Exploration de Graphes par Automates Lien avec SL=L (Omer Reingold)

David Ilcinkas

LRI, Université Paris-Sud, France

Les Sept Laux
18 janvier 2005

Première partie I

Exploration de graphes par automates

Exploration de graphes

But

Une entité mobile doit traverser chaque arête d'un graphe inconnu et anonyme.

Motivation

- exploration d'endroits inaccessibles à l'homme
- maintenance d'un réseau informatique
- cartographie

Exploration de graphes

But

Une entité mobile doit traverser chaque arête d'un graphe inconnu et anonyme.

Motivation

- exploration d'endroits inaccessibles à l'homme
- maintenance d'un réseau informatique
- cartographie

Inconnu, Anonyme

Inconnu

- topologie inconnue
- pas de bornes sur la taille

Anonyme

- pas d'étiquetage des sommets
- étiquetage local des arêtes

Inconnu, Anonyme

Inconnu

- topologie inconnue
- pas de bornes sur la taille

Anonyme

- pas d'étiquetage des sommets
- étiquetage local des arêtes

Inconnu, Anonyme

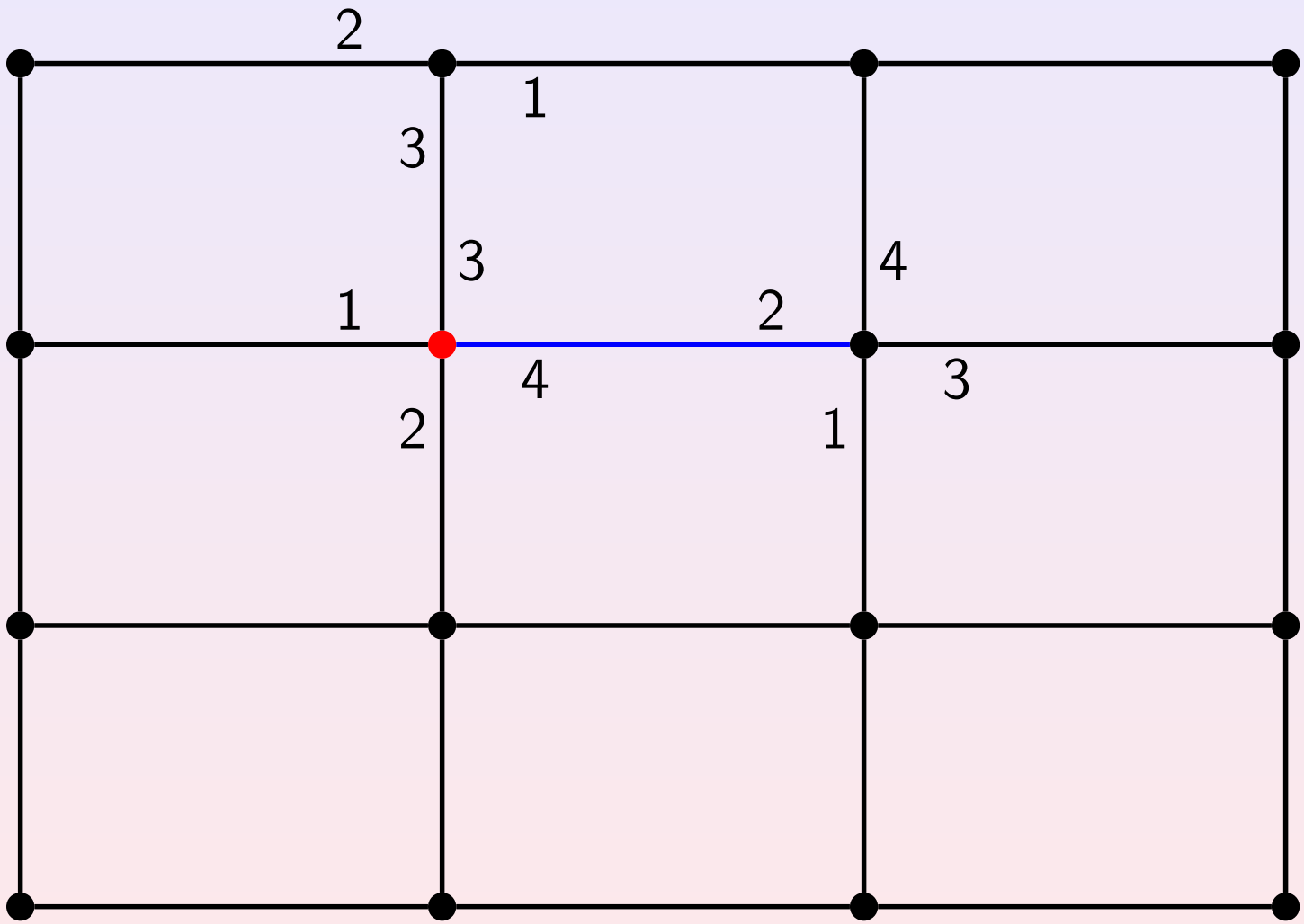
Inconnu

- topologie inconnue
- pas de bornes sur la taille

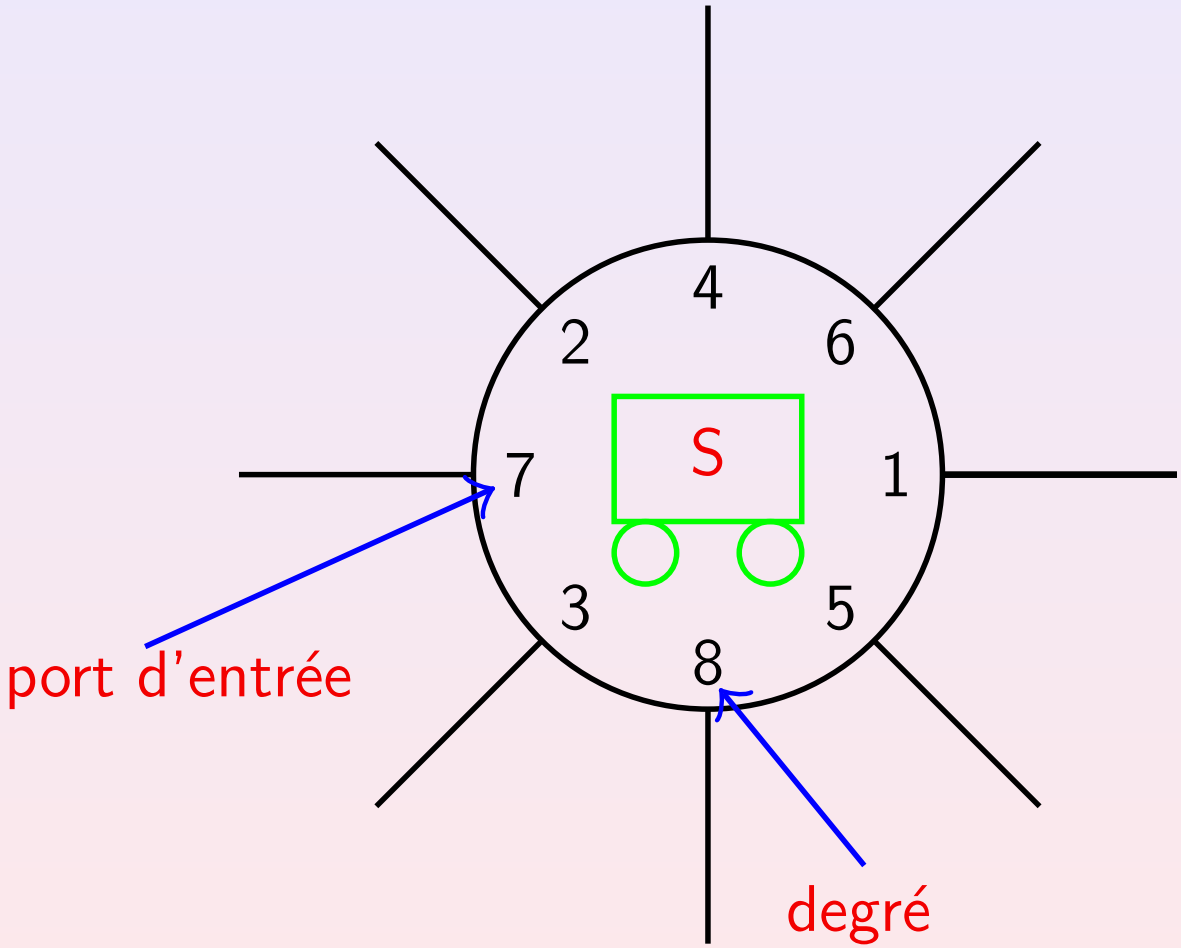
Anonyme

- pas d'étiquetage des sommets
- étiquetage local des arêtes

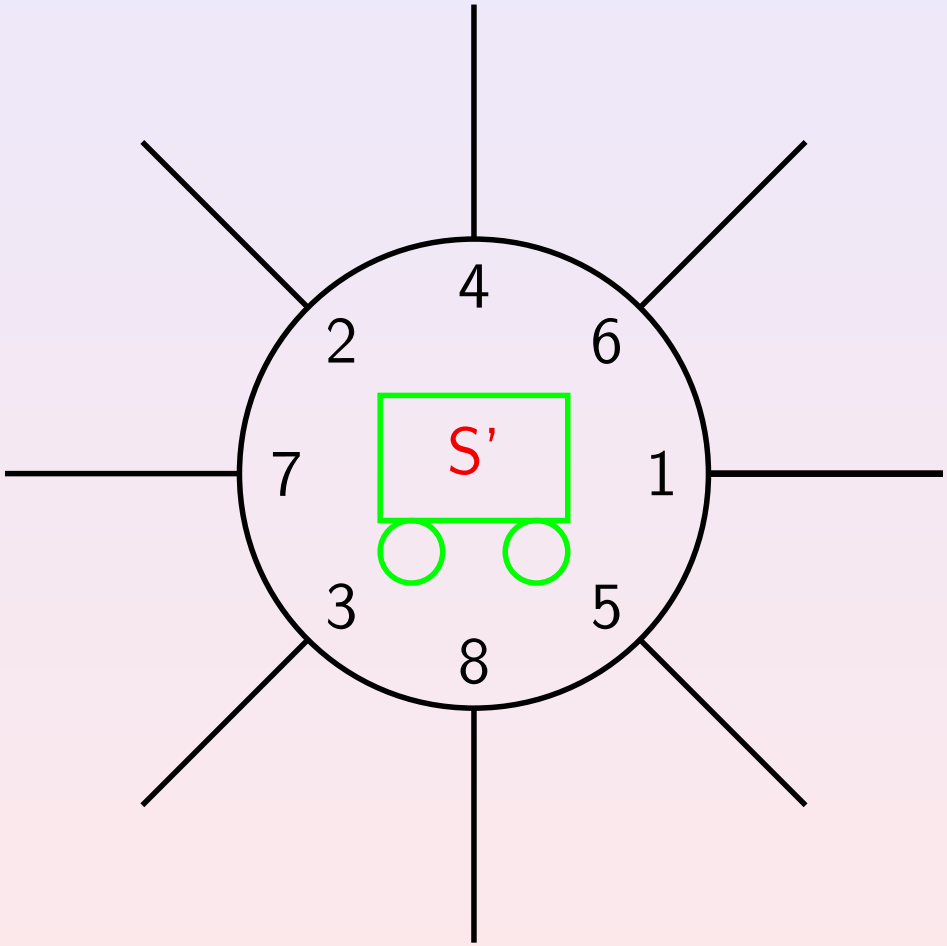
Exemple d'un graphe anonyme



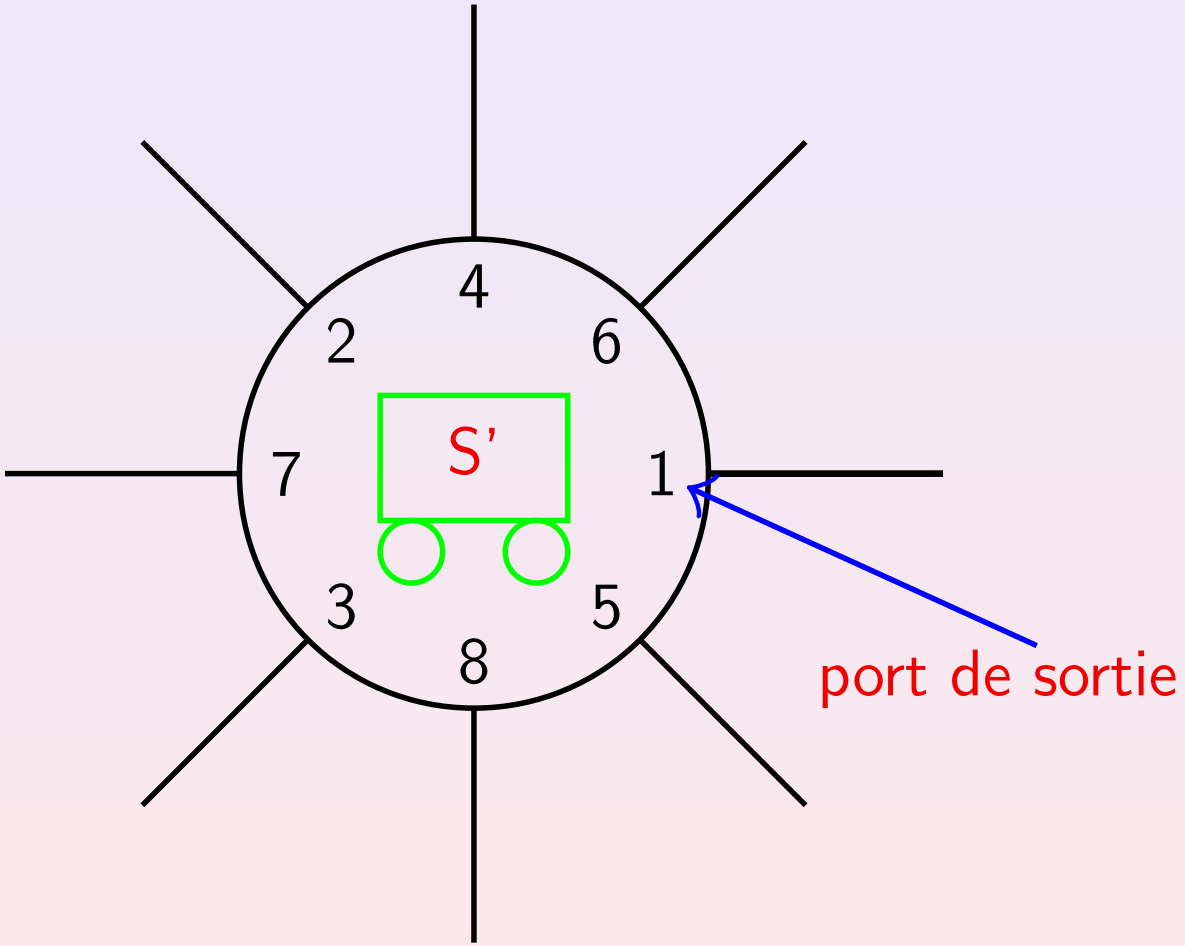
Entrée de l'automate



Entrée de l'automate



Entrée de l'automate



Plan

1 Introduction

2 Faisabilité
- Labyrinthes
- Graphes
- Hydre

3 Minimiser la mémoire

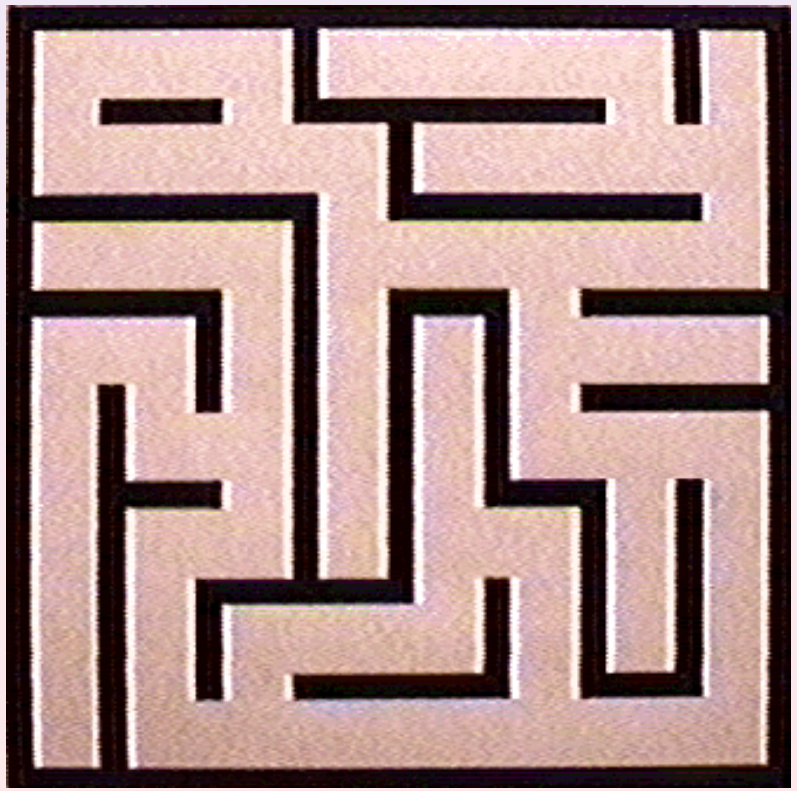
4 Coloriage par un oracle

5 Autres modèles

Cas particulier : Labyrinthes

Définition

- grille avec des arêtes manquantes
- étiquetage cohérent des arêtes : Nord, Sud, Est, Ouest



Résultats sur les labyrinthes

Budach, Math. Nachrichten 1978
Automata and Labyrinths

$\nexists$ automate fini universel

Blum, Kozen, FOCS 1978
On the power of the compass

$\exists$ automate fini universel à deux cailloux
$\exists$ deux automates coopératifs universels pour les labyrinthes
$\nexists$ deux automates coopératifs universels pour les graphes

Hoffmann, FCT 1981
One pebble does not suffice to search plane labyrinths

$\nexists$ automate fini universel à un caillou

Résultats sur les labyrinthes

Budach, Math. Nachrichten 1978
Automata and Labyrinths

$\nexists$ automate fini universel

Blum, Kozen, FOCS 1978
On the power of the compass

$\exists$ automate fini universel à deux cailloux

$\exists$ deux automates coopératifs universels pour les labyrinthes

$\nexists$ deux automates coopératifs universels pour les graphes

Budach, FCT 1981
One pebble does not suffice to search plane labyrinths

$\nexists$ automate fini universel à un caillou

Résultats sur les labyrinthes

Budach, Math. Nachrichten 1978
Automata and Labyrinths

$\nexists$ automate fini universel

Blum, Kozen, FOCS 1978
On the power of the compass

$\exists$ automate fini universel à deux cailloux
$\exists$ deux automates coopératifs universels pour les labyrinthes
$\nexists$ deux automates coopératifs universels pour les graphes

Budach, FCT 1981
One pebble does not suffice to search plane labyrinths

$\nexists$ automate fini universel à un caillou

Résultats sur les labyrinthes

Budach, Math. Nachrichten 1978
Automata and Labyrinths

$\nexists$ automate fini universel

Blum, Kozen, FOCS 1978
On the power of the compass

$\exists$ automate fini universel à deux cailloux
$\exists$ deux automates coopératifs universels pour les labyrinthes
$\nexists$ deux automates coopératifs universels pour les graphes

Hemmerling, FCT 1981
One pebble does not suffice to search plane labyrinths

$\nexists$ automate fini universel à un caillou

Résultats sur les labyrinthes

Budach, Math. Nachrichten 1978
Automata and Labyrinths

$\nexists$ automate fini universel

Blum, Kozen, FOCS 1978
On the power of the compass

$\exists$ automate fini universel à deux cailloux
$\exists$ deux automates coopératifs universels pour les labyrinthes
$\nexists$ deux automates coopératifs universels pour les graphes

Hoffmann, FCT 1981
One pebble does not suffice to search plane labyrinths

$\nexists$ automate fini universel à un caillou

Graphes arbitraires

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

Aucun automate fini n'explore tous les labyrinthes ⇒ graphes.

Rabin, Acta Informatica, 1980
Automaten in planaren Graphen

Aucune équipe finie d'automates finis n'explore tous les graphes (mêmes planaires).

Graphes arbitraires

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

Aucun automate fini n'explore tous les labyrinthes $\Rightarrow$ graphes.

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

Aucune équipe finie d'automates finis n'explore tous les graphes (mêmes planaires).

Comment piéger un automate (Rollik)

Piège pour un automate spécifique

- automate fini $\iff$ comportement devient périodique
- piéger l'automate dans un cycle

Piège pour k automates non coopératifs

- défini récursivement
- placer un piège pour $k-1$ dans chaque arête

Piège pour k automates coopératifs

- défini récursivement
- placer un piège pour $k-1$ dans chaque arête
- méta-structure plus complexe

Comment piéger un automate (Rollik)

Piège pour un automate spécifique

- automate fini $\iff$ comportement devient périodique
- piéger l'automate dans un cycle

Piège pour k automates non coopératifs

- défini récursivement
- placer un piège pour $k-1$ dans chaque arête

Piège pour k automates coopératifs

- défini récursivement
- placer un piège pour $k-1$ dans chaque arête
- méta-structure plus complexe

Comment piéger un automate (Rollik)

Piège pour un automate spécifique

- automate fini $\Longleftrightarrow$ comportement devient périodique
- piéger l'automate dans un cycle

Piège pour k automates non coopératifs

- défini récursivement
- placer un piège pour $k-1$ dans chaque arête

Piège pour k automates coopératifs

- défini récursivement
- placer un piège pour $k-1$ dans chaque arête
- méta-structure plus complexe

Modèles de coopération

Coopération locale (Rollik)

Deux robots ne peuvent communiquer que s'ils sont sur le même nœud au même instant.

Coopération globale

Les robots communiquent en permanence.
En particulier, ils se communiquent les rencontres.

Modèles de coopération

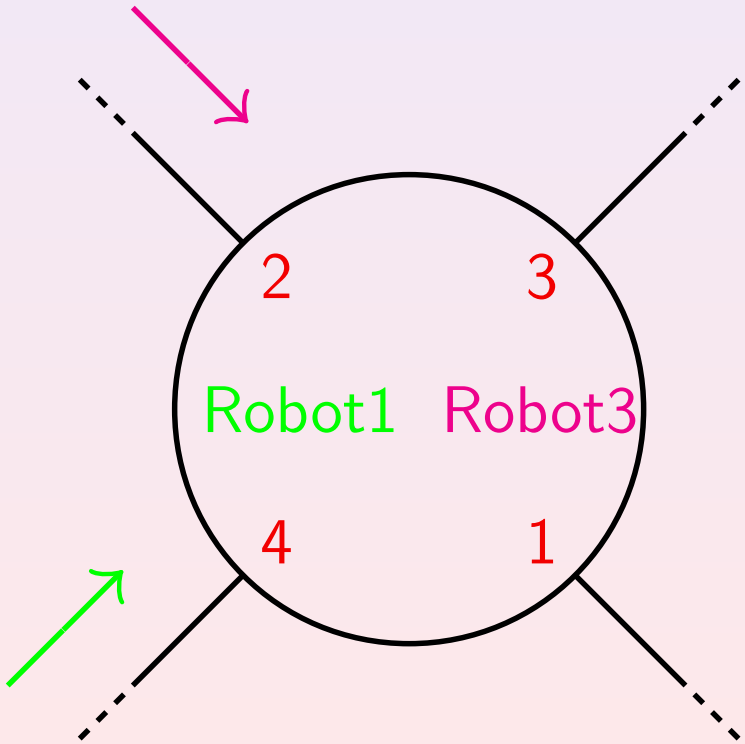
Coopération locale (Rollik)

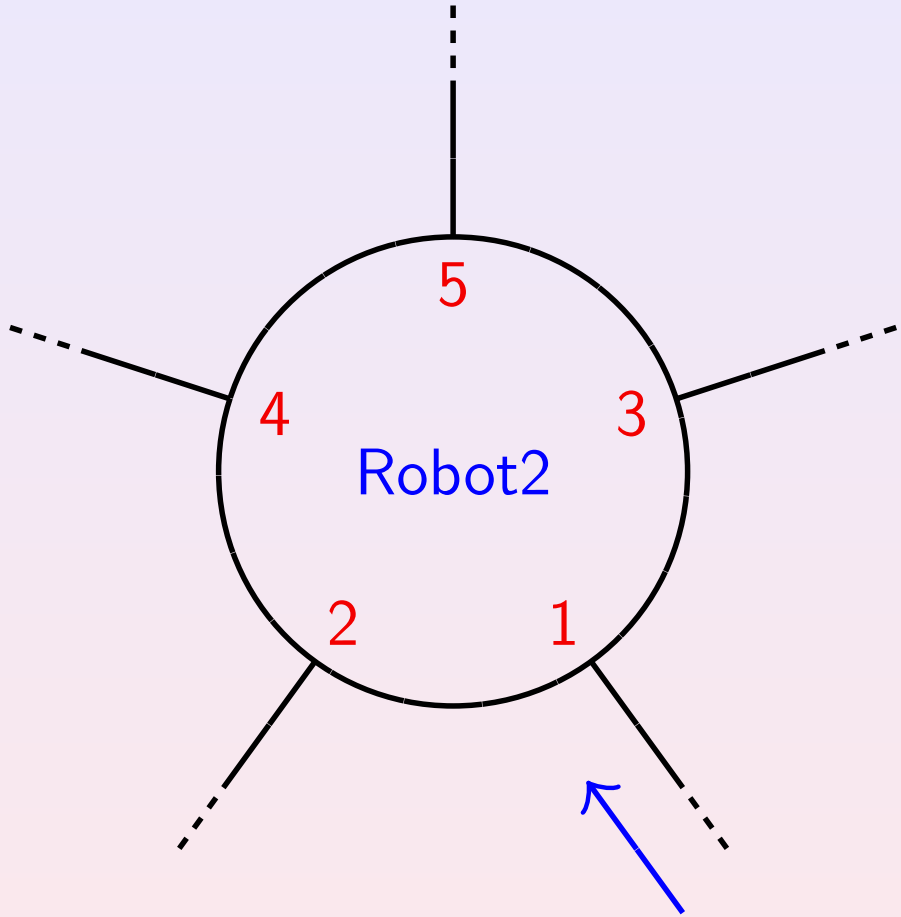
Deux robots ne peuvent communiquer que s'ils sont sur le même nœud au même instant.

Coopération globale

Les robots communiquent en permanence.
En particulier, ils se communiquent les rencontres.

Exemple





Hydre

connaissance totale $\iff$ état unique

Définition: Hydre

Automate fini à têtes multiples

k automates coopérant globalement $\iff$ une hydre à k têtes

Hydre

connaissance totale $\Longleftrightarrow$ état unique

Définition: Hydre

Automate fini à têtes multiples

k automates coopérant globalement $\Longleftrightarrow$ une hydre à k têtes

Hydre

connaissance totale $\iff$ état unique

Définition: Hydre

Automate fini à têtes multiples

k automates coopérant globalement $\iff$ une hydre à k têtes

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succeq$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succeq$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succ$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succ$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succ$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succ$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Puissance de l'hydre

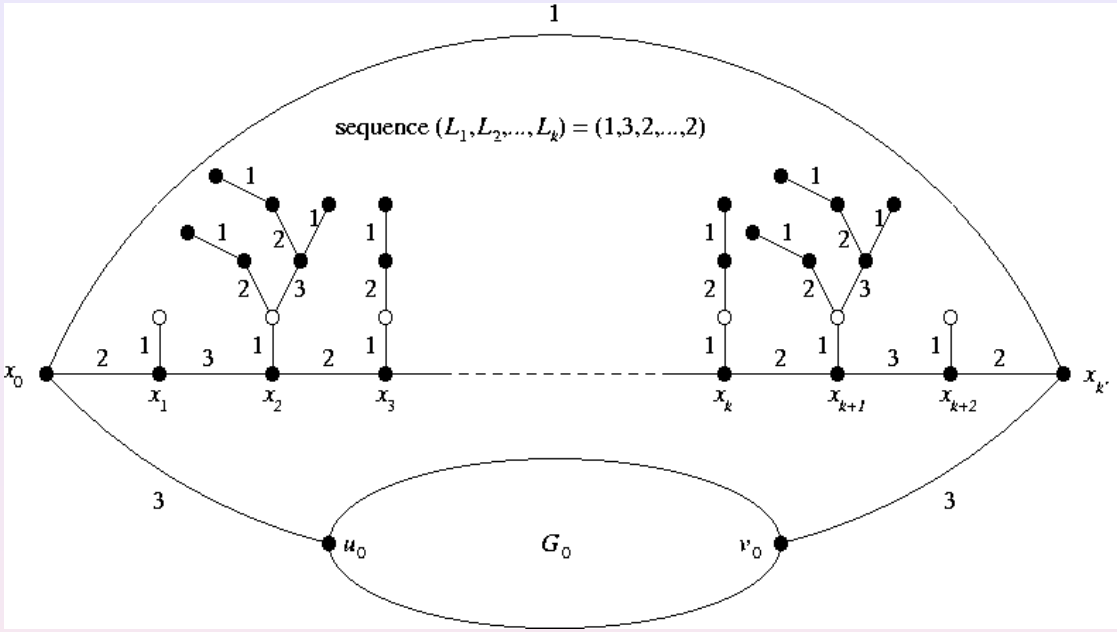
Fraigniaud, Ilcinkas, Markou, Pelc

Power of communication in cooperative exploration of graphs

- une hydre à 2 têtes $\not\succ$ une équipe finie d'automates finis
- une hydre à k têtes $\succ$ k automates coopérant localement
- une hydre à 2 têtes n'est pas universelle
- une hydre à 3 têtes $\succ$ une hydre à 2 têtes
- une hydre à 3 têtes n'est pas universelle

Conjecture : Aucune hydre n'est universelle

Hydre vs local



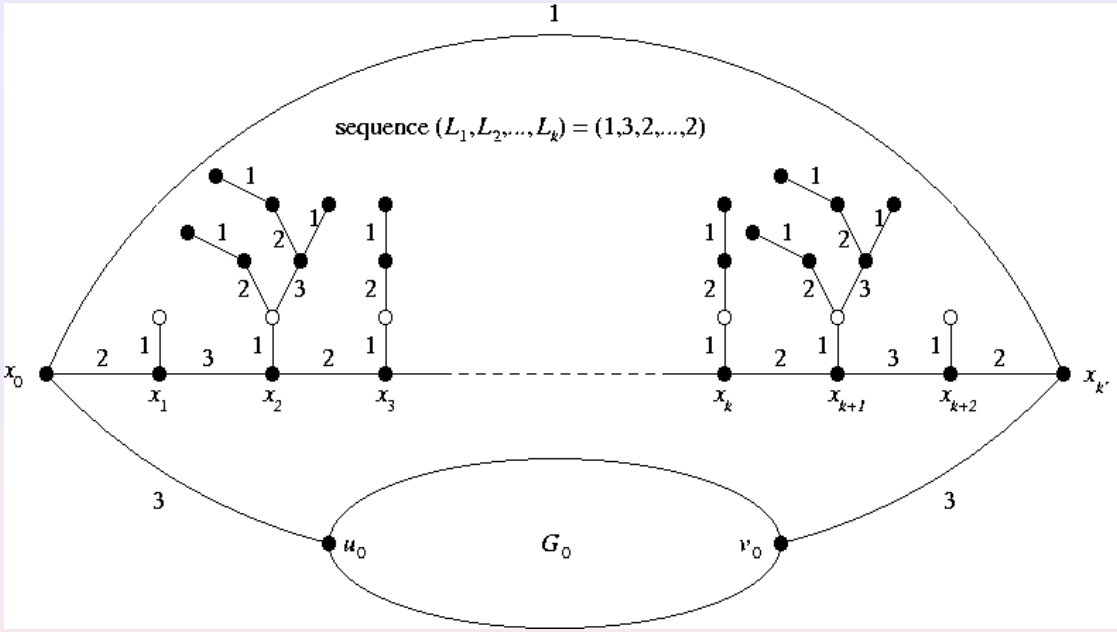
Conclusion

une hydre à 2 têtes $\not\preceq$ une équipe finie d'automates finis

Corollaire

une hydre à k têtes $\succ$ k automates coopérant localement

Hydre vs local



Conclusion

une hydre à 2 têtes $\not\preceq$ une équipe finie d'automates finis

Corollaire

une hydre à k têtes $\succ$ k automates coopérant localement

Piège pour une hydre à 2 têtes

Graphe d-homogène

- d-régulier
- à arêtes coloriées (même étiquette aux deux extrémités)

Dans les graphes d-homogènes :
hydre à deux têtes $\equiv$ équipe de deux automates

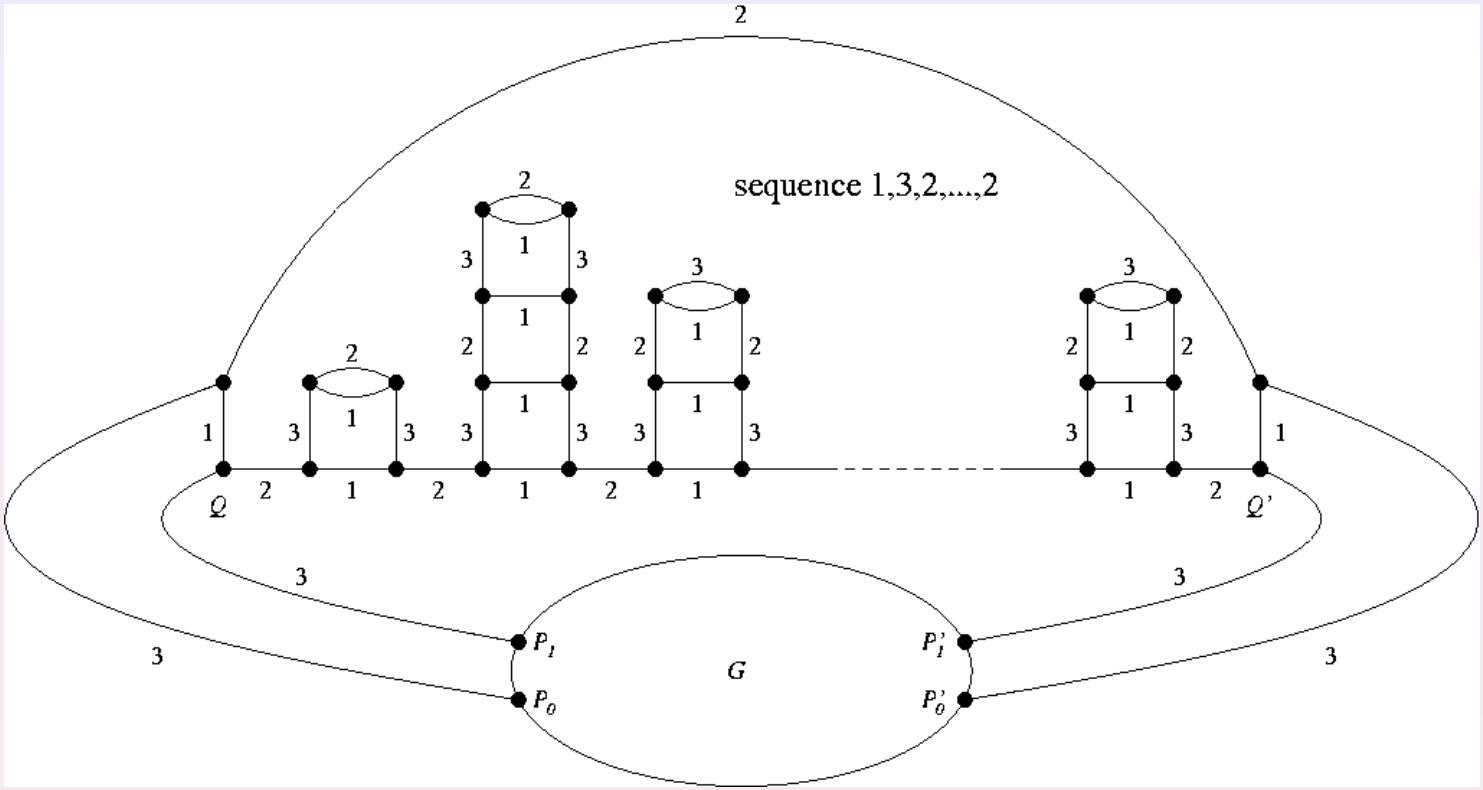
Piège pour une hydre à 2 têtes

Graphe d-homogène

- d-régulier
- à arêtes coloriées (même étiquette aux deux extrémités)

Dans les graphes d-homogènes :
hydre à deux têtes $\equiv$ équipe de deux automates

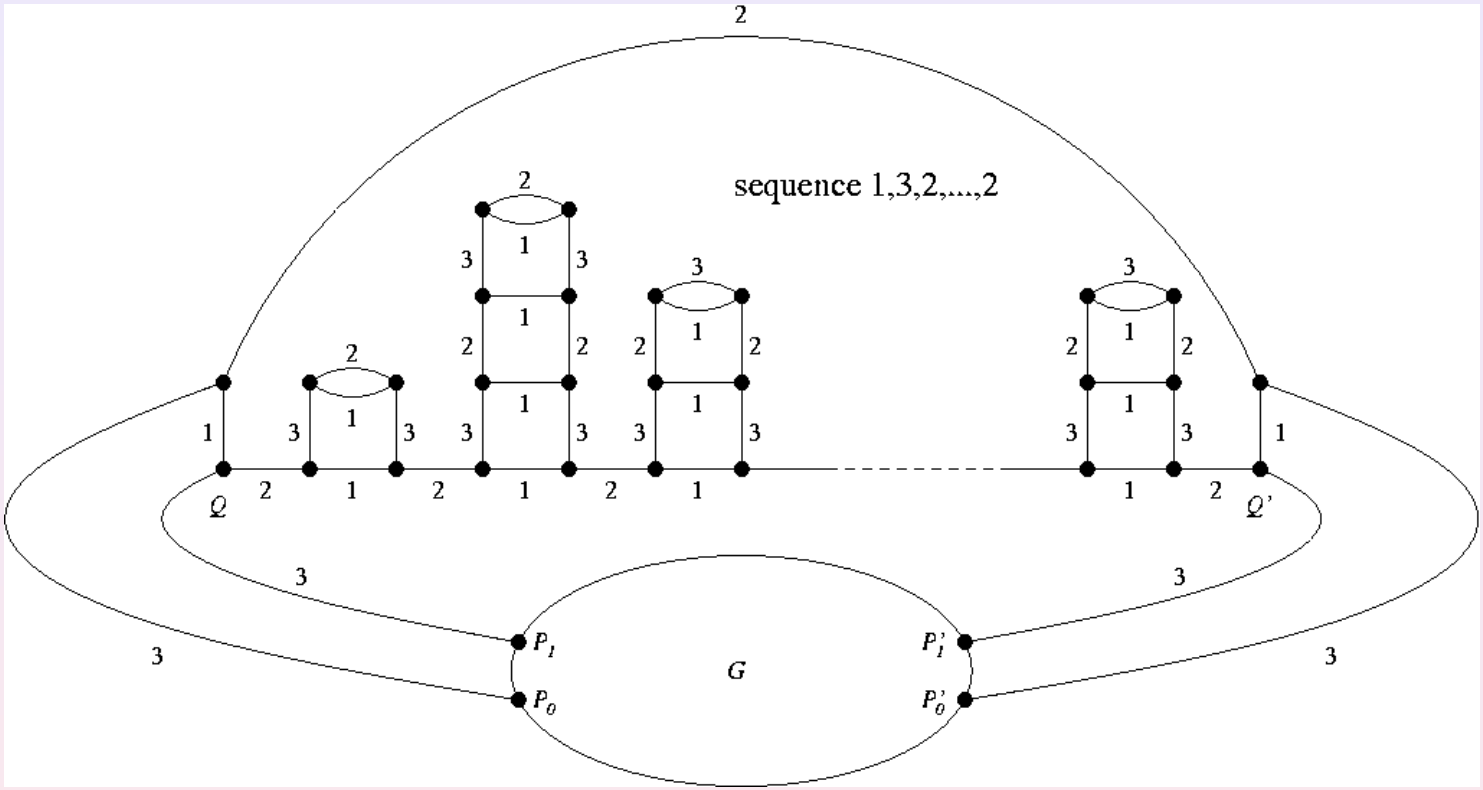
2-hydre vs 3-hydre



Conclusion

une hydre à 3 têtes $\succ$ une hydre à 2 têtes

2-hydre vs 3-hydre



Conclusion

une hydre à 3 têtes $\succ$ une hydre à 2 têtes

Piège pour une hydre à 3 têtes

Piège pour une hydre à 3 têtes

- Têtes séparées
  - Deux têtes (ensemble ou pas)
    - Aucune information utile de la troisième tête
    - Piège pour une hydre à deux têtes
  - Une tête séparée
    - Reçoit des informations périodiques des deux autres
    - Simulable par un gros automate
    - Tête piégée dans une méta-structure
- Têtes non séparées (distance bornée les unes des autres)
  - Simulable par un gros automate
  - Têtes piégées dans une méta-méta-structure

Piège pour une hydre à 3 têtes

- Têtes séparées
  - Deux têtes (ensemble ou pas)
    - Aucune information utile de la troisième tête
    - Piège pour une hydre à deux têtes
  - Une tête séparée
    - Reçoit des informations périodiques des deux autres
    - Simulable par un gros automate
    - Tête piégée dans une méta-structure
- Têtes non séparées (distance bornée les unes des autres)
  - Simulable par un gros automate
  - Têtes piégées dans une méta-méta-structure

Piège pour une hydre à 3 têtes

- Têtes séparées
  - Deux têtes (ensemble ou pas)
    - Aucune information utile de la troisième tête
    - Piège pour une hydre à deux têtes
  - Une tête séparée
    - Reçoit des informations périodiques des deux autres
    - Simulable par un gros automate
    - Tête piégée dans une méta-structure
- Têtes non séparées (distance bornée les unes des autres)
  - Simulable par un gros automate
  - Têtes piégées dans une méta-méta-structure

Piège pour une hydre à 3 têtes

- Têtes séparées
  - Deux têtes (ensemble ou pas)
    - Aucune information utile de la troisième tête
    - Piège pour une hydre à deux têtes
  - Une tête séparée
    - Reçoit des informations périodiques des deux autres
    - Simulable par un gros automate
    - Tête piégée dans une méta-structure
- Têtes non séparées (distance bornée les unes des autres)
  - Simulable par un gros automate
  - Têtes piégées dans une méta-méta-structure

Plan

1 Introduction

2 Faisabilité

3 Minimiser la mémoire
- Différentes tâches
- Arbres
- Graphes arbitraires

4 Coloriage par un oracle

5 Autres modèles

Différentes tâches

- Exploration perpétuelle
- Exploration avec arrêt
  le robot doit s'arrêter en étant sûr d'avoir fini l'exploration
- Exploration avec retour
  le robot doit revenir s'arrêter sur le sommet de départ
- Cartographie
  le robot doit fournir une carte étiquetée isomorphe au graphe

Arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

- Perpétuelle : $\Theta(\log \Delta)$ bits
- Avec arrêt : $\Omega(\log \log \log n)$ bits
- Avec retour : $\Omega(\log n)$, $O(\log^2 n)$ bits

Arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

- Perpétuelle : $\Theta(\log \Delta)$ bits
- Avec arrêt : $\Omega(\log \log \log n)$ bits
- Avec retour : $\Omega(\log n)$, $O(\log^2 n)$ bits

Arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

- Perpétuelle : $\Theta(\log \Delta)$ bits
- Avec arrêt : $\Omega(\log \log \log n)$ bits
- Avec retour : $\Omega(\log n)$, $O(\log^2 n)$ bits

Graphes arbitraires

Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, MFCS 2004
Graph exploration by a finite automaton

Exploration perpétuelle :

- Pour tout automate à K états, il existe un piège d'au plus $K + 1$ nœuds.
- Donc un automate qui explore tous les graphes d'au plus K sommets a au moins K états.
- DFS est optimal en mémoire : $\Theta(D \log \Delta)$ bits

Graphes arbitraires

Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, MFCS 2004
Graph exploration by a finite automaton

Exploration perpétuelle :

- Pour tout automate à K états, il existe un piège d'au plus $K + 1$ nœuds.
- Donc un automate qui explore tous les graphes d'au plus K sommets a au moins K états.
- DFS est optimal en mémoire : $\Theta(D \log \Delta)$ bits

Graphes arbitraires

Fraigniaud, Ilcinkas, Peer, Pelc, Peleg, MFCS 2004
Graph exploration by a finite automaton

Exploration perpétuelle :

- Pour tout automate à K états, il existe un piège d'au plus $K + 1$ nœuds.
- Donc un automate qui explore tous les graphes d'au plus K sommets a au moins K états.
- DFS est optimal en mémoire : $\Theta(D \log \Delta)$ bits

DFS est optimal

Algorithme (borne supérieure)

- Depth-first search (DFS) de profondeur croissante
- Mémoire : pile des numéros de port menant à la racine
  $\longrightarrow O(D \log \Delta) bits$

Borne inférieure

Attacher un arbre au piège pour réduire le diamètre.

DFS est optimal

Algorithme (borne supérieure)

- Depth-first search (DFS) de profondeur croissante
- Mémoire : pile des numéros de port menant à la racine $\longrightarrow O(D \log \Delta) bits$

Borne inférieure

Attacher un arbre au piège pour réduire le diamètre.

Plan

1 Introduction

2 Faisabilité

3 Minimiser la mémoire

4 Coloriage par un oracle
- Idées de base
- Trois couleurs suffisent
- Seulement deux couleurs ?

5 Autres modèles

Colorier les nœuds

Un oracle colorie (étiquette) les nœuds pour aider les automates.

Idée de base

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

Etiquetage amélioré

seulement indiquer l'arête menant au père $\longrightarrow \Delta$ couleurs

Colorier les nœuds

Un oracle colorie (étiquette) les nœuds pour aider les automates.

Idée de base

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

Etiquetage amélioré

seulement indiquer l'arête menant au père $\longrightarrow \Delta$ couleurs

Colorier les nœuds

Un oracle colorie (étiquette) les nœuds pour aider les automates.

Idée de base

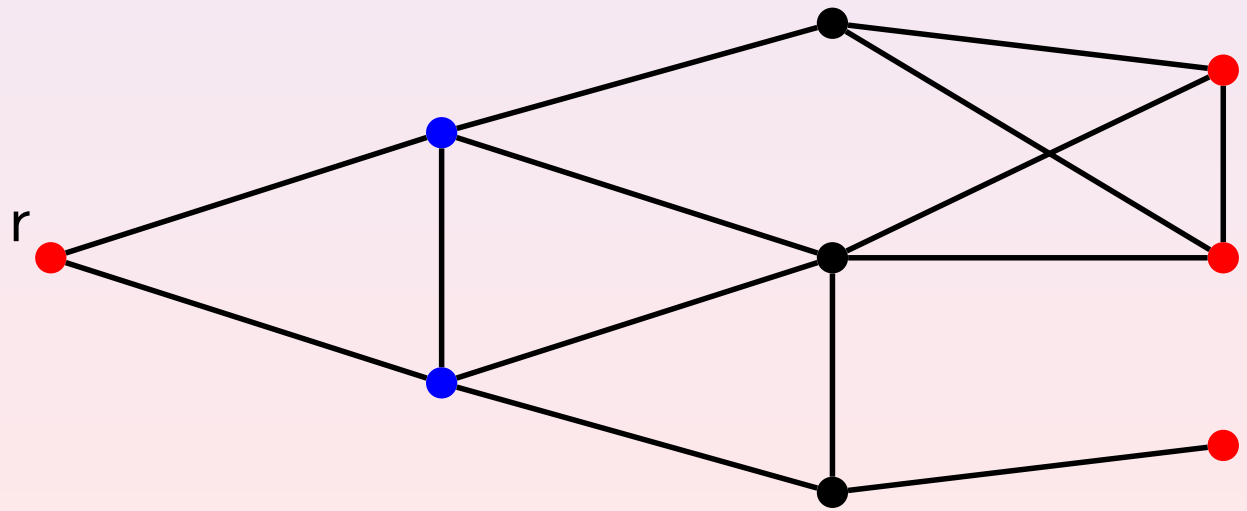
arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

Etiquetage amélioré

seulement indiquer l'arête menant au père $\longrightarrow$ Δ couleurs

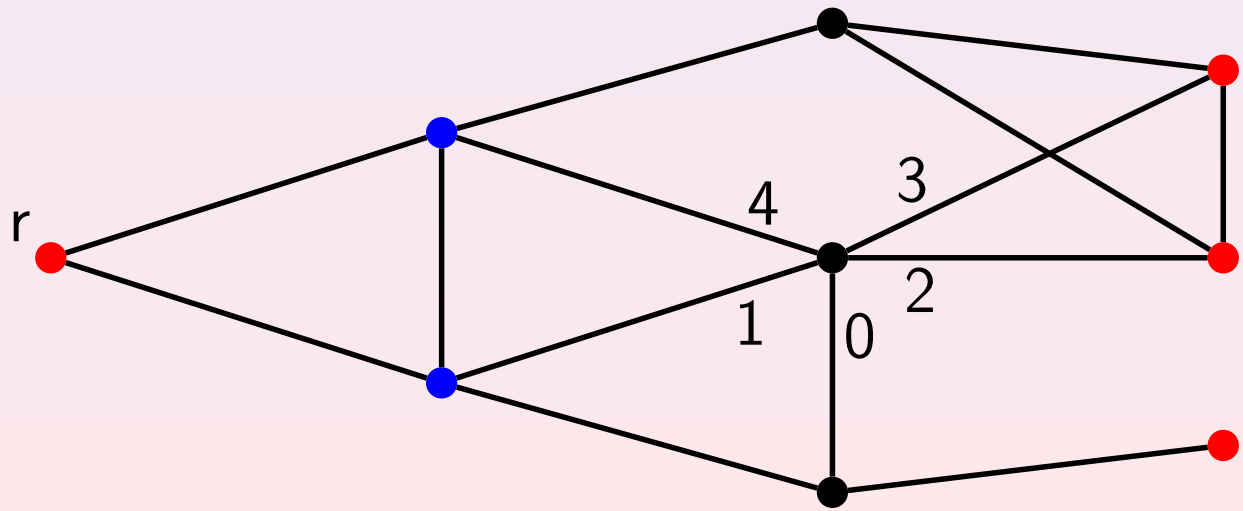
Trois couleurs suffisent

- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
  - distance $d \cong 0[n]$ rouge
  - distance $d \cong 1[n]$ bleu
  - distance $d \cong 2[n]$ noir



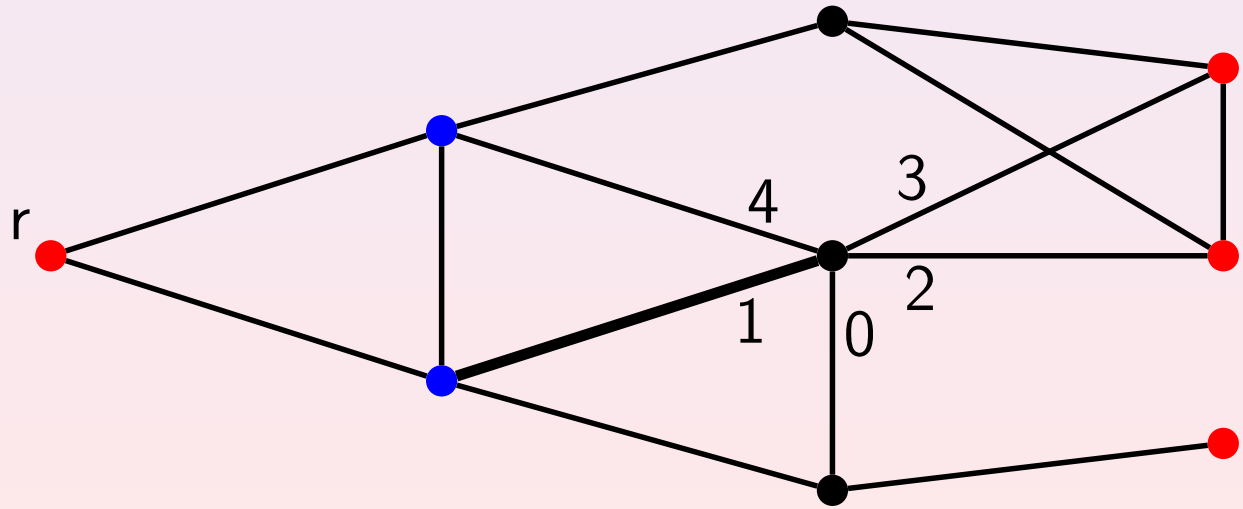
Trois couleurs suffisent

- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
  - distance $d \cong 0[n]$ rouge
  - distance $d \cong 1[n]$ bleu
  - distance $d \cong 2[n]$ noir



Trois couleurs suffisent

- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
  - distance $d \cong 0[n]$ rouge
  - distance $d \cong 1[n]$ bleu
  - distance $d \cong 2[n]$ noir



Seulement deux couleurs ?

- Layer 1: rouge
- Layer 2: bleu
- Layer 3: rouge
- Layer 4: rouge
- Layer 5: rouge
- Layer 6: bleu
- Layer 7: bleu
- Layer 8: bleu

Plan

1 Introduction

2 Faisabilité

3 Minimiser la mémoire

4 Coloriage par un oracle

5 Autres modèles
- Minimiser le temps
- Autres cadres
- Problème continu
- UTS, UXS

Minimiser le temps

Temps = nombre de traversées d'arêtes

Dudek, Jenkin, Milios, Wilkes, IEEE TRA 1991
Robotic exploration as graph construction

- Cartographie avec cailloux en temps $O(mn)$

Dudek, Milios, IEEE 1996
Competitive robot mapping with homogeneous markers

- Competitive ratio: cartographie / vérification de carte
- Avec un seul caillou, l'algorithme de DJMW est optimal (relaxed depth-one strategies)

Minimiser le temps

Temps = nombre de traversées d'arêtes

Dudek, Jenkin, Milios, Wilkes, IEEE TRA 1991
Robotic exploration as graph construction

- Cartographie avec cailloux en temps $O(mn)$

Dudek, Milios, IEEE 1996
Competitive robot mapping with homogeneous markers

- Competitive ratio: cartographie / vérification de carte
- Avec un seul caillou, l'algorithme de DJMW est optimal (relaxed depth-one strategies)

Minimiser le temps

Temps = nombre de traversées d'arêtes

Dudek, Jenkin, Milios, Wilkes, IEEE TRA 1991
Robotic exploration as graph construction

- Cartographie avec cailloux en temps $O(mn)$

Deng, Mirzaian, IEEE 1996
Competitive robot mapping with homogeneous markers

- Competitive ratio: cartographie / vérification de carte
- Avec un seul caillou, l'algorithme de DJMW est optimal (relaxed depth-one strategies)

Autres cadres

Autres cadres

- Graphes étiquetés
- Graphes géométriques
- Graphes orientés
- Exploration par morceaux

Autres cadres

- Graphes étiquetés
- Graphes géométriques
- Graphes orientés
- Exploration par morceaux

Autres cadres

- Graphes étiquetés
- Graphes géométriques
- Graphes orientés
- Exploration par morceaux

Autres cadres

- Graphes étiquetés
- Graphes géométriques
- Graphes orientés
- Exploration par morceaux

Pièces avec obstacles

Deng, Kameda, Papadimitriou, JACM 1998

How to learn an unknown environment I: the rectilinear case

Pièces avec obstacles

Deng, Kameda, Papadimitriou, JACM 1998
How to learn an unknown environment I: the rectilinear case

Pièces avec obstacles

Deng, Kameda, Papadimitriou, JACM 1998
How to learn an unknown environment I: the rectilinear case

- GT : Gallery Tour
  Visiter tout le musée, depuis l'entrée jusqu'à la sortie
- WR : Watchman's Route
  Passer par tous les recoins du musée et revenir au point de départ

Pièces avec obstacles

Deng, Kameda, Papadimitriou, JACM 1998
How to learn an unknown environment I: the rectilinear case

- GT : Gallery Tour
  Visiter tout le musée, depuis l'entrée jusqu'à la sortie
- WR : Watchman's Route
  Passer par tous les recoins du musée et revenir au point de départ

UTS, UXS

UTS = Universal Traversal Sequence

Séquence de numéros de port, i.e., d'arêtes à prendre

UXS = Universal Exploration Sequence

Séquence d'incréments de numéros de port

"0" signifie "revenir en arrière"

"1" signifie "prendre la première à droite" par exemple

But

- minimiser la taille de la séquence
- minimiser la quantité de mémoire pour la construire

UTS, UXS

UTS = Universal Traversal Sequence

Séquence de numéros de port, i.e., d'arêtes à prendre

UXS = Universal Exploration Sequence

Séquence d'incréments de numéros de port
"0" signifie "revenir en arrière"
"1" signifie "prendre la première à droite" par exemple

But

- minimiser la taille de la séquence
- minimiser la quantité de mémoire pour la construire

UTS, UXS

UTS = Universal Traversal Sequence

Séquence de numéros de port, i.e., d'arêtes à prendre

UXS = Universal Exploration Sequence

Séquence d'incréments de numéros de port
"0" signifie "revenir en arrière"
"1" signifie "prendre la première à droite" par exemple

But

- minimiser la taille de la séquence
- minimiser la quantité de mémoire pour la construire

Deuxième partie II

SL=L (Omer Reingold)

Plan

6 USTCON, SL et L
- Le problème USTCON
- USTCON, SL et L
- Algorithme

7 Augmenter la connectivité

8 Algorithme

9 Lien entre USTCON et UTS,UXS

10 Conclusion

Le problème USTCON

USTCON = undirected st-connectivity

Données

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

Question

s et t sont-ils dans la même composante connexe ?

Le problème USTCON

USTCON = undirected st-connectivity

Données

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

Question

s et t sont-ils dans la même composante connexe ?

Le problème USTCON

USTCON = undirected st-connectivity

Données

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

Question

s et t sont-ils dans la même composante connexe ?

USTCON, SL et L

L = Log-space

Class of problems solvable by deterministic log-space computations.

SL=Symmetric Log-space

Class of problems solvable by symmetric, non-deterministic, log-space computations.

USTCON est complet pour SL :

- USTCON $\in$ SL
- USTCON est au moins aussi difficile que tout problème de SL

USTCON $\in$ L $\Rightarrow$ SL=L

USTCON, SL et L

L = Log-space

Class of problems solvable by deterministic log-space computations.

SL=Symmetric Log-space

Class of problems solvable by symmetric, non-deterministic, log-space computations.

USTCON est complet pour SL :

- USTCON $\in$ SL
- USTCON est au moins aussi difficile que tout problème de SL

USTCON $\in$ L $\Rightarrow$ SL=L

USTCON, SL et L

L = Log-space

Class of problems solvable by deterministic log-space computations.

SL=Symmetric Log-space

Class of problems solvable by symmetric, non-deterministic, log-space computations.

USTCON est complet pour SL :

- USTCON $\in$ SL
- USTCON est au moins aussi difficile que tout problème de SL

USTCON $\in$ L $\Rightarrow$ SL=L

USTCON, SL et L

L = Log-space

Class of problems solvable by deterministic log-space computations.

SL=Symmetric Log-space

Class of problems solvable by symmetric, non-deterministic, log-space computations.

USTCON est complet pour SL :

- USTCON $\in$ SL
- USTCON est au moins aussi difficile que tout problème de SL

USTCON $\in$ L $\Rightarrow$ SL=L

Idées générales de l'algorithme

1. Transformer le graphe G en un graphe régulier G' de degré constant
2. Transformer G' en un expander G'' sans trop augmenter la taille ni le degré
3. Faire une exploration exhaustive de G'', de diamètre logarithmique

Idées générales de l'algorithme

1. Transformer le graphe G en un graphe régulier G' de degré constant
2. Transformer G' en un expander G'' sans trop augmenter la taille ni le degré
3. Faire une exploration exhaustive de G'', de diamètre logarithmique

Idées générales de l'algorithme

1. Transformer le graphe G en un graphe régulier G' de degré constant
2. Transformer G' en un expander G'' sans trop augmenter la taille ni le degré
3. Faire une exploration exhaustive de G'', de diamètre logarithmique

Idées générales de l'algorithme

1. Transformer le graphe G en un graphe régulier G' de degré constant
2. Transformer G' en un expander G'' sans trop augmenter la taille ni le degré
3. Faire une exploration exhaustive de G'', de diamètre logarithmique

Idées générales de l'algorithme

1. Transformer le graphe G en un graphe régulier G' de degré constant
2. Transformer G' en un expander G'' sans trop augmenter la taille ni le degré
3. Faire une exploration exhaustive de G'', de diamètre logarithmique

Plan

6 USTCON, SL et L

7 Augmenter la connectivité
- Notations
- Expander
- Opérations classiques
- Produit zig-zag

8 Algorithme

9 Lien entre USTCON et UTS,UXS

10 Conclusion

Notations

- $[N] = \{1, 2, \ldots, N\}$
- Un (N, D, λ)-graphe est un graphe à N sommets, D-régulier et dont la deuxième valeur propre (normalisée) vaut au plus λ.
- L'étiquetage est défini par : $Rot_G : [N] \times [D] \to [N] \times [D]$
  $Rot_G(v, i) = (w, j)$

Notations

- $[N] = \{1, 2, \ldots, N\}$
- Un (N, D, λ)-graphe est un graphe à N sommets, D-régulier et dont la deuxième valeur propre (normalisée) vaut au plus λ.
- L'étiquetage est défini par : $Rot_G : [N] \times [D] \longrightarrow [N] \times [D]$
  $Rot_G(v, i) = (w, j)$

Notations

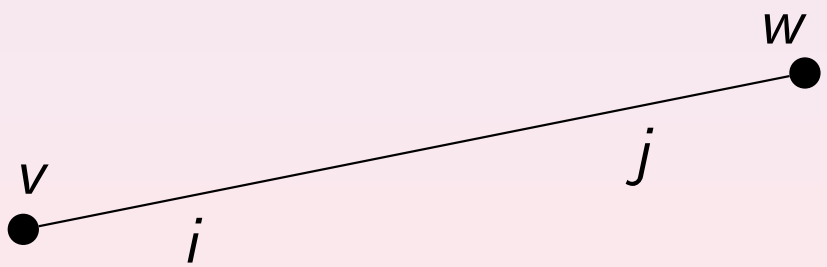
- $[N] = \{1, 2, \ldots, N\}$
- Un (N, D, λ)-graphe est un graphe à N sommets, D-régulier et dont la deuxième valeur propre (normalisée) vaut au plus λ.
- L'étiquetage est défini par : $Rot_G : [N] \times [D] \longrightarrow [N] \times [D]$
  $Rot_G(v, i) = (w, j)$

Notations

- $[N] = \{1, 2, \ldots, N\}$
- Un (N, D, λ)-graphe est un graphe à N sommets, D-régulier et dont la deuxième valeur propre (normalisée) vaut au plus λ.
- L'étiquetage est défini par : $Rot_G : [N] \times [D] \longrightarrow [N] \times [D]$
  $Rot_G(v, i) = (w, j)$

Notations

- $[N] = \{1, 2, \ldots, N\}$
- Un (N, D, λ)-graphe est un graphe à N sommets, D-régulier et dont la deuxième valeur propre (normalisée) vaut au plus λ.
- L'étiquetage est défini par : $Rot_G : [N] \times [D] \longrightarrow [N] \times [D]$
  $Rot_G(v, i) = (w, j)$



Expander : définitions intuitives

Graphe à grande connectivité et donc petit diamètre.

Vertex expansion

Le voisinage d'un ensemble $S \subset V$ a une taille strictement plus grande que $|S|$.

Edge expansion

Si δS est le nombre d'arêtes sortant d'un ensemble $S \subset V$, alors le rapport $\delta S/S$ est supérieure à une constante $\epsilon > 0$.

Spectral gap

Ecart entre la première et la deuxième valeur propre (en valeur absolue) de la matrice d'adjacence du graphe.

Toutes ces (pseudo) définitions sont plus ou moins équivalentes.

Expander : définitions intuitives

Graphe à grande connectivité et donc petit diamètre.

Vertex expansion

Le voisinage d'un ensemble $S \subset V$ a une taille strictement plus grande que $|S|$.

Edge expansion

Si δS est le nombre d'arêtes sortant d'un ensemble $S \subset V$, alors le rapport $\delta S/S$ est supérieure à une constante $\epsilon > 0$.

Spectral gap

Ecart entre la première et la deuxième valeur propre (en valeur absolue) de la matrice d'adjacence du graphe.

Toutes ces (pseudo) définitions sont plus ou moins équivalentes.

Expander : définitions intuitives

Graphe à grande connectivité et donc petit diamètre.

Vertex expansion

Le voisinage d'un ensemble $S \subset V$ a une taille strictement plus grande que $|S|$.

Edge expansion

Si δS est le nombre d'arêtes sortant d'un ensemble $S \subset V$, alors le rapport $\delta S/S$ est supérieure à une constante $\epsilon > 0$.

Spectral gap

Ecart entre la première et la deuxième valeur propre (en valeur absolue) de la matrice d'adjacence du graphe.

Toutes ces (pseudo) définitions sont plus ou moins équivalentes.

Expander : définitions intuitives

Graphe à grande connectivité et donc petit diamètre.

Vertex expansion

Le voisinage d'un ensemble $S \subset V$ a une taille strictement plus grande que $|S|$.

Edge expansion

Si δS est le nombre d'arêtes sortant d'un ensemble $S \subset V$, alors le rapport $\delta S/S$ est supérieure à une constante $\epsilon > 0$.

Spectral gap

Ecart entre la première et la deuxième valeur propre (en valeur absolue) de la matrice d'adjacence du graphe.

Toutes ces (pseudo) définitions sont plus ou moins équivalentes.

Expander : définitions intuitives

Graphe à grande connectivité et donc petit diamètre.

Vertex expansion

Le voisinage d'un ensemble $S \subset V$ a une taille strictement plus grande que $|S|$.

Edge expansion

Si δS est le nombre d'arêtes sortant d'un ensemble $S \subset V$, alors le rapport $\delta S/S$ est supérieure à une constante $\epsilon > 0$.

Spectral gap

Ecart entre la première et la deuxième valeur propre (en valeur absolue) de la matrice d'adjacence du graphe.

Toutes ces (pseudo) définitions sont plus ou moins équivalentes.

Mise à la puissance

Pour une puissance t, transforme un (N, D, λ)-graphe G en un (N, D^t, λ^t)-graphe G^t.

Définition

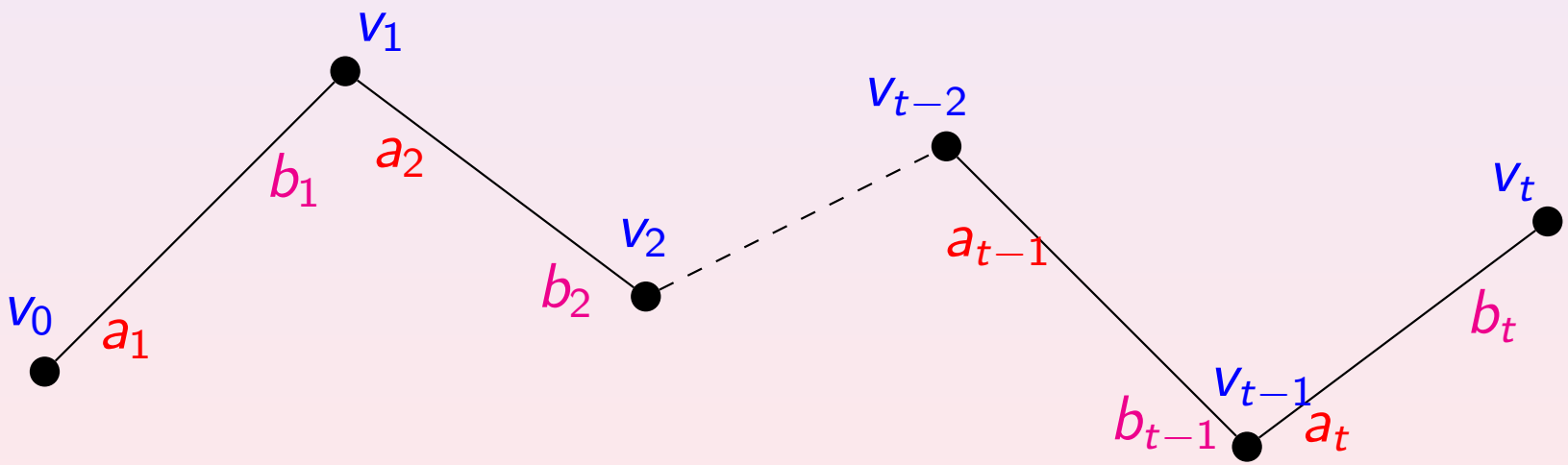
$$Rot_{G^t}(v_0, (a_1, a_2, \ldots, a_t)) = (v_t, (b_t, b_{t-1}, \ldots, b_1))$$

Mise à la puissance

Pour une puissance t, transforme un (N, D, λ)-graphe G en un (N, D^t, λ^t)-graphe G^t.

Définition

$Rot_{G^t}(v_0, (a_1, a_2, \ldots, a_t)) = (v_t, (b_t, b_{t-1}, \ldots, b_1))$



Produit de remplacement

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d+1, f(\lambda, \alpha))$-graphe G'.

Définition

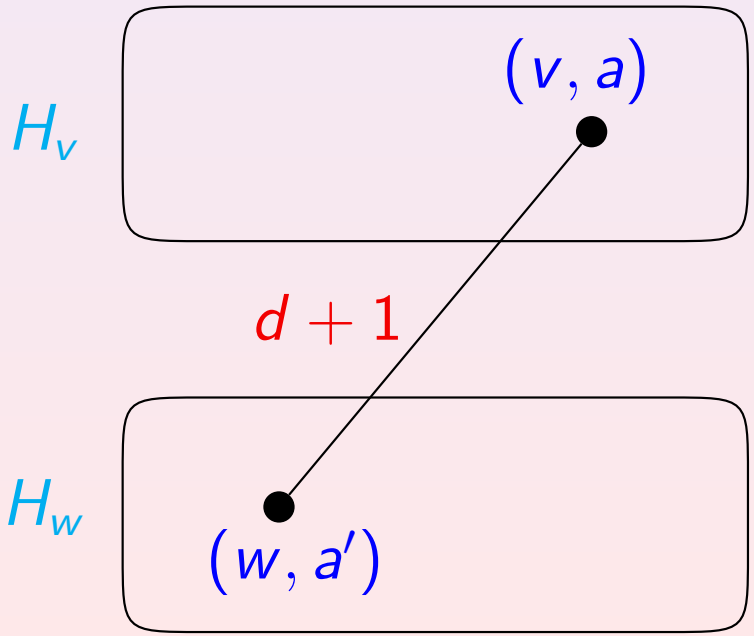
- $Rot_{G'}((v,a),i) = ((v,a'),i')$ avec $(a',i') = Rot_H(a,i)$
- $Rot_{G'}((v,a),d+1) = ((w,a'),d+1)$

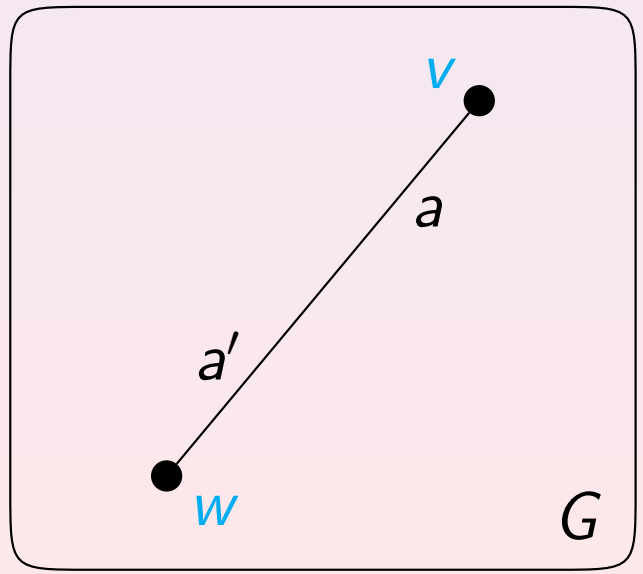
Produit de remplacement

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d+1, f(\lambda, \alpha))$-graphe G'.

Définition

- $Rot_{G'}((v,a),i) = ((v,a'),i')$ avec $(a',i') = Rot_H(a,i)$
- $Rot_{G'}((v,a),d+1) = ((w,a'),d+1)$





Produit zig-zag

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d^2, g(\lambda, \alpha)$-graphe $G \bar{z} H$.

Définition

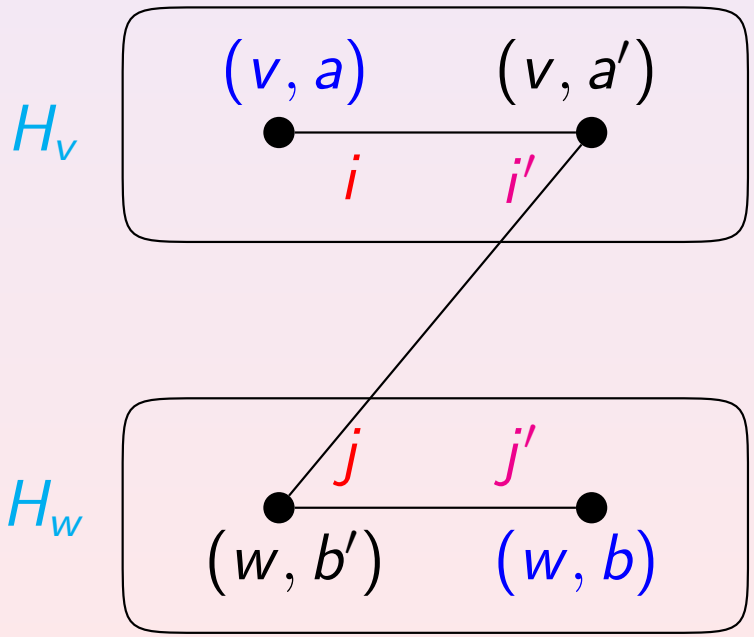
$$Rot_{G \bar{z} H}((v, a), (i, j)) = ((w, b), (i', j'))$$

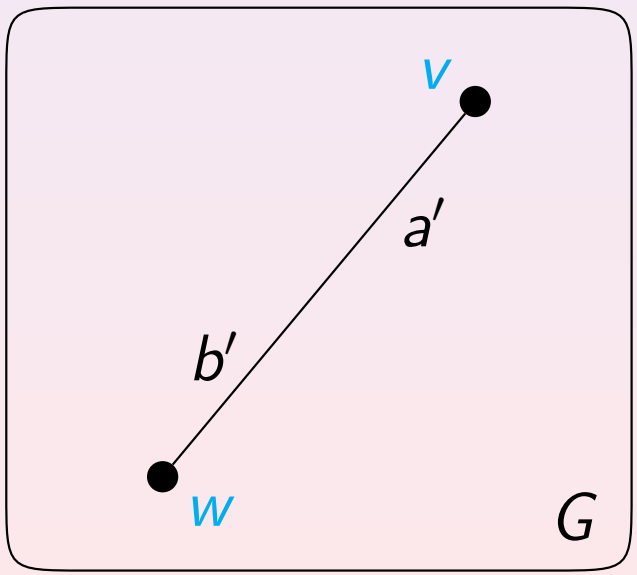
Produit zig-zag

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d^2, g(\lambda, \alpha)$-graphe $G \bar{z} H$.

Définition

$Rot_{G \bar{z} H}((v, a), (i, j)) = ((w, b), (j', i'))$





Propriété du produit zig-zag

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d^2, g(\lambda, \alpha)$-graphe $G \bar{z} H$.

Propriété (Reingold, Vadhan, Wigderson, FOCS 2000)

$$g(\lambda, \alpha) = 1/2(1 - \alpha^2)\lambda + 1/2\sqrt{(1 - \alpha^2)^2\lambda^2 + 4\alpha^2}$$

Corollaire

$$1 - \lambda(G \bar{z} H) \geq 1/2(1 - \alpha^2)(1 - \lambda)$$

Propriété du produit zig-zag

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d^2, g(\lambda, \alpha)$-graphe $G \bar{z} H$.

Propriété (Reingold, Vadhan, Wigderson, FOCS 2000)

$$g(\lambda, \alpha) = 1/2(1 - \alpha^2)\lambda + 1/2\sqrt{(1 - \alpha^2)^2\lambda^2 + 4\alpha^2}$$

Corollaire

$$1 - \lambda(G \bar{z} H) \geq 1/2(1 - \alpha^2)(1 - \lambda)$$

Propriété du produit zig-zag

Transforme un (N, D, λ)-graphe G et un (D, d, α)-graphe H en un $(N \cdot D, d^2, g(\lambda, \alpha)$-graphe $G \bar{z} H$.

Propriété (Reingold, Vadhan, Wigderson, FOCS 2000)

$$g(\lambda, \alpha) = 1/2(1 - \alpha^2)\lambda + 1/2\sqrt{(1 - \alpha^2)^2\lambda^2 + 4\alpha^2}$$

Corollaire

$$1 - \lambda(G \bar{z} H) \geq 1/2(1 - \alpha^2)(1 - \lambda)$$

Plan

6 USTCON, SL et L

7 Augmenter la connectivité

8 Algorithme
- Etapes
- Synthèse

9 Lien entre USTCON et UTS,UXS

10 Conclusion

Préparation du graphe

Soit $d > 3$ une constante.
On transforme le graphe G à N sommets en un (N^2, d^{16}, λ)-graphe G_{reg} avec $\lambda < 1$.

Construction de G_{reg}

- Chaque sommet est remplacé par un cycle C_N dont les arêtes sont étiquetées 1 et 2.
- Si $\{v, w\} \in E(G)$, alors $Rot_{G_{reg}}((v, w), 3) = ((w, v), 3)$.
- Pour $3 < i < d^{16}$, on crée des boucles.

Grâce aux boucles, chaque composante connexe de G_{reg} est un (N', d^{16}, λ)-graphe.

Préparation du graphe

Soit $d > 3$ une constante.
On transforme le graphe G à N sommets en un (N^2, d^{16}, λ)-graphe G_{reg} avec $\lambda < 1$.

Construction de G_{reg}

- Chaque sommet est remplacé par un cycle C_N dont les arêtes sont étiquetées 1 et 2.
- Si $\{v, w\} \in E(G)$, alors $Rot_{G_{reg}}((v, w), 3) = ((w, v), 3)$.
- Pour $3 < i < d^{16}$, on crée des boucles.

Grâce aux boucles, chaque composante connexe de G_{reg} est un (N', d^{16}, λ)-graphe.

Préparation du graphe

Soit $d > 3$ une constante.
On transforme le graphe G à N sommets en un (N^2, d^{16}, λ)-graphe G_{reg} avec $\lambda < 1$.

Construction de G_{reg}

- Chaque sommet est remplacé par un cycle C_N dont les arêtes sont étiquetées 1 et 2.
- Si $\{v, w\} \in E(G)$, alors $Rot_{G_{reg}}((v, w), 3) = ((w, v), 3)$.
- Pour $3 < i < d^{16}$, on crée des boucles.

Grâce aux boucles, chaque composante connexe de G_{reg} est un (N', d^{16}, λ)-graphe.

Transformation principale

1. $G_0 = G_{reg}$
2. $G_{i+1} = (G_i \bar{z} H)^8$
3. $G_\ell = \mathcal{T}(G, H)$ avec $\ell = O(\log N)$

Lemme

Si $\lambda(H) \leq 1/2$ et G est connexe bipartie, alors $\lambda(\mathcal{T}(G, H)) \leq 1/2$.

G_ℓ est un $(N^2(d^{16})^\ell, d^{16}, 1/2)$-graphe.

Transformation principale

1. $G_0 = G_{reg}$
2. $G_{i+1} = (G_i \bar{z} H)^8$
3. $G_\ell = \mathcal{T}(G, H)$ avec $\ell = O(\log N)$

Lemme

Si $\lambda(H) \leq 1/2$ et G est connexe bipartie, alors $\lambda(\mathcal{T}(G, H)) \leq 1/2$.

G_ℓ est un $(N^2(d^{16})^\ell, d^{16}, 1/2)$-graphe.

Transformation principale

1. $G_0 = G_{reg}$
2. $G_{i+1} = (G_i \bar{z} H)^8$
3. $G_\ell = \mathcal{T}(G, H)$ avec $\ell = O(\log N)$

Lemme

Si $\lambda(H) \leq 1/2$ et G est connexe bipartie, alors $\lambda(\mathcal{T}(G, H)) \leq 1/2$.

G_ℓ est un $(N^2(d^{16})^\ell, d^{16}, 1/2)$-graphe.

Diamètre de l'expander

$\lambda(G) < 1 \Rightarrow \exists\ \epsilon > 0$ G a une vertex expansion de $1 + \epsilon$.

Vertex expansion de $1 + \epsilon$

Pour tout $S \subset V$ de taille au plus $N/2$, au moins $(1 + \epsilon)|S|$ sommets sont connectés à un sommet de S.

Diamètre logarithmique

- On part de $S = \{s\}$ et on prend le voisinage, qui devient S et ainsi de suite $(S \leftarrow \mathit{voisinage}(S))$.
- A l'étape i, on a $|S| \geq (1 + \epsilon)^i$.
- Donc au moins $N/2$ sommets sont à distance $O(\log N)$.
- Diamètre du graphe en $O(\log N)$.

Diamètre de l'expander

$\lambda(G) < 1 \Rightarrow \exists\ \epsilon > 0$ G a une vertex expansion de $1 + \epsilon$.

Vertex expansion de $1 + \epsilon$

Pour tout $S \subset V$ de taille au plus $N/2$, au moins $(1 + \epsilon)|S|$ sommets sont connectés à un sommet de S.

Diamètre logarithmique

- On part de $S = \{s\}$ et on prend le voisinage, qui devient S et ainsi de suite ($S \leftarrow \mathit{voisinage}(S)$).
- A l'étape i, on a $|S| \geq (1 + \epsilon)^i$.
- Donc au moins $N/2$ sommets sont à distance $O(\log N)$.
- Diamètre du graphe en $O(\log N)$.

Diamètre de l'expander

$\lambda(G) < 1 \Rightarrow \exists\ \epsilon > 0$ G a une vertex expansion de $1 + \epsilon$.

Vertex expansion de $1 + \epsilon$

Pour tout $S \subset V$ de taille au plus $N/2$, au moins $(1 + \epsilon)|S|$ sommets sont connectés à un sommet de S.

Diamètre logarithmique

- On part de $S = \{s\}$ et on prend le voisinage, qui devient S et ainsi de suite ($S \leftarrow \mathit{voisinage}(S)$).
- A l'étape i, on a $|S| \geq (1 + \epsilon)^i$.
- Donc au moins $N/2$ sommets sont à distance $O(\log N)$.
- Diamètre du graphe en $O(\log N)$.

Récapitulatif de l'algorithme

1. Transformer le graphe G en un graphe régulier G_{reg} de degré constant
2. Transformer G_{reg} en un expander G_ℓ par produit zig-zag et mise à la puissance
3. Faire une exploration exhaustive de G_ℓ, de diamètre logarithmique

Théorème 1

USTCON $\in$ SL

Récapitulatif de l'algorithme

1. Transformer le graphe G en un graphe régulier G_{reg} de degré constant
2. Transformer G_{reg} en un expander G_ℓ par produit zig-zag et mise à la puissance
3. Faire une exploration exhaustive de G_ℓ, de diamètre logarithmique

Théorème 1

$\text{USTCON} \in \text{SL}$

Plan

6 USTCON, SL et L

7 Augmenter la connectivité

8 Algorithme

9 Lien entre USTCON et UTS,UXS

10 Conclusion

Chercher un chemin

Le calcul de $Rot_{G_\ell}(\bar{v}, \bar{a})$ peut fournir un chemin sous jacent dans $G_0 = G_{reg}$ et donc dans G.

Théorème 2

L'algorithme pour USTCON peut aussi fournir un chemin entre s et t.

Etiquetage π-consistant

$\forall v, w \in [N]\ \forall i, j \in [D]\ Rot(v, i) = (w, j) \Rightarrow j = \pi(i)$

Chercher un chemin

Le calcul de $Rot_{G_\ell}(\bar{v}, \bar{a})$ peut fournir un chemin sous jacent dans $G_0 = G_{reg}$ et donc dans G.

Théorème 2

L'algorithme pour USTCON peut aussi fournir un chemin entre s et t.

Etiquetage π-consistant

$$\forall v, w \in [N]\ \forall i, j \in [D]\ Rot(v, i) = (w, j) \Rightarrow j = \pi(i)$$

Chercher un chemin

Le calcul de $Rot_{G_\ell}(\bar{v}, \bar{a})$ peut fournir un chemin sous jacent dans $G_0 = G_{reg}$ et donc dans G.

Théorème 2

L'algorithme pour USTCON peut aussi fournir un chemin entre s et t.

Etiquetage π-consistant

$\forall v, w \in [N]\ \forall i, j \in [D]\ Rot(v, i) = (w, j) \Rightarrow j = \pi(i)$

Lien avec UTS, UXS

La séquence de numéros des arêtes prises par l'algorithme ne dépend pas vraiment du graphe, et ne dépend que de π si le graphe est π-consistant.

Corollaire 1

On peut construire en log-space une UTS pour les graphes à étiquetage π-consistant.

Koucky a développé une méthode pour transformer une UTS en une UXS.

Corollaire 2

On peut construire en log-space une UXS pour tous les graphes.

Lien avec UTS, UXS

La séquence de numéros des arêtes prises par l'algorithme ne dépend pas vraiment du graphe, et ne dépend que de π si le graphe est π-consistant.

Corollaire 1

On peut construire en log-space une UTS pour les graphes à étiquetage π-consistant.

Koucky a développé une méthode pour transformer une UTS en une UXS.

Corollaire 2

On peut construire en log-space une UXS pour tous les graphes.

Lien avec UTS, UXS

La séquence de numéros des arêtes prises par l'algorithme ne dépend pas vraiment du graphe, et ne dépend que de π si le graphe est π-consistant.

Corollaire 1

On peut construire en log-space une UTS pour les graphes à étiquetage π-consistant.

Koucky a développé une méthode pour transformer une UTS en une UXS.

Corollaire 2

On peut construire en log-space une UXS pour tous les graphes.

Plan

6 USTCON, SL et L

7 Augmenter la connectivité

8 Algorithme

9 Lien entre USTCON et UTS,UXS

10 Conclusion

Problèmes ouverts

Hydre

- $\forall k$ une hydre à $(k+1)$ têtes $\succ$ une hydre à k têtes ?
- Existe-t-il k tel que l'hydre à k têtes soit universelle ?

USTCON, UTS, UXS

- Existe-t-il une UTS pour tous les graphes constructible en log-space ?
- Meilleure compréhension des UTS, UXS, automates, hydres.

Problèmes ouverts

Hydre

- $\forall k$ une hydre à $(k+1)$ têtes $\succ$ une hydre à k têtes ?
- Existe-t-il k tel que l'hydre à k têtes soit universelle ?

USTCON, UTS, UXS

- Existe-t-il une UTS pour tous les graphes constructible en log-space ?
- Meilleure compréhension des UTS, UXS, automates, hydres.