Approximation Algorithms for NP-Complete Problems

1. Performance ratios
2. Some polynomial approximation algorithms
   a. Vertex-Cover
   b. TSP
   c. Scheduling
   d. Set-Cover
   e. Maximum Set-Cover
   f. Independent-Set
   g. 3-Coloring
   h. Subset-Sum
3. Weighed Independent Set and Vertex Cover

1. Definitions

Definition 1 *An approximation algorithm has approximation ratio $\rho(n)$, if for any input of size n one has:*

$$\max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \le \rho(n),$$

where C and C^* are the costs of the approximated and the optimal solution, respectively.

An algorithm with approximation ratio ρ is sometimes called ρ-approximation algorithm.

Usually there is a trade-off between the running time of an algorithm and its approximation quality.

Definition 2 *An approximation scheme for an optimization problem is an approximation algorithm that takes as input an instance of the problem and a number $\epsilon(n) > 0$ and returns a solution within the approximation rate $1 + \epsilon$.*

An approximation scheme is called fully polynomial-time approx. scheme if it is an approximation scheme and its running time is polynomial both in $1/\epsilon$ and in size n of the input instance.

2a. The VERTEX-COVER Problem

Instance: An undirected graph $G = (V, E)$.
Problem: Find a vertex cover of minimum size.

Algorithm 1 APPROX-VERTEX-COVER(G);

$C := \emptyset$
$E' := E$
while $E' \neq \emptyset$
do $C := C \cup \{u, v\}$ /* here $(u, v) \in E'$ */
$E' := E' - \{$edges of E' incident to u or $v\}$
return C

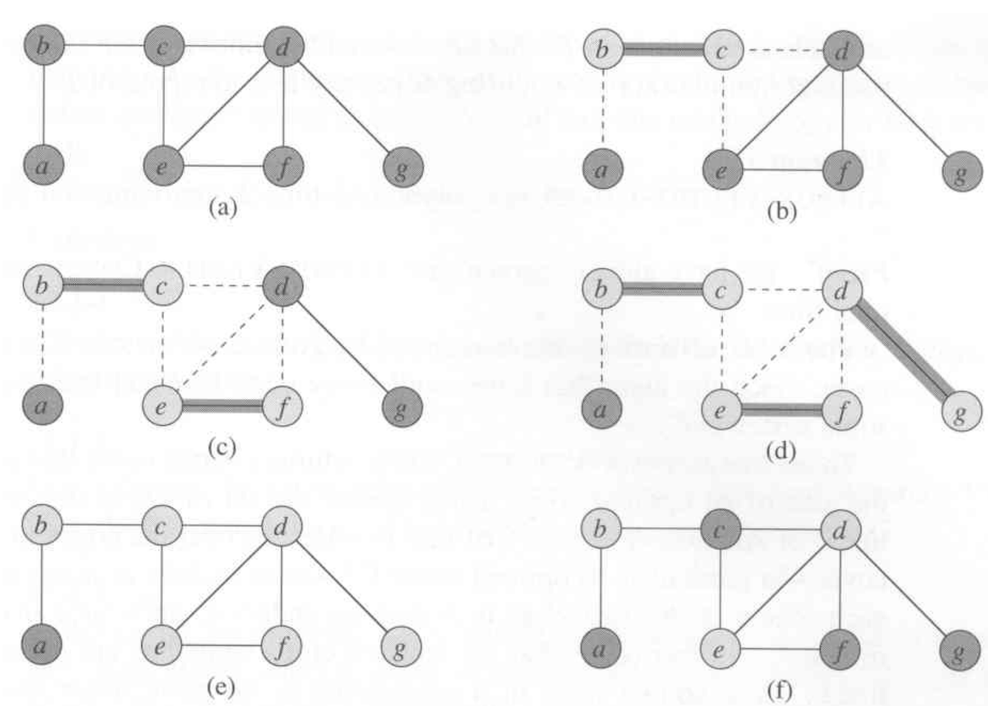

Figure 1: APPROX-VERTEX-COVER in action

Theorem 1 *The* APPROX-VERTEX-COVER *is a polynomial-time 2-approximation algorithm.*

Proof. The set of vertices C constructed by the algorithm is a vertex cover. Let C^* be a minimum vertex cover.

Let A be the set of edges that were picked by the algorithm. Then

$$|C| = 2 \cdot |A|.$$

Since the edges in A are independent,

$$|A| \leq |C^*|.$$

Therefore:

$$|C| \leq 2 \cdot |C^*|.$$

2b. The TSP Problem

Instance: A complete graph $G = (V, E)$ and a weight function $c : E \to \mathbf{R}^{\geq 0}$.
Problem: Find a Hamilton cycle in G of minimum weight.

For $A \subseteq E$ define

$$c(A) = \sum_{(u,v) \in A} c(u, v).$$

We assume the weights satisfy the triangle inequality:

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

for all $u, v, w \in V$.

Remark 1 *The* TSP *problem is NP-complete even under this assumption.*

Algorithm 2 APPROX-TSP(G, c);

1. Choose a vertex $v \in V$.
2. Construct a minimum spanning tree T for G rooted in v (use, e.g., MST-PRIM algorithm).
3. Construct the pre-order traversal W of T.
4. Construct a Hamilton cycle that visits the vertices in order W.

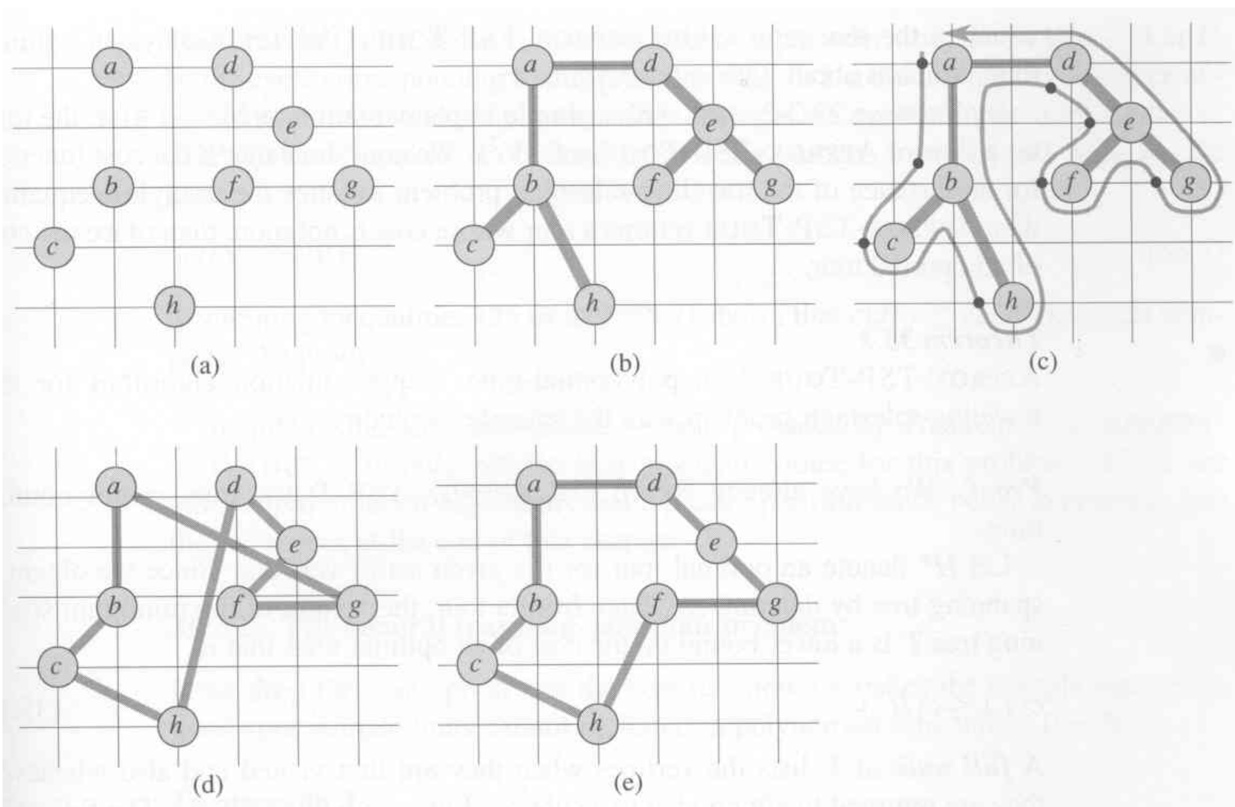

Figure 2: APPROX-TSP in action

Theorem 2 *The* APPROX-TSP *is a polynomial-time 2-approx. algorithm for the TSP problem with the triangle inequality.*

Proof. Let H^* be an optimal Hamilton cycle.
We construct a cycle H with $c(H) \leq 2 \cdot c(H^*)$.

Since T is a minimal spanning tree, one has:

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

We construct a list L of vertices taken in the same order as in the MST-PRIM algorithm and get a walk W around T.

Since W goes through every edge twice, we get:

$$c(W) = 2 \cdot c(T),$$

which implies

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

The walk W is, however, not Hamiltonian.

We go through the list L and delete from W the vertices which have already been visited.

This way we obtain a Hamilton cycle H. The triangle inequality provides

$$c(H) \leq c(W).$$

Therefore,

$$c(H) \leq 2 \cdot c(H^*).$$

The TSP problem for an arbitrary weight function c is intractable.

Theorem 3 *Let $p \geq 1$. If P$\neq$NP, then there is no polynomial-time p-approximation algorithm for the* TSP *problem.*

Proof. W.l.o.g. assume $p \in I\!N$.
Suppose that for some $p \geq 1$ there exists a polynomial p-approx. algorithm A.

We show how the algorithm A can be applied to solve the HC problem in polynomial time.

Let $G = (V, E)$ be an instance for the HC problem. Construct a complete graph $G' = (V, E')$ with the following weight function:

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

G is Hamiltonian $\Rightarrow$ G' contains a Ham. cycle of weight $|V|$.
G is not Hamiltonian $\Rightarrow$ G' has a Ham. cycle of weight

$$\geq (p|V| + 1) + (|V| - 1) > p|V|.$$

We apply A to the instance (G', c). Then A constructs a cycle of length no more than p times longer than the optimal one. Hence:
G is Hamiltonian $\Rightarrow$ A constructs a cycle in G of length $\leq p|V|$.
G is not Hamiltonian $\Rightarrow$ A constructs a cycle in G' of length $> p|V|$.

Comparing the length of the cycle in G' with $p|V|$ we can recognize whether G is Hamiltonian or not in polynomial time, so P=NP. $\square$

2c. Scheduling

Let $J_1, \ldots, J_n$ be tasks to be performed on m identical processors $M_1, \ldots, M_m$.

Assumptions:

- The task J_j has duration $p_j > 0$ and must not be interrupted.
- Each processor M_i can execute only one task in a time.

The problem: construct a schedule

$$\Sigma : \{J_j\}_{j=1}^n \mapsto \{M_i\}_{i=1}^m$$

that provides a fastest completion of all tasks.

Theorem 4 (Graham '66). *There exists an* $(2-1/m)$ *approximation scheduling algorithm.*

Proof:
Assume the tasks are listed in some order.

Heuristics G: as soon as some processor becomes free, assign to it the next task from the list.

Denote by s_j and e_j the start- and end-times of the tasks J_j in the heuristics G. Let J_k be the task completed last. Then no processor is free at time s_k. This implies $m \cdot s_k$ does not exceed the total duration of all other tasks, i.e.

$$m \cdot s_k \leq \sum_{j \neq k} p_j. \tag{1}$$

For the running time C_n^* of the optimal schedule one has:

$$C_n^* \geq \frac{1}{m} \cdot \sum_{j=1}^{n} p_j. \tag{2}$$

$$C_n^* \geq p_k \tag{3}$$

The inequality (2) follows from the fact that if there exists a schedule of time complexity $C < \frac{1}{m} \cdot \Sigma_{j=1}^n p_j$ then for the total duration P of all tasks one has $P = \Sigma_{j=1}^n p_j \leq mC < \Sigma_{j=1}^n p_j$, which is a contradiction.

The heuristics G provides:

$$\begin{aligned}
C_n^G = e_k &= s_k + p_k \\
&\leq \frac{1}{m} \cdot \sum_{j \neq k} p_j + p_k && \text{by (1)} \\
&= \frac{1}{m} \cdot \sum_{j=1}^{n} p_j + \left(1 - \frac{1}{m}\right) p_k \\
&\leq C_n^* + \left(1 - \frac{1}{m}\right) C_n^* && \text{by (2) \& (3)} \\
&= \left(2 - \frac{1}{m}\right) C_n^*. \qquad \square
\end{aligned}$$

A better approximation can be obtained by following the **LPT Rule** (Longest Processing Time):
Sort the tasks w.r.t. p_i in non-increasing order and assign the next task from the sorted list to a processor that becomes free earliest.

Theorem 5 *It holds*

$$C_n^{\mathit{LPT}} \leq (3/2 - 1/(2m))\, C_n^*.$$

Proof:
Let J_k be the task completed last. Since time s_k all processors are busy, there is a set S of m tasks that are processed at that time. For any $J_j \in S$ one has $p_j \geq p_k$ (the LPT heuristics).

Now, if $p_k > (1/2)C_n^*$, then $\exists\ m+1$ tasks of length at least $(1/2)C_n^*$ each, which is a contradiction (no schedule just for these tasks cannot be completed in time C_n^*).

Hence, $p_k \leq (1/2)C_n^*$. One has

$$\begin{aligned}
C_n^{\mathsf{LPT}} &= s_k + p_k \\
&\leq \frac{1}{m} \cdot \sum_{j \neq k} p_j + p_k & \text{by (1)} \\
&= \frac{1}{m} \cdot \sum_{j=1}^{n} p_j + \left(1 - \frac{1}{m}\right) p_k \\
&\leq C_n^* + \left(1 - \frac{1}{m}\right) (1/2) C_n^* \\
&\leq \left(\frac{3}{2} - \frac{1}{2m}\right) C_n^*. \qquad \square
\end{aligned}$$

A deeper analysis leads to even better bound for the LPT heuristics.

Theorem 6 *It holds:*

$$C_n^{LPT} \leq (4/3 - 1/(3m))C_n^*.$$

Proof:
Let J_k be the task completed last in the LPT schedule S_n.

Assume $p_k \leq C_n^*/3$. Then, similarly to the proof of the last theorem,

$$\begin{aligned} C_n^{\text{LPT}} &\leq (1/m) \sum_{j=1}^{n} p_j + (1 - 1/m)p_k \\ &\leq C_n^* + (1 - 1/m)C_n^*/3 \\ &= (4/3 - 1/(3m))C_n^*. \end{aligned}$$

Assume $p_k > C_n^*/3$. Construct the reduced schedule S_k for the tasks $J_1, \ldots, J_k$ by dropping the tasks $J_{k+1}, \ldots, J_n$ from S_n. Then $C_k^{\text{LPT}} = C_n^{\text{LPT}}$ (definition of J_k).

Each processor got at most 2 tasks to perform in the optimal schedule C_k^* for $J_1, \ldots, J_k$ (if some got 3, say p_i, p_j, p_l, then $p_i + p_j + p_l \geq 3p_k > C_n^* \geq C_k^*$).

Therefore $k \leq 2m$. In this case the schedule S_k for $J_1, \ldots, J_k$ provided by the LPT heuristics is optimal. But then S_n is optimal for $J_1, \ldots, J_n$ because

$$C_n^* \geq C_k^* = C_k^{\text{LPT}} = C_n^{\text{LPT}} \geq C_n^*. \qquad \square$$

2d. The SET-COVER Problem

Instance: A finite set X and a collection of its subsets $\mathcal{F}$ such that $\bigcup_{S \in \mathcal{F}} S = X$.

Problem: Find a minimum set $C \subseteq \mathcal{F}$ that covers X.

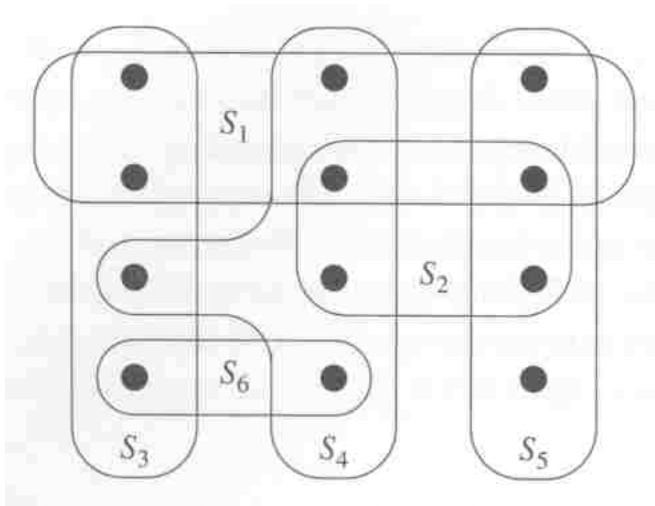

Figure 3: An instance of the SET-COVER problem

Remark 2 *The* SET-COVER *problem is NPC (Reduction from* VC *problem. Both problems can be formulated as vertex-covering problems in bipartite graphs. The bipartition sets for* SET-COVER *graph are formed by the sets X and $\mathcal{F}$. The bipartition sets for* VC *graph for $G = (V, E)$ are formed by the sets V and E).*

Algorithm 3 GREEDY-SET-COVER$(X, \mathcal{F})$;

$U := X$
$C := \emptyset$
while $U \neq \emptyset$
  do Choose $S \in \mathcal{F}$ with $|S \cap U| \to \max$
    $U := U - S$
    $C := C \cup \{S\}$
return C

Since the **while**-loop is executed at most $\min\{|X|, |\mathcal{F}|\}$ times and each its iteration requires $O(|X|\cdot|\mathcal{F}|)$ computations, the running time of GREEDY-SET-COVER is $O(|X| \cdot |\mathcal{F}| \cdot \min\{|X|, |\mathcal{F}|\})$.

Theorem 7 *The* $\textsc{Greedy-Set-Cover}$ *is a polynomial time* $\rho(n)$*-approximation algorithm, where* $\rho(n) = H(\max\{|S| \mid S \in \mathcal{F}\})$ *and* $H(d) = \sum_{i=1}^{d}(1/i)$.

Proof. Let C be the set cover constructed by the $\textsc{Greedy-Set-Cover}$ algorithm and let C^* be a minimum cover.

Let S_i be the set chosen at the i-th execution of the **while** -loop. Furthermore, let $x \in X$ be covered for the first time by S_i. We set the weight c_x of x as follows:

$$c_x = \frac{1}{|S_i - (S_1 \cup \cdots \cup S_{i-1})|}.$$

One has:

$$|C| = \sum_{x \in X} c_x \le \sum_{S \in C^*} \sum_{x \in S} c_x. \tag{4}$$

We will show later that for any $S \in \mathcal{F}$

$$\sum_{x \in S} c_x \le H(|S|). \tag{5}$$

From (4) and (5) one gets:

$$|C| \le \sum_{S \in C^*} H(|S|) \le |C^*| \cdot H(\max\{|S| \mid S \in \mathcal{F}\}),$$

which completes the proof of the theorem.

To show (5) we define for a fixed $S \in \mathcal{F}$ and $i \le |C|$

$$u_i = |S - (S_1 \cup \cdots \cup S_i)|,$$

that is, $\#$ of elements of S which are not covered by $S_1, \ldots, S_i$.

Let $u_0 = |S|$ and k be the minimum index such that $u_k = 0$. Then $u_{i-1} \geq u_i$ and $u_{i-1} - u_i$ elements of S are covered for the first time by S_i for $i = 1, \ldots, k$.

One has:

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

Since for any $S \in \mathcal{F} \setminus \{S_1, \ldots, S_{i-1}\}$

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

due to the greedy choice of S_i, we get:

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

Since for any integers a, b with $a < b$ it holds:

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

we get a telescopic sum:

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

which completes the proof of (5). □

Corollary 1 *Since $H(d) \leq \ln d + 1$, the* GREEDY-SET-COVER *algorithm has the approximation rate* $(\ln |X| + 1)$.

2e. The MAXIMUM SET-COVER-Problem

Instance: A finite set X, a weight function $w : X \mapsto \mathbf{R}$, a collection $\mathcal{F}$ of subsets of X and $k \in I\!N$.
Problem: Find a collection $C \subseteq \mathcal{F}$ of subsets with $|C| = k$ such that $\sum_{x \in C} w(x)$ is maximum.

Algorithm 4 MAXIMUM-COVER$(X, \mathcal{F}, w)$;

$U := X$
$C := \emptyset$
for $i := 1$ **to** k **do**
  Choose $S \in F$ with $w(S \cap U) \to \max$
  $U := U - S$
  $C := C \cup S$
return C

Theorem 8 *The* MAXIMUM-COVER *is a polynomial time* $(1 - 1/e)^{-1}$*-approximation algorithm* $((1 - 1/e)^{-1} \approx 1.58)$*.*

Proof.
Let C be the set constructed by the algorithm and let C^* be the optimal solution. Furthermore, let S_i be the set chosen at step i of the algorithm.

The greedy choice of S_l implies:

$$w(\bigcup_{i=1}^{l} S_i) - w(\bigcup_{i=1}^{l-1} S_i) \geq \frac{w(C^*) - w(\bigcup_{i=1}^{l-1} S_i)}{k}, \quad l = 1, \ldots, k. \quad (6)$$

Indeed, for any subset $A \subseteq X$, there exists a set $S \in C^*$ with

$$w(S - A) \geq w(C^* - A)/k$$

(if for any $S \in C^*$ the inverse inequality is satisfied, then $\sum_{S \in C^*} w(S - A) < k \cdot w(C^* - A)/k = w(C^* - A)$, which is a contradiction since not the whole part of $w(C^*)$ outside of A is covered).

Note that $w(C^* - A) \geq w(C^*) - w(A)$ and apply this observation for $A = \cup_{i=1}^{l-1} S_i$. By the greedy choice of S_l one has $w(S_l - \cup_{i=1}^{l-1} S_i) \geq w(S - \cup_{i=1}^{l-1} S_i)$ for any $S \subseteq X$. So,

$$\begin{aligned} w(\bigcup_{i=1}^{l} S_i) - w(\bigcup_{i=1}^{l-1} S_i) &= w(S_l - \bigcup_{i=1}^{l-1} S_i) \\ &\geq w(S - \bigcup_{i=1}^{l-1} S_i) \\ &\geq w(C^* - \bigcup_{i=1}^{l-1} S_i)/k \\ &\geq \frac{w(C^*) - w(\bigcup_{i=1}^{l-1} S_i)}{k}. \end{aligned}$$

We show by induction on l:

$$w(\bigcup_{i=1}^{l} S_i) \geq \left(1 - (1 - 1/k)^l\right) \cdot w(C^*).$$

It is true for $l = 1$, since $w(S_1) \geq w(C^*)/k$ follows from (6).

For $l \geq 1$ one has:

$$\begin{aligned}
w(\bigcup_{i=1}^{l+1} S_i) &= w(\bigcup_{i=1}^{l} S_i) + w(\bigcup_{i=1}^{l+1} S_i) - w(\bigcup_{i=1}^{l} S_i) \\
&\geq w(\bigcup_{i=1}^{l} S_i) + \frac{w(C^*) - w(\bigcup_{i=1}^{l} S_i)}{k} \\
&= (1 - 1/k) \cdot w(\bigcup_{i=1}^{l} S_i) + w(C^*)/k \\
&\geq \left(1 - \frac{1}{k}\right)\left(1 - \left(1 - \frac{1}{k}\right)^l\right) \cdot w(C^*) + \frac{w(C^*)}{k} \\
&= \left(1 - (1 - 1/k)^{l+1}\right) \cdot w(C^*),
\end{aligned}$$

so the induction goes through. For $l = k$ we get:

$$w(C) \geq \left(1 - (1 - 1/k)^k\right) \cdot w(C^*) > (1 - 1/e) \cdot w(C^*).$$

The last inequality follows from

$$\begin{aligned}
(1 - 1/k)^k &= (1 + (1/(-k)))^{-(-k)} \\
&= (1 + 1/n)^{-n} \qquad \text{(for } n = -k\text{)} \\
&= ((1 + 1/n)^n)^{-1} \\
&\leq e^{-1} = 1/e.
\end{aligned}$$

Remark 3 *Sometimes it is difficult to choose the set S according to the algorithm. However, if one one would be able to make a choice for S which differs from the optimum in a factor β ($\beta < 1$) then the same algorithm provides the approx. ratio $(1 - 1/e^{\beta})^{-1}$.*

2f. The Independent-Set Problem

Instance: An undirected graph $G = (V, E)$.
Problem: Find a maximum independent set.

For $v \in V$ and $n = |V|$ define $\delta = \frac{1}{n} \sum_{v \in V} \mathrm{deg}(v)$ and

$$N(v) = \{u \in V \mid \mathrm{dist}(u, v) = 1\}.$$

Algorithm 5 $\quad$ Independent-Set(G);

$S := \emptyset$
while $V(G) \neq \emptyset$ **do**
  Find $v \in V$ with $\mathrm{deg}(v) = \min_{u \in V} \mathrm{deg}(u)$
  $S := S \cup \{v\}$
  $G := G - (v \cup N(v))$
return S

Theorem 9 *The* Independent-Set *algorithm computes an independent set S of size $q \geq n/(\delta + 1)$.*

Proof. Let v_i be the vertex chosen at step i and let $d_i = \mathrm{deg}(v_i)$. One has: $\sum_{i=1}^{q}(d_i + 1) = n$. Since at step i we delete $d_i + 1$ vertices of degree at least d_i each, for the sum of degrees S_i of the deleted vertices one has $S_i \geq d_i(d_i + 1)$. Therefore,

$$\delta n = \sum_{v \in V} \mathrm{deg}(v) \geq \sum_{i=1}^{q} S_i \geq \sum_{i=1}^{q} d_i(d_i + 1).$$

This implies

$$\delta n + n \geq \sum_{i=1}^{q} (d_i(d_i + 1) + (d_i + 1)) = \sum_{i=1}^{q} (d_i + 1)^2 \geq n^2/q$$

$\Rightarrow \frac{n}{\delta+1} \leq q = |S| \leq |S^*| \leq n$ and $|S^*|/|S| \leq \delta + 1$. $\square$

2g. 3-Coloring

Theorem 10 *Let G be a graph with $\chi(G) \leq 3$. There exists a polynomial algorithm that colors G with $O(\sqrt{n})$ colors.*

Proof: We will use the following observations

- If $\chi(G) = 2$ (i.e. G ist bipartite), then G can be colored in 2 colors in polynomial time.
- If G is a graph with max. vertex degree Δ, then G can be colored in $\Delta + 1$ colors in polynomial time (by a greedy method).

W.l.o.g. we assume $\chi(G) = 3$ and $\Delta(G) \geq \sqrt{n}$.

For $v \in V(G)$ denote $N(v) = \{u \in V \mid \mathrm{dist}_G(u, v) = 1\}$.
$\chi(G) = 3 \Rightarrow$ the subgraph induced by $G[N(v)]$ is bipartite $\forall v \in V$ and 2-colorable in polynomial time.
$\Rightarrow$ the subgraph induced by $G[v \cup N(v)]$ is 3-colorable in polynomial time.

Algorithm 6 3-Coloring;

while $\Delta(G) \geq \sqrt{n}$ **do**
  Find $v \in V(G)$ with $\deg(v) \geq \sqrt{n}$
  Color $G[v \cup N(v)]$ with 3 colors (by using a new set of
    3 colors for every v)
  Set $G := G - (v \cup N(v))$
Color G with $\Delta(G) + 1$ (new) colors.

Obviously, the running time is polynomial in n and the number of used colors is $\leq 3\frac{n}{\sqrt{n}} + \sqrt{n} + 1 = O(\sqrt{n})$. □

2h. The SUBSET-SUM Problem

Decision problem:

Instance: A set $S = \{x_1, \ldots, x_n\}$ of integers and $t \in \mathbb{N}$.
Question: Is there a subset $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} x_i = t$?

Optimization problem:

Instance: A set $S = \{x_1, \ldots, x_n\}$ of integers and $t \in \mathbb{N}$.
Problem: Find a subset $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} x_i \leq t$ and $\sum_{i \in I} x_i$ maximum.

For $A \subseteq S$ and $s \in \mathbb{N}$ define

$$A + s = \{a + s \mid a \in A\}.$$

Let P_i be the set of all partial sums of $\{x_1, \ldots, x_i\}$. One has

$$P_i = P_{i-1} \cup (P_{i-1} + x_i).$$

Algorithm 7 EXACT-SUBSET-SUM(S, t);
$n := |S|$
$L_0 := \langle 0 \rangle$
for $i = 1$ **to** n **do**
  $L_i :=$ MERGE-LISTS$(L_{i-1}, L_{i-1} + x_i)$
  $L_i := L_i - \{x \in L_i \mid x > t\}$
return the maximal element of L_n

It can be shown by induction on i that L_i is the sorted set

$$L_i = \{x \in P_i \mid x \leq t\}.$$

Polynomial Approximation Scheme

Let $L = \langle y_1, \ldots, y_m \rangle$ be a sorted list and $0 < \delta < 1$. We construct a list $L' = \langle z_1, \ldots, z_k \rangle \subseteq L$ such that:

$$\forall y \in L \;\; \exists z \in L' \;\text{ with }\; \frac{y - z}{z} \leq \delta \qquad \text{(i.e. } \; y/(1+\delta) \leq z \leq y),$$

and $|L'|$ is minimum (for the given δ).

The element $z \in L'$ will represent $y \in L$ with accuracy δ.

For example, if

$$L = \langle 10, 11, 12, 15, 20, 21, 22, 23, 24, 29 \rangle$$

then trimming of it with $\delta = 0.1$ results in

$$L' = \langle 10, 12, 15, 20, 23, 29 \rangle$$

with 11 represented by 10, 21 & 22 by 20, and 24 by 23.

<u>**Algorithm 8**</u> $\textsc{Trim}(L, \delta)$;
$m := |L|$
$L' := \langle y_1 \rangle$
$last := y_1$
for $i = 2$ **to** m **do**
    if $y_i/(1+\delta) > last$ **then**
        $\textsc{Append}(L', y_i)$
        $last := y_i$
return L'

Algorithm 9 $\textsc{Approx-Subset-Sum}(S, t, \epsilon)$;

$n := |S|$
$L_0 := \langle 0 \rangle$
for $i = 1$ **to** n **do**
  $L_i := \textsc{Merge-Lists}(L_{i-1}, L_{i-1} + x_i)$
  $L_i := \textsc{Trim}(L_i, \epsilon/2n)$
  $L_i := L_i - \{x \in L_i \mid x > t\}$
return The maximal element of L_n

Theorem 11 $\textsc{Approx-Subset-Sum}$ *is a fully polynomial time approximation scheme for the* $\textsc{Subset-Sum}$ *problem.*

Proof.
The output of the algorithm is the value z^* which is a sum of elements in the subset S. We show that $y^*/z^* \leq 1+\epsilon$, where y^* is the optimal solution.

By induction on i:

$$\forall y \in P_i \text{ with } y \leq t \quad \exists z \in L_i \text{ with } \quad y/(1+\epsilon/2n)^i \leq z \leq y.$$

Let $y^* \in P_n$ be the optimal solution. Then $\exists z \in L_n$ with

$$y^*/(1+\epsilon/2n)^n \leq z \leq y^*.$$

The output of the algorithm is the largest z.
Since the function $(1+\epsilon/2n)^n$ is monotonically increasing on n,

$$(1+\epsilon/2n)^n \leq e^{\epsilon/2} \leq 1+\epsilon/2+(\epsilon/2)^2 \leq 1+\epsilon \quad \Rightarrow \quad y^* \leq z(1+\epsilon).$$

Finally, we show that $\textsc{Approx-Subset-Sum}$ terminates in a polynomial time. For this we get a bound for L_i.

After iteration of the for-loop, for any two consecutive elements $z_{i+1}, z_i \in L_i$ one has:

$$\frac{z_{i+1}}{z_i} \geq 1 + \epsilon/2n.$$

If $L = \langle 0, z_1, \ldots, z_k \rangle$ with $0 < z_1 < z_2 < \cdots < z_k \leq t$, then

$$t \geq \frac{z_k}{z_1} = \frac{z_k}{z_{k-1}} \cdot \frac{z_{k-1}}{z_{k-2}} \cdots \frac{z_2}{z_1} \geq (1 + \epsilon/2n)^{k-1}$$

since $z_1 \geq 1$. This implies $k - 1 \leq \log_{(1+\epsilon/2n)} t$.

Taking into account $\frac{x}{1+x} \leq \ln(1 + x)$ for $x > -1$, we get

$$\begin{aligned} |L_i| &= k + 1 \\ &\leq \log_{(1+\epsilon/2n)} t + 2 \\ &= \frac{\ln t}{\ln(1 + \epsilon/2n)} + 2 \\ &\leq \frac{2n(1 + \epsilon/2n) \ln t}{\epsilon} + 2 \\ &\leq \frac{4n \ln t}{\epsilon} + 2. \end{aligned}$$

This bound is polynomial in terms of n and $1/\epsilon$. $\square$

3. Weighted Independent Set and Vertex Cover

Let $G = (V, E)$ be an undirected graph with vertex weights w_j, $j = 1, \ldots, |V| = n$. Consider the following IP for the weighed VC problem:

$$
\begin{array}{rll}
\text{Minimize} & z = \sum_{j=1}^{n} w_j x_j & \\
\text{subject to} & x_i + x_j \geq 1 & \text{for every edge } (i, j) \in E \\
 & x_j \in \{0, 1\} & \text{for every vertex } j \in V
\end{array}
$$

We relax the restriction $x_j \in \{0, 1\}$ to $0 \leq x_j \leq 1$ and get an LP approximation. The LP provides a lower bound for the IP. That is, if C^* is an optimal VC and $x^* = (x_1^*, \ldots, x_n^*)$ and Z^* is a solution to the LP, then

$$z^* \leq w(C^*).$$

Since the complement of VC is an IS, for its optimal solution S^* we get

$$w(S^*) = \sum_{i=1}^{n} w_i - w(C^*) \leq \sum_{i=1}^{n} w_i - z^*.$$

We partition V in 4 subsets:

$$
\begin{aligned}
P &= \{j \in V \mid x_j^* = 1\} \\
Q' &= \{j \in V \mid 1/2 \leq x_j^* < 1\} \\
Q'' &= \{j \in V \mid 0 < x_j^* < 1/2\} \\
R &= \{j \in V \mid x_j^* = 0\}
\end{aligned}
$$

For a set $A \subseteq V_G$ denote $w(A) = \sum_{v \in A} w(v)$.

Theorem 12 *There exist a polynomial approximation algorithm for the weighted VC with approximation rate 2.*

Proof.
We solve the LP and let $C = P \cup Q'$. One has

$$\begin{aligned} w(C^*) &\geq z^* & &= \sum_{j=1}^{n} w_j x_j \\ &= \sum_{j \in P \cup Q' \cup Q''} w_j x_j & &\geq \sum_{j \in P \cup Q'} w_j x_j \\ &= \sum_{x_j \geq 1/2} w_j x_j & &\geq \frac{1}{2} \sum_{x_j \geq 1/2} w_j \\ &= \frac{1}{2} w(C). \end{aligned}$$

Corollary 2 *For the minimum weight vertex cover C^* one has*

$$w(C^*) \geq w(P) + w(Q')/2.$$

Corollary 3 *For the maximum weight indep. set S^* one has*

$$w(S^*) \leq w(R) + w(Q')/2 + w(Q'').$$

Indeed,

$$\begin{aligned} w(S^*) &= w(G) - w(C^*) \leq w(G) - \sum_{j \in P \cup Q' \cup Q''} w_j x_j \\ &= w(R) + \sum_{j \in Q'} w_j (1 - x_j) + \sum_{j \in Q''} w_j (1 - x_j) \\ &\leq w(R) + \frac{1}{2} w(Q') + w(Q''). \end{aligned}$$

Theorem 13 *Assume $\chi = \chi(G) \geq 2$ and the optimal coloring for G is known. Then there exist polynomial approxim. algorithms for IS (resp. VC) with approxim. rate $\chi/2$ (resp. $2 - 2/\chi$).*

Proof. First, we solve the LP to find the sets P, Q', Q'', and R. Let F_i be the set of vertices with color i, $i = 1, \ldots, \chi$. Each F_i is an independent set. Denote $S = F_j \cap Q'$ with $|F_j| = \max_i |F_i \cap Q'|$. Then $w(S) \geq w(Q')/\chi$. Note that $R \cup Q''$ is an IS and there are no edges between R and Q' (so as between R and S), consider LP restrictions to check this. Hence, $R \cup Q'' \cup S$ is an IS and

$$\begin{aligned} w(R \cup Q'' \cup S) &\geq w(R) + w(Q'') + \frac{1}{\chi} w(Q') \\ &\geq \frac{2}{\chi}\left(w(R) + w(Q'') + \frac{1}{2} w(Q')\right) \\ &\geq \frac{2}{\chi} w(S^*) \qquad \text{(by Coro. (3))}. \end{aligned}$$

Furthermore, $C = V \setminus (R \cup Q'' \cup S)$ is a vertex cover and

$$\begin{aligned} w(C) &= w(G) - w(R \cup Q'' \cup S) \\ &= w(P) + (w(Q') - w(S)) \\ &\leq w(P) + \frac{\chi - 1}{\chi} w(Q') \\ &\leq \frac{2(\chi-1)}{\chi}\left(w(P) + \frac{1}{2} w(Q')\right) \\ &\leq \left(2 - \frac{2}{\chi}\right) w(C^*) \qquad \text{(by Coro. (2))}. \end{aligned}$$

If G is a connected graph of max-degree $\Delta > 3$ and $G \neq K_{\Delta+1}$, then $\chi(G) \leq \Delta$ (Brooks Theorem). Therefore,

Corollary 4 *There exist polynomial approx. algorithms for IS (resp. VC) with approx. rate $\Delta/2$ (resp. $2 - 2/\Delta$).*

Since $\chi(G) = 4$ for any planar graph, we get

Corollary 5 *For planar graphs there exist polynomial approx. algorithms for IS (resp. VC) with approx. rate 2 (resp. $3/2$).*