The k-center problem

- Input is set of cities with intercity distances ($G = (V, V \times V)$)
- Select k cities to place warehouses
- Goal: minimize maximum distance of a city to a warehouse

Other application: placement of ATMs in a city

Results

- NP-hardness
- Greedy algorithm, approximation ratio 2
- Technique: parametric pruning
- Second algorithm with approximation ratio 2
- Generalization of Algorithm 2 to weighted problem

Theorem 1. *It is* **NP***-hard to approximate the general k-center problem within any factor* α.

Proof. Reduction from Dominating Set ... □

Dominating set = subset S of vertices such that every vertex which is not in S is adjacent to a vertex in S.

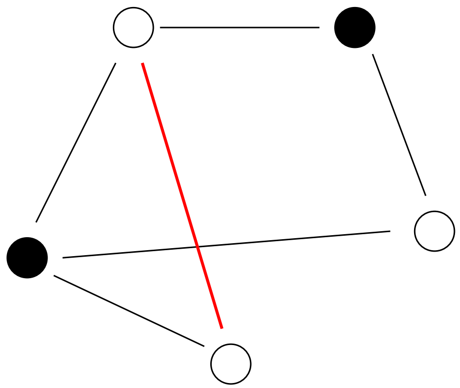
Finding a dominant set of minimal size is NP-hard

For a graph G, $\mathrm{dom}(G)$ is the size of the smallest possible dominating set

Dominating set is similar to but not the same as vertex cover!

Dominating set and vertex cover

Vertex cover = subset S of vertices such that every edge has at least one endpoint in S



The black vertices form a dominating set but not a vertex cover.

Also, not every vertex cover is a dominating set.

Proof We want to find a Dominating Set in $G = (V, E)$.

Consider $G' = (V, V \times V)$ and the weight function

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2\alpha & else \end{cases}$$

Proof We want to find a Dominating Set in $G = (V, E)$.

Consider $G' = (V, V \times V)$ and the weight function

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2\alpha & else \end{cases}$$

Suppose G has a dominating set of size at most k.

Then there is a k-center of cost 1 in G'

$\rightarrow$ an α-approx. algorithm delivers one with weight $\leq \alpha$

Proof We want to find a Dominating Set in $G = (V, E)$.
Consider $G' = (V, V \times V)$ and the weight function

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2\alpha & else \end{cases}$$

Suppose G has a dominating set of size at most k.
Then there is a k-center of cost 1 in G'
$\rightarrow$ an α-approx. algorithm delivers one with weight $\leq \alpha$

If there is no such dominating set in G, every k-center has weight $\geq 2\alpha > \alpha$.

Proof (continued)

Assume that there exists an α-approximation algorithm for the k-center problem.

Decision algorithm: Run α-approx algorithm on G'

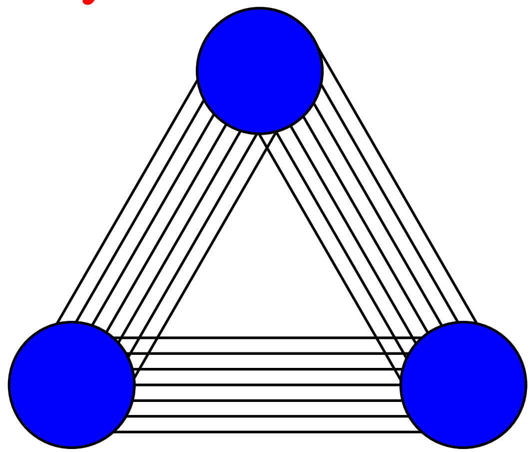
Solution has weight $\leq \alpha \rightarrow$ dominating set of size at most k exists

Else there is no such dominating set. $\square$

Metric k-center

G is undirected and obeys the triangle inequality

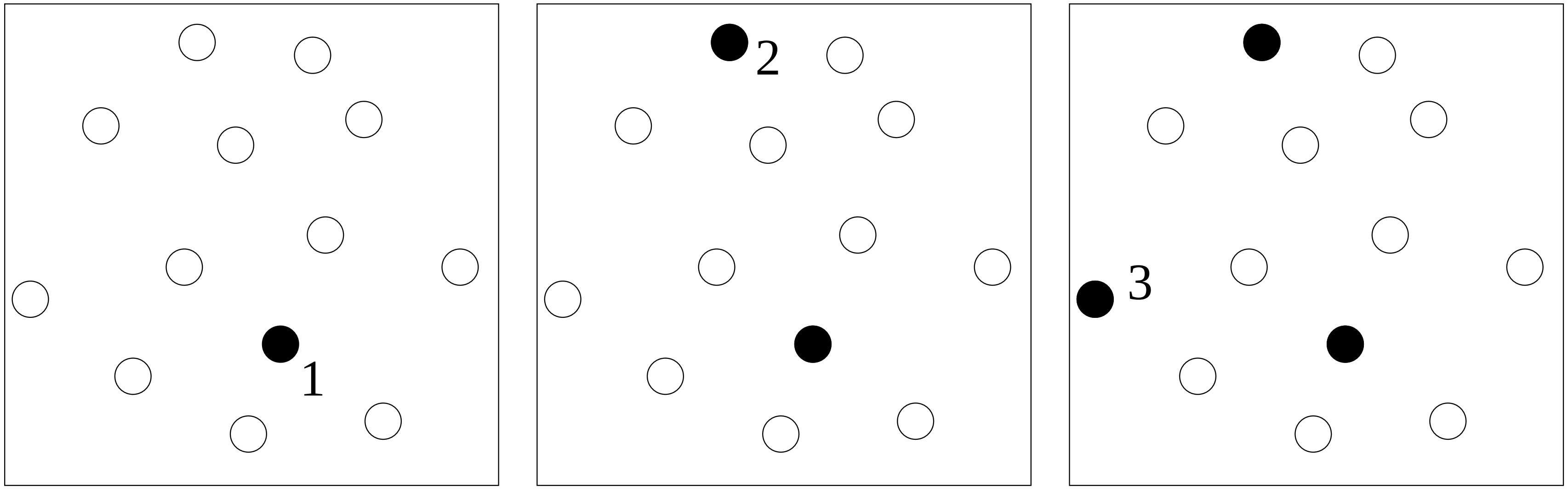
$\forall u, v, w \in V : d(u,w) \leq d(u,v) + d(v,w)$



We show two 2-approximation algorithms for this problem.

The Greedy algorithm

- Choose the first center arbitrarily
- At every step, choose the vertex that is furthest from the current centers to become a center
- Continue until k centers are chosen



Analysis

- Note that the sequence of distances from a new chosen center, to the closest center to it (among previously chosen centers) is non-increasing
- Consider the point that is furthest from the k chosen centers
- We need to show that the distance from this point to the closest center is at most $2 \cdot \mathrm{OPT}$
- Assume by negation that it is $> 2 \cdot \mathrm{OPT}$

Analysis

- We assumed that the distance from the furthest point to all centers is $> 2 \cdot \mathrm{OPT}$
- This means that distances between all centers are also $> 2 \cdot \mathrm{OPT}$
- We have $k+1$ points with distances $> 2 \cdot \mathrm{OPT}$ between every pair

Analysis

- Each point has a center of the optimal solution with distance $\leq$ OPT to it
- There exists a pair of points with the same center X in the optimal solution (pigeonhole principle: k optimal centers, $k+1$ points)
- The distance between them is at most $2 \cdot$ OPT (triangle inequality)
- Contradiction!

Technique: parametric pruning

Idea: remove irrelevant parts of the input

- Suppose OPT $= t$
- We want to show a 2-approximation
- Any edges of cost more than $2t$ are useless: if two vertices are connected by such an edge, and one of them gets a warehouse, the other one is still too far away
- We can remove edges that are too expensive

Of course, we don't know OPT. But we can guess.

Technique: parametric pruning

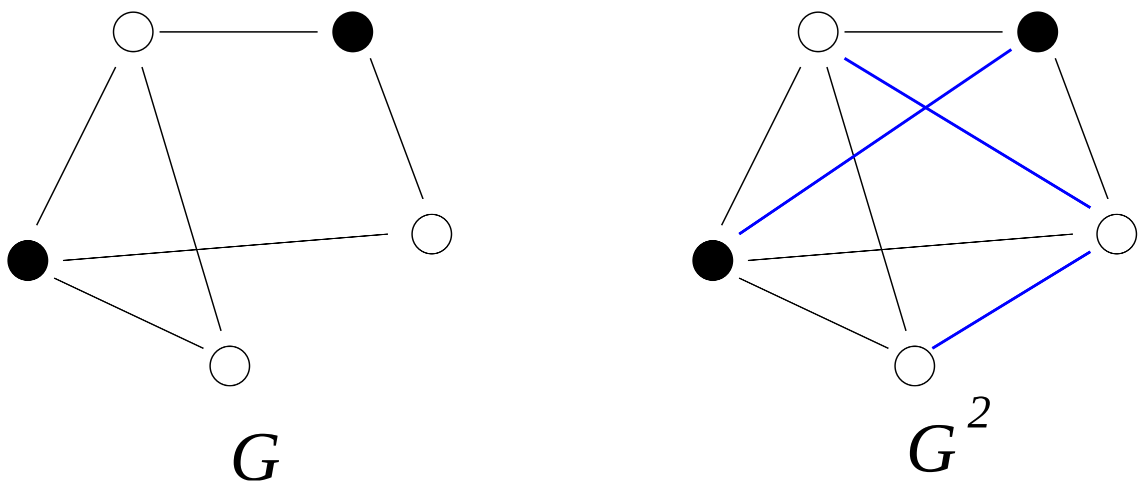
- We can order the edges by cost: $\text{cost}(e_1) \leq ... \leq \text{cost}(e_m)$
- Let $G_i = (V, E_i)$ where $E_i = \{e_1, \ldots, e_i\}$
- The k-center problem is equivalent to finding the minimal i such that

  $$G_i \text{ has a dominating set of size } k$$

- Let i^* be this minimal i
- Then, $\text{OPT} = \text{cost}(e_{i^*})$

Graph squaring

For a graph G, the square $G^2 = (V, E')$ where $(u, v) \in E'$ if there is a path of length at most 2 between u and v in G (and $u \neq v$)



Lemma 2. *For any independent set I in G^2, we have $|I| \leq dom(G)$.*

Proof. Let D be a minimum dominating set in G.

(The size of D is $\mathrm{dom}(G)$.)

Lemma 2. *For any independent set I in G^2, we have* $|I| \leq dom(G)$.

Proof. Let D be a minimum dominating set in G.
Then G contains $|D|$ stars spanning all vertices (the nodes of D are the centers of the stars).

Lemma 2. *For any independent set I in G^2, we have* $|I| \le dom(G)$.

Proof. Let D be a minimum dominating set in G.
Then G contains $|D|$ stars spanning all vertices (the nodes of D are the centers of the stars).
A star in G becomes a clique in G^2.

Lemma 2. *For any independent set I in G^2, we have* $|I| \leq dom(G)$.

Proof. Let D be a minimum dominating set in G.
Then G contains $|D|$ stars spanning all vertices (the nodes of D are the centers of the stars).
A star in G becomes a clique in G^2.
So G^2 contains $|D| = \mathrm{dom}(G)$ cliques spanning all vertices.

Lemma 2. *For any independent set I in G^2, we have* $|I| \leq dom(G)$.

Proof. Let D be a minimum dominating set in G.
Then G contains $|D|$ stars spanning all vertices (the nodes of D are the centers of the stars).
A star in G becomes a clique in G^2.
So G^2 contains $|D| = \text{dom}(G)$ cliques spanning all vertices.
There can only be one vertex of each clique in I. □

Algorithm

We use that maximal independent sets can be found in polynomial time.

- Construct $G_1^2, G_2^2, \ldots, G_m^2$
- Find a maximal independent set M_i in each graph G_i^2
- Determine the smallest i such that $|M_i| \leq k$, call it j
- Return M_j.

Lemma 3. *For this j, $cost(e_j) \leq OPT$.*

Lemma 4. *This algorithm gives a 2-approximation.*

Lemma 3. *For this j, $cost(e_j) \leq OPT$.*

Proof. For every $i < j$...

- $|M_i| > k$ by the definition of our algorithm
- $\mathrm{dom}(G_i) > k$ by Lemma 2
- Then $i^* > i$

Therefore, $i^* \geq j$. □

Lemma 4. *This algorithm gives a 2-approximation.*

Proof.

- Any maximal independent set I in G_j^2 is also a dominating set (if some vertex v were not dominated, $I \cup v$ were also independent)

Lemma 4. *This algorithm gives a 2-approximation.*

Proof.

- Any maximal independent set I in G_j^2 is also a dominating set (if some vertex v were not dominated, $I \cup v$ were also independent)
- In G_j^2, we have $|M_j|$ stars centered on the vertices in M_j

Lemma 4. *This algorithm gives a 2-approximation.*

Proof.

- Any maximal independent set I in G_j^2 is also a dominating set (if some vertex v were not dominated, $I \cup v$ were also independent)
- In G_j^2, we have $|M_j|$ stars centered on the vertices in M_j
- These stars cover all the vertices

Lemma 4. *This algorithm gives a 2-approximation.*

Proof.

- Any maximal independent set I in G_j^2 is also a dominating set (if some vertex v were not dominated, $I \cup v$ were also independent)
- In G_j^2, we have $|M_j|$ stars centered on the vertices in M_j
- These stars cover all the vertices
- Each edge used in constructing these stars has cost at most $2 \cdot \mathrm{cost}(e_j) \leq 2 \cdot \mathrm{OPT}$

The last inequality follows from Lemma 3. □

Lemma 5. *If $P \neq NP$, no approximation algorithm gives a $(2-\varepsilon)$-approximation for any* $\varepsilon > 0$.

- We again use a reduction from Dominating Set
- This time, the graph must satisfy the triangle inequality
- We define G' as follows:

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2 & else \end{cases}$$

This graph satisfies the triangle inequality (proof?)

Suppose G has a dominating set of size at most k.

Then there is a k-center of cost 1 in G'

$\rightarrow$ a $(2-\varepsilon)$-approx. algorithm delivers one with weight < 2

If there is no such dominating set in G, every k-center has weight $\geq 2 > 2-\varepsilon$.

Thus, a $(2-\varepsilon)$-approximation algorithm for the k-center problem can be used to determine whether or not there is a dominating set of size k.

Weighted k-center problem

- Input is set of cities with intercity distances ($G = (V, V \times V)$)
- Each city has a cost
- Select cities of cost at most W to place warehouses
- Goal: minimize maximum distance of a city to a warehouse

Ideas

- We use the same graphs $G_1, \ldots, G_m$ as before
- Let $\mathrm{wdom}(G)$ be the weight of a minimum weight dominating set in G
- We look for the smallest index i such that $\mathrm{wdom}(G_i) \geq W$
- We also use graph squaring again

The set of light neighbors

- Let I be an independent set in G^2
- For any node u, let $s(u)$ be the lightest neighbor of u
- Here, we also consider u to be a neighbor of itself
- Let $S = \{s(u)|u \in I\}$
- We claim $w(S) \leq wdom(G)$

(Compare the unweighted problem, where we had $|I| \leq \mathrm{dom}(G)$)

Lemma 6. $w(S) \leq wdom(G)$

Proof. Let D be a minimum weight dominating set in G.
Then G contains $|D|$ stars spanning all vertices (the nodes of D are the centers of the stars).
A star in G becomes a clique in G^2.
So G^2 contains $|D|$ cliques spanning all vertices.
There can only be one vertex of each clique in I.
For each vertex in I, the center of the corresponding star is available as a neighbor in G (this might not be the lightest neighbor). Therefore $w(S) \leq \mathrm{wdom}(G)$. □

Algorithm

Let $s_i(u)$ denote a lightest neighbor of u in G_i.

- Construct $G_1^2, \ldots, G_m^2$
- Compute a maximal independent set M_i in each graph G_i^2
- Compute $S_i = \{s_i(u) | u \in M_i\}$
- Find the minimum index i such that $w(S_i) \leq W$, say j
- Return S_j

Lemma 7. *This algorithm achieves a 3-approximation.*

- As before we have $\text{OPT} \geq \text{cost}(e_j)$

For every $i < j$...

- $w(S_i) > W$ by the definition of our algorithm
- $\text{wdom}(G_i) > W$ by Lemma 6
- Then $i^* > i$

Therefore, $i^* \geq j$.

Lemma 7. *This algorithm achieves a 3-approximation.*

- As before we have $\text{OPT} \geq \text{cost}(e_j)$
- M_j is a dominating set in G_j^2

It is a maximal independent set

Lemma 7. *This algorithm achieves a 3-approximation.*

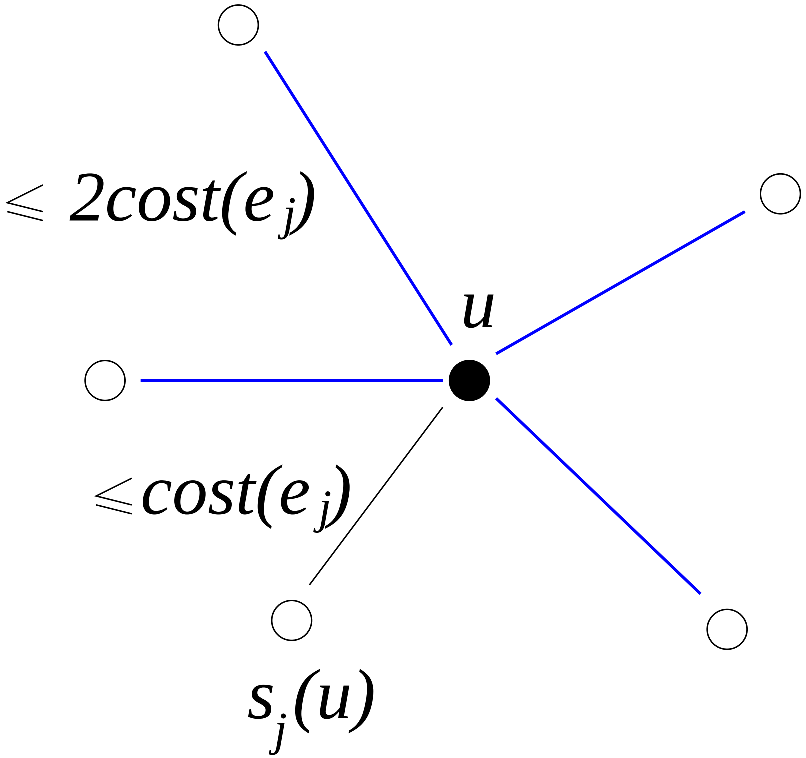
- As before we have $\mathrm{OPT} \geq \mathrm{cost}(e_j)$
- M_j is a dominating set in G_j^2
- We can cover V with stars of G_j^2 centered in vertices of M_j

Lemma 7. *This algorithm achieves a 3-approximation.*

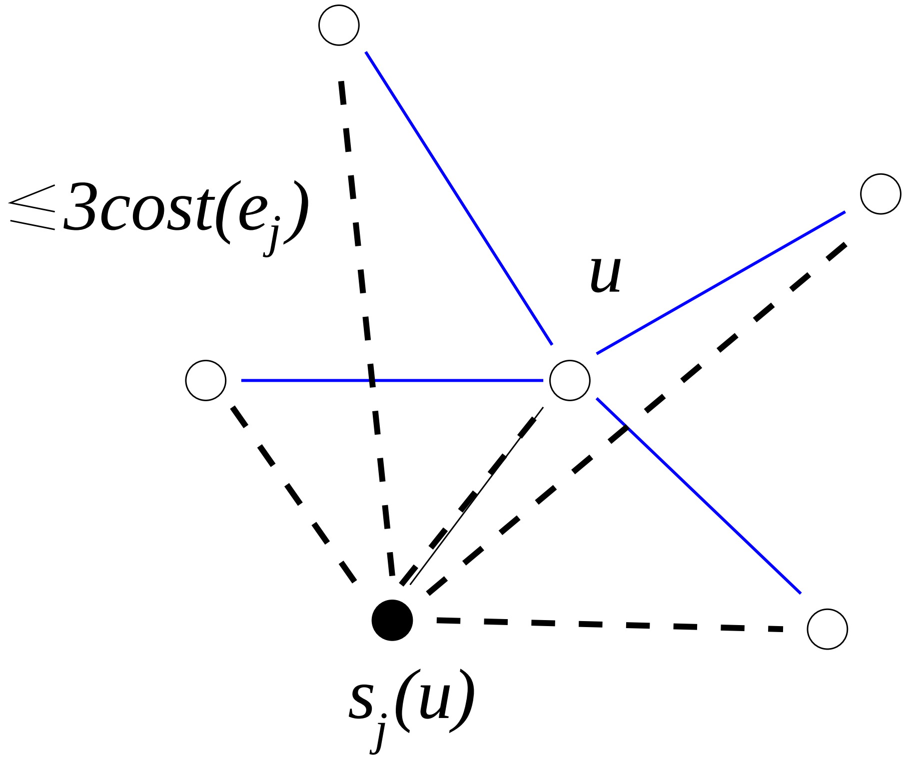
- As before we have $\text{OPT} \geq \text{cost}(e_j)$
- M_j is a dominating set in G_j^2
- We can cover V with stars of G_j^2 centered in vertices of M_j
- These stars as before use edges of cost at most $2 \cdot \text{cost}(e_j)$ (triangle inequality)

Lemma 7. *This algorithm achieves a 3-approximation.*

- As before we have $\mathrm{OPT} \geq \mathrm{cost}(e_j)$
- M_j is a dominating set in G_j^2
- We can cover V with stars of G_j^2 centered in vertices of M_j
- These stars as before use edges of cost at most $2 \cdot \mathrm{cost}(e_j)$ (triangle inequality)
- Each star center is adjacent to a vertex in S_j, using an edge of cost at most $\mathrm{cost}(e_j)$



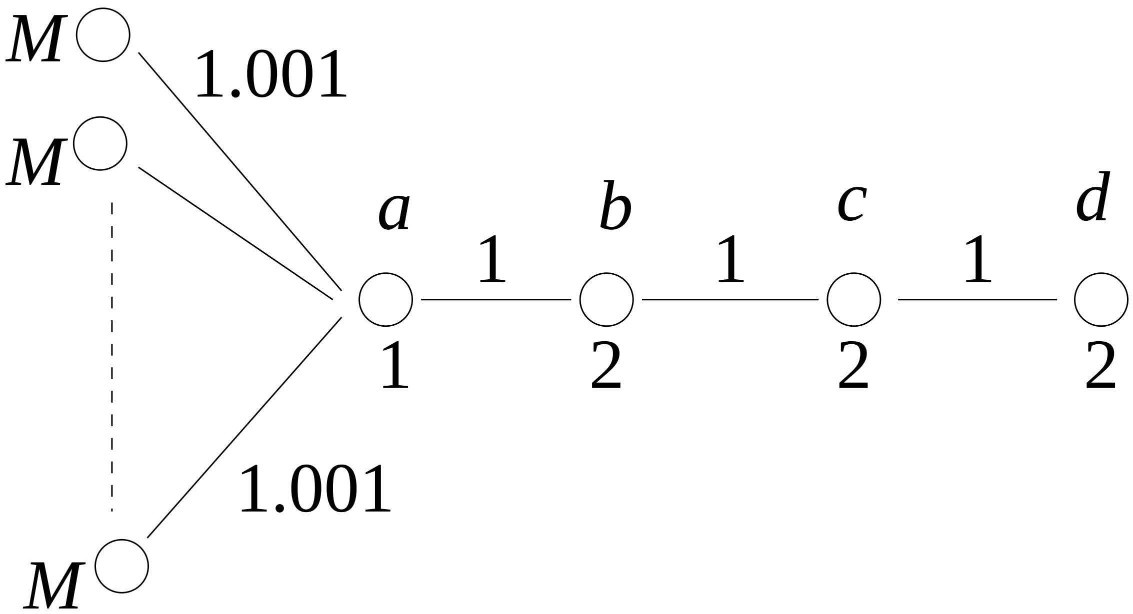
A star in G_j^2



A star in G_j^2 with redefined centers

Thus every node in G_j can be reached at cost at most $3 \cdot \mathrm{cost}(e_j)$ from some vertex in S. This completes the proof.

Lower bound for this algorithm



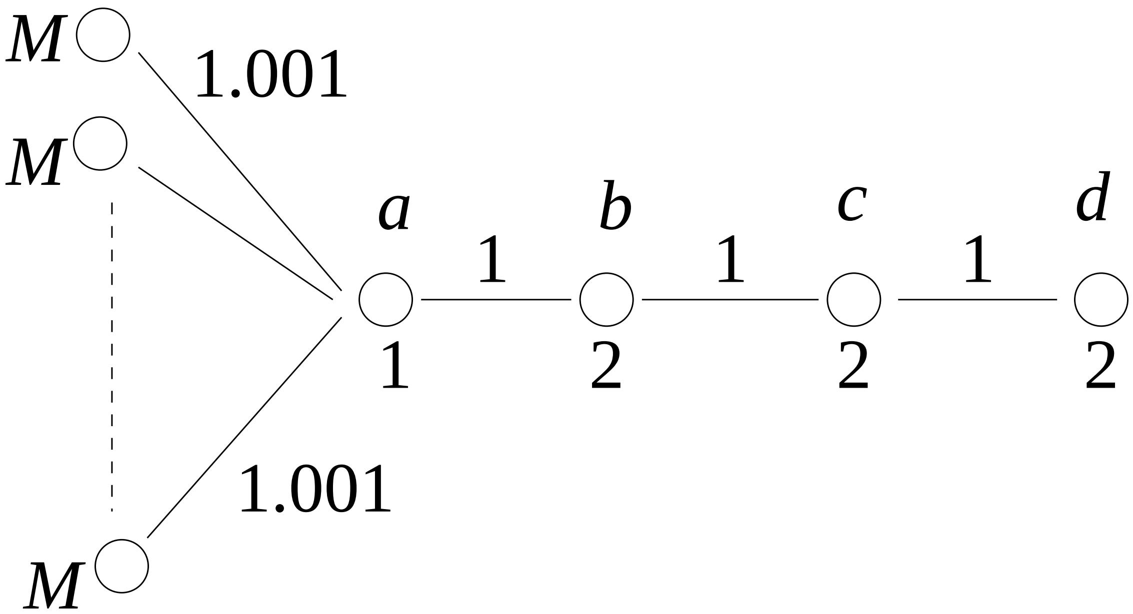
There are n nodes of weight M. The bound $W = 3$.

All edges not shown have weight equal to the length of the shortest path in the graph that is shown

For $i < n+3$, G_i is missing at least one edge of weight 1.001.

One vertex will be isolated (also in G_i^2) so it will be in S_i

Lower bound for this algorithm



There are n nodes of weight M. The bound $W = 3$.

All edges not shown have weight equal to the length of the shortest path in the graph that is shown

For $i = n+3$, $\{b\}$ is a maximal independent subset

If our algorithm chooses $\{b\}$, it outputs $S_{n+3} = \{a\}$. Cost is 3.