Approximation Algorithms

EXERCISES

Elad Aigner-Horev

EXE 1:

1.1 Assume that the edges of K_n are weighted using $\{1, 2, \ldots, K\}$ for some $K \in \mathbb{N}$. Prove that any Hamilton cycle in such a weighted graph forms a K-approximation for an optimal TSP Tour

1.2 Assume that the edges of K_n are weighted using a weight function ω satisfying

$$\omega(v_1, v_K) \leq C \cdot \left(\omega(v_1, v_2) + \ldots + \omega(v_{K-1}, v_K) \right)$$

whenever $(v_1, \ldots, v_K)$ is a path and where $0 < C \in \mathbb{R}$ is some fixed constant.

Devise a $1.5C$-approximation algo. for the TSP problem in such instances

1.3 Assume that the edges of K_n are weighted using $\{1, \ldots, K\}$ for some $K \in \mathbb{N}$.

Devise a $0.75 \cdot K$-approximation algo for the TSP-problem in such a graph

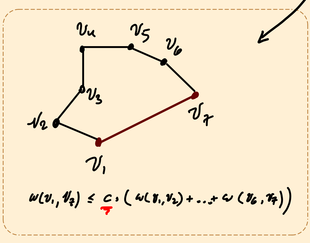
Solution 1:

1.1 Any Hamilton cycle in the aforementioned weighted graph has weight at most $k \cdot n$. On the other hand, any TSP-tour has weight at least n. The claim follows.

1.2 We prove that executing Christofides' algorithm on the graph fetches the desired approximation ratio.

The Christofides algo takes a min. spanning tree (MST, henceforth) and a certain matching M (see details in the algo) and then proceeds to produce a TSP-tour via shortcutting edges in the union graph: $MST + M$.

In the instances considered in this exercise, each shortcut performed by the algo may lead to an increase by a factor of c to the actual used to connect the endpoints connected by the path (see figure).



Writing OPT to denote the weight of an optimal TSP-tour, the bound $w(MST) \leq OPT$ still holds (see lectures for an explanation). In the worst case, each edge of the MST is shortcutted, hence overall we incur a weight of $\leq c \cdot OPT$ from the MST and shortcuts performed to it. By the same token it follows, that $w(M) \leq c \cdot \frac{OPT}{2}$.

(In the lemma appearing in the slides pertaining to $w(M)$, note that here we would have $w(T') \leq c \cdot w(T)$, with T, T' as defined in the lemma)

Overall, the weight of the tour produced is at most

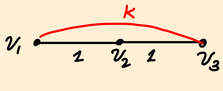
$$w(MST + \text{shortcuts}) + w(M) \leq c \cdot OPT + \frac{c}{2} \cdot OPT = 1.5 \cdot c \cdot OPT$$

as required.

1.3 Let w denote the edge-weight function of the given graph. Then

$$w(v_1, v_\ell) \leq \frac{k}{2} \cdot \left(w(v_1, v_2) + \ldots + w(v_{\ell-1}, v_\ell) \right)$$

holds, whenever $(v_1, \ldots, v_\ell)$ is a path in the graph. To see this, note that as the edge weights are all ≥ 1, it suffices to prove the above claim for $\ell = 2$ (as this would be the sole case where the inequality can be tight). Indeed, the worst case is



and indeed $k \leq \frac{k}{2}(1+1)$ (in fact $=$ holds)

By part 1.2 (to this problem) there is an algo for the TSP-problem having approx. ratio $\leq 1.5 \cdot \frac{k}{2} = 0.75 \cdot k$ as required

EXE 2:

For a proper vx-colouring $\varphi : V(G) \to \mathbb{N}$ of a graph G, write

$$S_\varphi := \sum_{v \in V(G)} \varphi(v)$$

to denote the sum of φ. For a graph G, define

$$S(G) := \min_{\varphi} S_\varphi$$

where the minimum ranges over all proper vx-colourings of G.

2.1 Prove that if there exists a (proper) vx-colouring φ of a graph G s.t.

$$\sum_{v \in V(G)} \varphi(v) \leq S \in \mathbb{N},$$

then there exists a set $U \subseteq V(G)$ s.t.

$$|U| \geq v(G)/2 \quad \text{and} \quad \varphi(u) \leq \frac{2S}{v(G)} \quad \forall u \in V(G).$$

2.2 Given a vx-colouring φ of G satisfying $\sum_v \varphi(v) \leq S$.

Prove that at least $\frac{v(G)}{2}$ of the vxs are coloured using at most $\frac{2S}{v(G)}$ colours under φ

2.3 Let $\alpha \geq 1$.

Prove that if there is an α-approx poly algo for computing $S(G)$, then there is an $O(\alpha \log n)$-approx algo for computing $\chi(G)$

Solution 2:

2.1 Suppose that the claim is false. Set

$$U := \left\{ u \in V(G) : \varphi(u) \leq \frac{2S}{V(G)} \right\}.$$

Then, $|U| < \frac{V(G)}{2}$, by assumption.

Then,

$$S \geq \sum_v \varphi(v) = \underbrace{\sum_{v \in U} \varphi(v)}_{\geq 0} + \underbrace{\sum_{v \notin U} \varphi(v)}_{|V(G) \setminus U| > \frac{V}{2}} > \frac{V(G)}{2} \cdot \frac{2S}{V(G)} = S$$

$\varphi(v) > \frac{2S}{V(G)} \quad \forall v \notin U$

and a contradiction is reached

2.2 By part 2.1, the set

$$U := \left\{ u \in V(G) : \varphi(u) \leq \frac{2S}{V(G)} \right\}$$

satisfies $|U| \geq \frac{V(G)}{2}$. The claim follows.

2.3 Let A_α denote the α-approx algo promised in the premise. Consider the following algo for computing $\chi(G)$:

(a) Apply A_α as to obtain a colouring φ satisfying

$$S_\varphi \leq \alpha S(G)$$

(b) By part 2.2, using φ, at least $\frac{V(G)}{2}$ of the vxs are coloured using $\leq \frac{2\alpha S(G)}{V(G)}$ colours. Let $U \subseteq V(G)$ denote this subset of vxs.

(c) Set $G_i := G - U$.

(d) If $V(G) > 0$, return to step (a) otherwise halt.

(*) In each iteration we colour using a new pallete of colours as to retain the validity of the resulting colouring.

In each iteration, the algo discards $\geq \frac{V(G)}{2}$ vxs (as being coloured). Consequently, at most $O(\log n)$ iterations are performed.

For iteration i, write

$G_i :=$ the subgraph of G associated with iteration i

$\varphi_i :=$ the colouring found by A_α in the ith iteration.

As $S(G) \leq \chi(G) \cdot V(G)$, $S(G_i) \leq \chi(G) \cdot V(G_i)$ holds $\forall i$.

In addition, $S_{\varphi_i} \leq \alpha S(G_i)$ is known to hold.

Then,

$$\#\text{ of colours in the resulting colouring} \leq \sum_{i=1}^{O(\log n)} \frac{2 S_{\varphi_i}}{V(G_i)} \leq \sum_{i=1}^{O(\log n)} \frac{2\alpha \chi(G) \cancel{V(G_i)}}{\cancel{V(G_i)}}$$

$$= O(2\alpha \log n)\, \chi(G)$$

and the claim follows.

EXE 3:

Given a graph G with $\alpha(G) \geq 3/4\, V(G)$, devise an efficient algo for producing an independent set of size $\geq V(G)/2$

(*) An independent set of size $\geq 3/4\, V(G)$ is **not** given to you here!

Solution 3:

As $\alpha(G) = |V(G)| - \tau(G)$, the assumption that $\alpha(G) \geq \frac{3}{4}|V(G)|$ implies that $\tau(G) \leq \frac{|V(G)|}{4}$. Applying a 2-approx. algo for the min vx-cover problem in G produces a vx-cover C of size $\leq \frac{|V(G)|}{2}$. As $V(G) \setminus C$ is independent the claim follows.

EXE 4:

Let $0 < \beta \leq 1$ be given.

Prove that if there is a poly β-approx. algo for computing $\alpha(G)$, where G is a graph, then there is an $O\left(\frac{\ln n}{\beta}\right)$-approx poly algo for computing $\chi(G)$

Solution 4:

Let A_β denote the algorithm presumed to exist in the premise. Consider the following algo.

Input: a graph G
Output: a (valid) vx-colouring of G

1. $c := 1$ // current colour
2. while $(V(G) \neq \emptyset)$ do ;
   - 3.1 $I := A_\beta(G)$
   - 3.2 Colour **all** vxs in I using the colour c
   - 3.3 $c := c+1$
   - 3.4 $G := G - I$
3. Return the resulting colouring

- It is trivial to see that the algo produces a valid vx-colouring in poly-time (check)

Claim: The colouring produced uses $\leq O\left(\frac{\log n}{\beta}\right)\chi(G)$ colours

Proof:

Observe that:

$$\#\text{ of colours used} = \#\text{ iterations performed}$$

Write G_i to denote the subgraph of G remaining at the end of the ith iteration. Put $n := v(G)$ and $n_i := v(G_i)$.

Owing to the triviality

$$\alpha(G) \geq \frac{n}{\chi(G)}$$

satisfied by **every** graph we may proceed to write as follows:

$n_0 = n$

$$n_1 \leq n_0 - \underbrace{\beta\,\alpha(G_0)}_{A_\beta(G) \text{ returns an IS of size} \geq \beta\alpha(G)} \leq n_0 - \beta\cdot\frac{n_0}{\chi(G_0)} = n - \frac{\beta n}{\chi(G)} = n\left(1-\beta/\chi(G)\right)$$

($\alpha(G_0) \geq \frac{n_0}{\chi(G_0)}$)

$$n_2 \leq n_1 - \beta\,\alpha(G_1) \leq n_1 - \beta\cdot\frac{n_1}{\chi(G_1)} = n_1\left(1-\frac{\beta}{\chi(G_1)}\right) \leq n_1\left(1-\beta/\chi(G)\right)$$

($\chi(G_1) \leq \chi(G)$)

$$\leq n\left(1-\beta/\chi(G)\right)^2$$

Induction now yields (do it) that

$$n_i \leq n\left(1-\beta/\chi(G)\right)^i$$

To estimate the # of iterations we solve $n_i < 1$ for i:

Set $r := \beta/\chi(G)$ for convenience. We seek to find a lower bound for i for which $(1-1/r)^i < 1/n$ holds.

Write $i := kr$ with $k := k(n)$ to be determined later on.

Then:

$$(1-1/r)^i = \left((1-1/r)^r\right)^k \leq e^{-k}$$

In order to have $e^{-k} < 1/n$ it suffices to set $k = \Omega(\ln n)$

It follows that $n_i < 1$ for $i = \Omega(\log n)\cdot r = \Omega\left(\frac{\log n}{\beta}\right)\chi(G)$

completing the argument

∎

EXES:

Let f denote the poly. reduction $\text{NAE-3-CNF-SAT} \leq_p \text{3COL}$.

5.1 Prove that for every 3-CNF formula, $\chi(f(\varphi)) \leq 4$.

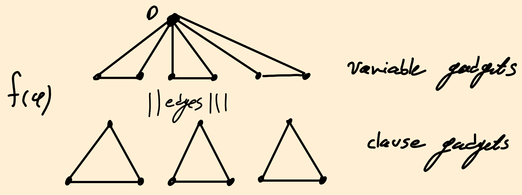
(Recall that $f(\varphi)$ is a graph)

5.2 Prove that if $P \neq NP$, then there is no $(\frac{4}{3}-\varepsilon)$-approx alg for computing $\chi(a)$ for any $\varepsilon > 0$

Put another way, prove that if the aforementioned $(\frac{4}{3}-\varepsilon)$-approx alg exists, then $\text{NAE-3-CNF-SAT} \in P$

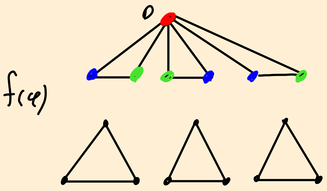
Solution 5 :

5.1 The general structure of $f(\varphi)$



Colour 0 using, say, red.

Pick two other colours, say, blue and green and colour all variable gadgets



To complete the colouring, fix a clause gadget.
At least two of the vxs of the clause gadget can be coloured using blue and green. Indeed, every vx in the clause gadget is connected to precisely one vx from all vxs spanned by all variable gadgets. Hence, one can colour all remaining uncoloured vxs in clause gadgets using the 4th (and thus far) never used colour.

5.2 From the soundness of f, we deduce that

φ is NAE-satisfiable $\Rightarrow$ $\chi(G_\varphi) = 3$

φ is not NAE-satisfiable $\Rightarrow$ $\chi(G_\varphi) = 4$

Consequently it follows that the following problem is NP-hard:

Given a graph G, determine whether $\chi(G) \leq 3$ or $\chi(G) \geq 4$

For if this problem is in P, then we can use an algo for it together with f to resolve NAE-3-CNF-SAT

Fix $\varepsilon \geq 0$ and assume that there exist a $(\frac{4}{3} - \varepsilon)$-approx. poly algo for computing $\chi(G)$. Then, the following algo resolves NAE-3-CNF-SAT in poly time.

> **Input:** φ a 3-CNF formula
> **Output:** Determination whether φ is NAE-satisfiable or not
>
> 1. Construct G_φ
> 2. Apply the $(\frac{4}{3} - \varepsilon)$-approx. algo (assumed to exist) on G_φ
>
> 2.1 If the approx. algo determines that $\chi(G) \leq 4 - 3\varepsilon$, then return "$\varphi$ is NAE-satisfiable"
>
> 2.2 Else, return that "φ is not NAE-satisfiable"

If φ is NAE-satisfiable, then $\chi(G_\varphi) = 3$ and the approx algo would return that $\chi(G) \leq (\frac{4}{3} - \varepsilon) \cdot 3 = 4 - 3\varepsilon$.

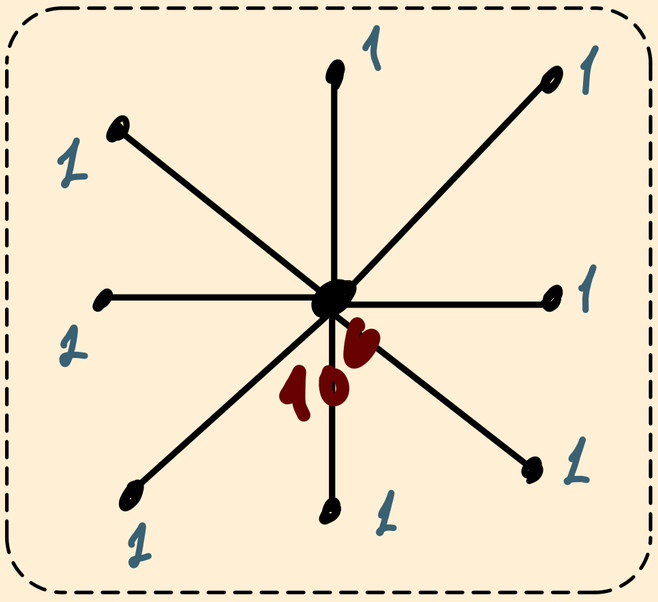
If φ is not NAE-satisfiable, then $\chi(G_\varphi) \geq 4$ and the approx algo (which we naturally assume to be correct) reports that strictly more than $4 - 3\varepsilon$ colours are required to colour G_φ

EXEC:

Draw a vx-weighted graph such that if we run the 2-approx. algo for the cardinality version of vx-cover that is based on max matchings, the output cover thus obtained would not constitute a 2-approximation for the least weight vx-cover

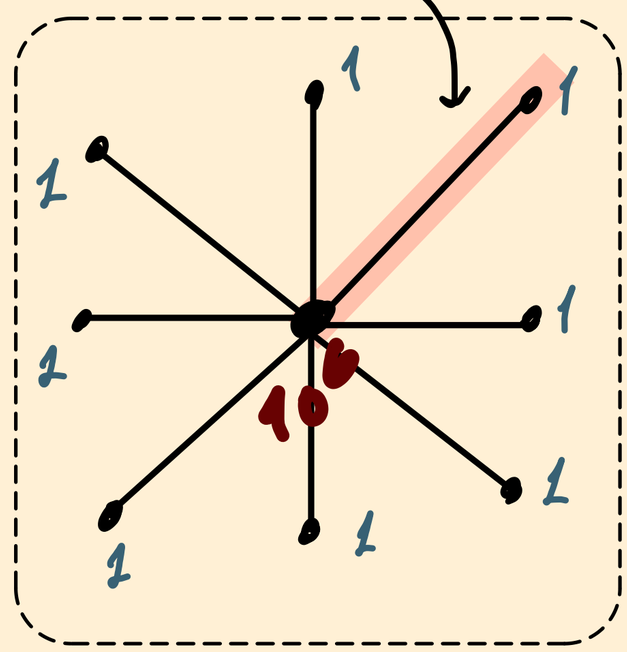
Solution 6:

Consider the following vx-weighted graph:

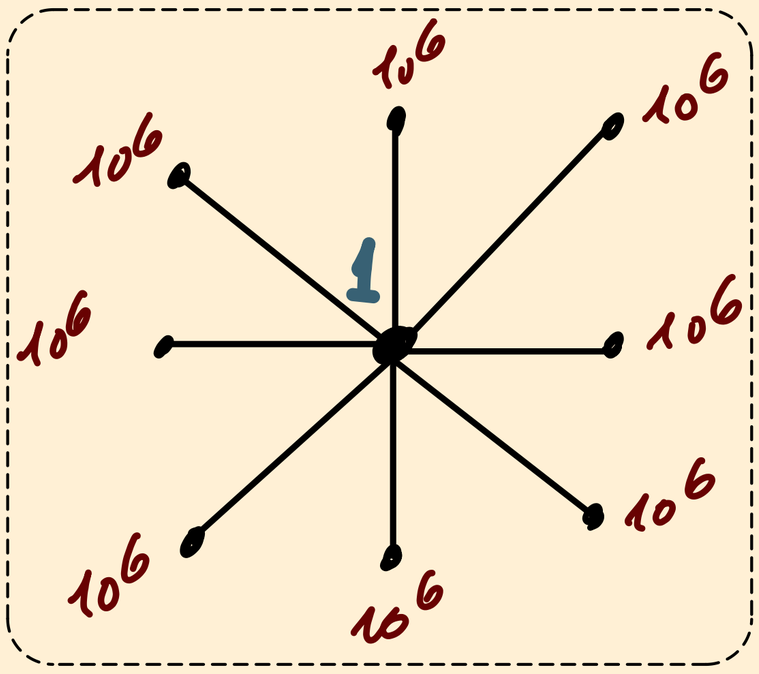


Here, $OPT = 8$ and it is given by all the vxs found on the spokes of the star.

The matching based 2-approx. algo for the cardinality version of of vx-covers returns, say, the cover which has weight $10^6 + 1$!!



This can also be seen through the following graph



Exe 7:

Recall that, for a graph G, we write

$$\alpha(G) := \text{size of a maximum independent set in } G$$

$$\tau(G) := \text{size of a minimum vx-cover in } G$$

Let A denote the max. matching based algo for min. vx-cover.

The identity $\alpha(G) + \tau(G) = V(G)$ suggests the following algo:

Algo B:

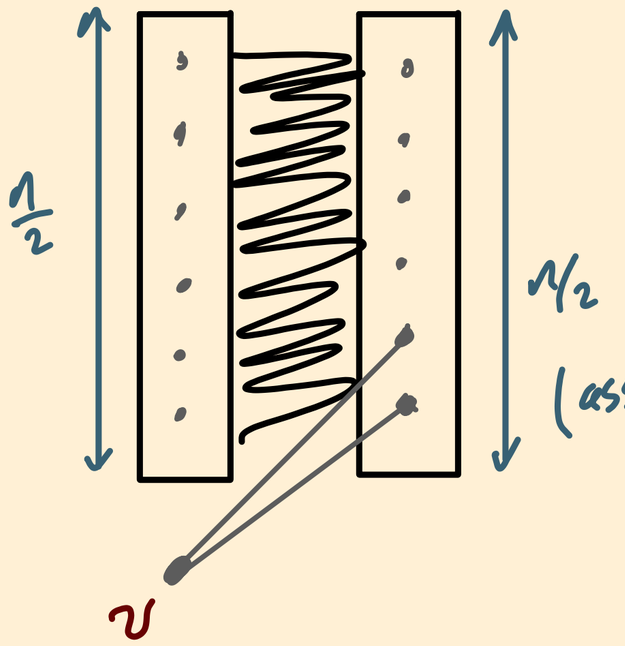
Input: graph G

Output: Independent set in G

1. $C := A(G)$
2. Return $V(G) \setminus C$ // Recall that the complement of any vx-cover is an independent set.

Draw an $(n+1)$-vx graph G for which $\alpha(G) > n/2$ yet algo. B applied to G returns an independent set of size $\frac{1}{3}$!

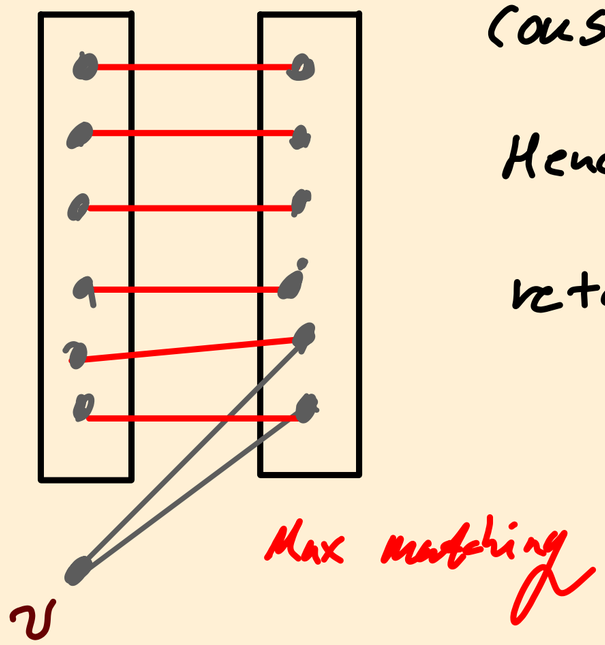
Solution 7:



Take G to be $K_{n/2,\,n/2}$ with an additional vx v added to one of it colour classes. Connect v to the other side arbitrarily.

The set $C := A(G)$ defined in alg. B is, say, consists of all vxs of G but v.

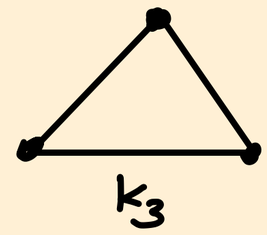
Hence, $V(G) \setminus C = \{v\}$ and alg B returns only $\{v\}$.



On the other hand, $\alpha(G) = \frac{n}{2} + 1$

Exe 8:

- By K_3 we denote the complete graph on 3-vxs.



  K_3 is also called the triangle

- A graph G is said to be K_3-free if it contains no copy of K_3

  - No copy of K_3 means:

    No subgraph of G is isomorphic to K_3

- We consider the following problem:

  Given a graph G, what is the least number of edges that one can remove from G in order to obtain a K_3-free graph

- Devise a 3-approx. poly. time alg for the problem.

Solution 8:

- The algo proposed is the following greedy algo:
  - While there is a triangle in the residual graph:
    - Find a triangle T in the residual graph
    - Remove $E(T)$ from the set of remaining edges

- That the above algo is poly. is trivial and left to the reader
- It is also trivial to see that when the algo terminates, the residual graph is K_3-free.

Claim:

The above algo is a 3-approx. for the aforementioned problem

Proof:

- The edges removed by the algo form a collection of edge-disjoint triangles in G.
  - Let S denote the collection of edge-disjoint triangles removed by the algo.
    - In particular, the algo removes $2|S|$ edges from G.
- Any feasible solution must meet each member of S in at least one edge.
  - Hence, $OPT \geq |S|$, where OPT is the size of an optimal solution
  - Then, # edges removed by algo $= 3|S| \leq 3 \cdot OPT$

∎

Exe 9:

In Exe 8 we considered the cardinality version of the triangle removal problem. In this exercise, we consider the weighted version of this problem.

∴ Let G be a graph and let $\omega : E(G) \to \mathbb{R}_{\geq 0}$ be a non-negative weight function set on the edges of G.

Amongst all sets $R \subseteq E(G)$ for which $G - R$ is K_3-free (Defined in Exe 8), we seek to find one which has the least weight.

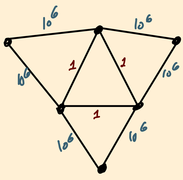
∴ Devise a poly. time 3-approx. algo for the above problem.

A) Show that the greedy algo presented in Exe 8 for the cardinality problem fails to deliver such an approximation (Hint: See Exe 6)

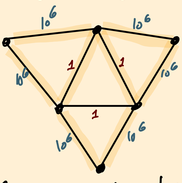
B) Use linear programming in order to devise the aforementioned algorithm (Hint: Study the Vx-cover problem)

Solution 9:

A) Consider the following edge-weighted graph



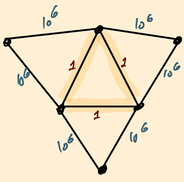
The greedy algo from EXE 8 could return



ALL EDGES

(Explain why)

An optimal solution, however has weight 3!



B)

∴ We start with an ILP for the problem

ILP

$$\min \sum_{e \in E(G)} x_e \cdot w(e)$$

$x_e + x_f + x_g \geq 1$ whenever $\{e,f,g\} \subseteq E(G)$ and form a triangle

$x_e \in \{0,1\} \quad \forall e \in E(G)$

→ Remark: The above ILP has a polynomial number of constraints. However, there are $2^{e(G)}$ possible assignments

CLAIM:

1. An optimal assignment for the ILP above defines a set of edges in G that form an optimal solution to the problem
2. A set of edges of G forming an optimal solution to the problem defines an optimal assignment for the ILP

Proof:

1. Let $x^* := (x^*_e)_{e \in E(G)}$ be an optimal assignment for the ILP. Then,

$$R := \{ e \in E(G) : x^*_e = 1 \}$$

forms an optimal solution to the problem.

By the first constraint in the ILP, R meets the edge-set of every triangle of G so that $G - R$ is K_3-free. This shows that R is a feasible solution.

To show optimality of R, assume towards a contradiction that R is not optimal and let R' be an optimal solution for the problem. Define an assignment $y := (y_e)_{e \in E(G)}$ given by:

$$y_e := \begin{cases} 1, & e \in R' \\ 0, & e \notin R' \end{cases}$$

Show that y is a feasible solution for the ILP.

Show that

$$\sum_{e \in E(G)} y_e w(e) < \sum_{e \in E(G)} x^*_e w(e)$$

which is a contradiction to the optimality of x^*

Continue on the next page →

Solution 9: (Continue)

- Relax the ILP as follows:

LP

$$\min \sum_{e \in E(G)} x_e \cdot w(e)$$

$$x_e + x_f + x_g \geq 1 \quad \text{whenever } \{e,f,g\} \subseteq E(G) \text{ and form a triangle}$$

$$0 \leq x_e \leq 1 \qquad \forall e \in E(G)$$

Remark: Instead of $0 \leq x_e \leq 1$ we could just write $x_e \geq 0$. If we do so, then we must argue that any solution x with $x_e > 1$ for some $e \in E(G)$ we can alter by setting $x_e = 1$ and obtain a "better" solution (why?)

- We use the LP above in order to formulate our 3-approx. algo.

Step 1: Solve LP optimally. Let x^* be an optimal solution for the LP.

Step 2: Construct $R := \{ e \in E(G) : x^*_e \geq 1/3 \}$

Step 3: Return R

- Next, we handle feasibility of the solution returned by our algo.

Claim:
$R :=$ the set defined by the algo above.
Then, $G - R$ is K_3-free

Proof:

- Fix a triangle (e,g,f) in G (i.e., $e,f,g \in E(G)$)
- Let x^* be as defined in the algo.
- As x^* is a feasible solution for the LP:

$$x^*_e + x^*_f + x^*_g \geq 1$$

Hence, at least one of the variables x^*_e, x^*_f, x^*_g is at least $1/3$.

- It follows that the set R meets the edge-set of every triangle in G.

∎

- Finally, we deal with the approx. ratio of our algo.

Claim: The above algo is a 3-approx. algo for the problem.

Proof:

- $OPT :=$ the weight of an optimal solution
- R, $x^* :=$ as defined in the algo.
- $y := (y_e)_{e \in E(G)}$ denote the rounding of x^* performed by the algo. That is:

$$y_e := \begin{cases} 1, & x^*_e \geq 1/3 \\ 0, & x^*_e < 1/3 \end{cases}$$

- Note that

$$w(R) = \sum_{e \in E(G)} y_e \, w(e)$$

- Note also that by definition, $y_e \leq 3 x^*_e$ holds for every $e \in E(G)$.

- We may now write

$$w(R) = \sum_e y_e \, w(e) \leq \sum_e 3 x^*_e \, w(e) = 3 \cdot OPT_f \leq 3 \cdot OPT$$

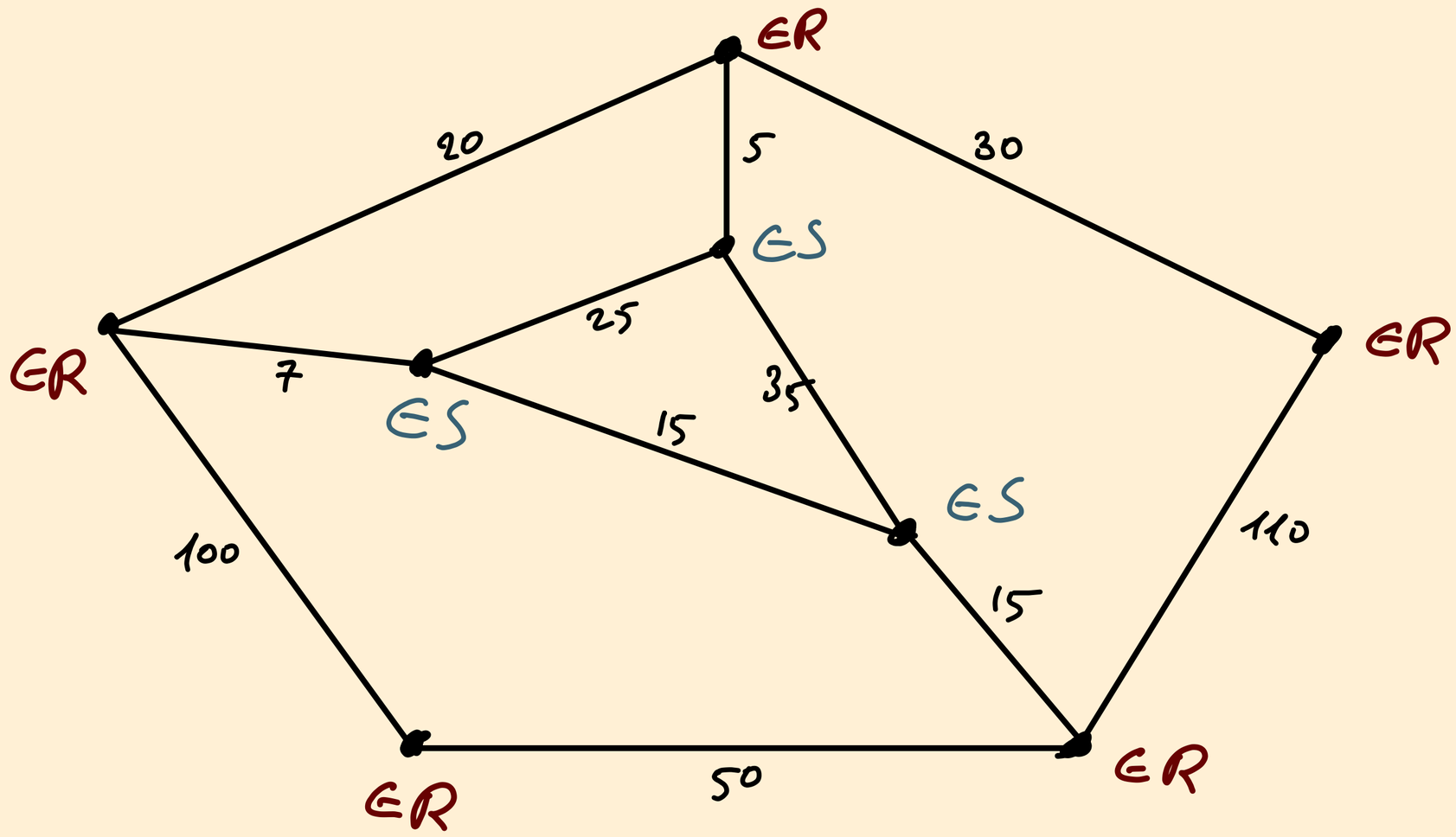
where OPT_f is the value of an optimal solution of the LP and OPT is the optimal value of the ILP.

The last inequality is due to the triviality $OPT_f \leq OPT$.

∎

EXE 10:

Execute the 2-approx. algo shown for the general Steiner tree problem on the following graph. SHOW YOUR WORK!

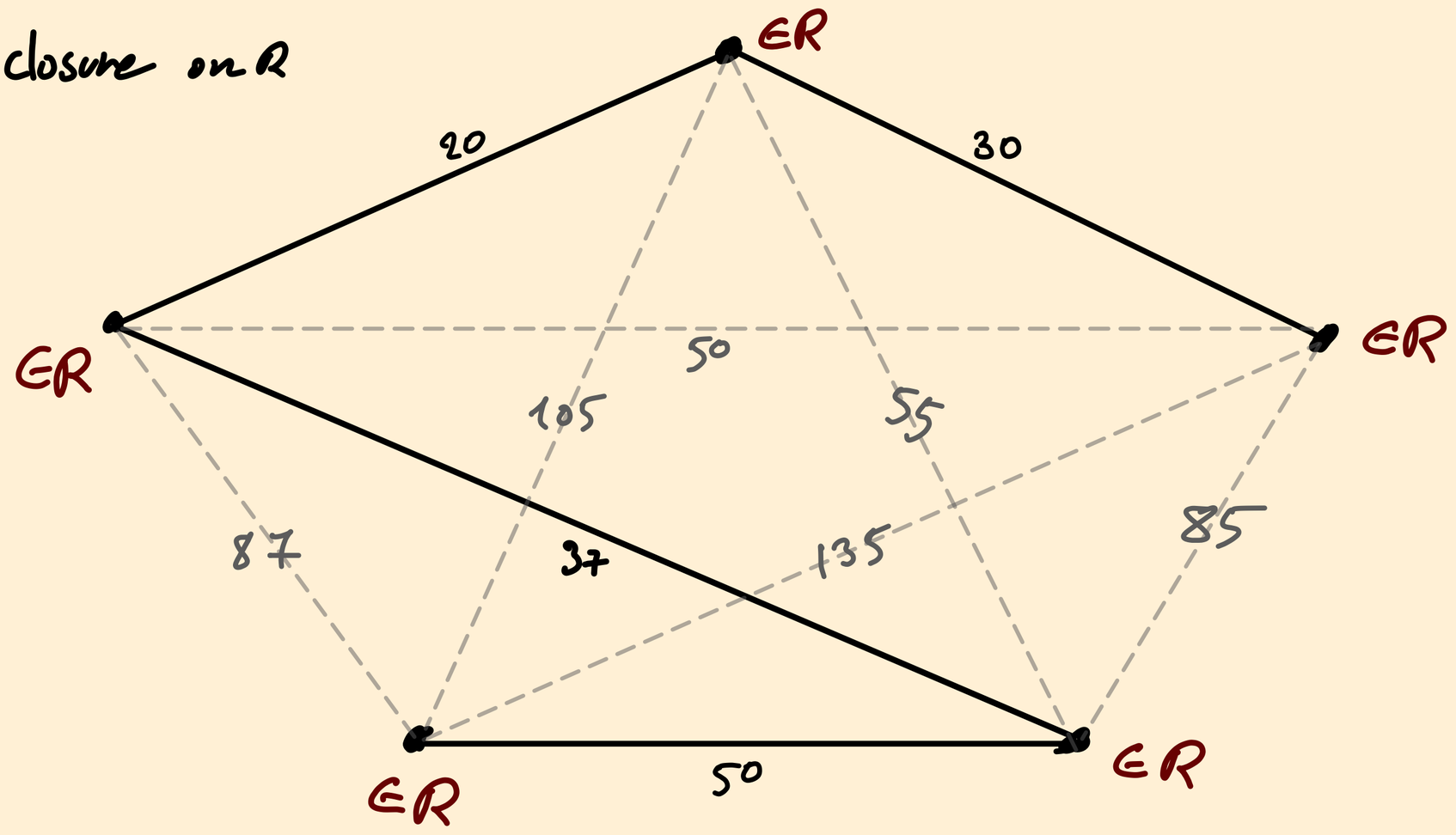


Solution 10:

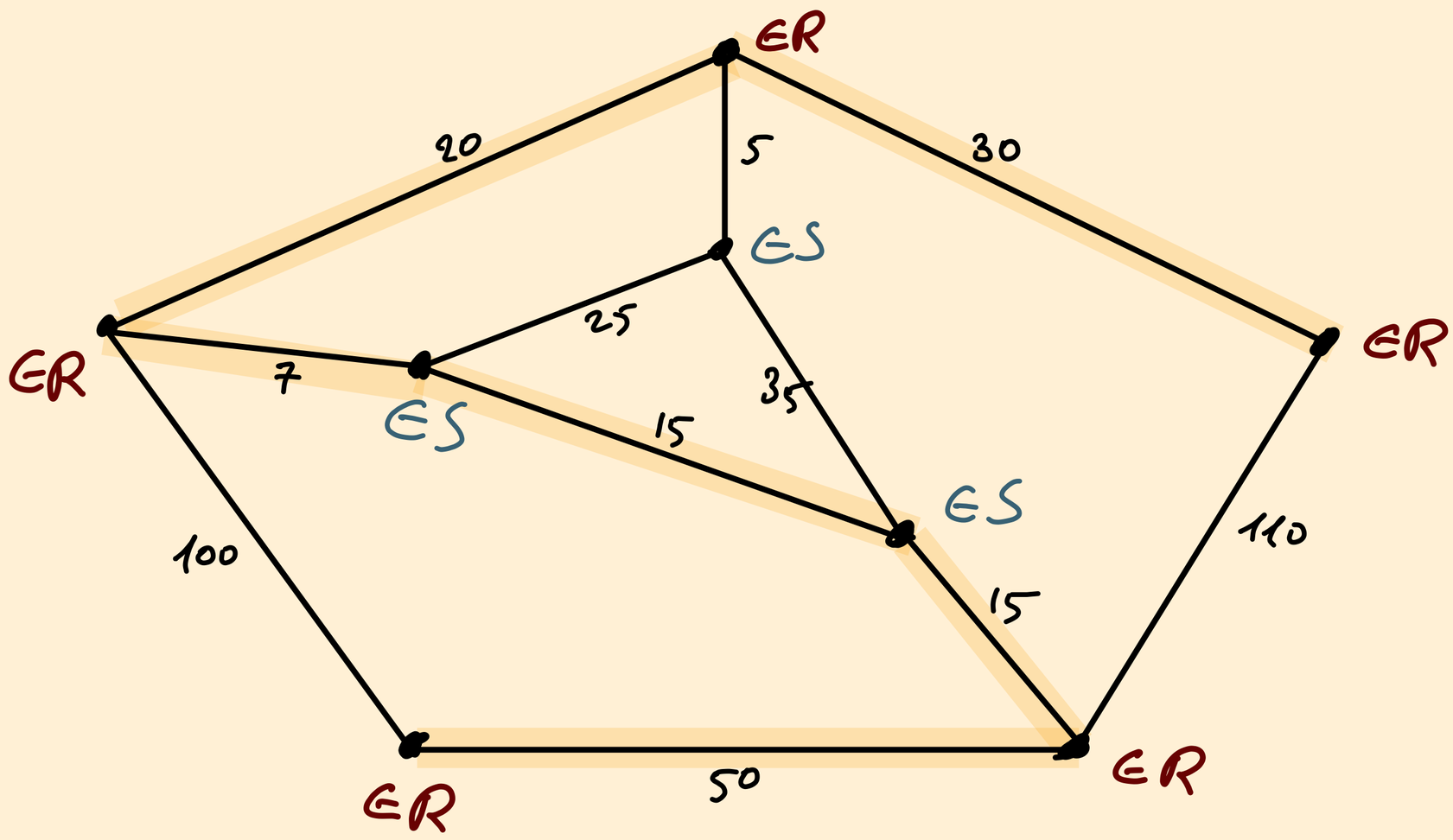
Metric closure on R

+

MST



The resulting Steiner tree



EXE 11:

A) For the metric Steiner tree problem, an MST over the required points formed a 2-approximation (see exact details in the approx. algs. for this problem).

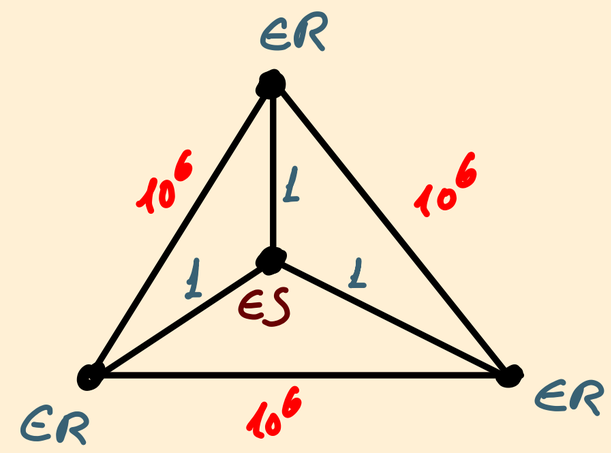
Show that in the GENERAL Steiner tree problem this is **not** the case.

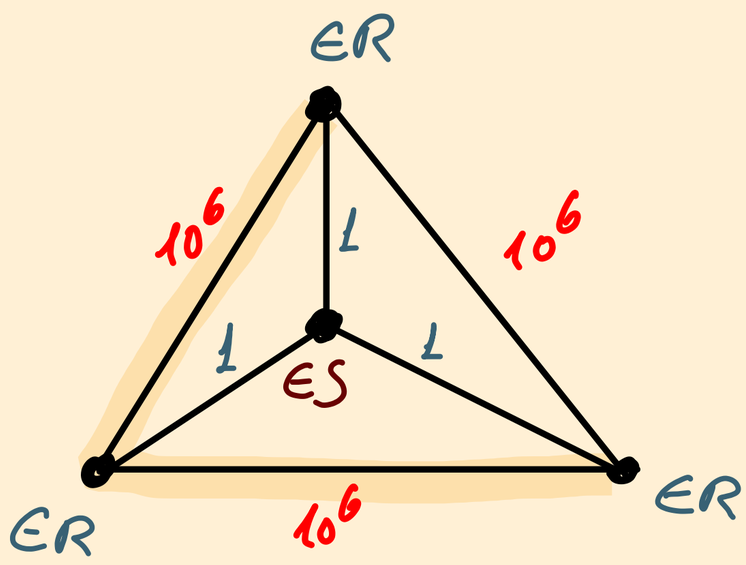
B) Draw 3 <u>different</u> instances for the <u>metric</u> Steiner problem such that for all you have:

- $S \neq \emptyset$ (i.e., there are Steiner points)
- The MST on R alone has larger weight than the optimal solution

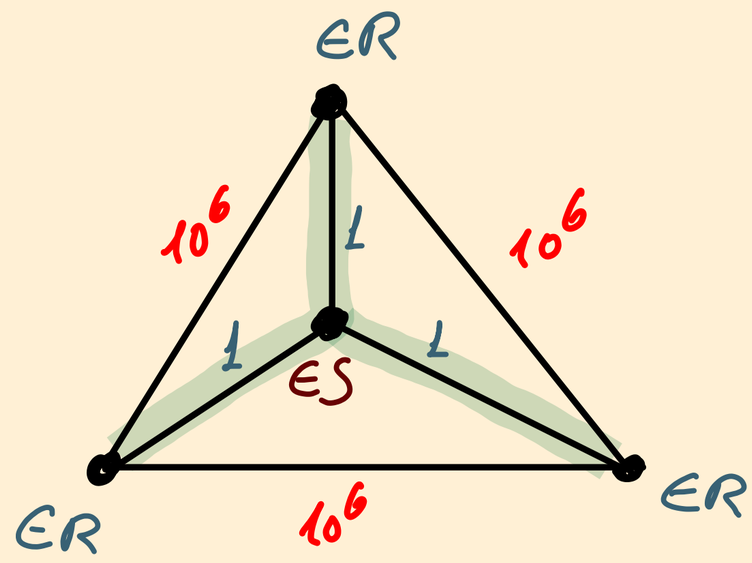
Solution 11:

A) Consider the following graph



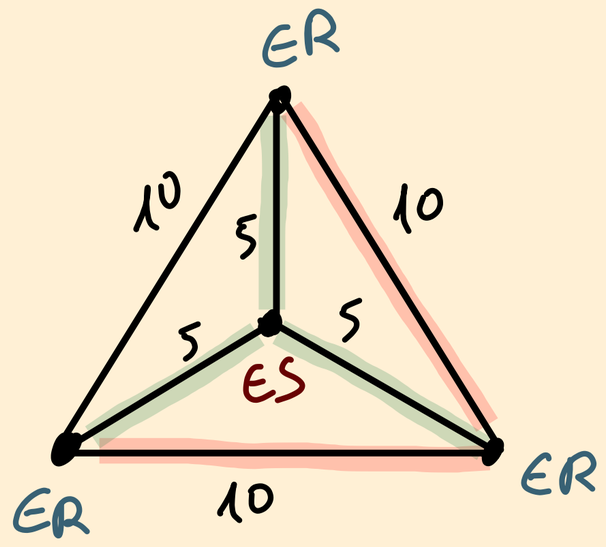


MST on R



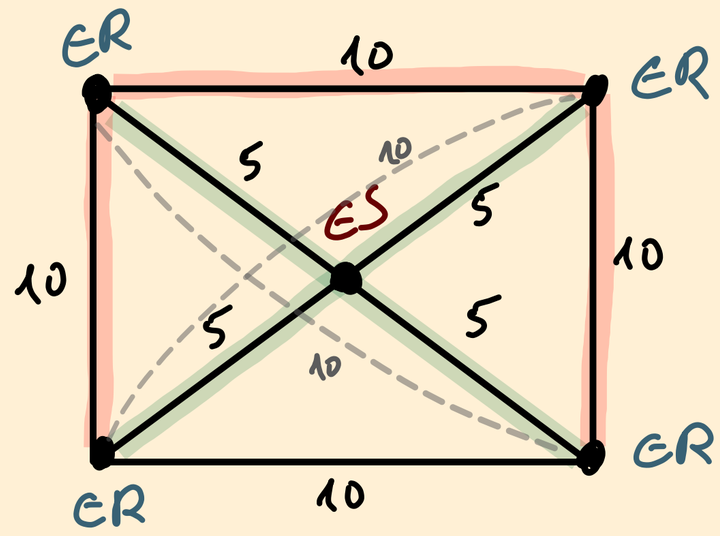
Optimal solution

B)



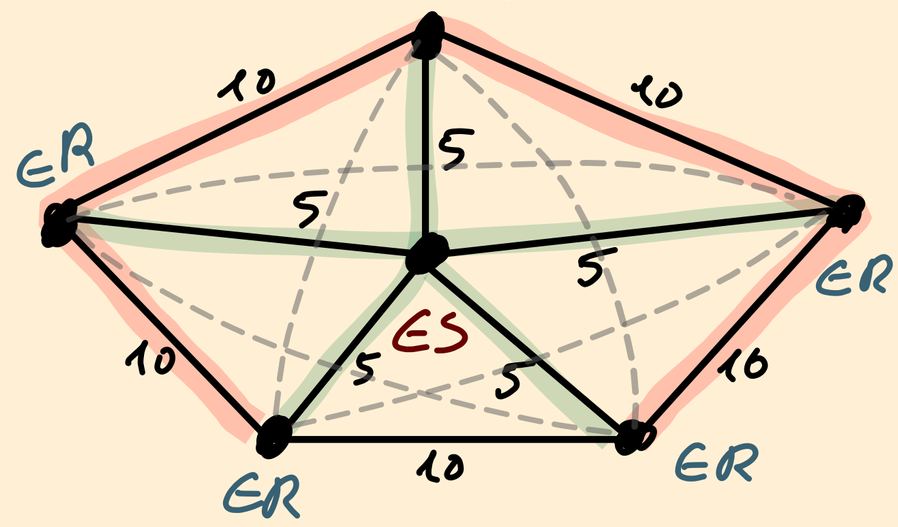
MST on R 20

optimal 15



MST on R 30

optimal 20



MST on R 40

optimal 25

All dotted lines are missing edges that also exist. Their weight is also 10. Note that this means that our metric is not Euclidean as $\sqrt{10^2 + 10^2} = \sqrt{200} = 10\sqrt{2} > 10$

EXE 12:

The purpose of this exercise is to devise a $2(1-1/n)$-approx. algo for the Metric Steiner tree problem where n denotes the number of required vxs.

More specifically, given an instance of the metric Steiner tree problem with n required vxs, prove that a minimum spanning tree of R forms a $2(1-1/n)$-approximation of the optimal Steiner tree

Hint: Consider the proof establishing the ratio of approximation of 2 for the Metric Steiner tree problem.

At the time of writing these lines the aforementioned argument can be found in Proposition 7.19.

In that proof, think of removing something from the Hamilton cycle defined there

Solution 12:

- In what follows, we relay on the notation seen in the proof establishing the 2 approx. ratio for the Metric Steiner tree problem

  - At the time of writing these notes the aforementioned proof can be found in Proposition 7.19 in the course booklet.

- Consider the Hamilton cycle C of $G[R]$ defined in the aforementioned proof.

  - Recall that $w(C) \leq 2 \cdot OPT$

  - By the pigeonhole pricciple there is an edge $e \in E(C)$ satisfying $w(e) \geq w(C)/n$.

  - Let J be an MST of $G[R] - e$

  - Then,

$$w(J) \leq w(C) - \frac{w(C)}{n} = w(C)\left(1 - \tfrac{1}{n}\right) \leq 2 \cdot \left(1 - \tfrac{1}{n}\right) \cdot OPT$$

□

Exe 13:

Devise a 2-approx. algo. for the following problem.

Given a connected graph G, a non-negative weight function $w : E(G) \to \mathbb{R}_{\geq 0}$, and two non-empty disjoint sets $V(G) = S \cup R$

(R := receivers S := senders)

find a minimum weight subgraph $G' \subseteq G$ having the property that for every $r \in R$ there is at least one $s \in S$ for which G' contains an rs-path.

Hint: Reduce to the general Steiner tree problem

Solution 13:

Consider the following reduction to the Steiner tree problem.

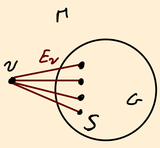
Algorithm:

Step 1: (Building Γ)

Define a graph Γ satisfying:

$V(\Gamma) = V(G) \cup \{v\}$, where $v \notin V(G)$

$E(\Gamma) = E(G) \cup E_v$, where $E_v := \{vs : s \in S\}$



Extend w into $w' : E(\Gamma) \to \mathbb{R}_{\geq 0}$ given by

$$w'(e) = \begin{cases} w(e), & e \in E(G) \\ 0, & e \in E_v \end{cases}$$

Define $R' := R \cup \{v\}$.

Step 2: Apply the 2-approximation algo for the general Steiner tree problem on: Γ, w', R' and S with R' forming the required set.

Let T denote the resulting Steiner tree from this application.

Step 3: Return $T - v$

Claim: The algo. above forms a 2-approx. for the problem

Proof:

- Feasibility of $T - v$ arises from the fact that T contains an rv-path for every $r \in R$. Any such path must pass through S as $N_\Gamma(v) = S$. Hence, $T - v$ retains the property that for every $r \in R$ there is an $s \in S$ s.t. $T - v$ contains an rs-path.

- Next, we address the approximation ratio
  - $OPT :=$ an optimal solution for the sender/receiver problem
  - Define $F := E(OPT) \cup E_v$
    - Note that $(V(\Gamma), F)$ forms a valid Steiner tree for $R \cup \{v\}$ in Γ.
    - As $w(e) = 0$ whenever $e \in E_v$, it follows that
      $$w(F) = w(OPT)$$
    - Hence, $OPT_\Gamma \leq w(F) = w(OPT)$, where OPT_Γ is the weight of an optimal Steiner tree in Γ w.r.t. $R \cup \{v\}$.
  - Finally observe that $w(T) \leq 2 \cdot OPT_\Gamma$ and the claim follows.

Exe 14:

Consider the maximum weight matching problem:

Instance: A graph G and $w : E(G) \to \mathbb{R}_{\geq 0}$

Goal: A maximum weight matching

Consider the following approximation algo for this problem.

Algo:

$M = \emptyset$

while $E(G) = \emptyset$ do:

$e :=$ heaviest edge in G

$M := M \cup \{e\}$

$G := G - e - \{$all edges incident to $e\}$

return M

Prove that the above algo forms a 2-approx. for the problem.

Solution 14:

- Let M_g be the greedy solution

  Let M_o be an optimal solution

- Set:

$$F_1 := M_o \cap M_g \qquad F_2 := M_g \setminus M_o \qquad F_3 := M_o \setminus M_g$$

- For every $e \in M_o \setminus M_g$, there must be an $e' \in M_g \setminus M_o$ incident to e s.t. $w(e') \geq w(e)$. Indeed, the fact that e is not chosen by the greedy implies that upon its deletion a heavier edge incident to it was chosen.

  - We use this fact to define the mapping $f : M_o \setminus M_g \to M_g \setminus M_o$ by $f(e) = e'$

- Then

$$\sum_{e \in M_o \setminus M_g} w(e) \leq \sum_{e \in M_o \setminus M_g} w(f(e)) \leq 2 \sum_{e \in M_g \setminus M_o} w(e)$$

where the last inequality is owing to the fact that an edge $e' \in M_g \setminus M_o$ appears as the f-image, namely $f(e)$, for some $e \in M_o \setminus M_g$ as e' is incident to at most two edges in $M_o \setminus M_g$.

- We may now write

$$w(M_g) = \sum_{e \in M_g \cap M_o} w(e) + \sum_{e \in M_g \setminus M_o} w(e) \geq \sum_{e \in M_g \cap M_o} w(e) + \frac{1}{2} \sum_{e \in M_o \setminus M_g} w(e) \geq \frac{1}{2} \cdot \sum_{e \in M_o} w(e)$$

$$= w(M_o)/2$$

and the claim follows.

∎

EXE 15:

Prove that the following greedy algo is **not** a $\log n$-approx. algo for the dominating set problem (see booklet for a definition)

Algo:

$D := \emptyset$

while $V(G) \neq \emptyset$ do:

Choose $u \in V(G)$ arbitrarily

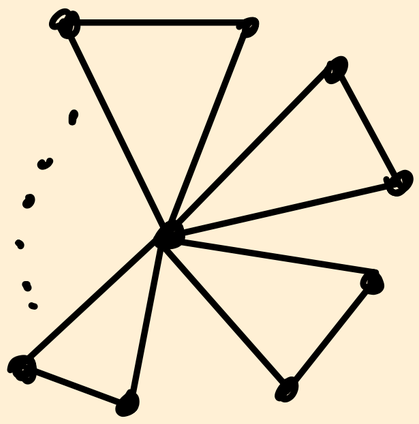
$D := D \cup \{u\}$

$G :=$ remove u and all its neighbours from G

Return D.

Solution 15:

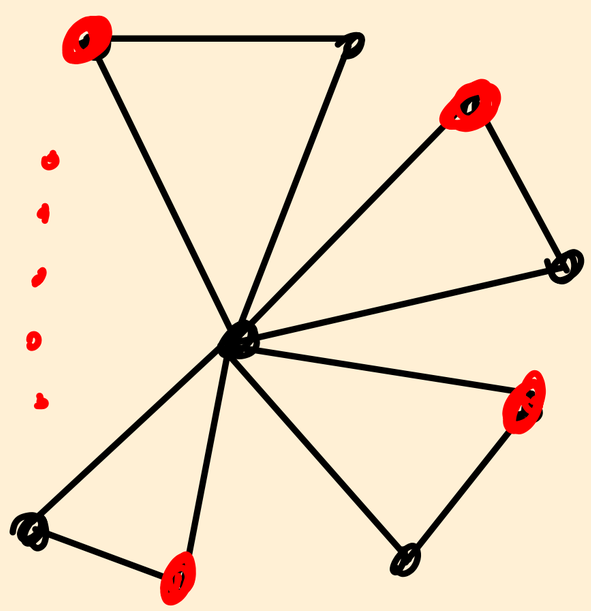
Consider the following n-vx graph



A "star" of triangles

$\frac{n-1}{2}$ triangles

Here $OPT = 1$, yet the greedy may return $\frac{n-1}{2}$ vxs



Exe 16:

A) Prove that if there exists a k-approx. algo for the minimum set-cover problem, then there is a k-approx. algo for the minimum dominating set problem

B) The minimum connected dominating subgraph problem reads as follows:

> Given a graph G, one seeks a subgraph $S \subseteq G$ s.t.
>
> (a) S is connected
>
> (b) $V(S)$ is dominating
>
> (c) Subject to (a) and (b), the quantity $e(S)$ is minimised

S must be a tree then !!

Given a connected graph G and a k-approx algo. for the min. set-cover problem, devise a $6k$-approx. algo for the min. conn. dom. subgraph problem.

Hint: Steiner trees

C) The travelling through neighbourhoods problem reads as follows.

A graph G is given. A neighbourhood tour of G is a closed walk (:= cycle with vxs allowed to repeat) C in G having the property that $V(C) \cap (N_G(v) \cup \{v\}) \neq \emptyset$ for every $v \in G$. Put another way, for each $v \in V(G)$ we either have $v \in V(C)$ or $N_G(v) \cap V(C) \neq \emptyset$.

We seek a neighbourhood tour of G of minimum length.

Devise an $O(\log n)$-approx. algo for the travelling through neighbourhoods problem

Hint:

1) There is an $O(\log n)$-approx algo for set-cover (see booklet)

2) DFS on a conn. dom. subgraph

Solution 16:

- Let G be an instance for the min. dominating set problem
  - Define the following hypergraph H_G where
    $$V(H_G) = V(G) \quad \text{and} \quad E(H_G) = \{ N_G(v) \cup \{v\} : v \in V(G) \}$$
    For future reference, set $\Gamma_v := N_G(v) \cup \{v\}$.
- Given a set-cover of H_G, namely I, note that the set of vxs $\{v : \Gamma_v \in I\}$ forms a dominating set of G.
  - Indeed, given $u \in V(G)$ let $\Gamma_v \in I$ be a set satisfying $u \in \Gamma_v$. Such a set exists by the assumption that I is a cover of H_G.
  - Conversely, given a dominating set $O \subseteq V(G)$, the set system $\{\Gamma_v : v \in O\}$ forms of set-cover of H_G. (Do this on your own)
  - We have just shown that an optimal set-cover of H_G has its size equal to the size of a least dominating set in G.
- Let A be a k-approx algo for min. set-cover and consider the following algo. for min. dominating set:

  Step 1: Given G, construct H_G as defined above

  Step 2: Set $I := A(H_G)$

  Step 3: Return $\{v \in V(G) : \Gamma_v \in I\}$

> **Claim:**
> The above algo. forms a k-approx algo for min dominating set

Proof:

- Let S_{OPT} be a set-cover of H_G of minimum size, and define
  $$O_{OPT} := \{ v \in V(G) : \Gamma_v \in S_{OPT} \},$$
  so that $|S_{OPT}| = |O_{OPT}|$.
- By definition of the algo A, $|I| \leq k \cdot |S_{OPT}|$ so that
  $$|\{v \in V(G) : \Gamma_v \in I\}| = |I| \leq k \cdot |S_{OPT}| = k \cdot |O_{OPT}|$$
  and the claim follows.

∎

B) By the 1st part of this problem, there is a k-approx. algo for the min. dominating set problem. Let $O \subseteq V(G)$ be a dominating set found using this approx. algo.

Let $S \subseteq G$ be a 2-approximate Steiner tree for O in G.

We show that S is a $6k$-approximation of the optimal subgraph. (connectivity and domination of S are trivial)

Let $C \subseteq G$ be an optimal connected dominating subgraph of G.

Let OPT_{DS} be the size of an optimal dominating set of G.

Let T be an optimal Steiner tree for O. T exists as G is connected.

Observe, now, that $e(T) \leq e(C) + |O|$. Indeed, as $V(C)$ dominates G, each $v \in O$ is either in $V(C)$ or adjacent to it. Hence, we may connect each $v \in O$ to C using at most one edge. The resulting graph is connected and a spanning tree of it has at most $e(C) + |O|$ edges and forms a steiner tree of O in G.

Hence, we may now write

$$\begin{aligned} e(S) &\leq 2 \cdot e(T) \leq 2(e(C) + |O|) \\ &\leq 2 \cdot e(C) + 2k \cdot OPT_{DS} \\ &\leq 2 \cdot e(C) + 2k \cdot (e(C) + 1) \\ &\leq 2\, e(C) + 4k\, e(C) \\ &\leq 6k\, e(C) \end{aligned}$$

where in the 3rd inequality we used the fact that $OPT_{DS} \leq e(C) + 1$ which is due to the fact that $V(C)$ is dominating and $e(C) \geq V(C) - 1$ as C is a tree yielding $e(C) \geq OPT_{DS} - 1$

∎

Solution 16: (Continue)

c) An $O(\log n)$-approx. algo for min. set-cover exists (see course booklet). Hence, by the 1st part of this problem an $O(\log n)$-approximate of the min. conn. dom. subgraph of G can be found efficiently. Let $S \subseteq G$ be such a subgraph.

A DFS-tour D of S forms a neighbourhood tour of G.

We claim that D is an $O(\log n)$-approximate of an optimal tour.

An optimal neighbourhood tour of G also forms a conn. dominating subgraph of G, by definition. Hence, $OPT_{CDS} \leq OPT_{Tour}$, where

OPT_{CDS} := size of an optimal conn. dom. subgraph of G

OPT_{Tour} := length of an optimal neighbourhood tour of G

By definition, $e(S) = O(\log n) \cdot OPT_{CDS}$. We also observe that $e(D) = 2e(S)$. The claim now follows:

$$e(D) = 2e(S) = O(\log n)\, OPT_{CDS} \leq O(\log n)\, OPT_{Tour}$$

□

EXE 17:

The following is a variant of the Hitting set problem to which we shall refer as u-Hitting set:

Let $P := \{p_1, \dots, p_n\}$ be a set. A collection $S_1, \dots, S_m \subseteq P$ of non-empty subsets of P is given s.t. $|S_i| \leq u \quad \forall i \in [m]$.

A Hitting set for $\{S_i\}_{i \in [m]}$ is a subset $H \subseteq P$ satisfying $|H \cap S_i| \geq 1$ for every $i \in [m]$.

We seek a hitting set of minimum size.

A) Formulate an ILP fitting the above program.

B) Devise an LP-based u-approx. algo for the problem.

Hint: The Hitting-set problem is simply the vx-cover problem for hypergraphs

Solution 17:

A)

$$\min \sum_{p \in P} x_p$$

$$\sum_{p \in S_j} x_p \geq 1 \quad \text{for every } j \in [m]$$

$$x_p \in \{0,1\} \quad \text{for every } p \in P$$

(explain the program)

B) We start with an LP-relaxation for the above ILP.

$$\min \sum_{p \in P} x_p$$

$$\sum_{p \in S_j} x_p \geq 1 \quad \text{for every } j \in [m]$$

$$0 \leq x_p \leq 1 \quad \text{for every } p \in P$$

Comment: We may ask for $x_p \geq 0$ instead of $0 \leq x_p \leq 1$ but it amounts to the same thing. (Explain)

The following algo. we claim is a 4-approx. algo for the problem.

Step 1: Solve the LP optimally and let x be an optimal solution for the LP

Step 2: Return $H := \{p \in P : x_p \geq 1/4\}$

Claim: The above algo. forms a 4-approximation for the problem.

Proof:

- For $p \in P$, set

$$y_p := \begin{cases} 1, & x_p \geq 1/4 \\ 0, & \text{otherwise} \end{cases}$$

  - Then, $y_p \leq 4 x_p$ for every $p \in P$

- We may now write

$$|H| = \sum_{p \in P} y_p \leq \sum_{p \in P} 4 x_p = 4 \sum_{p \in P} x_p = 4 \cdot OPT_f \leq 4 \cdot OPT$$

where OPT_f is the optimal value of the LP and OPT is the optimal value of the ILP