiment 13.16 but with population seeding. Do 3 sets of runs. In the first, put a tour generated with Algorithm 7.1 on a vertex 5% of the time. In the second, put a tour generated with Algorithm 7.2 on a vertex 5% of the time. In the third, use both heuristics, each on 5% of the vertices. Give the performance partial order on 95% confidence intervals for time-to-solution. What is the impact of the 3 population seeding methods?*

In Chapter 11, we studied a number of representations for evolving logic gates. Let's check the impact of adding graphs to the experiments using a couple of these representations (the direct representation and connection lists). First, let's add graphs to the experiment using a direct representation.

Experiment 13.18 *Review Experiment 11.5. For the 3-input parity problem only, redo the experiment as a GBEA, using the graphs and graph algorithm parameters from Experiment 13.14. Give the performance partial order for 95% confidence intervals on time-to-solution. What impact does the use of graphs have on this logic gate evolution problem?*

Now, let's add graphs to the experiment using a connection list representation. Since this is a substantially different representation it may behave differently.

Experiment 13.19 *Review Experiment 11.7. For the 3-input parity problem only, redo the experiment as a GBEA, using the graphs and graph algorithm parameters from Experiment 13.14. What impact does the use of graphs have on this logic gate evolution problem? Compare the results with those from Experiment 13.18.*

This section draws on material from many previous chapters. Prior to this, we have studied interactions of the underlying problem we are trying to solve with the choice of variation operators. In Chapter 3, we argued that these variation operators create the connectivity between values of the independent variable, while the fitness function computes the dependent variable - the fitness of a point in the search space. With GBEAs we add another element of complexity to the system: interaction with the graph-based geography. That geography controls the spread of information both by the nominal connectivity of the graph and by the choice of local mating rule.

Given the number of design features available (representation, fitness function(s), choice of variation operators, rate of application of those operators, model of evolution, and now choice of graph) a coherent, predictive theory of behavior for a GBEA system seems distant. Until someone has a clever idea, we must be guided by rules of thumbs and previous experience with similar experiments. Thus far in this chapter we have simply reprised various experiments from previous chapters to assess the impact of using graphs. We have not yet explored the effect of the local mating rule.

Experiment 13.20 *Pick one or more experiments in this section that you have already performed. Modify the local mating rule to use elite rather than absolute replacement and perform the experiment(s) again. What is the impact? Does it change the relative impact of the graphs?*

Experiment 13.20 tests what happens when a GBEA changes its local mating rule. Experiment 13.8 tested what happened when we changed population size while leaving the graph as close to the same as possible.

Experiment 13.21 *Pick one or more experiments in this section that you have already performed. Perform it again using the graphs $T_{n,m}$ for the following values of n and m: 5,24; 6,20; 8,15; 10,24; 12,20; 16,15; 10,48; 12,40; and 16,30. What has more impact: shape or population size? Time should be measured in mating events, so as to fairly compare the amount of effort expended.*

Chapter 5 demonstrated that we could obtain fairly complex behavior from very simple structures (symbots). Review Experiment 5.8 in which we tested the effect of various types of walls on symbots trying to capture multiple sources. In the next experiment, we will test the impact of different graphs on the ability to adapt to those walls.

Experiment 13.22 *Rebuild Experiment 5.8 as a GBEA using random selection of the parent and roulette selection of the co-parent with absolute replacement of the parent by a randomly selected child. Test the original algorithm against the graph-based algorithm with the graphs C_{256}, $T_{16,16}$ and H_8. Are different graphs better for different types of walls?*

So far we have not experimented with competing populations on a graph. Let's draw on the Sunburn model from Chapter 4 for a foray in this direction.

Experiment 13.23 *Implement or obtain software for the following version of the Sunburn evolutionary simulator. Use a graph topology to control choice of opponent. Choose a first ship at random and one of its neighbors at random. Permit these two ship designs to fight. If there is no victor, repeat the random selection until a victor arises. Now, pick a neighbor of the losing ship and one of its neighbors. Permit these ships to fight. If there is not a victory, then pick a neighbor of the first ship picked in the ambigiguous combat and one of its neighbors and try again until a victory is achieved. The victors breed to replace the losers, as before. Notice that the graph is controlling choice of opponent and, to a lesser degree, choice of mating partner.*

Perform 100 standard Sunburn runs and 100 graph-based Sunburn runs for the graphs C_{256}, $T_{16,16}$ and H_8. Randomly sampling from final populations, compare opponents drawn from all 6 possible pairs of simulations. Is there any competitive edge created by constraining the topology of evolution with graphs?

The material covered in this section gives a few hints about the richness of interactions between graphs and evolutionary computation. Students looking for final projects will find a rich field here. In Experiment 13.17, we introduced yet another parameter for population seeding, the rate for each heuristic used. Further exploration of that is not a bad idea. Experiment 13.23 opens a very small crack in the door to a huge number of experiments on the impact of graphs on competing populations.

Problems

Problem 13.29 *The graphs used in the experiments thus far have had degree 2,3,4, $log_2(n)$, and $n-1$, where n is the population size. Give constructions for an infinite family of graphs of degree 5, 6, and 7.*

Problem 13.30 *Suppose that we have a graph on n vertices created by flipping a coin for each pair of vertices and putting an edge between them if the coin shows heads. Compute the probability, as a function of n, that such a graph has diameter 1, 2, and more than 2.*

Problem 13.31 *Suppose we are generating random regular graphs of degree 2 and 3 starting with C_{400} and $P_{200,1}$, respectively. Experimentally or logically, estimate the probability that a given random regular graph will be connected.*

Problem 13.32 *If we generate a random regular graph, and, by accident, it is not a connected graph, does this cause a problem? Why? Is the answer different for different problems?*

Problem 13.33 *In the definition of partial order, divisibility of the positive integers was given as an example. Prove that divisibility on the positive integers is a partial order (by checking properties (i)-(iii)) and also show that divisibility does* not *partially order the nonzero integers.*

Problem 13.34 *Does the relationship "s is a prefix of t" on strings form a partial order? Prove your answer.*

Problem 13.35 *A* total order *is a partial order with the added property that any two elements can be compared, e.g., the traditional operation $<$ on the real numbers. What prevents the performance partial order from being a total order?*

Problem 13.36 *Reread Problem 13.35 and give 3 examples of total orders, including a total order on the set of complex numbers.*

Problem 13.37 *Verify that the performance partial order is, in fact, a partial order. This is done by checking properties (i)-(iii).*

Problem 13.38 *For Experiments 13.14-13.19, decide if it possible to compute the edge and entropy of the graphs as we did in the neutral graph behavior experiments in Section 13.1. What is required to be able to make these computations?*

Problem 13.39 *Compute the diameter of $T_{n,m}$, the $n \times m$ torus.*

Problem 13.40 *Compute the diameter of H_n, the n-hypercube.*

Problem 13.41 *What is the smallest number of edges that can be deleted from the 5-hypercube to drive the diameter to exactly 6?*

Problem 13.42 *The operation* simplexification *is described in Appendix C. We can create graphs with degree n by starting with K_{n+1} and simplexifying vertices. For $n = 3, 4, 5$, determine what population sizes are available, by starting with a complete graph and repeatedly simplexifying vertices.*

Problem 13.43 *Reread Problem 13.42. Would graphs created by simplexification behave differently from other graphs used in this section in a GBEA?*

Problem 13.44 Essay. *The list of graphs used in this chapter is modest. Pick and defend a choice of graph for use with the Traveling Salesman problem. An experimental defense is time consuming, but superior to a purely rhetorical one.*

Problem 13.45 Essay. *The list of graphs used in this chapter is modest. Pick and defend a choice of graph for use with the 3-input parity problem. An experimental defense is time consuming, but superior to a purely rhetorical one.*

Problem 13.46 Essay. *A lexical fitness function seeks to smooth the landscape of a difficult fitness function by adding a tie-breaker function that points evolution in helpful directions. A graph-based algorithm breaks up a population, preventing an early good gene from taking over. Do these effects interfere, reinforce, or act independently?*

Problem 13.47 Essay. *The ordered sequence of degrees, the number of vertices, and number of edges in the graph are all examples of invariants. Choose the invariant that you think most affects performance in an experiment you have performed. Defend your choice.*

Problem 13.48 Essay. *Crossover is the most controversial of the variation operators used. At one extreme, people claim that the ability to mix and match building blocks is the key one; at the other extreme, people claim that crossover is unnecessary and even counterproductive. Since both sides have experimental evidence in favor of their propositions, the truth is almost certainly that crossover is very helpful when there are building blocks to be mixed and matched. Question: given what you have learned from the experiments in this chapter, can the behavior of a problem for graphs of different connectivities be used as a probe for the presence of building blocks? Good luck, this is a hard question.*

13.4 Genetic Programming on Graphs

The most complex types of representations we've examined have been various different genetic programming representations including parse trees, GP-automata, and ISAc lists. In this section, we will check the impact of graphs on solving problems using these representations. The simplest genetic programming problem available is the PORS problem from Chapter 8. Review the PORS problem and Experiments 8.2-8.4. Let's check the impact of graphs on the three classes of PORS trees.

Experiment 13.24 *Build or obtain software for a graph-based evolutionary algorithm to work with the PORS problem. Use random selection of the parent and roulette selection of the co-parent, with elite replacement of the parent with the better of the two children. Use subtree mutation 50% of the time and subtree crossover 50% of the time, with the 50% chances being independent. Use the graphs* C_{720}, $P_{360,1}$, $P_{360,17}$, $T_{4,180}$, $T_{24,30}$, H_9, *modified by simplexifying 26 randomly selected vertices, and* K_{512}. *Simplexification is described in Appendix C. Be sure to create these graphs once and save them, so that the same graph is used in each case.*

Do 400 runs per graph for the Efficient Node Use Problem on $n = 14, 15,$ *and* 16 *nodes. Document the impact of the graphs on time-to-solution with 95% confidence intervals. Do any of the graphs change the relative difficulty of the three cases of the PORS problem?*

We have not explored, to any great extent, the impact of local mating rules on the behavior of the system. Experiment 13.24 uses a very extreme form of local mating rule which insists on improvement before permitting change and which refuses to destroy a creature currently being selected with a fitness bias (the co-parent). Let's check the impact of protecting the co-parent in this fashion.

Experiment 13.25 *Modify Experiment 13.24 to use elite replacement of parent and co-parent by both children. Of the 4 structures, the best two take the slots occupied by the parent and co-parent. Compare the results to those obtained in Experiment 13.24.*

Elitism amounts to enforced hill climbing when used in the context of local mating rules. Different regions of the graph may be working on different hills. If a given problem has local optima or other traps, then this hill climbing may cause problems. On the other hand, the boundary between sub-populations on distinct hills may supply a source of innovation. Let's do the experiment.

Experiment 13.26 *Modify Experiment 13.24 to use absolute replacement of parent and co-parent by both children. Compare the results to those obtained in Experiments 13.24 and 13.25. Is the impact comparable on the different PORS problems?*

The PORS problem is very simple and highly abstract. Fitting to data, using genetic programming to perform symbolic regression, is less simple and a good deal more abstract. In Experiment 9.5, we found that it was not difficult to perform symbolic regression to obtain formulas that accurately interpolate points drawn from the fake bell curve

$$f(x) = \frac{1}{x^2 + 1}.$$

Let's see if time-to-solution or the rate of accurate solutions can be increased with a GBEA.

Experiment 13.27 *Rebuild Experiment 9.6, symbolic regression to samples taken from the fake bell curve, as a GBEA. Use random selection for the parent and roulette selection for the co-parent and absolute replacement of the parent by the better child. Also, perform baseline studies that use tournament selection. For each graph, perform tournament selection with tournament size equal to the graph's degree plus one. Don't use normal tournament selection - rather, replace the second most fit member of the tournament with the best child. This makes the tournament selection as similar as possible to the local mating rule, so that we are comparing graphs to mixing at the same rate without the graph topology. Use the graphs* C_{512}, $P_{256,7}$, $T_{16,32}$ *and* H_9.

Perform 400 runs per graph. Report the impact both by examining the number of runs that find a correct solution (squared error less than 10^{-6} *over the entire training set) and by examining the time-to-solution on those runs that achieve a correct solution.*

The somewhat nonstandard tournament selection used in Experiment 13.27 is meant to create amorphous graph-like structures that have the same degree but constantly moving edges. This controls for the effect of degree as opposed to topology in another way than using random regular graphs. It somewhat begs the question of comparing normal tournament selection to a GBEA. It's also not clear that it's the "right" control.

Experiment 13.28 *Perform Experiment 13.27 again, but this time use standard tournament selection of the appropriate degree (reuse your graph results). Compare the results with both the graphs and the nonstandard tournaments from the previous experiment.*

To complete the sweep of the controls for degree versus connectivity in the graphs, let's perform the experiment again with random regular graphs.

Experiment 13.29 *Perform an extension of Experiment 13.27 as follows. Pick the best and worst performing graphs and generate 5 regular random graphs of the same degree using twice as many edge swaps as there are edges in the graphs. Run the GBEA again with these graphs. Does this control for degree versus topology have a different effect than the one use in Experiment 13.27?*

Data Inputs				Encoding Inputs		Output
0	1	2	3	lo	hi	
0	*	*	*	0	0	0
1	*	*	*	0	0	1
*	0	*	*	1	0	0
*	1	*	*	1	0	1
*	*	0	*	0	1	0
*	*	1	*	0	1	1
*	*	*	0	1	1	0
*	*	*	1	1	1	1

Figure 13.5: Truth table for the 4-multiplexer (* entries may take on either value without affecting the output.)

We've already reprised the neural net 3-input parity problem from Chapter 11 in Section 13.3. Evolving a parity gate is one of the standard test problems in evolutionary computation. Let's take a look at the problem using genetic programming techniques.

Experiment 13.30 *Rebuild Experiment 11.12 to run as a GBEA. Use two local mating rules: roulette selection of the parent and co-parent, with absolute replacement of both parent and co-parent; random selection of the parent and rank selection of the co-parent with absolute replacement of both parent and co-parent. Use the graphs* C_{720}, $P_{360,1}$, $P_{360,17}$, $T_{4,180}$, $T_{24,30}$, *and* K_{720}. *For each graph and local mating rule, perform 400 evolutionary runs. Compare the performance of the different graphs. Were different graphs better for the AND and parity problems?*

While parity is a common target problem for evolutionary computation, the *multiplexing* problem is also a common target. The 2^n- multiplexing problem takes 2^n *data inputs* and n *encoding inputs*. The encoding inputs are interpreted as a binary integer that selects one of the data inputs. The output is set to the value of the selected data input. The truth table for a 4-input/2 encoding bit 4-multiplexer is given in Figure 13.5.

The truth table given in Figure 13.5 nominally has 64 entries, one for each of the 2^6 possible inputs. The use of the symbol * for "either value" compresses the table to one with

8 inputs. The number of times you can use a * in this fashion in writing a truth table of minimal length is the *degeneracy* of a logic function. On average, logic functions are easier to create via evolutionary computation, if they have a higher degeneracy. Let's check this assertion experimentally.

Experiment 13.31 *Modify the software for Experiment 13.30 to work with the 4-multiplexing function and the 6-parity function. Use whichever local mating rule worked best for the 3-input parity problem. Which of these two problems is harder?*

Let us now turn to the various grid robot tasks in Chapters 10 and 12. Experiment 13.7 has already touched on the Tartarus problem and the need to use generational rather than steady state GBEAs on these problems. This requirement for a generational algorithm comes from the need to compare oranges to oranges in any problem where sampled fitness is used as a surrogate for the actual fitness. Recall that in the 6×6 Tartarus problem, there are in excess of 300,000 boards, and we can typically afford no more than a few hundred boards for each fitness evaluation. This means that, rather than computing the true fitness in each generation (the average score over all boards), we use a sample of the boards to compute an estimated fitness.

Experiment 13.32 *Use the GP-language from Experiment 10.13, without the RND terminal, to run a generational GBEA, i.e., one with deferred updating. Use 100 Tartarus boards rather than 40 for the fitness function, selecting the 100 boards at random in each generation. For each vertex in the graph, roulette select a co-parent and create a pair of children using subtree crossover 25% of the time and subtree mutation 50% of the time, independently. Use 20-node random initial trees and chop trees that exceed 60 nodes. Run the algorithm on C_{256}, $T_{16,16}$, and H_8, as well as 2 random regular graphs of degree 4 and 8.*

For each graph, perform 100 runs. Compare the graphs with two statistics: the time for a run to first exhibit best fitness (3.0) and the mean final fitness after 500 generations. If results are available for Experiment 10.13, also compare with those results. Compute the performance partial order for this experiment.

As we know, the GP-trees with 3 memories were not our best data structures for Tartarus. Both GP-automata and ISAc lists exhibit superior performance.

Experiment 13.33 *Repeat Experiment 13.32 but use GP-automata this time. Use the GP-automata with 8 states and null actions (λ-transitions) of the same kind as were used in Experiment 10.18. Be sure to use the* same *random regular graphs as in Experiment 13.32. Does the identity of the best graph change at all? Compute the performance partial order for this experiment and compare it with the one from Experiment 13.32.*

When we have a sampled fitness, as with Tartarus, there is room to experiment with the allocation of fitness trials. The graph topology gives us another tool for allocating fitness trials. Recall that the hypercube graph can be thought of as having a vertex set consisting of all binary words of a given length. Its edges are pairs of binary words that differ in one position. The *weight* of a vertex is the number of 1s in its binary word.

Experiment 13.34 *Modify the software from Experiment 13.33 as follows. First, run only on the graph H_8. The possible vertex weights are thus $0, 1, 2, \ldots, 8$. For words of weight 0,1,7, or 8, evaluate fitness on 500 boards. For words of weight 2 or 6, evaluate fitness on 100 boards. For words of weight 3 or 5, evaluate fitness on 40 boards. For words of weight 4, evaluate on 20 boards.*

Use a fixed set of 500 boards in each generation, giving GP-automata that require fewer evaluations boards from the initial segment of the 500. In a given evolutionary run of 500 generations, this will result in 20,480 instances of a dozer being tested on a board. Also, rerun the unmodified software so that each dozer on the H_9 graph uses 80 fitness evaluations. This results in the exact same number of fitness evaluations being used.

Perform 100 runs for both methods of allocating fitness. Using a fixed 5000 board test set, as in Experiment 10.20, make histograms of the 100 best-of-run dozers from each set of runs. Are the methods different? Which was better?

There is a possibility implicit in the use of the distributed geography of a GBEA that we have not yet considered. Suppose that we have different fitness functions in different parts of the graph. If the tasks are related, then the easier instance of the problem may prime progress on the harder instance.

Experiment 13.35 *Modify the software from Experiment 13.33 as follows. First, run only on the graph H_8. Do two sets of runs with 100 Tartarus boards used for each fitness evaluation in the usual fashion. In one set of runs, use the 8×8 Tartarus problem with 10 boxes, only. In the other, use the 8×8 Tartarus problem on those vertices with odd weight and the 6×6 Tartarus problem on those vertices with even weight. For both sets of runs, use a fixed 5000 board test set, as in Experiment 10.20, to make histograms of the 100 best-of-run dozers* for the 8×8 task *from each set of runs. How different are the histograms?*

Let's shift both the virtual robotics task and the representation in the next experiment. The Herbivore task was an easier problem when judged by the rate of early progress in enhancing fitness.

Experiment 13.36 *Rebuild the software from Experiment 12.12 to work as a generational GBEA. Use the same graphs as in Experiment 13.32 and check the impact of the graphs on performance in the Herbivore task. Compute the performance partial order for this experiment and compare it with the one from Experiments 13.32 and 13.33, if they are available.*

The North Wall Builder task from Chapter 12 has the advantage that it has only one fitness case (board) and, so, runs much faster than Tartarus or Herbivore. Let's do a bi-factorial study of graph and board size.

Experiment 13.37 *Rebuild the software from Experiment 12.17 to work as a GBEA. Use the same graphs as in Experiment 13.32, but change the local mating rule to be random selection of the parent, roulette selection of the co-parent, and elite replacement of the parent by the better of the two children. Check the impact of the graphs on performance of the North Wall Builder task for board sizes:* 5×5, 7×7, *and* 9×9. *Compute the performance partial order for each board size and compare these orders with one another and with all available experiments using the same graphs.*

We have not experimented much with the impact of graphs on competitive tasks (other than Sunburn). We leave this topic for the future, but invite you to design and perform your own experiments. One thing to consider is that it may be hard to compare to populations of competitive agents meaningfully.

Problems

Problem 13.49 *In the PORS system, the two "subroutines"* **(+ (Sto T) Rcl)** *(multiply by 2) and* **(+ (Sto T) (+ Rcl Rcl))** *or* **(+ (+ (Sto T) Rcl) Rcl)** *(multiply by 3, which has two forms) together with the very similar trees,* **(+ 1 1)**, **(+ (+ 1 1) 1)**, *and* **(+ 1 (+ 1 1))**, *that encode the constants 2 and 3 can be used to build up all optimal PORS trees. For the PORS* $n = 15$ *Efficient Node Use Problem, either compute or experimentally estimate the probability that a random initial tree will contain a subtree that encodes 3 or multiplication by 3.*

Problem 13.50 *Reread Problem 13.49 and either compute or experimentally estimate the probability that a random initial tree will contain a subtree that encodes 2 or multiplication by 2.*

Problem 13.51 *Why do we use large numbers of edge swaps when we generate random regular graphs? What would the effects of using a small number be?*

Problem 13.52 *In Experiment 13.29 we checked for the difference in behavior of algorithms using the standard graphs for this chapter and those using random graphs with the same degree. How does the diameter of the standard graphs compare with the diameter of random regular graphs of the same degree? Why?*

Problem 13.53 *In order to generate random regular graphs of degree* d *with* n *vertices, we need a starting graph with the given degree and vertex count. Give a scheme for creating starting graphs of degree* d *with* n *vertices for as many degrees and vertex sizes as you can. Remember that the number of vertices of odd degree must be even.*

Problem 13.54 *The notion of degeneracy of a truth table is given on page 361. Compute the degeneracy for each of the following families of logic functions and give the resulting shortest possible truth tables.*

(i) n-input OR

(ii) n-input AND

(iii) 2^n-multiplexing

(iv) n-bit parity

Problem 13.55 *Prove that a logic function and its negation have the same degeneracy.*

Problem 13.56 *Prove that the parity function and its negation are the only logic functions whose truth tables have zero degeneracy.*

Problem 13.57 *Carefully verify the assertion in Experiment 13.34 that there will be 20,480 evaluations of a dozer on a board in each generation.*

Problem 13.58 *Reread Experiment 13.34. Come up with compatible methods of varying the number of fitness trials for (i) C_{256}, (ii) $T_{16,16}$, (iii) $P_{128,1}$, and $P_{128,7}$. Make sure that your total fitness evaluations in a generation are a multiple of the number of vertices, to permit evaluation on a fixed number of boards as a baseline.*

Problem 13.59 *Experiment 13.35 mixed the 8×8 and 6×6 Tartarus problems. Is there a problem in this experiment with having to compare dozers evaluated with different fitness functions?*

Problem 13.60 *Reread Experiment 13.35. Suppose that, instead of dividing the 8×8 and and 6×6 problems by odd and even weight vertices, we had divided the hypercube so that the 6×6 fitness function was on the vertices with most significant bit "0" and the 8×8 fitness function was on the vertices with most significant bit "1". In this case, there would be vertices that had neighbors evaluated with each of these fitness functions. Give a means of finding a scaling factor that permits comparison of these two fitness functions and defend it. Consider: while the 8×8 function can return higher values, initial progress is probably more rapid on the 6×6 function.*

Problem 13.61 *In some sense, Experiment 13.35 mixes the 8×8 and 6×6 Tartarus problems as much as possible. Is this good or bad?*

Problem 13.62 *Reread Experiment 13.35. Would you expect this sort of fitness mixing to work better or worse on the PORS problem with* $n = 12$ *and* $n = 15$*? Assume that trees are chopped to fit their node.*

Problem 13.63 Essay. *For the most part, we have used regular graphs, as a way of controlling for one important graph parameter. Is there any reason to think the performance with graphs that are* not *regular, but have similar average degree, would be different? Explain.*

Problem 13.64 Essay. *The correct answer to the PORS* $n = 15$ *Efficient Node Use Problem is EVAL(T)=32. Given the answers you found to Problems 13.50 and 13.49 discuss why the cycle is the best graph, of those used, for the PORS* $n = 15$ *Efficient Node Use Problem.*

Problem 13.65 Essay. *Is the tournament selection used in Experiment 13.27 better or worse than the standard type of tournament selection? For which problems?*

Problem 13.66 Essay. *Create a system for evolving graphs. Give a representation including data structure and variation operators. Do not worry about the fitness function.*

Problem 13.67 Essay. *A persistent theme in this chapter is the comparison of graphs to see which graph helps the most on a given problem. Discuss the practicality of searching for good graphs by using performance of a GBEA as a fitness function.*

Problem 13.68 Essay. *Explain why high degeneracy in a logic function yields an easier evolutionary search problem.*

Problem 13.69 Essay. *Suppose that we use a graph with several thousand vertices and place 5 Tartarus boards as well as a dozer on each vertex. We then run a GBEA in which parents are selected at random and co-parents are selected by roulette selection after they are evaluated on the boards sitting on the parent's node. Will this scheme find effective dozers? Explain.*

Problem 13.70 Essay. *Invent and describe a system for using GBEAs to locate hard Tartarus boards for* 8×8 *or larger boards.*