DM865 – <u>Heuristics</u> & <u>Approximation Algorithms</u>
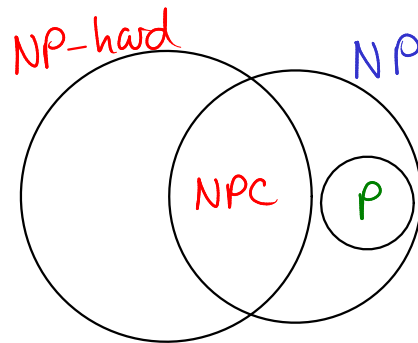
(Marco) (Lene)

Combinatorial problems:

First
2 weeks → Traveling Salesman (TSP)
MAX SAT
Set Cover
Knapsack
Bin packing
Scheduling

decision version
$\in$ NPC

Polynomial algorithm:

algo. with running time $O(n^c)$, for some constant c.

P: The set of decision problems that allow for a poly. algo.

NP: A problem belongs to NP, if solutions can be verified in poly. time.

If _any_ NP-hard problem has a poly. algo., then _all_ problems in NPC have poly. algo.s.

$\underbrace{\text{Optimal solutions}}_{(1)}$ $\underbrace{\text{in poly. time}}_{(2)}$ $\underbrace{\text{for all instances}}_{(3)}$

- Choose two! ((2) & (3))

Section 1.1

An approximation algorithm comes with a performance guarantee:

Def 1.1: α-approximation algorithm

An α-approximation algorithm for an optimization problem P is a poly. time algo. ALG s.t. for any instance I of P,

- $\dfrac{ALG(I)}{OPT(I)} \leq \alpha$, if P is a <u>minimization</u> problem
- $\dfrac{ALG(I)}{OPT(I)} \geq \alpha$, if P is a <u>maximization</u> problem

Thus, for max. problems, $0 \leq \alpha \leq 1$, and, for min. problems, $\alpha \geq 1$.

The approximation factor / approximation ratio is
- the smallest possible α (for min. problems)
- the largest possible α (for max. problems)

More precisely, the approx. factor R is

$$R = \inf \left\{ \alpha \mid \forall I : \frac{ALG(I)}{OPT(I)} \leq \alpha \right\} \text{ for min. problems}$$

$$R = \sup \left\{ \alpha \mid \forall I : \frac{ALG(I)}{OPT(I)} \geq \alpha \right\} \text{ for max. problems}$$

We will cover the rest of Section 1.1 later.

$$\boxed{\text{Section 2.4: TSP}}$$

The Traveling Salesman Problem (TSP)

Input: Weighted complete graph G

$c_{ij} = c_{ji}, \qquad i,j \in V$

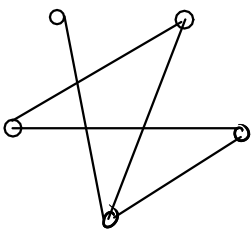$c_{ii} = 0, \qquad i \in V$

$c_{ij} \geq 0, \qquad i,j \in V$

Output: Hamiltonian cycle of min. total weight

$\underbrace{\hspace{4cm}}$

Cycle visiting each vertex exactly once.
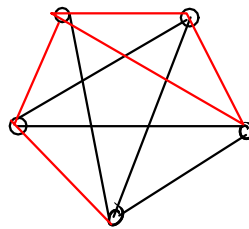
Decision version of TSP:

Does there exist a tour of cost $\leq X$?

Decision version of TSP is NP-hard

Reduction from Hamilton Cycle:

Ham. cycle   TSP

$c_{ij} = 1$

$c_{ij} = 2$

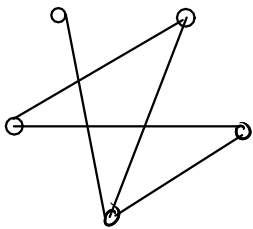$\exists$ ham. cycle $\iff$ $\exists$ tour of cost n

Even worse, no approximation guarantee possible:
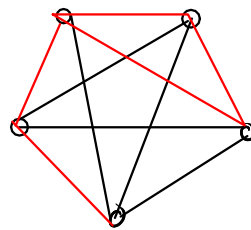
Theorem 2.9

$\forall \alpha > 1$, $\nexists$ α-approx alg. for TSP (unless P = NP)

Proof: Reduction from Hamilton Cycle:

Ham. cycle   TSP

$c_{ij} = 1$
$c_{ij} = \alpha n + 1$

$\exists$ ham. cycle $\iff$ $\exists$ tour of cost n

$\implies$ α-approx. alg. gives tour of cost $\leq \alpha n$

$\impliedby$ α-approx. alg. returns a tour with no red edges, i.e., a tour of cost n.

$\exists$ ham. cycle $\impliedby$ α-approx alg. returns a tour of cost n (i.e. a tour with no red edges). □
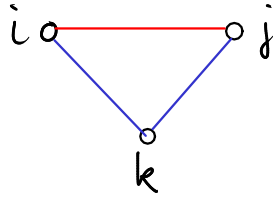
Note: The proof does not require α to be a constant. In fact, it could be 2^n, or any function computable in poly. time.

Thus, we will only consider a special case of TSP:

Metric TSP:

The edge weights satisfy the triangle inequality:

$$c_{ij} \leq c_{ik} + c_{kj}, \quad \text{for all} \quad i, j, k \in V$$

For metric TSP, the proof of Thm 2.9 does not work (the max. possible cost of the red edges would be 2).

For Metric TSP, we will consider three algorithms:

The Nearest Addition algorithm	2-approx.
The Double Tree algorithm	2-approx.
Christofide's Algorithm	3/2-approx

Nearest Addition (NA)

$u, v \leftarrow$ two nearest neighbors in V
$\overline{\text{Tour}} \leftarrow \langle u, v, u \rangle$
For $i \leftarrow 1$ to $n-2$
    $v \leftarrow$ nearest neighbour of Tour
    $u_1 \leftarrow$ nearest neighbor of v in Tour
    $u_2 \leftarrow u_1$'s successor in Tour
    Add v to Tour between u_1 and u_2

Ex:
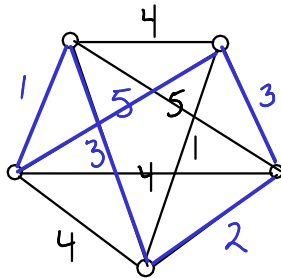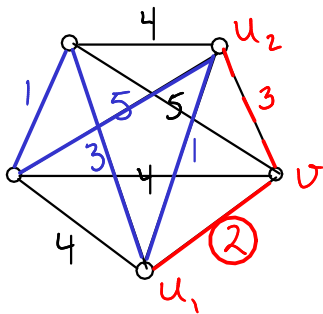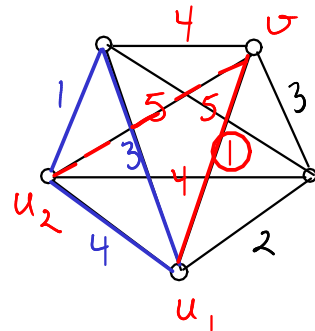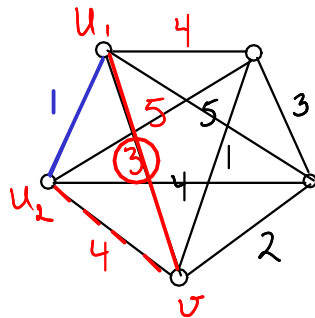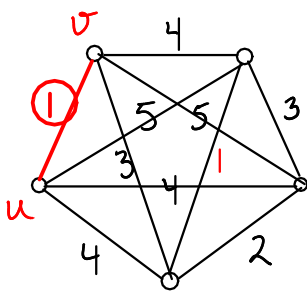
$$C_{NA} = 1 + 3 + 2 + 3 + 5$$
$$= 14$$
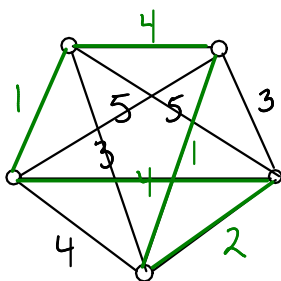
$$C_{OPT} \leq 1 + 4 + 1 + 2 + 4 = 12$$

For Metric TSP, Nearest Neighbor is a 2-approx. alg.:
We will prove that

(1) $\boxed{C_{NA} \leq 2 \cdot C(MST)}$

(2) $\boxed{C(MST) \leq C_{OPT}}$ (Lemma 2.10)

(1): The solid red edges are exactly those chosen by Prim's Algorithm:
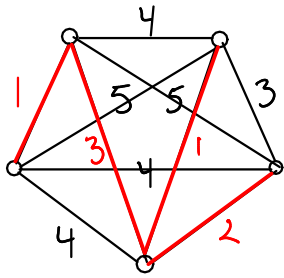
$C = 1 + 3 + 1 + 2 = 7$

Thus, the total cost C of these edges is that of a minimum spanning tree:

$$\boxed{C = c(MST)}$$

Adding a new vertex v to the tour, we add two edges and delete one:

$y \leq z$ by the $\triangle$-ineq.
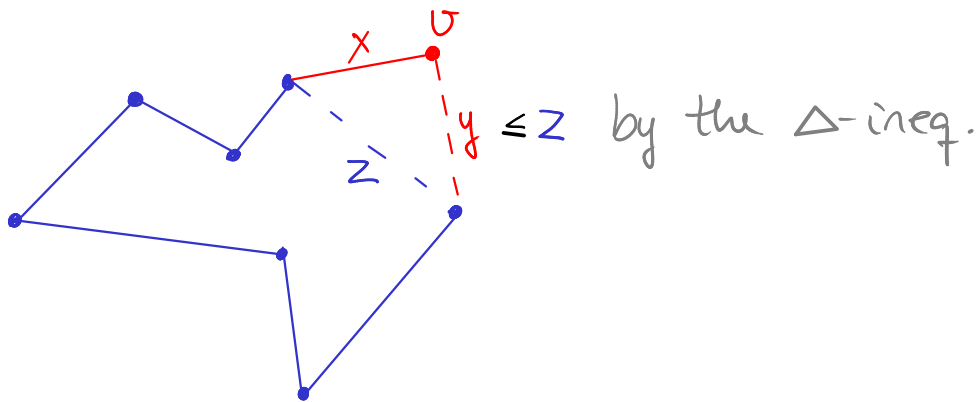
Adding v costs
$$x + y - z \leq x + (x + z) - z = 2x$$
where x is the cost of Prim's Alg. in this step

Thus,
$$\boxed{C_{NA} \leq 2C = 2c(MST)}$$

(2): Deleting any edge from a tour, we get a spanning tree:
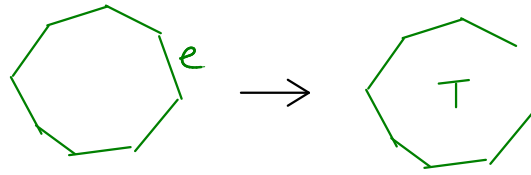
For any spanning tree T obtained by deleting an edge e from an optimal tour,

$$\boxed{c_{OPT} \geq c(T)}, \quad \text{since} \quad w(e) \geq 0$$

$$\geq \boxed{c(MST)}$$

Now,

$$(1) \& (2) \implies c_{NA} \leq 2\,c(MST) \leq 2\,c_{OPT}$$

This proves:

Theorem 2.11

For Metric TSP, Nearest Addition is a 2-approx. alg.

Double Tree algorithm

Noting that NA adds the edges of a MST one by one, we could also make a MST T and traverse T, making shortcuts whenever we would otherwise visit a node for the second time:
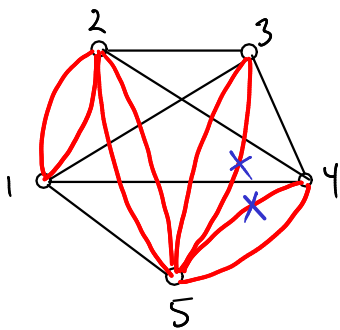
By the triangle inequality, this distance is no longer

$\langle 1, 2, 5, 3, 5, 4, 5, 2, 1 \rangle$

then this total distance

↓ shortcut 5
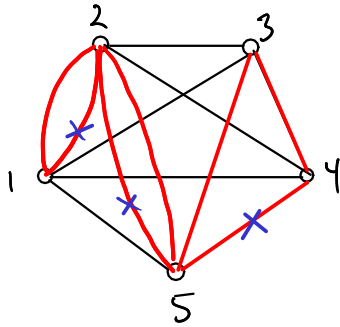
$\langle 1, 2, 5, 3, 4, 5, 2, 1 \rangle$

↓ shortcut 5 and 2
  (using △-ineq. twice)
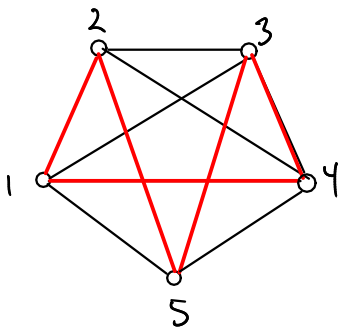
$\langle 1, 2, 5, 3, 4, 1 \rangle$

An Euler tour is a traversal of a graph that traverses each edge exactly once.

A graph that has an Euler tour is called eulerian.

A graph is eulerian if and only if all vertices have even degree.
Constructive proof of „if" in exercises for Wednesday.

Double Tree Algorithm (DT)

$T \leftarrow$ MST
$DT \leftarrow T$ with all edges doubled
Etour $\leftarrow$ Euler tour in DT
Tour $\leftarrow$ vertices in order of first appearance in ETour

---

Same analysis as for NA :

$$c_{DT} \leq 2\, c(MST) \leq 2 \cdot c_{OPT}$$

Hence :

Theorem 2.12

Double Tree is a 2-approx. alg

Next idea: Not necessary to add n-1 edges
to obtain even degree for all vertices
Instead: add a minimum perfect matching on
vertices of odd degree in the MST.
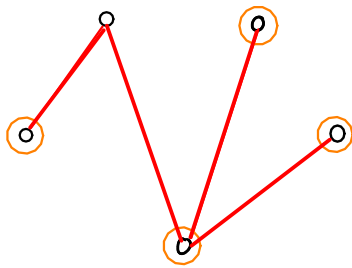
MST
Odd degree
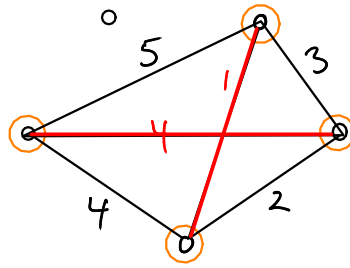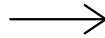
Min. matching
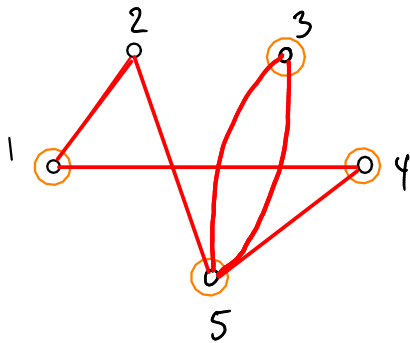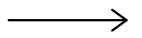
Euler tour : ⟨1,2,5,3,5,4,1⟩

short cutting
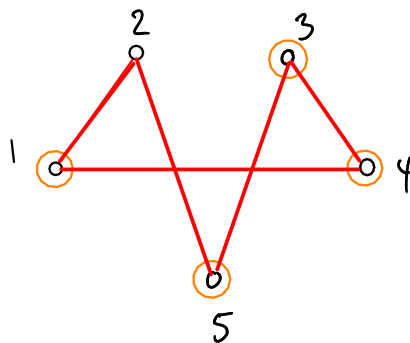
TSP tour : ⟨1,2,5,3,4,1⟩

Note that it is always possible to find a perfect matching,
since there is always an even #odd-degree vertices in T.

Christofide's Algorithm (CA)

$T \leftarrow$ MST

$M \leftarrow$ minimum perfect matching on odd degree vertices in T

ETour $\leftarrow$ Euler tour in the subgraph $(V, E(T) \cup M)$

Tour $\leftarrow$ vertices in order of first appearance in ETour

Theorem 2.13

Christofide's Algorithm is a $\frac{3}{2}$-approx. alg.

Proof:

By the triangle inequality,

$$C_{CA} \leq C(T) + C(M)$$
$$\leq C_{OPT} + C(M), \text{ by Lemma 2.10}$$

Thus, we just need to prove that

$$C(M) \leq \frac{1}{2} C_{OPT}$$
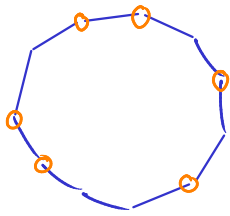
short cutting
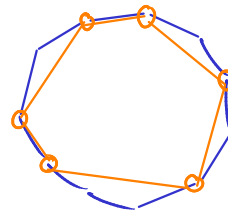
Optimal TSP tour
Odd degree vertices
in T

C: cost of orange cycle

$\boxed{C \leq c_{OPT}}$, by $\triangle$-ineq.

Since the cycle on the odd degree vertices has an even #edges, it consists of two perfect matchings:
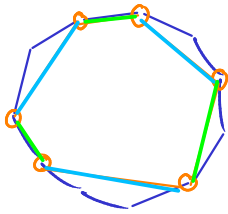
$C = C + C$

$\Downarrow$

$\boxed{\min\{C, C\} \leq \frac{1}{2} \cdot C \leq \frac{1}{2} \cdot c_{OPT}}$

Since M is a __minimum__ matching on the odd degree vertices,

$\boxed{c(M) \leq \min\{C, C\} \leq \frac{1}{2} \cdot c_{OPT}}$

$\square$

No alg. with an approx. ratio better than $\frac{3}{2}$ is currently known. Moreover:

Theorem 2.14

For $\alpha < \frac{220}{219}$, $\nexists$ α-approx. alg. for Metric TSP

The result of Thm 2.14 is from 2000.
In 2015, the same result was proven for $\alpha < \frac{185}{184}$.

Chapter 5: Maximum Satisfiability

SAT: For a given boolean formula φ in CNF, does there exist a truth assignment satisfying φ?

Conjunctive normal form (CNF): the formula is a conjunction ($\wedge$) of disjunctions ($\vee$). Each disjunction is called a clause.

Ex:
$$\varphi \equiv \underbrace{(X_1 \vee \overline{X_2} \vee X_3)}_{\text{clause } C_1} \wedge \underbrace{\overline{X_3}}_{C_2} \wedge \underbrace{(X_1 \vee X_2)}_{C_3}$$

positive literal → X_1

negative literal → $\overline{X_2}$

X_1, X_2, X_3 are variables

C_j has length / size l_j:
$$l_1 = 3, \quad l_2 = 1, \quad l_3 = 2$$

$X_1 \leftarrow T, \quad X_3 \leftarrow F$ will satisfy φ

MAX SAT

Input: Boolean formula φ in CNF
with variables $x_1, x_2, ..., x_n$
and clauses $C_1, C_2, ..., C_m$
Each clause, C_j, has a weight w_j

Output: Truth assignment maximizing the total weight of satisfied clauses

Ex: $(x_1 \vee \overline{x}_2) \wedge x_3 \wedge (x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)$
$w_1 = 2$ $w_2 = 2$ $w_3 = 1$ $w_4 = 3$

$x_1 \leftarrow T, \; x_2 \leftarrow F, \; x_3 \leftarrow T$ satisfies C_1, C_2, C_4
with a total weight of 7.

This is optimal, since we cannot satisfy
all clauses:
     C_2 requires $x_3 \leftarrow T$
     C_3 then requires $x_2 \leftarrow T$
     C_1 then requires $x_1 \leftarrow T$
     But then C_4 is false.

SAT, and hence, MAX SAT, is NP-hard.
How can we approximate?

Consider the following alg:

Rand

For $i \leftarrow 1$ to n
    With prob. $\frac{1}{2}$ set x_i true

This corresponds to choosing a solution uniformly at random.

Ex: $(x_1 \vee \overline{x}_2) \wedge x_3 \wedge (x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)$
          2   2   1   3

Tree:

$x_1 \leftarrow T$ $x_1 \leftarrow F$

$x_2 \leftarrow T$ $x_2 \leftarrow F$ $x_2 \leftarrow T$ $x_2 \leftarrow F$

$x_3 \leftarrow T$ $x_3 \leftarrow F$

$W = 2+2+1 = 5$

$W = 2+1+3 = 6$

$W = 2+2+3 = 7$

$W = 6$

$W = 6$

$W = 4$

$W = 7$

$W = 6$
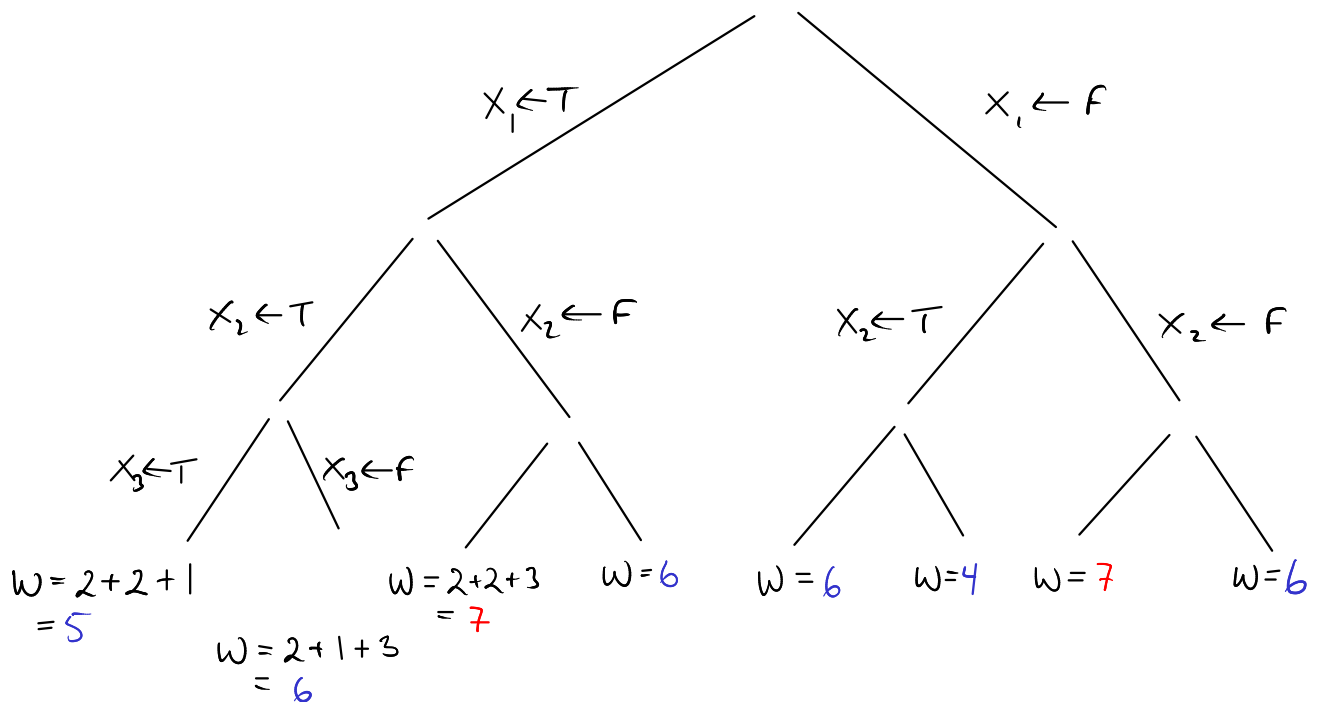
Thus, for this example,
$$E[Rand] = \frac{1}{8}(5+6+7+6+6+4+7+6) = 5\frac{7}{8}$$

We don't need to calculate the weight of each possible output...

Instead, we can calculate the exp. weight of each clause:

$$(X_1 \vee \overline{X}_2) \wedge X_3 \wedge (X_2 \vee \overline{X}_3) \wedge (\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3)$$

Satisfied, unless $X_1 \equiv F$ and $X_2 \equiv T$, i.e., satisfied with prob.
$$1 - \tfrac{1}{2} \cdot \tfrac{1}{2} = \tfrac{3}{4}$$

satisfied w. prob.
$$1 - \tfrac{1}{2} = \tfrac{1}{2}$$

Satisfied w. prob.
$$1 - \tfrac{1}{2} \cdot \tfrac{1}{2} \cdot \tfrac{1}{2} = \tfrac{7}{8}$$

Thus, by linearity of expectation,
$$E[\text{Rand}] = \tfrac{3}{4} \cdot 2 + \tfrac{1}{2} \cdot 2 + \tfrac{3}{4} \cdot 1 + \tfrac{7}{8} \cdot 3 = 5\tfrac{7}{8}$$

In general, clause C_j is satisfied with prob. $1 - \left(\tfrac{1}{2}\right)^{\ell_i}$.

We let $W \equiv \sum_{j=1}^{m} w_j$.

> **Theorem 5.1:** Rand is a $\frac{1}{2}$-approx. alg

Proof:

$$OPT \leq W$$

By linearity of expectation:

$$E[Rand] = \sum_{j=1}^{m} \left(1 - \left(\tfrac{1}{2}\right)^{l_j}\right) w_j$$

$$\geq \tfrac{1}{2} W \quad , \quad \text{since } l_j \geq 1 \qquad \square$$

In Section 5.1 we got a simple algorithm with a guarantee on the _expected_ performance. We can turn it into a guarantee on the worst-case performance:

$\boxed{\text{Section 5.2 : Derandomization}}$

$\underline{\text{Ex}}$ from before:

$\phi : (X_1 \vee \overline{X}_2) \wedge X_3 \wedge (X_2 \vee \overline{X}_3) \wedge (\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3)$

$\quad w_1 = 2 \qquad w_2 = 2 \qquad w_3 = 1 \qquad\qquad w_4 = 3$

If we let $\boxed{X_1 \leftarrow \top}$, the formula becomes

$\quad \phi_\top \equiv \top \wedge X_3 \wedge (X_2 \vee \overline{X}_3) \wedge (\overline{X}_2 \vee \overline{X}_3)$

and

$\quad E[\text{Rand}(\phi_\top)] = 2 + \tfrac{1}{2} \cdot 2 + \tfrac{3}{4} \cdot 1 + \tfrac{3}{4} \cdot 3 = \boxed{6}$

Or, recalling the probability tree,

$\quad E[\text{Rand}(\phi_\top)] = \tfrac{1}{4}(5 + 6 + 7 + 6) = 6$

Similarly, if we let $X_1 \leftarrow F$, the formula becomes

$\quad \phi_F \equiv \overline{X}_2 \wedge X_3 \wedge (X_2 \vee \overline{X}_3) \wedge \top$

and

$\quad E[\text{Rand}(\phi_F)] = \tfrac{1}{2} \cdot 2 + \tfrac{1}{2} \cdot 2 + \tfrac{3}{4} \cdot 1 + 3 = 5\tfrac{3}{4}$

$\quad$ Or $E[\text{Rand}(\phi_F)] = \tfrac{1}{4}(6 + 4 + 7 + 6) = 5\tfrac{3}{4}$

Note that $E[\text{Rand}]$ is the average of 6 and $5\tfrac{3}{4}$:

$\quad E[\text{Rand}] = \tfrac{1}{2} \cdot E[\text{Rand}(\phi_\top)] + \tfrac{1}{2} \cdot E[\text{Rand}(\phi_F)]$

Thus,

$\quad \max\{ E[\text{Rand}(\phi_\top)], E[\text{Rand}(\phi_F)] \} \geqslant E[\text{Rand}]$

$\phi: (X_1 \vee \bar{X}_2) \wedge X_3 \wedge (X_2 \vee \bar{X}_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3)$

$E[Rand(\phi)] = 5\frac{7}{8}$

$X_1 \leftarrow T$

$X_1 \leftarrow F$

$\phi_T: T \wedge X_3 \wedge (X_2 \vee \bar{X}_3) \wedge (\bar{X}_2 \vee \bar{X}_3)$

$E[Rand(\phi_T)] = 6$

$\phi_F: \bar{X}_2 \wedge X_3 \wedge (X_2 \vee \bar{X}_3) \wedge T$

$E[Rand(\phi_F)] = 5\frac{3}{4}$

$X_2 \leftarrow T$

$X_2 \leftarrow F$

$\phi_{TT}: T \wedge X_3 \wedge T \wedge \bar{X}_3$

$E[Rand(\phi_{TT})] = 5\frac{1}{2}$

$\phi_{TF}: T \wedge X_3 \wedge \bar{X}_3 \wedge T$

$E[Rand(\phi_{TF})] = 6\frac{1}{2}$

$X_3 \leftarrow T$

$X_3 \leftarrow F$

$\phi_{TFT}: T \wedge T \wedge F \wedge T$

$Rand(\phi_{TFT}) = 7$

$\phi_{TFF}: T \wedge F \wedge T \wedge T$

$Rand(\phi_{TFF}) = 6$

In general:

$$\max \{ E[Rand(\phi_T)] , E[Rand(\phi_F)] \} \geq E[Rand] \geq \tfrac{1}{2} W,$$

The same is true for ϕ_T and ϕ_F:

$$\max \{ E[Rand(\phi_{TT})], E[Rand(\phi_{TF})] \} \geq E[Rand(\phi_T)]$$

and

$$\max \{ E[Rand(\phi_{FT})], E[Rand(\phi_{FF})] \} \geq E[Rand(\phi_F)]$$

Inductively, this proves that the following alg. is a $\tfrac{1}{2}$-approx. alg.:

DeRand(ϕ)

For $i \leftarrow 1$ to n
    If $E[Rand(\phi_{x_1 \cdots x_{i-1} T})] \geq E[Rand(\phi_{x_1 \cdots x_{i-1} F})]$
      $x_i \leftarrow T$
    Else
      $x_i \leftarrow F$

This method of derandomization is sometimes called the method of conditional expectations. (We calculate the conditional exp. of Rand given that $x_i \leftarrow T$ and given that $x_i \leftarrow F$.)

Note that short clauses are „harder" than long clauses:

If all clauses have $\ell \geq 2$, (De)Rand is a $3/4$-approx. alg.
(In Section 5.3, we will pursue this obs. to obtain a ≈ 0.6-approx. alg.)

If all clauses have $\ell \geq 3$, (De)Rand is a $7/8$-approx alg.

In some sense, this is optimal:

MAX E3SAT: The special case of MAX SAT where $\ell = 3$ for all clauses.

Theorem 5.2:

$\exists \varepsilon > 0 : \exists (\frac{7}{8} + \varepsilon)$-approx alg for MAX E3SAT $\Rightarrow$ P = NP

Section 5.3: A biased rand. alg.

Since **unit clauses** (clauses of exactly one literal) are the „hardest", we should focus on these to obtain a better approx. ratio.

For each i, $1 \leq i \leq n$, we define:

$$u_i \equiv \begin{cases} \text{weight of unit clause } x_i, & \text{if it exists} \\ 0, & \text{otherwise} \end{cases}$$

$$v_i \equiv \begin{cases} \text{weight of unit clause } \bar{x}_i, & \text{if it exists} \\ 0, & \text{otherwise} \end{cases}$$

Idea: If $u_i \geq v_i$, set x_i true with prob. $> \frac{1}{2}$, and vice versa.

For ease of presentation, assume that
$$u_i \geq v_i, \quad 1 \leq i \leq n$$
Why is this not a restriction?

Thus, each variable will be set true with prob. $> \frac{1}{2}$:

For any $p > \frac{1}{2}$, we define the following alg:

Rand$_p$

for $i \leftarrow 1$ to n
    With prob. p set x_i true

What is an optimal value of p?

Lemma 5.4

For any clause C_j which does not consist of one negated variable,
Rand$_p$ satisfies C_j with prob $\geq \min\{p, 1-p^2\}$

Proof:

If $\ell_j = 1$, C_j consists of one unnegated variable.
In this case, C_j is satisfied with prob. p.

If $\ell_j = 2$, the worst case is if both literals
are negated variables, since $p > \frac{1}{2}$. Thus, in this
case, C_j is satisfied with prob. $\geq 1-p^2$.

If $\ell_j \geq 3$, the prob. of C_j being satisfied is at
least the worst-case prob. for $\ell_j = 2$. $\square$

> **Lemma 5.6:** $OPT \leq W - \sum_{i=1}^{n} v_i$

Proof:

By assumption, $u_i \geq v_i$, for all i.

Thus, if $v_i > 0$, there is both an x_i- and an $\bar{x}_i$-clause. Both clauses cannot be satisfied.

Thus, for each $v_i > 0$, there is an unsatisfied clause of weight $\geq \min\{u_i, v_i\} = v_i$. $\qquad \square$

We can obtain an alg. with approx. ratio $\frac{1}{2}(\sqrt{5}-1) \approx 0{,}618$:

> **Theorem 5.7:**
> For $\rho = \frac{1}{2}(\sqrt{5}-1)$, $Rand_\rho$ is a ρ-approx. alg.

Proof:

By Lemma 5.4,

Total weight of clauses that are $\underbrace{\text{not } negated \text{ unit clauses}}$

$$E[\text{Rand}_\rho] \geq \min\{\rho, 1-\rho^2\} \cdot \left(W - \sum_{i=1}^{n} v_i\right)$$

$$= \rho\left(W - \sum_{i=1}^{n} v_i\right), \text{ for } \rho = \tfrac{1}{2}(\sqrt{5}-1):$$

$$1 - \left(\tfrac{1}{2}(\sqrt{5}-1)\right)^2 = 1 - \tfrac{1}{4}(5+1-2\sqrt{5}) = 1 - \tfrac{3}{2} + \tfrac{1}{2}\sqrt{5}$$

$$= \tfrac{1}{2}(\sqrt{5}-1)$$

By Lemma 5.6, $OPT \leq W - \sum_{i=1}^{n} v_i$.

Hence, for $\rho = \tfrac{1}{2}(\sqrt{5}-1)$, $\dfrac{E[\text{Rand}_\rho]}{OPT} \geq \rho$ $\square$

Note that

$\boxed{\text{Rand}_\rho \text{ can be } \textcolor{green}{\text{derandomized}} \text{ exactly like Rand}}$

$\boxed{\text{Section 5.4: Randomized rounding}}$

In Section 5.3 we saw that biasing the prob. of setting each variable true resulted in a better approx. guarantee.

The approximation ratio can be further improved by allowing a different bias for each variable. We will develop an LP-formulation of the problem.

For each clause, C_j, we define:

P_j: the set of indices of variables that occur positively in C_j

N_j: ——————————————— " ——————————————— negatively —" —

Then, C_j can be written as $\displaystyle\bigvee_{i \in P_j} x_i \;\vee\; \bigvee_{i \in N_j} \bar{x}_i$

$$\text{Ex:} \quad \cdots \wedge \overbrace{\left(x_1 \vee \bar{x}_2 \vee \bar{x}_4 \vee x_7 \right)}^{C_j} \wedge \cdots$$

$$P_j = \{1, 7\}, \quad N_j = \{2, 4\}$$

$$\left((x_1 \vee x_7) \vee (\bar{x}_2 \vee \bar{x}_4) \right)$$

If $y_i = 0 \iff x_i \equiv F$ and $y_i = 1 \iff x_i \equiv T$, then C_j is true, iff

$$\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq 1$$

This leads to the following IP-formulation:

IP_ϕ:

$$\max \quad \sum_{j=1}^{m} z_j w_j$$

subject to

$$\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, \quad 1 \leq j \leq m$$

$$y_i \in \{0, 1\}, \quad 1 \leq i \leq n$$

$$z_j \in \{0, 1\}, \quad 1 \leq j \leq m$$

Let LP_ϕ be the LP-relaxation of IP_ϕ, i.e.,

LP_ϕ:

$$\max \quad \sum_{j=1}^{m} z_j w_j$$

subject to

$$\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, \quad 1 \leq j \leq m$$

$$0 \leq y_i \leq 1, \quad 1 \leq i \leq n$$

$$0 \leq z_j \leq 1, \quad 1 \leq j \leq m$$

Clearly, $Z^*_{LP_\phi} \geq Z^*_{IP_\phi} = OPT$

value of opt. sol. to LP_ϕ

value of opt. sol. to IP_ϕ

value of opt. sol. to corresponding MAXSAT problem

$(\vec{y^*}, \vec{z^*}) \leftarrow$ opt. sol. to LP_ϕ $\left(\text{i.e., } Z^*_{LP_\phi} = \sum_{j=1}^{m} z^*_j w_j\right)$
For $i \leftarrow 1$ to n
    Set x_i true with prob. y_i^*

The approx. ratio of RandRounding is at least $1 - \frac{1}{e} \approx 0.632$.
For proving this, we will use the following two facts:

Fact 5.8 (Arithmetic-geometric mean inequality):
For any $a_1, a_2, \ldots, a_k \geq 0$,

$$\left(\prod_{i=1}^{k} a_i\right)^{1/k} \leq \frac{1}{k} \sum_{i=1}^{k} a_i$$
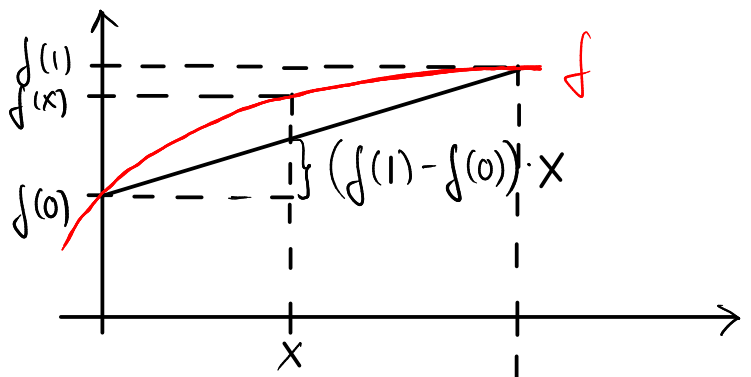
$\Downarrow$

$$\prod_{i=1}^{k} a_i \leq \left(\frac{1}{k} \sum_{i=1}^{k} a_i\right)^k$$

A function f is concave on an interval I,
if $f''(x) \leq 0$ for any $x \in I$. (the slope is nonincreasing)

Fact 5.9:
$\Downarrow$ f is concave on $[0,1]$
$\forall x \in [0,1]: f(x) \geq (f(1) - f(0))x + f(0)$

$$\boxed{\text{Theorem 5.10}: \text{RandRounding is a } (1-\tfrac{1}{e}) - \text{approx. alg}}$$

<u>Proof</u>:

For $1 \leq j \leq m$, let p_j be the probability that C_j is satisfied, and let $\overline{p_j} = 1 - p_j$.

Our goal is to show that $p_j \geq (1-\tfrac{1}{e}) z_j^*$.

This will establish the approx. factor, since $OPT \leq \sum_{j=1}^{m} z_j^* w_j$

$$\overline{p_j} = \overline{\prod_{i \in P_j} (1-y_i^*)} \ \overline{\prod_{i \in N_j} y_i^*}$$

$$\leq \left(\frac{1}{\ell_j} \left(\sum_{i \in P_j} (1-y_i^*) + \sum_{i \in N_j} y_i^* \right) \right)^{\ell_j}, \quad \text{by Fact 5.8}$$

$$= \left(\frac{1}{\ell_j} \left(|P_j| - \sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - 1 + y_i^*) \right) \right)^{\ell_j}$$

$$= \left(\frac{1}{\ell_j} \left(|P_j| - \sum_{i \in P_j} y_i^* + |N_j| - \sum_{i \in N_j} (1-y_i^*) \right) \right)^{\ell_j}$$

$$= \left(1 - \frac{1}{\ell_j} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1-y_i^*) \right) \right)^{\ell_j}, \quad \text{since } |P_j| + |N_j| = \ell_j$$

$$\leq \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j}, \quad \text{since } (\vec{y}^*, \vec{z}^*) \text{ is a solution to } LP_\phi$$

Thus,

$$p_j \geq 1 - \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j} \equiv f(z_j^*)$$

which is a concave function of z_j^*:

$$f'(z_j^*) = - \ell_j \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j - 1} \cdot \left(-\frac{1}{\ell_j} \right) = \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j - 1}$$

$$f''(z_j^*) = (\ell_j - 1) \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j - 2} \cdot \left(-\frac{1}{\ell_j} \right) = \left(\frac{1}{\ell_j} - 1 \right) \left(1 - \frac{z_j^*}{\ell_j} \right)^{\ell_j - 2}$$

$$\leq 0 \qquad\qquad \underbrace{\phantom{\left(\frac{1}{\ell_j}-1\right)}}_{\leq 0} \ \underbrace{\phantom{\left(1-\frac{z_j^*}{\ell_j}\right)}}_{\geq 0}$$

Note that
$$f(0) = 1 - \left(1 - \frac{0}{\ell_j}\right)^{\ell_j} = 1 - 1 = 0$$
$$f(1) = 1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}$$

Thus,
$$p_j \geq f(z_j^*)$$
$$\geq \left(f(1) - f(0)\right) z_j^* + f(0), \quad \text{by Fact 5.9}$$
$$\geq \left(1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right) z_j^*$$

Hence,
$$E[\text{Rand Rounding}] = \sum_{j=1}^{m} p_j w_j$$
$$\geq \sum_{j=1}^{m} \left(1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right) z_j^* w_j$$
$$\geq \left(1 - \frac{1}{e}\right) \cdot \underbrace{\sum_{j=1}^{m} z_j^* w_j}_{= z_{LP_\phi}^* \geq OPT}$$

$\square$
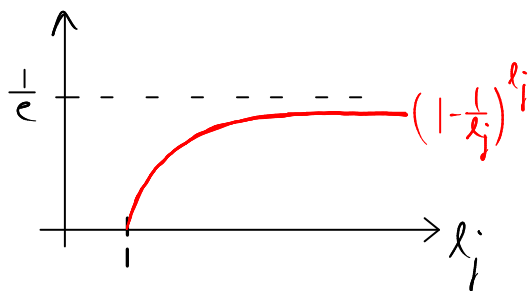
Note that

Section 5.5 : Choosing the better of two solutions

Combining the alg.s of Sections 5.1 and 5.4 gives a better approx. factor than using any one of them separately. This is because they have different worst-case inputs:

Rand satisfies clause C_j with prob. $P_R = 1 - \left(\frac{1}{2}\right)^{l_j}$.

RandRounding satisfies C_j with prob. $P_{RR} \geq \left(1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right) z_j^*$.

While P_R increases with l_j, the lower bound on P_{RR} decreases with l_j.

Best Of Two (ϕ)

$\vec{X}_R \leftarrow$ Rand(ϕ)

$\vec{X}_{RR} \leftarrow$ RandRounding(ϕ)

If $w(\phi, \vec{X}_R) \geq w(\phi, \vec{X}_{RR})$

   Return $\vec{X}_R$

Else

   Return $\vec{X}_{RR}$

Note that

BestOfTwo is **derandomized** by using the derandomized versions of Rand and RandRounding.

$\boxed{\text{Theorem 5.11 : } \text{BestOfTwo} \text{ is a } \frac{3}{4}\text{-approx. alg.}}$

Proof:

$$E[\text{BestOfTwo}(\phi)] = E\left[\max\{\text{Rand}(\phi), \text{RandRounding}(\phi)\}\right]$$

$$\geq E\left[\frac{1}{2}\text{Rand}(\phi) + \frac{1}{2}\text{RandRounding}(\phi)\right]$$

$$= \frac{1}{2}E[\text{Rand}(\phi)] + \frac{1}{2}E[\text{RandRounding}(\phi)] \text{, by lin. of exp.}$$

$$\geq \frac{1}{2}\sum_{j=1}^{m}(1-2^{-\ell_j})w_j + \frac{1}{2}\sum_{j=1}^{m}\left(1-\left(1-\frac{1}{\ell_j}\right)^{\ell_j}\right)z_j^* w_j$$

$$\geq \sum_{j=1}^{m} z_j^* w_j \cdot \underbrace{\frac{1}{2}\left(1-2^{-\ell_j} + 1-\left(1-\frac{1}{\ell_j}\right)^{\ell_j}\right)}_{\equiv\, p_j} \text{, since } z_j^* \leq 1.$$

For $\ell_j = 1$, $p_j = \frac{1}{2}\left(1-\frac{1}{2} + 1 - 0\right) = \frac{3}{4}$

For $\ell_j = 2$, $p_j = \frac{1}{2}\left(1-\frac{1}{4} + 1 - \left(1-\frac{1}{2}\right)^2\right) = \frac{3}{4}$

For $\ell_j \geq 3$, $p_j \geq \frac{1}{2}\left(1-\frac{1}{8} + 1 - \frac{1}{e}\right) > \frac{3}{4}$

Hence,

$$E[\text{BestOfTwo}] \geq \sum_{j=1}^{m} z_j^* w_j \cdot \frac{3}{4} \geq \frac{3}{4}\cdot\text{OPT} \qquad \square$$

$$\boxed{\text{Section 5.6: Nonlinear randomized rounding}}$$

$(\vec{y^*}, \vec{z^*}) \leftarrow$ opt. sol. to LP_ϕ

For $i \leftarrow 1$ to n

   Set x_i true with prob. $f(y_i^*)$

Theorem 5.12

RandRounding$_f$ is a $\frac{3}{4}$-approx. alg., if $1 - 4^{-x} \le f(x) \le 4^{x-1}$

Proof:

Prob. that C_j is **not** satisfied:

$$\bar{P}_j = \prod_{i \in P_j} (1 - f(y_i^*)) \prod_{i \in N_j} f(y_i^*)$$

$$\le \prod_{i \in P_j} 4^{-y_i^*} \prod_{i \in N_j} 4^{y_i^* - 1}$$

$$= 4^{-\left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} 1 - y_i^* \right)}$$

$$\le 4^{-z_j^*}$$

Prob. that C_j **is** satisfied:

$$P_j = 1 - \bar{P}_j \ge 1 - 4^{-z_j^*}$$

$$\ge 0 + \left(\frac{3}{4} - 0 \right) z_j^*, \quad \text{by Fact 5.9}$$

$$= \frac{3}{4} z_j^*$$

$\square$
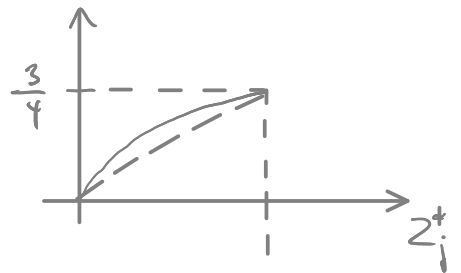
Ex: $\phi \equiv (x_1 \lor x_2) \land (x_1 \lor \bar{x}_2) \land (\bar{x}_1 \lor x_2) \land (\bar{x}_1 \lor \bar{x}_2)$

$w_1 = w_2 = w_3 = w_4 = 1$

$OPT = 3$

$y_1 = y_2 = \frac{1}{2} \implies Z = 4$

Hence, the integrality gap for the IP problem for MaxSat is

$$\min_{\psi} \left\{ \frac{Z^*_{IP_\psi}}{Z^*_{LP_\psi}} \right\} \leq \frac{Z^*_{IP_\phi}}{Z^*_{LP_\phi}} = \frac{3}{4}$$

On the other hand, the proof that $Rand_f$ is a $\frac{3}{4}$-approx. alg. shows that for any instance ψ of MaxSat, $Rand_f(\psi) \geq \frac{3}{4} Z^*_{LP_\psi}$. Hence,

$$\frac{Z^*_{IP_\psi}}{Z^*_{LP_\psi}} \geq \frac{Rand_f(\psi)}{Z^*_{LP_\psi}} \geq \frac{3}{4}$$

Hence, the integrality gap is exactly $\frac{3}{4}$.

The upper bound of $\frac{3}{4}$ on the integrality gap shows that we cannot prove an approx. factor better than $\frac{3}{4}$, if the approximation guarantee is based on a comparison to Z^*_{LP}:

$$\min_{\psi} \left\{ \frac{ALG(\psi)}{Z^*_{LP_\psi}} \right\} \leq \min \left\{ \frac{OPT(\psi)}{Z^*_{LP_\psi}} \right\} \leq \frac{3}{4}$$

Set Cover

Techniques: (with Set Cover as an example)
- Solve LP and round solution (Sec 1.3 + 1.7)
- Primal-dual alg.: combinatorial alg.
  based on LP formulation (Sec. 1.4 + 1.5)
- Greedy alg. (Sec. 1.6)

Section 1.2 : Set Cover as an LP

Set Cover

Input:

$E = \{e_1, e_2, \ldots, e_n\}$

$\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where

$S_j \subseteq E$ has weight w_j.

Objective: Find a cheapest possible subset of $\mathcal{S}$ covering all elements

OPT : value (total weight) of optimum solution

Ex:
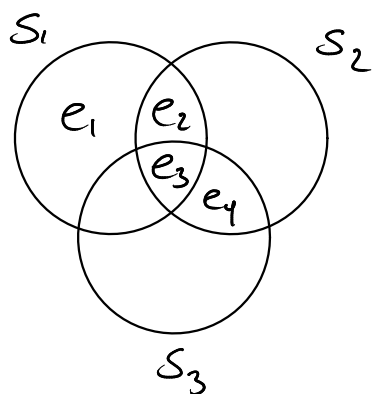
$w_1 = 1$

$w_2 = 2$

$w_3 = 3$

$\{S_1, S_2\}$ is a sol. of total weight 3.

This is optimal, so OPT $= 3$ for this instance of Set Cover.

To cover e_1, we need S_1

—"— e_2 —"— S_1 or S_2

—"— e_3 —"— S_1, S_2 or S_3

—"— e_4 —"— S_2 or S_3

IP-formulation:

$$\min \quad X_1 W_1 + X_2 W_2 + X_3 W_3$$

$$\text{s.t.} \quad X_1 \geq 1$$

$$X_1 + X_2 \geq 1$$

$$X_1 + X_2 + X_3 \geq 1$$

$$X_2 + X_3 \geq 1$$

$$X_1, X_2, X_3 \in \{0, 1\}$$

More generally:

$$\min \quad \sum_{j=1}^{m} x_j w_j$$

$$\text{s.t.} \quad \sum_{j: e_i \in S_j} x_j \geq 1, \qquad i = 1, 2, \ldots, n$$

$$x_j \in \{0, 1\}, \qquad j = 1, 2, \ldots, m$$

Z_{IP}^* : optimum solution value, i.e., $Z_{IP}^* = OPT$

$$\min \quad \sum_{j=1}^{m} x_j w_j$$

$$\text{s.t.} \quad \sum_{j: e_i \in S_j} x_j \geq 1, \qquad i = 1, 2, \ldots, n$$

$$0 \leq x_j \leq 1, \qquad j = 1, 2, \ldots, m$$

redundant

Z_{LP}^* : Optimum solution value

Note that $\boxed{Z_{LP}^* \leq Z_{IP}^* = OPT}$

Section 1.3: A deterministic rounding algo.

The *frequency* of an *element* e is the #sets containing e:

$$f_e = |\{ S \in \mathcal{S} \mid e \in S \}|$$

The *frequency* of an *instance* of Set Cover:

$$f = \max_{e \in E} \{ f_e \}$$

Alg. 1 for Set Cover: LP-rounding

$$\vec{x^*} \leftarrow \text{opt. sol. to LP}$$
$$\mathbb{I} \leftarrow \{ j \mid x_j^* \geq \tfrac{1}{f} \}$$

We prove that Alg 1 produces a set cover (Lemma 1.5) of total weight $\leq f \cdot OPT$ (Thm 1.6)

$\{ S_j \mid j \in I \}$ is a set cover

Proof:

For each $e_i \in E$, $\sum_{j: e_i \in S_j} x_j \geq 1$.

Since $\sum_{j: e_i \in S_j} x_j$ has at most f terms, at least one of the terms is at least $\frac{1}{f}$.

Thus, there is a set S_j s.t. $e_i \in S_j$ and $x_j \geq \frac{1}{f}$.

This j is included in I $\qquad\qquad\square$

Alg. 1 is an f-approx. algo. for Set Cover.

Proof:

Correct by Lemma 1.5

Poly, since LP-solving is poly.

Approx. factor f:

Each x_j is rounded up to 1, only if it is already at least $\frac{1}{f}$.

Thus, each x_j is multiplied by at most f, i.e.,

$$\sum_{j \in I} w_j \leq \sum_{j \in I} f \cdot x_j^* \cdot w_j \leq \sum_{j=1}^{m} f \cdot x_j^* \cdot w_j = f \cdot Z_{LP}^* \leq f \cdot OPT$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

The Vertex Cover problem is a special case of
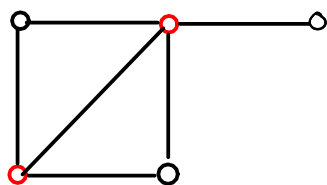Set Cover:

Vertex Cover

Input:

$$G = (V, E)$$

Objective:

Find a min. card. vertex set $C \subseteq V$
s.t. each edge $e \in E$ has at least one
endpoint in C.

Ex:

With $\mathcal{S} = V$ and $E = E$,
Alg. 1 is a 2-approx. alg. for Vertex Cover.

One of the exercises for Tuesday:
Write down LP for Vertex Cover.

$$\boxed{\text{Section 1.4: The dual LP}}$$

What is a dual?

Ex:

$$\boxed{\begin{array}{l} P \\ \min \quad 7x_1 + x_2 + 5x_3 \\ \text{s.t.} \quad x_1 - x_2 + 3x_3 \geqslant 10 \\ \qquad\quad 5x_1 + 2x_2 - x_3 \geqslant 6 \\ \qquad\quad x_1, x_2, x_3 \geqslant 0 \end{array}}$$

Primal

$$7x_1 + x_2 + 5x_3 \geqslant x_1 - x_2 + 3x_3 \geqslant 10$$

$$7x_1 + x_2 + 5x_3 \geqslant x_1 - x_2 + 3x_3 + 5x_1 + 2x_2 - x_3$$
$$\geqslant \qquad 10 \qquad + \quad 6 \qquad = 16$$

$$7x_1 + x_2 + 5x_3 \geqslant 2(x_1 - x_2 + 3x_3) + 5x_1 + 2x_2 - x_3$$
$$\geqslant \qquad 2 \cdot 10 \qquad + \quad 6 \qquad = 26$$

To find a largest possible lower bound on $7x_1 + x_2 + 5x_3$, we should determine y_1 and y_2 maximizing $10y_1 + 6y_2$, under the constraints that

$$7x_1 + x_2 + 5x_3 \overset{(*)}{\geqslant} \underbrace{y_1(x_1 - x_2 + 3x_3)}_{\geqslant 10} + \underbrace{y_2(5x_1 + 2x_2 - x_3)}_{\geqslant 6}$$
$$= \underbrace{(y_1 + 5y_2)}_{\leqslant 7} x_1 + \underbrace{(-y_1 + 2y_2)}_{\leqslant 1} x_2 + \underbrace{(3y_1 - y_2)}_{\leqslant 5} x_3$$

necessary to satisfy (*)

and $\underbrace{y_1, y_2, y_3 \geqslant 0}_{\text{otherwise}}$
"$\geqslant$" becomes "$\leqslant$"

Thus, we arrive at the following problem:

D

$$\max \quad 10y_1 + 6y_2$$

$$\text{s.t.} \quad y_1 + 5y_2 \leq 7$$
$$-y_1 + 2y_2 \leq 1$$
$$3y_1 - y_2 \leq 5$$
$$y_1, y_2 \geq 0$$

Dual

In general:

Primal:

$$\min \quad c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$
$$\text{s.t.} \quad a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{in} x_n \geq b_i \quad, \quad i = 1, 2, \ldots, m$$
$$x_j \geq 0 \quad, \quad j = 1, 2, \ldots, n$$

Dual:

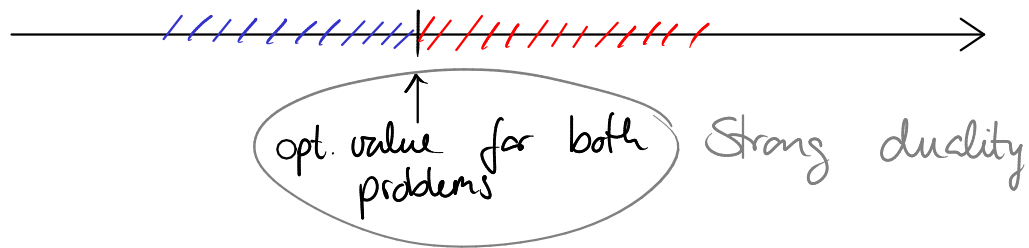$$\max \quad b_1 y_1 + b_2 y_2 + \ldots + b_m y_m$$
$$\text{s.t.} \quad a_{1j} y_1 + a_{2j} y_2 + \ldots + a_{mj} y_m \leq c_1 \quad, \quad j = 1, 2, \ldots, n$$
$$y_i \geq 0, \quad i = 1, 2, \ldots, m$$

Returning to the example above:

The constraints of D ensure that the value of any sol. to D is a lower bound on the value of any sol. to P, i.e., for any pair x, y of sol. to P and D resp.,

$$10 y_1 + 6 y_2 \le 7x_1 + x_2 + 5x_3$$ Weak duality

opt. value for both problems Strong duality

Consider again the inequality leading to the constraints of the dual:

$$7x_1 + x_2 + 5x_3 \ge y_1(x_1 - x_2 + 3x_3) + y_2(5x_1 + 2x_2 - x_3)$$

Looking at the right hand side:

$$y_1 = 0 \lor x_1 - x_2 + 3x_3 = 10 \iff \qquad\qquad y_2 = 0 \lor 5x_1 + 2x_2 - x_3 = 6 \iff$$

$$\underbrace{y_1(x_1 - x_2 + 3x_3)}_{= 10 y_1} + \underbrace{y_2(5x_1 + 2x_2 - x_3)}_{= 6y_2}$$

$$= \underbrace{(y_1 + 5y_2) x_1}_{= 7x_1} + \underbrace{(-y_1 + 2y_2) x_2}_{= x_2} + \underbrace{(3y_1 - y_2) x_3}_{= 5x_3}$$

$$\Updownarrow \qquad\qquad\qquad \Updownarrow \qquad\qquad\qquad \Updownarrow$$

$$y_1 + 5y_2 = 7 \qquad\qquad -y_1 + 2y = 1 \qquad\qquad 3y_1 - y_2 = 5$$

$$\lor \ x_1 = 0 \qquad\qquad\quad \lor \ x_2 = 0 \qquad\qquad\quad \lor \ x_3 = 0$$

Thus,

$$7x_1 + x_2 + 5x_3 = 10y_1 + 6y_2$$

$\Updownarrow$

Complementary
Slackness
Conditions
(c.s.c.)

$$\begin{cases} x_1 = 0 \quad \vee \quad y_1 + 5y_2 = 7 \\ x_2 = 0 \quad \vee \quad -y_1 + 2y_2 = 1 \\ x_3 = 0 \quad \vee \quad 3y_1 - y_2 = 5 \end{cases} \right\} \text{ primal c.s.c.}$$

$$\begin{cases} y_1 = 0 \quad \vee \quad x_1 - x_2 + 3x_3 = 10 \\ y_2 = 0 \quad \vee \quad 5x_1 + 2x_2 - x_3 = 6 \end{cases} \right\} \text{ dual c.s.c.}$$

Since $p \vee q \equiv \neg p \Rightarrow q$, this can also be written as:

$$7x_1 + x_2 + 5x_3 = 10y_1 + 6y_2$$

$\Updownarrow$

Complementary
Slackness
Conditions
(c.s.c.)

$$\begin{cases} x_1 > 0 \quad \Rightarrow \quad y_1 + 5y_2 = 7 \\ x_2 > 0 \quad \Rightarrow \quad -y_1 + 2y_2 = 1 \\ x_3 > 0 \quad \Rightarrow \quad 3y_1 - y_2 = 5 \end{cases} \right\} \text{ primal c.s.c.}$$

$$\begin{cases} y_1 > 0 \quad \Rightarrow \quad x_1 - x_2 + 3x_3 = 10 \\ y_2 > 0 \quad \Rightarrow \quad 5x_1 + 2x_2 - x_3 = 6 \end{cases} \right\} \text{ dual c.s.c.}$$

By The Strong Duality Theorem (which we will not prove), there exist solutions fulfilling the c.s.c.

Moreover, if the c.s.c. are „close" to being satisfied, the values of the primal and dual sol. are „close":

Relaxed Complementary Slackness Conditions

$$
\begin{cases}
x_1 > 0 & \Rightarrow & y_1 + 5y_2 \geq 7/b \\
x_2 > 0 & \Rightarrow & -y_1 + 2y_2 \geq 1/b \\
x_3 > 0 & \Rightarrow & 3y_1 - y_2 \geq 5/b \\[2mm]
y_1 > 0 & \Rightarrow & x_1 - x_2 + 3x_3 \leq 10c \\
y_2 > 0 & \Rightarrow & 5x_1 + 2x_2 - x_3 \leq 6c
\end{cases}
$$

$$\Downarrow$$

$$7x_1 + x_2 + 5x_3 \leq bc\,(10y_1 + 6y_2)$$

Proof:

$$(y_1 + 5y_2)\,x_1 + (-y_1 + 2y_2)\,x_2 + (3y_1 - y_2)\,x_3$$

$$\geq \frac{7}{b}x_1 + \frac{1}{b}x_2 + \frac{5}{b}x_3 \;,\quad \text{by the Primal relaxed c.s.c.}$$

$$= \frac{1}{b}\left(7x_1 + x_2 + 5x_3\right)$$

$$\Downarrow$$

$$7x_1 + x_2 + 5x_3 \leq b\Big((y_1 + 5y_2)\,x_1 + (-y_1 + 2y_2)\,x_2 + (3y_1 - y_2)\,x_3\Big)$$

$$= b\Big((x_1 - x_2 + 3x_3)\,y_1 + (5x_1 + 2x_2 - x_3)\,y_2\Big)$$

$$\leq b\big(10c\,y_1 + 6c\,y_2\big)\;,\quad \text{by the Dual r.c.s.c.}$$

$$= bc\,(10y_1 + 6y_2)$$

Set Cover Primal

$$\min \quad \sum_{j=1}^{m} x_j w_j$$

$$\text{s.t.} \quad \sum_{j:\, e_i \in S_j} x_j \geq 1, \quad i = 1, 2, \ldots, n$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, m$$

Covering problem

Set Cover Dual

$$\max \quad \sum_{i=1}^{n} y_i$$

$$\text{s.t.} \quad \sum_{e_i \in S_j} y_i \leq w_j, \quad j = 1, 2, \ldots, m$$

$$y_i \geq 0, \quad i = 1, 2, \ldots, n$$

Packing problem

The primal problem is a covering problem:
Each element has to be covered by at least one set.

The dual problem can be viewed as a packing problem:
Each set S_j has a capacity of w_j.
We interpret y_i as the weight of e_i, and the total weight of elements in S_j must not exceed w_j.
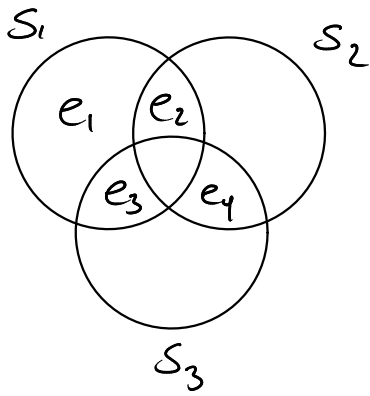
Recall that the dual is constructed such that the value of any solution to the dual is a lower bound on the value of any solution to the primal:

$$Z_{Primal} \geq Z_{Dual} \quad \text{(weak duality property)}$$

In fact,

$$Z^*_{Primal} = Z^*_{Dual} \quad \text{(strong duality property)}$$

Ex:

S_1 S_2

e_1 e_2
e_3 e_4

S_3

$w_1 = 1$
$w_2 = 2$
$w_3 = 3$

Primal:

$$\min \quad x_1 + 2x_2 + 3x_3$$

$$\text{s.t.} \quad x_1 \qquad\qquad \geqslant 1$$
$$x_1 + x_2 \qquad \geqslant 1$$
$$x_1 + \quad x_3 \geqslant 1$$
$$x_2 + x_3 \geqslant 1$$
$$x_1, x_2, x_3 \geqslant 0$$

$OPT = 3:$

$x_1 = x_2 = 1$

Dual:

$$\max \quad y_1 + y_2 + y_3 + y_4$$

$$\text{s.t.} \quad y_1 + y_2 + y_3 \qquad \leq 1$$
$$y_2 + \quad y_4 \leq 2$$
$$y_3 + y_4 \leq 3$$
$$y_1, y_2, y_3, y_4 \geqslant 0$$

$OPT = 3:$

$y_1 = 1$ or $y_3 = 1$
$y_4 = 2$ $y_4 = 2$

$x_1 = 1, \ x_2 = \frac{2}{3}, \ x_3 = \frac{1}{3}$

$x_1 = x_2 = 1$

$x_1 = x_2 = x_3 = 1$

$y_1 = y_2 = y_3 = y_4$

$y_2 = y_4 = 1, \ y_1 = y_3 = 0$

$y_1 = 1, \ y_4 = 2, \ y_2 = y_3 = 0$

$y_3 = 1, \ y_4 = 2, \ y_1 = y_2 = 0$
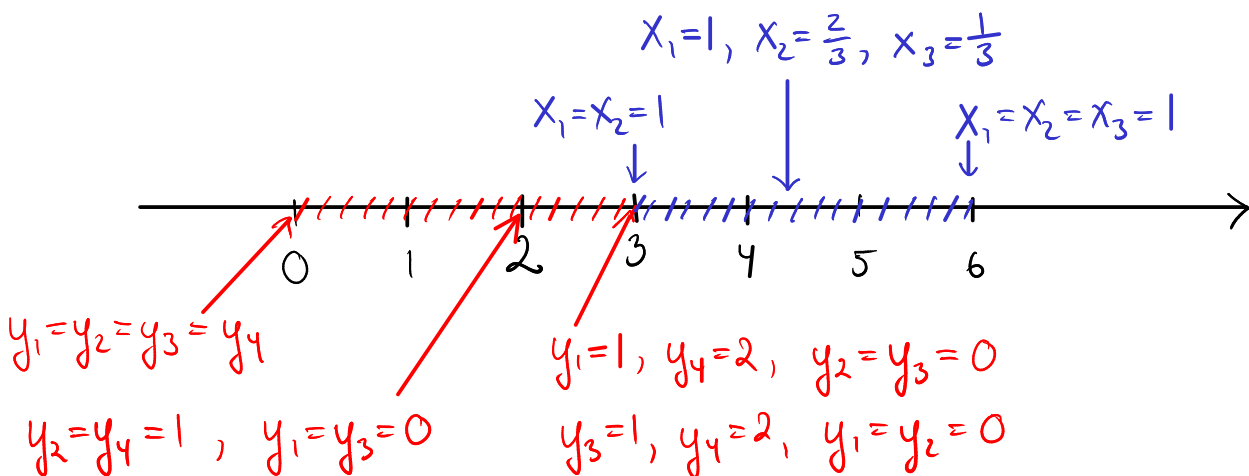
Now back to approximation algorithms for Set Cover.

We have seen an alg. (Alg.1) which solves the LP relaxation and then does deterministic rounding.

We will now look at an alg. (Alg.2) which solves the **dual** of the LP relaxation.
Then each set corresponding to a tight constraint is selected:

Alg.2 for Set Cover

$\vec{y}^* \leftarrow$ opt. sol. to dual LP

$I' \leftarrow \{ j \mid \sum_{e_i \in S_j} y_i = w_j \}$

Let's see how the alg. would work on the example from before. Recall that there were two optimal solutions to the dual:

$y_1 = 1, \ y_2 = y_3 = 0, \ y_4 = 2$ and $y_1 = y_2 = 0, \ y_3 = 1, \ y_2 = 2$.

In the ex. above:

with $y_1^* = 1$, $y_4^* = 2$, Alg 2 would choose S_1 and S_2 with a total weight of 3.

with $y_3^* = 1$, $y_4^* = 2$, Alg. 2 would choose $S_1, S_2,$ and S_3 with a total weight of 6.

The first solution is optimal, and the latter is a 2-approximation (i.e., an f-approximation).

Alg. 2 is an f-approximation algo. on this example:
If the algo. chooses $S_1, S_2,$ and S_3, the total weight is $W = w_1 + w_2 + w_3$, and
$$w_1 + w_2 + w_3 = (y_1^* + y_2^* + y_3^*) + (y_2^* + y_4^*) + (y_3^* + y_4^*),$$
since the algo. chooses exactly those sets that have LHS = RHS.
Since each y_i is present in at most f constraints,
$$W \leq f \cdot (y_1^* + y_2^* + y_3^* + y_4^*)$$
$$= f \cdot Z_{dual}^*$$
$$\leq f \cdot Z_{primal}^*, \text{ by the weak duality property}$$
$$= f \cdot OPT$$

Before giving the general proof that Alg. 2 is an f-approx. alg., we show that it always produces a valid set cover:

Lemma 1.7

Alg. 2 produces a set cover

Proof:

Assume for the sake of contradiction that some element e_k is not covered by $\{S_j \mid j \in I'\}$.

Then $\sum_{e_i \in S_j} y_i < w_j$ for all S_j containing e_k.

These are exactly the constraints involving y_k.

Thus, none of the constraints involving y_k are tight.

This means that y_k can be increased without violating any constraint.

Since this will increase the value $\sum_{i=1}^{n} y_i$ of the sol., we conclude that

the solution $\vec{y}$ was not optimal. $\square$

<u>Ex:</u>

In the ex. above, assume

$$y_1 = y_2 = y_3 = 0$$
$$y_4 = 2$$

Then, only the second constraint is tight, so only S_2 is picked:

$$y_1 + y_2 + y_3 \qquad = 0 < 1$$
$$y_2 \qquad + y_4 = 2$$
$$y_3 + y_4 = 2 < 3$$

e_4 is not covered, since none of the two constraints involving y_4 are tight.

We can increase y_4 from 0 to 1 without violating any constraints
(Then two other constraints become tight.)
This increases the sol. value from 2 to 3.
Thus, the sol. above was not optimal.

Or we could increase y_1 from 0 to 1.
Then only the first constraint becomes tight, resulting in an optimal solution.

This illustrates the idea of the primal-dual alg of Section 1.5.

We now give a more formal proof that
Alg 2 is an f-approximation algo.

Thm 1.8
Alg.2 is an f-approx. algo.

Proof :

The correctness follows from Lemma 1.7.

Approx. guarantee :

$$\sum_{j \in I'} w_j = \overbrace{\sum_{j \in I'} \sum_{e_i \in S_j} y_i^*}^{\text{y_i^* appears once for each set in the sol. containing e_i}}$$

$$= \sum_{i=1}^{n} \underbrace{\left| \{ j \in I' \mid e_i \in S_j \} \right|}_{\text{\#sets in the sol. containing e_i}} \cdot y_i^*$$

$$\leq \sum_{i=1}^{n} \underbrace{f_{e_i}}_{\text{\#sets containing e_i}} \cdot y_i^*$$

$$\leq \sum_{i=1}^{n} f \cdot y_i^*$$

$$= f \cdot Z_{dual}^*$$

$$\leq f \cdot Z_{primal}^* \text{, by the weak duality property}$$

$$= f \cdot \text{OPT}$$

□

Note that for proving the above theorem, we could also use the relaxed C.S.C. (with $b=1$, $c=f$), since

- $\sum\limits_{j : e_i \in S_j} x_j \leq f$, for all $i = 1, 2, \ldots, n$, by the def. of f.

- $x_j = 1 \implies \sum\limits_{e_i \in S_j} y_i = w_j$, by the def. of the alg.

Both Alg. 1 and Alg. 2 rely on solving an LP (optimally). In Section 1.5, we will study a more time efficient alg.

The key observation is that in the proof of Thm 1.8, we did not need the fact that $\vec{y}^*$ is optimal, since $z_{dual} \leq z_{primal}^*$, for any feasible dual solution.

Thus, the crux is to obtain an index set I'' s.t.

- $\{ S_j \mid j \in I'' \}$ is a set cover

- $\sum\limits_{j \in I''} w_j = \sum\limits_{j \in I''} \sum\limits_{e_i \in S_j} y_i$, for some feasible sol. $\vec{y}$ to the dual LP

without solving an LP optimally.

Alg. 1.1 for Set Cover: Primal - Dual

$I'' \leftarrow \emptyset$

$\vec{y} \leftarrow \vec{0}$

While $\exists e_k \notin \bigcup_{j \in I''} S_j$

　　Increase y_k until some constraint, ℓ, becomes tight, i.e., $\sum_{e_i \in S_\ell} y_i = w_\ell$

　　　$I'' \leftarrow I'' \cup \{\ell\}$

Note that $e_k \in S_\ell$

Thm 1.9

Alg. 1.1 is an f-approx. alg. for Set Cover

Proof:

Alg. 3 produces a set cover, since as long as some element is not covered, the corresponding dual constraints are non-tight.

The approx. guarantee follows from the same calculations as in the proof of Thm. 1.8, since

$$\sum_{j \in I''} w_j = \sum_{j \in I''} \sum_{e_i \in S_j} y_i \leq f \cdot Z_{dual} \leq f \cdot Z_{dual}^*$$

In contrast to Alg. 2 from Section 1.4,
Alg. 1.1 does not necessarily produce an optimal
dual solution:

In the example above, it might do the following.

$y_2 \leftarrow 1$ (S_1 is picked, e_4 still uncovered)
$y_4 \leftarrow 1$ (S_2 is picked)

(This is fine, since the proof of Thm. 1.8
does not use that $\sum y_i = OPT$, only that
$\sum y_i \leq OPT$, which is true for _any_ feasible
sol. to the dual.)

Section 1.6: A Greedy Algorithm

A natural greedy choice would be to „pay" as little as possible for each additional covered element:

Alg 1.2 for Set Cover: Greedy

$I \leftarrow \emptyset$

For $j \leftarrow 1$ to m

$\quad \hat{S}_j \leftarrow S_j$ $\qquad$ (uncovered part of S_j)

While $\{S_j \mid j \in I\}$ is not a set cover

$\quad \ell \leftarrow \underset{j : \hat{S}_j \neq \emptyset}{\arg\min} \dfrac{w_j}{|\hat{S}_j|}$ $\quad$ (S_ℓ: set with smallest cost per uncovered element)

$\quad I \leftarrow I \cup \{\ell\}$

$\quad$ For $j \leftarrow 1$ to m

$\quad\quad \hat{S}_j \leftarrow \hat{S}_j - S_\ell$

The greedy alg. is an H_n-approx. alg

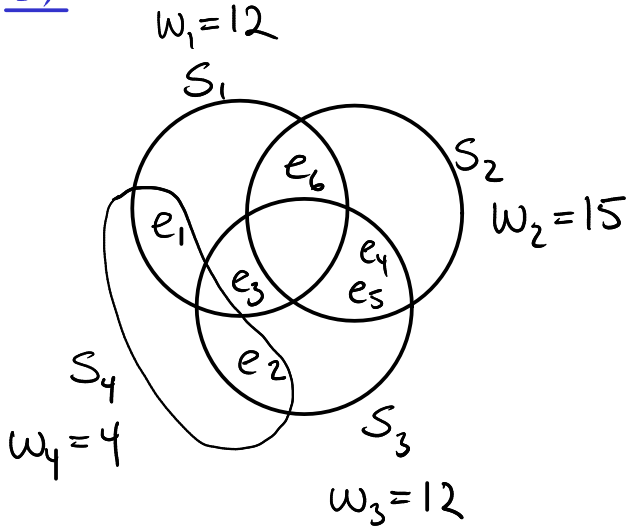Recall: $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \approx \ln(n)$

It is „likely" that no significantly better approx. ratio can be obtained:

Thm 1.13:

Approx. factor $\frac{\ln n}{c}$, $c > 1$, for unweighted Set Cover

$\implies n^{O(\lg \lg n)}$ - approx alg. for NPC

<u>Ex:</u>

$W_1 = 12$
S_1

e_6

S_2
$W_2 = 15$

e_1

e_4
e_3
e_5

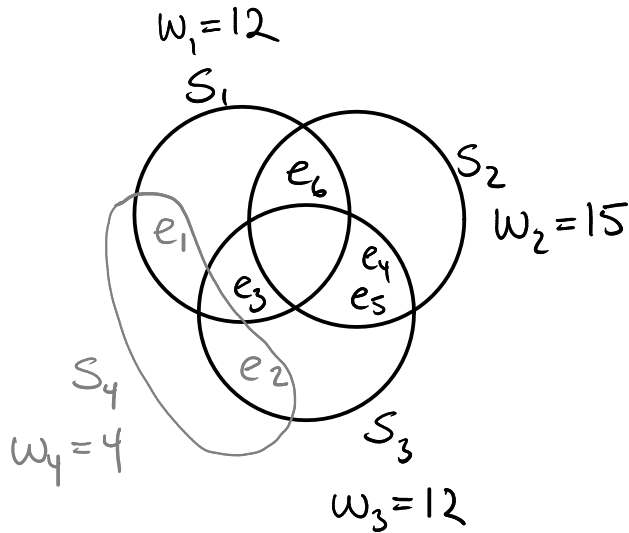S_4
e_2

$W_4 = 4$
S_3

$W_3 = 12$

$$\frac{W_1}{|S_1|} = \frac{12}{3} = 4$$

$$\frac{W_2}{|S_2|} = \frac{15}{3} = 5$$

$$\frac{W_3}{|S_3|} = \frac{12}{4} = 3$$

$$\frac{W_4}{|S_4|} = \frac{4}{2} = \boxed{2} \leftarrow \text{price}$$
per element in
first iteration

Pick S_4

$W_1 = 12$
S_1

e_6

S_2
$W_2 = 15$

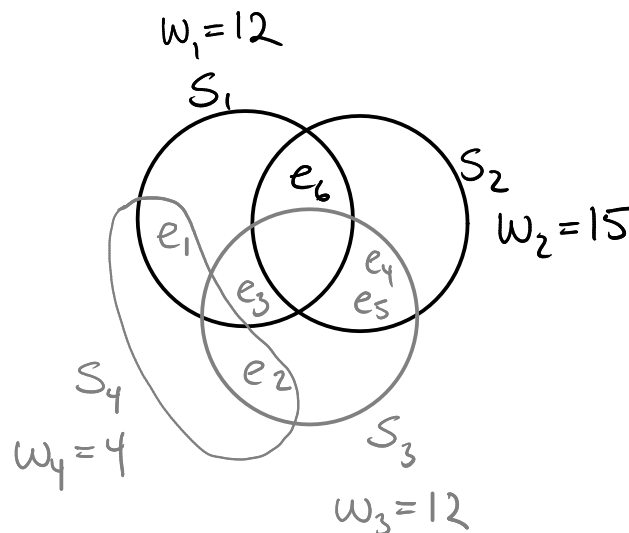e_1

e_4
e_3
e_5

S_4
e_2

$W_4 = 4$
S_3

$W_3 = 12$

$$\frac{W_1}{|\hat{S}_1|} = \frac{12}{2} = 6$$

$$\frac{W_2}{|\hat{S}_2|} = \frac{15}{3} = 5$$

$$\frac{W_3}{|\hat{S}_3|} = \frac{12}{3} = \boxed{4} \leftarrow \text{price}$$
per element in
second it.

Pick S_3

$W_1 = 12$
S_1

e_6

S_2
$W_2 = 15$

e_1

e_4
e_3
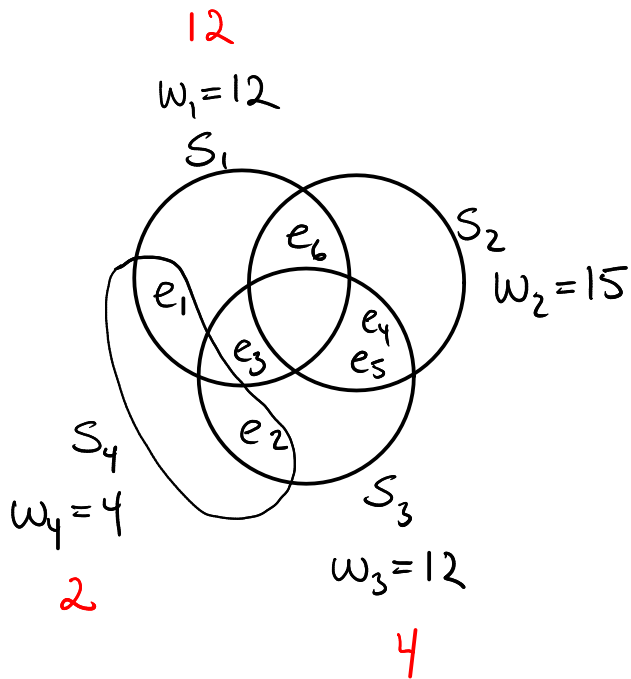e_5

S_4
e_2

$W_4 = 4$
S_3

$W_3 = 12$

$$\frac{W_1}{|\hat{S}_1|} = \frac{12}{1} = \boxed{12} \leftarrow \text{price}$$
per element in
third it.

$$\frac{W_2}{|\hat{S}_2|} = \frac{15}{1} = 15$$

Pick S_1

12

$w_1 = 12$

S_1

e_6

S_2

e_1

$W_2 = 15$

e_4

e_5

e_3

e_2

S_4

S_3

$w_4 = 4$

2

$w_3 = 12$

4

Greedy = 28

$= w_4 + w_3 + w_1$

$= 4 + 12 + 12$

$= 2 + 2 + 4 + 4 + 4 + 12$

$= \sum_{i=1}^{6} \text{price}(e_i)$

OPT = 24

$= w_3 + w_1$

$= 12 + 12$

$= 4 + 4 + 4 + 4 + 6 + 6$

We will now use this ex. to illustrate the proof of Thm 1.11 stating that Greedy is an H_n-approx. alg. :

<u>H_6 – approximation:</u> $\left(H_6 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} = \frac{147}{60} < 2.5 \right)$
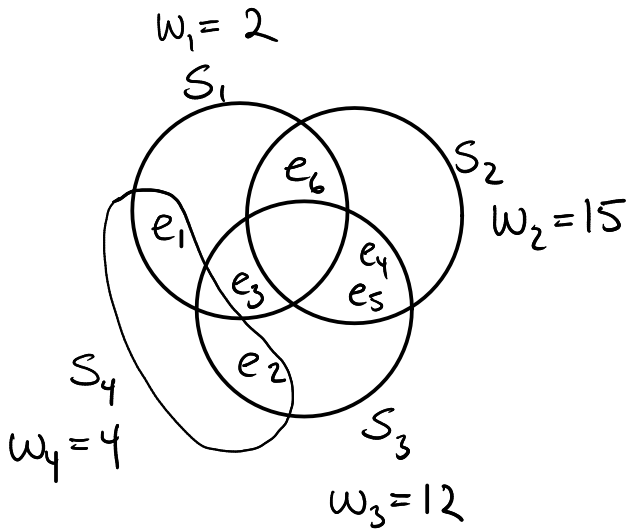
$OPT \geqslant 6 \cdot price(e_1)$

$OPT \geqslant 6 \cdot price(e_2)$

since S_4 gives the best average weight per element.
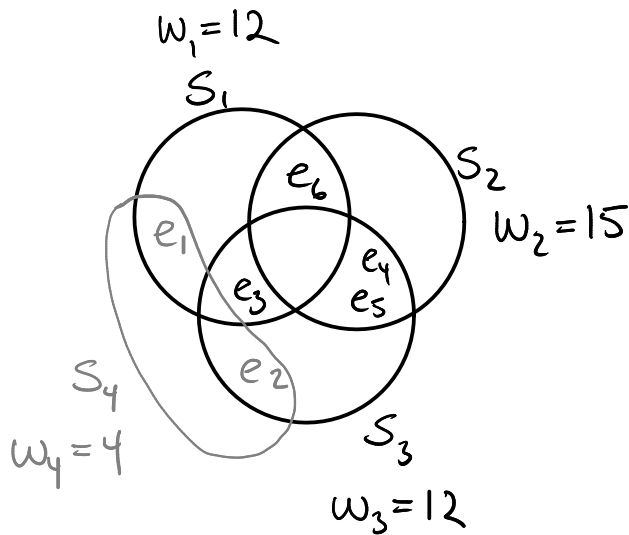
S_4 cannot cover any of the elements e_3, e_4, e_5, e_6. Thus, the average weight of these elements cannot be lower than $price(e_5)$, even for OPT:

$OPT \geqslant 4 \cdot price(e_3)$

$OPT \geqslant 4 \cdot price(e_4)$

$OPT \geqslant 4 \cdot price(e_5)$

Similarly:

$OPT \geqslant price(e_6)$

$w_1 = 12$
S_1
e_6
S_2
$w_2 = 15$
e_1
e_4
e_3
e_5
S_4
e_2
$w_4 = 4$
S_3
$w_3 = 12$

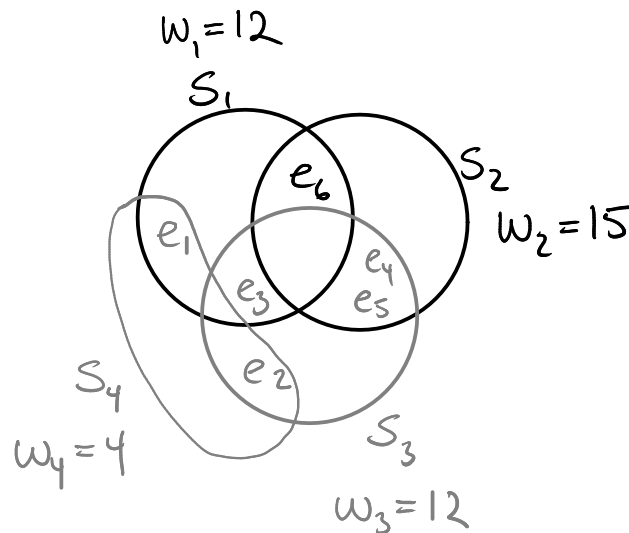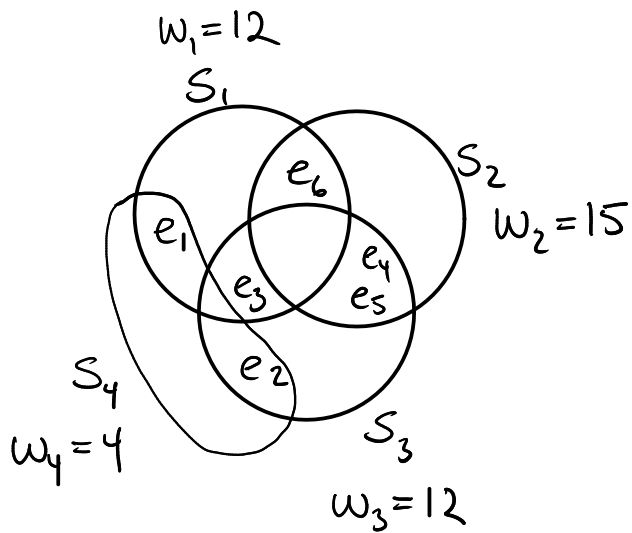$$\text{OPT} \geq 6 \cdot \text{price}(e_1) \qquad \Longleftrightarrow \qquad \text{price}(e_1) \leq \frac{\text{OPT}}{6}$$

$$\text{OPT} \geq 6 \cdot \text{price}(e_2) \qquad \Longleftrightarrow \qquad \text{price}(e_2) \leq \frac{\text{OPT}}{6}$$

$$\text{OPT} \geq 4 \cdot \text{price}(e_3) \qquad \Longleftrightarrow \qquad \text{price}(e_3) \leq \frac{\text{OPT}}{4}$$

$$\text{OPT} \geq 4 \cdot \text{price}(e_4) \qquad \Longleftrightarrow \qquad \text{price}(e_4) \leq \frac{\text{OPT}}{4}$$

$$\text{OPT} \geq 4 \cdot \text{price}(e_5) \qquad \Longleftrightarrow \qquad \text{price}(e_5) \leq \frac{\text{OPT}}{4}$$

$$\text{OPT} \geq \text{price}(e_6) \qquad \Longleftrightarrow \qquad \text{price}(e_6) \leq \text{OPT}$$

$$\text{Greedy} = \sum_{i=1}^{6} \text{price}(e_i)$$

$$\leq \frac{\text{OPT}}{6} + \frac{\text{OPT}}{6} + \frac{\text{OPT}}{4} + \frac{\text{OPT}}{4} + \frac{\text{OPT}}{4} + \frac{\text{OPT}}{1}$$

$$\leq \frac{\text{OPT}}{6} + \frac{\text{OPT}}{5} + \frac{\text{OPT}}{4} + \frac{\text{OPT}}{3} + \frac{\text{OPT}}{2} + \frac{\text{OPT}}{1}$$

$$= H_6 \cdot \text{OPT}$$

Thm 1.11

Alg. 1.2 is an H_n-approx. alg. for Set Cover

Proof:

n_k: #uncovered elements at the beginning of the k'th iteration

Above ex.: $n_1 = 6$, $n_2 = 4$, $n_3 = 1$, $n_4 = 0$

$n_1 - n_2 = 2$, $n_2 - n_3 = 3$, $n_3 - n_4 = 1$

Any algorithm, including OPT, has to cover these n_k elements using only sets in $\mathcal{S} - \{S_j \mid j \in I\}$, since none of them are contained in $\{S_j \mid j \in I\}$.

Hence, there must be at least one element with a price of at most OPT/n_k. Otherwise, OPT would not be able to cover the n_k elements (and certainly not all n elements) at a cost of only OPT.

Hence, the $n_k - n_{k+1}$ elements covered in iteration k cost at most $(n_k - n_{k+1}) OPT/n_k$ in total.

Thus, the cost of the set cover produced by the greedy alg. is

$$\sum_{j \in I} w_j \le \sum_{k=1}^{r} \frac{n_k - n_{k+1}}{n_k} \; OPT \qquad \text{\#iterations}$$

$$= OPT \sum_{k=1}^{r} (n_k - n_{k+1}) \cdot \frac{1}{n_k}$$

$$\le OPT \sum_{k=1}^{r} \left(\underbrace{\frac{1}{n_k} + \frac{1}{n_k - 1} + \ldots + \frac{1}{n_{k+1} + 1}}_{n_k - n_{k+1} \text{ terms that are each} \ge \frac{1}{n_k}} \right)$$

$$= OPT \sum_{s=1}^{n} \frac{1}{s}$$

$$= OPT \cdot H_n \qquad \qquad \square$$

Let $g = \max \{ |S_i| \mid S_i \in \mathcal{S} \}$.

Thm 1.12

Alg. 1.2 is an H_g-approx. alg. for Set Cover

Proof: By Dual Fitting:

Consider the dual D of the LP for Set Cover.
We will construct

- an infeasible solution $\vec{y}$ and
- a feasible solution $\vec{y'}$

such that

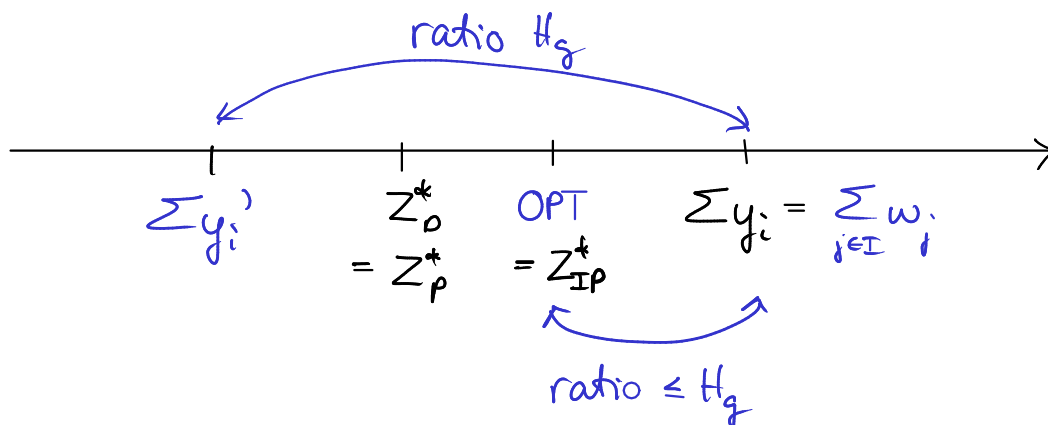- $\sum\limits_{i=1}^{n} y_i = \sum\limits_{j \in I} w_j$ (obtained, if $y_i = \text{price}(e_i)$)

- $y_i' = \frac{1}{H_g} \cdot y_i$

Then,

$$\sum_{j \in I} w_j = \sum_{i=1}^{n} y_i = H_g \sum_{i=1}^{n} y_i' \leq H_g \, z_D^* \leq H_g \cdot \text{OPT},$$

proving the claimed approximation factor. $\qquad \square$

ratio H_g

$\sum y_i'$ $z_D^* = z_P^*$ OPT $= z_{IP}^*$ $\sum y_i = \sum\limits_{j \in I} w_j$

ratio $\leq H_g$

For $1 \leq i \leq n$, let $y_i = \text{price}(e_i)$. Then, $\sum_{1 \leq i \leq n} y_i = \sum_{j \in I} w_j$.

Hence, we just need to show that $\vec{y}$ is feasible:

Consider an arbitrary set S_j.

Let a_k be #uncovered elements in S_j at the beginning of the k'th iteration.

Let S_p be the set chosen by Greedy in the k'th iteration.

S_p covers $a_k - a_{k+1}$ previously uncovered elements in S_j

$a_k - a_{k+1}$ elements

S_p
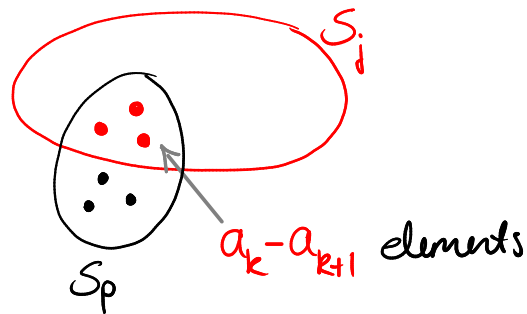
The price per elem. in S_j covered in the k'th it. is

$$\frac{w_p}{|\hat{S}_p|} \leq \frac{w_j}{|\hat{S}_j|} \leq \frac{w_j}{a_k}$$

since otherwise S_j would be a _more_ greedy choice. ↯

Thus,

$$\sum_{e_i \in S_j} y_i \leq \sum_{k=1}^{r} (a_k - a_{k+1}) \frac{w_j}{a_k}$$

$$\leq w_j \sum_{i=1}^{|S_j|} \frac{1}{i}, \text{ by the same arguments as in the proof of Thm 1.12.}$$

$$\leq w_j \sum_{i=1}^{g} \frac{1}{i}$$

$$= w_j \cdot H_g$$

Hence,

$$\sum_{e_i \in S_j} y_i' = \frac{1}{H_g} \sum_{e_i \in S_j} y_i \leq w_j$$
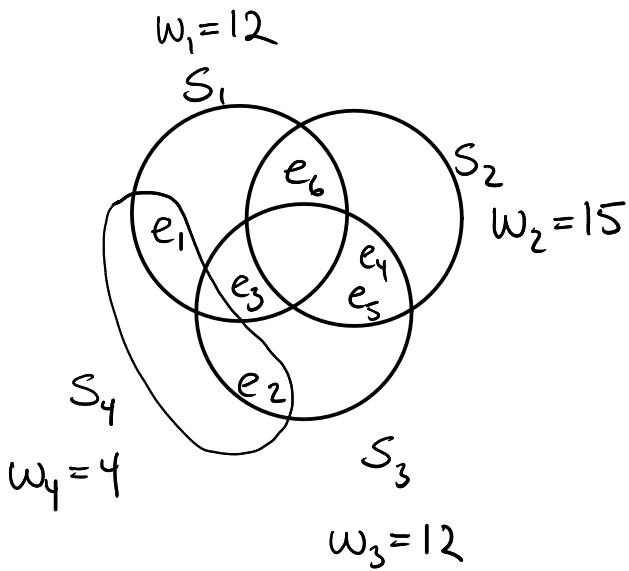
$\square$

Compare the proof of Thm 1.12 (dual fitting) to the proof of Thm 1.11:
- Simpler: Compare prices to w_j instead of OPT
- Stronger result: H_g instead of H_n
    (could also have been obtained with the technique of the proof of Thm 1.11)

Ex from before:

$g = 4 \implies H_{12} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12} < 2.1$

$y_1 = y_2 = 2$

$y_3 = y_4 = y_5 = 4$

$y_6 = 12$

$y_1' = y_2' = \frac{24}{25}$

$y_3' = y_4' = y_5' = \frac{48}{25}$

$y_6' = \frac{144}{25}$

Dual constraints:

$y_1' + y_3' + y_6' \leq 12$

$y_4' + y_5' + y_6' \leq 15$

$y_2' + y_3' + y_4' + y_5' \leq 12$

$y_1' + y_2' \leq 4$

$\frac{24}{25} + \frac{48}{25} + \frac{144}{25} = \frac{216}{25} < 12$

$\frac{48}{25} + \frac{48}{25} + \frac{144}{25} = \frac{240}{25} < 15$

$\frac{24}{25} + \frac{48}{25} + \frac{48}{25} + \frac{48}{25} = \frac{168}{25} < 12$

$\frac{24}{25} + \frac{24}{25} < 4$
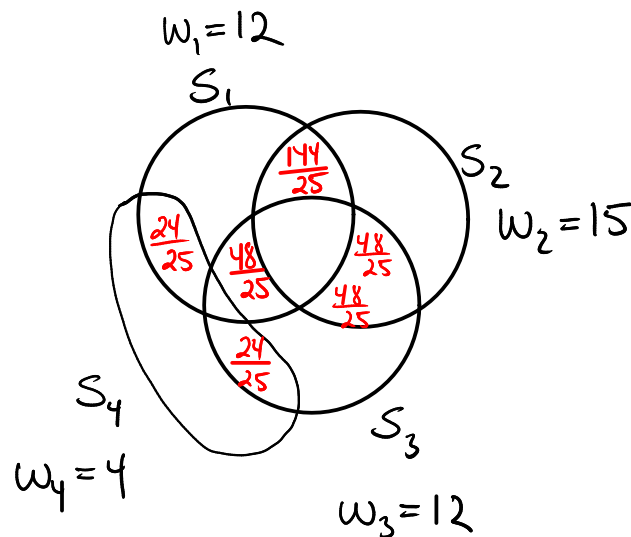
Is the upper bound of H_n tight?

If it is, the matching lower bound must come
from an instance with
- one set containing all elements
  (follows from the upper bound of H_g)
- only one additional element covered in each it.
  (otherwise, some of the terms in $\frac{1}{n} + \frac{1}{n-1} + \dots + 1$
  would be replaced by smaller terms.)

Ex:

$1+\varepsilon$

$\frac{1}{n}$ $\frac{1}{n-1}$ $\dots$ 1
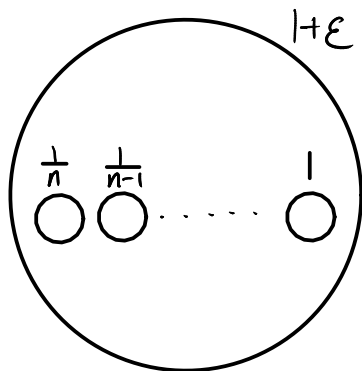
Summary

Greedy

H_n-approx.:

$$\text{price}(e_i) \leq \frac{\text{OPT}}{n-i} \quad , \quad i = 0, 1, \ldots, n-1$$

H_g-approx: (g: size of largest set)

Dual fitting:

$$y_i' \leftarrow \frac{\text{price}(e_i)}{H_g} \quad \text{is a feasible sol. to dual}$$

$\frac{\text{Greedy}}{H_g} \quad\quad \text{OPT} \quad\quad \text{Greedy}$
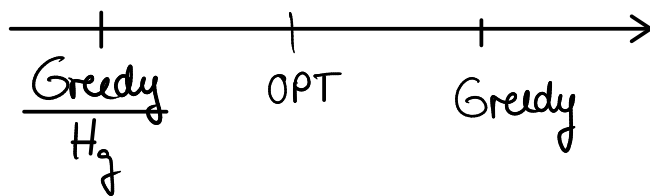
Section 1.7: Randomized Rounding

Alg RR_1

Solve LP
$I \leftarrow \emptyset$
For $j \leftarrow 1$ to m
    With probability x_j
        $I \leftarrow I \cup \{j\}$

$E[w(I)] = Z_{LP}^* \leq OPT$,
but the result is most likely <u>not</u> a set cover.

Alg RR_2

Solve LP
  $I \leftarrow \emptyset$
  For $i \leftarrow 1$ to $2 \cdot \ln(n)$
    For $j \leftarrow 1$ to m
        With probability x_j
            $I \leftarrow I \cup \{j\}$

$E[w(I)] \leq 2 \cdot \ln(n) \cdot Z_{LP}^* \leq 2 \cdot \ln(n) \cdot OPT$,
and with high prob. all elements are covered.
(Calculations below)

Alg RR_3

Solve LP
Repeat
    $I \leftarrow \emptyset$
    For $i \leftarrow 1$ to $2 \cdot \ln(n)$
        For $j \leftarrow 1$ to m
            With probability x_j
                $I \leftarrow I \cup \{j\}$
    Until $\{S_j \mid j \in I\}$ is a set cover
       and $W(I) \leq 4\ln(n) Z_{LP}^*$

$W(I) \leq 4 \cdot \ln(n) \cdot Z_{LP}^* \leq 4 \cdot \ln(n) \cdot OPT$,
and the result is a set cover
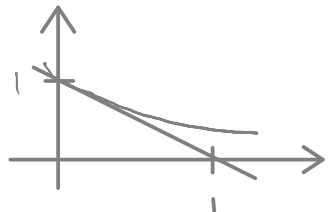The expected running time is polynomial.
(Calculations below)

p_i: prob. that e_i is covered

$\overline{p_i} = 1 - p_i$: prob. that e_i is not covered

Alg RR_1:

$$\overline{p_i} = \overline{\prod_{j: e_i \in S_j} (1-x_j)}^{\le e^{-x_j}, \text{ for any } x_j \in \mathbb{R}}$$

$$\le \overline{\prod_{j: e_i \in S_j} e^{-x_j}} = e^{-\underbrace{\sum_{j: e_i \in S_j} x_j}_{\le -1,}} \le e^{-1} \approx 0.37$$

by the LP constraint corresponding to e_i

Alg RR_2:

$$\overline{p_i} \le (e^{-1})^{2\ln n} = e^{-2\ln n} = \left(e^{\ln n}\right)^{-2} = n^{-2}$$

$\Downarrow$

$$Pr[\text{not set cover}] \le \sum_{i=1}^{n} \overline{p_i} \le \sum_{i=1}^{n} n^{-2} = n \cdot n^{-2} = n^{-1}$$

$$\underbrace{Pr\left[w(I) \ge 4 \cdot \ln(n) \cdot z_{LP}^*\right]}_{> \frac{1}{2} \text{ would give } E[w(I)] > 2 \cdot \ln(n) \cdot z_{LP}^* \nleftrightarrow} \le \frac{1}{2}, \quad \text{by Markov's Inequality:}$$

Alg RR_3:

$$Pr\left[\text{„not set cover" or „too expensive"}\right] \le n^{-1} + \frac{1}{2}$$

Thus,

$$E[\#\text{iterations}] \le \frac{1}{1-(n^{-1}+\frac{1}{2})} \approx 2$$

Sometimes randomized algorithms are simpler /
easier to describe/come up with.
Sometimes randomized algorithms can be derandomized
as we saw in Chapter 5.
Exercise sheet 7: derandomize $Alg\,RR_3$ (Ex. 5.7)

Exercise 5.7:

Derandomize the rounding alg. from Section 1.7, using the method of conditional expectations.

Hint: Use the following obj. fct. with random variables X_j, $1 \leq j \leq m$, and Z.

$$C = \sum_{j=1}^{m} X_j \omega_j + \lambda Z$$

where $X_j = \begin{cases} 1, & \text{if } S_j \text{ incl.} \\ 0, & \text{otherwise} \end{cases}$

$\lambda = n \cdot \ln n \cdot Z_{LP}^*$

$Z = \begin{cases} 0, & \text{if set cover} \\ 1, & \text{otherwise} \end{cases}$

With this obj. fct.,

any infeasible sol. has $C \geq \lambda = n \cdot \ln n \cdot Z_{LP}^*$ (*)

For AlgRR$_2$,

$$E[C] = E\left[\sum_{j=1}^{m} X_j \omega_j\right] + \lambda E[Z], \text{ by lin. of exp.}$$

$$\leq 2 \cdot \ln n \cdot Z_{LP}^* + n \cdot \ln n \cdot Z_{LP}^* \cdot n^{-1}, \text{ by the analysis in Sec. 1.7}$$

$$= 3 \cdot \ln n \cdot Z_{LP}^*$$

Thus, using the method of cond. exp., we can find a sol with $C \leq E[C] \leq 3 \cdot \ln n \cdot Z_{LP}^*$, and by (*), such a sol. is a set cover (assuming $n > 3$).

In order to do this, we must be able to calculate conditional exp values, i.e., calculate $E[C]$, given that decisions about $S_1, ..., S_\ell$ have already been made:

Let $\vec{X_\ell} = (X_1, X_2, ..., X_\ell)$. Then,

$$E[C \mid \vec{X_\ell}] = \sum_{j=1}^{\ell} X_j w_j + \sum_{j=\ell+1}^{m} x_j w_j + \lambda E[z \mid \vec{X_\ell}]$$

where $E[z \mid \vec{X_\ell}]$ can be calculated in the following way. For each element e_i,

$$Pr[e_i \text{ covered} \mid \vec{X_\ell}]$$

$$= \begin{cases} 1, & \text{if } e_i \text{ is contained in a set } S_j \\ & \text{st. } j \leq \ell \text{ and } X_j = 1 \quad (\text{i.e., } e_i \text{ is} \\ & \text{covered by one of the sets } S_1, ..., S_\ell) \\ \\ 1 - \underbrace{\prod_{\substack{j: e_i \in S_j \\ \wedge j > \ell}} (1-x_j)}, & \text{otherwise} \end{cases}$$

prob. that e_i will <u>not</u> be covered by any of the sets $S_{\ell+1}, ..., S_m$

$$\begin{aligned} E[z \mid \vec{X_\ell}] &= Pr(\text{set cover}) \cdot 0 + Pr(\text{not set cover}) \cdot 1 \\ &= Pr(\text{not set cover}) \\ &= 1 - \underbrace{\prod_{i=1}^{n} Pr[e_i \text{ covered} \mid \vec{X_\ell}]}_{Pr(\text{set cover})} \end{aligned}$$

DeRR$_2$

Solve LP optimally
For $l \leftarrow 1$ to m
    If $E[C \mid (X_1, X_2, \ldots, X_{l-1}, 0)] \leq E[C \mid (X_1, X_2, \ldots, X_{l-1}, 1)]$
        $X_l \leftarrow 0$
    Else
        $X_l \leftarrow 1$

1. Primal-dual for unweighted VC

Primal:

$$\min \sum_{v \in V} x_v$$

$$\text{s.t.} \quad x_u + x_v \geq 1, \quad (u,v) \in E$$

$$0 \leq x_v \leq 1, \quad v \in V$$

Dual:

$$\max \sum_{e \in E} y_e$$

$$\text{s.t.} \quad \sum_{e \in Adj(v)} y_e \leq 1, \quad v \in V$$

$$0 \leq y_e \leq 1, \quad e \in E$$

a) What does the alg. do?

For each $e \in E$: $y_e \leftarrow 0$
While some edge (u,v) is not covered
$\quad y_{(u,v)} \leftarrow 1$ // The two dual constr. corr. to u and v
$\qquad\qquad\qquad\qquad\qquad\qquad$ become tight
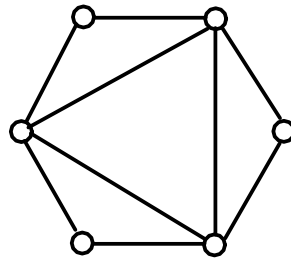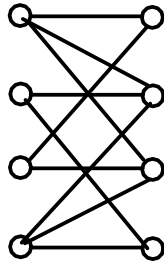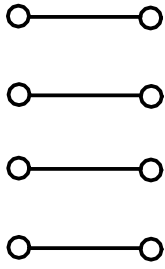$\quad$ Select u and v

b) Alg. without mention of LP

While some edge (u,v) is not covered
$\quad$ Select both endpoints u and v

c) Lower bound on approx. factor

Section 3.1: The Knapsack Problem

Knapsack

Input:

Knapsack with a capacity $B \in \mathbb{Z}^+$

Items $I = \{1, 2, \ldots, n\}$

Item i has size $s_i \in \mathbb{Z}^+$ and value $v_i \in \mathbb{Z}^+$

Objective:

Find a set of items with total size $\leq B$ and maximum total value

Greedy alg.

Consider items in order of decreasing v/s ratio

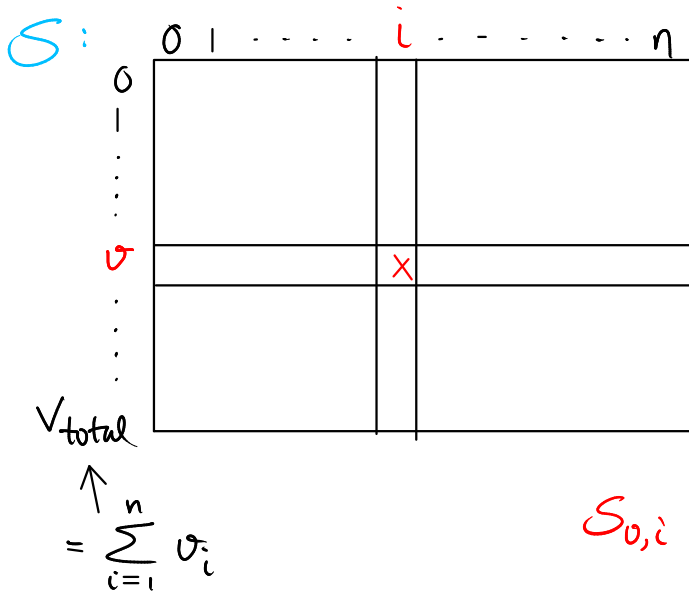Does not have any constant approximation factor:

Ex:

i	1	2
v_i	1	$B-1$
s_i	1	B

$\dfrac{v_1}{s_1} = 1 > \dfrac{v_2}{s_2} = 1 - \dfrac{1}{B}$

$\Rightarrow$ Greedy $= 1 = \dfrac{1}{B-1} \cdot \text{OPT}$

Dynamic prg alg:

S:

$S_{v,i}$: smallest possible total size of a subset of $\{1,...,i\}$ with total value v, i.e.,

$$S_{v,i} = \min_{I \subseteq \{1,...,i\}} \left\{ \sum_{j \in I} s_j \;\middle|\; \sum_{j \in I} v_j = v \right\}$$

$V_{total} = \sum_{i=1}^{n} v_i$

Ex:

B=5

i	1	2	3
v_i	③	1	②
s_i	2	3	2

v_i \\ i	0	1	2	3
0	0	0	0	0
1	∞	∞	3→3	
2	∞	∞	∞	2
3	∞	2→2→2		
4	∞	∞	5→5	
5	⑤ ∞	∞	∞	④ ≤B
6	∞	∞	∞	7 >B

Optimal total value →

S:

$S_{v,i}$: smallest possible total size of a subset of $\{1, ..., i\}$ with total value v, i.e.,

$$S_{v,i} = \min_{I \subseteq \{1, ..., i\}} \left\{ \sum_{j \in I} s_j \;\middle|\; \sum_{j \in I} v_j = v \right\}$$

If $i = 0$ and $1 \leq v \leq V_{total}$
$$S_{v,i} = \infty$$

Otherwise,

$$S_{v,i} = \begin{cases} S_{v, i-1}, & \text{if} \quad 0 \leq v < v_i \\ \min\{ \underbrace{S_{v, i-1}}_{\substack{\text{best solution} \\ \underline{\text{without}} \text{ item } i}}, \underbrace{S_{v - v_i, i-1} + s_i}_{\substack{\text{best solution} \\ \underline{\text{with}} \text{ item } i}} \}, & \text{if} \quad v \geq v_i \end{cases}$$

Not necessary to fill in the ∞-entries
($A[i]$ corresponds to column i):

Alg 3.1:

$A[1] \leftarrow \{ (0,0), (s_1, v_1) \}$

For $i \leftarrow 2$ to n

    $A[i] \leftarrow A[i-1]$

    For each $(S,V) \in A[i-1]$

        If $S + s_i \leq B$

            $A[i] \leftarrow A[i] \cup \{ (S+s_i, V+v_i) \}$

    Remove dominated pairs from $A[i]$

Return $\max\limits_{(S,V) \in A[n]} \{ V \}$

$B = 5$

i	1	2	3
v_i	③	1	②
s_i	2	3	2

$A[1] = \{ (0,0), (3,2) \}$

$A[2] = A[1] \cup \{ (1,3), (4,5) \}$
$= \{ (0,0), (1,3), (3,2), (4,5) \}$

$A[3] = A[2] \cup \{ (2,2), (3,5), (5,4) \}$
$= \{ (0,0), (1,3), (2,2), (3,2), \cancel{(3,5)}, (4,5), (5,4) \}$

dominated by

Analysis

Running time: $O(n \cdot V_{total})$

Input size: $O(\log B + n(\log M + \log S))$, where

$$M = \max_{1 \leq i \leq n} \{v_i\} \quad \text{and} \quad S = \max_{1 \leq i \leq n} \{s_i\}.$$

Poly. time?

Ex: Consider a family of instances where
$V_{total} = 2^n$ and
$B, S \leq 2^n$
Then
Running time $T(n) \in \Omega(n \cdot 2^n)$ and
Input size $S(n) \in O(n^2)$
$\Rightarrow T(n) \in \Omega\left((S(n))^{c\sqrt{n}}\right)$

No

But if the numeric part of the input (i.e., B, v_i, s_i) were written in unary, the input size would be $\Theta(B + V_{total} + S_{total})$, and the running time would be poly. in the input size.
Hence, the running time is pseudopolynomial.

Note: if V_{total} is poly. in n for all possible input instances, the dyn. prg. alg. is poly.
Leading to the following idea...

> Idea for approximation algorithm :

Round values s.t. there are only a poly. number of (equidistant) values :

- Choose a value μ
- Round down each item value to the nearest multiple of μ
- Do dyn. prg. on the rounded values

How to choose μ ?

- Approximation :
  When rounding, each item looses a value of less than μ. Hence, the value of any solution is changed by less than $n\mu$.
  Thus, if we want a precision of ε,
  $$\mu = \frac{\varepsilon M}{n}$$
  will do, since then $n\mu = \varepsilon M \leq \varepsilon \cdot OPT$.
  (We will add more detail to this argument in the proof of Thm 3.5.)

- Running time :
  $$n \cdot \frac{V_{total}}{\mu} \leq n \cdot \frac{nM}{\mu} = n \cdot nM \cdot \frac{n}{\varepsilon M} = \frac{1}{\varepsilon} \cdot n^3$$

Since each rounded value is a multiple of μ, we might as well scale by a factor of $\frac{1}{\mu}$ s.t. the possible values will be $1, 2, \ldots, \lfloor \frac{V_{total}}{\mu} \rfloor$ instead of $\mu, 2\mu, \ldots, \lfloor \frac{V_{total}}{\mu} \rfloor \mu$:

Alg 3.2

$M \leftarrow \max\limits_{1 \leq i \leq n} v_i$

$\mu = \frac{\varepsilon M}{n}$

for $i \leftarrow 1$ to n

$\qquad v_i' \leftarrow \lfloor \frac{v_i}{\mu} \rfloor$

Do dyn. prg. with values v_i' (and sizes s_i)
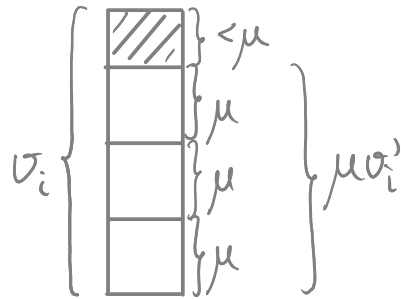
Theorem 3.5

Alg. 3.2 is a $(1-\varepsilon)$-approx. alg. with a running time poly. in both input size and $\frac{1}{\varepsilon}$

Proof:

Approximation ratio:

For each item i, $\mu v_i'$ equals v_i rounded down to the nearest multiple of μ. $\quad\}$ (*)

Thus, $v_i - \mu v_i' < \mu$ (each item „loses" less than μ in the rounding.) $\quad\}$ (**)

Let **A** be the set of items selected by Alg. 3.2

   This is an *optimal solution* to the instance with values v_i', and hence, *to the instance with values* μv_i. $\quad\}$ (***)

Let **O** be the set of items in an *optimal solution to the original instance with values* v_i.

The total value produced by Alg. 3.2 is

$$\sum_{i \in A} v_i \geqslant \sum_{i \in A} \mu v_i', \qquad \text{by } (*)$$

$$\geqslant \sum_{i \in O} \mu v_i', \qquad \text{by } (***)$$

$$> \sum_{i \in O} (v_i - \mu), \qquad \text{by } (**)$$

$$\geqslant \left(\sum_{i \in O} v_i \right) - n\mu, \qquad \text{since } |O| \leq n$$

$$= OPT - n \cdot \frac{\varepsilon M}{n}$$

$$= OPT - \varepsilon M$$

$$\geqslant (1 - \varepsilon) OPT, \qquad \text{since } OPT \geqslant M$$

Running time:

$$O\left(\frac{1}{\varepsilon} \cdot n^3\right) \text{ as proven above.}$$

$\square$

According to Thm 3.5, Alg. 3.2 is a
fully **polynomial time** **approximation scheme** **(FPTAS)**

also   poly. in input   Family $\{A_\varepsilon\}$ of alg., where A_ε
poly.   size   has precision ε.
in $\frac{1}{\varepsilon}$ $\big((1-\varepsilon)-$approx. alg for max. problems,
                                          $(1+\varepsilon)-$approx. alg for min. problems$\big)$

Def. 3.4   Def. 3.3

Thus, Thm 3.5 could also be stated like this:

Theorem 3.5: Alg 3.2 is a FPTAS

Multiple Knapsack problem: Fixed #knapsacks
Bin Packing can be seen as a dual version of
Multiple Knapsack.

Bin Packing

Input: n items with sizes between 0 and 1.
Objective: Pack items in bins of size 1,
using as few bins as possible.

Simple approximation algorithms:

Alg.	Running time	Asymp. approx. factor
Next-Fit	$O(n)$	2
First-Fit	$O(n\log n)$	1.7
Best-Fit	—"—	—"—
Next-Fit-Decreasing	—"—	≈ 1.69
First-Fit-Decreasing	—"—	$1.\overline{2}$
Best-Fit-Decreasing	—"—	—"—

Approximation scheme?
Can we do the same kind of rounding for Bin Packing
as we did for Knapsack?

No:

Assume that each item size is rounded up to the nearest multiple of μ, for some $0 < \mu < 1$.
(We need to round <u>up</u> to ensure the packing will be valid.)

<u>Ex:</u>
Let $a = \max\{k\mu \mid k \in \mathbb{Z} \wedge k\mu \leq \frac{1}{2}\}$.
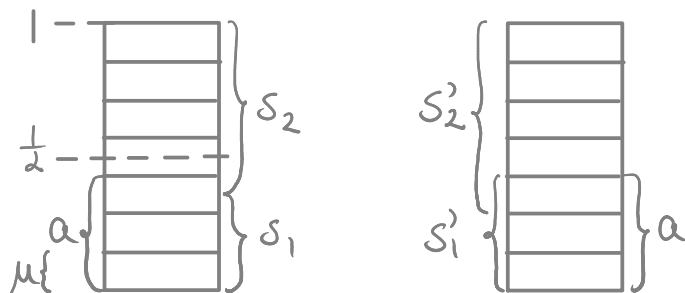Then, $\frac{1}{2} - \mu < a \leq \frac{1}{2}$.
Assume $\mu < \frac{1}{6}$. (An approx. scheme should work for any $\mu > 0$.)
Then, $\frac{1}{3} < a \leq \frac{1}{2}$.

Consider an input consisting of
- m items of size $s_1 = a - \mu/2 \rightarrow$ rounded up to $s_1' = a > \frac{1}{3}$
- m items of size $s_2 = 1 - a + \mu/2 > \frac{1}{2}$, since $a \leq \frac{1}{2}$

For this instance, the items fit pairwise in m bins, but for the rounded instance, $m + \frac{m}{2}$ bins are needed.

This would yield an approx. factor of at least $\frac{3}{2}$.

Ex. 3.1:

New variant of Greedy:

Sort items s.t $v_1/s_1 \geq v_2/s_2 \geq \ldots \geq v_n/s_n$

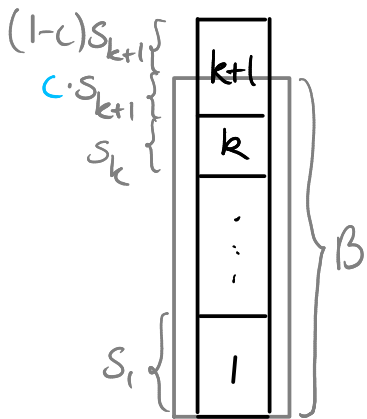Choose k s.t. $\sum_{i=1}^{k} s_i \leq B$, but $\sum_{i=1}^{k+1} s_i > B$

Choose i^* s.t. $v_{i^*} = \max_{1 \leq i \leq n} v_i$

If $\sum_{i=1}^{k} v_i > v_{i^*}$

    Return $\{1,\ldots,k\}$

Else

    Return $\{i^*\}$

Since no solution has more value per size unit than the „solution" containing items $1,\ldots,k$ and a c-fraction of item $k+1$:

$$OPT \leq \sum_{i=1}^{k} v_i + c \cdot v_{k+1}$$

$$< \sum_{i=1}^{k} v_i + v_{k+1}, \quad \text{since } c < 1$$

$$\leq \sum_{i=1}^{k} v_i + v_{i^*}, \quad \text{since } v_{i^*} \geq v_{k+1}$$

$$\max\left\{\sum_{i=1}^{k} v_i, \ v_{i^*}\right\} \geq \frac{1}{2}\left(\sum_{i=k}^{k} v_i + v_{i^*}\right)$$

$$\geq \frac{1}{2} \cdot OPT$$

Section 3.3: Bin Packing

Bin Packing

Input: n items with sizes between 0 and 1.

Objective: Pack items in bins of size 1, using as few bins as possible.

Last time we discussed simple approximation algorithms. Today we will develop an approximation scheme:

$A_\varepsilon(I)$

Split input I into
- I_s: items smaller than $\varepsilon/2$ (small items)
- I_ℓ: remaining items   (large items)

1. Pack large items:
   - a. Round up item sizes $(I_\ell \rightarrow I_\ell')$
     $\Rightarrow O(\frac{1}{\varepsilon^2})$ different sizes
   - b. Do dyn. prg. on I_ℓ'
     $\Rightarrow A_\varepsilon(I_\ell') = OPT(I_\ell')$

2. Add small items to the packing using First-Fit (or any other Any-Fit alg.)

The rounding scheme (1.a.) will be described later.

Lemma 3.10

$$A_\varepsilon(I) \leq \max\left\{ A_\varepsilon(I_\ell), \; \underbrace{\frac{2}{2-\varepsilon} \cdot \text{size}(I)}_{} + \underbrace{1}_{} \right\}$$

$$\underbrace{\hspace{2cm}}_{\leq 1+\varepsilon,} \quad \underbrace{\hspace{2cm}}_{\leq OPT(I)}$$
for $\varepsilon \leq 1$

Proof:

If no extra bin is needed for adding the small items, $A_\varepsilon(I) = A_\varepsilon(I_\ell)$.

Otherwise, all bins, except possibly the last one, are filled to more than $1 - \varepsilon/2$.
In this case,

$$A_\varepsilon(I) \leq \left\lceil \frac{\text{size}(I)}{1 - \varepsilon/2} \right\rceil < \frac{\text{size}(I)}{1 - \varepsilon/2} + 1$$

$$= \frac{2}{2-\varepsilon} \, \text{size}(I) + 1 \qquad \qquad \square$$

Thus, we just need to ensure that
$A_\varepsilon(I_\ell) \leq (1+\varepsilon)\, OPT.$

Rounding scheme (1.a.)

Last time we saw that a rounding scheme similar to the one we used for Knapsack would at best yield an approx. factor of 1.5. Instead, we will use:

Linear grouping:

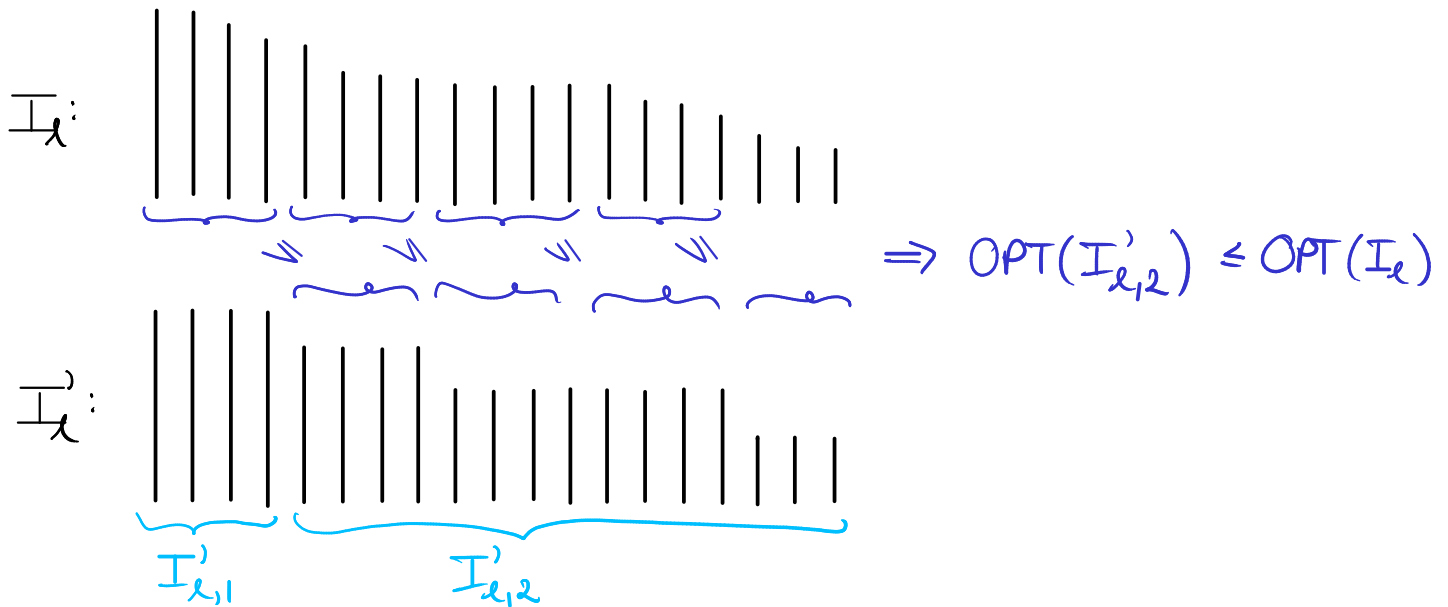- Sort items in I_ℓ by decreasing sizes.
- Partition sorted I_ℓ in groups of k consecutive items. (k will be determined later.)
- For each group, round up item sizes to largest size in the group.

The result is called I_ℓ'.

Ex: ($k=4$)
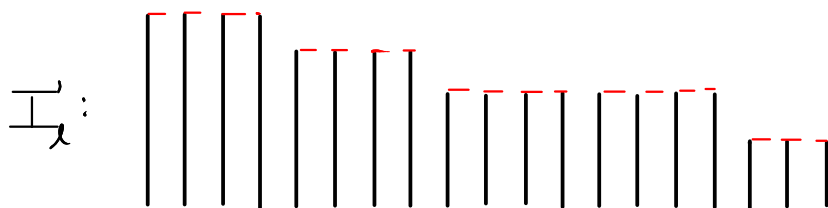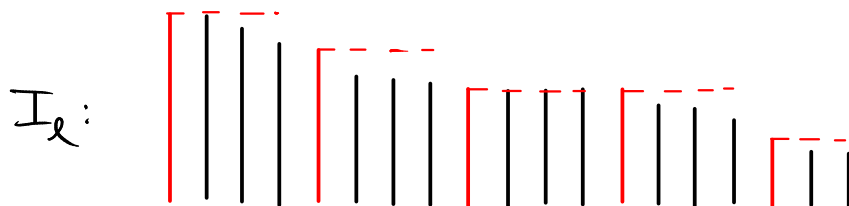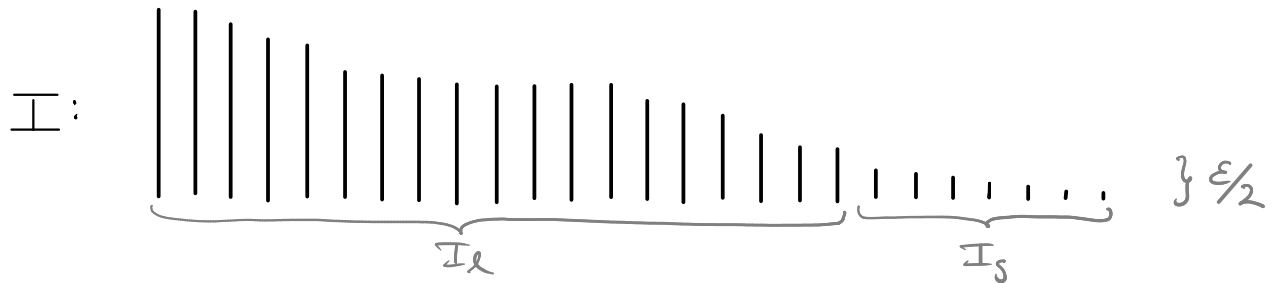
Each item in the i'th group of I_ℓ is at least as large as any item in the $(i+1)$st group of I'_ℓ:

$$\Rightarrow OPT(I'_{\ell,2}) \leq OPT(I_\ell)$$

$$OPT(I'_\ell) \leq \underbrace{OPT(I'_{\ell,1})}_{\leq k,} + \underbrace{OPT(I'_{\ell,2})}_{\leq OPT(I_\ell)}$$

since $|I'_{\ell,1}| = k$

This proves:

Lemma 3.11: $\underbrace{OPT(I'_\ell)}_{= A_\varepsilon(I'_\ell)} \leq OPT(I_\ell) + k$

Thus, letting $k = \lfloor \varepsilon \cdot \text{size}(I) \rfloor \overset{(*)}{\leq} \varepsilon \cdot OPT(I)$

will ensure that

$$A_\varepsilon(I_\lambda) = A_\varepsilon(I_\ell')$$
$$= OPT(I_\ell')$$
$$\leq OPT(I_\ell) + k, \quad \text{by Lemma 3.11}$$
$$\leq OPT(I_\ell) + \varepsilon \cdot OPT(I), \quad \text{by } (*)$$
$$\leq (1+\varepsilon) \cdot OPT(I), \quad \text{since } I_\ell \subseteq I$$

Now, by Lemma 3.10,

$$A_\varepsilon(I) \leq \max\left\{ \underbrace{A_\varepsilon(I_\ell)}, \underbrace{\frac{2}{2-\varepsilon} \cdot \text{size}(I) + 1} \right\}$$

$\leq (1+\varepsilon)OPT(I),$ $\leq (1+\varepsilon)OPT + 1$;

as just
shown

$$\frac{2}{2-\varepsilon} \leq 1+\varepsilon \iff$$
$$2 \leq (1+\varepsilon)(2-\varepsilon) \iff$$
$$2 \leq 2 + \varepsilon - \varepsilon^2 \iff$$
$$\varepsilon \leq 1$$

Thus, $A_\varepsilon(I) \leq (1+\varepsilon) \cdot OPT(I) \boxed{+1}$

asymptotic approximation scheme

Packing I'_ℓ using dynamic programming (l.b.)

At most $2/\varepsilon$ items fit into one bin,
since all items in I_ℓ have size at least $\varepsilon/2$.

There are $N \leq \lceil n/k \rceil$ different sizes $s_1, \ldots, s_N$ in I'_ℓ.

Hence, any packing of a bin can be represented by
a vector $(m_1, \ldots, m_N)$, where m_i, $1 \leq i \leq N$, is
the #items of size s_i in the bin and $0 \leq m_i \leq 2/\varepsilon$.
A vector representing the contents of a bin is
called a configuration.
Let $\mathcal{C}$ denote the set of possible bin configurations.
Note that $|\mathcal{C}| < \left(\frac{2}{\varepsilon}\right)^N$

Let n_i be the #items of size s_i in I'_ℓ

For the dyn. prg. we use an N-dimensional table B
with $n_i + 1$ rows in the i'th dimension.
$B[m_1, \ldots, m_N]$ will be the minimum #bins required
to pack m_i items of size s_i, $1 \leq i \leq N$.

Ex:

$I = \langle 0.6, 0.5, 0.5, 0.4, 0.4, 0.4, 0.3, \underbrace{0.1, 0.1}_{< \varepsilon/2} \rangle$

$\varepsilon = 0.4, \quad k = 4$

$I_\ell = \langle 0.6, 0.5, 0.5, 0.4 \,|\, 0.4, 0.4, 0.3 \rangle$

$I'_\ell = \langle 0.6, 0.6, 0.6, 0.6, 0.4, 0.4, 0.4 \rangle$

$$S_1 = 0.6 \qquad\qquad S_2 = 0.4$$
$$n_1 = 4 \qquad\qquad n_2 = 3$$

$\mathcal{C} = \{(0,1), (0,2), (1,0), (1,1)\}$

$B[4,3] = 1 + \min_{(m_1, m_2) \in \mathcal{C}} \{ B[4-m_1, 3-m_2] \}$

$\qquad = 1 + \min \{ B[4,2], B[4,1], B[3,3], B[3,2] \}$

$\qquad = 1 + B[3,2] = 4$

In general:

$\qquad B[m_1, ..., m_N] = 1 + \min_{(c_1, ..., c_N) \in \mathcal{C}} \{ m_1 - c_1, ..., m_N - c_N \}$

Left table:

0.4 / 0.6	0	1	2	3
0	0	1	1	2
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3
4	4	4	4	4

Right table:

0.4 / 0.6	0	1	2	3
0	0	1	1	2
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3
4	4	4	4	4

I_e':

Left: [0.4 / 0.6] [0.4 / 0.6] [0.4 / 0.6] [/ 0.6]

Right: [0.4 / 0.6] [0.4 / 0.6] [/ 0.6] [0.4 / 0.6]

I_e:

[0.4 / 0.6] [0.4 / 0.5] [0.3 / 0.5] [/ 0.4]

I:

[0.4 / 0.6] [0.1 / 0.4 / 0.5] [0.1 / 0.3 / 0.5] [/ 0.4]

$\boxed{\text{Running time}}$

Let $n_\ell = |I_\ell|$. Then,

$$\text{size}(I) \geqslant \text{size}(I_\ell) \geqslant n_\ell \cdot \frac{\varepsilon}{2}, \qquad (*)$$

since I_ℓ contains only large items.

Thus,

$$k = \lfloor \varepsilon \cdot \text{size}(I) \rfloor \geqslant \lfloor n_\ell \cdot \frac{\varepsilon^2}{2} \rfloor \geqslant n_\ell \cdot \frac{\varepsilon^2}{4} \qquad (**)$$

$\qquad \qquad \qquad \qquad \qquad \qquad \nwarrow$ by $(*)$

Hence,

$$N \leqslant \left\lceil \frac{n_\ell}{k} \right\rceil \leqslant \left\lceil \frac{4}{\varepsilon^2} \right\rceil \qquad (***)$$

$\qquad \quad \nwarrow$ by $(**)$

Table size $\leqslant n_\ell^N \leqslant n^N$

Time per entry $O(|\mathcal{B}|) \leqslant O\left(\left(\frac{2}{\varepsilon}\right)^N\right)$

Running time $O\left(n^N \cdot \left(\frac{2}{\varepsilon}\right)^N\right) = O\left(\left(\frac{2n}{\varepsilon}\right)^N\right) \subseteq O\left(\left(\frac{2n}{\varepsilon}\right)^{\lceil \frac{4}{\varepsilon^2} \rceil}\right)$

$\qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \nearrow$ by $(***)$

$\qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \downarrow$

not <u>fully</u> poly. time

Hence, $\{A_\varepsilon\}$ is an

Asymptotic Poly. Time Approx. Scheme (APTAS)

This proves:

$\boxed{\text{Thm 3.12: There is an APTAS for Bin Packing}}$

There is no PTAS for Bin Packing:

Theorem 3.8

No alg. for Bin Packing has an absolute approx. ratio better than $3/2$, unless $P = NP$.

Proof:

Reduction from Partition Problem:

Given a set S of integers, can S be partitioned into two sets S_1 and S_2, such that $\sum_{s \in S_1} s = \sum_{s \in S_2} s$?

For a given instance S of the partition problem, let $B = \sum_{s \in S} s$ and $I = \{ s \cdot \frac{2}{B} \mid s \in S \}$.

Then, $\sum_{i \in I} i = B \cdot \frac{2}{B} = 2$

If we use I as input for the bin packing problem,
- at least 2 bins are needed, and
- 2 bins suffice, iff S is a yes-instance for the partition problem.

If we had a bin packing alg. with an approx. factor $< 3/2$, it would always use only 2 bins, whenever 2 bins suffice. Thus, the alg. could be used to decide any instance of the partition problem.

□

Section 2.3: Scheduling to minimize makespan

Makespan Scheduling on Identical Machines

Input:
- m machines
- n jobs w. processing times $p_1, \ldots, p_n \in \mathbb{Z}^+$

Output:
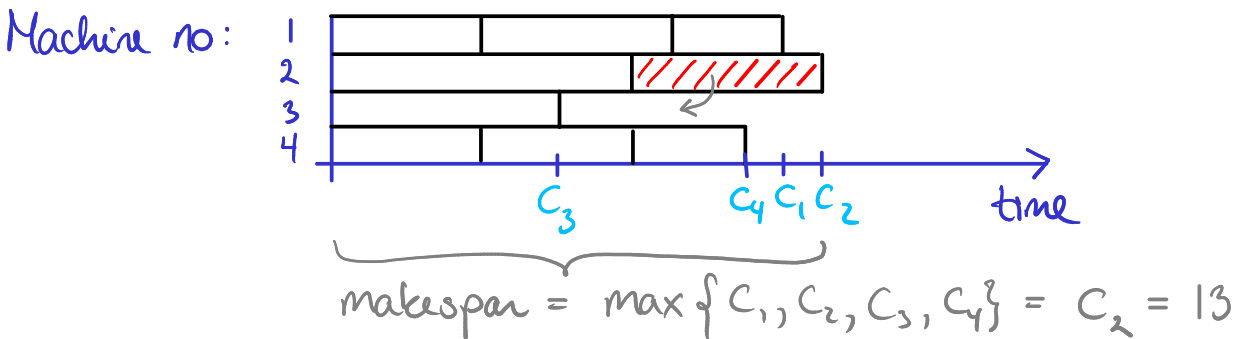- Assignment of jobs to machines s.t. the makespan is minimized
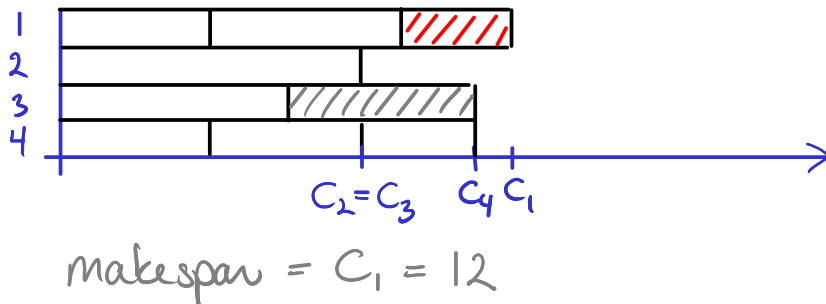
↑ time when last machine finishes processing

Ex:

Input: $4, 5, 3, 8, 5, 6, 4, 4, 3$

Output:

Machine no:

makespan $= \max \{C_1, C_2, C_3, C_4\} = C_2 = 13$

The schedule can be improved:

makespan $= C_1 = 12$

Local Search Alg.

Repeat
   job $\ell \leftarrow$ job that finishes last
   If $\exists$ machine i where job ℓ would finish earlier
     Move job ℓ to machine i
Until job ℓ is not moved

Theorem 2.5

The local search alg. is a $(2 - \frac{1}{m})$-approx. alg.

Proof:

Let $P_{max} = \max\limits_{1 \leq j \leq n} P_j$ and $P = \sum\limits_{j=1}^{n} P_j$

Lower bounds on OPT:
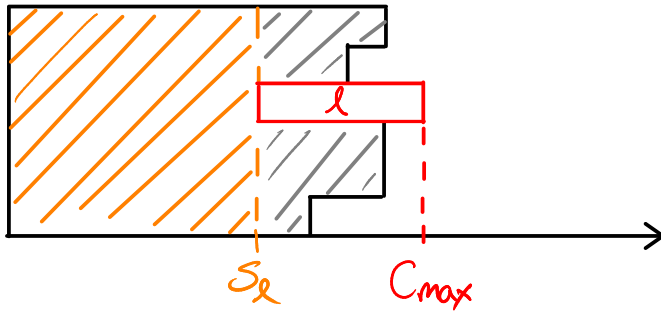
$$OPT \geq P_{max} \quad (*)$$
   since the machine i with the longest job P_j
   has $C_i \geq P_j$

$$OPT \geq \frac{P}{m} \quad (**)$$
   since this is the average completion time
   of the machines.

Upper bound on alg.'s makespan:

$\Leftrightarrow$ $P \geq m \cdot S_\ell + p_\ell$, since all machines are busy until S_ℓ

$S_\ell \leq \dfrac{P - p_\ell}{m}$ (***)

$$C_{max} = S_\ell + p_\ell$$

$$\leq \frac{P - p_\ell}{m} + p_\ell , \quad \text{by } (***)$$

$$= \frac{P}{m} + \left(1 - \frac{1}{m}\right) p_\ell$$

$$\leq OPT + \left(1 - \frac{1}{m}\right) OPT , \quad \text{by } (*) \text{ and } (**)$$

$$= \left(2 - \frac{1}{m}\right) OPT$$

$\square$

What would be a natural greedy algorithm?

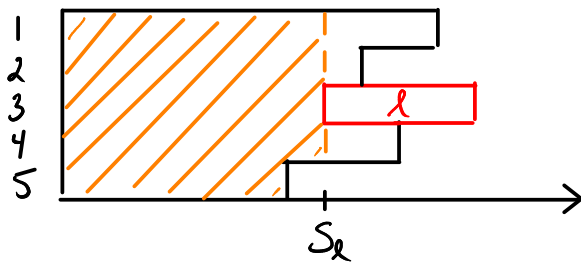**List Scheduling (LS)**

For $j \leftarrow 1$ to n

    Schedule job j on currently least loaded machine

Approx. ratio?

What properties of the local search alg. did we use to prove $2 - \frac{1}{m}$?

We used only the fact that all machines are busy at least until S_ℓ (this was enough to prove (★)). This is also true for LS:

LS would not have placed job ℓ on machine 3.

Theorem 2.6: LS is a $(2 - \frac{1}{m})$-approx. alg.

Note that $\frac{LS}{OPT} < 2 - \frac{1}{m}$, unless $p_\ell = p_{max}$ and all other machines finish by the time job ℓ starts. Thus, it seems advantageous to schedule short jobs last.

Longest Processing Time (LPT)

For each job j, in order of decreasing processing times
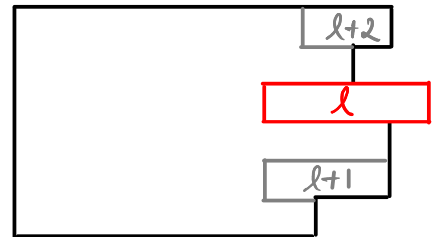Schedule job j on currently least loaded machine

Theorem 2.7: LPT is a $\left(\frac{4}{3} - \frac{1}{3m}\right)$-approx. alg.

Proof:

Number the jobs s.t. $p_1 \geq p_2 \geq \cdots \geq p_n$.
(Then, the indices indicate the order in which the jobs are scheduled.)

Let job ℓ be a job to finish last:

We can assume that $\ell = n$:

Let $I = \{p_1, \ldots, p_n\}$ and $I_\ell = \{p_1, \ldots, p_\ell\}$.

Then, $LPT(I) = LPT(I_\ell)$, since jobs $\ell+1, \ldots, n$ finish no later than job ℓ.

Moreover, $OPT(I) \geq OPT(I_\ell)$, since $I_\ell \subseteq I$.

Thus, proving $\frac{LPT(I_\ell)}{OPT(I_\ell)} \leq \frac{4}{3} - \frac{1}{3m}$ will imply

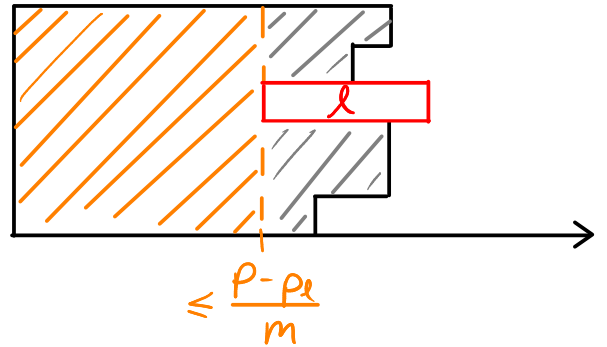$$\frac{LPT(I)}{OPT(I)} \leq \frac{LPT(I_\ell)}{OPT(I_\ell)} \leq \frac{4}{3} - \frac{1}{3m}.$$

(Or said in a different way, we can ignore the jobs $\ell+1, \ldots, n$.)

Thus, we can assume that no job is shorter than job ℓ. This will be used in Case 2 below.

Case 1: $p_\ell \leq \frac{1}{3} \cdot OPT$

Similarly to the proof of Thm 2.5:

$$LPT \leq \frac{P - p_\ell}{m} + p_\ell$$

$$= \frac{P}{m} + \left(1 - \frac{1}{m}\right) p_\ell$$

$$\leq OPT + \left(1 - \frac{1}{m}\right) p_\ell$$

$$\leq OPT + \left(1 - \frac{1}{m}\right) \cdot \frac{1}{3} OPT$$

$$= \left(\frac{4}{3} - \frac{1}{3m}\right) OPT$$

$$\leq \frac{P - p_\ell}{m}$$

Case 2: $p_\ell > \frac{1}{3} \cdot OPT$

In this case, all jobs are longer than $\frac{1}{3} OPT$. Hence, in OPT's schedule, each machine has at most 2 jobs, i.e., $n \leq 2m$.

Claim: In this case $LPT = OPT$.

Proof of claim: Exercise 2.2.

$\square$

From the proof of Thm. 2.7, we learned:

- If $p_\ell > \frac{1}{3} \text{OPT}$, $\text{LPT} = \text{OPT}$.
- Otherwise, $\text{LPT} < \frac{4}{3} \text{OPT}$.

Can we balance the two cases better?

What if we first **schedule all jobs of length at least $\frac{1}{4} \text{OPT}$ optimally,** and then use LPT for the remaining jobs? What approx. factor would be obtained?

$$\overbrace{\text{length} \geq \frac{1}{4} \text{OPT}}$$

Would the schedule of the long jobs have to be optimal to achieve this approx. factor?

From the proof of Thm. 2.7, we learned:
- If $p_\ell > \frac{1}{3} OPT$, $LPT = OPT$.
- Otherwise, $LPT \leq \frac{4}{3} OPT$.

Can we balance the two cases better?

What if we first schedule all jobs of length at least $\frac{1}{4} OPT$ optimally, and then use LPT for the remaining jobs?

If the last job to finish is a long job,
$$C_{max} = OPT,$$
   since the long jobs are scheduled optimally.

Otherwise,
$$C_{max} \leq OPT + \left(1-\tfrac{1}{m}\right) p_\ell, \quad \text{by the proof of Thm. 2.5}$$
$$\leq OPT + \left(1-\tfrac{1}{m}\right) \cdot \tfrac{1}{4} \cdot OPT$$
$$< \tfrac{5}{4} \cdot OPT$$

Would the schedule of the long jobs have to be optimal?

No, a $\frac{5}{4}$-approx. would suffice:

If the last job to finish is a long job,
$$C_{max} \leq \tfrac{5}{4} \cdot OPT$$

Otherwise,
$$C_{max} < OPT + p_\ell \leq \tfrac{5}{4} \cdot OPT.$$

This sketches the idea for a PTAS...

1. Schedule long jobs ($> \varepsilon \cdot OPT$) using rounding and dyn. prg.

$\Rightarrow C_{max} \leq (1+\varepsilon) OPT$

2. Add short jobs ($\leq \varepsilon \cdot OPT$) to the schedule using LPT.

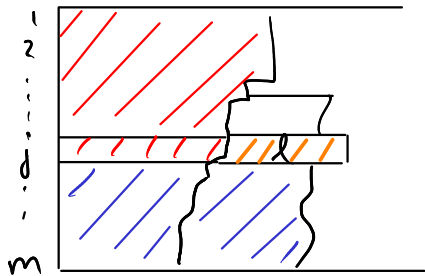$\Rightarrow C_{max} \leq (1+\varepsilon) OPT$

If $P_\ell > \frac{1}{3} OPT$, $LPT = OPT$

Proof:

$n \leq 2m$

Assume for the sake of contradiction $P_i + P_\ell > OPT$, and consider an optimal schedule.

In that schedule no two red jobs are combined, and job ℓ is not combined with a red job.

Furthermore, no blue job can be combined with a red job, since the blue jobs are at least as large as job ℓ.

Thus, the j red jobs must be scheduled on separate machines, and they cannot be combined with job ℓ or any of the $2(m-j)$ blue jobs.

This gives a total of $2(m-j)+1$ jobs that must be scheduled on $m-j$ machines.

Thus, there must be a machine with at least three of these jobs that each have a size of $\geq P_\ell > \frac{1}{3} \cdot OPT$.
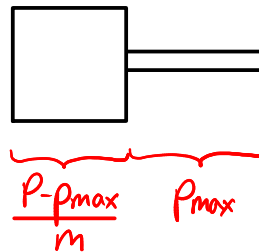
Hence, the machine has a total load $> OPT$ ⚡

Exercise: Give an instance I, where $LS(I) = (2 - \frac{1}{m}) \, OPT$.

To prove $LS(I) \leq (2 - \frac{1}{m}) \cdot OPT(I)$, we used
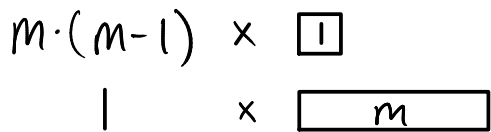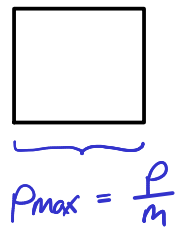
- $OPT \geq \frac{P}{m}$

- $OPT \geq p_{max}$

- $LS \leq \frac{P - p_{max}}{m} + p_{max}$

For $LS = OPT$, all inequalities need to be tight:

LS:

OPT:

$\underbrace{\qquad}_{\frac{P - p_{max}}{m}} \underbrace{\qquad}_{p_{max}}$

$\underbrace{\qquad}_{p_{max} = \frac{P}{m}}$

$m \cdot (m-1) \times \boxed{1}$

$1 \qquad \times \boxed{\qquad m \qquad}$

LS:

$\overbrace{\qquad\qquad}^{2m-1}$

$m \left\{ \vphantom{\rule{0pt}{2em}} \right.$

$\underbrace{\qquad}_{m-1}$

OPT:

$m-1 \left\{ \vphantom{\rule{0pt}{1.5em}} \right.$

$\underbrace{\qquad}_{m}$

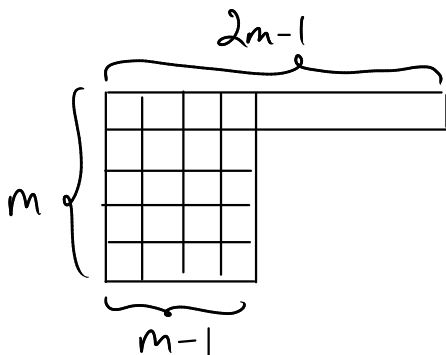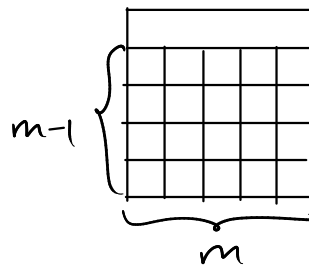## Section 3.2: Makespan Scheduling — A PTAS

Sketch of PTAS:

1. Schedule long jobs ($> \varepsilon \cdot OPT$) using rounding and dyn. prg.
   $\Rightarrow C_{max} \leq (1+\varepsilon) OPT$

2. Add short jobs ($\leq \varepsilon \cdot OPT$) to the schedule using LPT.
   $\Rightarrow C_{max} \leq (1+\varepsilon) OPT$

How to identify long/short jobs when we don't know OPT?

- We need the short jobs to be $\leq \varepsilon \cdot OPT$ to ensure the approx. factor. For this purpose, we could use any lower bound on OPT, like P/m.

- But we also need the long jobs to be $\geq \varepsilon \cdot OPT$ to ensure the running time.

We will develop a *family of algorithms* with an algorithm B_k for each $k \in \mathbb{Z}^+$. $(\varepsilon = \frac{1}{k})$

Scheduling the long jobs:

(1) „Guess" an optimal makespan T.
The long jobs are those longer than T/k.

(2) Round down each job size to the nearest multiple of T/k^2.

(3) Use dyn. prg. to check whether optimal makespan $\leq T$ for rounded long jobs.

Do binary search for T on the interval $[L, U]$, where

$$L = \max \left\{ \left\lceil \frac{P}{m} \right\rceil, \, p_{max} \right\} \text{ and}$$

$$U = \left\lfloor \frac{P}{m} + \left(1 - \frac{1}{m} \right) p_{max} \right\rfloor,$$

where P is the total size of long jobs.

$B_k(I)$

$L \leftarrow \max\left\{ \lceil \frac{P}{m} \rceil, P_{max} \right\}; \quad U \leftarrow \lceil \frac{P}{m} + (1 - \frac{1}{m}) P_{max} \rceil$

While $L \neq U$

$\quad T \leftarrow \frac{1}{2} \lceil L + U \rceil$

$\quad I_\ell \leftarrow \{ \text{job } j \in I \mid P_j > T/k \}$ // Update set of long jobs

$\quad I'_\ell \leftarrow I_\ell$ with each job size rounded down to nearest multiple of T/k^2

$\quad$ Use dyn. prg. to pack I'_ℓ in bins of size T

$\quad$ If #bins $\leq m$

$\quad\quad U \leftarrow T$

$\quad$ else

$\quad\quad L \leftarrow T+1$

$S'_\ell \leftarrow$ schedule of I'_ℓ corresponding to the packing found by dyn. prg.

$S_\ell \leftarrow$ schedule of I_ℓ corresponding to S'_ℓ

$S \leftarrow$ schedule of I obtained by adding short jobs to S_ℓ using LPT

(left margin) Binary search for T

Dyn. prg. as for bin packing:

S_ℓ' places $\le k$ jobs on each machine:

    Each long job has size $\ge T/k$

    Since T/k is a multiple of T/k^2, each job in I_ℓ' also has size $\ge T/k$.

There are $\le k^2$ different job sizes in I_ℓ', since no job is longer than T.

Hence, the configuration of a machine can be represented by a vector $(s_1, s_2, \ldots, s_{k^2})$, where $0 \le s_i \le k$.

Thus, $|\mathcal{C}| \le k^{k^2}$.

Table $(\mathcal{B})$:

    $\le k^2$ dimensions   (one for each size in I_ℓ')

    $n_i + 1$ rows in dim. i ($n_i = \#$items of size $i \cdot T/k^2$ in I_ℓ')

$$B(n_1, \ldots, n_{k^2}) = 1 + \min_{\vec{s} \in \mathcal{C}} \{ B(n_1 - s_1, \ldots, n_{k^2} - s_{k^2}) \}$$

    is the min. #bins of size T it takes to pack n_i items of size $i \cdot T/k^2$, $0 \le i \le k^2$.

Running time:

#table entries: $O(n^{k^2})$

Time per entry: $|\mathcal{C}| \le k^{k^2}$

#iterations of while loop: $\log(U - L) \le \log(P_{max})$

Total time: $O\left((nk)^{k^2} \cdot \log(P_{max}) \right)$

Approximation ratio:

When B_k terminates the while loop,
$$\text{makespan}(S'_\ell) = T = OPT(I'_\ell)$$

Since each of the $\leq k$ jobs on a machine loses $< T/k^2$ in the rounding,
$$\text{makespan}(S_\ell) < \text{makespan}(S'_\ell) + k \cdot \frac{T}{k^2}$$
$$= T + \frac{T}{k}$$
$$= \left(1 + \frac{1}{k}\right) OPT(I'_\ell)$$
$$\leq \left(1 + \frac{1}{k}\right) OPT(I), \quad \text{since } I_\ell \subseteq I, \text{ and}$$

the job sizes are rounded <u>down</u> to obtain I'_ℓ.

Thus, if the last job to finish is a long job,
$$B_k(I) < \left(1 + \frac{1}{k}\right) OPT(I).$$
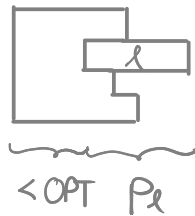
Otherwise, the last job to finish has
$$p_\ell \leq \frac{T}{k} \leq \frac{OPT(I'_\ell)}{k} \leq \frac{OPT(I)}{k}$$

Hence,
$$B_k(I) < OPT(I) + p_\ell \leq \left(1 + \frac{1}{k}\right) OPT$$

By the same argument as in the analysis of LS:

$< OPT \quad p_\ell$

Thus, in both cases, $B_k(I) < \left(1 + \frac{1}{k}\right) OPT.$

$\boxed{\text{Theorem 3.7 : } \{B_k\} \text{ is a PTAS}}$

Proof:

Let $k = \lceil \frac{1}{\varepsilon} \rceil$. Then,

B_k achieves an approx. factor of $1+\varepsilon$ with running time

$$O\left(\left(\frac{n}{\varepsilon}\right)^{\left(\frac{1}{\varepsilon}\right)^2} \log\left(p_{max}\right)\right).$$

If $\varepsilon \in O(1)$, this is poly. in the input size, since it takes $\geqslant \log(p_{max})$ bits to represent the job sizes. $\square$

$\{B_k\}$ is **not** a FPTAS, since the running time is exponential in $\frac{1}{\varepsilon}$.

Note that we did not expect a FPTAS, since the problem is strongly NP-complete...

The problem is strongly NP-complete,
meaning that even the special case where
$\exists$ polynomial q s.t. $p_{max} \leq q(n)$, for all input instances,
is NP-complete.
(This means that, in contrast to Knapsack,
$\nexists$ pseudopoly. alg., unless $P = NP$.)

This implies that $\boxed{\nexists \text{FPTAS}, \text{ unless } P = NP}$:

Assume to the contrary that $\exists$ FPTAS for the problem, i.e.,
$\exists$ family of algorithms $\{A_\varepsilon\}$, $\varepsilon > 0$, with approx. factor $1 + \varepsilon$
and running time poly. in n and $\frac{1}{\varepsilon}$.

Consider the special case of the problem where
$\exists$ polynomial q s.t. $p_{max} \leq q(n)$, for all instances.
In this case, $P \leq n \cdot q(n) \equiv p(n)$.

For $\varepsilon = \frac{1}{p(n)}$,

$-$ $\frac{1}{\varepsilon} = p(n)$. Since the running time of A_ε is poly. in
$\frac{1}{\varepsilon}$ and n, the running time of A_ε is a poly. of n.

$-$ $A_\varepsilon(I) \leq (1 + \frac{1}{p(n)}) \cdot OPT(I)$, for any input I
$< OPT(I) + 1$, since $OPT(I) < P \leq p(n)$

Thus, since $A_\varepsilon(I)$ is integer, $A_\varepsilon(I) = OPT(I)$.

If $P \neq NP$, this contradicts the fact that the
problem is strongly NP-complete.