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 the set bits indicate one side of the bisection, and the unset bits indicate the other side. Chromosomes must have an equal number of 1’s and 0’s in order to represent a feasible bisection.

The graph is read from a file that lists the vertices and edges in two separate sets. The file is parsed and loaded in 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 in the same order as the ‘vertices’ (or bits representing them) are in the chromosome. Any vertex structure can be accessed by a chromosome bit by using its bit position as the graph table’s array index. Chromosomes are stored as a dynamic array of 64-bit integers. This allows many operations to process 64-vertices “at once” or in one quad-word for 64-bit processors. If the number of vertices is not a multiple of 64, the last 64-bit integer in the array needs a value to mask out the extra bits. This value is computed at startup.

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. Summation requires iterating over all set bits in a chromosome using a find-first-set-bit operator.