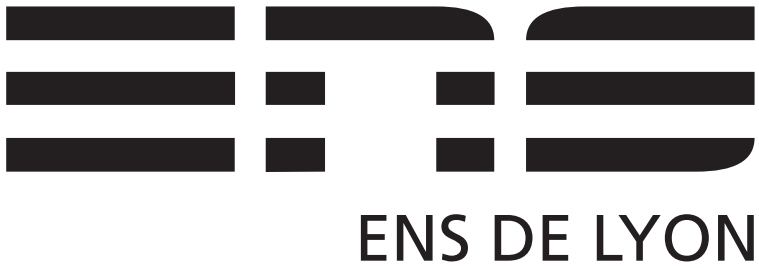
Compression d'ordonnancement

Jérémie Dumas

École Normale Supérieure de Lyon

Janvier 2012



Contexte et motivations

- Ordonnancement sur un DAG (graphe de tâches).
- Minimiser le makespan, les communications ?
- Problème NP-dur comme on les aime.
- Algos `PLW`, `TCSD` et `CPFD` : optimisent le makespan, pas le nb de procs.
- Ici, algos `SC` et `SDS` : réduction de 70% à 90% du nb de procs.

Les algos présentés

- `SC` : compression du nb de processeurs, $\mathcal{O}(|\mathcal{N}|^3)$.
- `SDS` : ordonnancement de DAG plus général, $\mathcal{O}(|\mathcal{N}|^3)$.

Définitions

Généralités

- DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, dépendance $(n_p, n_t) \in \mathcal{E}$.
- w_t, temps d'exécution de n_t.
- $c_{p,t}$, temps de communication si n_p et n_t sur deux procs différent.
- P peut calculer et communiquer en même temps.
- Processeurs et communications homogènes, réseau complet.
- Ordonnancement non-préemptif.
- Algo. existants : avec ou sans réplication de tâches.

Définitions

Ordonnancement

- Notions de chemin critique.
- Bottom-level $b_t = w_t + \max(c_{t,c} + b_c)$.
- Start time et finish time $st(n_t, P_l)$ et $ft(n_t, P_l)$.
- Fils local ou hors-processeur.
- Latest finish time $lft(n_t)$.
- Dans SC, notion de copie fixe d'une tâche.
- Data arrival time $dat(n_t, n_p)$.
- Earliest start time $est(n_t, P_l)$.

Schedule Compaction

Principe général

Trois phases

- Fixer la copie qui enverra les données hors-processeurs.
- Éliminer un maximum de copies redondantes.
- Fusionner itérativement des ordonnancements partiels.

Les règles d'or

VC1 Réception des données à temps pour $n_t \in P_m$.

VC2 Envoi aux fils hors-processeur.

VC3 Tâche n_t exécutée avant ses fils locaux.

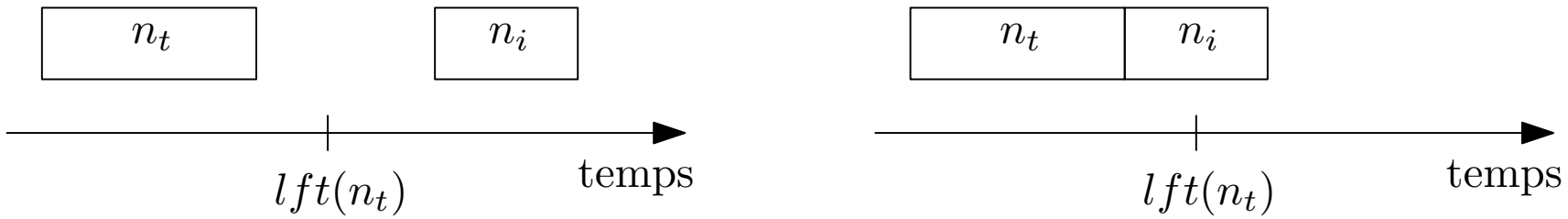
VC4 Durée totale $\leqslant \sigma$.

Schedule Compaction

Première phase

Parcours des tâches par bottom-level croissant.

- Calcul de $lft(n_t)$ grâce à $\mathcal{Q}(n_t)$ l'ensemble des fils hors-processeurs.
- Fixer la copie de n_t (voir figure).
- Décaler les tâches le plus possible vers la fin (créer des fils hors-proc).



(a) Minimiser l'inactivité

(b) Finir au plus tôt

Figure 1 : Fixer une copie de n_t : deux cas de figure.

Schedule Compaction

Deuxième phase

- Calcul itératif de $est(n_t, P_l)$ pour chaque copie de chaque tâche n_t.
- Regarder les contraintes de données entre les tâches. 3 cas possibles :
  1. n_t reçoit des données de n_p hors-processeur.
  2. n_t reçoit des données de n_p local (n_p essentiel).
  3. n_t peut recevoir des données locales ou hors-processeur. La copie de n_p sur P_l est peut-être superflue : la supprimer plus tard ?
- Re-décaler les tâches le plus possible vers le début (VC1 émetteur).

Schedule Compaction

Troisième phase

- Fusions itératives de deux processeurs.
- $et_l, et_t, et_{l,k}$: somme de temps d'exécution de tâches.
- Mesure de similarité :

$$sim_{l,k} = \begin{cases} \frac{et_{l,k}}{\min(et_l, et_k)} & \text{si } et_l + et_k - et_{l,k} \leqslant \max(\sigma_l, \sigma_k) \\ -1 & \text{sinon} \end{cases}$$

- Placer les tâches de P_l ou de P_k sur P_m (en partant de la fin).
- Éliminer les redondances, maj de $est, lft, \mathcal{Q}(n_t)$.
- Si $\mathcal{Q}(n_t)$ devient vide, libérer n_t.

Sub-DAGs Scheduling

Idée générale

- Placer chaque n_i sur un proc P_i, et laisser à SC le soin de recoller.
- Optimiser P_i pour faire terminer n_i le plus tôt possible.
- Dupliquer les parents de n_i sur P_i récursivement.
- Tâche computation-bounded, communication-bounded.
- Notion de parent critique et de bottleneck.

Sub-DAGs Scheduling

Schéma de duplication

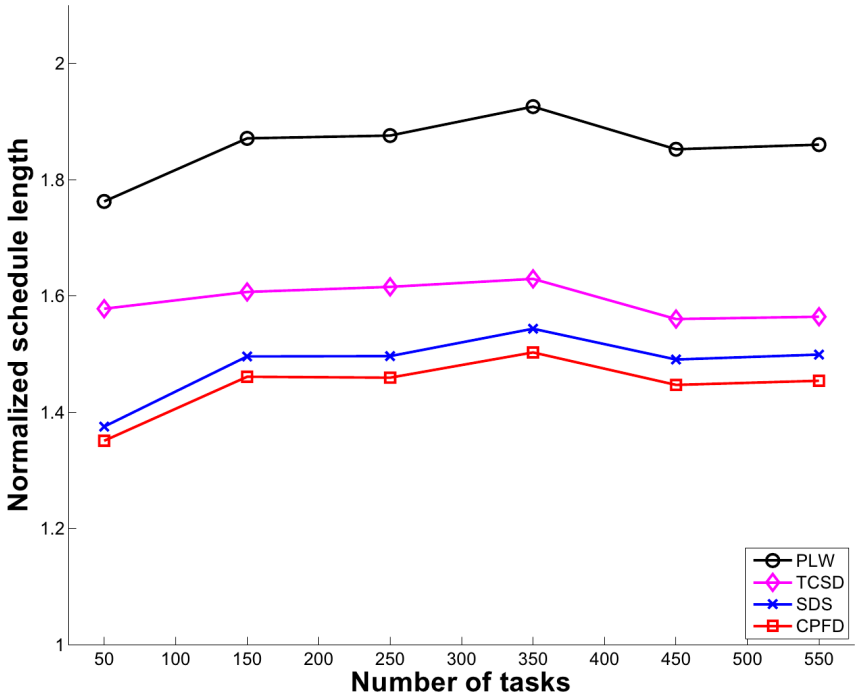
- Dupliquer le parent critique n_p du bottleneck n_b de P_i.
- L'insérer dès que possible (et décaler la suite).
- Produit différents ordonnancement P_t possible sur le proc P_i.
- On garde le meilleur des P_t obtenu (dont la longueur n'est pas forcément monotone).
- Répéter $|\mathcal{N}| \times \kappa$ fois (complexité bornée).

Résultats expérimentaux

Cadre expérimental

- Graphes aléatoires, graphes de la vie réelle.
- Paramètres : # parents, # tâches, # matrice, CCR.
- SC plutôt bon en pratique (graphes aléatoires et réels).
- Plus σ est élevé, plus on pourra compresser # proc.

Résultats expérimentaux



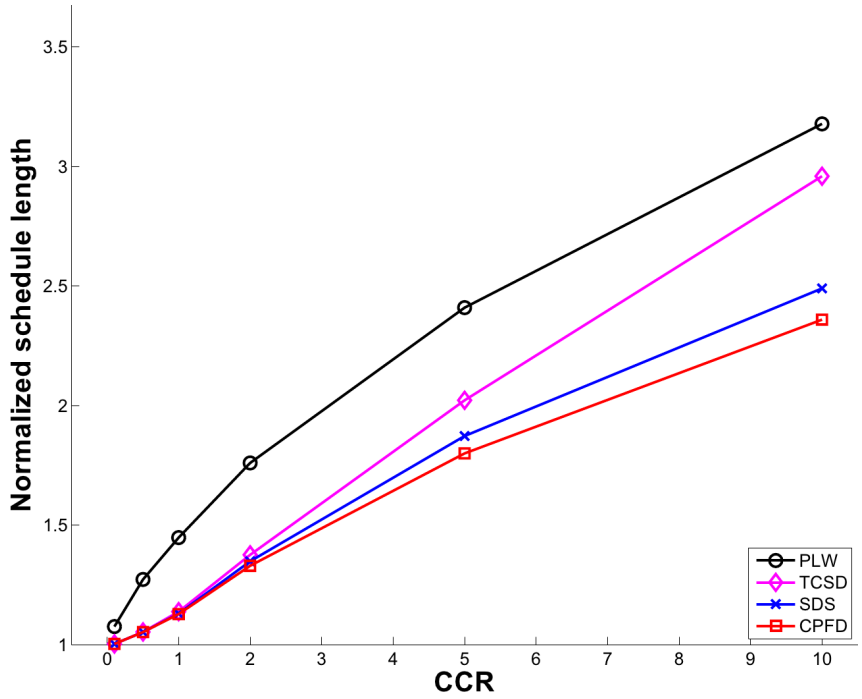
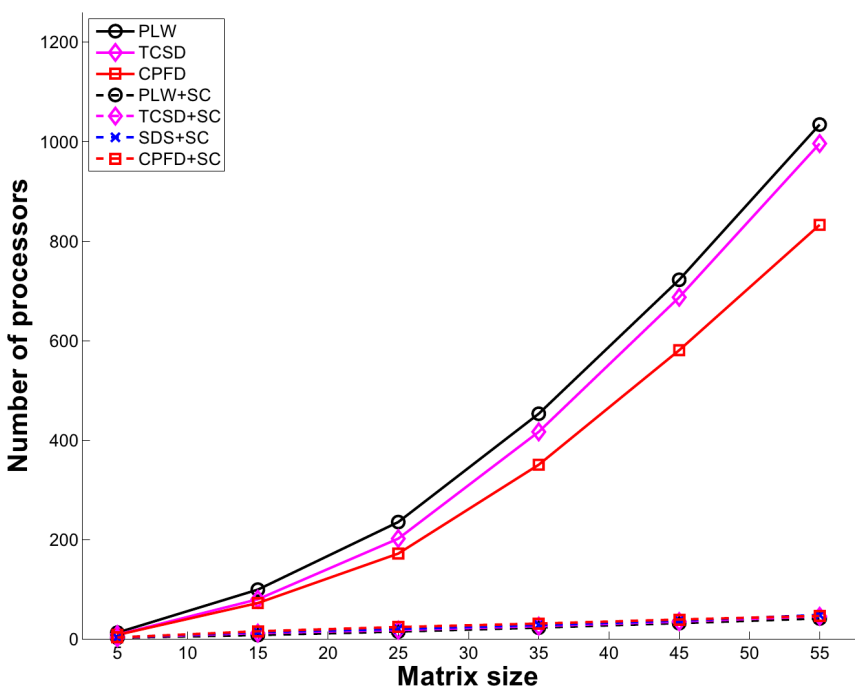


Figure 2 : Longueur normalisée des ordonnancements sur des DAGs aléatoires.

Résultats expérimentaux



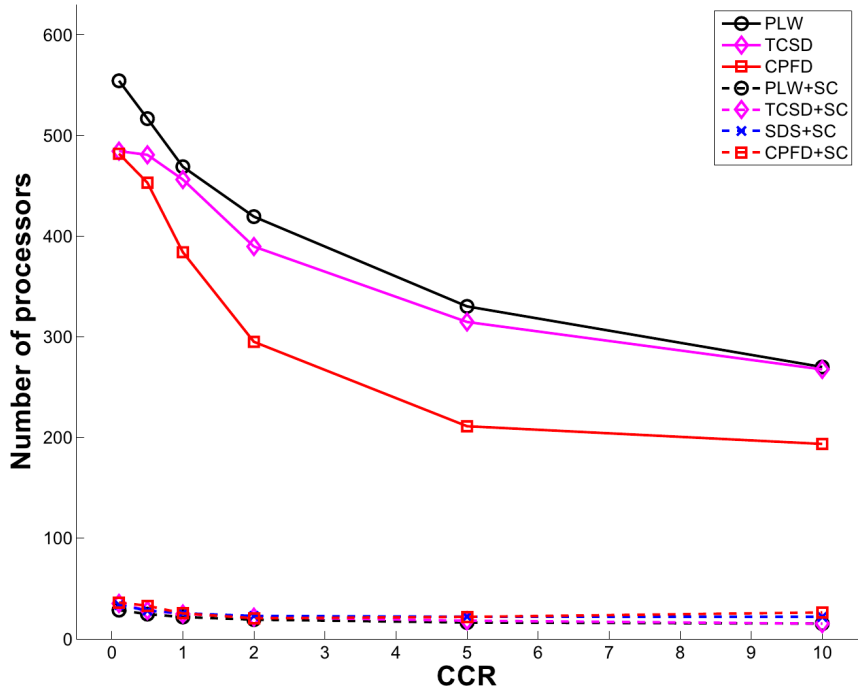


Figure 3 : Nombre de processeurs requis pour une élimination gaussienne.

Conclusion

- SC et SDS efficaces et en $\mathcal{O}(|\mathcal{N}|^3)$ (alors que CPFD en $\mathcal{O}(|\mathcal{N}|^4)$).
- SC s'applique à n'importe quel ordo valide.
- Correction et complexité facile à prouver.
- Efficacité heuristique (pas de borne théorique).
- Multiples hypothèses, en particulier homogénéité des systèmes.

Merci de votre attention.