Web Algorithms

Eng. Fabio Persia, PhD

Approximation algorithms

- QUESTION: Suppose I need to solve an NP-hard problem. What should I do?

- ANSWER. Sacrifice one of three desired features:
  1. Solve arbitrary instances of the problem.
  2. Solve problem to optimality.
  3. Solve problem in polynomial time.

Coping strategies.

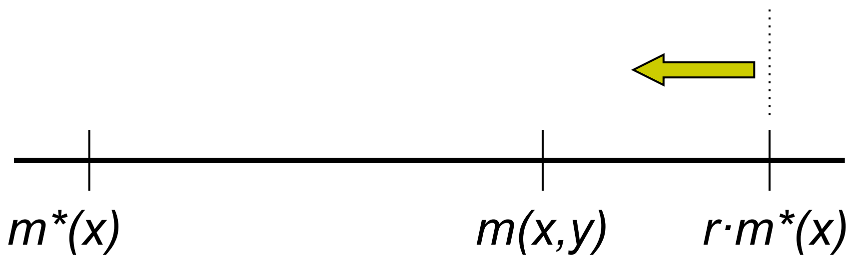
1. Design algorithms for special cases of the problem.
2. Design approximation algorithms or heuristics.
3. Design algorithms that may take exponential time.

From now on we focus on NP-hard optimization problems, that is problems that cannot be solved efficiently (unless *P=NP*)

For such problems we will design algorithms able to determine solutions that are close to optimal ones, that is good ”approximations”

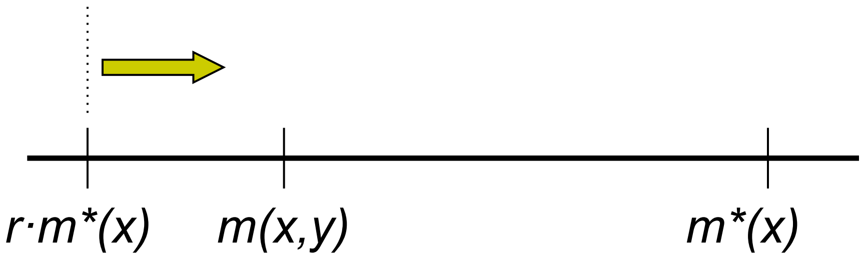
Def: Given a minimization problem π and a number $r \geq 1$, an algorithm A is an r-approximation algorithm for π if for every input $x \in I$ it always returns an *r-approximate* solution, that is a feasible solution $y \in S(x)$ such that:

$$\frac{m(x,y)}{m^*(x)} \leq r$$



Def: Given a maximization problem π and a number $r \leq 1$, an algorithm A is an r-approximation algorithm for π if for every input $x \in I$ it always return an *r-approximate* solution, that is a feasible solution $y \in S(x)$ such that:

$$\frac{m(x,y)}{m^*(x)} \geq r$$



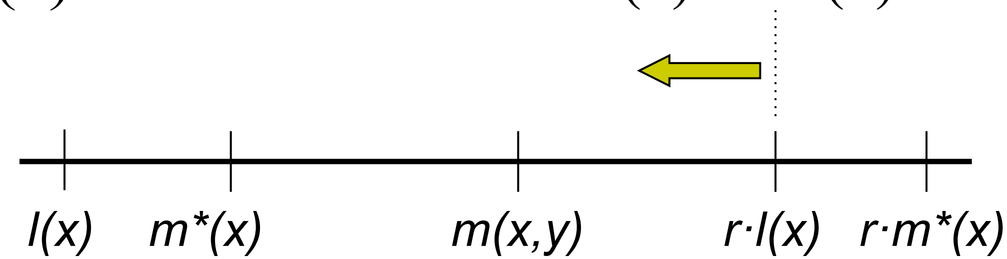
Determination of the approximation factor

How can we determine the approximation factor *r* if we don't know the value m^* of an optimal solution?

For minimization (resp. maximization) problems, we compare the value of the returned solution *m(x,y)* with a proper lower bound (resp. upper bound) *l(x)* of $m^*(x)$.

If their ratio is at most *r* for min or at least *r* for max then the algorithm is r-approximating.

Min: if $\dfrac{m(x,y)}{l(x)} \le r$ then $\dfrac{m(x,y)}{m^*(x)} \le \dfrac{m(x,y)}{l(x)} \le r$



Max: analogous

Approximation algorithm for Min Vertex Cover

Definition of Min Vertex Cover

Algorithm Approx Cover

Begin

M=Ø. *//edges chosen by the algorithm*

U=Ø. *//nodes chosen in the cover*

Repeat

Select an edge $\{u,v\} \in E$.

$U = U \cup \{u,v\}$.

$E = E \setminus \{ e \in E \mid e$ is incident to u or $v \}$.

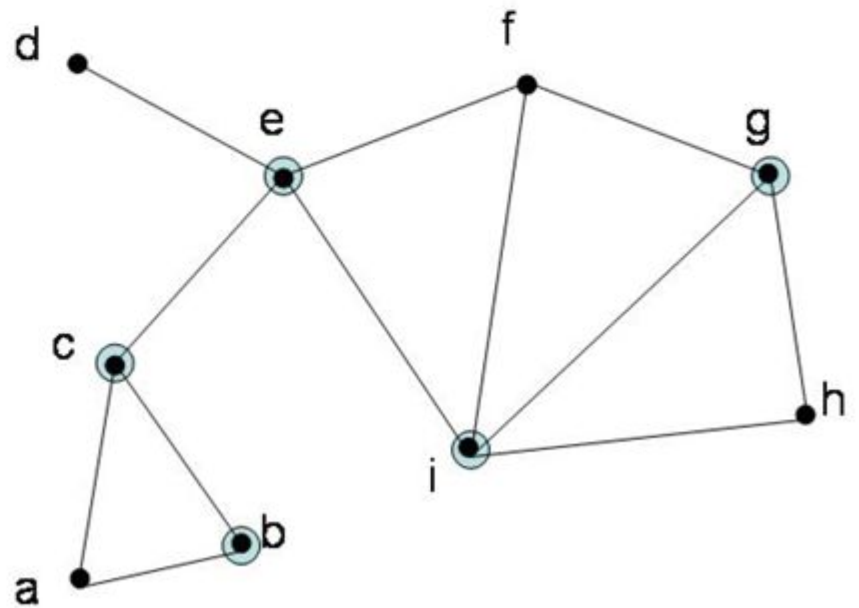
$M = M \cup \{ \{u,v\} \}$.

Until (E=Ø).

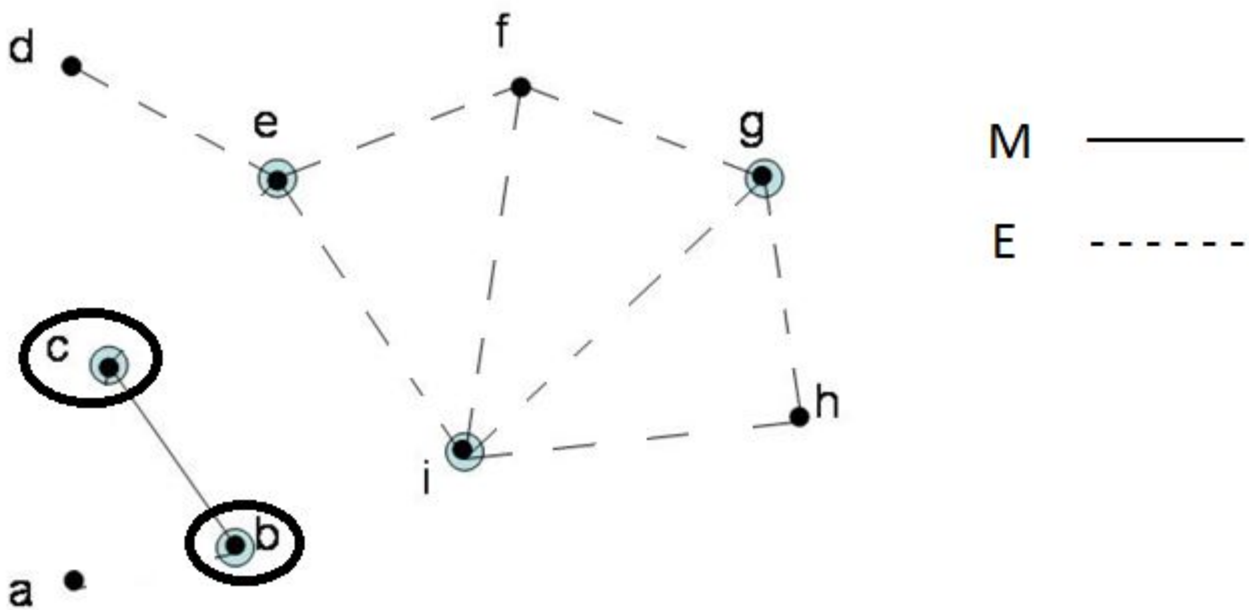
Return *U*.

End

Example of execution

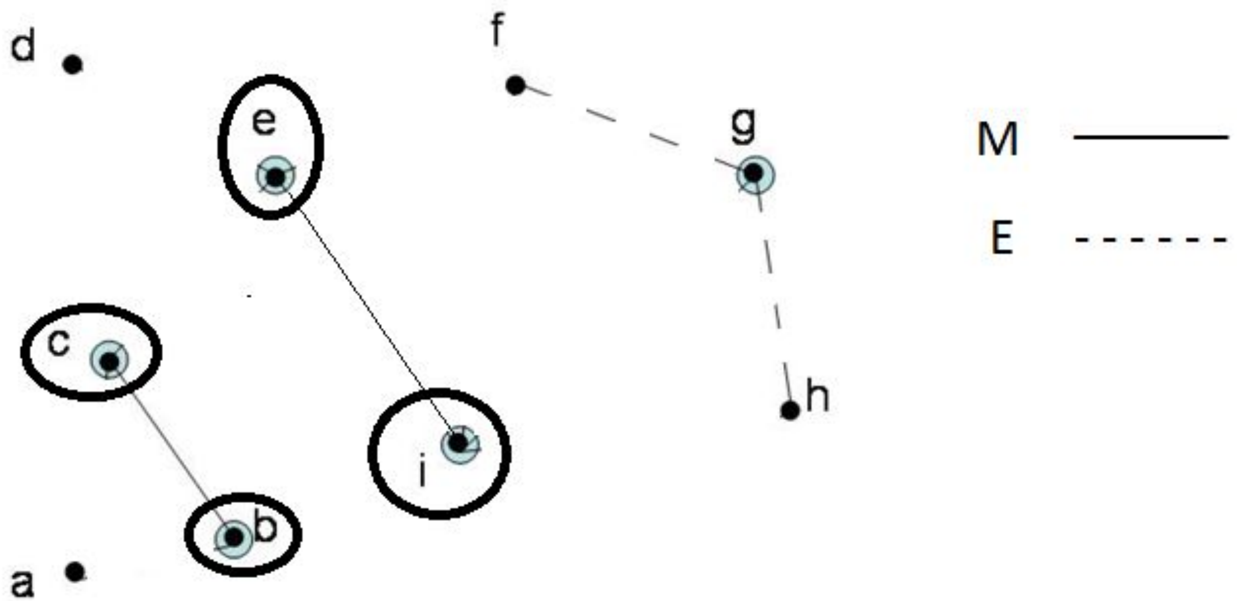


Example of execution - Step 1



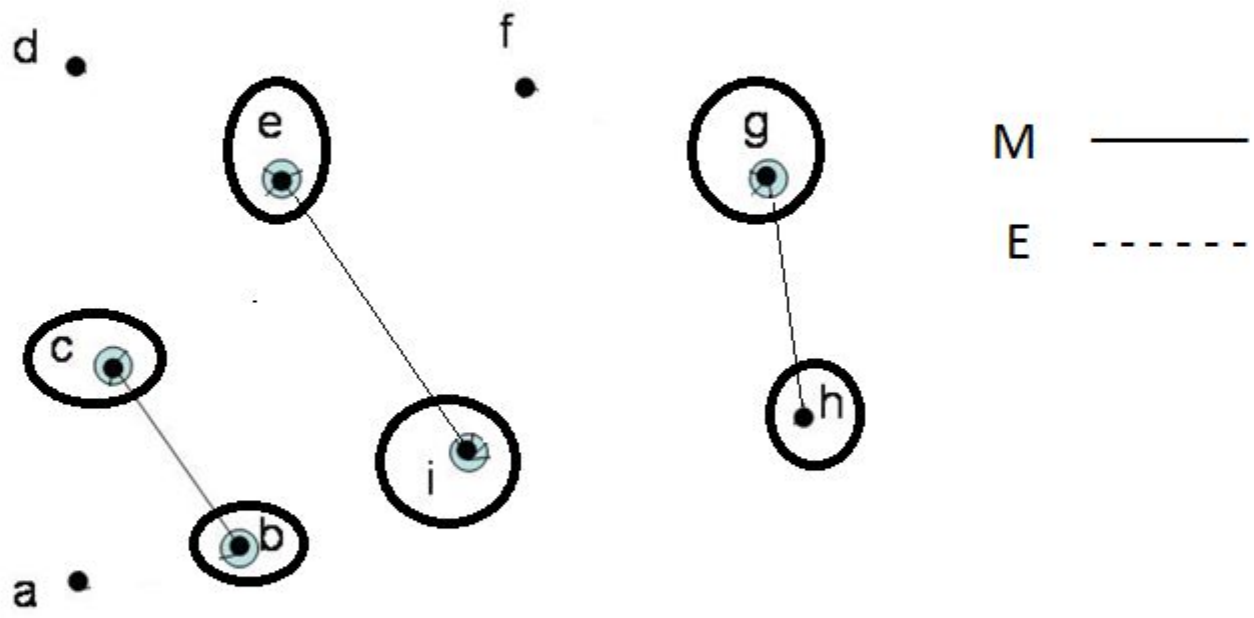
- U = {b, c}
- M = {{b,c}}

Example of execution - Step 2



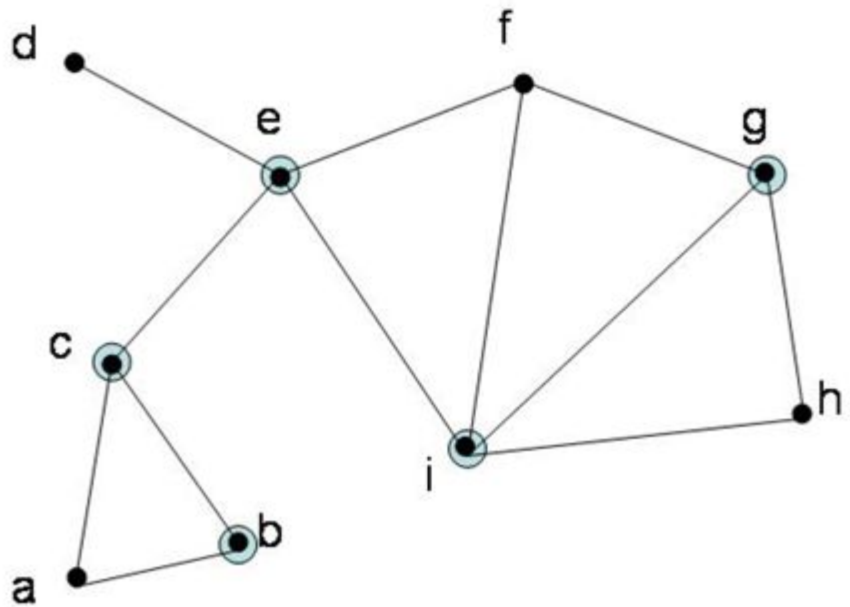
- U = {b, c, e, i}
- M = {{b,c}, {e, i}}

Example of execution - Step 3



- U = {b, c, e, i, g, h}
- M = {{b,c}, {e, i}, {g, h}}

Optimal solution



- $U^* = \{b, c, e, i, g\}$

Lemma: At the end of the execution of Approx-Cover *M* forms a matching, that is the edges in *M* do not share any endpoint.

Proof: Trivially, every time an edge *e* is selected in *M*, all the edges with an endpoint in common with *e* are deleted from *E.*

Therefore, in the following steps no edge with an endpoint in common with *e* can be chosen by the algorithm.

□

Theorem: Approx-Cover is 2-approximating.

Proof: The value of the solution returned by the algorithm is

$$m = |U| = 2 \cdot |M|.$$

Let U^* be an optimal cover. Since the edges in M do not share any endpoint (M is a matching) and each of them must have an endpoint in U^*,

$$m^* = |U^*| \geq |M|.$$

Therefore

$$\frac{m}{m^*} \leq \frac{2|M|}{|M|} = 2.$$

$\square$