Advanced Algorithmic Techniques (COMP523)

Approximation Algorithms 3

Recap and plan

- **Previous lecture:**
  - The Pricing Method (Primal-Dual Method).
    - Application: Vertex Cover.
- This lecture:
  - Linear Programming and Rounding.
    - Application: Vertex Cover.
  - Inapproximability of Vertex Cover.
  - Vertex Cover on Bipartite Graphs.

Methods for approximation algorithms

- Greedy algorithms.
- Pricing method (also known as the Primal-Dual method).
- Linear Programming and Rounding.
- Dynamic Programming on rounded inputs.

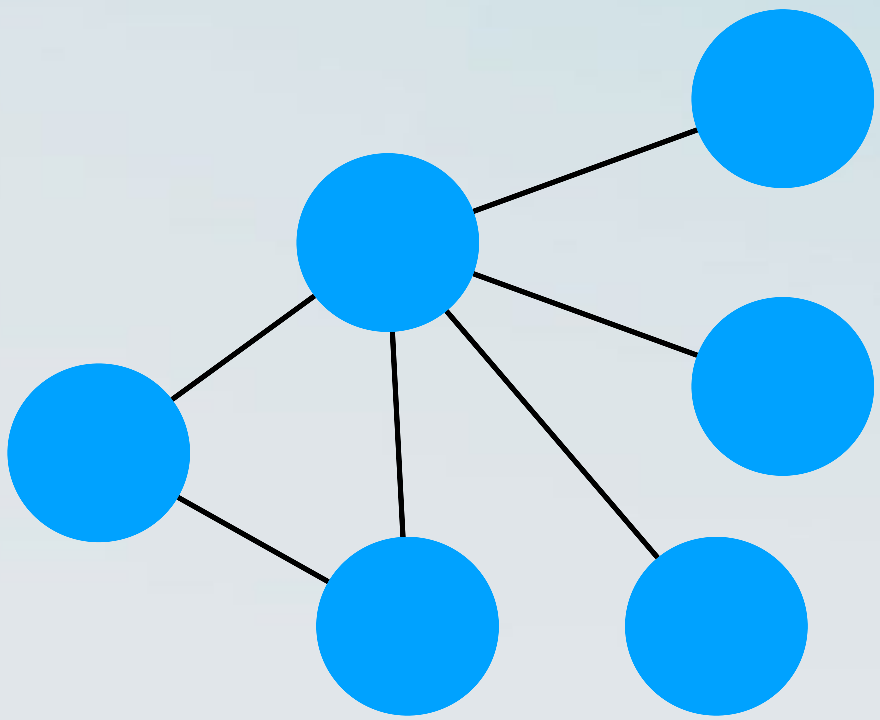
Vertex Cover

- Definition: A vertex cover C of a graph G=(V, E) is a subset of the nodes such that every edge e in the graph has at least one endpoint in C.

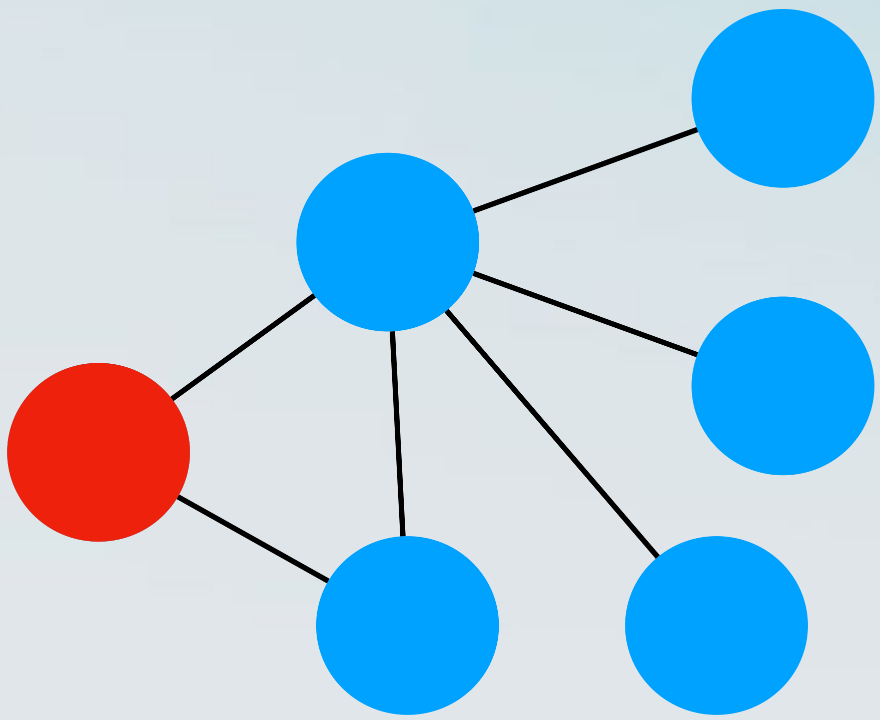
- Definition: A minimum vertex cover is a vertex cover of the smallest possible size.

- Vertex Cover
  Input: A graph G=(V, E)
  Output: A minimum vertex cover.

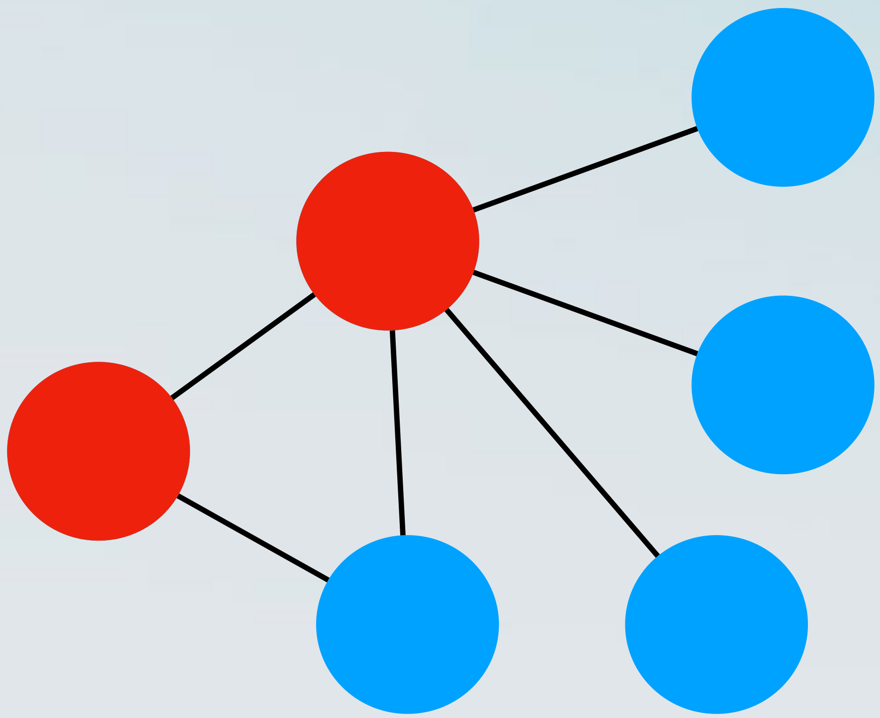
Example



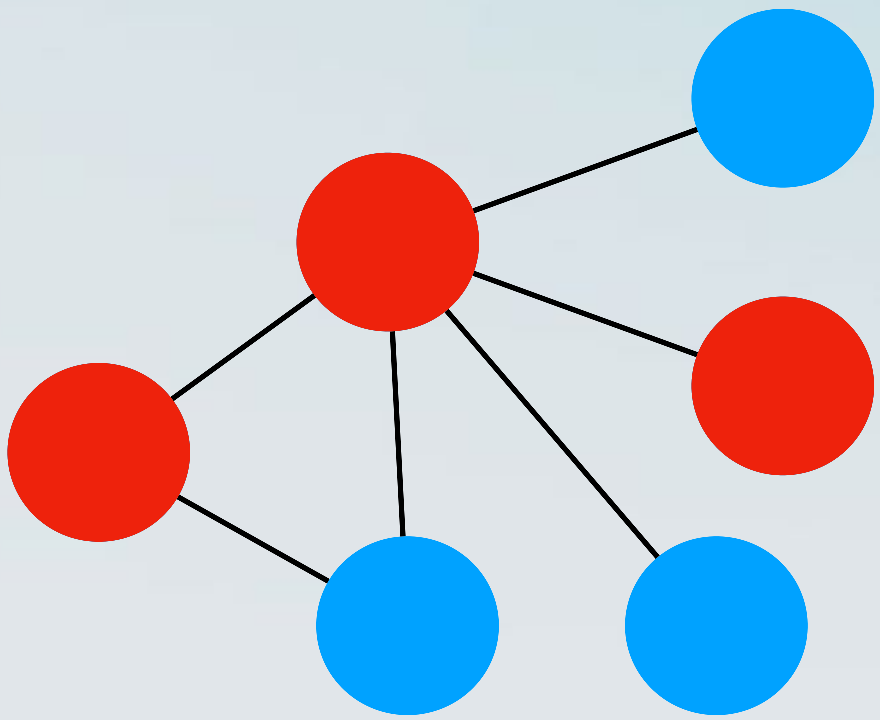
Example



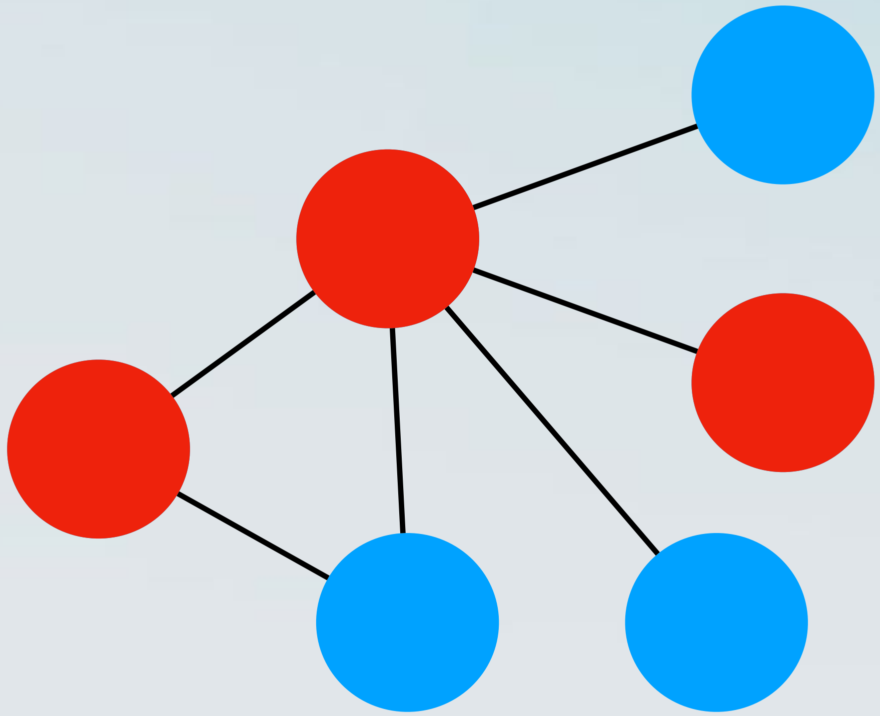
Example



Example

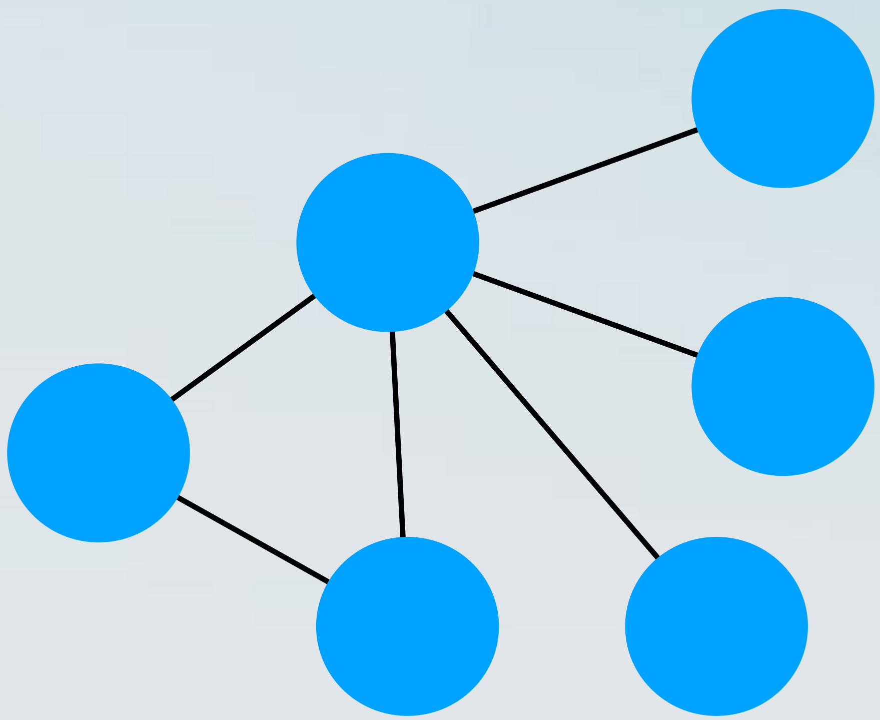


Example

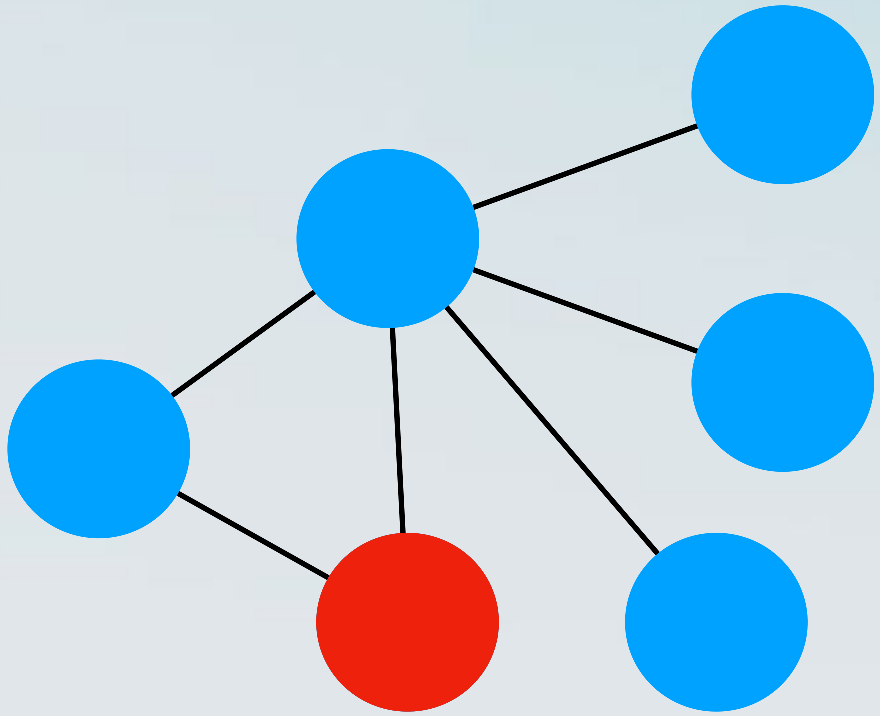


A vertex cover

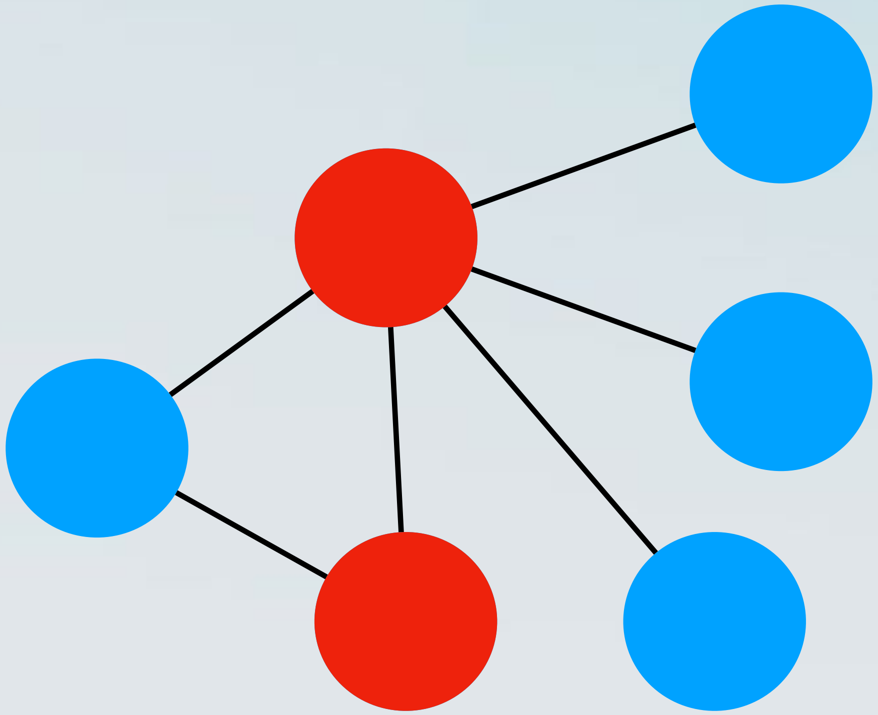
Example



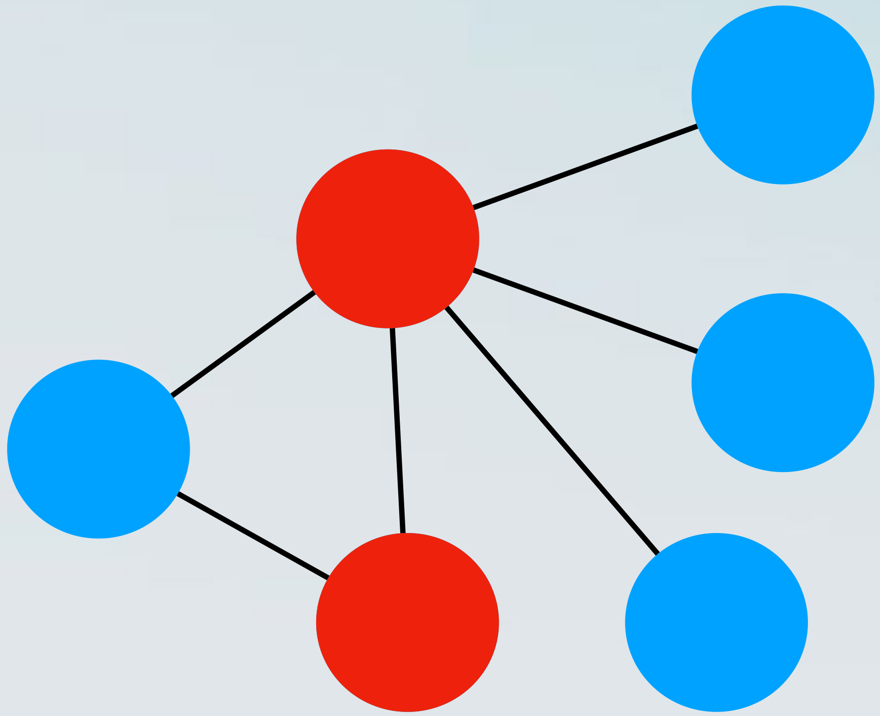
Example



Example



Example



A minimum vertex cover

Vertex Cover

- Definition: A vertex cover C of a graph $G=(V, E)$ is a subset of the nodes such that every edge e in the graph has at least one endpoint in C.

- Each vertex i has a weight w_i.

- Definition: A minimum vertex cover is a vertex cover of the smallest possible total weight.

Vertex Cover

- Definition: A vertex cover C of a graph G=(V, E) is a subset of the nodes such that every edge e in the graph has at least one endpoint in C.

- Each vertex i has a weight w_i.

- Definition: A minimum vertex cover is a vertex cover of the smallest possible total weight.

- Vertex Cover
  Input: A graph G=(V, E)
  Output: A minimum (weight) vertex cover.

Vertex Cover

- Last time, we designed a polynomial time approximation algorithm for the weighted vertex cover problem.
- We will design another polynomial time approximation algorithm for the weighted vertex cover problem.
    - The second algorithm will be based on a technique called “ILP relaxation and rounding”.

Vertex Cover as an ILP

Minimise $\sum_{i \in V} x_i w_i$

subject to

$x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

$x_i \in \{0, 1\},$ **for all** $i \in V$

Vertex Cover as an ILP

Minimise $\sum_{i \in V} x_i w_i$

For each edge, one of the endpoints has to be in the vertex cover.

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

$x_i \in \{0, 1\},$ **for all** $i \in V$

A vertex is either included in the vertex cover (value 1) or not (value 0).

Vertex Cover LP-relaxation

Minimise $\sum_{i \in V} x_i w_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

Vertex Cover LP-relaxation

Minimise $\sum_{i \in V} x_i w_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

Possible fractional values, e.g.,
$x_i = 0.3$, $x_j = 0.7$

Solving the LP-relaxation

Solving the LP-relaxation

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.

Solving the LP-relaxation

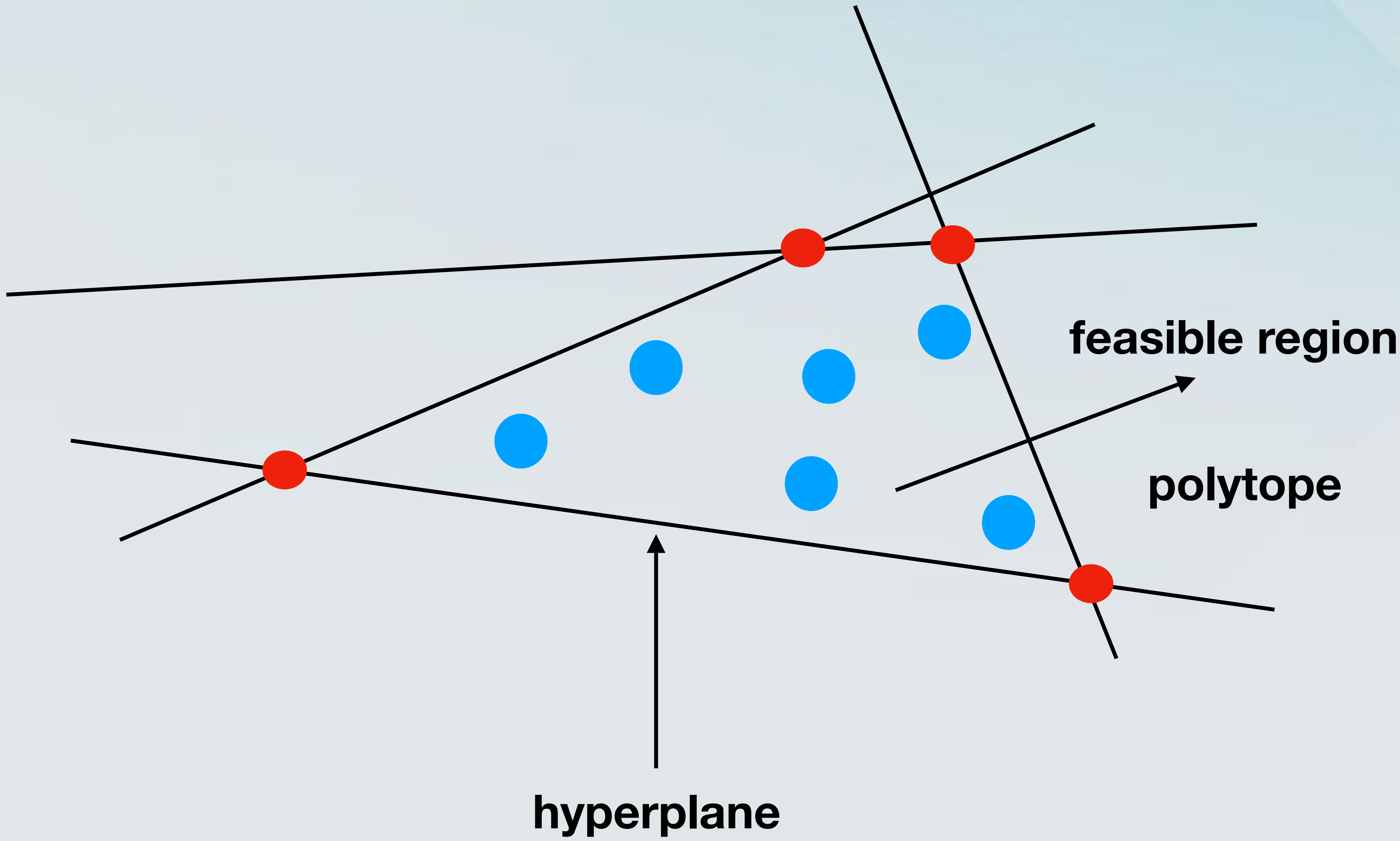
- We can solve the LP-relaxation in polynomial time, to find an optimal solution.

- The optimal solution is a “fractional” vertex cover, where variables can take values between *0* and *1*.

Solving the LP-relaxation

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a “fractional” vertex cover, where variables can take values between *0* and *1*.
- Is the value of this “fractional” vertex cover, smaller or larger than the value of the minimum weight vertex cover?

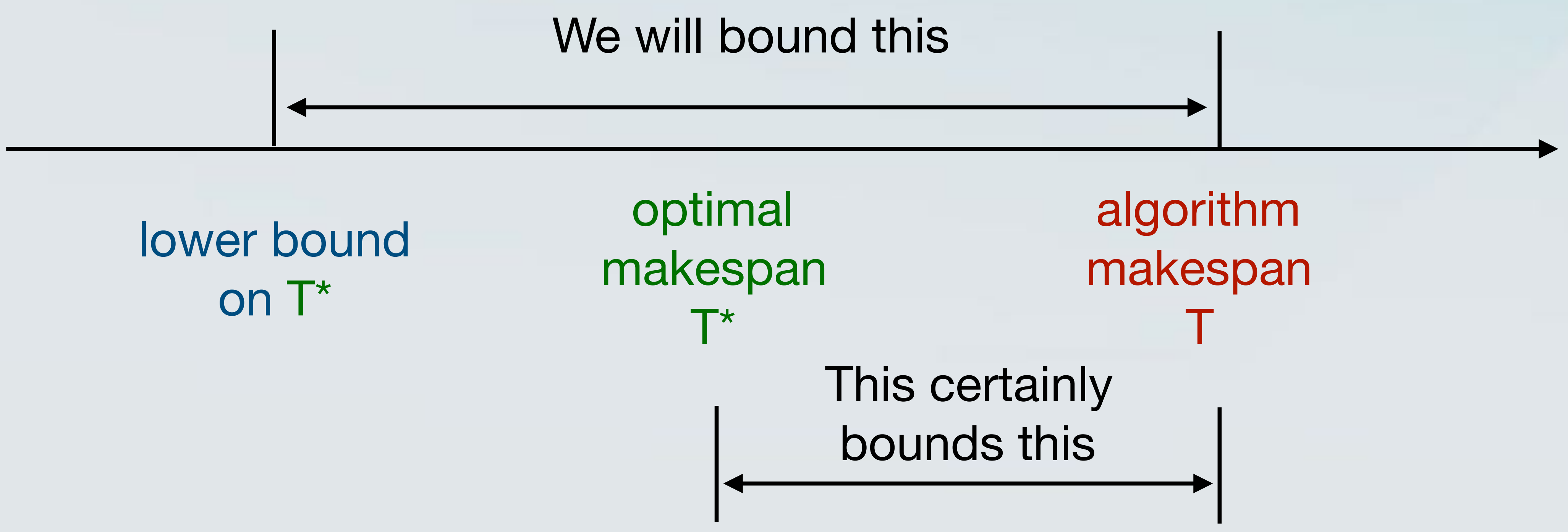
ILP vs LP-relaxation



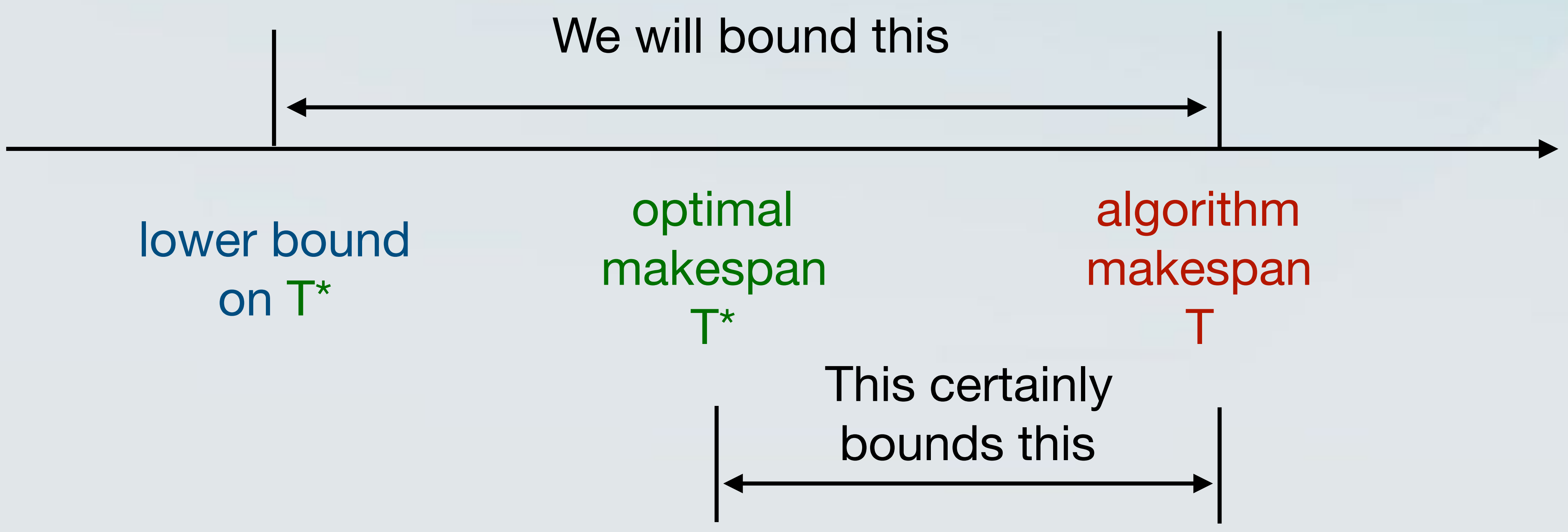
candidate optimal solution to the relaxation

candidate optimal solution to the ILP

Recall: Load Balancing



Recall: Load Balancing



We do not know the optimal, so we will use a lower bound for the optimal.

Back to vertex cover

- What can we use as a lower bound for the weight of the minimum weight vertex cover?

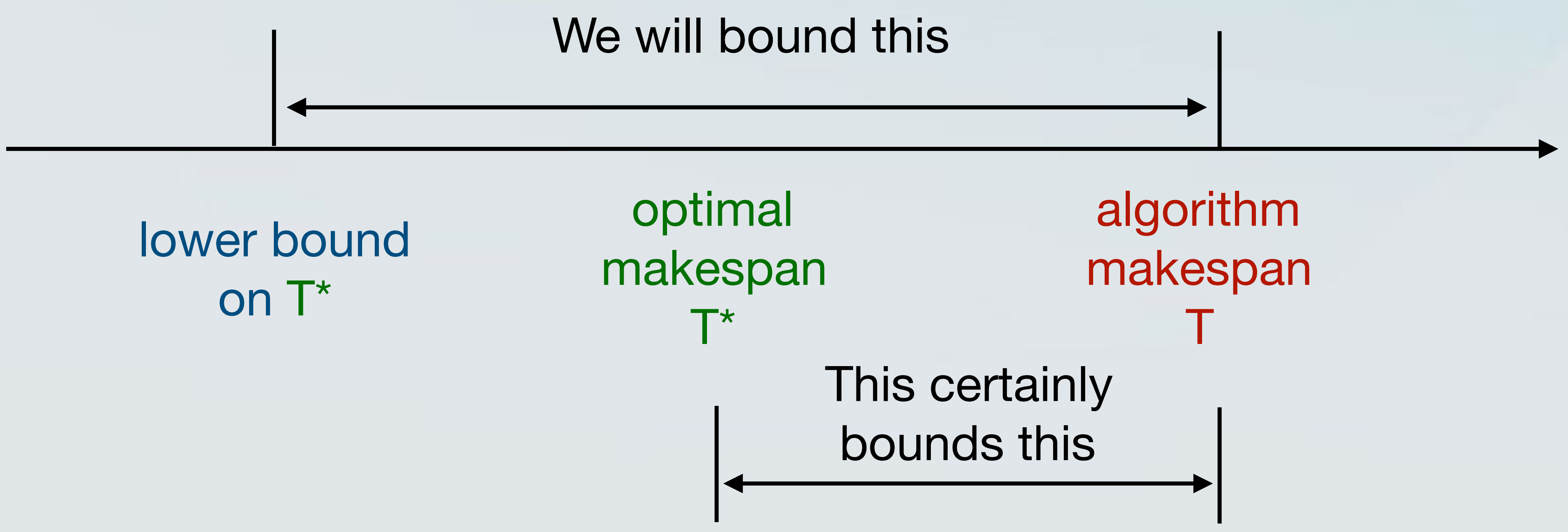
Back to vertex cover

- What can we use as a lower bound for the weight of the minimum weight vertex cover?
    - The weight of the minimum weight "fractional" vertex cover.

Back to vertex cover

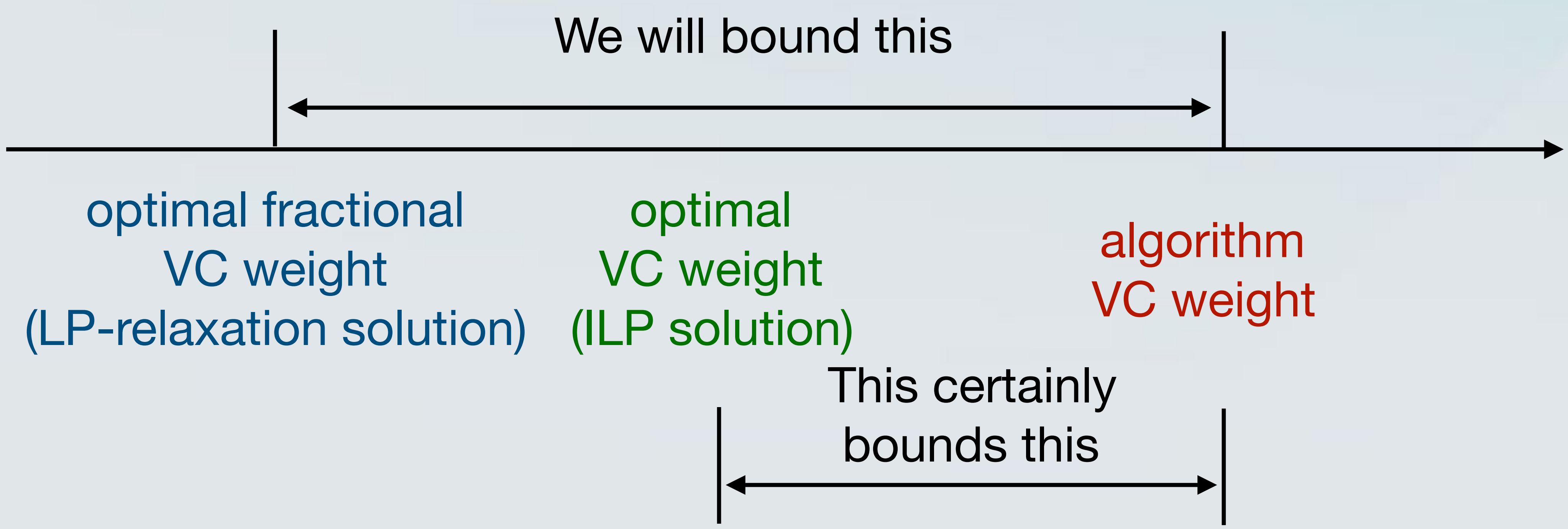
- What can we use as a lower bound for the weight of the minimum weight vertex cover?
  - The weight of the minimum weight "fractional" vertex cover.
  - i.e., the optimal value of the LP-relaxation of the vertex cover ILP.

Recall: Load Balancing



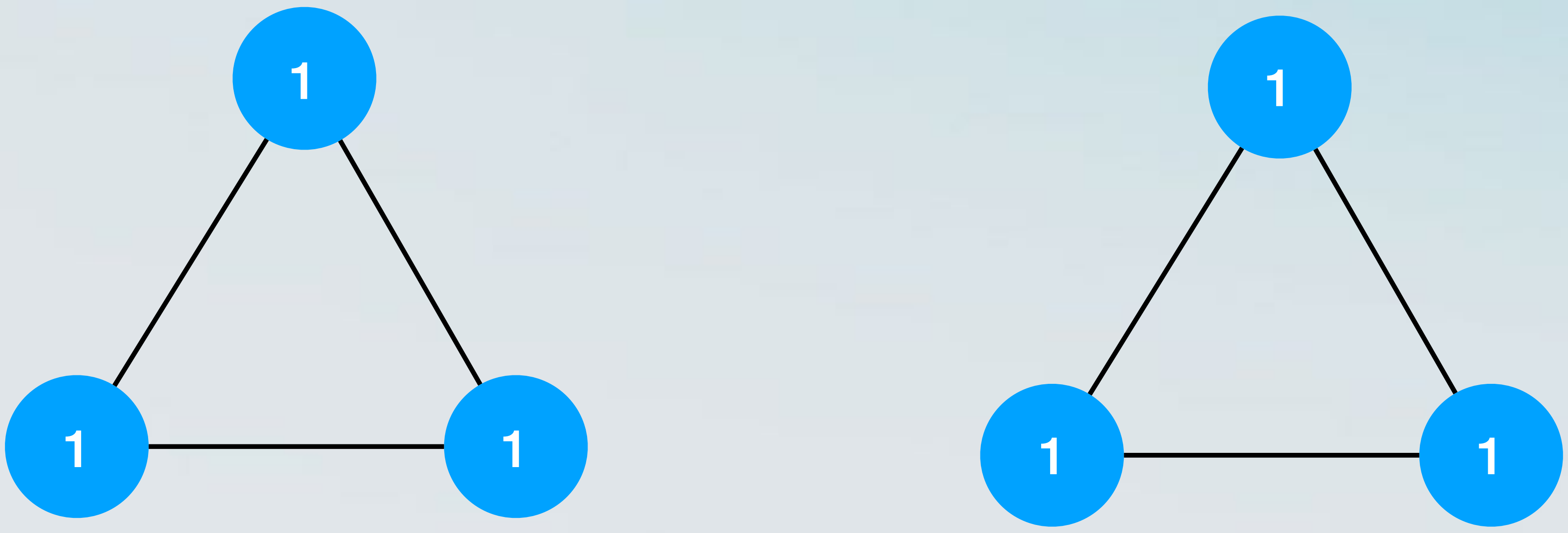
We do not know the optimal, so we will use a lower bound for the optimal.

Lower bounding the optimal

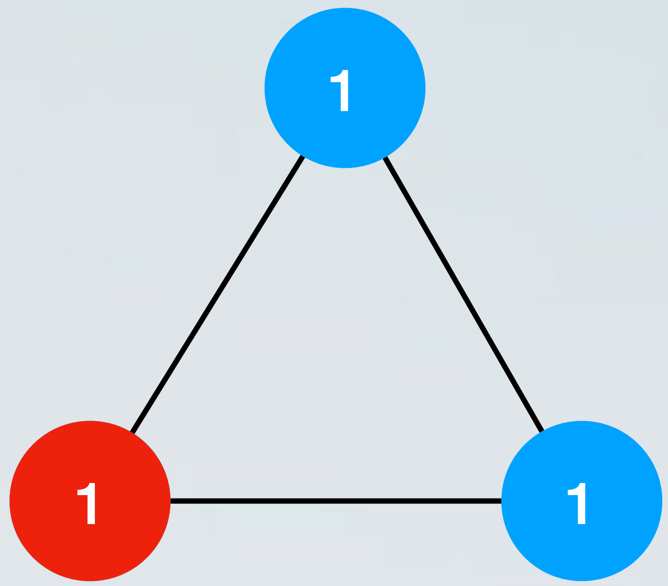


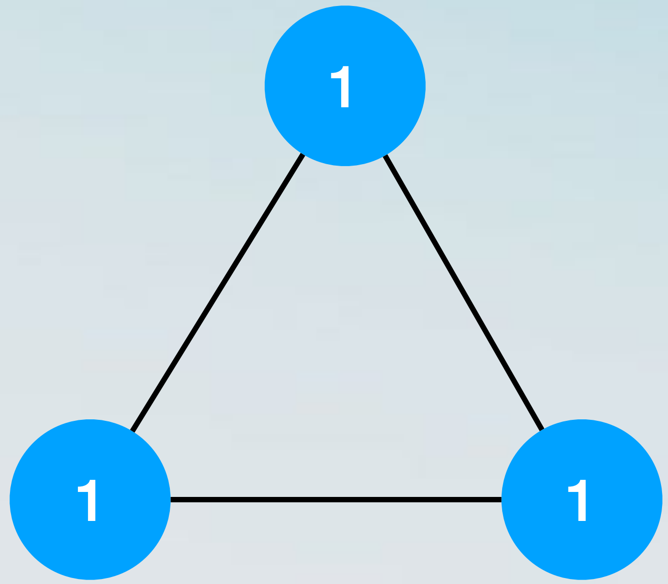
We do not know the optimal, so we will use
a lower bound for the optimal.

Example

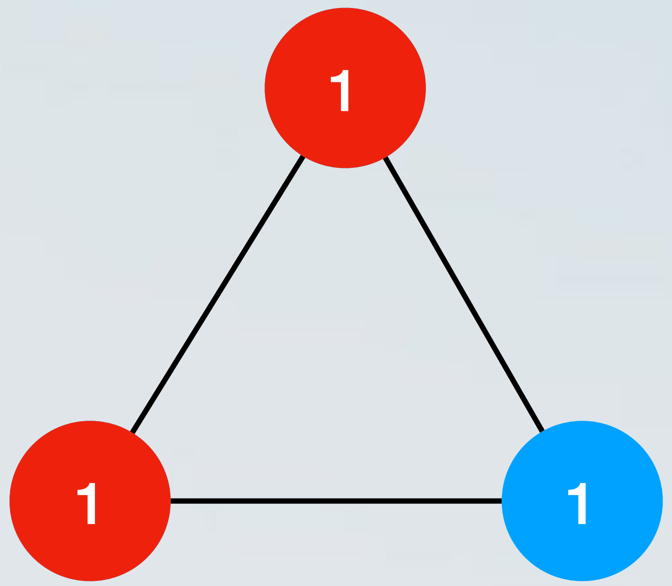


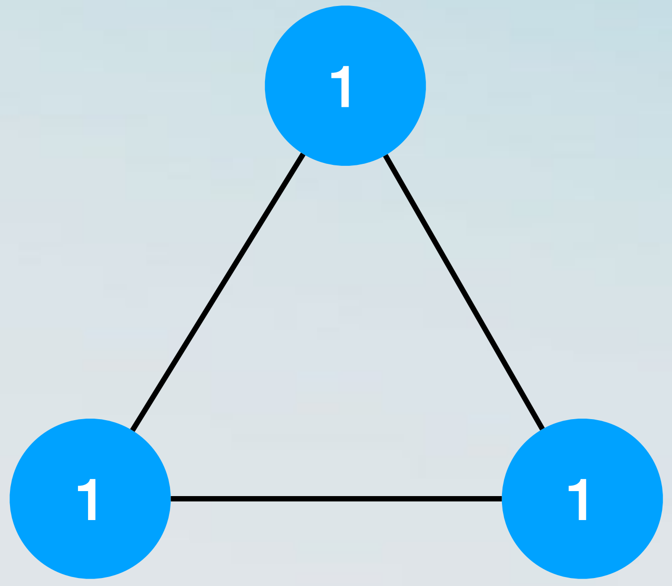
Example



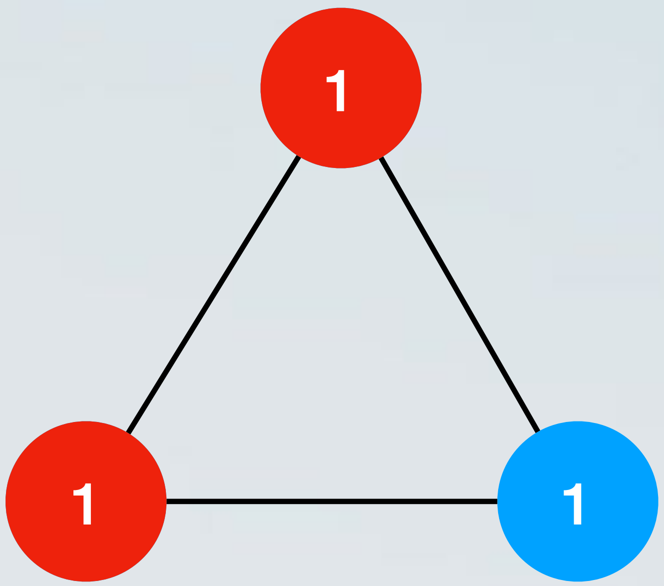


Example

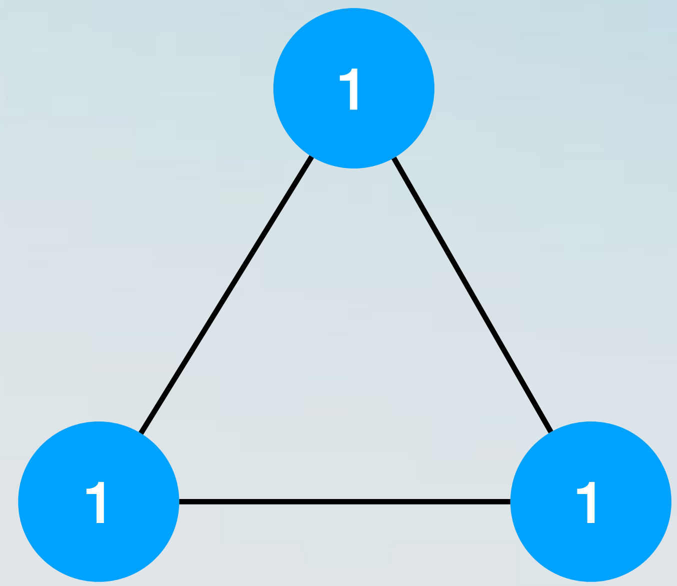




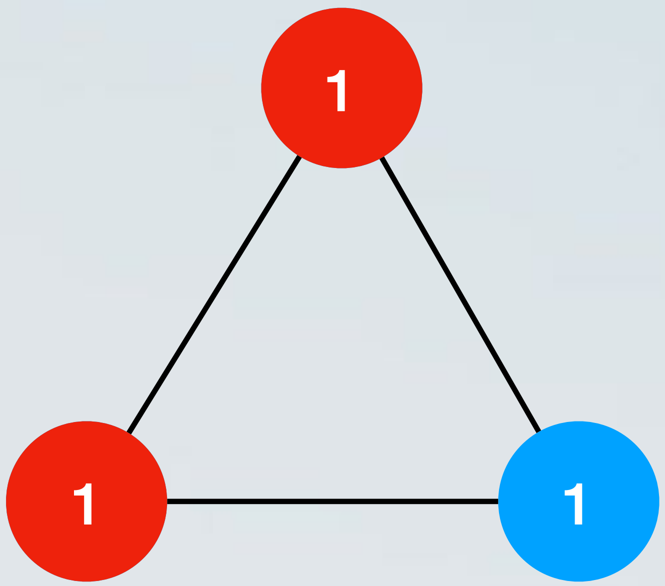
Example



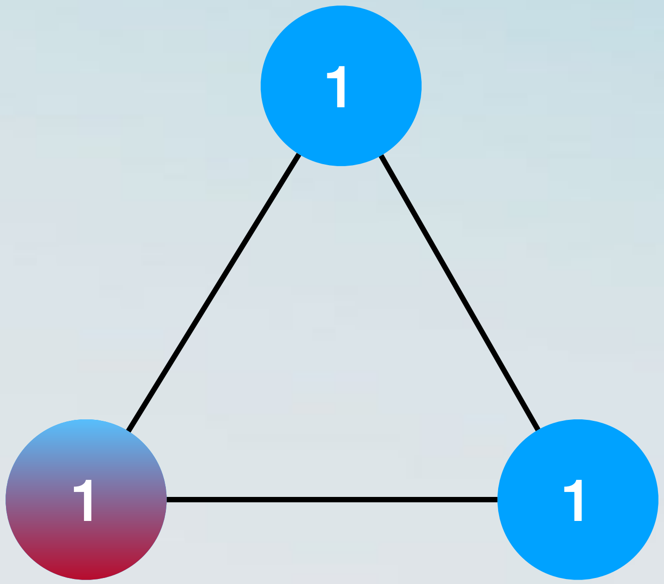
Min weight integral
vertex cover.
weight = 2



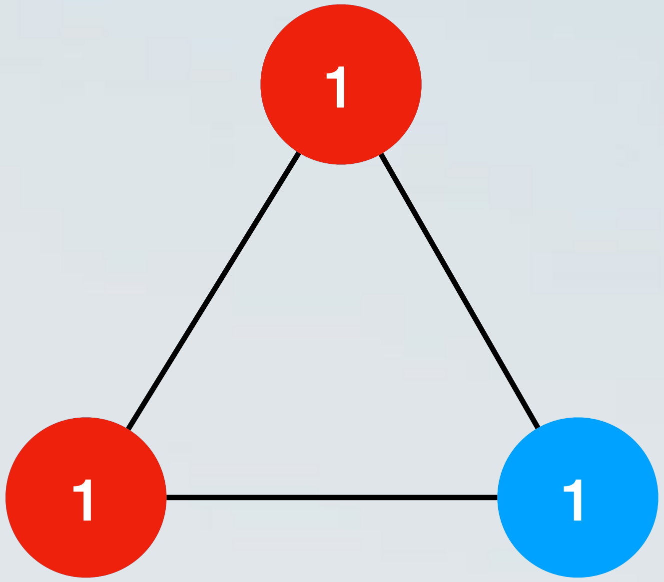
Example



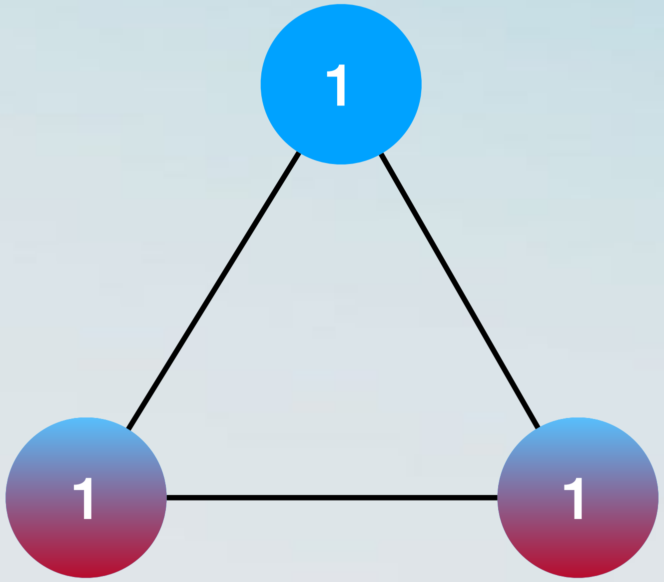
Min weight integral
vertex cover.
weight = 2



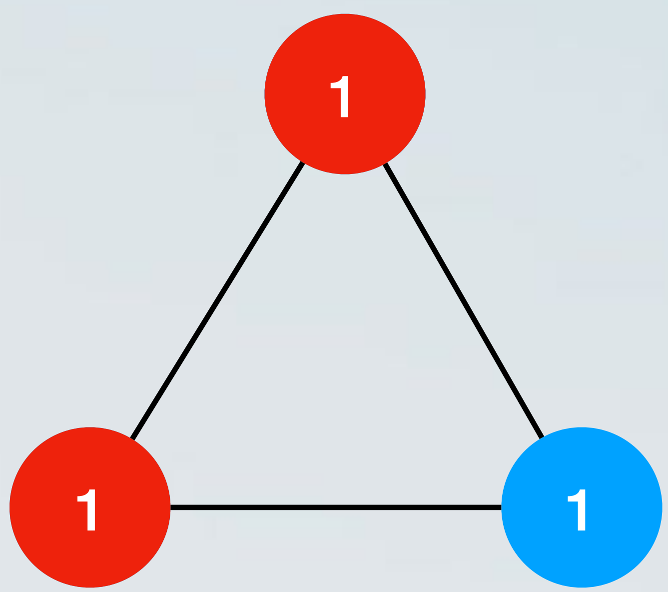
Example



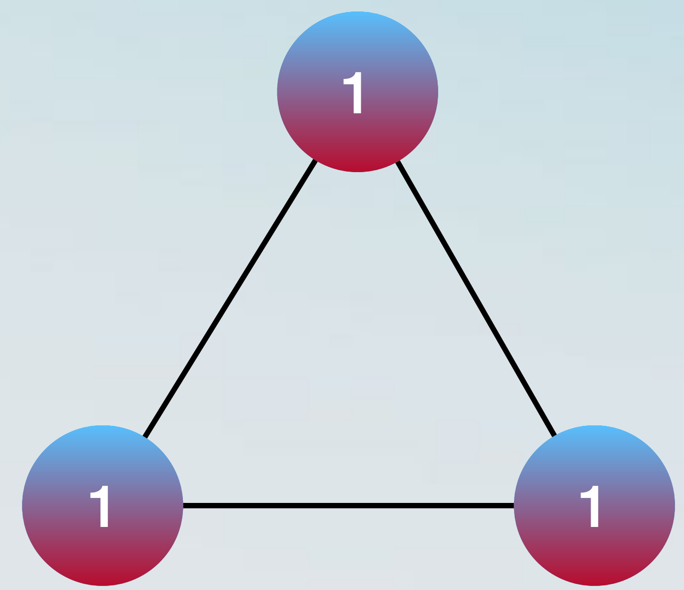
Min weight integral
vertex cover.
weight = 2



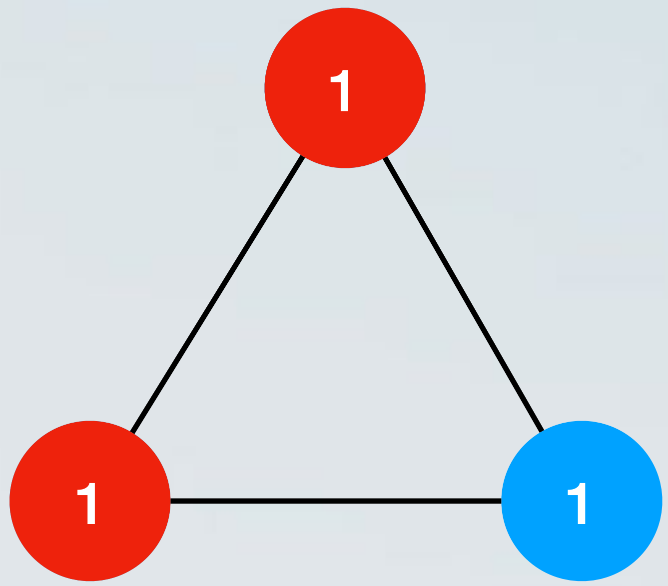
Example



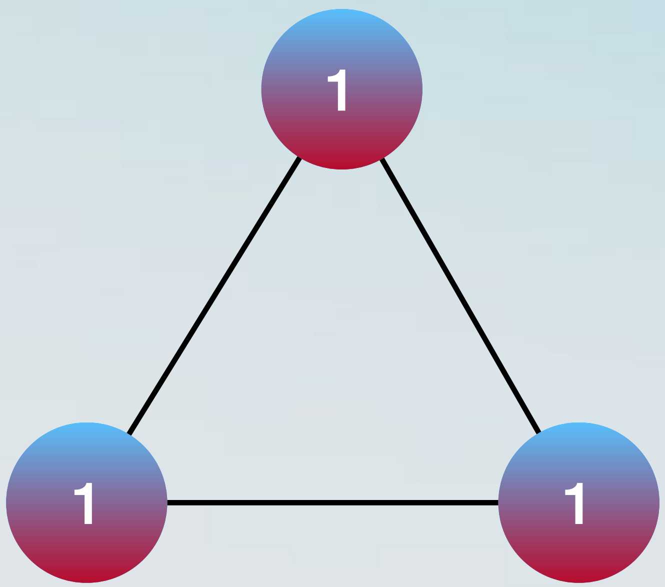
Min weight integral
vertex cover.
weight = 2



Example



Min weight integral
vertex cover.
weight = 2



Min weight fractional
vertex cover.
weight = 3/2

Solving the LP-relaxation

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a "fractional" vertex cover, where variables can take values between *0* and *1*.

Rounding the solution

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a "fractional" vertex cover, where variables can take values between *0* and *1*.
- We round the fractional solution to an integer solution.

Rounding the solution

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a "fractional" vertex cover, where variables can take values between *0* and *1*.
- We round the fractional solution to an integer solution.
  - We pick a variable x_i and we set it to *1* or *0*.

Rounding the solution

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a "fractional" vertex cover, where variables can take values between *0* and *1*.
- We round the fractional solution to an integer solution.
  - We pick a variable x_i and we set it to *1* or *0*.
  - If we set everything to *0*, it is not a vertex cover.

Rounding the solution

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a "fractional" vertex cover, where variables can take values between *0* and *1*.
- We round the fractional solution to an integer solution.
  - We pick a variable x_i and we set it to *1* or *0*.
  - If we set everything to *0*, it is not a vertex cover.
  - If we set everything to *1*, we "pay" too much.

Rounding the solution

- We can solve the LP-relaxation in polynomial time, to find an optimal solution.
- The optimal solution is a "fractional" vertex cover, where variables can take values between *0* and *1*.
- We round the fractional solution to an integer solution.
  - We pick a variable x_i and we set it to *1* or *0*.
  - If we set everything to *0*, it is not a vertex cover.
  - If we set everything to *1*, we "pay" too much.
  - We set variable x_i to *1* if $x_i \geq 1/2$ and to *0* otherwise.

Feasibility

- Claim: The solution that we obtain this way is a vertex cover.

Feasibility

- Claim: The solution that we obtain this way is a vertex cover.
  - Or, to put it differently, that the solution that the rounded solution is a feasible solution to the ILP.

Feasibility

- Claim: The solution that we obtain this way is a vertex cover.
    - Or, to put it differently, that the solution that the rounded solution is a feasible solution to the ILP.
- Easy proof: Assume that it is not.

Feasibility

- Claim: The solution that we obtain this way is a vertex cover.
  - Or, to put it differently, that the solution that the rounded solution is a feasible solution to the ILP.
- Easy proof: Assume that it is not.
  - There there is an edge e = (i, j) such that both x_i and x_j were set to *0*.

Feasibility

- Claim: The solution that we obtain this way is a vertex cover.
  - Or, to put it differently, that the solution that the rounded solution is a feasible solution to the ILP.
- Easy proof: Assume that it is not.
  - There there is an edge $e = (i, j)$ such that both x_i and x_j were set to *0*.
  - This means that in the LP-relaxation, we had that $x_i < 1/2$ and $x_j < 1/2$.

Feasibility

- Claim: The solution that we obtain this way is a vertex cover.
  - Or, to put it differently, that the solution that the rounded solution is a feasible solution to the ILP.
- Easy proof: Assume that it is not.
  - There there is an edge $e = (i, j)$ such that both x_i and x_j were set to *0*.
  - This means that in the LP-relaxation, we had that $x_i < 1/2$ and $x_j < 1/2$.
  - But then the constraint $x_i + x_i \geq 1$ would be violated, and this would not be a feasible solution to the LP-relaxation.

Vertex Cover LP-relaxation

Minimise $\sum_{i \in V} x_i w_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

Approximation ratio

- Claim: The vertex cover that we obtain this way has weight at most twice that of the minimum weight vertex cover.

Approximation ratio

- Claim: The vertex cover that we obtain this way has weight at most twice that of the minimum weight vertex cover.

- Intuition is easy: We included all the vertices for which the LP-relaxation "paid" at least 1/2, and we rounded them up to 1, so we "paid" at most twice as much.

Approximation ratio

- Claim: The vertex cover that we obtain this way has weight at most twice that of the minimum weight vertex cover.
- Intuition is easy: We included all the vertices for which the LP-relaxation "paid" at least 1/2, and we rounded them up to 1, so we "paid" at most twice as much.
- One line proof:

$$\sum_i w_i x_i^* \geq \sum_{i \in S} w_i x_i^* \geq \frac{1}{2} \sum_{i \in S} w_i = w(S)$$

Approximation ratio

- Claim: The vertex cover that we obtain this way has weight at most twice that of the minimum weight vertex cover.

- Intuition is easy: We included all the vertices for which the LP-relaxation "paid" at least 1/2, and we rounded them up to 1, so we "paid" at most twice as much.

- One line proof:

$$\sum_i w_i x_i^* \geq \sum_{i \in S} w_i x_i^* \geq \frac{1}{2} \sum_{i \in S} w_i = w(S)$$

weight of min-weight fractional VC

Approximation ratio

- Claim: The vertex cover that we obtain this way has weight at most twice that of the minimum weight vertex cover.
- Intuition is easy: We included all the vertices for which the LP-relaxation "paid" at least 1/2, and we rounded them up to 1, so we "paid" at most twice as much.
- One line proof:

$$\sum_{i} w_i x_i^* \geq \sum_{i \in S} w_i x_i^* \geq \frac{1}{2} \sum_{i \in S} w_i = w(S)$$

weight of the VC that we computed

weight of min-weight fractional VC

Approximation ratio

- The LP-relaxation and round algorithm for vertex cover has an approximation ratio of 2.
- We already knew that 2 was possible, from the Pricing method algorithm (Primal-Dual).
- In this case, the ILP-relaxation and round algorithm seems conceptually simpler.
- In other cases, rounding the solution will not be so straightforward.

Limitations of algorithms

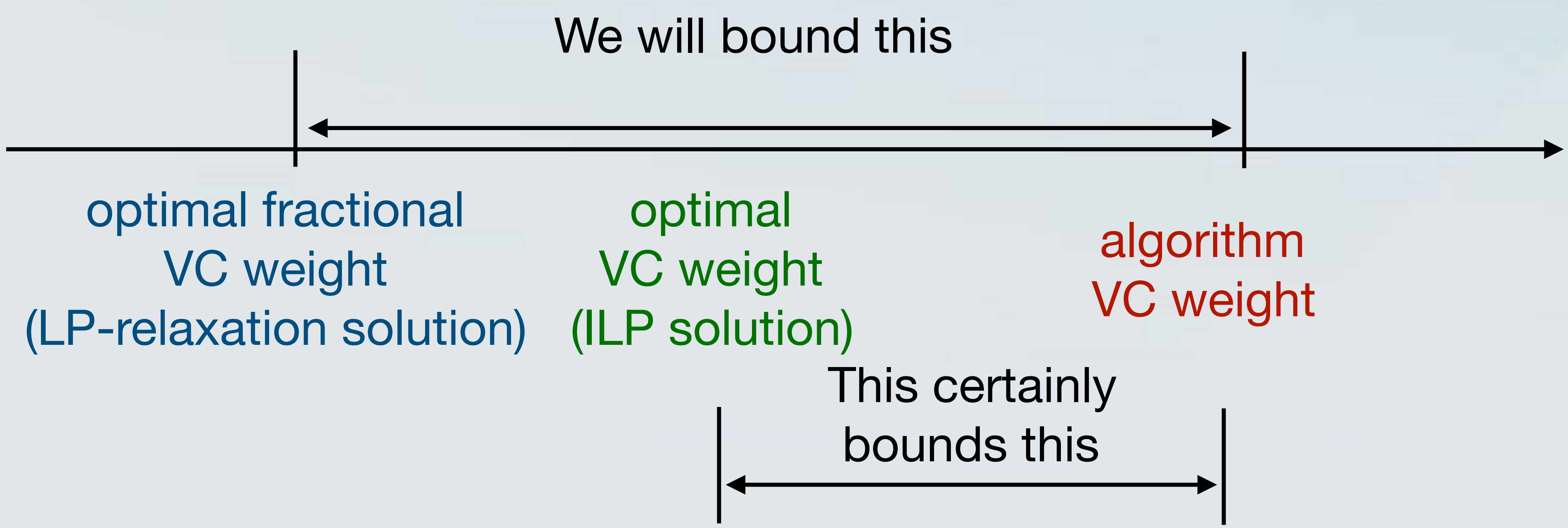
Limitations of algorithms

- How do we know if our algorithm is the best possible? Can we get "*closer*" to the optimal?

Limitations of algorithms

- How do we know if our algorithm is the best possible? Can we get “*closer*” to the optimal?
- Maybe we can use the same technique but round more cleverly?

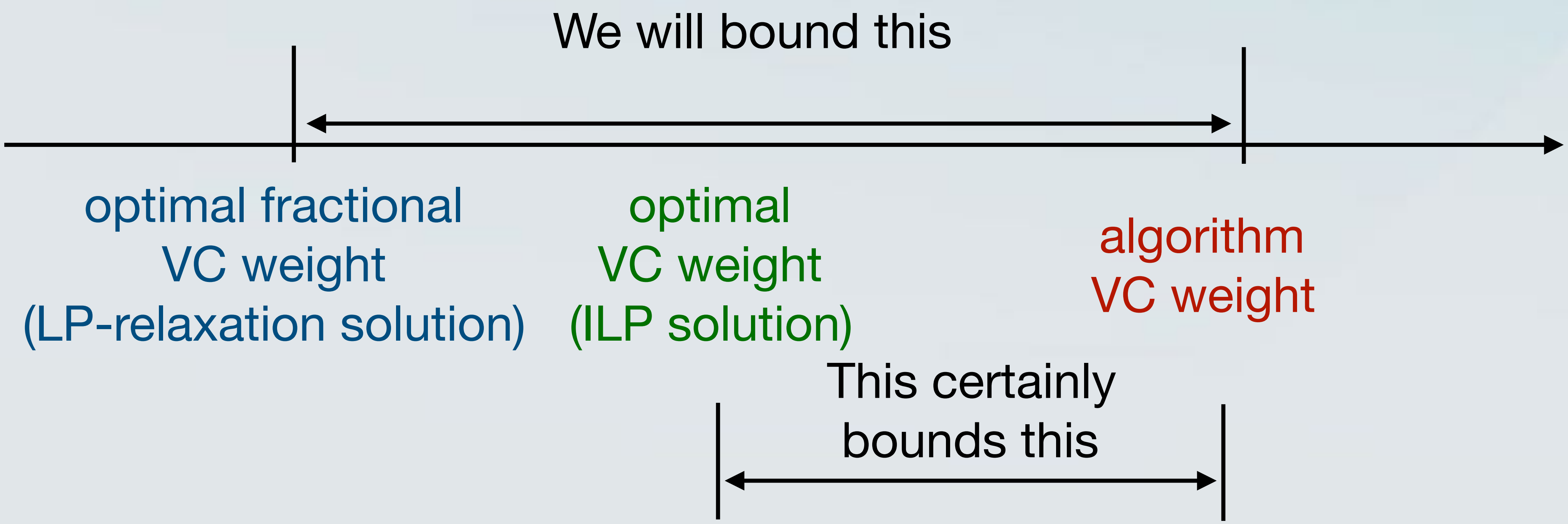
Lower bounding the optimal



We do not know the optimal, so we will use
a lower bound for the optimal.

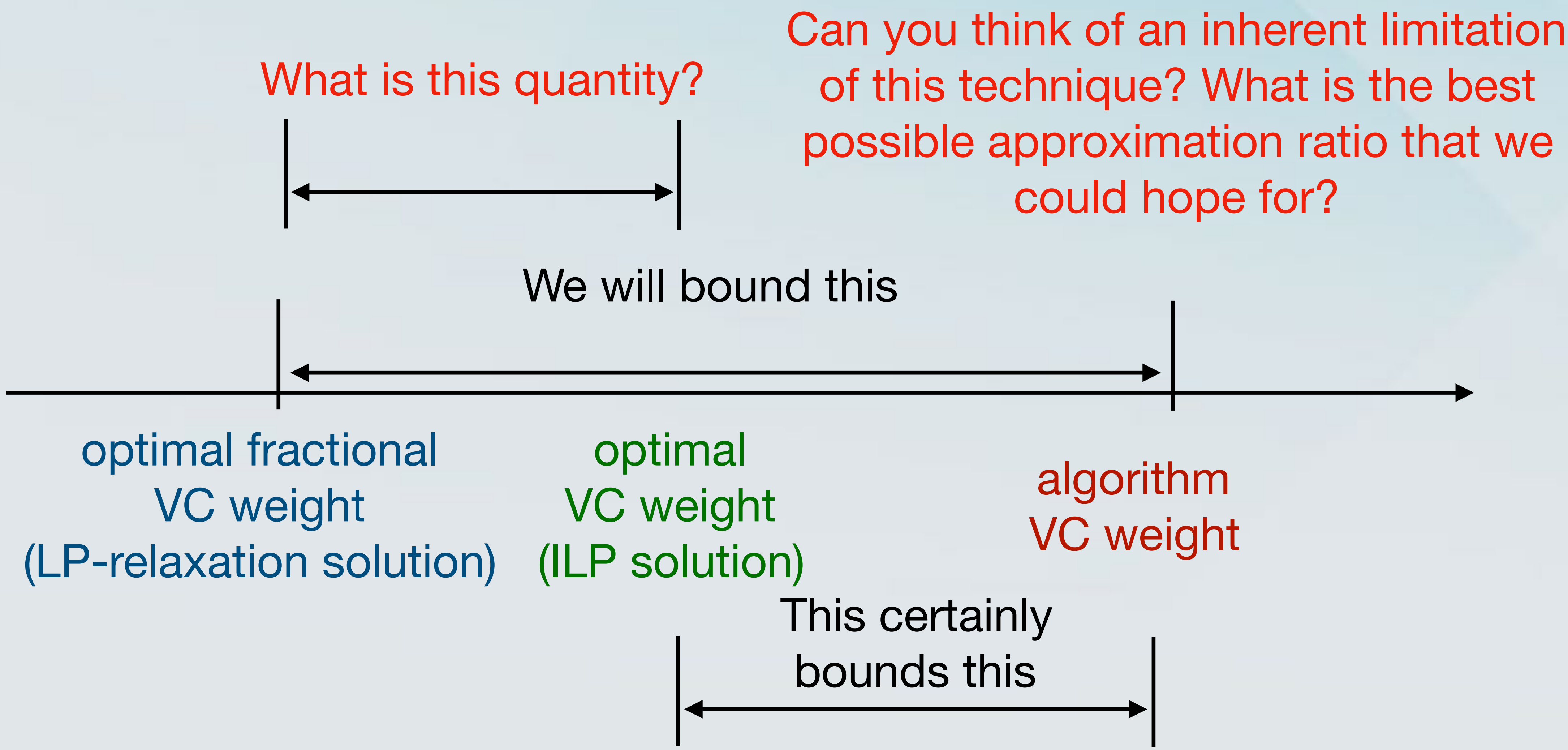
Lower bounding the optimal

Can you think of an inherent limitation of this technique? What is the best possible approximation ratio that we could hope for?



We do not know the optimal, so we will use a lower bound for the optimal.

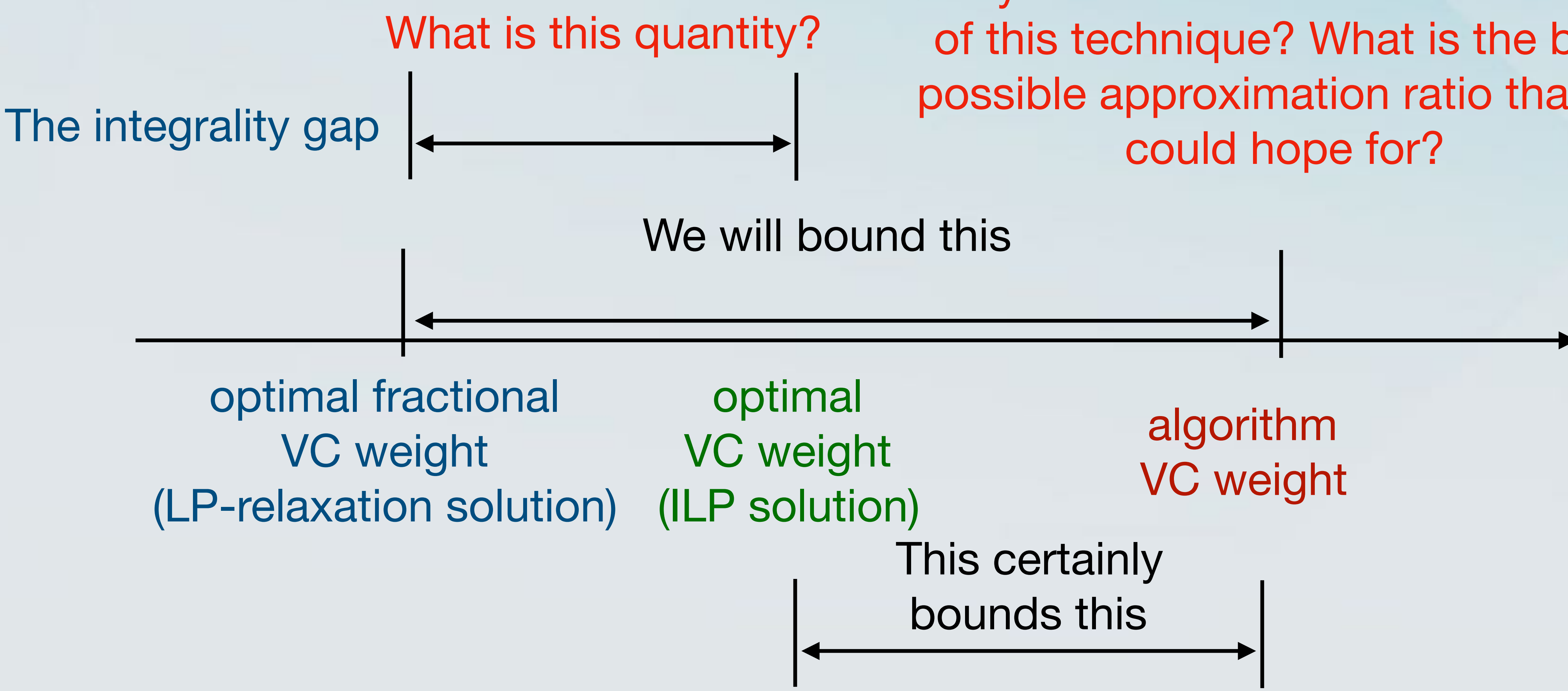
Lower bounding the optimal



We do not know the optimal, so we will use a lower bound for the optimal.

Lower bounding the optimal

Can you think of an inherent limitation of this technique? What is the best possible approximation ratio that we could hope for?



We do not know the optimal, so we will use a lower bound for the optimal.

Limitations of the technique

- The LP-relaxation and round technique cannot provide approximation ratios better than the integrality gap of the ILP-LP formulation.

Limitations of the technique

- The LP-relaxation and round technique cannot provide approximation ratios better than the integrality gap of the ILP-LP formulation.

- Caution: The integrality gap is a quantity that has to do with the formulation of the ILP-LP, not the problem.

Limitations of the technique

- The LP-relaxation and round technique cannot provide approximation ratios better than the integrality gap of the ILP-LP formulation.
- Caution: The integrality gap is a quantity that has to do with the formulation of the ILP-LP, not the problem.
    - An integrality gap of α does not mean that it is not possible to design an algorithm with approximation ratio better than α.

Limitations of the technique

- The LP-relaxation and round technique cannot provide approximation ratios better than the integrality gap of the ILP-LP formulation.
- Caution: The integrality gap is a quantity that has to do with the formulation of the ILP-LP, not the problem.
    - An integrality gap of α does not mean that it is not possible to design an algorithm with approximation ratio better than α.
    - Actually, it does not even mean that LP-relaxation and round technique cannot give you an algorithm with approximation ratio better than α.

Limitations of the technique

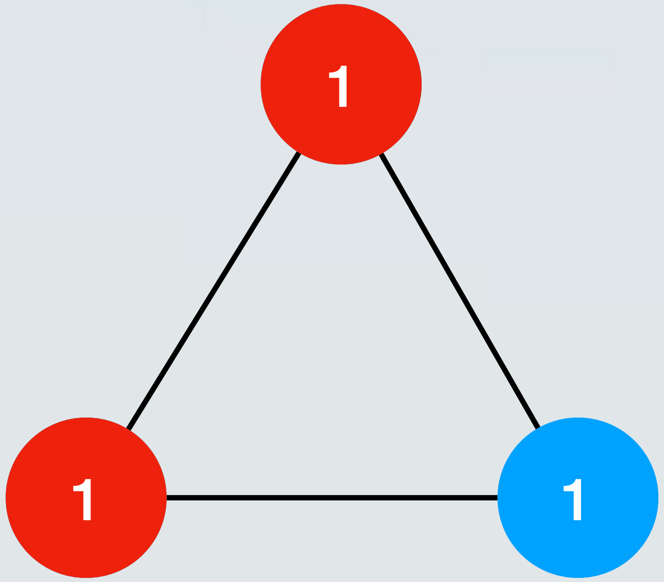
- The LP-relaxation and round technique cannot provide approximation ratios better than the integrality gap of the ILP-LP formulation.
- Caution: The integrality gap is a quantity that has to do with the formulation of the ILP-LP, not the problem.
  - An integrality gap of α does not mean that it is not possible to design an algorithm with approximation ratio better than α.
  - Actually, it does not even mean that LP-relaxation and round technique cannot give you an algorithm with approximation ratio better than α.
  - It means that *with this formulation of the ILP-LP*, α is the best you can hope for.

Integrality gap of VC

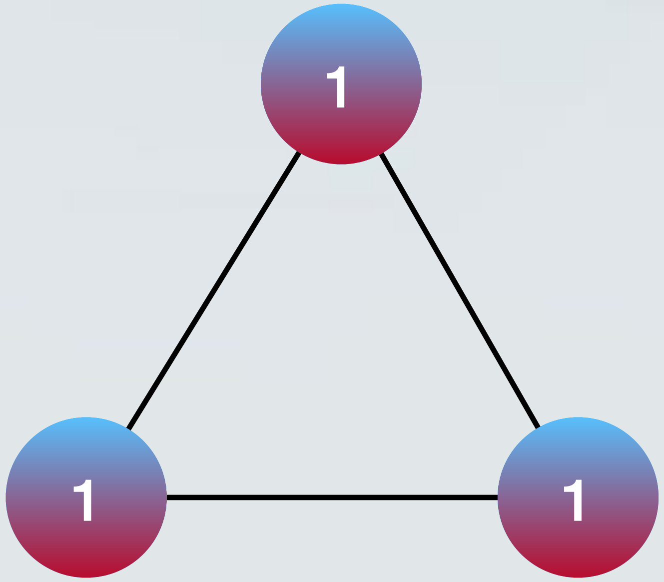
- Do we know *any lower bound* on the integrality gap of vertex cover?
    - i.e., “the IG is at least this much”.

Integrality gap of VC

- Do we know *any lower bound* on the integrality gap of vertex cover?
  - i.e., "the IG is at least this much".



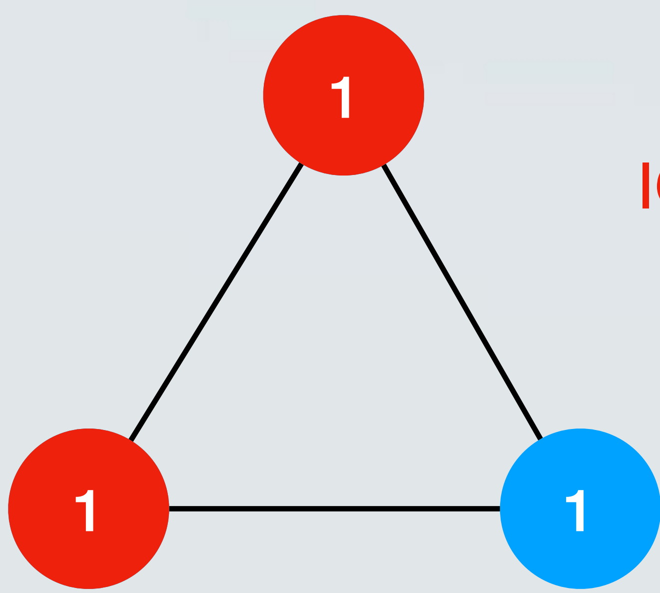
Min weight integral
vertex cover.
weight = 2



Min weight fractional
vertex cover.
weight = 3/2

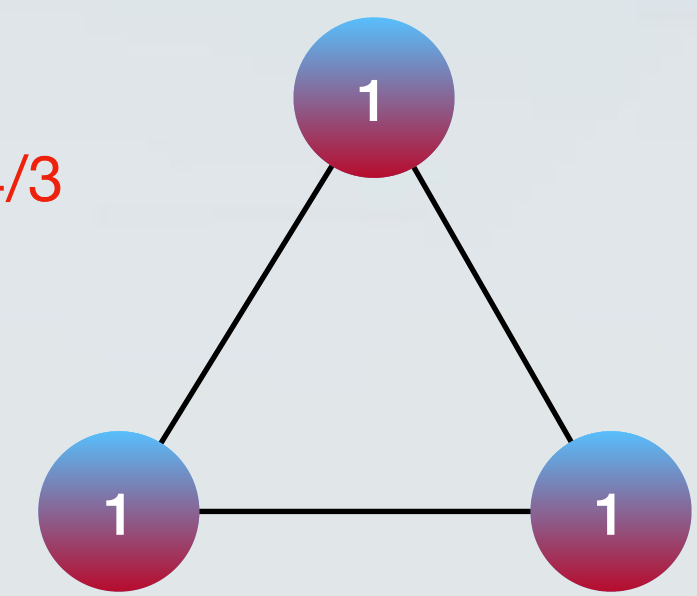
Integrality gap of VC

- Do we know *any lower bound* on the integrality gap of vertex cover?
    - i.e., “the IG is at least this much”.



Min weight integral vertex cover.
weight = 2

IG ≥ 2/(3/2) = 4/3



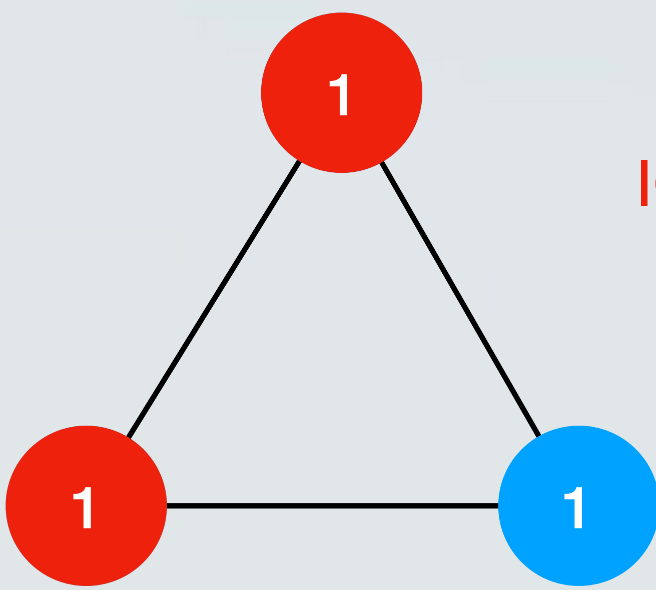
Min weight fractional vertex cover.
weight = 3/2

Integrality gap of VC

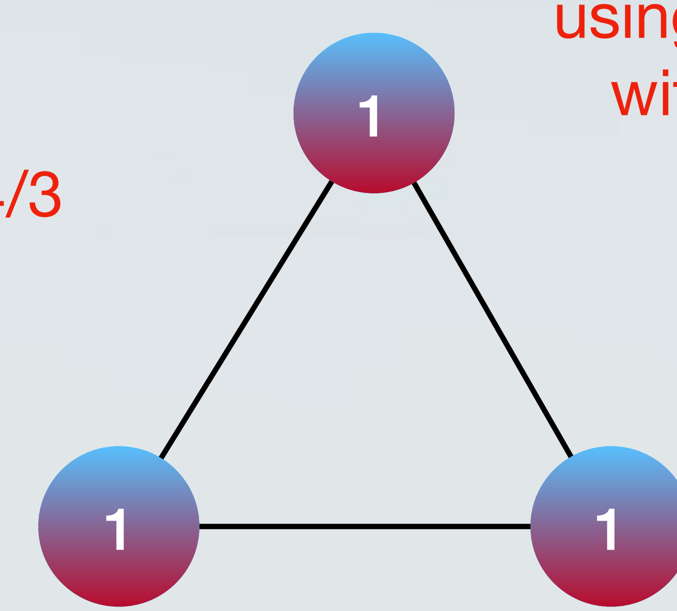
- Do we know *any lower bound* on the integrality gap of vertex cover?
  - i.e., "the IG is at least this much".

We cannot hope to design an algorithm using this formulation with ration better than 4/3.

IG ≥ 2/(3/2) = 4/3



Min weight integral vertex cover.
weight = 2



Min weight fractional vertex cover.
weight = 3/2

Integrality gap of VC

- Can we get any better lower bounds?
- The integrality gap of VC approaches 2 as the number of vertices goes to infinity.
- 5-min exercise: Try to prove this statement.

Limitations of algorithms

Limitations of algorithms

- How do we know if our algorithm is the best possible? Can we get "*closer*" to the optimal?

Limitations of algorithms

- How do we know if our algorithm is the best possible? Can we get "*closer*" to the optimal?

- Maybe we can use the same technique but round more cleverly? No.

Inapproximability of VC

- We saw two different algorithms, both provided an approximation ratio of 2.

Inapproximability of VC

- We saw two different algorithms, both provided an approximation ratio of 2.

- Could we hope to design a better algorithm for the problem?

Inapproximability of VC

- We saw two different algorithms, both provided an approximation ratio of 2.

- Could we hope to design a better algorithm for the problem?

- The answer to this question depends on our definition of “hope”.

Inapproximability of VC

- We saw two different algorithms, both provided an approximation ratio of 2.
- Could we hope to design a better algorithm for the problem?
- The answer to this question depends on our definition of “hope”.
- Known fact 1: It is impossible (unless P=NP) to design an algorithm with approximation ratio better than 1.363.

Inapproximability of VC

- We saw two different algorithms, both provided an approximation ratio of 2.

- Could we hope to design a better algorithm for the problem?

- The answer to this question depends on our definition of “hope”.

- Known fact 1: It is impossible (unless P=NP) to design an algorithm with approximation ratio better than 1.363.

- Known fact 2: It is impossible (unless Unique Games is an NP-hard problem) to design an algorithm with approximation ratio better than 2.

Inapproximability of VC

- We saw two different algorithms, both provided an approximation ratio of 2.
- Could we hope to design a better algorithm for the problem?
- The answer to this question depends on our definition of “hope”.
- Known fact 1: It is impossible (unless P=NP) to design an algorithm with approximation ratio better than 1.363.
- Known fact 2: It is impossible (unless Unique Games is an NP-hard problem) to design an algorithm with approximation ratio better than 2.
- Both facts are quite involved to prove.

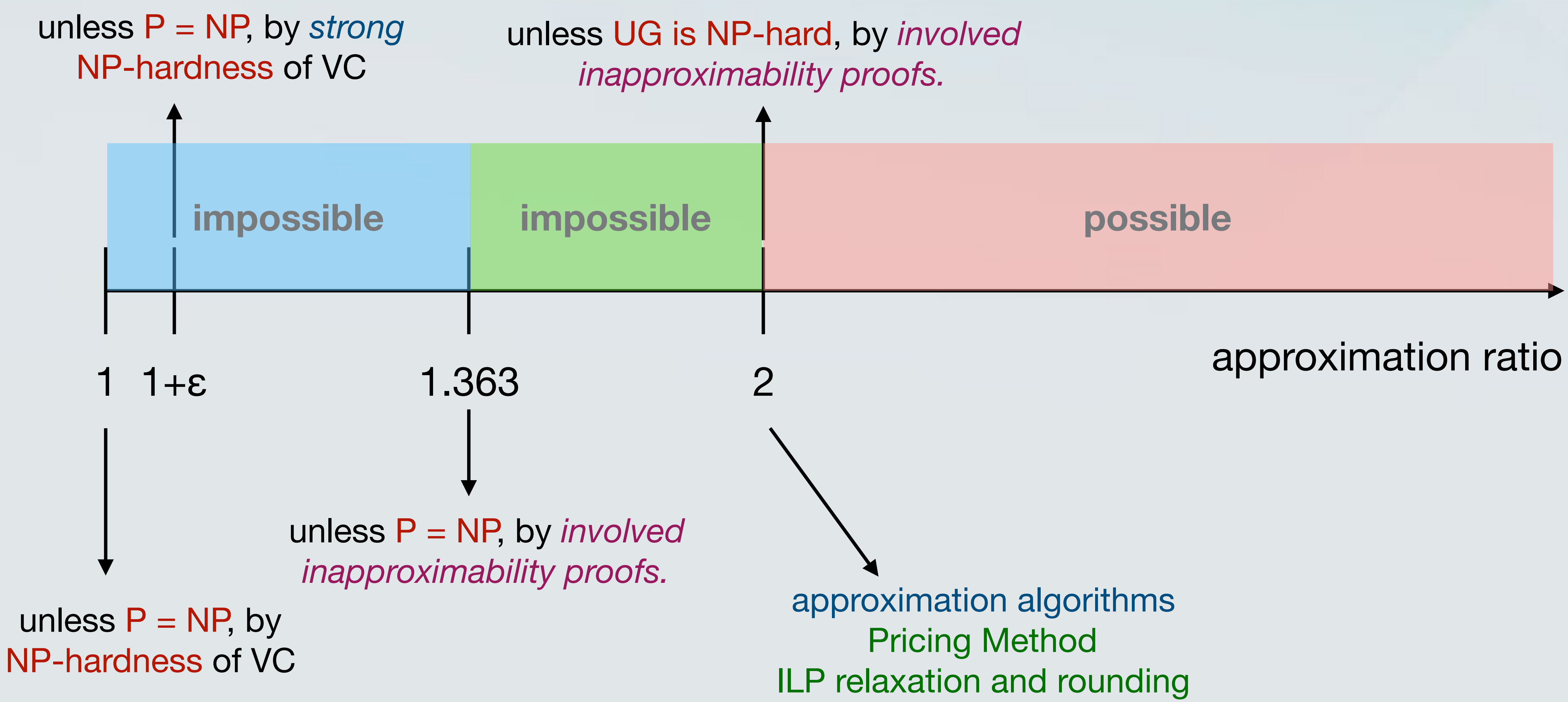
Easier inapproximability

- Definition: A problem P is *strongly* NP-hard, when there is a polynomial time reduction from a *strongly* NP-hard to problem to it.

- For a *strongly* NP-hard problem P,
  - There is **no** Fully Polynomial Time Approximation Scheme (FPTAS - *next lecture*).
  - There is **no** pseudopolynomial time algorithm that solves it exactly.

The approximation landscape for Vertex Cover

- Vertex Cover is *strongly* NP-hard.



Vertex Cover on bipartite graphs

- Definition: A vertex cover C of a bipartite graph G=(A ∪ B, E) is a subset of the nodes such that every edge e in the graph has at least one endpoint in C.

- Definition: A minimum vertex cover is a vertex cover of the smallest possible size.

- Vertex Cover on bipartite graphs
  Input: A bipartite graph G=(A ∪ B, E).
  Output: A minimum vertex cover.

Vertex Cover on bipartite graphs

- We will establish via a series of arguments that VC on bipartite graphs can be solved in polynomial time.

Vertex Cover as an ILP

Minimise $\sum_{i \in V} x_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

$x_i \in \{0, 1\},$ **for all** $i \in V$

Vertex Cover LP-relaxation

Minimise $\sum_{i \in V} x_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

The dual

Maximise $$\sum_{j \in E} y_j$$

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \text{for all } i \in V$$

$$y_j \leq 1, \quad \text{for all } j \in E$$

The ILP corresponding to the dual

Maximise $\sum_{j \in E} y_j$

subject to $\sum_{j \text{ is incident to vertex } i} y_j \leq 1,$ **for all** $i \in V$

$y_j \in \{0,1\}$ **for all** $j \in E$

The ILP corresponding to the dual

Maximise $$\sum_{j \in E} y_j$$

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \text{for all } i \in V$$

$$y_j \in \{0,1\} \quad \text{for all } j \in E$$

Include as many edges as possible …

The ILP corresponding to the dual

Maximise $$\sum_{j \in E} y_j$$

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \text{for all } i \in V$$

$$y_j \in \{0,1\} \quad \text{for all } j \in E$$

Include as many edges as possible …

such that for every vertex of the graph …

The ILP corresponding to the dual

Maximise $$\sum_{j \in E} y_j$$

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \textbf{for all } i \in V$$

$$y_j \in \{0,1\} \quad \textbf{for all } j \in E$$

Include as many edges as possible …

such that for every vertex of the graph …

among the edges that are incident to that vertex …

The ILP corresponding to the dual

Maximise $$\sum_{j \in E} y_j$$

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \text{for all } i \in V$$

$$y_j \in \{0,1\} \quad \text{for all } j \in E$$

Include as many edges as possible …

such that for every vertex of the graph …

among the edges that are incident to that vertex …

we take at most 1.

The ILP corresponding to the dual

What is this?

Maximise $\sum_{j \in E} y_j$

subject to $\sum_{j \text{ is incident to vertex } i} y_j \leq 1,$ **for all** $i \in V$

$y_j \in \{0,1\}$ **for all** $j \in E$

Include as many edges as possible …

such that for every vertex of the graph …

among the edges that are incident to that vertex …

we take at most 1.

The ILP corresponding to the dual

Maximise $$\sum_{j \in E} y_j$$

What is this?

maximum matching!

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \textbf{for all } i \in V$$

$$y_j \in \{0,1\} \quad \textbf{for all } j \in E$$

Include as many edges as possible …

such that for every vertex of the graph …

among the edges that are incident to that vertex …

we take at most 1.

König's Theorem

- In a bipartite graph, the *size of the maximum matching* is equal to the *size of the minimum vertex cover*.
    - König's proof is constructive: If starts from a maximum matching and produces a vertex cover, proving that it is minimum.
    - Alternative proof based on *total unimodularity*.

This is Maximum Matching

Maximise $\sum_{j \in E} y_j$

subject to $\sum_{j \text{ is incident to vertex } i} y_j \leq 1,$ **for all** $i \in V$

$y_j \in \{0,1\}$ **for all** $j \in E$

This is the LP-relaxation

Maximise $\sum_{j \in E} y_j$

subject to $\sum_{j \text{ is incident to vertex } i} y_j \leq 1,$ **for all** $i \in V$

$y_j \leq 1,$ **for all** $j \in E$

This is the LP-relaxation

Maximise $\sum_{j \in E} y_j$

subject to $\sum_{j \text{ is incident to vertex } i} y_j \leq 1,$ **for all** $i \in V$

$y_j \leq 1,$ **for all** $j \in E$

Fact: The incidence matrix of a bipartite graph is totally unimodular.

This is the LP-relaxation

Maximise $$\sum_{j \in E} y_j$$

subject to $$\sum_{j \text{ is incident to vertex } i} y_j \leq 1, \quad \text{for all } i \in V$$

$$y_j \leq 1, \quad \text{for all } j \in E$$

Fact: The incidence matrix of a bipartite graph is totally unimodular.

This means that size of maximum matching = size of maximum fractional matching.

This is Vertex Cover

Minimise $\sum_{i \in V} x_i$

subject to

$$x_i + x_j \geq 1, \quad \textbf{for all } (i, j) \in E$$

$$x_i \geq 0, \quad \textbf{for all } i \in V$$

$$x_i \in \{0, 1\}, \quad \textbf{for all } i \in V$$

This is the LP-relaxation

Minimise $\sum_{i \in V} x_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

This is the LP-relaxation

Minimise $\sum_{i \in V} x_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

Fact: The constraint matrix is also totally unimodular. It is just the transpose of the constraint matrix of the maximum bipartite matching problem.

This is the LP-relaxation

Minimise $\sum_{i \in V} x_i$

subject to $x_i + x_j \geq 1,$ **for all** $(i, j) \in E$

$x_i \geq 0,$ **for all** $i \in V$

Fact: The constraint matrix is also totally unimodular.
It is just the transpose of the constraint matrix of the maximum bipartite matching problem.

This means that size of minimum VC = size of minimum fractional VC

Putting everything together

Putting everything together

This means that size of maximum matching = size of maximum fractional matching.

Putting everything together

This means that size of maximum matching = size of maximum fractional matching.

This means that size of minimum VC = size of minimum fractional VC.

Putting everything together

This means that size of maximum matching = size of maximum fractional matching.

This means that size of minimum VC = size of minimum fractional VC.

But size of maximum fractional matching = size of minimum fractional VC (why?).

Putting everything together

This means that size of maximum matching = size of maximum fractional matching.

This means that size of minimum VC = size of minimum fractional VC.

But size of maximum fractional matching = size of minimum fractional VC (why?).

This means that size of maximum matching = size of minimum VC.

How do we find the size of the minimum VC on a bipartite graph?

- Solve the maximum matching on the same bipartite graph.
- The size of this matching is the size of the minimum vertex cover.

How do we find the minimum VC on a bipartite graph?

- Solve the maximum matching on the same bipartite graph.

- The size of this matching is the size of the minimum vertex cover.

How do we find the minimum VC on a bipartite graph?

- Solve the maximum matching on the same bipartite graph.
- The size of this matching is the size of the minimum vertex cover.
- Find the minimum vertex cover using the size of the minimum vertex cover.

How do we find the minimum VC on a bipartite graph?

- Solve the maximum matching on the same bipartite graph.
- The size of this matching is the size of the minimum vertex cover.
- Find the minimum vertex cover using the size of the minimum vertex cover.
  - How?

From previous lecture...

- Pick a vertex v in the graph.
  - Remove it (and the incident edges) to get graph G - {v}.
  - Property: If v was in any minimum vertex cover, G - {v} has a minimum vertex cover of size k*-1.
  - Check if the graph G - {v} has a vertex cover of size at most k*-1.
    - Yes: Include v in the vertex cover.
    - No: Do not include v in the vertex cover.
    - Then move to the next vertex.

Summing up

- Vertex Cover is strongly NP-hard in general.
  - In fact, hard to approximate better than 1.363.
  - There exist 2-approximate polynomial time algorithms for the problem.
- On bipartite graphs, the problem is solvable in polynomial time.