Jonathan Hamm

Evolutionary Computation (CS 4623)

Projects 1,2,3

Graph Bisection Problem (Minimum Bisection)

The goal of this problem is find the minimum cut for a weighted graph, where the cut is a bisection. The problem uses a binary representation for chromosomes, where set bits indicate one side of the bisection, and the unset bits indicate the opposite side. Chromosomes must have an equal number of 1’s and 0’s in order to model a feasible bisection.

My fitness function works by summing the weights of all cut edges. Infeasibles are dealt with by multiplying the difference in number of 1’s and 0’s by 16, and adding the result to the sum of cut edges. The number 16 was simply chosen through testing. The population size varies in performance by the size of the problem. In general, however, I used population sizes between 30 and 50, which yielded the best performance. Elitism is implemented by checking if one of the current parents is the best solution. If it is, the child chromosome is replaces a less fit chromosome so the parent is preserved. Elitism only applies to feasible chromosomes in my implementation. The algorithm terminates after a number of generations are reached. Since larger problems will require more generations, the number of generations is a function of the chromosome size. In my case, results showed that a good approximation was 10x the chromosome size. I implemented roulette, rank, and tournament selection. Roulette and tournament yielded the best performance in my implementation. I chose to make my crossover rate a function of the population size.

The graph is read from a file that lists the vertices and edges in two separate sets. The file is parsed and loaded into a data structure. The structure contains a table of vertices where each vertex is a structure containing the node name and an array of all edges connected to it. An edge contains pointers to the vertices it connects along with a floating-point value for its weight. The indices of vertices in the graph table are arranged in the same order as the ‘vertices’ (or bits representing them) in the chromosome. For example, a bit at position 5 in a chromosome corresponds to index 5 in the graph’s vertex table. Chromosomes are stored as a dynamic array of 64-bit integers. This allows certain operations to process 64-vertices “at once” or in one register for 64-bit processors. Since the memory allocated to chromosomes must be 64-bit aligned, the program stores an additional 64-bit mask for chromosome sizes that are not multiples of 64. The largest performance constraint is the summation of cut edges. The basic algorithm is as follows: