Lecture 20
Approximation using linear programming

Markus Jalsenius
A GOOD DAY TO APPROXIMATE NP-HARDNESS
Vertex cover

Let $G = (V, E)$ be an undirected graph.
Vertex cover

- Let $G = (V, E)$ be an undirected graph.
- If $S \subseteq V$ is a set of vertices and $e$ is an edge, we say that $S$ covers $e$ if at least one endpoint of $e$ is in $S$.

**Example**

- The vertices $\bigcirc$ are in $S$.
- The red edges are covered by $S$. 
Vertex cover

Let $G = (V, E)$ be an undirected graph.

If $S \subseteq V$ is a set of vertices and $e$ is an edge, we say that $S$ covers $e$ if at least one endpoint of $e$ is in $S$.

**Example**

- The vertices \[ \text{\textcolor{yellow}{\circ}} \] are in $S$.
- The red edges are covered by $S$.

$S$ is a vertex cover if it covers every edge.
Vertex cover

- Let $G = (V, E)$ be an undirected graph.
- If $S \subseteq V$ is a set of vertices and $e$ is an edge, we say that $S$ covers $e$ if at least one endpoint of $e$ is in $S$.

**Example**

- The vertices \( \bullet \) are in $S$.
- The red edges are covered by $S$.

- $S$ is a vertex cover if it covers every edge.

**Example**

- The vertices \( \bullet \) are in $S$.
- The red edges are covered by $S$.
- $S$ is a vertex cover.
Vertex cover

In the **minimum vertex cover problem** we want to compute a vertex cover of minimal size.
Vertex cover

- In the **minimum vertex cover problem** we want to compute a vertex cover of minimal size.

- In the decision version, called the **vertex cover problem**, we want to determine if there is a vertex cover of size at most $k$, where $k$ is part of the instance.
Vertex cover

- In the **minimum vertex cover problem** we want to compute a vertex cover of minimal size.
- In the decision version, called the **vertex cover problem**, we want to determine if there is a vertex cover of size at most $k$, where $k$ is part of the instance.
- This problem is NP-complete.
Vertex cover

In the **minimum vertex cover problem** we want to compute a vertex cover of minimal size.

In the decision version, called the **vertex cover problem**, we want to determine if there is a vertex cover of size at most $k$, where $k$ is part of the instance.

This problem is NP-complete.

Hence, minimum vertex cover is NP-hard.

No worries...  
...I've got LPs.
Weighted vertex cover

Let $G = (V, E)$ be an undirected graph where each vertex $v$ has a weight $w_v \geq 0$. 

Weighted vertex cover

Let $G = (V, E)$ be an undirected graph where each vertex $v$ has a weight $w_v \geq 0$.

Given a vertex cover $S$, the weight of $S$ is the sum of the weights of the vertices in $S$. 
Weighted vertex cover

Let $G = (V, E)$ be an undirected graph where each vertex $v$ has a weight $w_v \geq 0$.

Given a vertex cover $S$, the weight of $S$ is the sum of the weights of the vertices in $S$.

In the **weighted vertex cover problem** we want to compute a minimum weight vertex cover.
Weighted vertex cover

- Let $G = (V, E)$ be an undirected graph where each vertex $v$ has a weight $w_v \geq 0$.
- Given a vertex cover $S$, the weight of $S$ is the sum of the weights of the vertices in $S$.
- In the **weighted vertex cover problem** we want to compute a minimum weight vertex cover.

**Example**

Two different vertex covers of a graph with vertex weights.

- The vertices $\bullet$ are a vertex cover of weight $8 + 7 + 5 + 3 = 23$.
- The vertices $\bullet$ are a vertex cover of weight $1 + 5 + 7 + 2 = 15$. 

Weighted vertex cover

- When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.
Weighted vertex cover

- When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.
- We will now express weighted vertex cover as an integer program, which must therefore also be NP-hard.
Weighted vertex cover

- When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.
- We will now express weighted vertex cover as an integer program, which must therefore also be NP-hard.
- For any set $S \subseteq V$ of vertices, let, for all $v \in V$, $x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise.} \end{cases}$
Weighted vertex cover

► When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.

► We will now express weighted vertex cover as an integer program, which must therefore also be NP-hard.

► For any set $S \subseteq V$ of vertices, let, for all $v \in V$, $x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$

► $S$ is a vertex cover if and only if $x_u + x_v \geq 1$ for every edge $(u, v)$. 
**Weighted vertex cover**

- When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.

- We will now express weighted vertex cover as an *integer program*, which must therefore also be NP-hard.

- For any set $S \subseteq V$ of vertices, let, for all $v \in V$, $x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise.} \end{cases}$

- $S$ is a vertex cover if and only if $x_u + x_v \geq 1$ for every edge $(u, v)$.

- Thus, weighted vertex cover can be expressed with the following integer program:

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \in \{0, 1\} \quad \forall v \in V
\end{align*}
\]
Weighted vertex cover

- When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.
- We will now express weighted vertex cover as an integer program, which must therefore also be NP-hard.
- For any set $S \subseteq V$ of vertices, let, for all $v \in V$, $x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise.} \end{cases}$
- $S$ is a vertex cover if and only if $x_u + x_v \geq 1$ for every edge $(u, v)$.
- Thus, weighted vertex cover can be expressed with the following integer program:

$$\text{minimize } \sum_{v \in V} w_v x_v$$
$$\text{subject to } x_u + x_v \geq 1 \quad \forall (u, v) \in E$$
$$x_v \in \{0, 1\} \quad \forall v \in V$$

- We will design an approximation algorithm for weighted vertex cover by allowing $x_v$ to take any non-negative real values.
Weighted vertex cover

- When all vertex weights are 1, the weighted vertex cover problem is the same as the minimum vertex cover problem, hence is NP-hard.
- We will now express weighted vertex cover as an integer program, which must therefore also be NP-hard.
- For any set $S \subseteq V$ of vertices, let, for all $v \in V$, $x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$.
- $S$ is a vertex cover if and only if $x_u + x_v \geq 1$ for every edge $(u, v)$.
- Thus, weighted vertex cover can be expressed with the following integer program:

We change $x_v \in \{0, 1\}$ to $x_v \geq 0$ and observe that $x_v$ is never more than 1 in the optimal solution anyway. Why?

We will design an approximation algorithm for weighted vertex cover by allowing $x_v$ to take any non-negative real values.
Linear program

We now have the following *linear program* (not integer program):

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]
We now have the following linear program (not integer program):

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]

Example

The smallest weight vertex cover is 2 for a 3-cycle where each vertex has weight 1.
We now have the following linear program (not integer program):

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]

**Example**

- The smallest weight vertex cover is 2 for a 3-cycle where each vertex has weight 1.

- However, if we call the vertices \(a, b, c\),
  the objective function in the linear program above evaluates to \(\frac{3}{2}\) if we set \(x_a = x_b = x_c = \frac{1}{2}\) (which is a valid solution to the LP).
Linear program

We now have the following *linear program* (not integer program):

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]

**Example**

- The smallest weight vertex cover is 2 for a 3-cycle where each vertex has weight 1.
- However, if we call the vertices \(a, b, c\), the objective function in the linear program above evaluates to \(\frac{3}{2}\) if we set \(x_a = x_b = x_c = \frac{1}{2}\) (which is a valid solution to the LP).

**Conclusion:** the linear program can produce fractional solutions, with an optimum that is less than that of the integer program.
Rounding

We can solve the linear program in polynomial time, but as we have seen, the solution may be fractional.
We can solve the linear program in polynomial time, but as we have seen, the solution may be fractional.

- The idea is to post-process the fractional solution to obtain an actual vertex cover.
Observation

We can solve the linear program in polynomial time, but as we have seen, the solution may be fractional.

The idea is to post-process the fractional solution to obtain an actual vertex cover.

For all $v \in V$, let $x_v$ be a solution to the linear program. We define

$$
\tilde{x}_v = \begin{cases} 
1 & \text{if } x_v \geq \frac{1}{2} \\
0 & \text{otherwise.}
\end{cases}
$$
The idea is to post-process the fractional solution to obtain an actual vertex cover.

For all $v \in V$, let $x_v$ be a solution to the linear program. We define

$$\tilde{x}_v = \begin{cases} 1 & \text{if } x_v \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Let $S = \{ v \mid \tilde{x}_v = 1 \}$. Then $S$ must be a vertex cover.
Rounding

**Observe**

We can solve the linear program in polynomial time, but as we have seen, the solution may be fractional.

- The idea is to post-process the fractional solution to obtain an actual vertex cover.

- For all $v \in V$, let $x_v$ be a solution to the linear program. We define

$$
\tilde{x}_v = \begin{cases} 
1 & \text{if } x_v \geq \frac{1}{2} \\
0 & \text{otherwise.}
\end{cases}
$$

- Let $S = \{v \mid \tilde{x}_v = 1\}$. Then $S$ must be a vertex cover.

To see this, note that for every edge $(u, v) \in E$, the constraint $x_u + x_v \geq 1$ is true, which implies that at least one of $x_u, x_v$ must be at least $\frac{1}{2}$. Thus, at least one endpoint of each edge is included in $S$. 
Approximation algorithm

- The approximation algorithm to weighted vertex cover is:
  - Solve the previous linear program.
  - Compute the vertex cover $S$ by rounding (previous slide).
Approximation algorithm

The approximation algorithm to weighted vertex cover is:

1. Solve the previous linear program.
2. Compute the vertex cover $S$ by rounding (previous slide).

The running time is polynomial, but what is the approximation ratio?
Approximation algorithm

- The approximation algorithm to weighted vertex cover is:
  - Solve the previous linear program.
  - Compute the vertex cover $S$ by rounding (previous slide).

- The running time is polynomial, but what is the approximation ratio?

- The rounding rule $\tilde{x}_v = \begin{cases} 1 & \text{if } x_v \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$

  has the property that for all $v$, $\tilde{x}_v \leq 2x_v$ (since $x_v$ is at most 1).
Approximation algorithm

- The approximation algorithm to weighted vertex cover is:
  - Solve the previous linear program.
  - Compute the vertex cover $S$ by rounding (previous slide).
- The running time is polynomial, but what is the approximation ratio?
- The rounding rule $\tilde{x}_v = \begin{cases} 1 & \text{if } x_v \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$
  has the property that for all $v$, $\tilde{x}_v \leq 2x_v$ (since $x_v$ is at most 1).
- Let OPT denote the optimum vertex cover. The weight of $S$ is

$$\sum_{v \in S} w_v$$
Approximation algorithm

- The approximation algorithm to weighted vertex cover is:
  - Solve the previous linear program.
  - Compute the vertex cover $S$ by rounding (previous slide).

- The running time is polynomial, but what is the approximation ratio?

- The rounding rule $\tilde{x}_v = \begin{cases} 1 & \text{if } x_v \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$

  has the property that for all $v$, $\tilde{x}_v \leq 2x_v$ (since $x_v$ is at most 1).

- Let OPT denote the optimum vertex cover. The weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in V} w_v \tilde{x}_v$$

By definition
Approximation algorithm

- The approximation algorithm to weighted vertex cover is:
  - Solve the previous linear program.
  - Compute the vertex cover $S$ by rounding (previous slide).

- The running time is polynomial, but what is the approximation ratio?

- The rounding rule $\tilde{x}_v = \begin{cases} 1 & \text{if } x_v \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$

  has the property that for all $v$, $\tilde{x}_v \leq 2x_v$ (since $x_v$ is at most 1).

- Let OPT denote the optimum vertex cover. The weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in V} w_v \tilde{x}_v \leq 2 \sum_{v \in V} w_v x_v$$

By definition
Approximation algorithm

- The approximation algorithm to weighted vertex cover is:
  - Solve the previous linear program.
  - Compute the vertex cover $S$ by rounding (previous slide).

- The running time is polynomial, but what is the approximation ratio?

- The rounding rule $\tilde{x}_v = \begin{cases} 1 & \text{if } x_v \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$

  has the property that for all $v$, $\tilde{x}_v \leq 2x_v$ (since $x_v$ is at most 1).

- Let $OPT$ denote the optimum vertex cover. The weight of $S$ is

  $$\sum_{v \in S} w_v = \sum_{v \in V} w_v \tilde{x}_v \leq 2 \sum_{v \in V} w_v x_v \leq 2 \sum_{v \in OPT} w_v.$$  

  By definition  

  The optimum of the linear program is less than or equal to the optimum of the integer program.  

  (The feasible region of the LP is at least as big as that of the integer program.)
Approximation algorithm

The approximation algorithm to weighted vertex cover is:

- Solve the previous linear program.
- Compute the vertex cover $S$ by rounding (previous slide).

The running time is polynomial, but what is the approximation ratio?

The rounding rule $\tilde{x}_v = \begin{cases} 
1 & \text{if } x_v \geq \frac{1}{2} \\
0 & \text{otherwise.} 
\end{cases}$

has the property that for all $v$, $\tilde{x}_v \leq 2x_v$ (since $x_v$ is at most 1).

Let OPT denote the optimum vertex cover. The weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in V} w_v \tilde{x}_v \leq 2 \sum_{v \in V} w_v x_v \leq 2 \sum_{v \in \text{OPT}} w_v.$$

By definition

The optimum of the linear program is less than or equal to the optimum of the integer program.

Thus, the approximation factor is 2.

(The feasible region of the LP is at least as big as that of the integer program.)
LP overkill

- Solving this linear program seems like a lot of work, and how fast is it anyway?
LP overkill

- Solving this linear program seems like a lot of work, and how fast is it anyway?
- It is polynomial time, but what if we want linear time?
LP overkill

- Solving this linear program seems like a lot of work, and how fast is it anyway?
- It is polynomial time, but what if we want *linear time*?
- Let’s do something crazy: let’s solve the *dual* of the linear program instead. This will, as we know, produce the same optimum value.
Constructing the dual

Here is the LP again:

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]
Constructing the dual

Here is the LP again:

Introduce multipliers $y_e$, one for each inequality (i.e. edge $e \in E$).

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]
Constructing the dual

Here is the LP again:

Introduce multipliers $y_e$, one for each inequality (i.e. edge $e \in E$).

Add the $|E|$ inequalities:

The right hand side is $\sum_{e \in E} y_e$, which is the new objective function we want to maximise.
Constructing the dual

Here is the LP again:

Introduce multipliers $y_e$, one for each inequality (i.e. edge $e \in E$).

Add the $|E|$ inequalities:

- The right hand side is $\sum_{e \in E} y_e$, which is the new objective function we want to maximise.
- Let $\delta(v)$ be the set of edges that has vertex $v$ as an endpoint. The total $x_v$ coefficient on the left hand side is $\sum_{e \in \delta(v)} y_e$. 

\[ \begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*} \]
Constructing the dual

Here is the LP again:

Introduce multipliers $y_e$, one for each inequality (i.e. edge $e \in E$).

Add the $|E|$ inequalities:

- The right hand side is $\sum_{e \in E} y_e$, which is the new objective function we want to maximise.
- Let $\delta(v)$ be the set of edges that has vertex $v$ as an endpoint. The total $x_v$ coefficient on the left hand side is $\sum_{e \in \delta(v)} y_e$, which must be upper bounded by $w_v$. 

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w_v x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& \quad x_v \geq 0 \quad \forall v \in V
\end{align*}
\]
Constructing the dual

Here is the LP again:

Introduce multipliers \( y_e \), one for each inequality (i.e. edge \( e \in E \)).

Add the \( |E| \) inequalities:

- The right hand side is \( \sum_{e \in E} y_e \), which is the new objective function we want to maximise.
- Let \( \delta(v) \) be the set of edges that has vertex \( v \) as an endpoint. The total \( x_v \) coefficient on the left hand side is \( \sum_{e \in \delta(v)} y_e \), which must be upper bounded by \( w_v \).

Thus, the dual is:

\[
\begin{align*}
\text{maxmise} & \quad \sum_{e \in E} y_e \\
\text{subject to} & \quad \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \\
& \quad y_e \geq 0 \quad \forall e \in E
\end{align*}
\]
The dual LP again:

\[
\text{maxmise } \sum_{e \in E} y_e \\
\text{subject to } \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \\
y_e \geq 0 \quad \forall e \in E
\]
The dual

The dual LP again:

Example

\[ y_1 + \cdots + y_6 \leq w_v \]

maximise \[ \sum_{e \in E} y_e \]

subject to \[ \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \]
\[ y_e \geq 0 \quad \forall e \in E \]
The dual

- The dual LP again:
  \[
  \text{maximise} \sum_{e \in E} y_e \\
  \text{subject to} \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \\
  y_e \geq 0 \quad \forall e \in E
  \]

- Example:

![Graph example](image)

- Observe:

- We may interpret
  - the variables \( y_e \) as *prices* associated to the edges,
  - and \( w_v \) as the *wealth* to pay for all of the edges incident to it.
The dual LP again:

\[ \text{maxmise} \sum_{e \in E} y_e \]

subject to

\[ \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \]

\[ y_e \geq 0 \quad \forall e \in E \]

**Example**

\[ y_1 + \cdots + y_6 \leq w_v \]

**Observe**

**We may interpret**

- the variables \( y_e \) as *prices* associated to the edges,
- and \( w_v \) as the *wealth* to pay for all of the edges incident to it.

**If edge prices satisfy the constraints in the dual, then every vertex has enough wealth to pay for its incident edges.**
The dual

The dual LP again:

maximise \[ \sum_{e \in E} y_e \]
subject to \[ \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \]
\[ y_e \geq 0 \quad \forall e \in E \]

Example

\[ y_1 + \cdots + y_6 \leq w_v \]

Observe

- We may interpret the variables \( y_e \) as prices associated to the edges,
- and \( w_v \) as the wealth to pay for all of the edges incident to it.
- If edge prices satisfy the constraints in the dual, then every vertex has enough wealth to pay for its incident edges.
- In particular, if \( S \) is a vertex cover then the combined wealth of the vertices in \( S \) must be at least \( \sum_{e \in E} y_e \).
The dual

The dual LP again:

\[
\begin{align*}
\text{maxmise} & \quad \sum_{e \in E} y_e \\
\text{subject to} & \quad \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \\
& \quad y_e \geq 0 \quad \forall e \in E
\end{align*}
\]

We want to maximise the combined price of all edges subject to the constraint that each vertex has enough wealth to pay for all the edges it covers.
The dual

The dual LP again:

\[
\begin{align*}
\text{maxmise} & \quad \sum_{e \in E} y_e \\
\text{subject to} & \quad \sum_{e \in \delta(v)} y_e \leq w_v \quad \forall v \in V \\
& \quad y_e \geq 0 \quad \forall e \in E
\end{align*}
\]

We want to maximise the combined price of all edges subject to the constraint that each vertex has enough wealth to pay for all the edges it covers.

We use a greedy approach:

Go through the edges in arbitrary order, increasing the price of each one as much as possible.
The algorithm

- Initialise: $S' = \emptyset$ (start with an empty vertex set)
- $s_v = 0 \quad \forall v \in V$. This is how much a vertex has spent.
The algorithm

- Initialise:  
  - $S = \emptyset$ (start with an empty vertex set)
  - $s_v = 0 \ \forall v \in V$. This is how much a vertex has spent.

- For all $e \in E$: 
  - $u$ has spent $s_u$ out of $w_u$ so far. Let $\Delta_u = w_u - s_u$.
  - $v$ has spent $s_v$ out of $w_v$ so far. Let $\Delta_v = w_v - s_v$. 
The algorithm

- Initialise: $S = \emptyset$ (start with an empty vertex set)
  - $s_v = 0 \ \forall v \in V$. This is how much a vertex has spent.

- For all $e \in E$:
  - $u$ has spent $s_u$ out of $w_u$ so far. Let $\Delta_u = w_u - s_u$.
  - $v$ has spent $s_v$ out of $w_v$ so far. Let $\Delta_v = w_v - s_v$.
  - Let $\Delta = \min\{\Delta_u, \Delta_v\}$.
The algorithm

- Initialise:
  - $S = \emptyset$ (start with an empty vertex set)
  - $s_v = 0 \ \forall v \in V$. This is how much a vertex has spent.

- For all $e \in E$:
  - $u$ has spent $s_u$ out of $w_u$ so far. Let $\Delta_u = w_u - s_u$.
  - $v$ has spent $s_v$ out of $w_v$ so far. Let $\Delta_v = w_v - s_v$.
  - Let $\Delta = \min\{\Delta_u, \Delta_v\}$.
  - Set the price of $y_e$ to $\Delta$ and add $\Delta$ to both $s_u$ and $s_v$. 
The algorithm

- Initialise: \( S = \emptyset \) (start with an empty vertex set)
  - \( s_v = 0 \) \( \forall v \in V \). This is how much a vertex has spent.

- For all \( e \in E \):
  - \( u \) has spent \( s_u \) out of \( w_u \) so far. Let \( \Delta_u = w_u - s_u \).
  - \( v \) has spent \( s_v \) out of \( w_v \) so far. Let \( \Delta_v = w_v - s_v \).
  - Let \( \Delta = \min\{ \Delta_u, \Delta_v \} \).
  - Set the price of \( y_e \) to \( \Delta \) and add \( \Delta \) to both \( s_u \) and \( s_v \).
  - A vertex that has now spent all its wealth is added to \( S \), which means at least one (or possibly both) of \( u \) and \( v \) is added to \( S \).
The algorithm

- Initialise: \( S = \emptyset \) (start with an empty vertex set)
  - \( s_v = 0 \) \( \forall v \in V \). This is how much a vertex has spent.

- For all \( e \in E \):
  - \( u \) has spent \( s_u \) out of \( w_u \) so far. Let \( \Delta_u = w_u - s_u \).
  - \( v \) has spent \( s_v \) out of \( w_v \) so far. Let \( \Delta_v = w_v - s_v \).
  - Let \( \Delta = \min\{\Delta_u, \Delta_v\} \).
  - Set the price of \( y_e \) to \( \Delta \) and add \( \Delta \) to both \( s_u \) and \( s_v \).
  - A vertex that has now spent all its wealth is added to \( S \), which means at least one (or possibly both) of \( u \) and \( v \) is added to \( S \).

- Return \( S \) when all edges have been considered.
The algorithm

- Initialise:  
  - $S = \emptyset$ (start with an empty vertex set)
  - $s_v = 0$ $\forall v \in V$. This is how much a vertex has spent.

- For all $e \in E$:
  - $u$ has spent $s_u$ out of $w_u$ so far. Let $\Delta_u = w_u - s_u$.
  - $v$ has spent $s_v$ out of $w_v$ so far. Let $\Delta_v = w_v - s_v$.
  - Let $\Delta = \min\{\Delta_u, \Delta_v\}$.
  - Set the price of $y_e$ to $\Delta$ and add $\Delta$ to both $s_u$ and $s_v$.
  - A vertex that has now spent all its wealth is added to $S$, which means at least one (or possibly both) of $u$ and $v$ is added to $S$.

- Return $S$ when all edges have been considered.

- The returned set $S$ is a vertex cover. Why?
The algorithm

- ** Initialise:**
  - \( S = \emptyset \)  (start with an empty vertex set)
  - \( s_v = 0 \ \forall v \in V \). This is how much a vertex has spent.

- **For all** \( e \in E \):
  - u has spent \( s_u \) out of \( w_u \) so far. Let \( \Delta_u = w_u - s_u \).
  - v has spent \( s_v \) out of \( w_v \) so far. Let \( \Delta_v = w_v - s_v \).
  - Let \( \Delta = \min\{\Delta_u, \Delta_v\} \).
  - Set the price of \( y_e \) to \( \Delta \) and add \( \Delta \) to both \( s_u \) and \( s_v \).
  - A vertex that has now spent all its wealth is added to \( S \), which means at least one (or possibly both) of \( u \) and \( v \) is added to \( S \).

- **Return** \( S \) when all edges have been considered.

- The returned set \( S \) is a vertex cover. **Why?**

- The running time is linear (constant time per edge).
The algorithm

- Initialise:  
  - \( S = \emptyset \) (start with an empty vertex set)  
  - \( s_v = 0 \ \forall v \in V \). This is how much a vertex has spent.

- For all \( e \in E \):
  - \( u \) has spent \( s_u \) out of \( w_u \) so far. Let \( \Delta_u = w_u - s_u \).
  - \( v \) has spent \( s_v \) out of \( w_v \) so far. Let \( \Delta_v = w_v - s_v \).
  - Let \( \Delta = \min\{\Delta_u, \Delta_v\} \).
  - Set the price of \( y_e \) to \( \Delta \) and add \( \Delta \) to both \( s_u \) and \( s_v \).
  - A vertex that has now spent all its wealth is added to \( S \), which means at least one (or possibly both) of \( u \) and \( v \) is added to \( S \).

- Return \( S \) when all edges have been considered.

- The returned set \( S \) is a vertex cover. Why?
- The running time is linear (constant time per edge).
- What is the approximation factor?
The approximation factor
The approximation factor

- Let $OPT$ be the optimum vertex cover.
- Run the algorithm on the previous slide. Once it has stopped, the weight of $S$ is

\[
\sum_{v \in S} w_v 
\]
The approximation factor

- Let OPT be the optimum vertex cover.

- Run the algorithm on the previous slide. Once it has stopped, the weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in S} s_v$$

By construction of $S$
The approximation factor

- Let OPT be the optimum vertex cover.
- Run the algorithm on the previous slide. Once it has stopped, the weight of $S$ is

$$
\sum_{v \in S} w_v = \sum_{v \in S} s_v \leq \sum_{v \in V} s_v
$$

By construction of $S$
The approximation factor

Let $OPT$ be the optimum vertex cover.

Run the algorithm on the previous slide. Once it has stopped, the weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in S} s_v \leq \sum_{v \in V} s_v = 2 \sum_{e \in E} y_e$$

By construction of $S$

The price $y_e$ is paid for by both endpoints of $e$. 

![Diagram of a vertex $u$ connected by an edge $e$ to another vertex $v$.]
The approximation factor

- Let $\text{OPT}$ be the optimum vertex cover.
- Run the algorithm on the previous slide. Once it has stopped, the weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in S} s_v \leq \sum_{v \in V} s_v = 2 \sum_{e \in E} y_e \leq 2 \sum_{v \in \text{OPT}} w_v.$$ 

By construction of $S$

The price $y_e$ is paid for by both endpoints of $e$.

Recall that the combined wealth of the vertices in any vertex cover must be at least $\sum_{e \in E} y_e$. 

\[ e \quad u \quad v \]
The approximation factor

Let OPT be the optimum vertex cover.

Run the algorithm on the previous slide. Once it has stopped, the weight of $S$ is

$$\sum_{v \in S} w_v = \sum_{v \in S} s_v \leq \sum_{v \in V} s_v = 2 \sum_{e \in E} y_e \leq 2 \sum_{v \in OPT} w_v.$$ 

By construction of $S$  

The price $y_e$ is paid for by both endpoints of $e$.

Recall that the combined wealth of the vertices in any vertex cover must be at least $\sum_{e \in E} y_e$.

Thus, the approximation factor is 2.

**Theorem**

There is a linear time 2-approximation for weighted vertex cover.