COMP 355
Advanced Algorithms
Approximation Algorithms: VC and TSP
Chapter 11 (KT)
Section 35.1-35.2(CLRS)

Rhodes College

1

Coping with NP-Completeness

Brute-force search: Viable option for small input sizes (e.g., n ≤ 20).

Heuristics: Produces a valid solution, but no guarantee on how close it is to optimal.

General Search Algorithms: Examples: branch-and-bound, Metropolis-Hastings, simulated annealing, and genetic algorithms. Performance varies considerably from one problem to problem and instance to instance. But in some cases they can perform quite well.

Approximation Algorithms: Algorithm that runs in polynomial time (ideally), and produces a solution that is guaranteed to be within some factor of the optimum solution.

2

Approximation Algorithms

Q. Suppose I need to solve an NP-hard problem. What should I do?
A. Theory says you're unlikely to find a poly-time algorithm.

Must sacrifice one of three desired features.
- Solve problem to optimality.
- Solve problem in poly-time.
- Solve arbitrary instances of the problem.

ρ-approximation algorithm.
- Guaranteed to run in poly-time.
- Guaranteed to solve arbitrary instance of the problem
- Guaranteed to find solution within ratio ρ of true optimum.

Challenge. Need to prove a solution's value is close to optimum, without even knowing what optimum value is!

3

Performance Bounds

How do we measure how good an approximation algorithm is?
- Given an instance I of our problem, let C(I) be the cost of the solution produced by our approximation algorithm, and let C∗(I) be the optimal solution. (assume that costs are strictly positive values.)
  - For a minimization problem we have $C(I)/C*(I) \geq 1$.
  - For a maximization problem we have $C*(I)/C(I) \geq 1$.
  - In either case, we want the ratio to be as small as possible.
- For any input size n, we say that the approximation algorithm achieves performance ratio bound ρ(n), if for all I, |I| = n we have:

$$\max_I \left(\frac{C(I)}{C^*(I)}, \frac{C^*(I)}{C(I)} \right) \leq \rho(n).$$

4

Performance Bounds

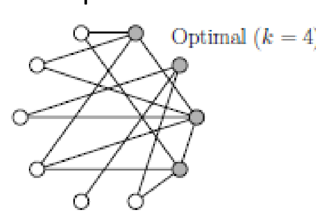
Some NP-complete are *inapproximable*: no polynomial time algorithm achieves a ratio bound smaller than ∞unless P = NP.

Some NP-Complete can be approximated:

- Ratio bound is a function of *n*. (Ex: Set Cover Problem (generalization of Vertex Cover) can be approximated within a factor of O(log n)
- Ratio bound is a constant.
- Approximated arbitrarily well. In particular, the user provides a parameter ε > 0 and the algorithm achieves a ratio bound of (1+ε). Of course, as ε approaches 0 the algorithm's running time gets worse. If such an algorithm runs in polynomial time for any fixed ε, it is called a polynomial time approximation scheme (PTAS).

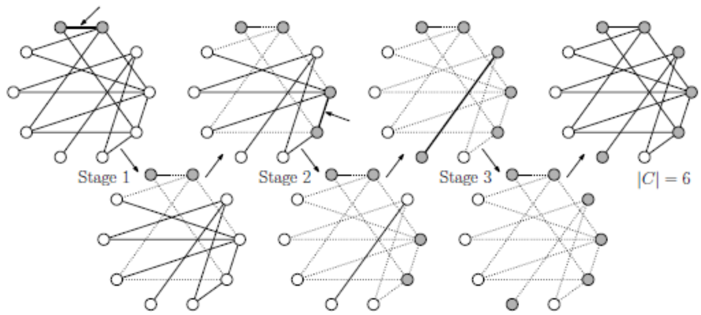
Vertex Cover

- There is an algorithm for vertex cover with a ratio bound of 2.
- This algorithm will be guaranteed to find a vertex cover whose size is at most twice that of the optimum.
- Recall that a vertex cover is a subset of vertices such that every edge in the graph is incident to at least one of these vertices.
- The vertex cover optimization problem is to find a vertex cover of minimum size



Vertex cover (optimal solution)

The 2-for-1 heuristic for VC



2-for-1 Approximation for VC

```
   two-for-one-VC(G=(V,E)) {
       C = empty
       while (E is nonempty) do {
(*)        let (u,v) be any edge of E
           add both u and v to C
           remove from E all edges incident to either u or v
       }
       return C
   }
```

7

2-for-1 Approximation for VC

Claim: The 2-for-1 approximation for VC achieves a performance ratio of 2.

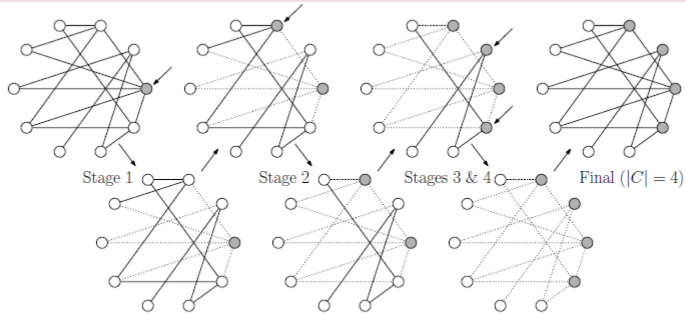
Proof: returns a vertex cover for G that is at most twice the size of the optimal vertex cover. Consider the set C output by two-for-one-VC(G). Let C^* be the optimum vertex cover. Let A be the set of edges selected by the line marked with "(∗)" in the code fragment. Because we add both endpoints of each edge of A to C, we have $|C| = 2|A|$. However, the optimum vertex cover C^* must contain at least one of these two vertices. Therefore, we have $|C^*| \geq |A|$. Therefore

$$|C| \;=\; 2|A| \;\leq\; 2|C^*| \qquad \Rightarrow \qquad \frac{|C|}{|C^*|} \;\leq\; 2$$

as desired.

8

The Greedy Heuristic



Greedy Approximation for VC

```
greedy-VC(G=(V,E)) {
    C = empty
    while (E is nonempty) do {
        let u be the vertex of maximum degree in G
        add u to C
        remove from E all edges incident to u
    }
    return C
}
```

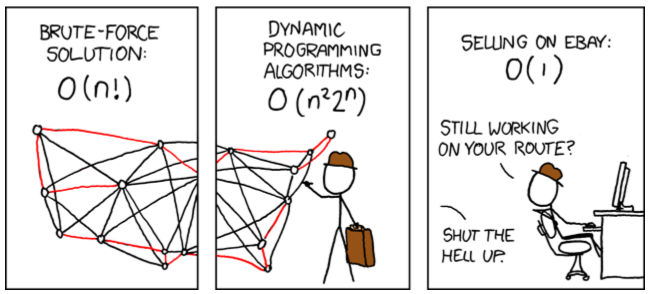

Reductions and Approximations

- Approximation factors are not generally preserved by transformations.
- Example: Recall that if V' is a vertex cover for G, then the complement vertex set, V \ V', is an independent set for G.
  - Suppose that G has n vertices, and a minimum vertex cover V' of size k. Then our heuristic is guaranteed to produce a vertex cover V'' that is of size at most 2k.
  - If we consider the complement set V \ V', we know that G has a maximum independent set of size n − k.
  - By complementing our approximation V \ V'' we have an "approximate" independent set of size n − 2k.
  - How good is this?
  - Performance ratio: $\rho(n,k) = \dfrac{n-k}{n-2k}$
- The problem is that this ratio may be arbitrarily large. For example, if n = 1001 and k = 500, then the ratio is 501/(1001 − 1000) = 500/1 = 500.

Traveling Salesman Problem

Traveling Salesperson Decision Problem (TSP) - Given a **complete** undirected graph with nonnegative edge weights, does there exist a cycle that visits all vertices and costs <= k?

- Let w(u, v) denote the weight on edge (u, v).
- Given a set of edges A forming a tour we define W(A) to be the sum of edge weights in A.



11

Traveling Salesman with Triangle Inequality

Traveling Salesperson Problem (TSP) - Given a complete undirected graph with nonnegative edge weights, and find a cycle that visits all vertices and is of minimum cost.

- Let w(u, v) denote the weight on edge (u, v).
- Given a set of edges A forming a tour we define W(A) to be the sum of edge weights in A.

Many of the applications of TSP, the edge weights satisfy a property called the triangle inequality

for all $u, v, x \in V$, $w(u, v) \le w(u, x) + w(x, v)$.

When the underlying cost function satisfies the triangle inequality there is an approximation algorithm for TSP with a ratio-bound of 2.

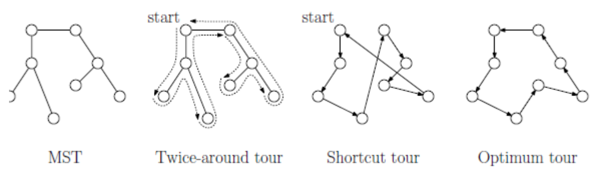
- The key insight is to observe that a TSP with one edge removed is just a spanning tree (not necessarily a MST).
- Cost of the minimum TSP tour is at least as large as the cost of the MST
- If we can find some way to convert the MST into a TSP tour while increasing its cost by at most a constant factor, then we will have an approximation for TSP
- If edge weights satisfy triangle inequality, this is possible.

12

Traveling Salesman with Triangle Inequality

TSP Approximation

```
approx-TSP(G=(V,E)) {
    T = minimum spanning tree for G
    r = any vertex
    H = list of vertices visited by a preorder walk of T starting at r
    return L
}
```



TSP Approximation

Approx-TSP Performance Ratio

Claim: Approx-TSP achieves a performance ratio of 2.

Proof: Let H denote the tour produced by this algorithm and let H^* be the optimum tour. Let T be the minimum spanning tree. As we said before, since we can remove any edge of H^* resulting in a spanning tree, and since T is the minimum cost spanning tree we have

$$W(T) \leq W(H^*).$$

Now observe that the twice around tour of T has cost $2 \cdot W(T)$, since every edge in T is hit twice. By the triangle inequality, when we short-cut an edge of T to form H we do not increase the cost of the tour, and so we have

$$W(H) \leq 2 \cdot W(T).$$

Combining these we have

$$W(H) \leq 2 \cdot W(T) \leq 2 \cdot W(H^*) \Rightarrow \frac{W(H)}{W(H^*)} \leq 2,$$

as desired.

Practice

1. Give an example of a graph for which the 2-for-1 VC algorithm yields a suboptimal solution.

2. We know that both the VERTEX COVER problem and the CLIQUE problem are NP-Complete, and as we showed previously, they are complementary in the sense that a minimum-size vertex cover is the complement of a maximum-size clique in the complementary graph.

Given the 2-for-1 VC algorithm, does the above relationship imply that there is a polynomial-time approximation algorithm with a constant approximation-ratio for the CLIQUE problem?