



# Optimisation Combinatoire pour la conception de circuits

## *Arbres de poids minimum*

Alix Munier Kordon  
Alix.Munier@lip6.fr

<http://www-asim.lip6.fr/~alix/>

Laboratoire Lip6  
Université Paris 6/ Paris 12

# Plan

- Arbre couvrant de poids minimum
- Approximant
- Algorithme de Kruskal
- Algorithme de Prim

# Arbre couvrant de poids minimum

Instance:  $G = (S, E)$  un graphe non orienté connexe,  $c : E \rightarrow \mathbb{Z}$ .

Question: peut on trouver un arbre  $H = (S, A)$  tel que  $\sum_{a \in A} c_a$  soit minimum ?

$H$  est un arbre couvrant de poids minimum.

# Approximant

**Définition 1.** Soient  $X$  et  $Y$  deux ensembles d'arêtes  $G$ .  $(X, Y)$  est un approximant si il existe un arbre couvrant de poids minimum  $H = (S, A)$  avec  $X \subset A$  et  $Y \subset E - A$ .

# Cocycle

**Définition 2.** Soit  $S' \subset S$ . Le cocycle  $\omega(S')$  est l'ensemble de arêtes  $\{x, y\} \in E$  telles que  $x \in S'$  et  $y \notin S'$ .

# Propriétés

**Lemme 3.** Soit  $Z \subset E - A$ .  $(\emptyset, \emptyset)$  et  $(A, Z)$  sont des approximants.

**Lemme 4.** Soit  $H = (S, A)$  un arbre couvrant de  $G$ , un cocycle et  $\gamma$  un cycle, alors  $\omega \cap H \neq \emptyset$  et  $\gamma \cap (E - A) \neq \emptyset$ .

# Règle des cocycles

**Lemme 5.** *Soit  $(X, Y)$  un approximant et soit  $\omega \neq \emptyset$  un cocycle tel que  $\omega \cap X = \emptyset$ . Soit  $a$  un arête de  $\omega \cap (E - Y)$  de poids  $c_a$  minimum. Alors,  $(X \cup \{a\}, Y)$  est un approximant.*

On note  $Rco(\omega)$

# Règle des cycles

**Lemme 6.** *Soit  $(X, Y)$  un approximant et soit  $\gamma \neq \emptyset$  un ensemble d'arêtes formant un cycle tel que  $\gamma \cap Y = \emptyset$ . Soit  $a$  un arête de  $\gamma \cap (E - X)$  de poids  $c_a$  maximum. Alors,  $(X, Y \cup \{a\})$  est un approximant.*

On note  $Rcy(\gamma)$

# Arbre de poids min

**fonction**  $\text{Arbre-min}(G = (S, E))$ :  $A \subset E$   
 $(X, Y) = (\emptyset, \emptyset)$ ;  
**tant que**  $\exists \omega, \text{Rco}(\omega)$  **ou**  $\exists \gamma, \text{Rcy}(\gamma)$   
choisir  $a \in \omega$  de poids min  
**ou**  $a \in \gamma$  de poids max;  
**si**  $a \in \omega$  **alors**  $X = X \cup \{a\}$   
**sinon**  $Y = Y \cup \{a\}$ ;  
**retourner**  $X$ ;

# Algorithme de Prim

```
fonction Algo-Prim( $G = (S, E)$ ):  $A \subset E$   
   $X = \emptyset$ ;  
  Soit  $x \in V$ .  $S' := \{x\}$ ;  
  tant que  $\omega(S') \neq \emptyset$   
    choisir  $a = \{y, z\} \in \omega(S')$  de poids min;  
     $S' = S' \cup \{y, z\}$ ;  
     $X := X \cup \{a\}$ ;  
retourner  $X$ ;
```

# Complexité de l'algorithme de Prim

On gère l'ensemble des sommets  $S - S'$  sous forme d'un tas. La clef de  $x$  est la valeur minimale d'une arête adjacente à  $y \in S'$  et  $x$ .  
Chaque arête permet d'évaluer au plus un sommet du tas:  $O(m \log n)$ .

# Algorithme de Kruskal

```
fonction Algo-Kruskal( $G = (S, E)$ ):  $A \subset E$   
   $X = \emptyset; i := 1;$   
  Trier les aretes telles que  $c_1 \leq c_2 \dots \leq c_m;$   
  tant que  $i \leq m$   
    si  $(S, X \cup \{a_i\})$  est sans cycle  
    alors  $X := X \cup \{a_i\};$   
     $i := i + 1;$   
  retourner  $X;$ 
```

# Complexité de l'algorithme de Kruskal

Tri des arêtes:  $O(m \log m)$ .

Utiliser une structure d'union-recherche pour déterminer si une arête ne crée pas de cycle. On obtient  $O(m \log n)$ .