35.Approximation Algorithms

Hsu, Lih-Hsing

An algorithm that returns near-optimal solutions is called an ***approximation algorithm***.

Performance bounds for approximation algorithms

We say an approximation algorithm for the problem has a ratio bound of $\rho(n)$ if for any input size n, the cost C of the solution produced by the approximation algorithm is within a factor of $\rho(n)$ of the C^* of the optimal solution:

$$\max\{\frac{C}{C^*}, \frac{C^*}{C}\} = \rho(n)$$

This definition applies for both minimization and maximization problems.

An ***approximation scheme*** for an optimization problem is an approximation algorithm that takes as input not only an instance of the problem, but also a value $\varepsilon > 0$ such that for any fixed ε, the scheme is a $(1+\varepsilon)$-approximation algorithm.

We say that an approximation scheme is a **polynomial-time approximation scheme** if for any fixed $\varepsilon > 0$, the scheme runs in time polynomial in the size n of its input instance.

We say that an approximation scheme is a ***polynomial-time approximation scheme*** if for any fixed $\varepsilon > 0$, the scheme runs in polynomial in size *n* of its input instance..

For example, the scheme might have a running time of $O((1/\varepsilon)^2 n^3)$.

We say that an approximation scheme is ***fully polynomial-time approximation scheme*** if its running time is polynomial both in $\frac{1}{\varepsilon}$ and in the size n of the input instance, where ε is the relative error bound for the scheme.

35.1 The vertex-cover problem

The ***vertex cover problem*** is to find a vertex cover of minimum size in a given undirected graph. We call such a vertex cover an ***optimal vertex cover***.

APPROX_VERTEX_COVER(G)

$C \leftarrow \phi$

$E' \leftarrow E(G)$

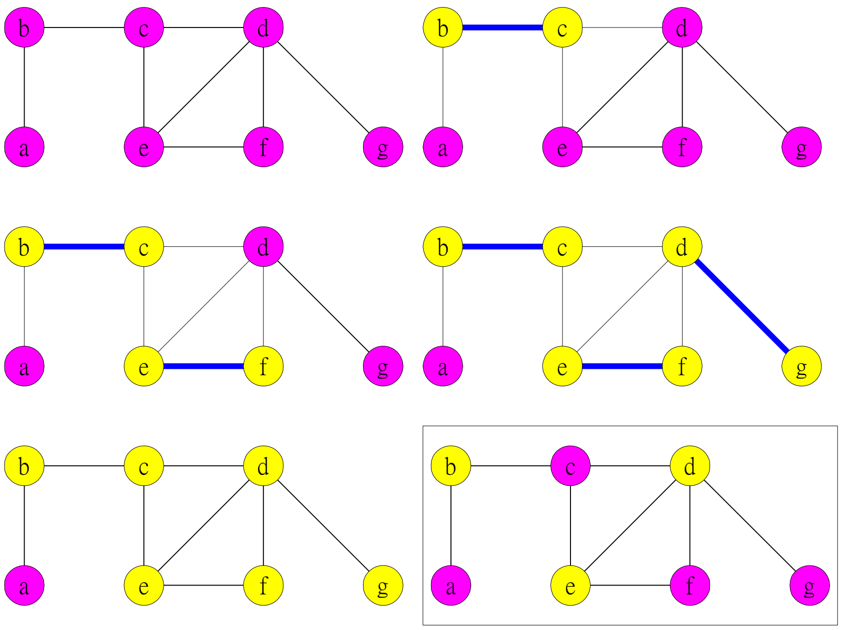
while $E' \neq \phi$

do let (u,v) be an arbitrary edge of E'

$C \leftarrow C \cup (u,v)$

remove from E' every edge incident on either u or v

return C



Complexity: O(E)

Theorem 35.1 APPROX_VERTEX_COVER has ratio bound of 2.

Proof.

Let A be the set of selected edges.

$|C| = 2|A|$

$|A| \le |C^*|$

$\Rightarrow |C| \le 2|C^*|$

35.2 The traveling salesman problem

triangle inequality

$$c(u,w) \le c(u,v) + c(v,w) \quad \forall u,v,w$$

35.2.1 The TSP with triangle inequality

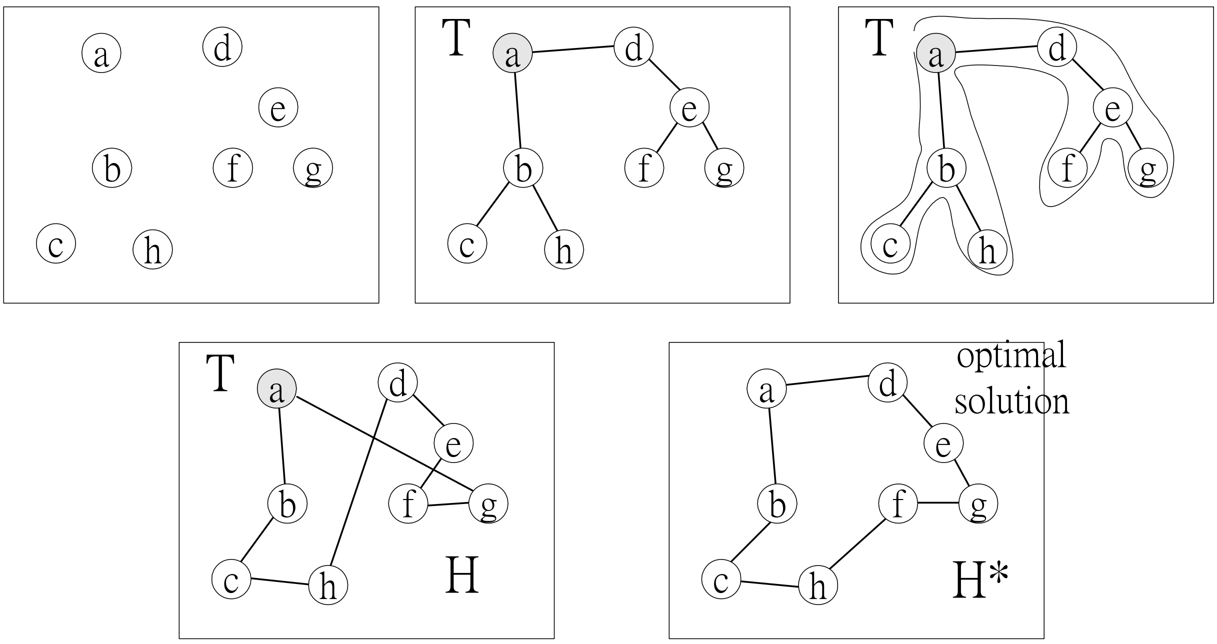
APPROX_TSP_TOUR(*G*,*c*)

1 Select a vertex $r \in V[G]$ to be a root vertex

2 grow a MST *T* for *G* from root *r* using MST_PRIM(*G*,*c*,*r*)

3 Let *L* be the list of vertices visited in a preorder walk of *T*.

4 **return** the hamiltonian cycle *H* that visit the vertices in the order *L*.



Theorem 35.2. APPROX_TSP_TOUR is an approximation algorithm with ratio bound of 2 for TSP with triangular inequality.

Proof.

$$c(T) \le c(H^*)$$

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

$$c(H) \le c(W)$$

$$\Rightarrow c(H) \le 2c(H^*)$$

35.2.2 The general TSP

Theorem 35.3 If $NP \neq P$ and $\rho \geq 1$, there is no polynomial time approximation algorithm with ratio bound ρ for some general TSP.

Proof.

Let $G = (V, E)$ be an instance of HC.

Let $G' = (V, E')$ be the complete graph of V.

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

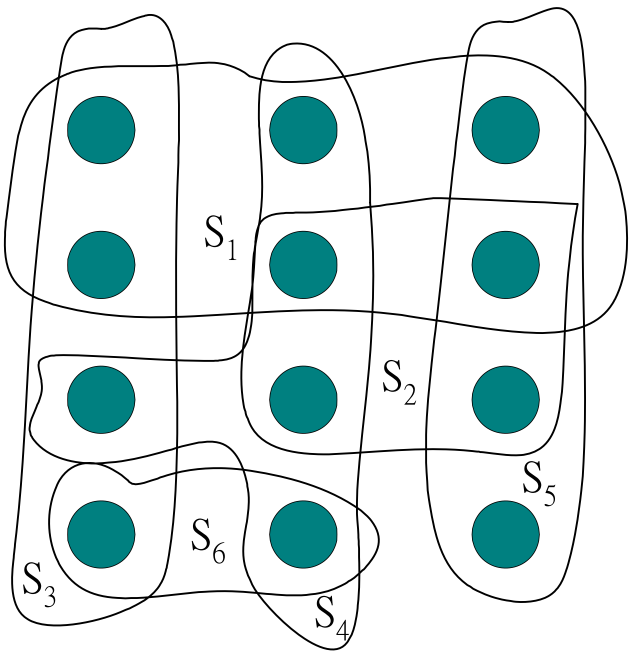
35.3 The set-covering problem

An instance (X,F) of the ***set-covering problem*** consists of a finite set X and a family F of elements of X, such that every element of X belongs to at least one subset in F: $X = \bigcup_{S \in F} S$. We say that a subset $S \in F$ ***covers*** its elements.

The problem is to find a minimum-size $C \subseteq F$ whose members covers all of X:

$$X = \bigcup_{S \in C} S. \quad (37.8)$$

We say that any C satisfies equation (37.8) ***covers*** X.



minimum-size set cover
$C^*=\{S_3,S_4,S_5\}$

Greedy set cover
$C^*=\{S_1,S_4,S_5,S_3\}$

GREEDY_SET_COVER(X,F)

$U \leftarrow X$

$C \leftarrow \phi$

while $C \neq \phi$

do select on $S \in F$ that maximize $|S \cap U|$

$U \leftarrow U - S$

$C \leftarrow C \cup \{S\}$

return C

running time: polynomial in $|X|$ and $|F|$.

Define $H_d = \sum_{i=1}^{d} \frac{1}{i}$

Theorem 35.4 GREEDY_SET_COVER has a ratio bound $H(max\{|S| | S \in F\})$

Proof.

- Define $c_x = \dfrac{1}{|S_i - (S_1 \cup S_2 \cup \ldots \cup S_{i-1})|}$ if

$x \in S_i - (S_1 \cup S_2 \cup \ldots \cup S_{i-1})$

Then $|C| = \sum_{x \in X} c_x \leq \sum_{S \in C^*} \sum_{x \in S} c_x$

- Suppose $\sum_{x \in S} c_x \leq H(|S|)$.

Then $|C| \leq \sum_{S \in C^*} H(|S|) \leq |C^*| H(\max\{|S| | S \in F\})$.

- Proof $\sum_{x \in S} c_x \leq H(|S|)$.

Define

$$u_i = |S - (S_1 \cup S_2 \cup \ldots \cup S_i)|$$

$$u_0 = |S|$$

Let k be the least index such that $u_k = 0$. Hence $S \subseteq \bigcup_{i=1}^{k} S_i$.

- $u_{i-1} \geq u_i$ and $u_{i-1} - u_i$ elements of S are covered for the first time by S_i for $i=1,2,\ldots,k$.

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1} - u_i) \frac{1}{|S_i - (S_1 \cup S_2 \cup \ldots \cup S_{i-1})|}.$$

- $|S_i - (S_1 \cup S_2 \cup \ldots S_{i-1})| \geq |S - (S_1 \cup S_2 \cup \ldots S_{i-1})|$
  $= u_{i-1}$

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} (u_{i-1} - u_i) \frac{1}{u_{i-1}}.$$

Computer Theory Lab.

NOTE: For any two positive integer $a>b$, we have

$$H(b) - H(a) = \sum_{i=a+1}^{b} \frac{1}{i} \geq (b-a)\frac{1}{b}$$

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} (H(u_{i-1}) - H(u_i))$$
$$= H(u_0) - H(u_k) = H(u_0) - H(0)$$
$$= H(u_0) = H(|S|).$$

Corollary 35.5. GREEDY_SET_COVER has a ratio bound of $(\ln|X|+1)$.

35.5 The subset-sum problem

An exponential-time algorithm

$$L=\{1,2,3,5,9\}$$

$$L+2=\{3,4,5,7,11\}$$

EXACT_SUBSET_SUM(S,t)

1 $n \leftarrow |S|$

2 $L_0 \leftarrow \langle 0 \rangle$

3 **for** $i \leftarrow 1$ **to** n

4 **do** $L_i \leftarrow MERGE\_LIST(L_i, L_{i-1} + x_i)$

5 remove from L_i every element

that is greater than t

6 **return** the largest element in L_n

$$S = \{1,4,5\}$$
$$P_1 = \{0,1\}$$
$$P_2 = \{0,1,4,5\}$$
$$P_3 = \{0,1,4,5,6,10\}$$

$$S = \{x_1, x_2, \ldots, x_k\}$$
$$P_i = P_{i-1} \cup (P_{i-1} + x_i)$$

A fully polynomial-time approximation scheme

Let $0 < \delta < 1$.

- To trim a list L by δ

  y is removed from L if $\exists z \le y$ still in L' such that

  $$\frac{y-z}{y} \le \delta \Leftrightarrow (1-\delta)y \le z \le y$$

Example: $\delta = 0.1$

$L = <10, 11, 12, 15, 20, 21, 22, 23, 24, 29>$

$L' = <10, 12, 15, 20, 23, 29>$

Trim(L, δ)

```
m ← |L|
L' ← < y_1 >
last ← y_1
for i ← 2 to m
  do if last < (1 − δ) y_i
    then append y_i onto the end of L'
      last ← y_i
return L'
```


APPROX_SUBSET_SUM(S, t, ε)

```
n ← |S|
L_0 ← < 0 >
for i ← 1 to n
  do Li ← MERGE_LIST(L_{i−1}, L_{i−1} + x_i)
     L_i ← TRIM(L_i, ε/n)
     remove from L_i every element
       that is greater than t
let z be the largest value of L
return z
```

$$L = <104, 102, 201, 101>$$

$$t = 308$$

$$\varepsilon = 0.20$$

$$\delta = \varepsilon / 4 = 0.05$$

2 $L_0 = <0>$

4 $L_1 = <0, 104>$

5 $L_1 = <0, 104>$

6 $L_1 = <0, 104>$

4 $L_2 = <0, 102, 104, 206>$

5 $L_2 = <0, 102, 206>$

6 $L_2 = <0, 102, 206>$

4 $L_3 = <0, 102, 201, 206, 303, 407>$

5 $L_3 = <0, 102, 201, 303, 407>$

6 $L_3 = <0, 102, 201, 303>$

4 $L_4 = <0, 101, 102, 201, 203, 302, 303, 404>$

5 $L_4 = <0, 101, 201, 302, 404>$

6 $L_4 = <0, 101, 201, 302>$

$$z = 302$$

$$z^* = 307 = 104 + 102 + 101$$

$$\varepsilon = 2\%$$

Theorem 35.8. APPROX_SUBSET_SUM is a fully polynomial approximation scheme for the subset-sum problem.

Proof.

- By induction on i, one can show that for every element y in P_i that is at most t, there is a $z \in L_i$ such that $(1 - \frac{\varepsilon}{n})^i y \le z \le y$.

$y^* \in P_n$ (optimal solution)

- $\Rightarrow (1-\frac{\varepsilon}{n})^n y^* \le z \le y^*$

$Note\ (1-\varepsilon) < (1-\frac{\varepsilon}{n})^n$

- $for\ \frac{d}{dn}(1-\frac{\varepsilon}{n})^n > 0 \Rightarrow (1-\frac{\varepsilon}{n})^n$ is increasing

$Thus\quad (1-\varepsilon) < (1-\frac{\varepsilon}{n})^n$

- $Hence\ (1-\varepsilon) y^* \le z \le y^*$
- Note that the successive element z and z' of L_i satisfy $\frac{z}{z'} > \frac{1}{1-\frac{\varepsilon}{n}}$. Thus the number of elements of L_i is at most $\log_{\frac{1}{1-\frac{\varepsilon}{n}}} t = \frac{\ln t}{-\ln(1-\frac{\varepsilon}{n})} \le \frac{n \ln t}{\varepsilon}$

- The algorithm is polynomial in the number *n* of the input values given, the number of bits ln *t* and $\frac{1}{\varepsilon}$.

- APPROX_SUBSET_SUM running time is $P(n, \ln t, \frac{1}{\varepsilon})$.