Approximate Algorithms

If we cannot find an **optimal** solution to an optimization problem, we might be able to **approximate** it.

Definition 1. *An approximate algorithm has a* **ratio bound** $\rho(n)$*, if for any input of size* n*, the optimal solution* $C^*(n)$ *and the algorithm solution* $C(n)$ *satisfy the relation:*

$$MAX\left[\frac{C(n)}{C^*(n)}, \frac{C^*(n)}{C(n)}\right] \leq \rho(n).$$

Vertex Cover

Given a graph $G = (V, E)$, a **vertex cover** of G is a set of vertices $V' \subseteq V$ such that each edge in E is adjacent to at least one vertex in V'.

The **vertex cover optimization problem** is to find a vertex cover of minimum size.

The problem is $\mathcal{NP}$-complete.

Approximation Algorithm

Approximate-Vertex-Cover(G)

1. $C \leftarrow \emptyset$

2. $E' \leftarrow E$

3. While $E' \neq \emptyset$ do

   3.1 Choose an arbitrary edge (u, v) in E'
   3.2 $C \leftarrow \{u, v\}$
   3.3 remove from E' every edge adjacent to u or v

4. return C

Analysis

Theorem 1. *The algorithm returns a vertex cover, and has a ratio bound of 2.*

Proof.

C is a vertex cover since the algorithm terminates when $E' = \emptyset$.

Let A be the set of edges chosen in line 3.1.

No two edges in A have a common vertex, thus any optimal vertex cover C^* satisfied

$$|C^*| > |A|$$

but $|C| = 2|A|$ thus,

$$\frac{|C|}{|C^*|} \leq 2$$

□

Traveling Salesman Problem

Given a complete graph $G = (V, E)$ with costs $c(u, v)$ on the edges, find a Hamiltonian cycle of minimum cost.

We approximate this problem in the case where the cost function $c()$ satisfied the **triangular inequality**: for all u, v and w,

$$c(u, w) \leq c(u, v) + c(v, w).$$

Approximate-TSP(G,c)

1. Compute a minimum spanning tree T of G.

2. Compute an Euler cycle of T starting at an arbitrary vertex a.

3. Compute the TSP by starting at vertex a, following the Euler path, skipping vertices that were already visited.

Analysis

Theorem 2. *The algorithm returns an Hamiltonian path of G, with approximate ratio bound of 2 on the total cost.*

Proof. Let H be the path computed by the algorithm, H^* an optimal path.

For a set of edges X, let $c(X) = \sum_{e \in X} c(e)$.

Since removing an edge from H^* gives a spanning tree

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

Let W be the Euler tour on T, it visits every edge twice, thus

$$c(W) = 2c(T) \leq 2c(H^*).$$

If W is not an Hamiltonian cycle, we remove vertices from W to get an Hamiltonian cycle.

Assume that W includes the segment $\ldots vuw \ldots$ and u already appears on the path.

We remove u and connect v directly to w, but

$$c(v, w) \leq c(v, u) + c(u, w)$$

so we don't increase the path cost.

Thus,

$$c(H) \leq c(W) = 2c(T) \leq 2c(H^*).$$

$\square$

Theorem 3. *The TSP problem with a cost function that satisfies the triangular inequality is NP-complete.*

Limits on Approximation

Theorem 4. *If $P \neq NP$ then there is no polynomial time approximation algorithm for the general TSP problem for any $\rho \geq 1$.*

Proof.

Assume that we have such an approximation algorithm, we'll use it to solve the Hamiltonian problem.

Given a graph $G = (V, E)$, let G' be a complete graph with cost function

$$c(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ \rho|V| + 1 & \text{otherwise} \end{cases}$$

If G has an Hamiltonian cycle, then G' has a TSP of cost $|V|$ (that cycle).

Any TSP solution in G' that is not an Hamiltonian cycle in G has cost at least

$$\rho|V| + 1 + |V| - 1 > \rho|V|.$$

Assume that we run an approximation algorithm AP with ratio bound ρ on G':

If G has an Hamiltonian path, AP will return that path.

If G does not have an Hamiltonian path, AP will return a TSP with cost more than $\rho|V|$.

Thus, AP solves the Hamiltonian path problem in G. □

Fully Polynomial-Time Approximation

The **error bound** of an approximation scheme is ϵ iff

$$\frac{|C - C^*|}{C^*} \leq \epsilon$$

A problem has a **fully polynomial-time approximation** scheme if for any $\epsilon > 0$ there is an algorithm for the problem with an ϵ ratio bound that is polynomial in both the problem size n and $1/\epsilon$.

The Subset-Sum Problem

The **subset-sum** decision problem: Given a set $S = \{x_1, \ldots, x_n\}$ of positive integers and an integer t, is there a subset of S that sums to t.

The subset-sum decision problem in $\mathcal{NP}$-complete.

The **subset-sum** optimization problem: Given a set $S = \{x_1, \ldots, x_n\}$ of positive integers and an integer t, find a subset of S with the largest sum less than t.

Exponential Algorithm

1. For $i = 0$ to n do

   1.1 Compute all the sums bounded by t from subsets of up to i elements of S.

Each iteration is polynomial in the number of sums in the previous iteration.

This algorithm is exponential since the number of different sums can grow exponentially in n.

Approximation Algorithm

1. For $i = 0$ to n do

   1.1 Compute all the sums bounded by t from subsets of up to i elements of S.

   1.2 Remove sums that are within $(1 - \epsilon/n)$ factor of other sums.

Run-Time

Let L_i be the collection of sums after the i-th iteration.

If $z, z' \in L_i$, then $z' > z(1 - \frac{\epsilon}{n})$, and all the elements are smaller than t.

Thus, there could be no more than k elements where

$$t(1 - \frac{\epsilon}{n})^k < 1$$

or

$$k = \frac{\log t}{-log(1 - \epsilon/n)} \leq \frac{n \log t}{\epsilon}$$

Thus, the run-time is polynomial in n and $1/\epsilon$.

How good is the approximation?

Let y be the optimal solution and z the approximate one.

Since we only removed elements from the list $z \leq y$.

Since whenever we removed an elements there was another elements in the list that was within $1 - \epsilon/n$ of the removed element

$$y(1 - \frac{\epsilon}{n})^n \leq z.$$

Thus,

$$(1 - \epsilon)y \leq z \leq y$$