Génération d'instances pour le benchmarking

Matthieu Pérotin

Laboratoire d'Informatique de Tours

25 janvier 2007

Le problème d'ordonnancement

- Une architecture matérielle peut être représentée par un graphe

Le problème d'ordonnancement

- Une architecture matérielle peut être représentée par un graphe
- Chaque nœud du graphe est une machine, caractérisée par les ressources qu'elle offre. Par exemple, sa capacité mémoire, la vitesse de son CPU, ...

Le problème d'ordonnancement

- Une architecture matérielle peut être représentée par un graphe
- Chaque nœud du graphe est une machine, caractérisée par les ressources qu'elle offre. Par exemple, sa capacité mémoire, la vitesse de son CPU, ...
- Chaque arc peut être valué par le débit du lien entre les deux machines, sa latence, ...

- Une application parallèle peut être représentée par un graphe orienté acyclique (DAG)

- Une application parallèle peut être représentée par un graphe orienté acyclique (DAG)
- Le graphe est un diagramme de précédence, chaque nœud représente une tâche.

- Une application parallèle peut être représentée par un graphe orienté acyclique (DAG)
- Le graphe est un diagramme de précédence, chaque nœud représente une tâche.
- Les caractéristiques des tâches sont généralement les mêmes que pour des processus classiques. Elles contiennent un certain nombre de demandes en ressources (mémoire, processeur, ...)

- Une application parallèle peut être représentée par un graphe orienté acyclique (DAG)
- Le graphe est un diagramme de précédence, chaque nœud représente une tâche.
- Les caractéristiques des tâches sont généralement les mêmes que pour des processus classiques. Elles contiennent un certain nombre de demandes en ressources (mémoire, processeur, ...)
- Chaque arête peut être valuée avec un volume d'information qui est échangé entre une tâche et ses successeurs.

- Une application parallèle peut être représentée par un graphe orienté acyclique (DAG)
- Le graphe est un diagramme de précédence, chaque nœud représente une tâche.
- Les caractéristiques des tâches sont généralement les mêmes que pour des processus classiques. Elles contiennent un certain nombre de demandes en ressources (mémoire, processeur, ...)
- Chaque arête peut être valuée avec un volume d'information qui est échangé entre une tâche et ses successeurs.
- On ne doit s'acquitter du coût de communication uniquement dans le cas où une tâche et son successeur ne sont pas affectés au même processeur.

Problème classique

- Ordonnancer les tâches sur les machines afin de minimiser le makespan

Problème classique

- Ordonnancer les tâches sur les machines afin de minimiser le makespan
- Rappel : $P||C_{max}$ NP Difficile au sens fort [Garey & Johnson 78]

Évaluer des algorithmes...

- Problème peu simple et comportant plusieurs aspects

Évaluer des algorithmes...

- Problème peu simple et comportant plusieurs aspects
- La création d'un simulateur

Évaluer des algorithmes...

- Problème peu simple et comportant plusieurs aspects
- La création d'un simulateur
- La création de jeux de données

Évaluer des algorithmes...

- Problème peu simple et comportant plusieurs aspects
- La création d'un simulateur
- La création de jeux de données
  - Pour la plateforme matérielle

Évaluer des algorithmes...

- Problème peu simple et comportant plusieurs aspects
- La création d'un simulateur
- La création de jeux de données
  - Pour la plateforme matérielle
  - Pour les applications

Créer un simulateur

- Cadre de travail grid computing, clusters, ...

Créer un simulateur

- Cadre de travail grid computing, clusters, ...
- Ou bien on a la chance de disposer d'une infrastructure matérielle permettant de faire ses tests

Créer un simulateur

- Cadre de travail grid computing, clusters, ...
- Ou bien on a la chance de disposer d'une infrastructure matérielle permettant de faire ses tests
- Ou bien on utilise des outils spécifiques (SimGrid)

Créer des jeux de données de plateformes matérielles

- Un problème théorique intéressant : étant donnée une application, quelle est l'infrastucture matérielle la plus pertinente pour son exécution ?

Créer des jeux de données de plateformes matérielles

- Un problème théorique intéressant : étant donnée une application, quelle est l'infrastucture matérielle la plus pertinente pour son exécution ?
- Le problème le plus fréquent est : étant donnée mon infrastucture matérielle, comment ordonnancer mes applications ?

Créer des jeux de données de plateformes matérielles

- Un problème théorique intéressant : étant donnée une application, quelle est l'infrastucture matérielle la plus pertinente pour son exécution ?
- Le problème le plus fréquent est : étant donnée mon infrastucture matérielle, comment ordonnancer mes applications ?
- La plateforme matérielle implique souvent des choix forts au niveau des applications (BOINC, MPI)

Créer des jeux de données d'application

- Le problème revient à générer des dags

Créer des jeux de données d'application

- Le problème revient à générer des dags
- Mais aussi des volumes

Créer des jeux de données d'application

- Le problème revient à générer des dags
- Mais aussi des volumes
- Comment le faire de façon sensée ?

Créer des jeux de données d'application

- Le problème revient à générer des dags
- Mais aussi des volumes
- Comment le faire de façon sensée ?
- Quels sont les critères caractérisant ces instances ?

Créer des jeux de données d'application

- Le problème revient à générer des dags
- Mais aussi des volumes
- Comment le faire de façon sensée ?
- Quels sont les critères caractérisant ces instances ?
- Comment évaluer des algorithmes ?

Buts d'un survey

- Selon [Kwok & Ahmad] dans Benchmarking the Task Graph Scheduling Algorithms

Buts d'un survey

- Selon [Kwok & Ahmad] dans Benchmarking the Task Graph Scheduling Algorithms
  1. Quelles sont les mesures de performance des algorithmes ?
     - Proximité de la solution optimale ? Temps d'exécution ?

Buts d'un survey

- Selon [Kwok & Ahmad] dans Benchmarking the Task Graph Scheduling Algorithms
  1. Quelles sont les mesures de performance des algorithmes ?
     - Proximité de la solution optimale ? Temps d'exécution ?
  2. Quels paramètres du problème affectent la performance ?
     - Comment caractériser les instances ?

Buts d'un survey

- Selon [Kwok & Ahmad] dans Benchmarking the Task Graph Scheduling Algorithms
  1. Quelles sont les mesures de performance des algorithmes ?
     - Proximité de la solution optimale ? Temps d'exécution ?
  2. Quels paramètres du problème affectent la performance ?
     - Comment caractériser les instances ?
  3. Quelles instances utiliser ?

Buts d'un survey

- Selon [Kwok & Ahmad] dans Benchmarking the Task Graph Scheduling Algorithms
  1. Quelles sont les mesures de performance des algorithmes ?
     - Proximité de la solution optimale ? Temps d'exécution ?
  2. Quels paramètres du problème affectent la performance ?
     - Comment caractériser les instances ?
  3. Quelles instances utiliser ?
  4. Pourquoi certains algorithmes fonctionnent-ils mieux que d'autres ?
     - Choix de conception de l'algorithme et caractéristiques

Générer uniformément

- Il semble peu pertinent de générer uniformément sur l'ensemble des graphes possibles

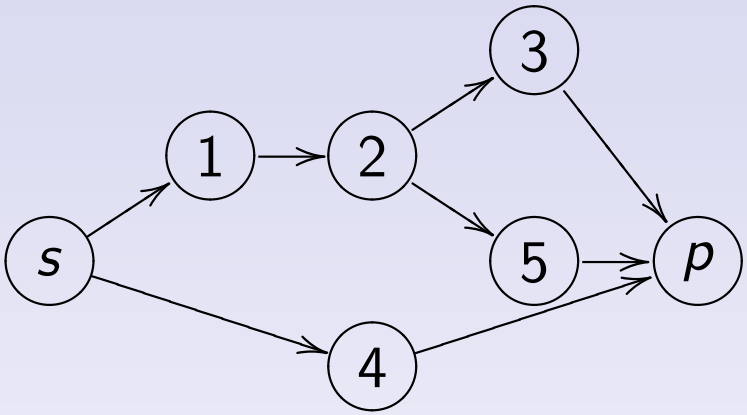
Générer uniformément

- Il semble peu pertinent de générer uniformément sur l'ensemble des graphes possibles
- On souhaite en général cibler certains types de graphes

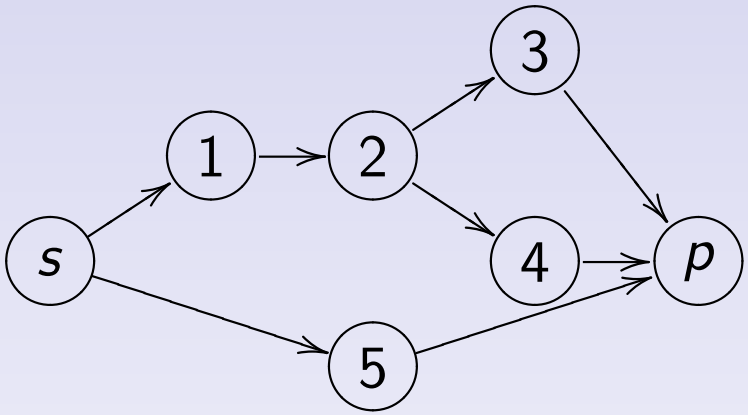
Générer uniformément

- Il semble peu pertinent de générer uniformément sur l'ensemble des graphes possibles
- On souhaite en général cibler certains types de graphes
- Définir une bonne taxonomie n'est pas facile

Caractériser les isomorphes



	2	3	4	5
1	1	1	–	1
2		1	–	1
3			–	–
4				–



	2	3	4	5
1	1	1	1	–
2		1	1	–
3			–	–
4				–

Combien d'isomorphes ?

- Un graphe à n sommets a $n!$ isomorphes

Combien d'isomorphes ?

- Un graphe à n sommets a $n!$ isomorphes
- C'est à dire autant que de numérotations possibles de ses sommets.

Combien d'isomorphes ?

- Un graphe à n sommets a $n!$ isomorphes
- C'est à dire autant que de numérotations possibles de ses sommets.
- Définition simpliste, qui ne reflète pas le nombre de matrices d'incidences uniques (exemple avec un graphe sans arête)

Métriques topologiques sur les graphes

- CI : Nombre minimal de réductions pour obtenir un graphe de précédence d'une seule arête

Métriques topologiques sur les graphes

- CI : Nombre minimal de réductions pour obtenir un graphe de précédence d'une seule arête
- OS : Nombre de relations de précédence sur nombre maximum de relations de précédence (n(n-1)/2)

Métriques topologiques sur les graphes

- CI : Nombre minimal de réductions pour obtenir un graphe de précédence d'une seule arête
- OS : Nombre de relations de précédence sur nombre maximum de relations de précédence (n(n-1)/2)
- CNC : Nombre d'arcs sur nombre de nœuds

Métriques topologiques sur les graphes

- CI : Nombre minimal de réductions pour obtenir un graphe de précédence d'une seule arête
- OS : Nombre de relations de précédence sur nombre maximum de relations de précédence (n(n-1)/2)
- CNC : Nombre d'arcs sur nombre de nœuds
- ...

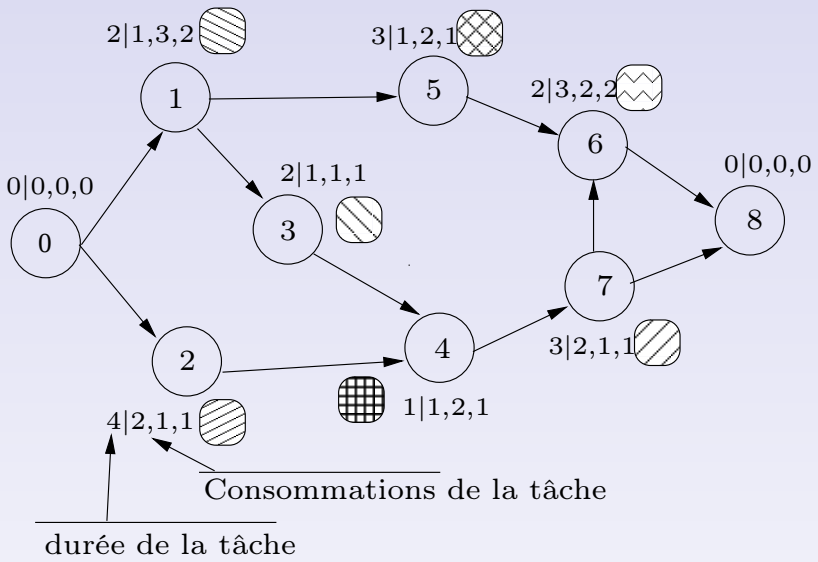
Métriques topologiques sur les graphes

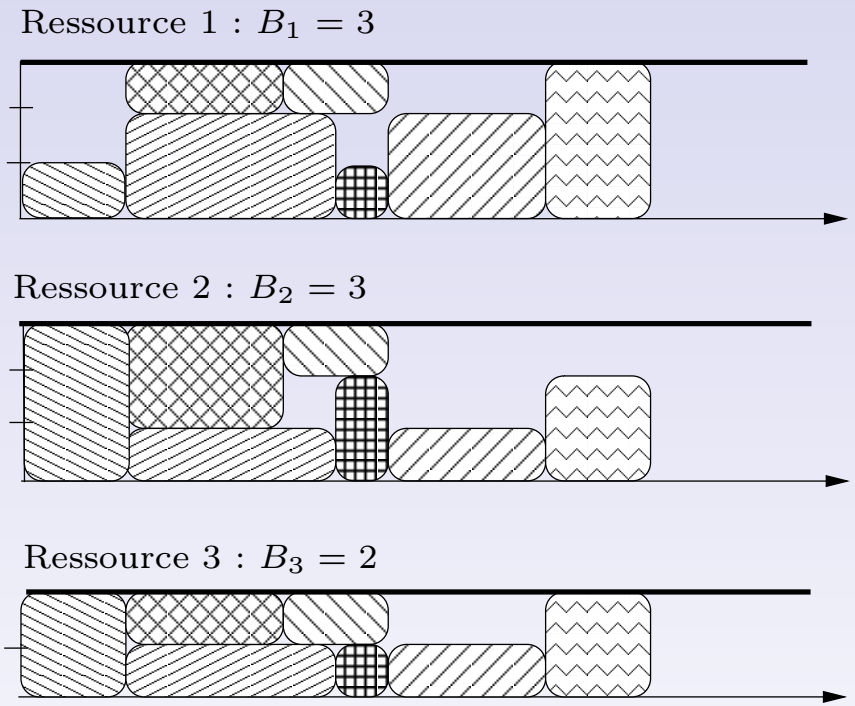
- CI : Nombre minimal de réductions pour obtenir un graphe de précédence d'une seule arête
- OS : Nombre de relations de précédence sur nombre maximum de relations de précédence (n(n-1)/2)
- CNC : Nombre d'arcs sur nombre de nœuds
- ...
- Quelle influence sur la complexité des instances ?

Le RCPSP - Resource Constrained Project Scheduling Problem

- X = 1, 2, . . ., n : l'ensemble des tâches. Les tâches 1 et n sont les tâches fictives de début et de fin du projet
- U = (i,j),. . .,(h,l) : l'ensemble des contraintes de précédence entre les tâches, $(i, j) \in U$ si j ne peut pas commencer avant la fin de i
- p_i : la durée de la tâche i
- K : le nombre de ressources à prendre en compte
- R = 1,2, . . .,K : les ressources utilisées par les tâches
- B_k : la disponibilité de la ressource k
- $b_{i,k}$: la quantité en ressource k requise par la tâche i
- Critère : durée Totale

Le RCPSP


2|1,3,2
3|1,2,1
2|3,2,2
0|0,0,0
2|1,1,1
0|0,0,0
1
5
6
0
3
8
7
2
4
3|2,1,1
1|1,2,1
4|2,1,1
Consommations de la tâche
durée de la tâche


Ressource 1 : $B_1 = 3$
Ressource 2 : $B_2 = 3$
Ressource 3 : $B_3 = 2$

Une incursion dans le monde du RCPSP

- En simplifiant notre problème avec des coûts de communication qui ne dépendent pas de l'affectation des tâches, on peut modéliser le problème sous forme d'un RCPSP multimode

Une incursion dans le monde du RCPSP

- En simplifiant notre problème avec des coûts de communication qui ne dépendent pas de l'affectation des tâches, on peut modéliser le problème sous forme d'un RCPSP multimode
  - en créant des tâches fictives pour les communications

Une incursion dans le monde du RCPSP

- En simplifiant notre problème avec des coûts de communication qui ne dépendent pas de l'affectation des tâches, on peut modéliser le problème sous forme d'un RCPSP multimode
  - en créant des tâches fictives pour les communications
  - en utilisant un modèle de RCPSP multimode avec timelags

Une incursion dans le monde du RCPSP

- En simplifiant notre problème avec des coûts de communication qui ne dépendent pas de l'affectation des tâches, on peut modéliser le problème sous forme d'un RCPSP multimode
  - en créant des tâches fictives pour les communications
  - en utilisant un modèle de RCPSP multimode avec timelags
- La seule limite est si les coûts de communication dépendent de l'affectation des tâches, qui nous fait sortir du RCPSP multimode avec timelags

Une incursion dans le monde du RCPSP

- En simplifiant notre problème avec des coûts de communication qui ne dépendent pas de l'affectation des tâches, on peut modéliser le problème sous forme d'un RCPSP multimode
  - en créant des tâches fictives pour les communications
  - en utilisant un modèle de RCPSP multimode avec timelags
- La seule limite est si les coûts de communication dépendent de l'affectation des tâches, qui nous fait sortir du RCPSP multimode avec timelags
- Travaux sur RCPSP multimode avec mode identity et timelags

Une incursion dans le monde du RCPSP

- En simplifiant notre problème avec des coûts de communication qui ne dépendent pas de l'affectation des tâches, on peut modéliser le problème sous forme d'un RCPSP multimode
  - en créant des tâches fictives pour les communications
  - en utilisant un modèle de RCPSP multimode avec timelags
- La seule limite est si les coûts de communication dépendent de l'affectation des tâches, qui nous fait sortir du RCPSP multimode avec timelags
- Travaux sur RCPSP multimode avec mode identity et timelags
- Mais utiliser le modèle du RCPSP nous oblige à des contorsions, et aboutit à la création d'instances très particulières

Générateurs d'instances pour le RCPSP

- Deux générateurs majeurs : RANGEN [Demeulemeester, Vanhoucke et Herroelen] et PROGEN [Kolisch et Sprecher].

Générateurs d'instances pour le RCPSP

- Deux générateurs majeurs : RANGEN [Demeulemeester, Vanhoucke et Herroelen] et PROGEN [Kolisch et Sprecher].
- Ne génèrent pas uniformément sur l'ensemble des graphes possibles

Générateurs d'instances pour le RCPSP

- Deux générateurs majeurs : RANGEN [Demeulemeester, Vanhoucke et Herroelen] et PROGEN [Kolisch et Sprecher].
- Ne génèrent pas uniformément sur l'ensemble des graphes possibles
- Génèrent des graphes pour des valeurs données de mesures topologiques.

RanGen

- Part d'un graphe complet

RanGen

- Part d'un graphe complet
- Tant que la valeur d'OS n'est pas atteinte : enlever une arête choisie au hasard

RanGen

- Part d'un graphe complet
- Tant que la valeur d'OS n'est pas atteinte : enlever une arête choisie au hasard
- A-t-on déjà obtenu ce graphe ?

RanGen

- Part d'un graphe complet
- Tant que la valeur d'OS n'est pas atteinte : enlever une arête choisie au hasard
- A-t-on déjà obtenu ce graphe ?
- Oui : On recommence du début, Non : on l'ajoute à notre ensemble

RanGen

- Part d'un graphe complet
- Tant que la valeur d'OS n'est pas atteinte : enlever une arête choisie au hasard
- A-t-on déjà obtenu ce graphe ?
- Oui : On recommence du début, Non : on l'ajoute à notre ensemble
- Si on a selectionné ce graphe, calculer son CI

RanGen

- Part d'un graphe complet
- Tant que la valeur d'OS n'est pas atteinte : enlever une arête choisie au hasard
- A-t-on déjà obtenu ce graphe ?
- Oui : On recommence du début, Non : on l'ajoute à notre ensemble
- Si on a selectionné ce graphe, calculer son CI
- Au bout d'un certain temps : selectionner dans l'ensemble résultats des graphes qui ont une valeur de CI choisie.

RanGen

- La seule garantie fournie est une borne temporelle

RanGen

- La seule garantie fournie est une borne temporelle
- Les demandes en ressources sont paramétrées, et générée aléatoirement, indépendamment de la topologie.

RanGen

- La seule garantie fournie est une borne temporelle
- Les demandes en ressources sont paramétrées, et générée aléatoirement, indépendamment de la topologie.
- La génération est basée sur des valeurs d'OS et de CI, car dans le cadre du RCPSP, il existe une corrélation entre les valeurs de ces métriques et la difficulté des instances.

Quels benchmarks utiliser ? [Kwok & Ahmad]

- Exemples de la littérature
- Instances aléatoires avec valeur optimale connue
- Instances aléatoires à partir de valeur optimale
- Instances réelles
- Instances aléatoires

Générer des instances et leurs optimaux

1. Générer un graphe et ses volumes

Générer des instances et leurs optimaux

1. Générer un graphe et ses volumes
2. Appliquer une méthode exacte (PSE, ...)

Générer des instances et leurs optimaux

1. Générer un graphe et ses volumes
2. Appliquer une méthode exacte (PSE, ...)

- Petites instances

Générer des instances et leurs optimaux

- Génération de graphes ayant des valeurs de CCR données

Générer des instances et leurs optimaux

- Génération de graphes ayant des valeurs de CCR données
- Tirage uniforme des durées opératoires

Générer des instances et leurs optimaux

- Génération de graphes ayant des valeurs de CCR données
- Tirage uniforme des durées opératoires
- Pour chaque noeuds, nombre de successeurs tirés uniformément avec une moyenne de n/10

Générer des instances et leurs optimaux

- Génération de graphes ayant des valeurs de CCR données
- Tirage uniforme des durées opératoires
- Pour chaque noeuds, nombre de successeurs tirés uniformément avec une moyenne de n/10
- Coûts de communication tirés uniformément avec moyenne correspondante aux durées pour obtenir la valeur de CCR donnée en moyenne

Générer des instances à partir de valeurs d'optimaux

1. Fixer des valeurs de critère

Générer des instances à partir de valeurs d'optimaux

1. Fixer des valeurs de critère
2. Créer une solution qui a pour valeur celle fixée

Générer des instances à partir de valeurs d'optimaux

1. Fixer des valeurs de critère
2. Créer une solution qui a pour valeur celle fixée

- La méthode est propre à chaque critère

Générer des instances à partir de valeurs d'optimaux

1. Fixer des valeurs de critère
2. Créer une solution qui a pour valeur celle fixée

- La méthode est propre à chaque critère
- Peu d'aléa sur l'ordonnancement optimal

Générer des instances à partir de valeurs d'optimaux

1. Fixer des valeurs de critère
2. Créer une solution qui a pour valeur celle fixée

- La méthode est propre à chaque critère
- Peu d'aléa sur l'ordonnancement optimal
- Qualité de l'instance ? La borne inférieure la plus mauvaise vaut l'optimal...

Générer des instances à partir de cas réels

- Cas dit du "rejeu de données"

Générer des instances à partir de cas réels

- Cas dit du "rejeu de données"
- Aucune garantie de se situer dans un cas général

Générer des instances à partir de cas réels

- Cas dit du "rejeu de données"
- Aucune garantie de se situer dans un cas général
- Architectures d'applications parallèles souvent les mêmes

Générer des instances quelconques

- Permet de générer de grandes instances

Générer des instances quelconques

- Permet de générer de grandes instances
- Les tests sur ces instances ne permettront de classer les algorithmes que relativement les uns aux autres

Survey

- Aucune garantie sur l'uniformité

Survey

- Aucune garantie sur l'uniformité
- On utilise le CCR faute de mieux

Survey

- Aucune garantie sur l'uniformité
- On utilise le CCR faute de mieux
- Le CCR se corrèle parfois au taux de dégradation par rapport à l'optimal

Uniformément sur l'ensemble des graphes [Melancon, et al.]

- Soit $N \geq 2$ un entier et soit $V = \{1, \ldots, N\}$ un ensemble fini de nœuds.

Uniformément sur l'ensemble des graphes [Melancon, et al.]

- Soit $N \geq 2$ un entier et soit $V = \{1, \ldots, N\}$ un ensemble fini de nœuds.
- Soit A l'ensemble de tous les graphes orientés acycliques sur V

Uniformément sur l'ensemble des graphes [Melancon, et al.]

- Soit $N \geq 2$ un entier et soit $V = \{1, \ldots, N\}$ un ensemble fini de nœuds.
- Soit A l'ensemble de tous les graphes orientés acycliques sur V
- Soit la chaîne de Markov M définie sur l'ensemble A, ie : chaque état de M est un DAG. Les transitions sont définies comme suit, étant donné X_t l'état de la chaîne à l'instant t. À chaque étape un couple d'entiers ordonnés (i, j) est tiré au hasard uniformément dans l'ensemble $V \times V$.

Uniformément sur l'ensemble des graphes [Melancon, et al.]

- Soit $N \geq 2$ un entier et soit $V = \{1, \ldots, N\}$ un ensemble fini de nœuds.
- Soit A l'ensemble de tous les graphes orientés acycliques sur V
- Soit la chaîne de Markov M définie sur l'ensemble A, ie : chaque état de M est un DAG. Les transitions sont définies comme suit, étant donné X_t l'état de la chaîne à l'instant t. À chaque étape un couple d'entiers ordonnés (i, j) est tiré au hasard uniformément dans l'ensemble $V \times V$.
  - Soit (i, j) est un arc du graphe de l'état X_t, alors $X_{t+1} = X_t - (i, j)$

Uniformément sur l'ensemble des graphes [Melancon, et al.]

- Soit $N \geq 2$ un entier et soit $V = \{1, \ldots, N\}$ un ensemble fini de nœuds.
- Soit A l'ensemble de tous les graphes orientés acycliques sur V
- Soit la chaîne de Markov M définie sur l'ensemble A, ie : chaque état de M est un DAG. Les transitions sont définies comme suit, étant donné X_t l'état de la chaîne à l'instant t. À chaque étape un couple d'entiers ordonnés (i, j) est tiré au hasard uniformément dans l'ensemble $V \times V$.
  - Soit (i, j) est un arc du graphe de l'état X_t, alors $X_{t+1} = X_t - (i, j)$
  - Sinon
    - Si ajouter l'arc (i, j) ne créé pas de cycle, alors $X_{t+1} = X_t \cup (i, j)$

Uniformément sur l'ensemble des graphes [Melancon, et al.]

- Soit $N \geq 2$ un entier et soit $V = \{1, \ldots, N\}$ un ensemble fini de nœuds.
- Soit A l'ensemble de tous les graphes orientés acycliques sur V
- Soit la chaîne de Markov M définie sur l'ensemble A, ie : chaque état de M est un DAG. Les transitions sont définies comme suit, étant donné X_t l'état de la chaîne à l'instant t. À chaque étape un couple d'entiers ordonnés (i, j) est tiré au hasard uniformément dans l'ensemble $V \times V$.
  - Soit (i, j) est un arc du graphe de l'état X_t, alors $X_{t+1} = X_t - (i, j)$
  - Sinon
    - Si ajouter l'arc (i, j) ne créé pas de cycle, alors $X_{t+1} = X_t \cup (i, j)$
    - Sinon $X_{t+1} = X_t$

Uniformité

- On remarque aisément que la matrice de transition est symétrique par construction

Uniformité

- On remarque aisément que la matrice de transition est symétrique par construction
- De même, par construction M est irréductible : on peut toujours passer d'un état à un autre (il suffit d'enlever tous les arcs et de tous les remettre).

Uniformité

- On remarque aisément que la matrice de transition est symétrique par construction
- De même, par construction M est irréductible : on peut toujours passer d'un état à un autre (il suffit d'enlever tous les arcs et de tous les remettre).
- La chaîne de Markov est aussi apériodique

Uniformité

- On remarque aisément que la matrice de transition est symétrique par construction
- De même, par construction M est irréductible : on peut toujours passer d'un état à un autre (il suffit d'enlever tous les arcs et de tous les remettre).
- La chaîne de Markov est aussi apériodique
- des éléments précédents, on déduit que la chaine de Markov est ergodique, ce qui implique la convergence vers une unique distribution stationnaire, et on peut vérifier que la distribution uniforme est stationnaire.

Uniformité

- $$\lim_{n \to +\infty} M^n d = U$$
- où $U = (\frac{1}{\|A\|}, \frac{1}{\|A\|}, \ldots, \frac{1}{\|A\|})$

Complexité

- La complexité de l'algorithme est dominée par le nombre d'itérations nécessaires à la convergence. Le problème d'estimation d'une bonne borne supérieure sur ce point est un problème ouvert.

Complexité

- La complexité de l'algorithme est dominée par le nombre d'itérations nécessaires à la convergence. Le problème d'estimation d'une bonne borne supérieure sur ce point est un problème ouvert.
- On peut cependant remarquer que la distance maximale entre deux graphes est de $N(N-1)$ (un graphe complet sur N sommets a $\frac{N(N-1)}{2}$ arêtes)

Complexité

- La complexité de l'algorithme est dominée par le nombre d'itérations nécessaires à la convergence. Le problème d'estimation d'une bonne borne supérieure sur ce point est un problème ouvert.
- On peut cependant remarquer que la distance maximale entre deux graphes est de $N(N-1)$ (un graphe complet sur N sommets a $\frac{N(N-1)}{2}$ arêtes)
- Cela signifie qu'en N^2 itérations, on a une probabilité non nulle d'atteindre n'importe quel état, mais n'implique pas l'uniformité pour autant (!).

Complexité

- La vérification de non création de cycle peut se faire en temps polynomial [Itai & Rodeh]

Complexité

- La vérification de non création de cycle peut se faire en temps polynomial [Itai & Rodeh]
- Complexité au pire des cas O(n+e) où e est le nombre d'arcs

Complexité

- La vérification de non création de cycle peut se faire en temps polynomial [Itai & Rodeh]
- Complexité au pire des cas O(n+e) où e est le nombre d'arcs
- complexité moyenne en O(N $\log_2$ N).

Complexité

- La vérification de non création de cycle peut se faire en temps polynomial [Itai & Rodeh]
- Complexité au pire des cas O(n+e) où e est le nombre d'arcs
- complexité moyenne en O(N $\log_2$ N).
- À noter que $e \leq \frac{N(N-1)}{2}$. D'où une complexité en $O(N^4)$

Extension de la méthode

- Cas des non isomorphes difficiles à discriminer, difficile de borner le temps d'exécution

Extension de la méthode

- Cas des non isomorphes difficiles à discriminer, difficile de borner le temps d'exécution
- Graph répondant à un critère donné...

Extension de la méthode

- Cas des non isomorphes difficiles à discriminer, difficile de borner le temps d'exécution
- Graph répondant à un critère donné...
- Méthode simple à implémenter et très rapide

Extension de la méthode

- Cas des non isomorphes difficiles à discriminer, difficile de borner le temps d'exécution
- Graph répondant à un critère donné...
- Méthode simple à implémenter et très rapide
- Adaptable simplement pour la génération d'arbres

Résultats expérimentaux

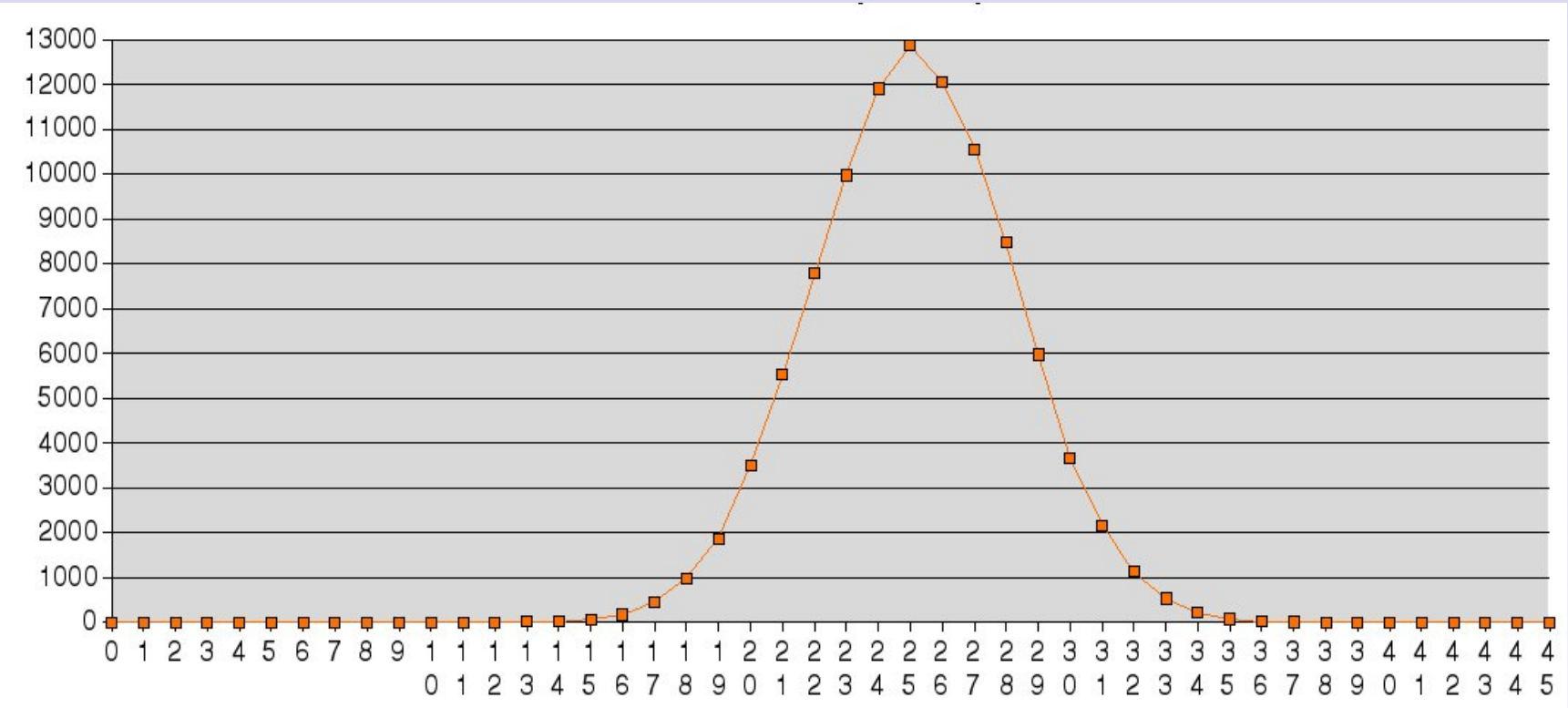
- Implémentation en C++, exécution sur P4M 2GHz

Résultats expérimentaux

- Implémentation en C++, exécution sur P4M 2GHz

N	Runs	T (s)	OS min	OS moy	OS max	C. Conn. moy.
10	10^5	59	0.38	0.88	1	1.00277
20	10^5	525	0.71	0.93	1	1
100	10^3	2208	0.977	0.987	0.995	1

Résultats expérimentaux



En conclusion

- Des méthodes très différentes, spécifiques à chaque problème

En conclusion

- Des méthodes très différentes, spécifiques à chaque problème
- Point commun : pas transcendantes

En conclusion

- Des méthodes très différentes, spécifiques à chaque problème
- Point commun : pas transcendantes
- Génération topologiques et volumiques toujours distinctes

En conclusion

- Des méthodes très différentes, spécifiques à chaque problème
- Point commun : pas transcendantes
- Génération topologiques et volumiques toujours distinctes
- Problème avec les isomorphes

En conclusion

- Des méthodes très différentes, spécifiques à chaque problème
- Point commun : pas transcendantes
- Génération topologiques et volumiques toujours distinctes
- Problème avec les isomorphes
- Il n'existe pas de métrique d'instances que l'on puisse corréler à coup sûr à la difficulté des instances