

Optimisation Combinatoire pour la conception de circuits

Introduction

Alix Munier Kordon
Alix.Munier@lip6.fr

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

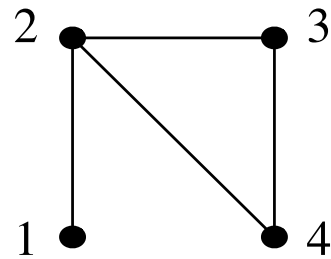
Laboratoire Lip6
Université Paris 6/ Paris 12

Plan

1. Terminologie sur les graphes et hypergraphes
2. Complexité
3. Méthodologie pour la résolution d'un problème d'optimisation
4. Représentation d'un circuit
5. Etapes de la conception physique

Terminologie sur les graphes et hypergraphes

Définition 1. *Un graphe non orienté G est définie par un couple $G = (V, E)$, où V est un ensemble de sommets et E un ensemble d'arêtes.*



$$V = \{1, 2, 3, 4\}, E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{2, 4\}\}.$$

Terminologie sur les graphes et hypergraphes

- Pour tout sommet $u \in V$,

$$\Gamma(u) = \{v \in V, \{u, v\} \in E\}$$

est l'ensemble des sommets *adjacents* à u . Le *degré* d'un sommet est $d(u) = |\Gamma(u)|$.

- Tout arête $e = \{u, v\} \in E$, e est *incidente* à u et v .
- Un sous graphe $G_s = (V_s, E_s)$ est un *sous-graphe partiel* de G si $V_s \subset V$ et

$$E_s = \{e \in E, e = \{u, v\}, u \in V_s, v \in V_s\}$$

Terminologie sur les graphes et hypergraphes

Soit $G = (V, E)$ un graphe.

$$\sum_{u \in V} d(u) = 2|V|$$

Terminologie sur les graphes et hypergraphes

- Un graphe *complet* est tel que tous les sommets sont adjacents deux à deux.
- Une *clique* de G est un sous-graphe partiel complet.
Par exemple, $G = (V_s, E_s)$ avec $V_s = \{2, 3, 4\}$ est une clique.
- Un *stable* de G est un sous-graphe partiel sans arêtes.
Par exemple, $G = (V_s, E_s)$ avec $V_s = \{3, 1\}$ est un stable.

Terminologie sur les graphes et hypergraphes

- Un *chaîne* est une séquence de sommets et d'arêtes telle que

$$v = v_1 e_1 v_2 e_2 \dots v_n e_n v_{n+1}$$

avec $e_i = \{v_i, v_{i+1}\}, i \in \{1, \dots, n\} \in E$. Un chemin élémentaire ne passe pas deux fois par le même sommet.

- Un *cycle* est un chemin v tel que $v_{n+1} = v_1$.
- Un graphe *connexe* est tel que, pour tout couple $(u, v) \in V^2$, il existe un chemin de u à v .

Terminologie sur les graphes et hypergraphes

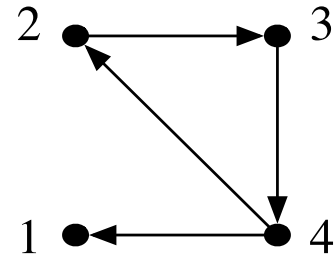
Un *arbre* est un graphe connexe sans cycle.
Les propriétés suivantes sont équivalentes:

- $T = (V, E)$ est un arbre.
- T est minimal connexe: pour tout $e \in E$, $T - E$ n'est pas connexe.
- Entre deux sommets quelconques, il existe une chaîne unique.

Si $T = (V, E)$ est un arbre, $|E| = |V| - 1$.

Terminologie sur les graphes et hypergraphes

Si $G = (V, E)$ est un graphe orienté:



arête $e = \{u, v\} \rightarrow$ arc $e = (u, v)$

cycle \rightarrow circuit

chaîne \rightarrow chemin

Terminologie sur les graphes et hypergraphes

Pour tout sommet $u \in V$,

$$\Gamma^+(u) = \{v \in V, \exists e \in E, (u, v) \in E\}$$

$$\Gamma^-(u) = \{v \in V, \exists e \in E, (v, u) \in E\}$$

$d^+(u) = |\Gamma^+(u)|$ est le $\frac{1}{2}$ -degré entrant de u .

$d^-(u) = |\Gamma^-(u)|$ est le $\frac{1}{2}$ -degré sortant de u .

$$d(u) = d^+(u) + d^-(u)$$

$$\sum_{u \in V} d^+(u) = \sum_{u \in V} d^-(u) = |V|$$

Terminologie sur les graphes et hypergraphes

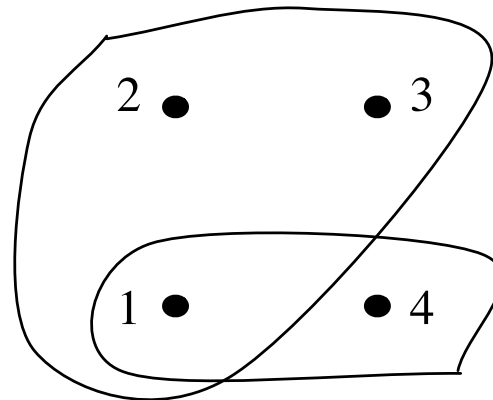
Un graphe orienté est *fortement connexe* ssi pour tout couple $(u, v) \in V^2$, il existe un chemin de u à v .

Tout graphe orienté est décomposable en un ensemble $G_i = (V_i, E_i), i = 1..k$ de sous-graphes partiels fortement connexes tels que:

- $V_i, i = 1..k$ forme une partition de V ,
- Le graphe obtenu en réduisant chaque G_i à un sommet est sans circuit.

Terminologie sur les graphes et hypergraphes

Un *hypergraphe* $G = (V, E)$ est défini par un ensemble de sommets V et un ensemble d'*hyperarêtes* $E \subset \mathcal{P}(V)$.



$$V = \{1, 2, 3, 4\}, E = \{\{1, 2, 3\}, \{1, 4\}\}.$$

Complexité

1. Problèmes de décision, problème d'optimisation
2. Complexité d'un algorithme
3. Classes de complexité

Problème

Un *problème* Π est définie par l'ensemble de définition de ses paramètres \mathcal{D} et une question à résoudre. \mathcal{R} est l'ensemble des réponses à la question.

Tout élément $d \in \mathcal{D}$ est une *instance* de Π .

La *taille du codage* des instances de P est le nombre d'octets nécessaires pour coder un élément $d \in \mathcal{D}$.

Problème d'optimisation

de

décision,

Un *problème de décision* est un problème Π pour lequel $\mathcal{R} = \{oui, non\}$.

Un *problème d'optimisation* est un problème Π pour lequel la question consiste à optimiser une fonction f .

Problème SAT

Instance: variables booléennes x_1, \dots, x_n , m clauses C_i sur $\{x_1, \dots, x_n\} \cup \{\bar{x}_1, \dots, \bar{x}_n\}$.

Question: peut on trouver une affectation des valeurs booléennes telle que toutes les clauses soient vraies ?

Problème d'optimisation de décision,

A tout problème d'optimisation, on peut associer un problème de décision: il suffit de se demander si il existe une instance $d \in \mathcal{D}$ telle que $f(d) \leq B$.

Si on sait résoudre le problème d'optimisation, on sait également résoudre le problème de décision.

Problème de décision, d'optimisation

Réciproquement, on peut résoudre un problème d'optimisation si on sait résoudre le problème de décision associé.

Il suffit de baisser la valeur B jusqu'à ce que réponse soit non (*Trying*).

Algorithme et problème

Un *algorithme* A est une suite séquentielle d'instructions qui modifient l'état de la mémoire. Il prend en entrée tout élément de $d \in \mathcal{D}$ et calcule un résultat $A(d) \in \mathcal{R}$.

Un algorithme A résout un problème P si, pour tout $d \in \mathcal{D}$, $A(d)$ résout la question posée pour l'instance d de P .

Complexité d'un algorithme

La complexité d'un algorithme est une évaluation du nombre d'instructions et de la place mémoire utilisé pour l'exécuter.

Ce sont généralement des fonctions de la taille de codage ou des valeurs des instances de A .

Ordre de grandeur

- Un algorithme *polynomial* (*resp. exponentiel*) est un algorithme dont la complexité est bornée par une fonction polynomiale (*resp. exponentielle*) de la taille de codage des instances.
- Un algorithme *pseudo-polynomial* est un algorithme dont la complexité est bornée par une fonction polynomiale de la valeur des instances.

Ordre de grandeur

- Un algorithme polynomial est beaucoup plus rapide qu'un algorithme exponentiel
- Un algorithme linéaire (en $O(n)$) est beaucoup plus rapide qu'un algorithme quadratique.
- Quand les données vérifient de bonnes propriétés, on peut obtenir des ordre de grandeur sous-linéaires (inférieurs à $O(n)$).

Problèmes indécidables

Un problème Π est dit *indécidable* si on ne peut trouver une solution algorithmique pour le résoudre.

On ne s'intéresse qu'à des problèmes décidables !

Nondeterministic Polynomial

Un problème de décision $\Pi \in NP$ si et seulement si, pour chaque instance $d \in \mathcal{D}_\Pi$ pour laquelle la réponse à Π est oui, on peut associer *un certificat* $I(d) \in \{0, 1\}^*$ tel que:

- La taille de codage de $I(d)$ est polynomiale en fonction de d ,
- il existe un algorithme polynomial qui permet de vérifier à partir de $(d, I(d))$ que la réponse de Π à d est oui.

Nondeterministic Polynomial

Si $\Pi \in NP$, on peut calculer en un temps exponentiel tous les certificats de Π .

Les problèmes de NP sont résolubles au minimum par un algorithme exponentiel. Ils sont donc décidables.

P **et** *NP*

P est l'ensemble des problèmes résolubles en temps polynomial.

$$P \subset NP$$

Transformation polynomiale

Soient Π_1 et Π_2 deux problèmes de décision, \mathcal{D}_{Π_1} , \mathcal{D}_{Π_2} leur ensemble d'instances.

Une *transformation polynomiale* de Π_1 vers Π_2 est une fonction $F : \mathcal{D}_{\Pi_1} \rightarrow \mathcal{D}_{\Pi_2}$ telle que:

- Pour tout $I \in \mathcal{D}_{\Pi_1}$, $F(I)$ se calcule de manière polynomiale en fonction de la taille de I .
- Pour tout $I \in \mathcal{D}_{\Pi_1}$, la réponse de Π_1 à I est oui si et seulement si la réponse de Π_2 à $F(I)$ est oui.

On note $\Pi_1 \leq \Pi_2$.

Problèmes NP-complet

Theorème 2. *Pour tout $\Pi \in NP$, $\Pi \leq SAT$*
Théorème de Cook (1971)

SAT est le problème le plus difficile de NP.

Problèmes NP-complet

Définition 3. *Un problème Π est NP-complet si $\Pi \in NP$ et $SAT \leq \Pi$*

Les problèmes NP-complets sont les problèmes équivalents en complexité à SAT.

Hypothèse de base

$P \subset NP$ par définition.

On suppose que $P \neq NP$ (non démontré).

Ce qui veut dire que tout problème NP-complet ne peut être résolu par un algorithme polynomia

Pour démontrer que Π est NP-complet

- Montrer que $\Pi \in NP$.
- Trouver un problème Π' NP-complet et montrer que $\Pi' \leq \Pi$.

Méthodologie pour la résolution d'un problème d'optimisation

Soit Π un problème d'optimisation dont la version de décision $\Pi_{des} \in NP$.

- Si Π_{des} est polynomial, Π l'est aussi.
- Sinon, on peut (peut-être) montrer que $\Pi_{des} \in NP$ -complet.

Résolution d'un problème d'optimisation

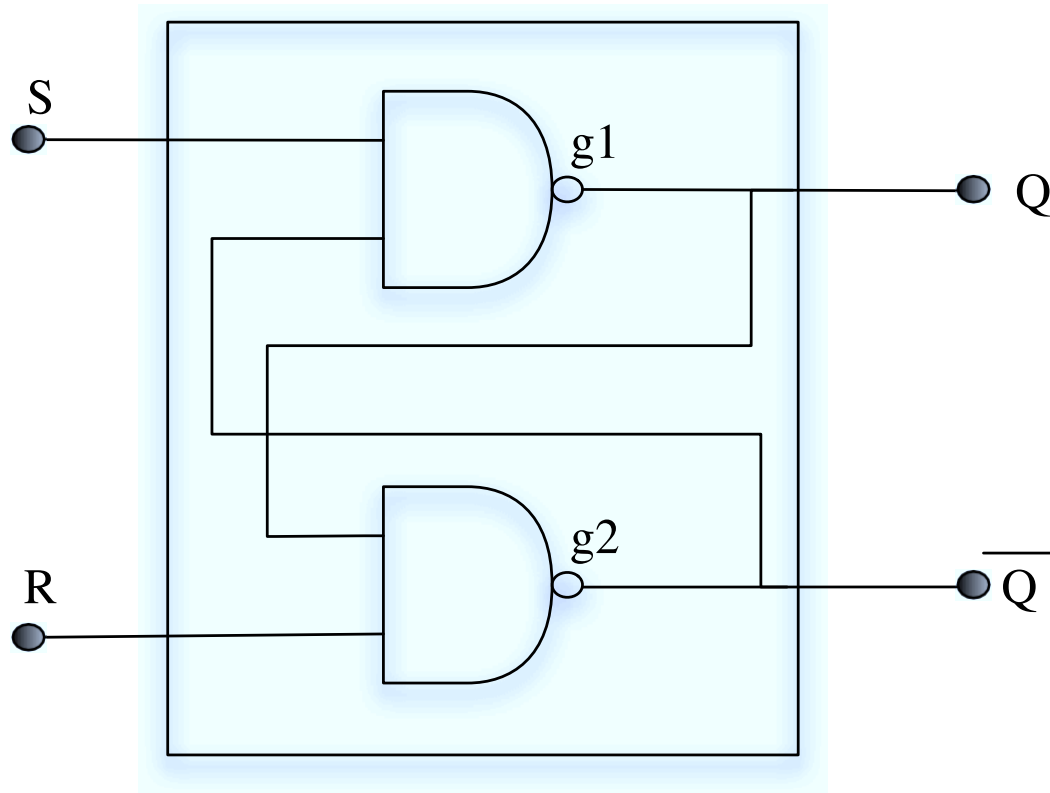
Si l'on ne dispose pas d'un algorithme polynomial pour Π , que peut on faire ?

- Si on recherche une solution exacte, on peut dans certains cas avoir un algorithme pseudo-polynomial.
- Sinon, il faut développer une méthode arborescente.

Résolution d'un problème d'optimisation

- Si on accepte une solution approximative, on peut développer des algorithmes approchés ou une heuristique.
- En simplifiant la structure des instances, on peut souvent trouver un sous-problème polynomial. Cette étude peut permettre d'avoir de bonnes idées pour une heuristique.

Représentation d'un circuit



Des cells, des ports et des nets...

Cell

Eléments de base.

- un type (ici, porte NAND),
- un nom unique,
- des ports de sortie et d'entrée.

port

Point de connection des éléments de base.

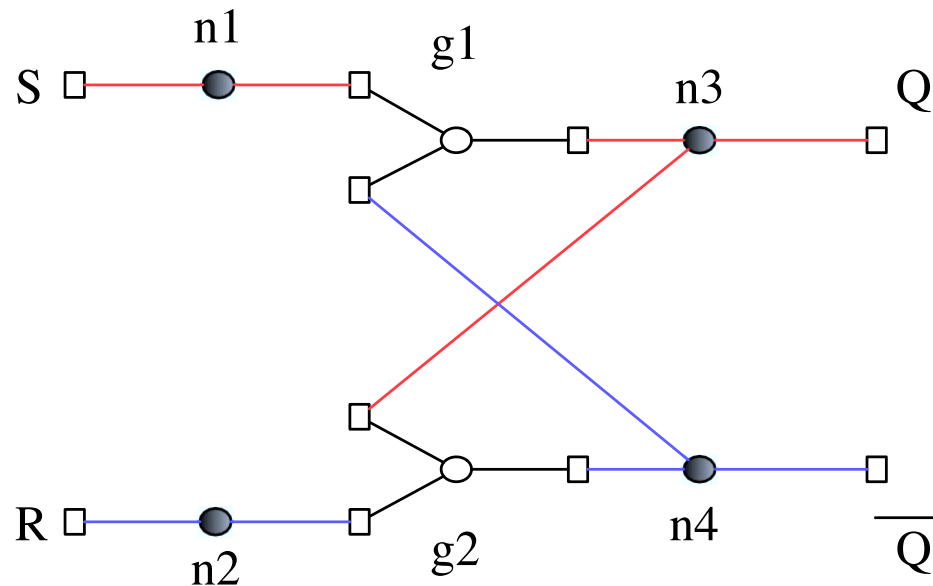
- un type,
- un nom unique,
- la cellule correspondante,
- le net qui le relie.

net

Le fil qui connecte plusieurs ports.

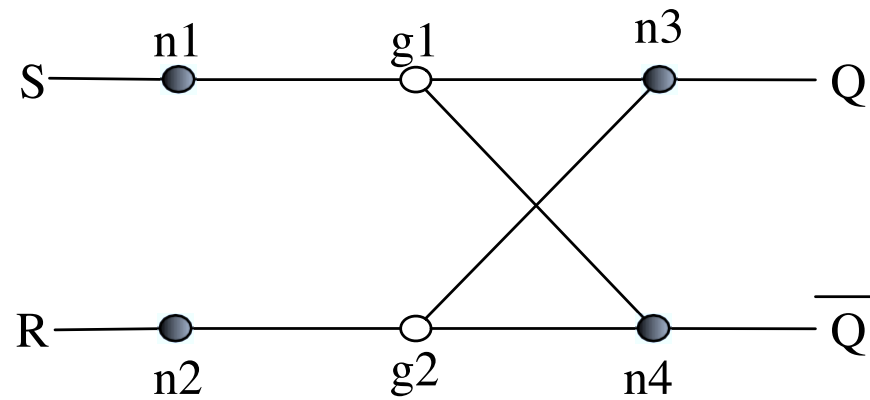
- un nom unique,
- la liste des ports connectés

Cells, ports et nets



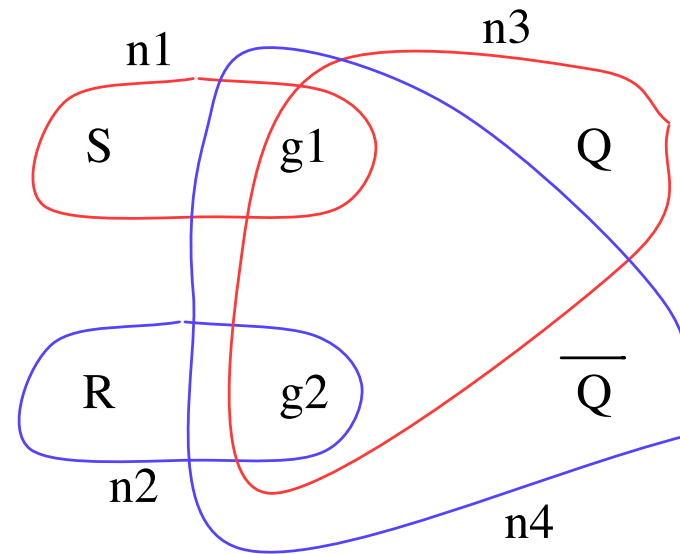
Graphe avec 3 types de noeuds...

Cells, ports et nets



Sans les ports...

Cells, ports et nets



Sous forme d'un hypergraphe..

Etapes de la conception physique

En entrée: une description du circuit (netlist).

En sortie: masques pour la fabrication de la puce

- Partitionnement
- Floorplanning et placement
- Routage
- Compaction