---------------------------- 35 ----------------------------

Approximation Algorithms

Two approaches to NP-hard problems:

(1) Exponential algorithms: (for small inputs)

- brute-force search
- branch-and-bound

(2) Near-optimal solutions: (polynomial time)

- approximation algorithms (with performance bounds)
- heuristic algorithms

Performance bounds (n is the input size)

ratio bound: $\begin{cases} C/C^* \le \rho(n) & \text{for minimization} \\ C^*/C \le \rho(n) & \text{for maximization} \end{cases}$

(Note that $\rho(n) \ge 1$.)

relative error bound: $\dfrac{|C - C^*|}{C^*} \le \varepsilon(n)$

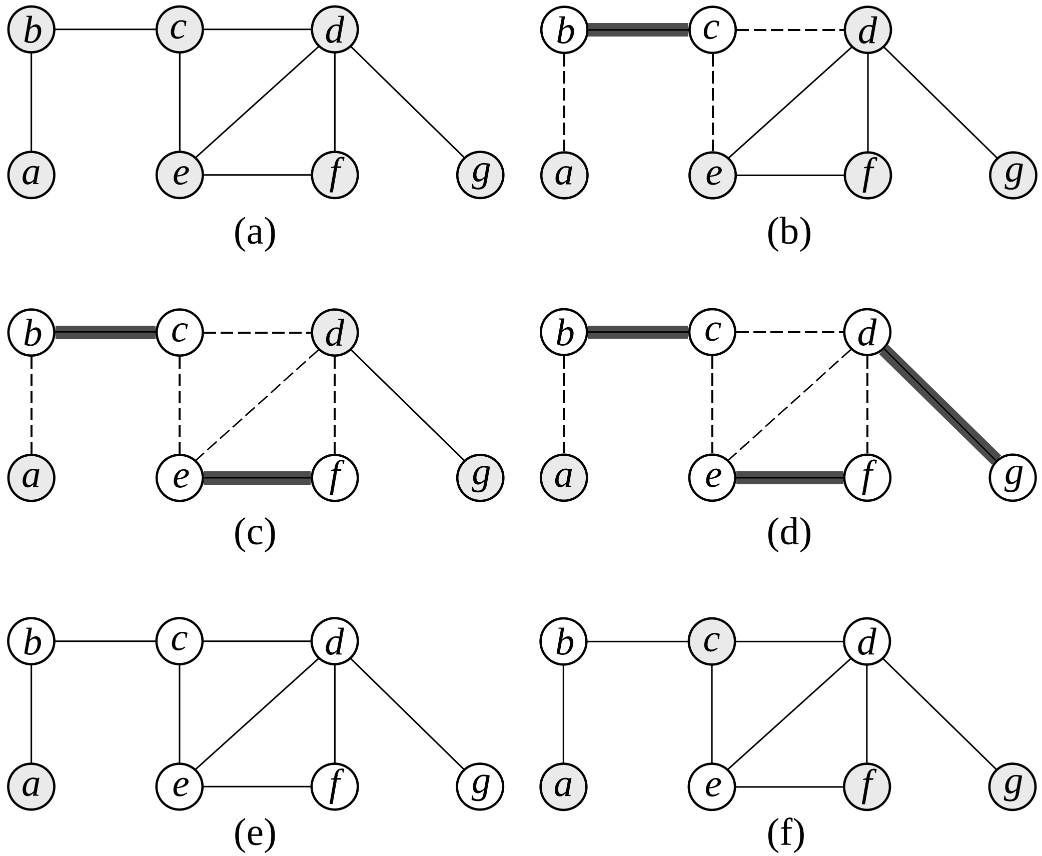
(for both minimization & maximization)

* For many problems, there are approximation algorithms with constant ratio bounds (relative error bounds), independent of n.

35.1 The vertex-cover problem

A ***vertex cover*** of an undirected graph $G=(V,E)$ is a subset C of V such that for each $(u, v)\in E$, either $u\in C$ or $v\in C$.

The ***vertex-cover problem*** is to find for G a vertex cover of minimum size. (an NP-hard problem)



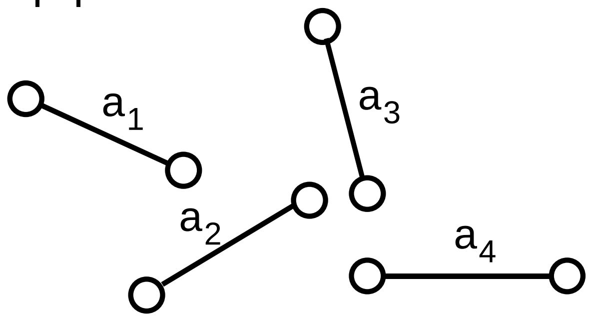
* (e): a vertex cover $C=\{b, c, d, e, f, g\}$
  (f): the optimal vertex cover $C^*=\{b, d, e\}$

APPROX-VERTEX-COVER(G)

```
1  C ← Ø
2  E′ ← E[G]
3  while E′ ≠ Ø
4       do let (u, v) be an arbitrary edge of E′
5            C ← C ∪ {u, v}
6            remove from E′ every edge incident
                                on either u or v
7  return C
```

* Time: $O(E)$

Theorem 35.1: Approx-Vertex-Cover has $\rho(n) = 2$.
Proof: Let A be the set of edges picked in Line 4. Since no two edges in A share an endpoint, we have $|C|=2|A|$.



Let C^* be an optimal cover. C^* should cover A. That is, C^* should include at least one endpoint of each edge in A. Since no two edges in A share an endpoint, we have $|C^*|\geq|A|$ $(=|C|/2)$ and thus $C/C^*\leq 2$. Q.E.D.

35.2 The traveling-salesman problem (NP-C)

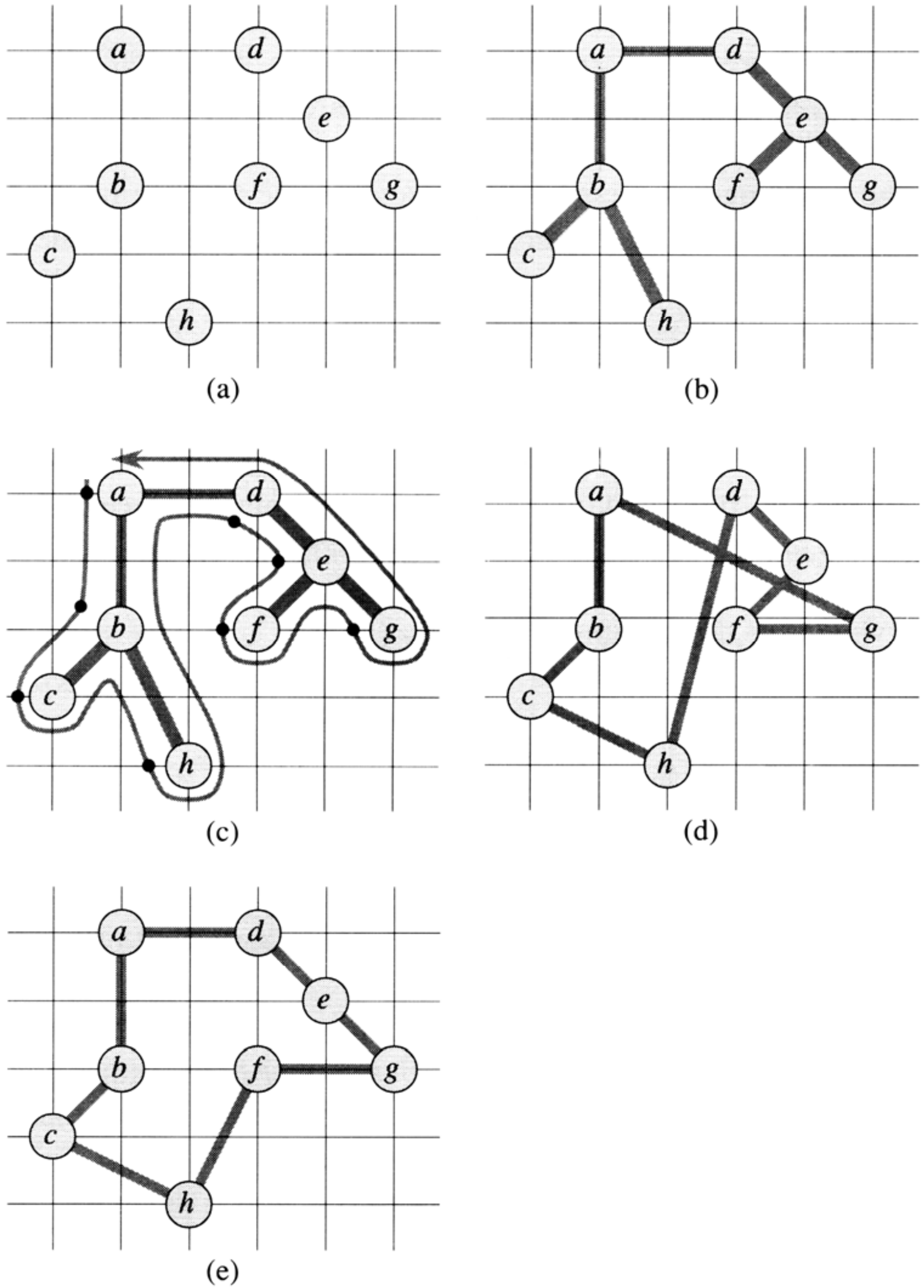
35.2.1 The Euclidean TSP problem (NP-C)

The ***Euclidean traveling-salesman problem*** is to find in a *complete* weighted undirected graph $G=(V,E)$ a hamiltonian cycle (a tour) with minimum cost. The edges weights $c(u,v)$ are nonnegative integers. And, the weight function satisfies the following ***triangle inequality:***

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

APPROX-TSP-TOUR(G, c)

```
select a vertex r ∈ G.V to be a "root" vertex
compute a minimum spanning tree T for G from root r
      using MST-PRIM(G, c, r)
let H be a list of vertices, ordered according to when
      they are first visited in a preorder tree walk of T
return the hamiltonian cycle H
```



* (b): T: a minimum spanning tree T
(c): W: a full walk of T
(d): H: a tour of length 19.074
(e): H^*: an optimal tour of length 14.715

Time: $O(E)=O(V^2)$

Theorem 35.2: Approx-TSP-Tour has $\rho(n) = 2$.
Proof: Let T be a minimum spanning tree. Deleting any edge from H^*, we can obtain a spanning tree. Thus, $|T|\leq|H^*|$.

A ***full walk***, denoted by W, of T lists the vertices when they are first visited and also whenever they are return to after a visit to a subtree. In our example,

$$W=(a, b, c, b, h, b, a, d, e, f, e, g, e, d, a).$$

Clearly, $|W|=2|T|$. Thus, $|W|\leq 2|H^*|$.

Note that W is not a tour. It visits a vertex more than once. However, by triangle inequality, we can delete unnecessary visits to a vertex without increasing the cost to obtain H. (In our example, $H=(a, b, c, h, d, e, f, g)$.) Thus, $|H|\leq|W|\leq 2|H^*|$.
Q.E.D.

35.2.2 The general TSP problem

Without triangle inequality, an approximation algorithm with constant ratio bound does not exist unless P = NP.

35.5 The subset-sum problem

Approximation scheme: an approximation algorithm takes as input not only an instance of the problem, but also a constant relative error bound $\varepsilon > 0$.

Polynomial-time approximation scheme: an approximation scheme runs in $O(n^k)$ time, where k is a constant. (e.g., $O(n^{3/\varepsilon})$.)

Fully polynomial-time approximation scheme: an approximation scheme runs in $O((1/\varepsilon)^c n^k)$ time, c and k are constants. (e.g., $O((1/\varepsilon)^2 n^3)$ time)

***The subset-sum problem*:**

Decision version: Given a set S of positive integers and an integer t, determine whether there is a subset of S that adds up exactly to the target t.

Optimalization version: find a subset of S whose sum is as large as possible but not larger than t.

An exponential-time algorithm

```
EXACT-SUBSET-SUM(S, t)
n ← |S|
L_0 ← ⟨0⟩
for i ← 1 to n
     do L_i ← MERGE-LISTS(L_{i-1}, L_{i-1} + x_i)
          remove from L_i every element that is greater than t
return the largest element in L_n
```

Example: Let S=(2, 2, 14, 3) and t = 15.

$\Rightarrow L_0=<0>$

$<0>\cup(<0>+2)=<0,2>$ $\Rightarrow L_1=<0,2>$

$<0,2>\cup(<0,2>+2)=<0,2,2,4> \quad \Rightarrow L_2=<0,2,4>$

$L_2\cup(L_2+14)=<0,2,4,14,16,18> \Rightarrow L_3=<0,2,4,14>$

$L_3\cup(L_3+3)=<0,2,3,4,5,7,14,17>$

$\Rightarrow L_4=<0,2,3,4,5,7,14>$

Time: $\sum_{0\le i\le n-1} 2|L_i| = O(2^n)$. Note that $|L_i|=O(2^i)$.

* In case t is polynomial in n, we have $|L_i|=O(t)$. Thus, the algorithm performs in polynomial time.

* In case all integers in S are bounded by a polynomial in n, the algorithm also performs in polynomial time.

A fully polynomial-time approximation scheme

To ***trim*** *a list L by* δ is to remove as many elements from L as possible, in such a way that if L' is the result of trimming L, then for every element y that was removed from L, there is an element $z\le y$ still in L' such that

$$y \le z(1+\delta) \qquad (y-z \le \delta z)$$

(We can think of "z representing y" in L'.)

Example:

Let $L = (10, \underline{11}, 12, 15, 20, \underline{21}, \underline{22}, 23, \underline{24}, 29)$.
If $\delta = 0.1$, we have
$L' = (10, 12, 15, 20, 23, 29)$.

Let $L = (y_1, y_2, \ldots, y_m)$. The following procedure trims L in $O(m)$ time.

TRIM(L, δ)

$m \leftarrow |L|$
$L' \leftarrow \langle y_1 \rangle$
$last \leftarrow y_1$
for $i \leftarrow 2$ **to** m
  do if $y_i > last \cdot (1+\delta)$
    then append y_i onto the end of L'
      $last \leftarrow y_i$
return L'

An approximation scheme ($0 < \varepsilon < 1$)

APPROX-SUBSET-SUM(S, t, ϵ)

$n \leftarrow |S|$
$L_0 \leftarrow \langle 0 \rangle$
for $i \leftarrow 1$ **to** n
  do $L_i \leftarrow$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
    $L_i \leftarrow$ TRIM$(L_i, \epsilon/n)$
    remove from L_i every element that is greater than t
let z^* be the largest value in L_n
return z^*

Example: Let S=<104, 102, 201, 101>, t=308, and ε = 0.2. We have $\delta = \varepsilon/4$ =0.05 and

line 2:	L_0	=	$\langle 0 \rangle$,
line 4:	L_1	=	$\langle 0, 104 \rangle$,
line 5:	L_1	=	$\langle 0, 104 \rangle$,
line 6:	L_1	=	$\langle 0, 104 \rangle$,
line 4:	L_2	=	$\langle 0, 102, 104, 206 \rangle$,
line 5:	L_2	=	$\langle 0, 102, 206 \rangle$,
line 6:	L_2	=	$\langle 0, 102, 206 \rangle$,
line 4:	L_3	=	$\langle 0, 102, 201, 206, 303, 407 \rangle$,
line 5:	L_3	=	$\langle 0, 102, 201, 303, 407 \rangle$,
line 6:	L_3	=	$\langle 0, 102, 201, 303 \rangle$,
line 4:	L_4	=	$\langle 0, 101, 102, 201, 203, 302, 303, 404 \rangle$,
line 5:	L_4	=	$\langle 0, 101, 201, 302, 404 \rangle$,
line 6:	L_4	=	$\langle 0, 101, 201, 302 \rangle$.

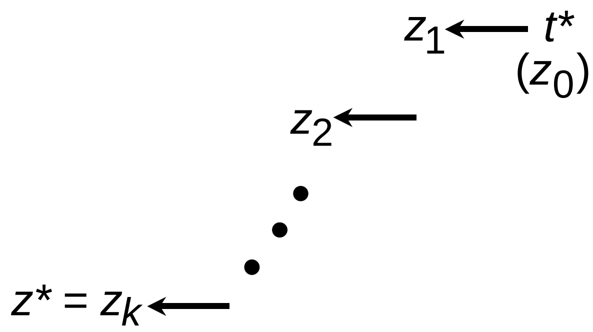
The answer is z^*=302, which is well within ε = 20%. (The optimal answer is 307 (=104+102+101).)

Theorem 35.8 Approx-Subset-Sum is a fully polynomial-time approximation scheme.

Proof:

(a) Clearly, the answer is legal. (not larger than t and being the sum of a subset).

(b) relative error bound is within ε:



$$t^* - z^* = \sum_{1 \le i \le k} (z_{i-1} - z_i)$$
$$\le \sum_{1 \le i \le k} \delta \times z_i$$
$$\le k\delta t^*$$
$$\le n\delta t^*$$
$$\le \varepsilon t^*$$

(c) **fully polynomial-time:**

L_i: $y_1 = 0, y_2, y_3, \ldots, y_k$

Since $y_2 \ge 1$ and $y_i > y_{i-1} \times (1+\delta)$, we have

$$y_k > (1+\delta)^{k-2}.$$

Since $y_k \le t$, we have

k-2 $\le \log_{1+\delta} t$. Thus,

$$|L_i| \le \log_{1+\delta} t + 2$$
$$= \frac{\ln t}{\ln(1+\delta)} + 2$$
$$\le \frac{(1+\delta)\ln t}{\delta} + 2$$

(by (3.17), $\frac{x}{1+x} \le \ln(1+x)$ for $x > -1$)

$$\le \frac{n(1+\frac{\varepsilon}{n})\ln t}{\varepsilon} + 2$$
$$\le \frac{2n\ln t}{\varepsilon} + 2 \quad \text{(by } \frac{\varepsilon}{n} < 1\text{)}$$

$$\text{Time} = O\Big(\sum_{0\le i\le n-1} |L_i|\Big)$$
$$= O\Big(n\Big(\frac{2n\ln t}{\varepsilon} + 2\Big)\Big)$$
$$= O\Big(\frac{1}{\varepsilon} n^2 \log t\Big)$$

Q.E.D.

Homework: Ex. 35.1-4, 35.5-4.

Differences in the 3rd Edition

Approximation scheme: (1st)
(defined by relative error bound)

Given a parameter: α (0.6)
Goal: $\varepsilon = \alpha$ (0.6)
(or simply "Given ε")
(set $\delta = \varepsilon / n$) (0.06 for n = 10)

Approximation scheme: (2nd, 3rd)
Approximation Scheme:
(defined by ratio bound)

Given a parameter: α (0.6)
Goal: $\rho = 1 + \alpha$ (1.6)
(for MAX, set $\delta = \alpha / 2n$) (0.03 for n = 10)
(In the textbook, ε is used to denote α)

```
APPROX-SUBSET-SUM(S, t, ϵ)
n ← |S|
L_0 ← ⟨0⟩
for i ← 1 to n
    do L_i ← MERGE-LISTS(L_{i-1}, L_{i-1} + x_i)
       L_i ← TRIM(L_i, ϵ/2n)
       remove from L_i every element that is greater than t
let z* be the largest value in L_n
return z*
```