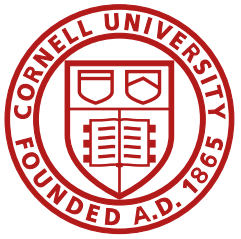
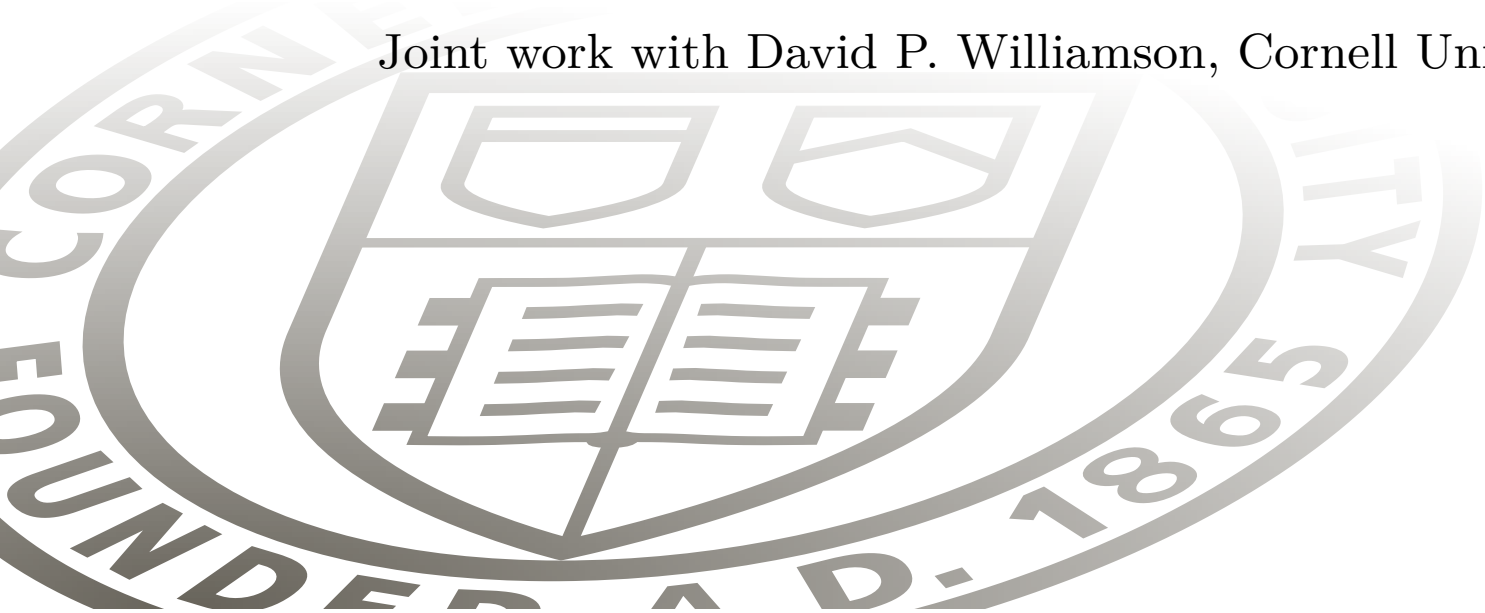


A $\frac{3}{2}$-Approximation Algorithm for Some Minimum-Cost Graph Problems

James M. Davis
Cornell University

Joint work with David P. Williamson, Cornell University



A Simple Graph Problem

MIN WCF(k)

Input:

- Undirected graph $G = (V, E)$
- Edge costs $c(e) \geq 0$ for all $e \in E$
- Positive integer k

Goal: Find a minimum-cost forest F such that each component has at least k vertices.

A Simple Graph Problem

MIN WCF(k)

Input:

- Undirected graph $G = (V, E)$
- Edge costs $c(e) \geq 0$ for all $e \in E$
- Positive integer k

Goal: Find a minimum-cost forest F such that each component has at least k vertices.

$k = 2 \Rightarrow$ minimum edge-cover problem

A Simple Graph Problem

MIN WCF(k)
Input:

- Undirected graph $G = (V, E)$
- Edge costs $c(e) \geq 0$ for all $e \in E$
- Positive integer k

Goal: Find a minimum-cost forest F such that each component has at least k vertices.

$k = 2 \Rightarrow$ minimum edge-cover problem
$k = n \Rightarrow$ minimum spanning tree problem

A Simple Graph Problem

MIN WCF(k)
Input:

- Undirected graph $G = (V, E)$
- Edge costs $c(e) \geq 0$ for all $e \in E$
- Positive integer k

Goal: Find a minimum-cost forest F such that each component has at least k vertices.

$k = 2 \Rightarrow$ minimum edge-cover problem
$k = n \Rightarrow$ minimum spanning tree problem
$k \geq 3$, constant $\Rightarrow$ NP-hard (Imielińska, Khachiyan, Kalantari 1993, Bazgan, Couëtoux, Tuza, 2011)

A 2-Approximation Algorithm

A component is *small* if $< k$ vertices, *big* otherwise.

$F \leftarrow \emptyset$
while F is not a feasible solution **do**
    Let e be the cheapest edge joining two comps C_1, C_2, at least one small
    $F \leftarrow F \cup \{e\}$
Return F

Due to Imielińska, Khachiyan, Kalantari 1993.

A Generalization

- Goemans and Williamson 1994 use functions $h : 2^V \to \{0, 1\}$
- Want min-cost edges F with $|\delta(S) \cap F| \geq h(S), \forall S \subset V$
- $\delta(S)$ is set of edges with exactly one endpoint in S

A Generalization

- Goemans and Williamson 1994 use functions $h : 2^V \to \{0, 1\}$
- Want min-cost edges F with $|\delta(S) \cap F| \geq h(S), \forall S \subset V$
- $\delta(S)$ is set of edges with exactly one endpoint in S

Easy change: small $C \Rightarrow h(C) = 1$, large $C \Rightarrow h(C) = 0$.

$F \leftarrow \emptyset$
while F is not a feasible solution **do**
    Let e be the cheapest edge joining two comps C_1, C_2, at least one small
    $F \leftarrow F \cup \{e\}$
Return F

Other Problems

- Gives 2-approximation alg. if h is *downwards monotone*
- *Downwards monotone*: $h(T) = 1 \Rightarrow h(S) = 1$ for all $S \subseteq T$

MIN WCF(k):

- $h(S) = 1$ if $|S| < k$

Other Problems

- Gives 2-approximation alg. if h is *downwards monotone*
- *Downwards monotone*: $h(T) = 1 \Rightarrow h(S) = 1$ for all $S \subseteq T$

MIN WCF(k):

- $h(S) = 1$ if $|S| < k$

Another example:

- Depots $D \subseteq V$, cost $c(d)$
- Find min-cost edges F, depots D' where each component has at least k vertices, at least 1 open depot

Couëtoux's Algorithm for MIN WCF(k)

In 2011, Couëtoux gives a $\frac{3}{2}$-approximation algorithm for MIN WCF(k), simple modification of Imielińska et al. algorithm.

Main Idea:

Couëtoux's Algorithm for MIN WCF(k)

In 2011, Couëtoux gives a $\frac{3}{2}$-approximation algorithm for MIN WCF(k), simple modification of Imielińska et al. algorithm.

Main Idea:

- Each edge added reduces number of small components

Couëtoux's Algorithm for MIN WCF(k)

In 2011, Couëtoux gives a $\frac{3}{2}$-approximation algorithm for MIN WCF(k), simple modification of Imielińska et al. algorithm.

Main Idea:

- Each edge added reduces number of small components
- Should be willing to pay twice as much for edge that eliminates *two* small components

Couëtoux's Algorithm for MIN WCF(k)

In 2011, Couëtoux gives a $\frac{3}{2}$-approximation algorithm for MIN WCF(k), simple modification of Imielińska et al. algorithm.

Main Idea:

- Each edge added reduces number of small components
- Should be willing to pay twice as much for edge that eliminates *two* small components
- e is *good* if it connects two small components and the result is a large component

Couëtoux's Algorithm for MIN WCF(k)

In 2011, Couëtoux gives a $\frac{3}{2}$-approximation algorithm for MIN WCF(k), simple modification of Imielińska et al. algorithm.

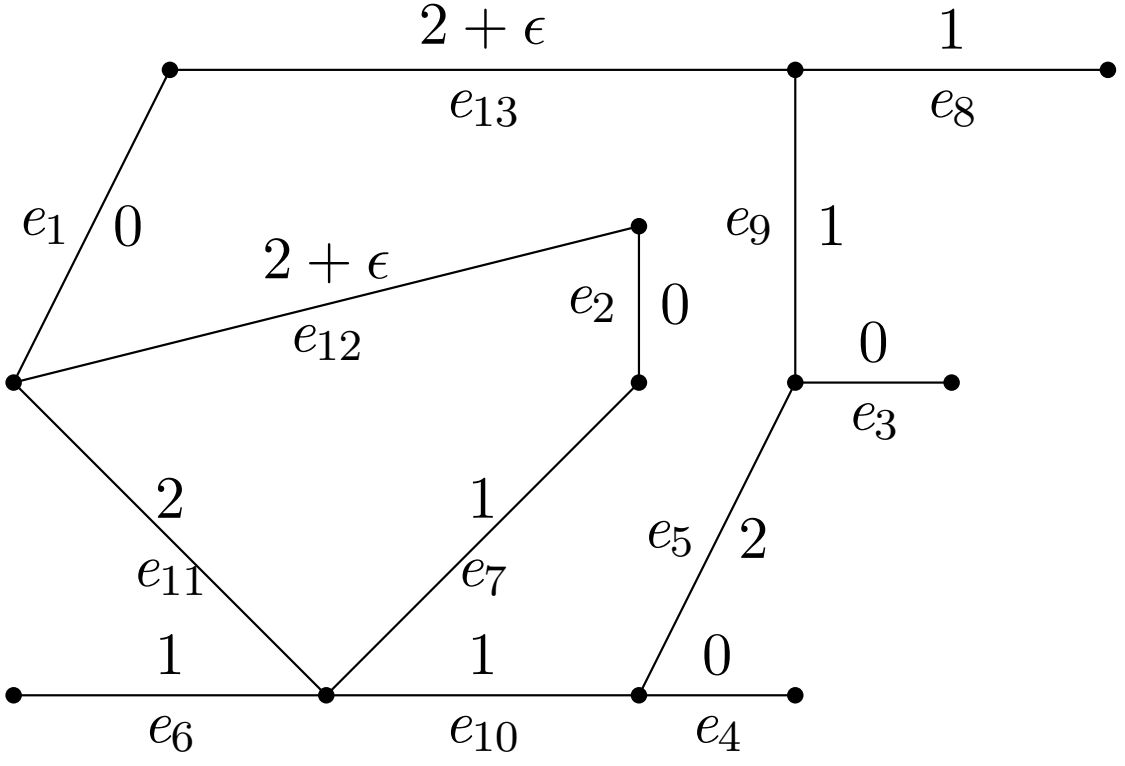
Main Idea:

- Each edge added reduces number of small components
- Should be willing to pay twice as much for edge that eliminates *two* small components
- e is *good* if it connects two small components and the result is a large component
- e is *bad* if it eliminates just one small component

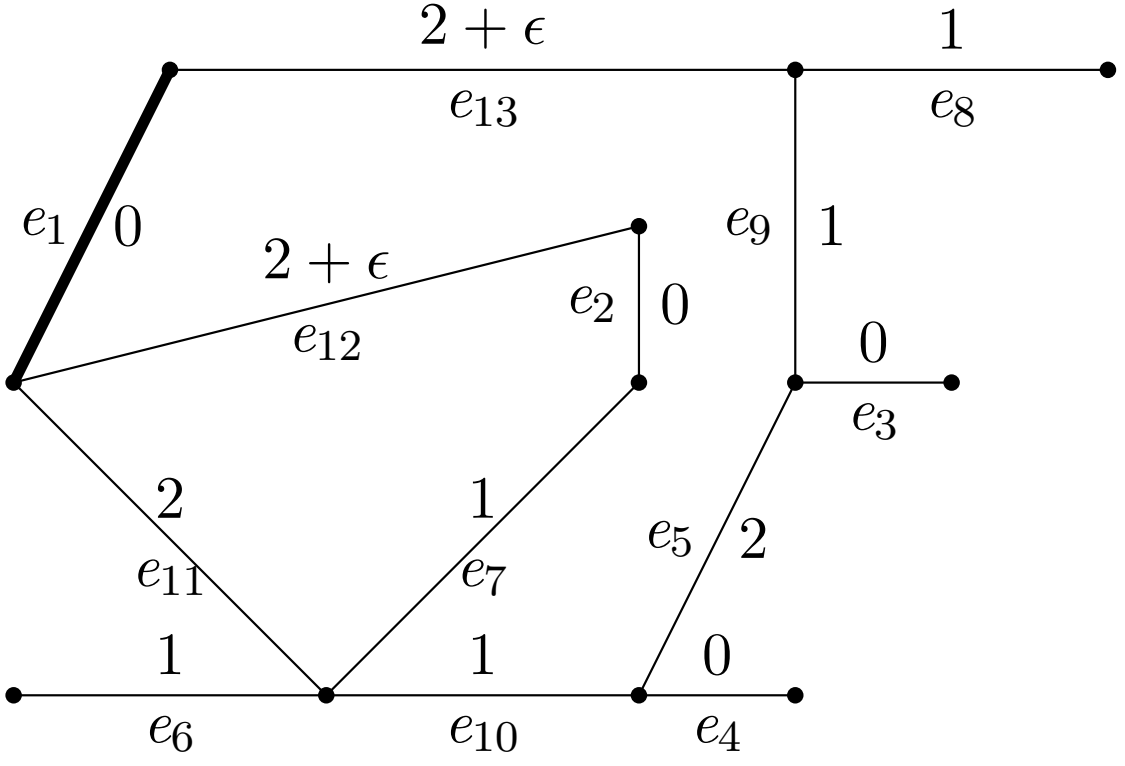
Couëtoux's Algorithm

$F \leftarrow \emptyset$
while F is not a feasible solution **do**
    Let e be the cheapest good edge (if such an edge exists); joins two small comps into a large comp
    Let e' be the cheapest bad edge; joins two comps, at least one small
    if good e exists and $c(e) \leq 2c(e')$ **then**
        $F \leftarrow F \cup \{e\}$
    else
        $F \leftarrow F \cup \{e'\}$
Return F

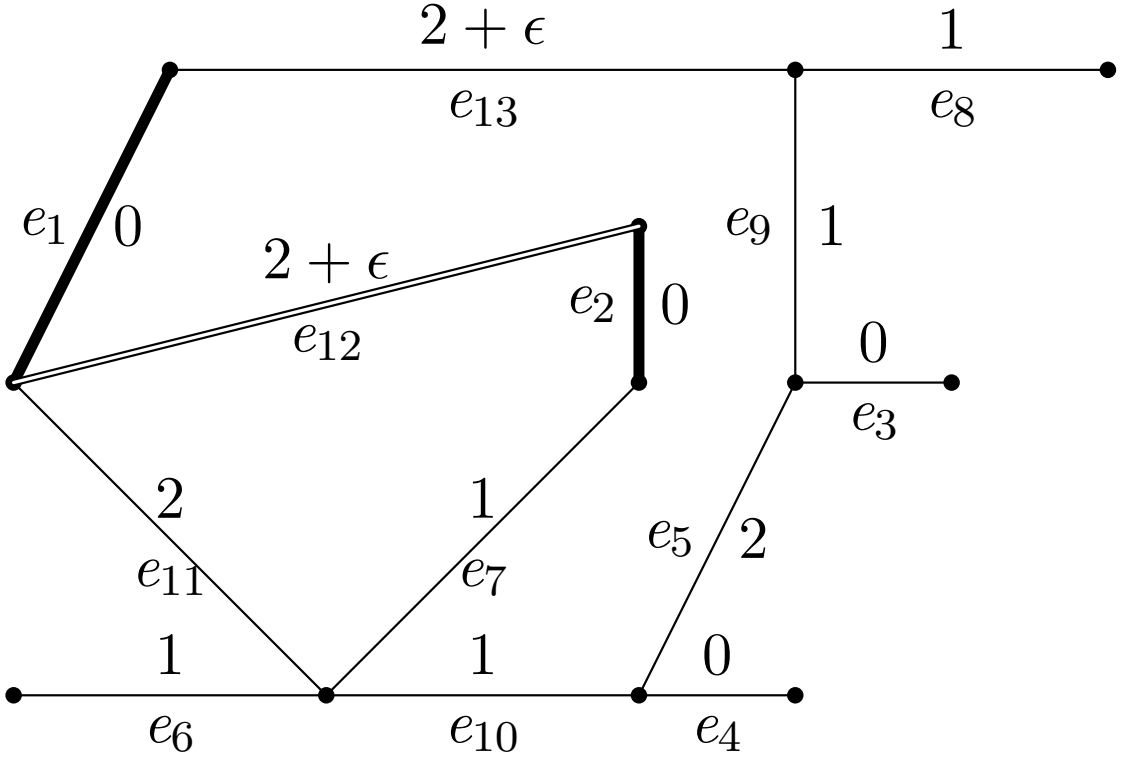
Example for $k = 4$



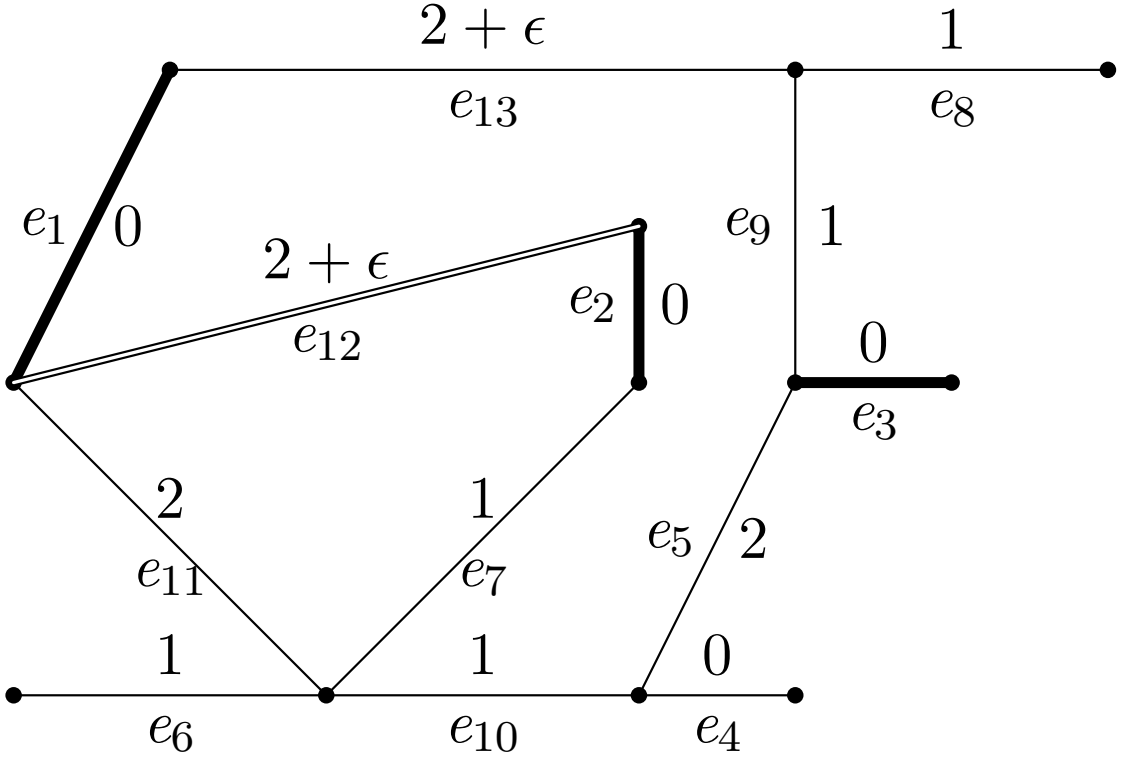
Example for $k = 4$



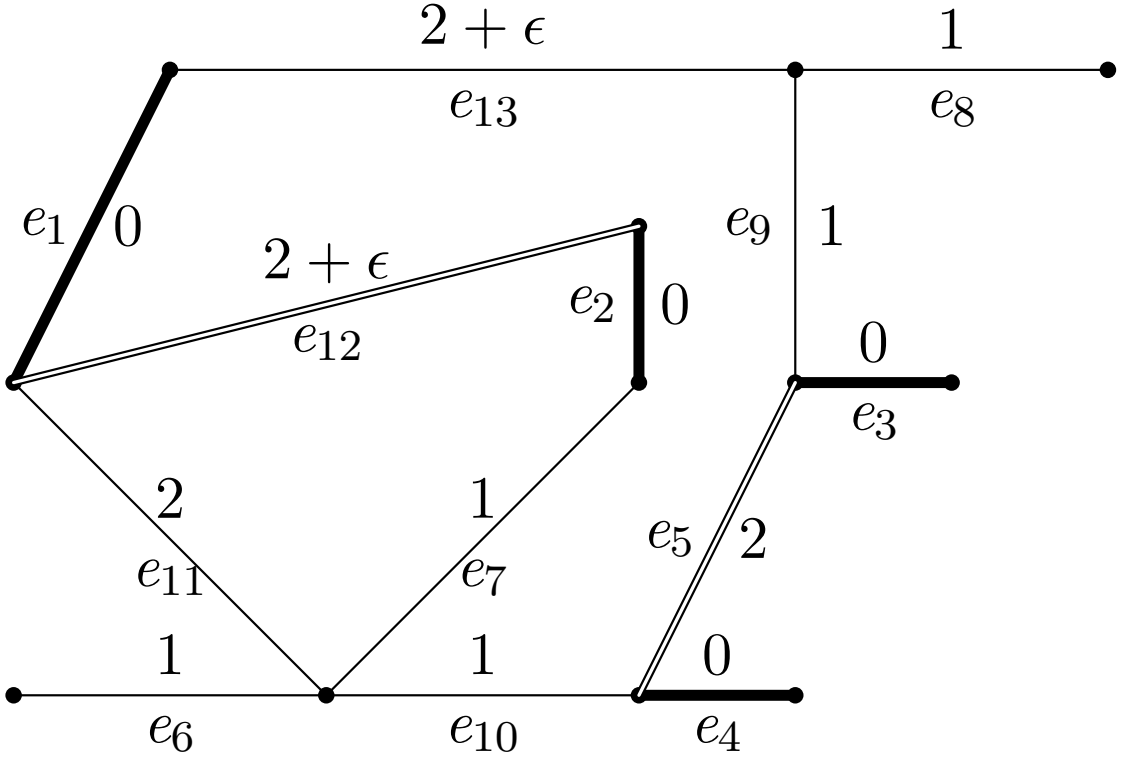
Example for $k = 4$



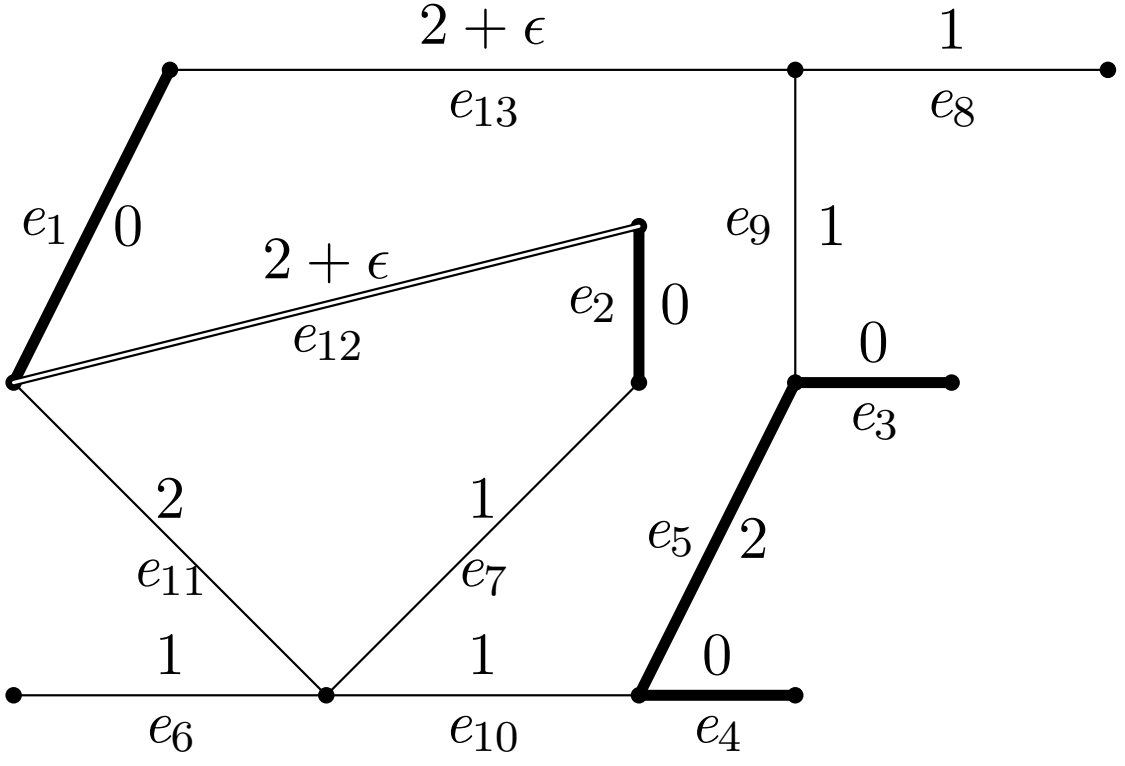
Example for $k = 4$



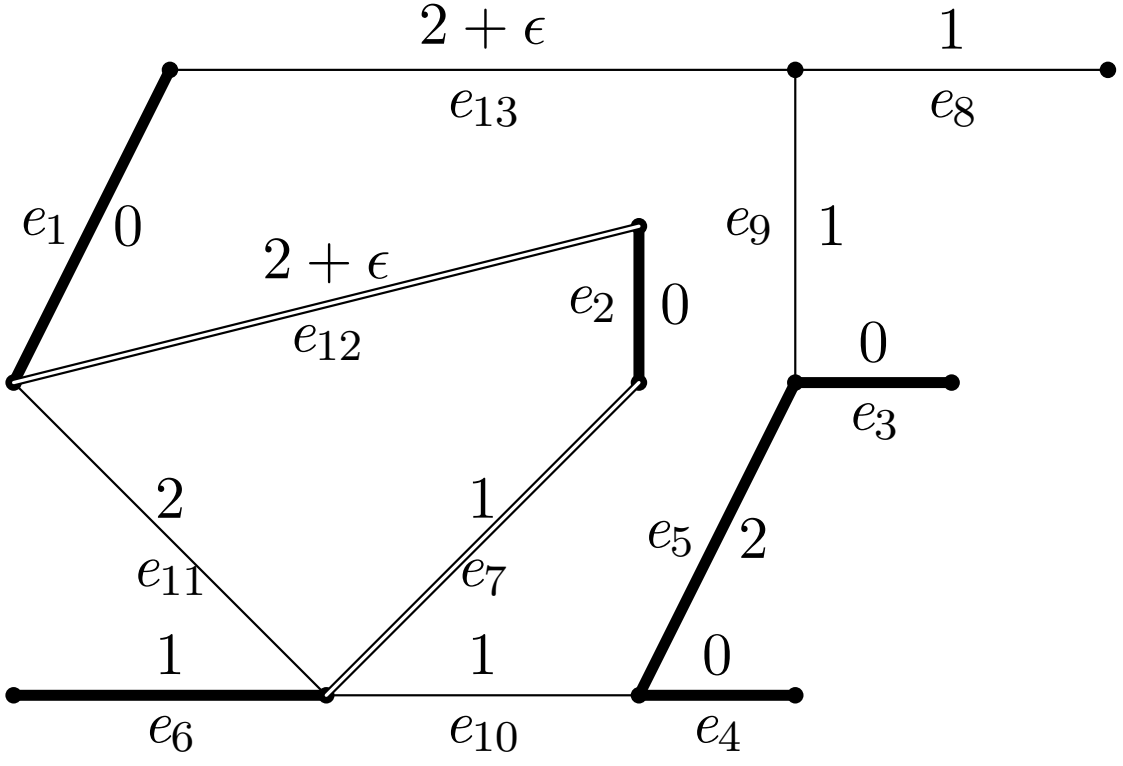
Example for $k = 4$



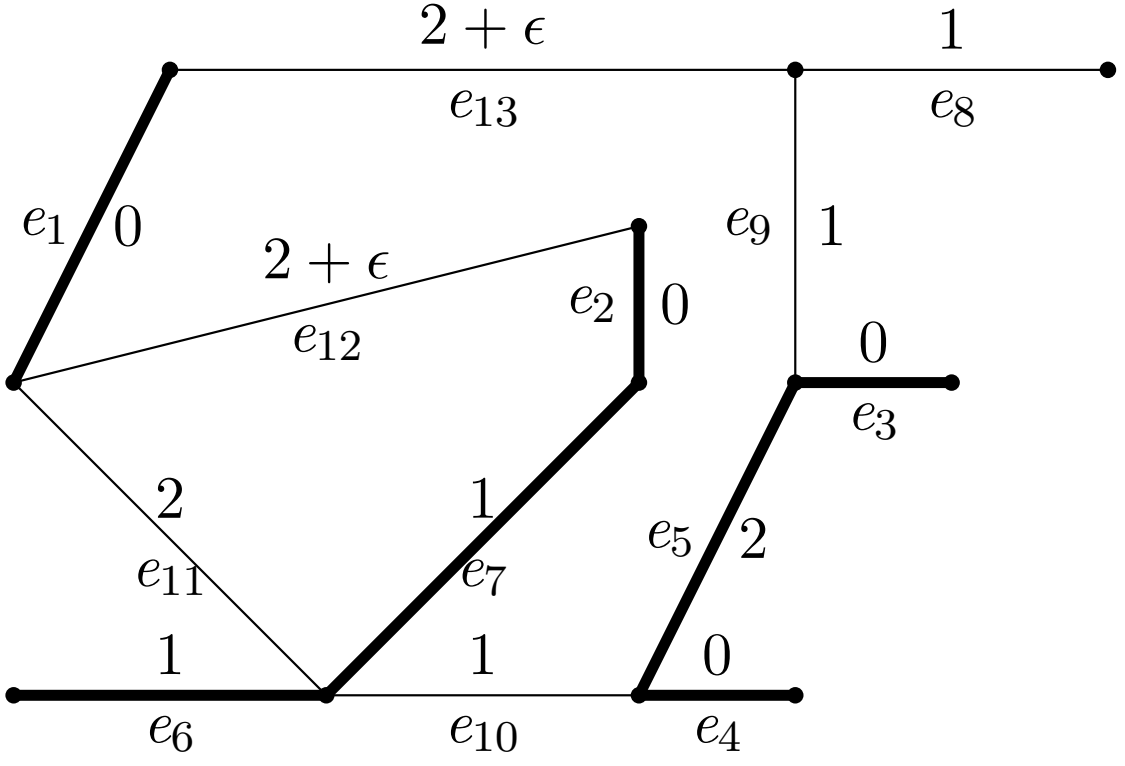
Example for $k = 4$



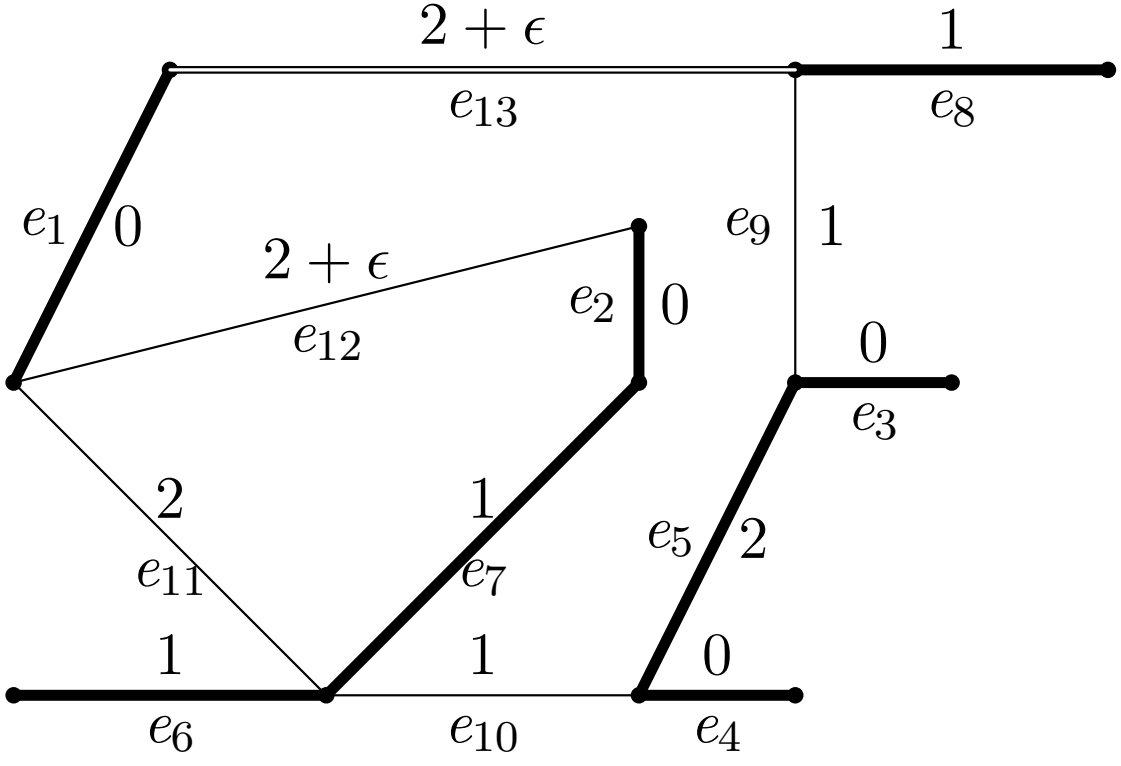
Example for $k = 4$



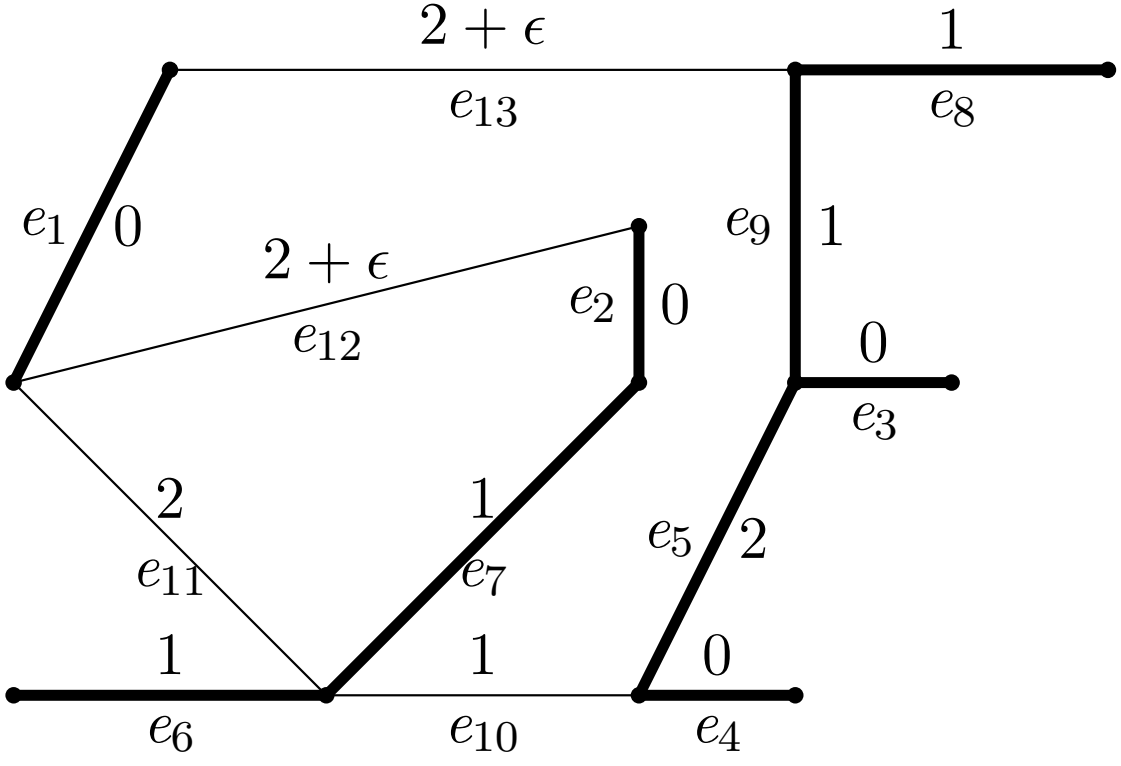
Example for $k = 4$



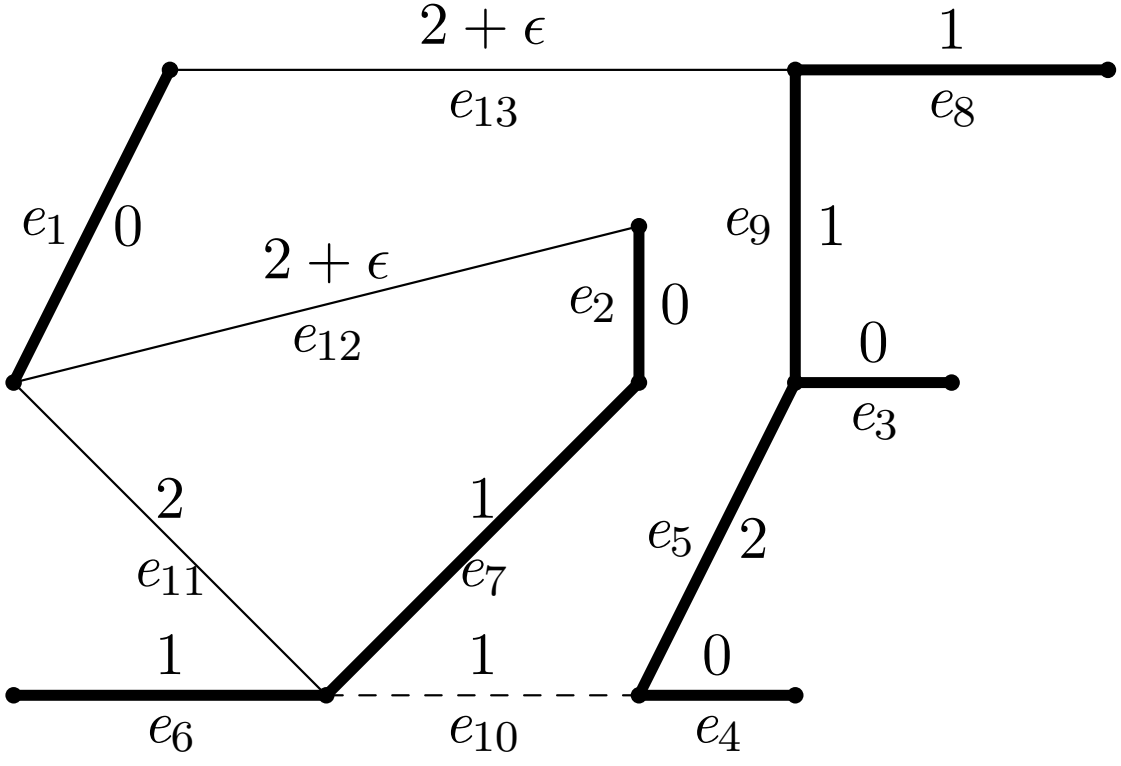
Example for $k = 4$



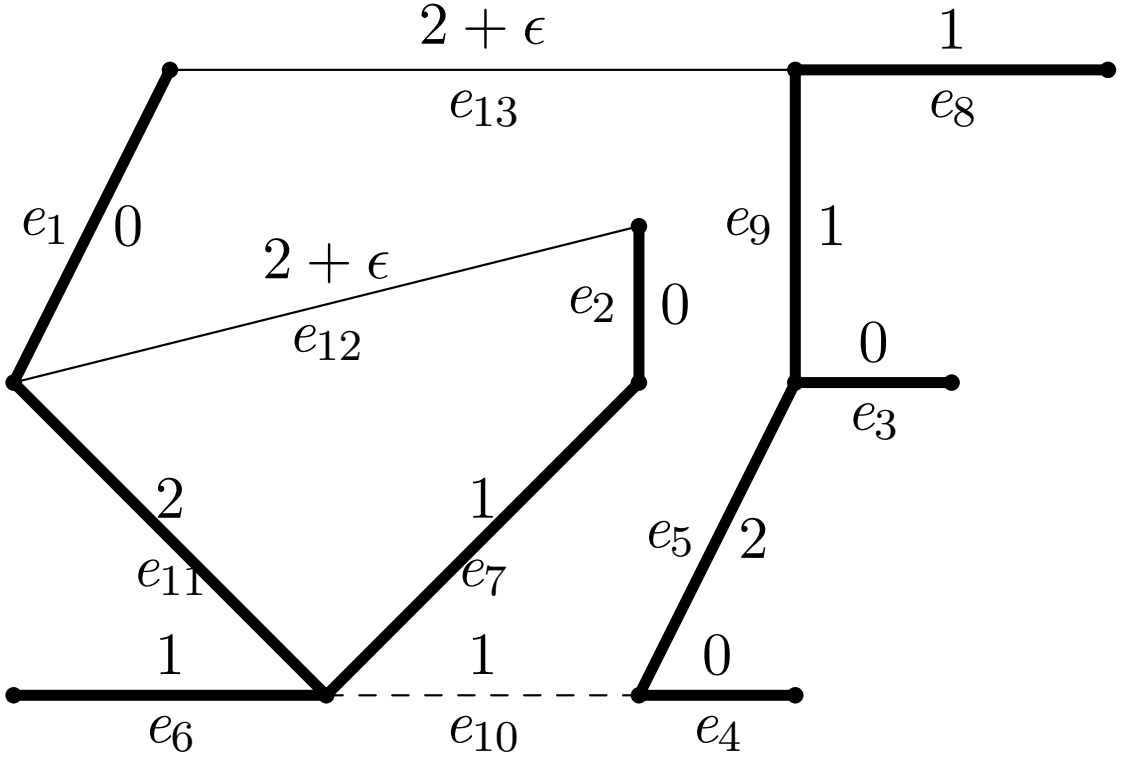
Example for $k = 4$



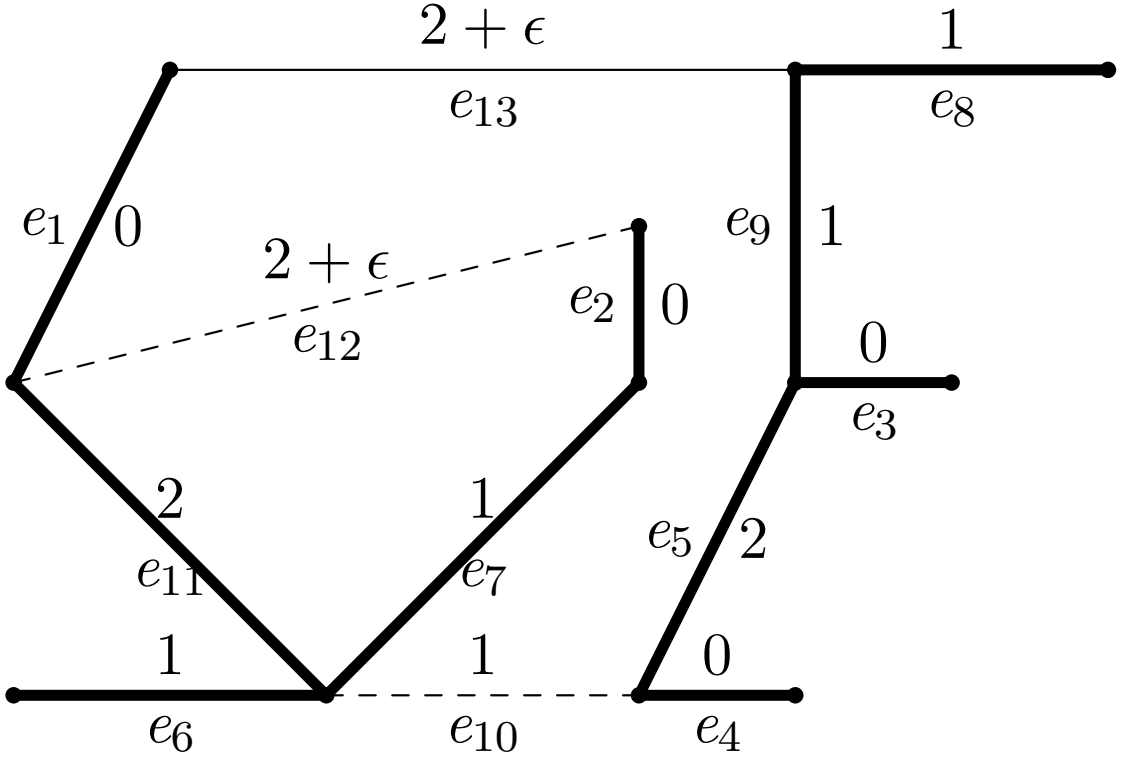
Example for $k = 4$



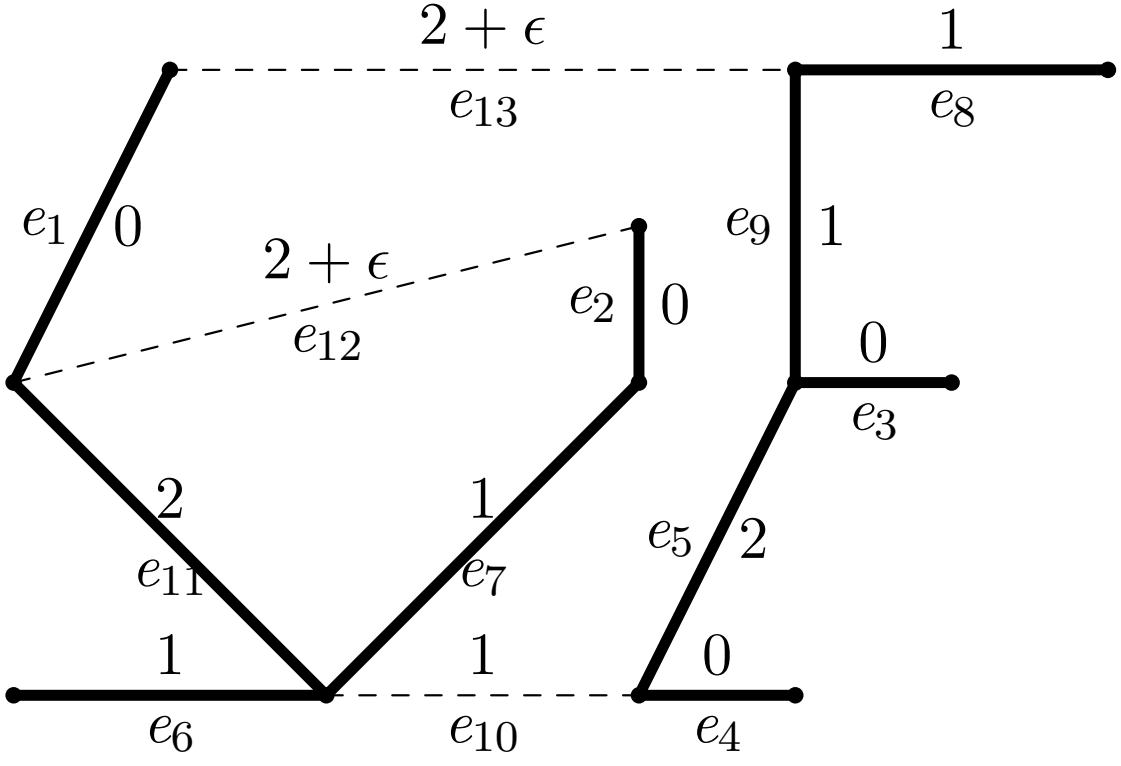
Example for $k = 4$



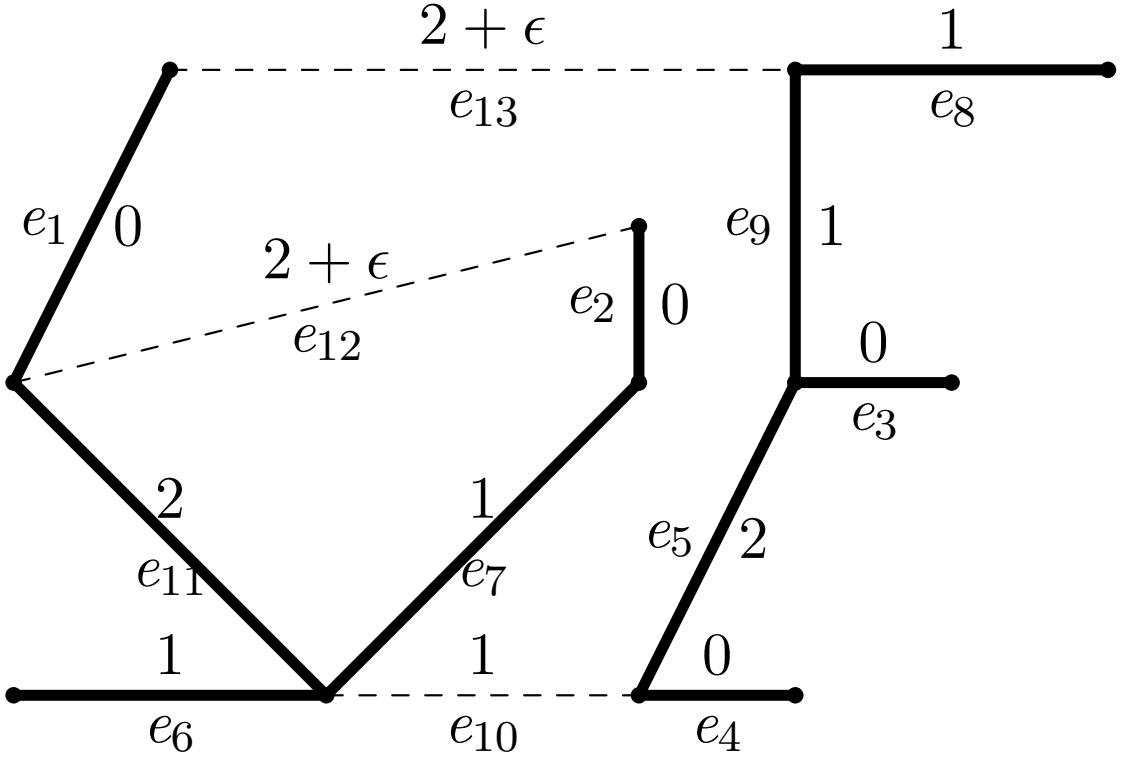
Example for $k = 4$



Example for $k = 4$

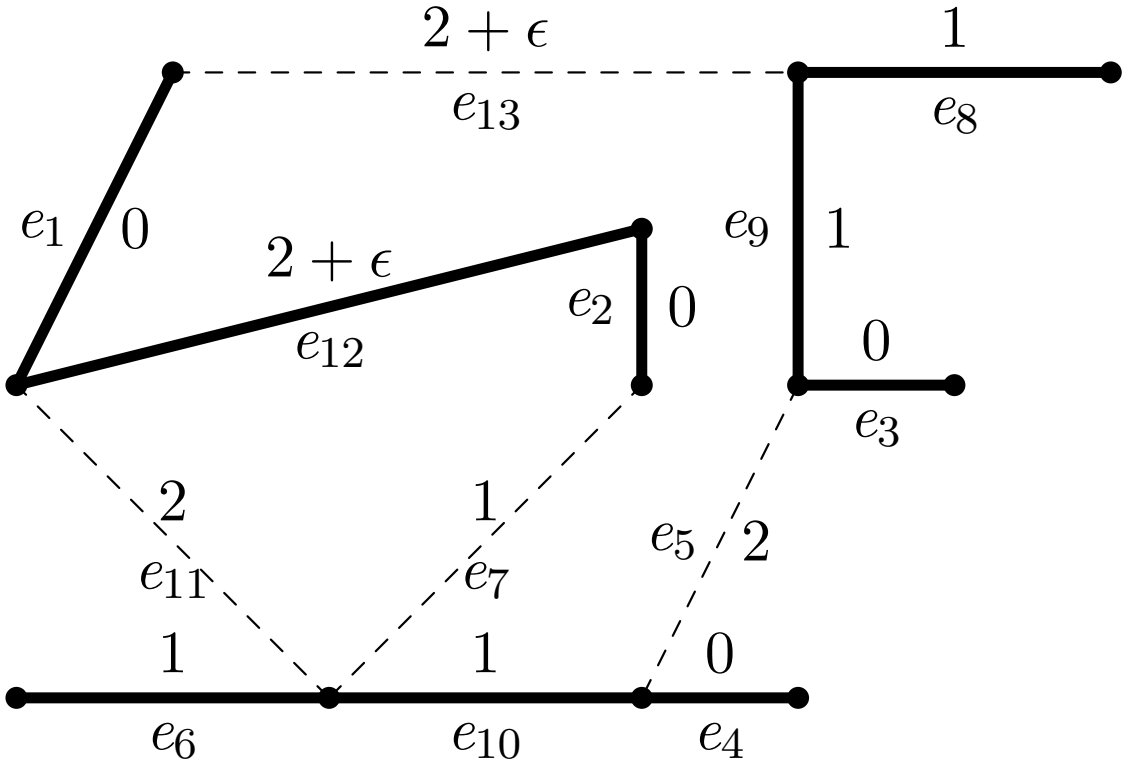


Example for $k = 4$



Cost is 8.

Optimal Solution



Optimal cost is $6+\epsilon$.

Our Contributions

- Extend Couëtoux's algorithm to downwards monotone functions
  - Easy: small $C \Rightarrow h(C) = 1$, large $C \Rightarrow h(C) = 0$
- We simplify the overall analysis (harder)

Our Contributions

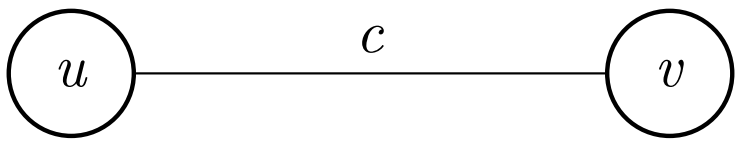
- Extend Couëtoux's algorithm to downwards monotone functions
  - Easy: small $C \Rightarrow h(C) = 1$, large $C \Rightarrow h(C) = 0$
- We simplify the overall analysis (harder)

Main idea: Generate a "dual" solution of value at least cost of algorithm's solution. Show that 2/3 of dual solution is a lower bound on any feasible solution.

Ideas of the analysis

For simplicity, assume there is no vertex v such that $h(\{v\}) = 0$

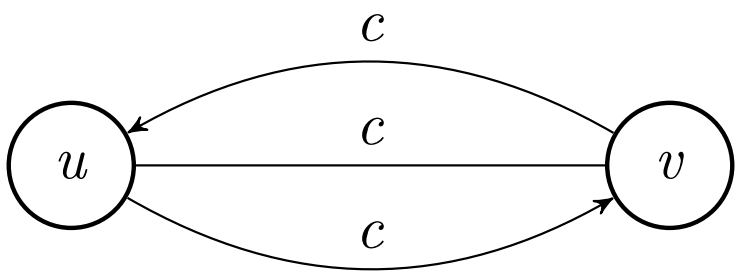
First, make a mixed graph by adding arcs for every edge



Ideas of the analysis

For simplicity, assume there is no vertex v such that $h(\{v\}) = 0$

First, make a mixed graph by adding arcs for every edge



Birooted Components

Observation:

Birooted Components

Observation:

- Every large component first became large when two small components were joined together

Birooted Components

Observation:

- Every large component first became large when two small components were joined together
- Every component has exactly one good edge

Birooted Components

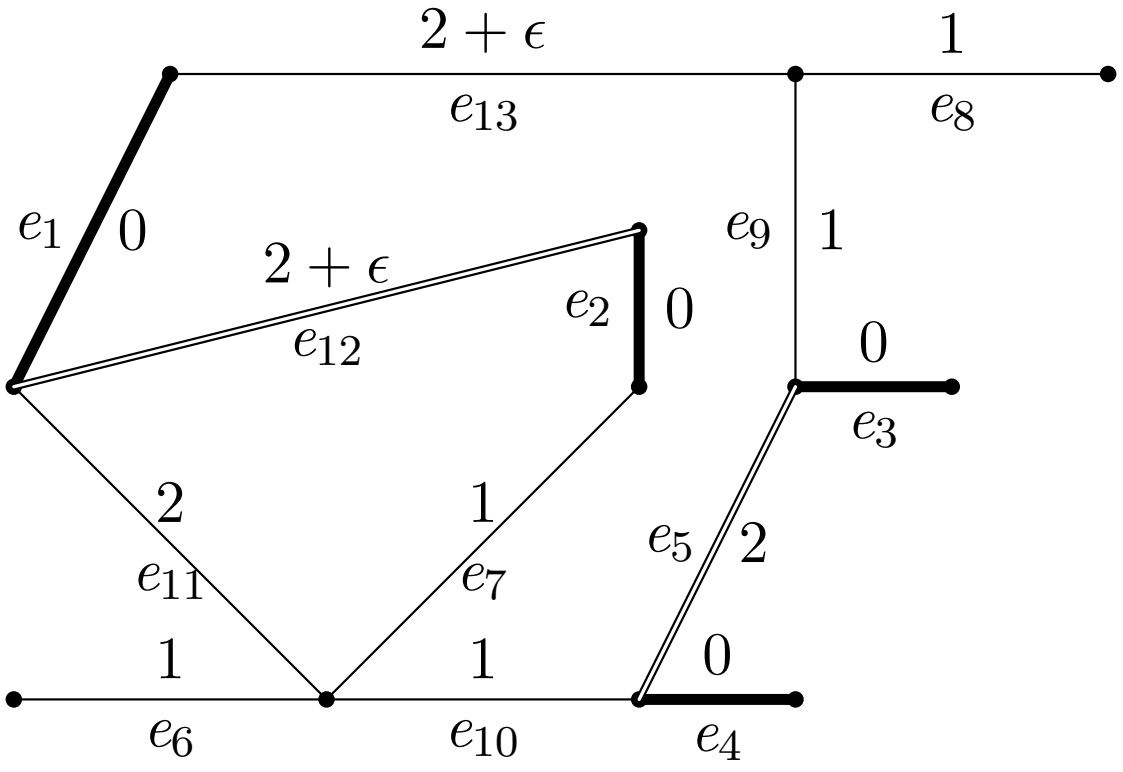
Observation:

- Every large component first became large when two small components were joined together
- Every component has exactly one good edge

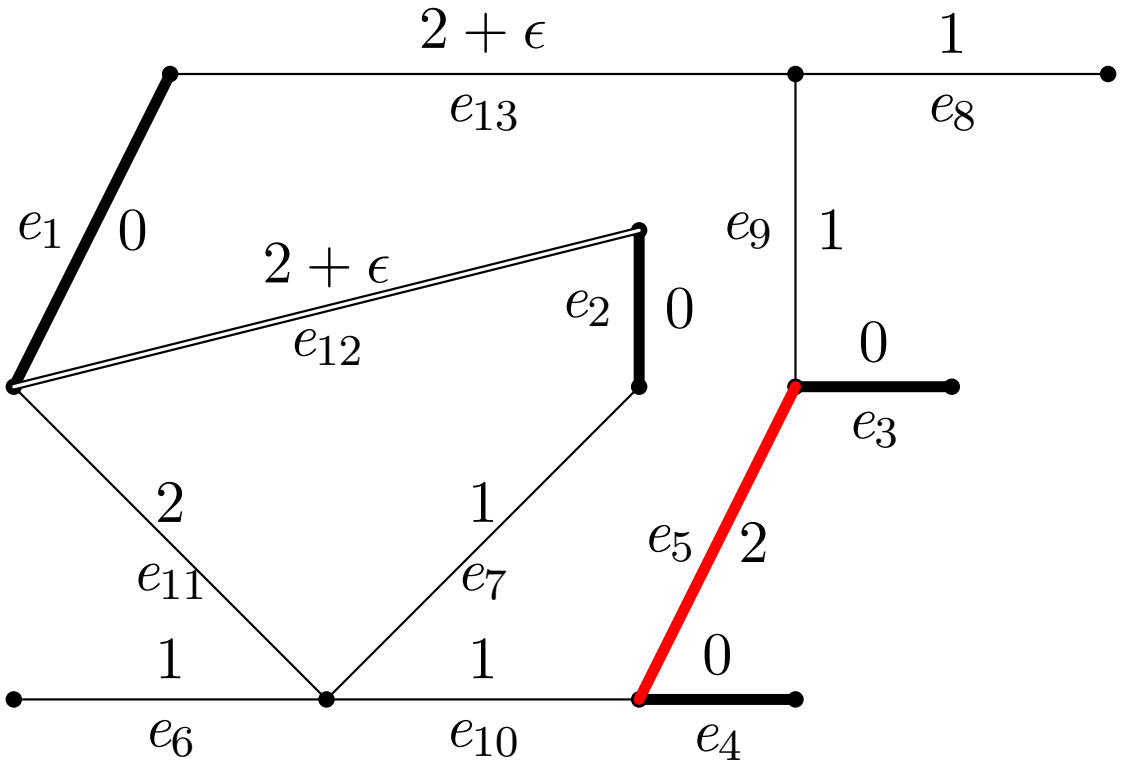
Constructing *birooted* components:

- Good edge is undirected
- All other edges in component directed towards the two endpoints of the good edge

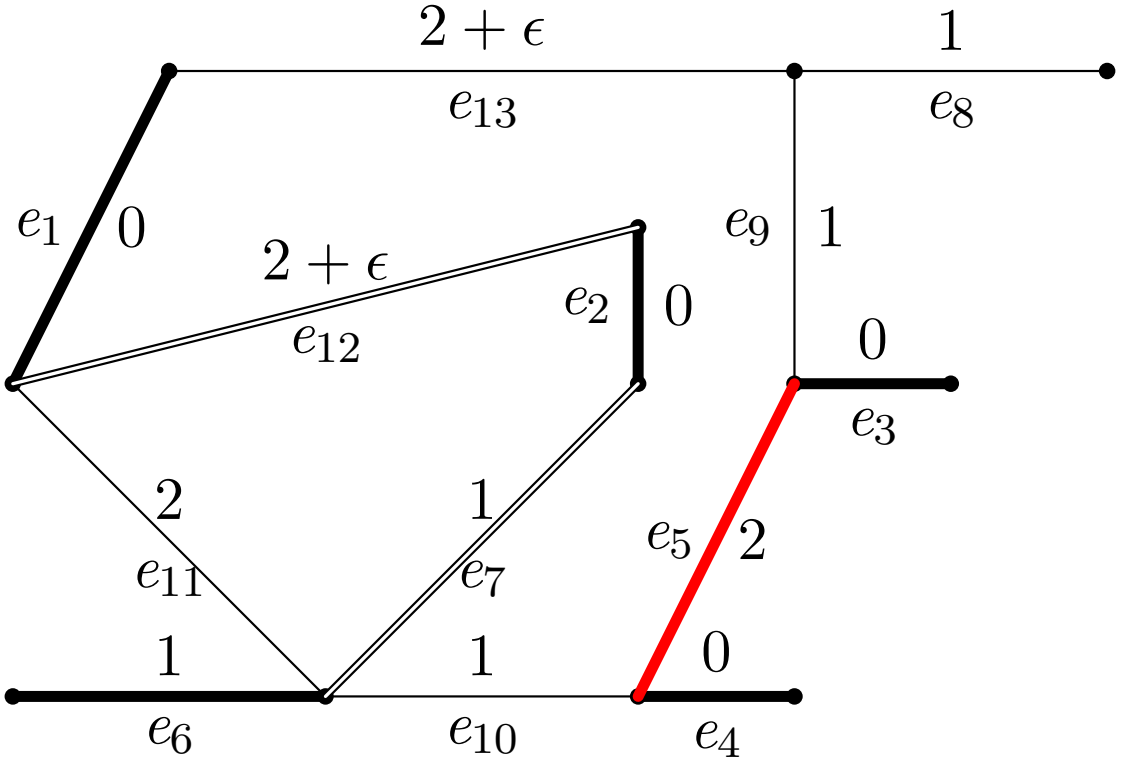
Back to the Example



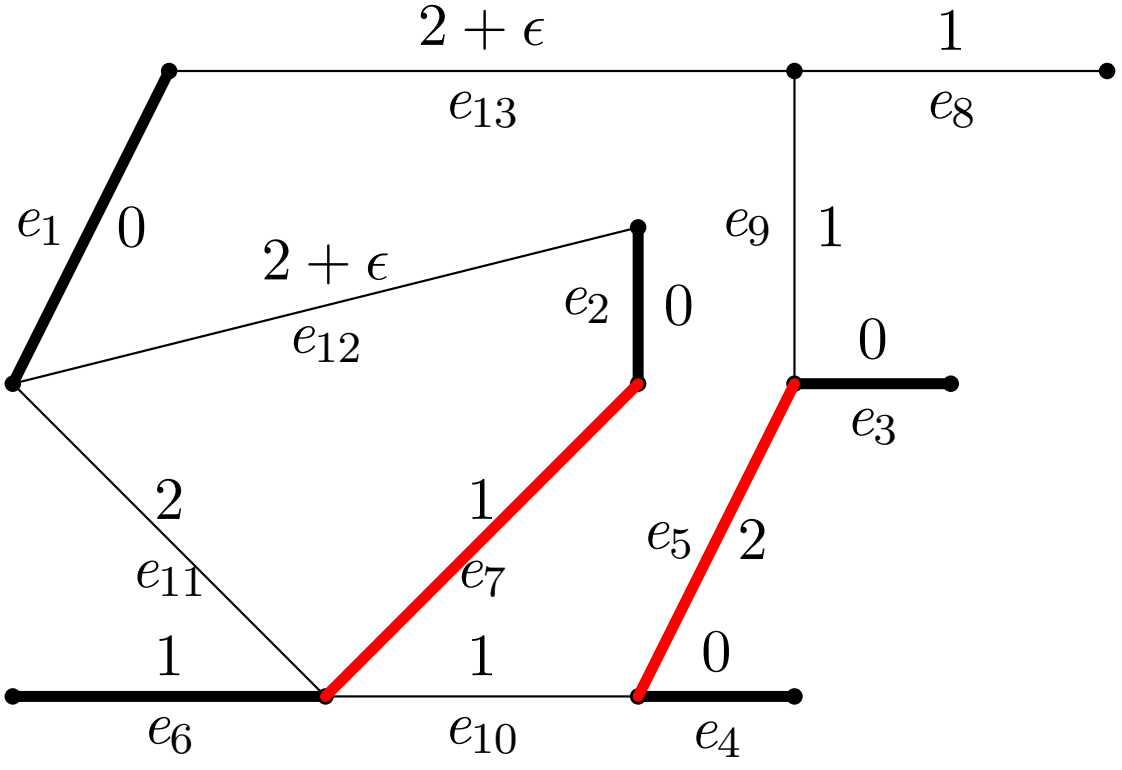
Back to the Example



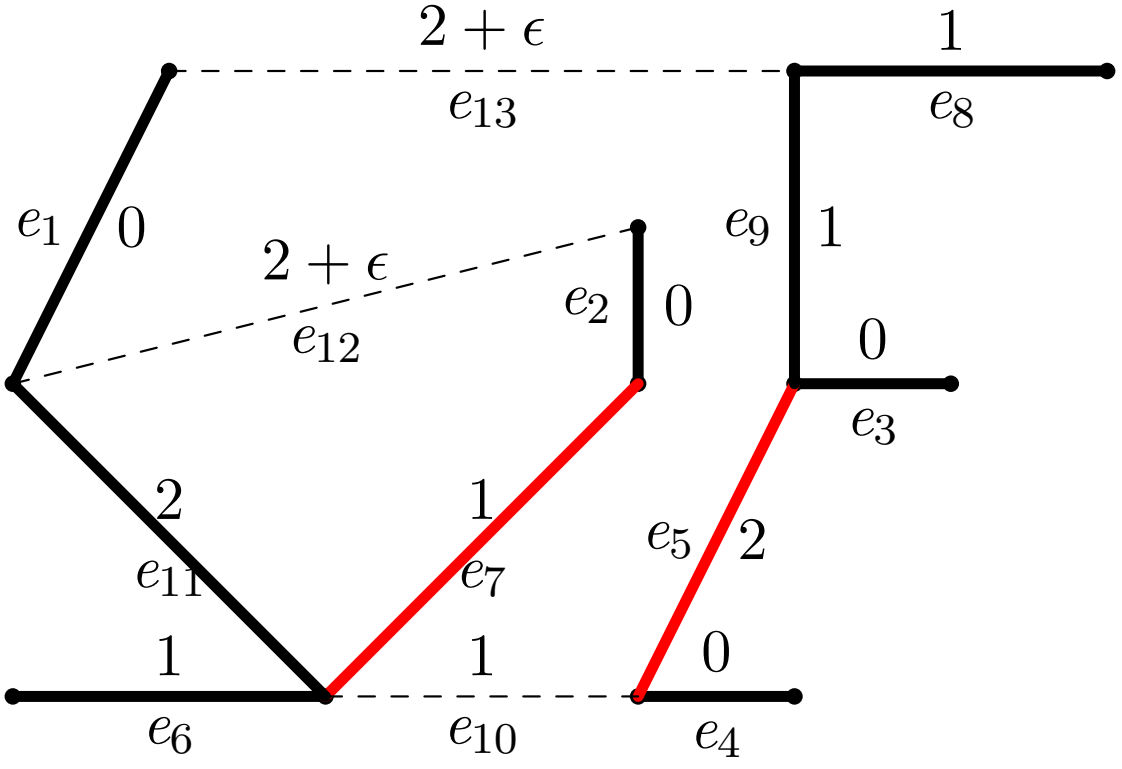
Back to the Example



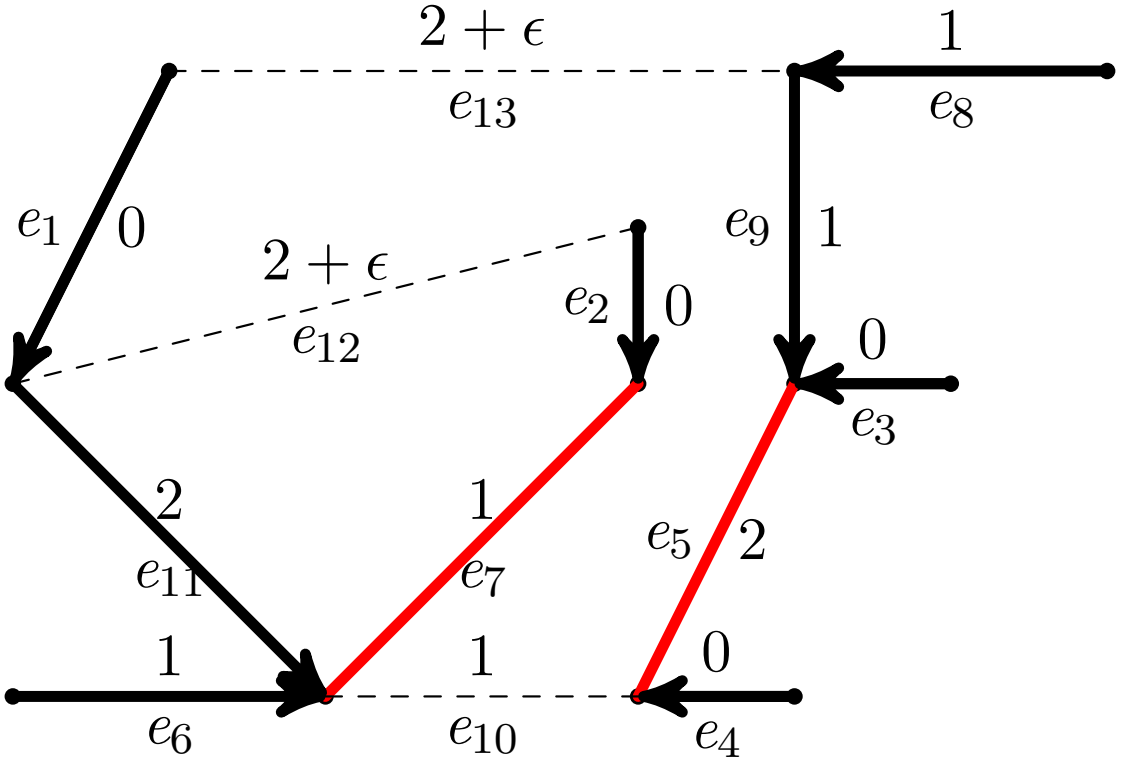
Back to the Example



Back to the Example



Back to the Example



"Duals"

Introduce dual variables $y(S)$ for each $S \subseteq V$

"Duals"

Introduce dual variables $y(S)$ for each $S \subseteq V$

- Duals satisfy an arc constraint for all arcs a:

$$\sum_{S : a \in \delta^+(S)} y(S) \leq c(a)$$

- $\delta^+(S)$ are arcs out of S

"Duals"

Introduce dual variables $y(S)$ for each $S \subseteq V$

- Duals satisfy an arc constraint for all arcs a:

$$\sum_{S : a \in \delta^+(S)} y(S) \leq c(a)$$

- $\delta^+(S)$ are arcs out of S
- Inequality may be violated for *edges* e; may have

$$\sum_{S : e \in \delta(S)} y(S) > c(e)$$

Duals Upper Bound Algorithm's Cost

Lemma

For birooted component C constructed by the algorithm, can show that cost of C at most

$$\sum_{S \subseteq C} y(S).$$

- *For each arc a in C, $\sum_{S:a \in \delta^+(S)} y(S) = c(a)$.*
- *For good edge e, $\sum_{e \in \delta(S)} y(S) \geq c(e)$.*

Key Lemma

Say that a component $C \in \delta(S)$ if some edge of C is in $\delta(S)$.

Lemma

For any feasible solution F^*, and any component C^* of F^*,

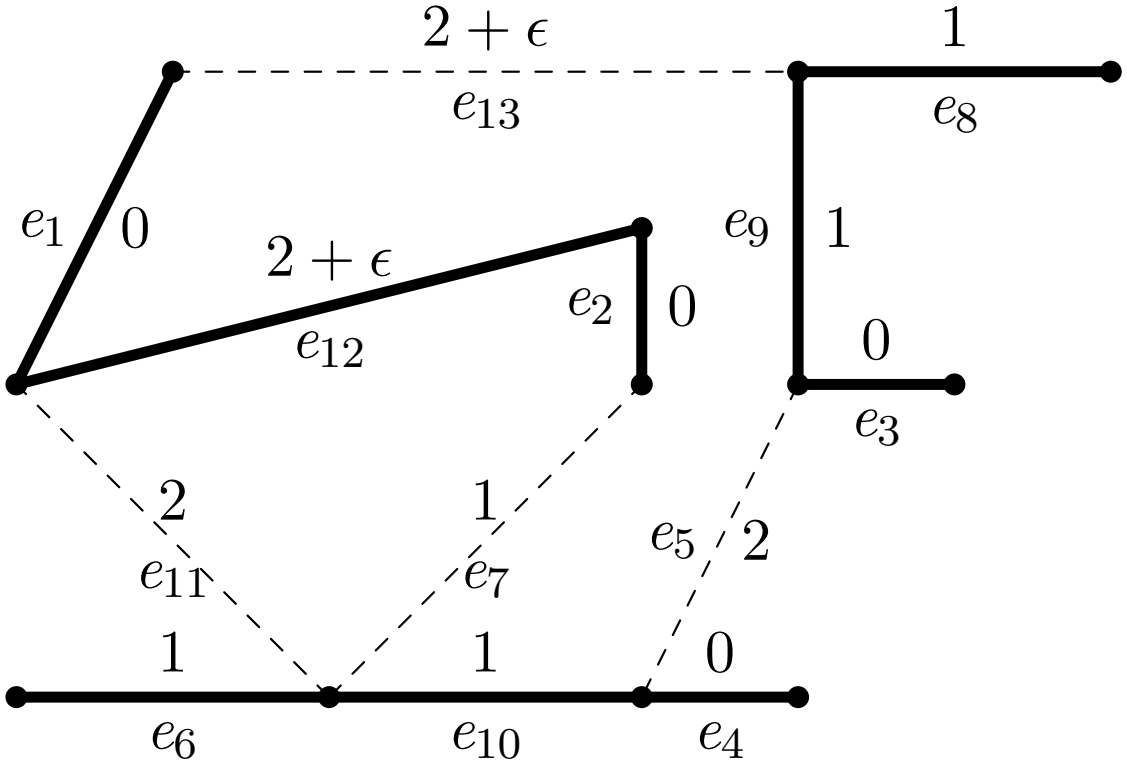
$$\sum_{S : C^* \in \delta(S)} y(S) \leq \frac{3}{2} \sum_{e \in C^*} c(e).$$

Then let F^* be an optimal solution, $\mathcal{C}^*$ its components, F the algorithm's solution. Then

$$\begin{aligned}
\sum_{e \in F} c(e) \leq \sum_{S} y(S) &\leq \sum_{C^* \in \mathcal{C}^*} \sum_{S : C^* \in \delta(S)} y(S) \\
&\leq \sum_{C^* \in \mathcal{C}^*} \left(\frac{3}{2} \sum_{e \in C^*} c(e) \right) \\
&= \frac{3}{2} \sum_{e \in F^*} c(e).
\end{aligned}$$

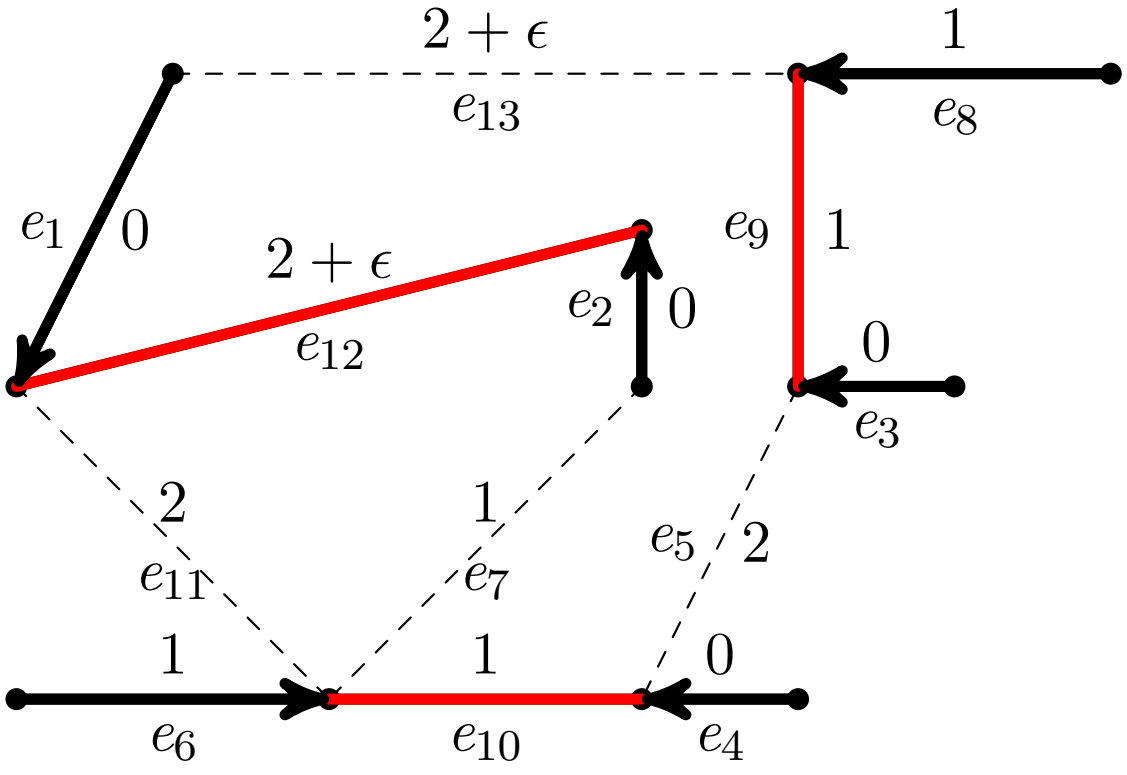
Proof Ideas for Key Lemma

For each component C^* of solution F^*, identify a good edge and biroot the component.



Proof Ideas for Key Lemma

For each component C^* of solution F^*, identify a good edge and biroot the component.



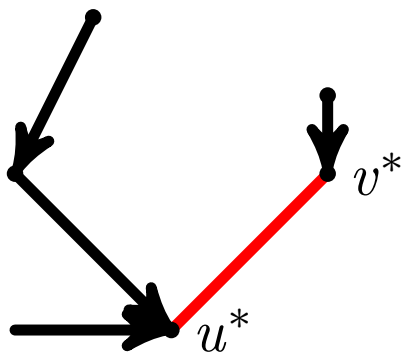
Proof Ideas for Key Lemma cont.

To prove:

$$\sum_{S: C^* \in \delta(S)} y(S) \leq \frac{3}{2} \sum_{e \in C^*} c(e).$$

Let $e^* = (u^*, v^*)$ be good edge of birooted component.

Since $\sum_{S: a \in \delta^+(S)} y(S) \leq c(a)$ for each arc a in birooted component, only need to bound $\sum_{S: u^* \text{or } v^* \in S} y(S)$.



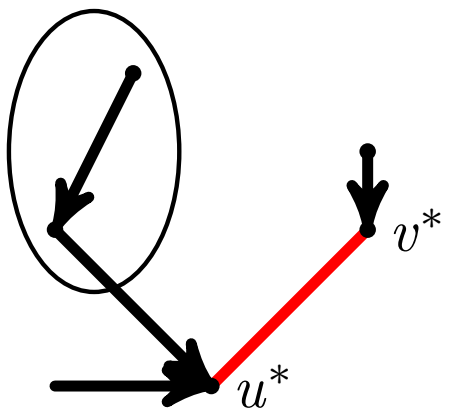
Proof Ideas for Key Lemma cont.

To prove:

$$\sum_{S: C^* \in \delta(S)} y(S) \leq \frac{3}{2} \sum_{e \in C^*} c(e).$$

Let $e^* = (u^*, v^*)$ be good edge of birooted component.

Since $\sum_{S: a \in \delta^+(S)} y(S) \leq c(a)$ for each arc a in birooted component, only need to bound $\sum_{S: u^* \text{or } v^* \in S} y(S)$.



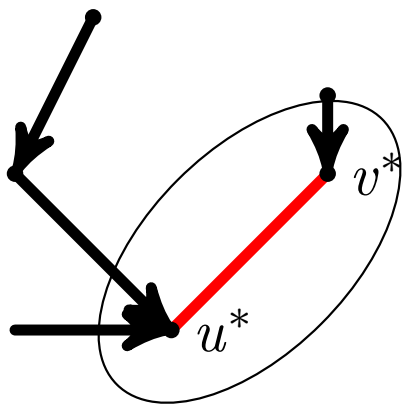
Proof Ideas for Key Lemma cont.

To prove:

$$\sum_{S: C^* \in \delta(S)} y(S) \leq \frac{3}{2} \sum_{e \in C^*} c(e).$$

Let $e^* = (u^*, v^*)$ be good edge of birooted component.

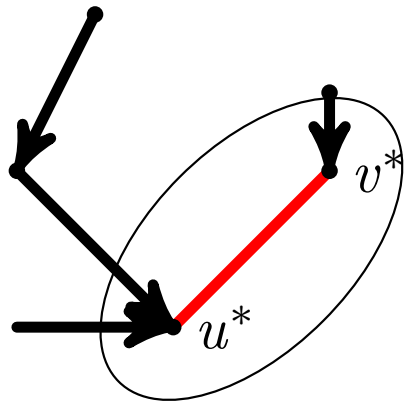
Since $\sum_{S: a \in \delta^+(S)} y(S) \leq c(a)$ for each arc a in birooted component, only need to bound $\sum_{S: u^* \text{or } v^* \in S} y(S)$.



Proof Ideas for Key Lemma cont.

To prove:

$$\sum_{S: C^* \in \delta(S)} y(S) \leq \frac{3}{2} \sum_{e \in C^*} c(e).$$

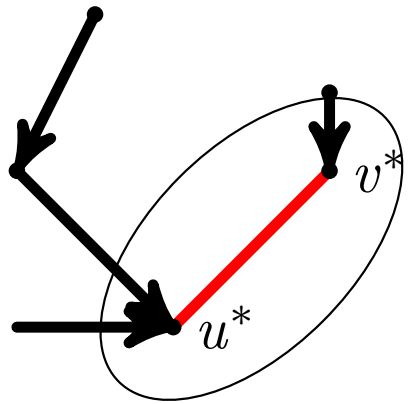


If $\sum_{S: u^* \text{or } v^* \in S} y(S) \leq \frac{3}{2} c(e^*)$, then done.

Proof Ideas for Key Lemma cont.

To prove:

$$\sum_{S: C^* \in \delta(S)} y(S) \leq \frac{3}{2} \sum_{e \in C^*} c(e).$$



If $\sum_{S: u^* \text{or } v^* \in S} y(S) \leq \frac{3}{2} c(e^*)$, then done.

Otherwise, C^* must have an edge $e' \neq e^*$ such that $\sum_{S: u^* \text{or } v^* \in S} y(S) \leq \frac{1}{2} c(e') + \frac{3}{2} c(e^*)$. Then also done.

Open questions

Open questions

- Goemans and Williamson 1994 actually applied to downwards monotone functions $h : 2^V \to \mathbb{N}$ (can take multiple copies of an edge). Can our algorithm be extended to this case?

Open questions

- Goemans and Williamson 1994 actually applied to downwards monotone functions $h : 2^V \to \mathbb{N}$ (can take multiple copies of an edge). Can our algorithm be extended to this case?
- What about *proper* functions $f : 2^V \to \{0, 1\}$? f proper if $f(S) = f(V - S)$ and $f(A \cup B) \leq \max(f(A), f(B))$ for disjoint A, B. Includes Steiner tree, generalized Steiner tree, and others. Only a 2-approximation algorithm known for this class (Goemans Williamson 1995).

Open questions

- Goemans and Williamson 1994 actually applied to downwards monotone functions $h : 2^V \to \mathbb{N}$ (can take multiple copies of an edge). Can our algorithm be extended to this case?
- What about *proper* functions $f : 2^V \to \{0, 1\}$? f proper if $f(S) = f(V - S)$ and $f(A \cup B) \leq \max(f(A), f(B))$ for disjoint A, B. Includes Steiner tree, generalized Steiner tree, and others. Only a 2-approximation algorithm known for this class (Goemans Williamson 1995).

Thank you for your attention.