**Background:**

An evolutionary algorithm is a technique used by computer scientists to find an approximation of a problem that, using conventional methods, would take an unreasonable amount of time to find an exact solution. In an evolutionary algorithm a population is maintained and operated on using techniques from biology such as recombination and mutation to evolve a solution.

A graph is a data structure consisting of nodes and edges that can represent networks such as power grids, computer networks, and much more. Graph bisection is a graph problem in graph theory where a graph’s nodes are partitioned between two sets. The graph bisection problem is an NP-Hard problem and can thus be approximated using evolutionary algorithms

**Motivation:**

Some graphs can be bisected easier than others. A co-evolutionary algorithm can be used to evolve a generalized partition on a graph with a fixed set of nodes, while simultaneously evolving a graph that is difficult to bisect.

It is often useful to penalize graphs whose partitions are not fully connected. A constraint satisfaction can be applied to the fitness function to penalize a graph based on the number of subgraphs it contains. This is simply done by multiplying the number of subgraphs by a constant and subtracting the result from the original fitness function.

**Methodology:**

Figure :

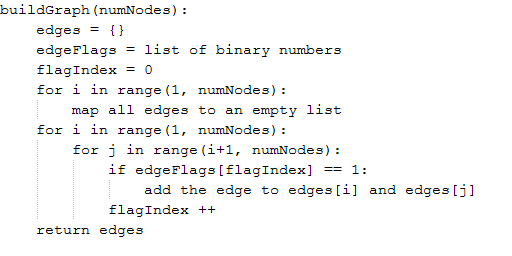


Figure 1 describes pseudocode for generating a random graph, represented using a list of dictionaries.

Figure

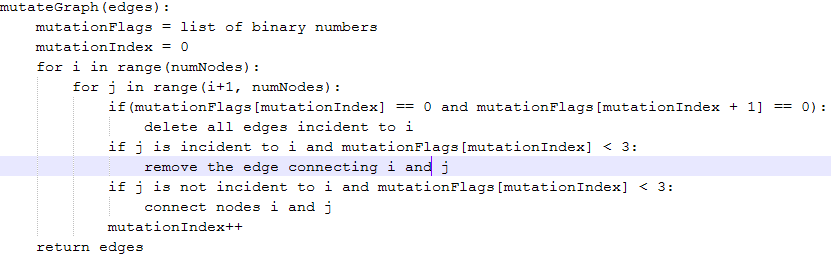


Figure 2 represents pseudocode for mutating a graph. This implementation uses Gaussian mutation in which there is a high chance of making a small change (adding or removing an edge) and a small chance of making a large change (deleting all edges incident to a node).

Figure

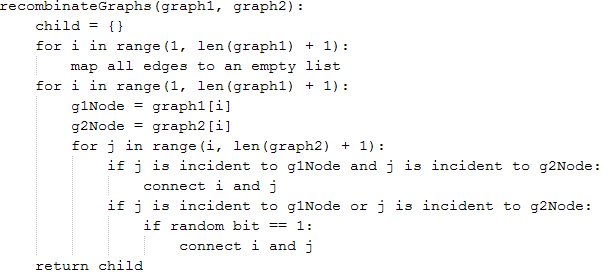


Figure 3 shows pseudocode for recombination of two graphs. All edges that exist in both graph1 and graph2 will appear in the child; however, if an edge exists in either graph1 or graph2 there is a 50% chance it will appear in the child. If an edge does not exist in graph1 or graph2 it will not appear in the child. In this situation the edge is not “dead” because it could be recreated in the mutation operator.

Figure

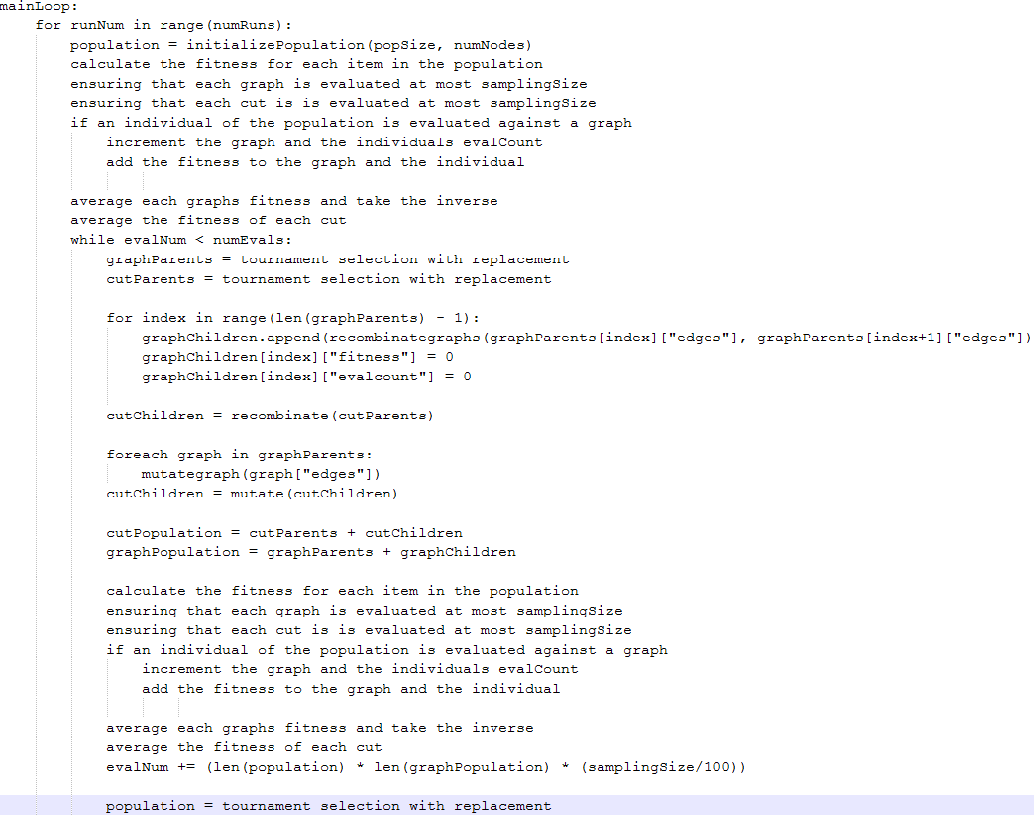


Figure 4 describes pseudocode for the body of the co-evolutionary algorithm. In which 100,000 evaluations are performed for each of all 30 runs. It is important to note that the fitness of a graph is inversely proportional to the fitness of the cuts evaluated against it. This is because a graph that is hard to partition is desired.

**Results:**

Figure

Figure 5 compares the average fitness of a partition using the constraint satisfaction with a penalized fitness against the same algorithm excluding the penalty for subgraphs.

Figure

Figure 6 compares the best fitness of a partition using the constraint satisfaction with a penalized fitness against the same algorithm excluding the penalty for subgraphs.

Figure

Figure 7 shows that on average a better graph is generated when the constraint satisfaction is implemented.

Figure

Figure 8 shows that the best fitness of a graph is chaotic.

The data in figures 7 and 8 show that there wasn’t a large increase in the fitness of the graphs. This could be a result of cyclic evolution where the populations evolve and devolve in a cycle. A solution to this problem would be the implementation of a “hall of fame” where the best graphs would be stored. If an individual is created that already exists in the hall of fame it would be removed from the population.

Table

|  |  |
| --- | --- |
|  |  |
| The mean for the simpleEA is less than the mean of the coEA and F is greater than Fcritical therefore assume equal variances. | The mean for the simpleEA is less than the mean of the coEA and F is greater than Fcritical therefore assume equal variances. |
|  |  |

Table 1 provides statistical analysis of an ordinary EA vs. the coEA where the constraint satisfaction is used. The best fitness was logged for 30 runs and averaged to get the data for each global run. The f-test was then performed on this data in order to determine if the variances are equal or unequal. Next the appropriate t-Test was calculated.

Table

|  |  |
| --- | --- |
|  |  |
| The mean for the simpleEA is less than the mean of the coEA and F is greater than Fcritical therefore assume equal variances. | The mean for the simpleEA is less than the mean of the coEA and F is less than Fcritical therefore assume unequal variances. |
|  |  |

Table 2 provides statistical analysis of an ordinary EA vs. the coEA where there is no penalty for subgraphs. The best fitness was logged for 30 runs and averaged to get the data for each global run. The f-test was then performed on this data in order to determine if the variances are equal or unequal. Next the appropriate t-Test was calculated.