Exploration de graphes orientés avec peu de mémoire

David ILCINKAS, LRI Orsay

En collaboration avec Pierre FRAIGNIAUD

Exploration de graphes

- But
  - l'entité mobile doit traverser au moins une fois chaque arc du graphe inconnu

Exploration de graphes

- But
  - l'entité mobile doit traverser au moins une fois chaque arc du graphe inconnu
- Motivations
  - exploration ou cartographie d'endroits inaccessibles à l'homme
  - maintenance d'un réseau informatique
  - recherche d'informations (P2P, Web)

Exploration de graphes

- But
  - l'entité mobile doit traverser au moins une fois chaque arc du graphe inconnu
- Motivations
  - exploration ou cartographie d'endroits inaccessibles à l'homme
  - maintenance d'un réseau informatique
  - recherche d'informations (P2P, Web)
- Hypothèse
  - graphes fortement connexes
  - anonyme : manque d'information, senseurs limités

Graphes anonymes

Graphes anonymes

- Taille et topologie inconnues

Graphes anonymes

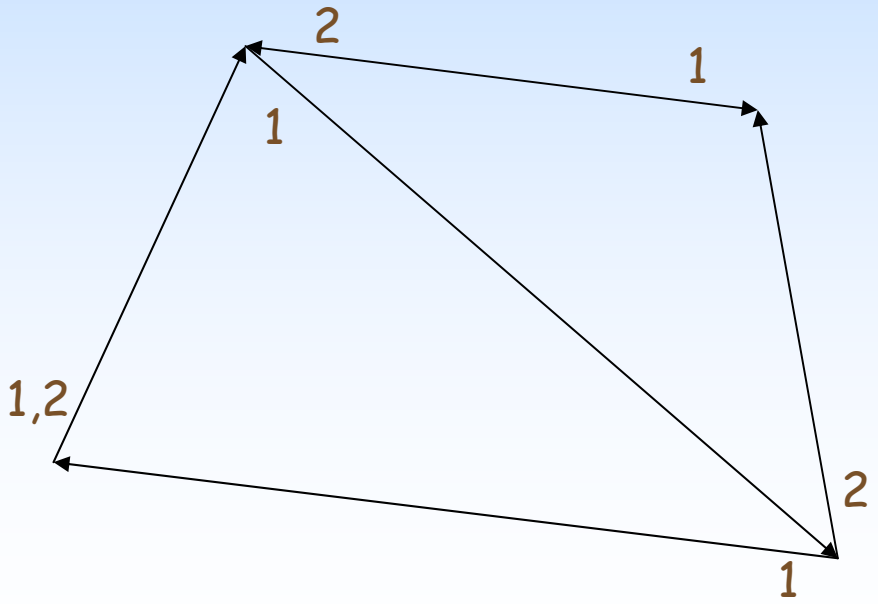
- Taille et topologie inconnues
- Pas d'étiquetage des sommets

Graphes anonymes

- Taille et topologie inconnues
- Pas d'étiquetage des sommets
- Numérotation locale des arcs

Graphes anonymes

- Taille et topologie inconnues
- Pas d'étiquetage des sommets
- Numérotation locale des arcs



Les différentes tâches

Les différentes tâches

- Exploration perpétuelle
  - en un temps fini, l'entité mobile doit explorer tous les arcs du graphe

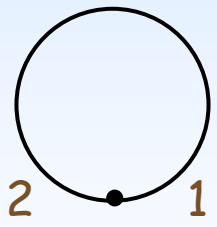
Les différentes tâches

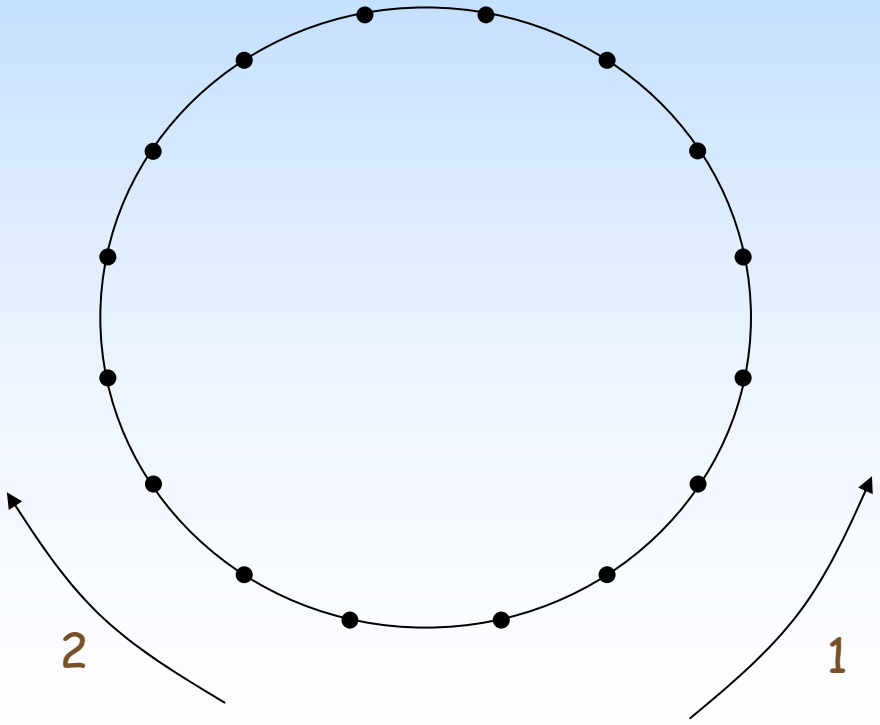
- Exploration perpétuelle
  - en un temps fini, l'entité mobile doit explorer tous les arcs du graphe
- Exploration avec arrêt
  - l'entité mobile doit s'arrêter en étant sûre d'avoir fini l'exploration

Les différentes tâches

- Exploration perpétuelle
  - en un temps fini, l'entité mobile doit explorer tous les arcs du graphe
- Exploration avec arrêt
  - l'entité mobile doit s'arrêter en étant sûre d'avoir fini l'exploration
- Exploration avec retour
  - l'entité mobile doit revenir s'arrêter sur le sommet de départ

Nécessité d'un marqueur pour l'arrêt



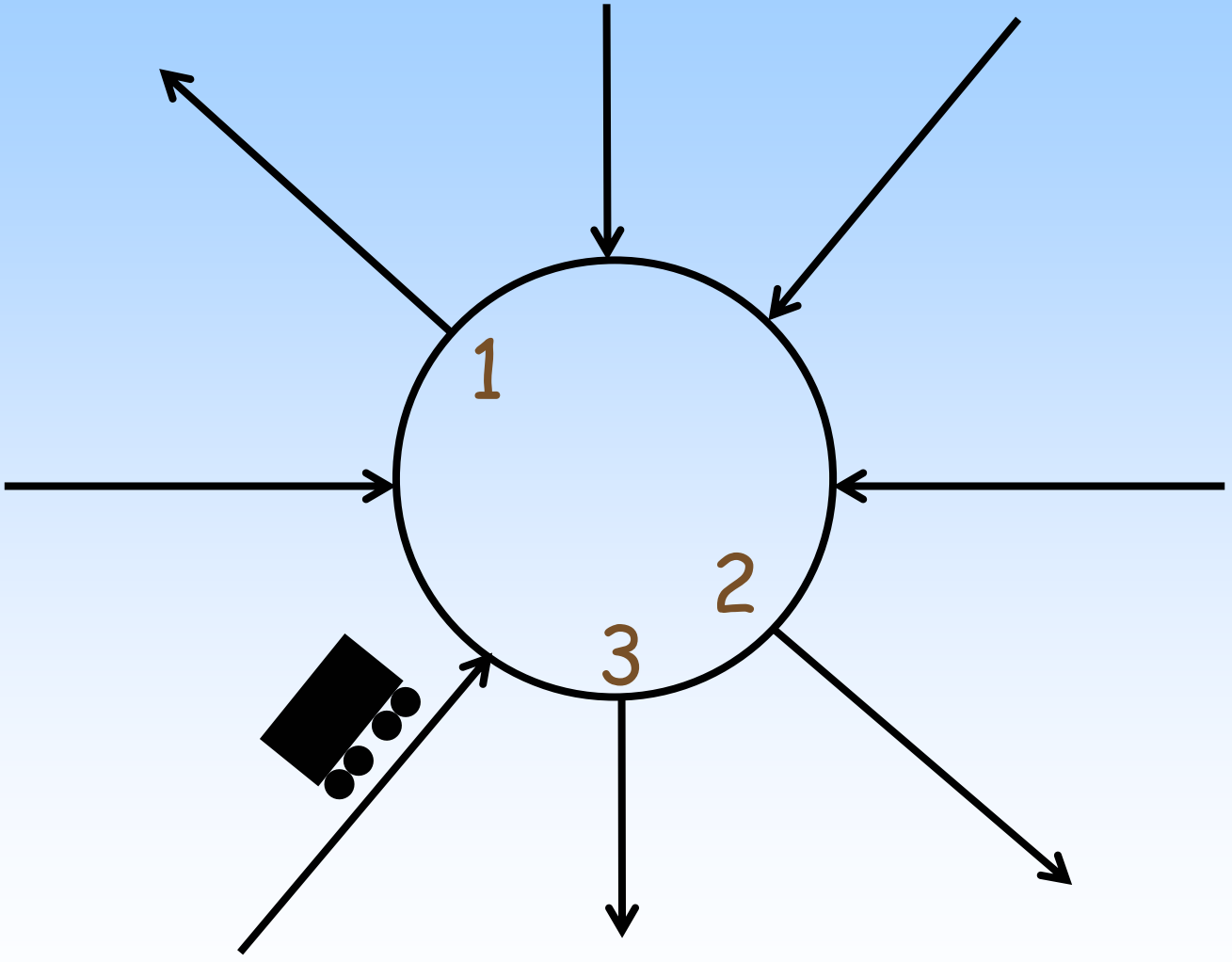


L'agent mobile

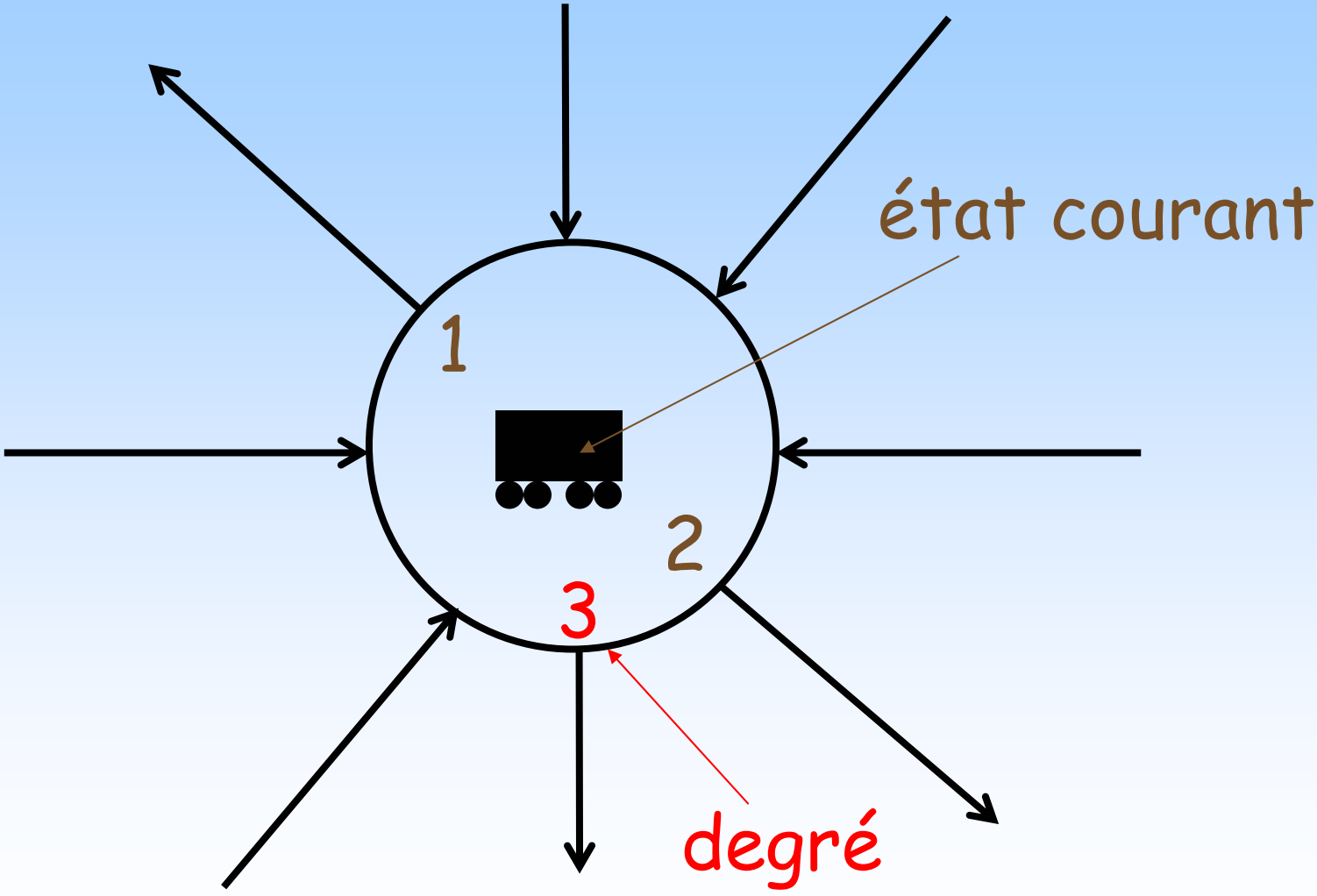
- Modèle Robot
  - robot avec k bits de mémoire

  → automate déterministe à $K=2^k$ états
  - possibilité de poser et de reprendre un caillou

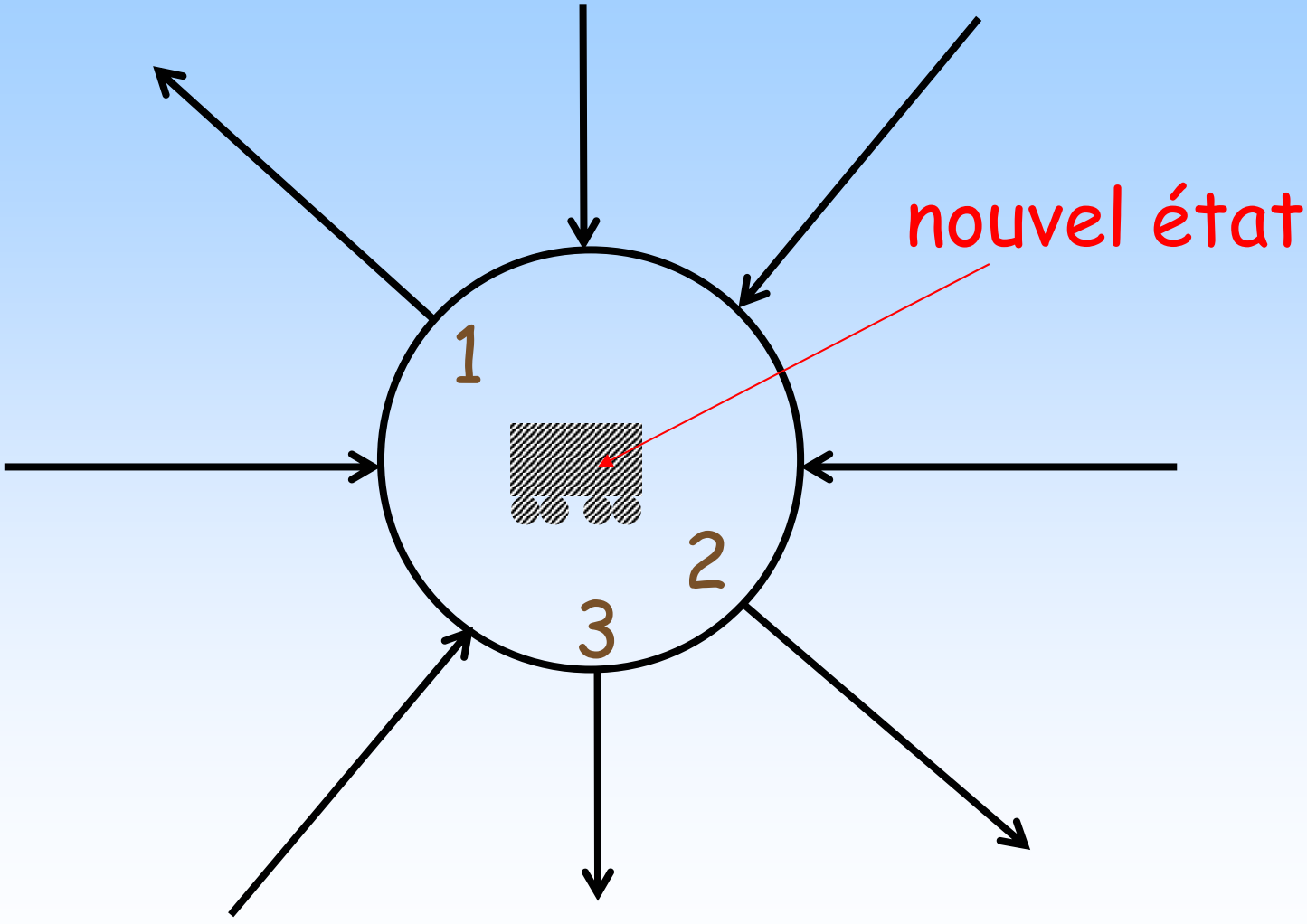
Actions du robot



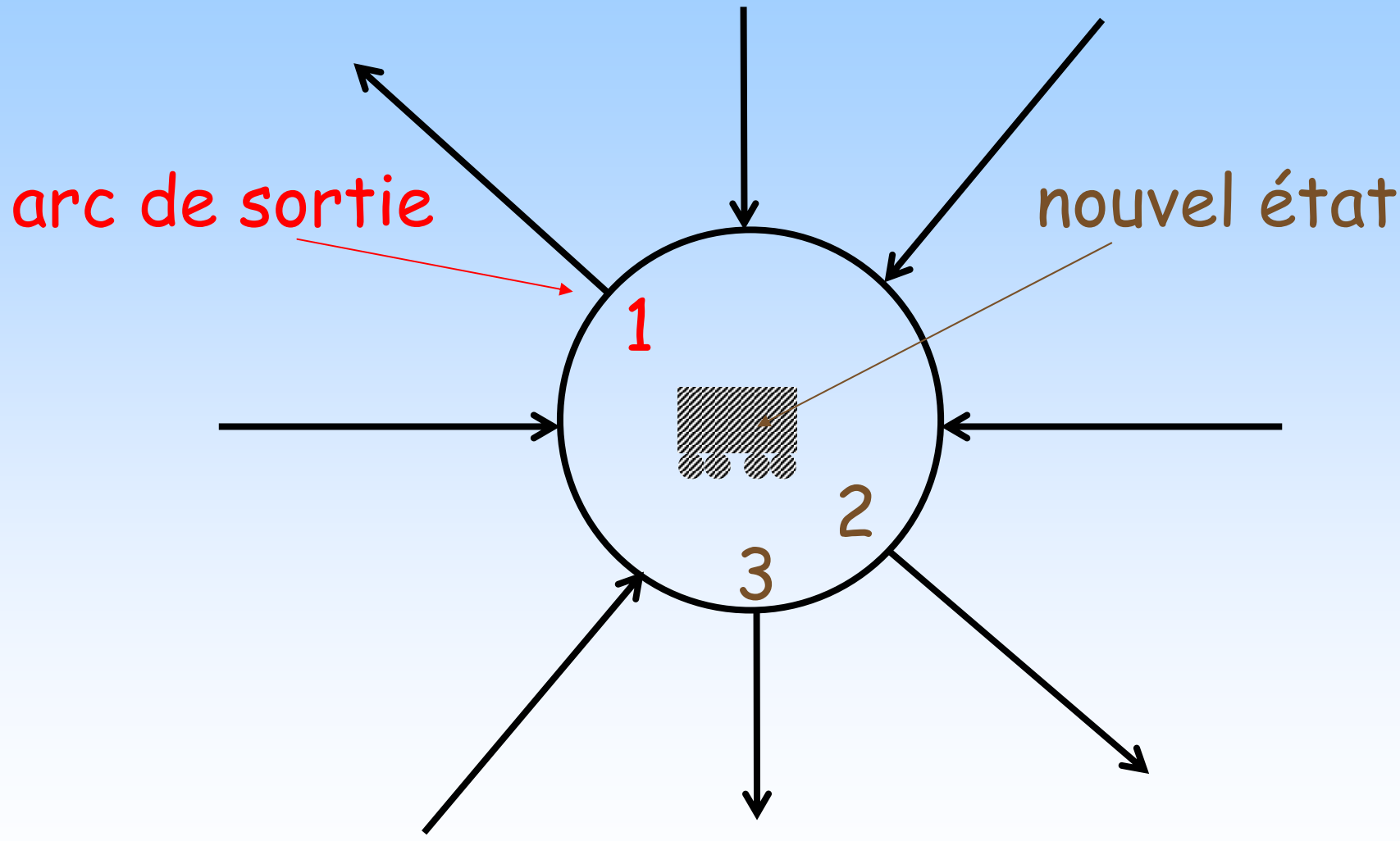
Actions du robot



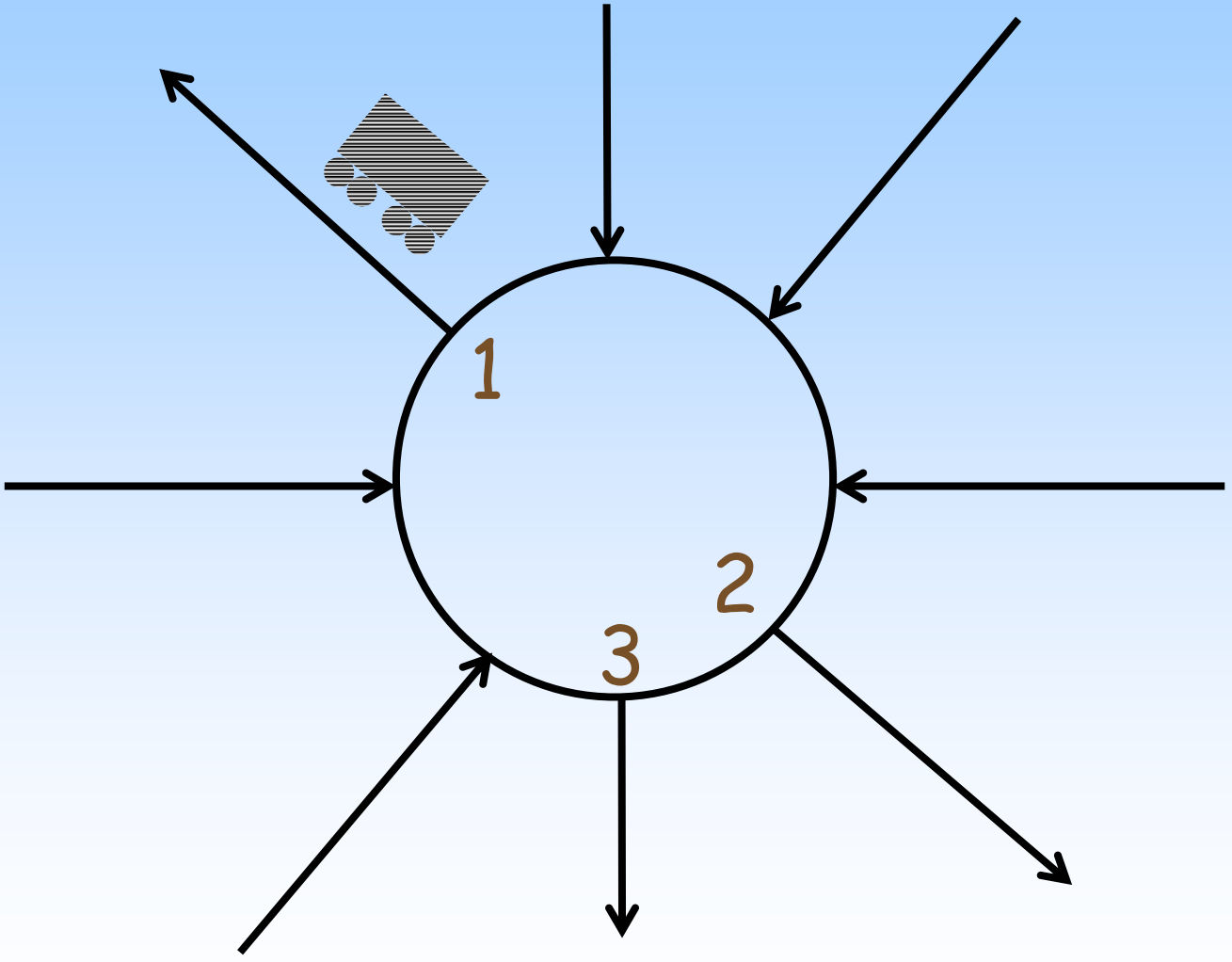
Actions du robot



Actions du robot



Actions du robot



Plan

- Références
- Borne inférieure

  $\Omega(n \log d)$
- Borne supérieure

  $O(nd \log n)$

Références

Références

- Bender, Slonim [FOCS '94]
  - exploration des graphes orientés par deux robots sans caillou en temps polynomial

Références

- Bender, Slonim [FOCS '94]
  - exploration des graphes orientés par deux robots sans caillou en temps polynomial
- Bender, Fernandez, Ron, Sahai, Vadhan [STOC '98]
  - l'exploration de graphes orientés en temps polynomial nécessite $\Omega(\log \log n)$ cailloux
  - algorithme d'exploration en temps polynomial par un robot avec $O(n^5 d \log n)$ bits de mémoire et $O(\log \log n)$ cailloux (une version optimisée n'utilise que $O(n^2 d \log n)$ bits)

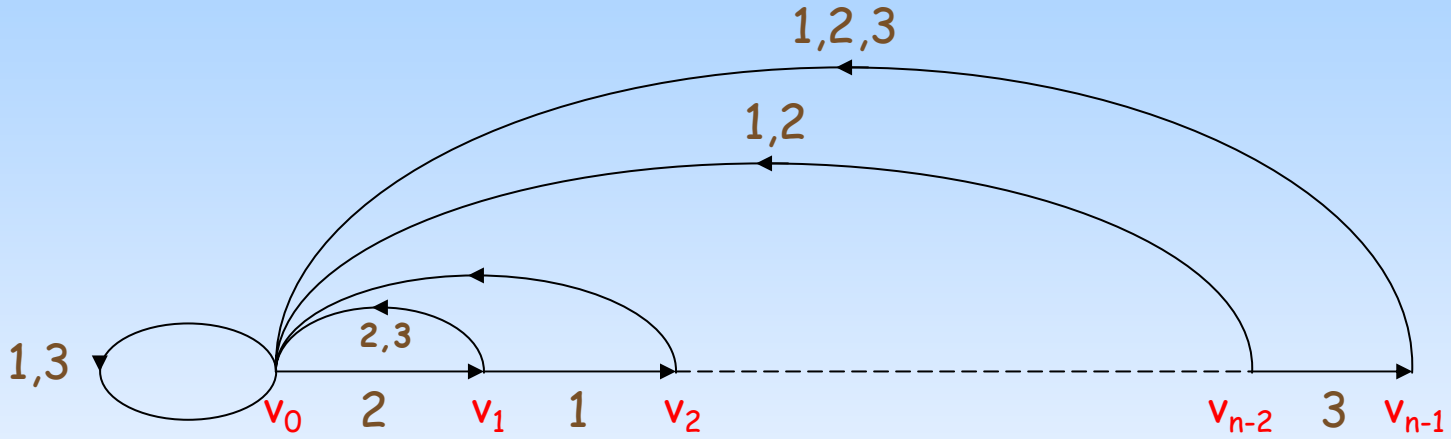
Borne inférieure : $\Omega(n \log d)$

- Théorème 1

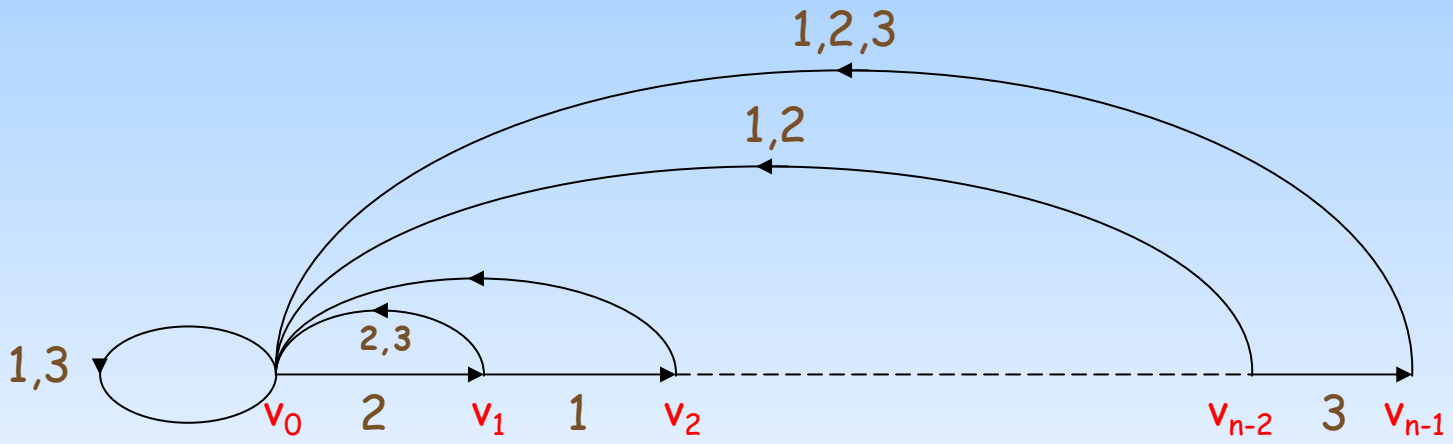
  L'exploration avec arrêt dans un graphe orienté de taille n et de degré sortant au plus $d > 2$, ne peut pas être accomplie par un robot avec moins de $\Omega(n \log d)$ bits de mémoire. Le résultat tient même si le robot dispose de n cailloux.

Preuve de $\Omega(n \log d)$

Preuve de $\Omega(n \log d)$

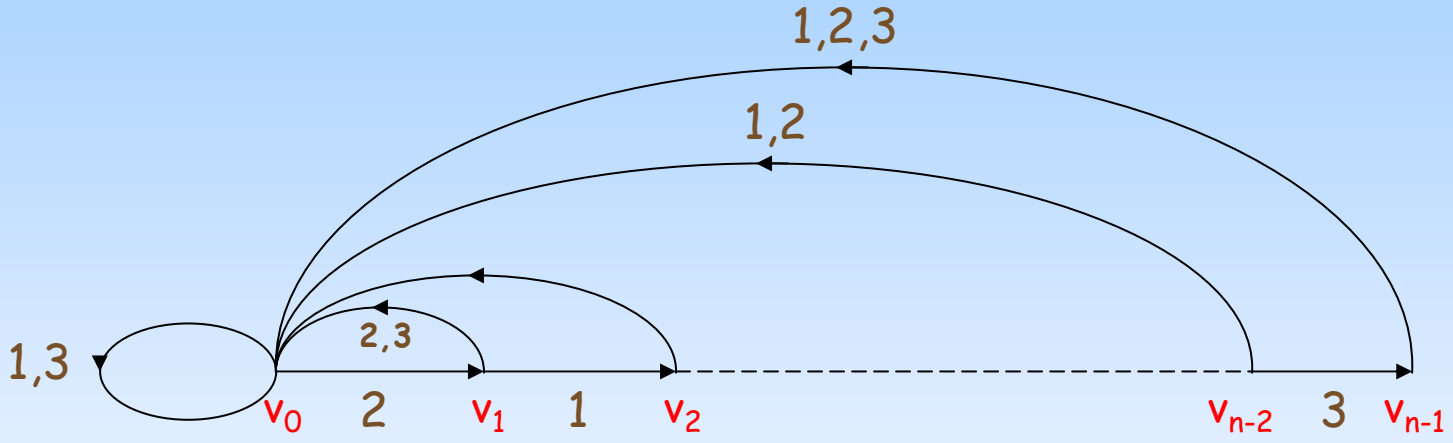


Preuve de $\Omega(n \log d)$



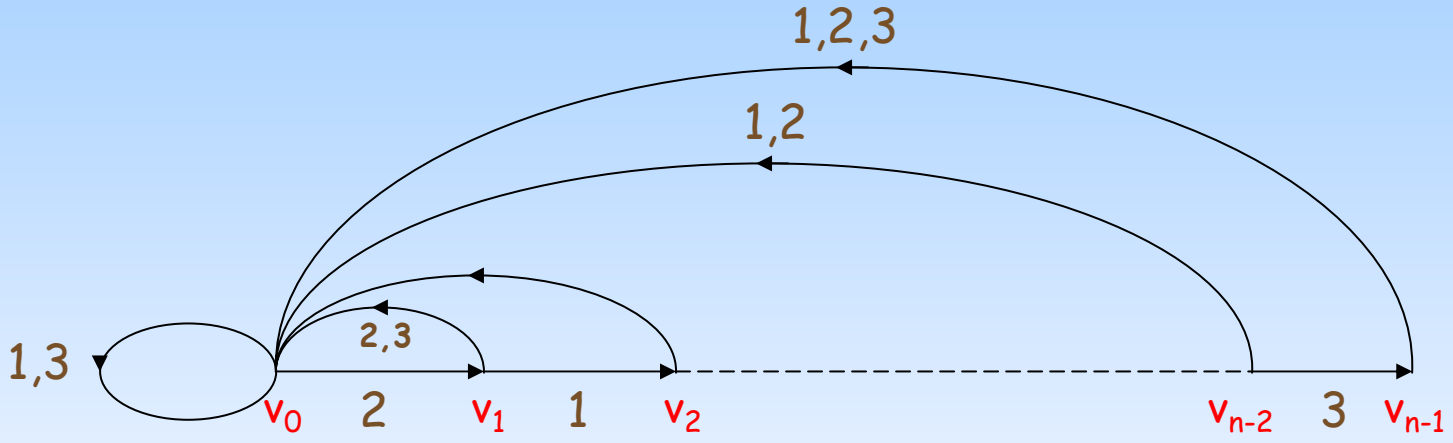
- nombre de verrous combinatoires : d^{n-1}

Preuve de $\Omega(n \log d)$



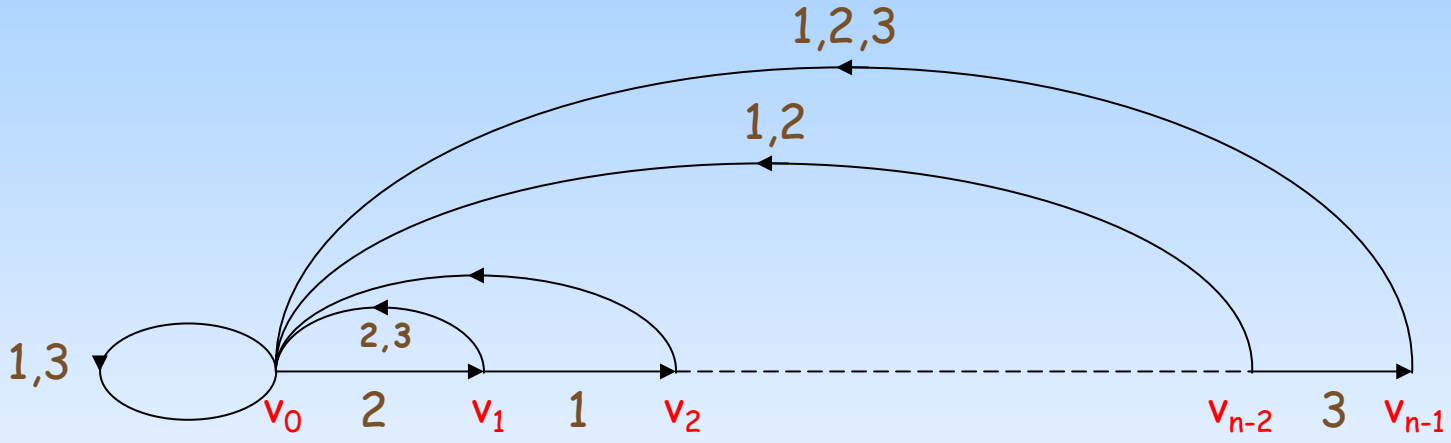
- nombre de verrous combinatoires : d^{n-1}
- nombre d'états du robot : 2^k (k bits de mémoire)

Preuve de $\Omega(n \log d)$



- nombre de verrous combinatoires : d^{n-1}
- nombre d'états du robot : 2^k (k bits de mémoire)
- disposition des cailloux : 2^n

Preuve de $\Omega(n \log d)$



- nombre de verrous combinatoires : d^{n-1}
- nombre d'états du robot : 2^k (k bits de mémoire)
- disposition des cailloux : 2^n
- il faut $2^k \times 2^n > d^{n-1} \rightarrow k > n \log(d/2)$

Borne supérieure : O(nd log n)

- Théorème 2

  Soit *G* un graphe orienté fortement connexe de taille n et de degré sortant d. L'algorithme teste-cartes résout le problème d'exploration avec arrêt et de cartographie par un robot de O(nd log n) bits de mémoire disposant d'un seul caillou.

Description de teste-cartes

Description de teste-cartes

- Idée de base
  - on teste toutes les cartes possibles

Description de teste-cartes

- Idée de base
  - on teste toutes les cartes possibles
- Vérification
  - on compare les vues du graphe depuis chaque sommet
  - en pratique, on pose le caillou et on se déplace dans le graphe

Points techniques

Points techniques

- Retrouver le caillou
  - pour la carte testée, on prend un chemin passant par tous les sommets

Points techniques

- Retrouver le caillou
  - pour la carte testée, on prend un chemin passant par tous les sommets
- Vérification
  - chemin passant par tous les arcs
  - vérification grâce au caillou

Points techniques

- Retrouver le caillou
  - pour la carte testée, on prend un chemin passant par tous les sommets
- Vérification
  - chemin passant par tous les arcs
  - vérification grâce au caillou
- Cas du sous-graphe
  - on vérifie le degré sortant de chaque sommet

Points techniques

- Retrouver le caillou
  - pour la carte testée, on prend un chemin passant par tous les sommets
- Vérification
  - chemin passant par tous les arcs
  - vérification grâce au caillou
- Cas du sous-graphe
  - on vérifie le degré sortant de chaque sommet
- Point de départ
  - on teste pour tous les sommets de départs
  - tableau de correspondance

Conclusion

- Résultats
  - Borne inférieure en $\Omega(n \log d)$
  - Borne supérieure en $O(nd \log n)$ bits
- Perspectives
  - Améliorer les bornes
  - Trouver un algorithme efficace en temps