Checkpoint Report

Description (details and pseudocode)

Branch and Bound Algorithm:

Given graph G= (E, V), let C' be the partial solution and G'= (E', V') be the subgraph of G that is not covered by C'.

Subproblem: (G', C')

Start with (G,ø), each time choose a vertex v if it helps to cover more edges and add it to the partial solution C'.

Approximate lower bound is obtained from Linear Programming Relaxation, whose description is as follows:

Define decision variables x_i, i = 1, …, n

x_i = 0 if vertex i is not included in VC

x_i = 1 if vertex i is included in VC

Our goal is to minimize $\sum_{i=1}^{n} xi$

subject to

$x_u + x_v \geq 1 \quad \forall \ (u,v) \in E$

$0 \leq x_i \leq 1$

Round xi in the solution to its nearest integer, 1 or 0.

Pseudo-code:

Given G = (V, E)

opt ← number of vertices in G

C ← ø

index ← 0

Function bnb (G, C, index)

```
    if (no edges in G & |C|<opt) {
        opt = |C|;
    }
    if (index > |V|) return;
    if ( |C| + LB(G) > opt) return;
    for (i← index; i<|V|; i++){
        C ← C + {vertex i}
        G ← G - {edges connected to vertex i}
    }
    recursively call bnb on (G, C)
    C ← C - {vertex i}
    G ← G + edges connected to vertex i
```

return opt

Approximation Algorithm:

Apply the maximum degree greedy algorithm. Each time select the vertex with maximum degree and delete the covered edges. Terminate when all edges are covered.

Pseudo-code:

Data: a graph G= (V, E)

Result: a vertex cover of G

```
C←0
while E≠ø do
    select a vertex u of maximum degree;
    V←V-{u};
    E←E-{e∈E: u∈E}
    C←C∪{u}
end
return C
```

Preliminary Results

	Branch and Bound			Approximation		
Dataset	Time(s)	VC Value	RelErr	Time(s)	VC Value	RelErr
football	538.14	100	0.06	0.00	96	0.02
email				0.03	605	0.02
jazz	107.98	182	0.15	0.00	159	0.01
karate	2.36	14	0.00	0.00	14	0.00
as-22july06				0.12	3307	0.00
power				0.04	2277	0.03
star				0.14	7374	0.07
star2				0.11	4697	0.03
delaunay				0.02	737	0.05
netscience				0.03	899	0.00

Cutoff time for branch and bound algorithm is set to 10 minutes, and the results in the above table are the best ones that can be found in 10 minutes. Only partial results for some small graphs are presented here since the algorithm needs improvement to apply on larger graphs.

Ideas for Remaining Algorithms

Local Search Algorithm 1:

The first strategy chosen will be randomized hill climbing with restarts. The initial solution will be constructed by adding random vertices to a candidate solution until the candidate solution is a vertex cover for the input graph. This is taken as the initial solution to the problem. The neighborhoods from the initial solution will consist of subsets of the chosen solution, or the current solution plus another vertex. Due to hill climbing's tendency to optimize locally, here each neighbor solution is found by removing one vertex from the currently considered candidate solution to form a solution that still is a vertex cover. After the best solution in the neighborhood from the given candidate solution is found, there will be a randomized restart until the ten-minute time cap is over. In the end, the best local solution from each of the hill

climbing runs is outputted with the time it was found.

Local Search Algorithm 2:

The second strategy will use simulated annealing to incorporate a different technique that does not always make greedy choices towards better solutions. Simulated annealing will use the same process to determine an initial solution as local search, ie. choose random new vertices until a vertex cover has been found. In addition, the neighborhoods are defined the same way as in hill climbing, but a temperature function is used to move between candidate solutions. This will implement more randomized behavior in the beginning to encourage exploration of worse solutions than the current one, and eventually use more greedy criteria as time passes, especially toward the end of the ten-minute runtime cap. Simulated annealing will periodically use “restarts”, which reset the candidate solution to the best one found thus far. This should be done every 2 minutes or after 20 iterations where a better solution is not found using the temperature function. After the ten-minute runtime cap, simulated annealing will return the best solution found and when it was found.