Genetic Algorithms and Graph Coloring

CS 591: Evolving Software

Assignment 2

Due: Feb. 12, 12:15 pm

1 Introduction

In this assignment you will learn about genetic Algorithms, graph coloring problems, Erdős-Renyi graphs, and neutral spaces. You will also be asked to design your own test cases and (optionally) write a short literature review.

We will use genetic algorithms (GAs) to solve the graph-coloring problem. The problem takes as input a graph $G = (V, E)$ and an integer k which designates the number of different colors (labels) to be used. A successful coloring assigns each $v \in V$ a color $\in \{1, 2, 3, \ldots k\}$ such that all adjacent vertices have different colors. The general problem of determining a graph coloring is computationally intractable for $k > 2$, so we will use the GA to search heuristically for solutions.

The assignment is due at the beginning of class on Feb. 12. Please hand in a printed copy of a 3-4 page paper describing your assignment, formatted according to the ACM SIG Proceedings format, available from:
`https://www.acm.org/publications/proceedings-template-16dec2016`,
together with the output, and a url link to the code you wrote to complete the assignment. I prefer to receive hard copy, but if you submit your assignment electronically, please send the paper in pdf format rather than docx or tex.

2 Genetic Algorithm

You are free to choose any publicly available genetic algorithm software or to write your own using the language of your choice. Before diving into the graph coloring problem, you should first get your GA running on a simple problem (like max ones) and convince yourself that it is working correctly and that you will be able to modify it easily to complete the assignment.

2.1 Representation

The input for each graph will be a file. The first line of the file will indicate how many colors to use, k, and the remainder of the file will be a list of edges represented as ordered pairs (the two connecting nodes separated by a space) with one edge per line, e.g.,

```
3
1 3
0 2
2 1
```

You can assume that the nodes are each labeled with a unique (for the node) positive integer starting at zero. Thus, in the example above there are 4 vertices, connected by 3 edges, and the task is to find a 3-coloring of the graph. We will provide the data sets a few days before the assignment is due, so you will need to design your own test cases to test your GA as you work through the assignment.

Your GA will maintain a population of individuals, where each individual specifies a candidate coloring for the graph. The most natural representation of a candidate graph coloring along the chromosome is one where the first position corresponds to Node 1, the second position to Node 2, etc. Then the value (allele) at each position will correspond to the color. If you use a bit string encoding you will need to allocate log_2k bits for each gene. This scheme has the drawback that for some values of k not all bit values will correspond to a legal color. There are various ways to address this complication, and your writeup should explain all of your design decisions, including this one.

2.2 Fitness Functions

For each individual, we want to know how close its encoded coloring is to a legal coloring of the graph (to review, a legal coloring is one where no nodes connected by an edge have the same color). Let $G = (V, E)$ be a graph with n nodes and m edges. If $c(i)$ is the assigned color of node i, then a coloring of a graph is $C(G) = \{c(0), c(1), \ldots, c(n)\}$. For a given edge, $i, j \in E$, let the function $\delta(i, j) = 1$ if $c(i) \neq c(j)$, and $\delta(i, j) = 0$ if $c(i) = c(j)$. A natural fitness function then is:

$$F(C(G)) = \frac{\sum_{i,j \in E} \delta(i, j)}{m}$$

That is the fraction of edges where the endpoints do not share the same color, divided by the total number of edges.

In addition to searching for valid colorings, we may also want to find *balanced* colorings. That is a coloring in which no particular color is used more than the rest. A possible fitness function for this type of problem might be:

$$F(C(G)) = \frac{\sum_{i,j \in E} \delta(i, j)}{m} \prod_{i=1}^{k} \frac{|V_j|}{n}$$

where $|V_j|$ is the size of the set of all nodes of color j.

3 Assignment

3.1 Part I: Finding Colorings

For each input graph, use your GA to find the best coloring of the graph. Be sure to report what values you used for the parameters of the GA (e.g. mutation rate, crossover rate, population size, number of generations, selection method, and the representation that you use). Create Figure 1 that reports the best, average and worst fitness of the population over generations for a representative run for one input graph, and print out the best coloring found during the run (you can simply print out the bit string of your highest fitness individual). Report in an appendix, the best coloring that you find for each input file (graph).

Since the GA is a stochastic algorithm, a single run may not be indicative of its average performance, so next you need to devise an experiment to study the GA's performance over multiple runs. Here you will pick a single input graph. Hint: Pick a more difficult graph, because your results will not be interesting if the problem is one that the GA can solve easily. Describe the experiment and report the results in Figure 2.

Next, you will use the GA to find balanced colorings. For each of the input graphs, run the GA to find the highest fitness balanced coloring. Now, create a Table 1 that reports the name of the graph input file, the fitness of the best coloring (from above), the fitness of the best balanced coloring, how many generations it took to find each, and whether the best balanced coloring is also a valid coloring.

4 Part II: Neutral Coloring Landscapes

In this section of the assignment we will explore the neutrality of graph colorings. First, you will need to implement 1-step neutrality. That is, starting with a single valid coloring, make copies in which one gene is changed to one of the other colors. Performing this process systematically should produce $n * (k - 1)$ individuals. You will then report the fraction of those individuals that are also legal colorings (*neutrality*).

For each of the input files, report the neutrality of the valid colorings of the graph in table format (Table 2).

4.1 Neutral Coloring Landscapes on Random Graphs

Next, we will study how the neutrality changes with some properties of random graphs. Here we will study a specific type of graph known as an Erdős-Renyi graph. Research Erdős-Renyi graphs, and implement an algorithm that can create an Erdős-Renyi graph with a given number of nodes n and a given average degree d. Describe your algorithm for producing these random graphs and give a small example of a graph generated from your code (Figure 3).

Fix $n = 50$. For various values of d (e.g., in the range from 0 to 5, increments less than 1), generate multiple random graphs. Use your GA to find the highest fitness coloring of these graphs using your original setup from Part I, with a fixed $k = 3$. Create a plot that reports the average degree d on the horizontal axis and the average maximum fitness on the vertical axis (Figure 4).

Finally, starting with a valid coloring found in the previous section, plot the neutrality of the colorings for these various values of d (Figure 5).

5 Extra Credit

There are many heuristic algorithms for find colorings on a graph. Find and implement some of these algorithms (or use existing libraries). Compare the results of these algorithms to the results you found above. Be sure to cite your sources appropriately.

6 What to hand in

Hand in a short report (3-4 pages) that describes your implementation. Think carefully about which runs to report and select those that best illustrate the phenomena you want to show. Document the programming language, any external or built-in libraries that you used, major design decisions, etc.

For each assigned experiment, include a short discussion of your results and what you think they mean. Think carefully about how to present your results in a convincing but succinct manner. Organize your paper into the following sections:

- Introduction: Summarize briefly the problem you are trying to solve and how you went about it.
- The Genetic Algorithm: Describe basic details about your algorithm, your default parameter settings, and (important) report the experiments you did to convince yourself that the GA is working correctly.
- Using the GA to find graph colorings
- Neutrality of graph colorings
- Discussion and conclusion

Remember to cite all of your sources using a consistent citation style and include a proper bibliography.

Please include a url pointer to your code and instructions for how to run it as an appendix to your report.