Introduction The Minimum Vertex Cover (MVC) problem seeks to find a subset of vertices for a given graph G such that all the said vertices touch all edges of the graph. This is a well known NP Complete problem that has many real world applications including finding phylogenetic trees based on protein domain information abu2004kernelization or finding where to place bunkers to cover all cities. For this project, our group was tasked with solving the MVC problem using three different approaches: branch-and-bound, local search, and approximation.

Approximation Algorithm The approximation approach is one that sacrifices accuracy of the result in favor of speed. Generally, approximation algorithms have a provable guarantee that they will be no worse than $p(n) \times OPT$, where $p(n)$ is some value greater than or equal to 1 that may depend on the size of the problem and OPT is the optimal solution. For the MVC problem, the approximation approach involves arbitrarily picking an edge in the given graph, then removing is and all neighboring edges from the graph. Then a new edge is arbitrarily picked from the remaining ones and the process is repeated until all the edges have been removed. The vertex cover will then be the nodes corresponding to all the arbitrarily picked edges.

Local Search In addition to the traditional hill climbing approach, the local search I algorithm has options such as forward stopping when the optimal vertex count is reached, remembering the best solution obtaining so far, growing and shrinking (either direction traversal). The local search approach begins with an initial solution and then tries to obtain the global minima by choosing neighbours and moving to the neighbour with the best eval function. The eval function for the local search approach is the number of edges incident on the vertex that is to be added to the current set of vertices in our current search iteration.

Branch and Bound Branch and bound algorithm derives from backtracking strategy for decision problems, which iterates through all the possible solutions in the search space and eliminates some unpromising alternatives from consideration. Branch and bound further refines this idea for optimization problems. It keeps the best solution found so far and computes a lower bound based on this solution. When a solution node for a sub-problem is worse than the lower bound, we consider it as an unpromising solution node and prune it and all its child nodes. Although branch and bound algorithm is a general idea for NP-complete problems, how to compute the lower bound has to be customized for each individual application. This strategy reduces the time consumption on searching for solutions and a global optimal solution is guaranteed, but the worst case is still exponential. Therefore, this algorithm can be useful only when the instances are small.

Problem Consider a graph $G = (V, E)$. Formally, a vertex cover V' is a subset of V such that for each $(u, v) \in E$ it follows that either $u \in V'$or $v \in V'$. This set V is said to cover the edges of G. For the project at hand, this problem is further complicated by searching for the smallest possible V that satisfies the above definition. The MVC problem is known to be NP Complete, and as such it requires specialized algorithms such as local search and approximation in order to be solved.

Related Work TBC Algorithms Approximation Algorithm As mentioned above, the approximation algorithm works by arbitrarily picking an edge in the graph, adding its nodes to the solution, then removing the edge and all of its neighboring edges from the graph, and repeating this process until all edges have been removed. In pseudocode, this looks like the following:

algorithm[ht] $C = \{\}$ E$' = E$ E$'$ is not emptyremove e in E' and add its nodes to C $removealledgesneighboringe from$E$'$ r