Comparaison de différentes approches de résolution par méta-heuristique pour le RCPSP multi-site

Arnaud Laurent

27 fevrier 2015

Sommaire

RCPSP

Référence

«Multiproject scheduling with limited resources : A zero-one programming approach» par Pritsker, Watters and Wolfe, 1969
«Resource-constrained project scheduling problem : Notation, classification, models and methods» par Brucker, Drexl, Möhring, Neumann et Pesch, 1999

- Ordonnancer un ensemble de tâches qui ont une durée déterminée
- Les tâches nécessitent des types de ressources en une certaine quantité
- Contraintes de précédence

Notations

N Nombre de tâches

P_j Ensemble de tâches qui doivent précéder $j = 1, N$

p_j Durée de la tâche $j = 1, N$

K Nombre de types de ressources

R_k Nombre de ressources de type $k = 1, K$

$r_{j,k}$ Nombre de ressources de type $k = 1, K$ nécessaires pour la tâche $j = 1, N$

T Nombre de périodes maximum

Multi-Sites RCPSP

Référence

« Une extension du RCPSP pour la mutualisation de ressources entre plusieurs sites : le Multi Location RCPSP », par Arnaud LAURENT, Laurent DEROUSSI, Nathalie GRANGEON, Sylvie NORRE, 2014

- Ajout de la notion de site
- Ajout de distance entre les sites entrainant des temps de transfert
- Deux cas où un temps de transfert s'applique :
  - Une contrainte de précédence entre deux tâches
  - Un déplacement d'une ressource
- Différentiation entre les ressources fixes et les ressources mobiles

Multi-Sites RCPSP

Référence

« A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags », par Roland Heilmann, 2003 « Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times », par Marek Mika and Grzegorz Waligora and Jan Weglarz, 2008

- MRCPSP/max qui est un multi-mode RCPSP avec des time lags minimaux et maximaux dépendant des modes.
- MRCPSP-SDST qui est un multi-mode RCPSP avec des temps de montage dépendant de la séquence.

Nouvelles notations

$M_{k,r}$ $= 1$ $r = 1, R_k$ de type $k = 1, K$, si la ressource est mobile, 0 si elle est fixe

S Nombre de sites

$\delta_{s,s'}$ Distance en temps entre le site $s = 1, S$ et le site $s' = 1, S$

$loc_{k,r}$ Site d'appartenance de la ressource $r = 1, R_k$ de type $k = 1, K$

Exemple

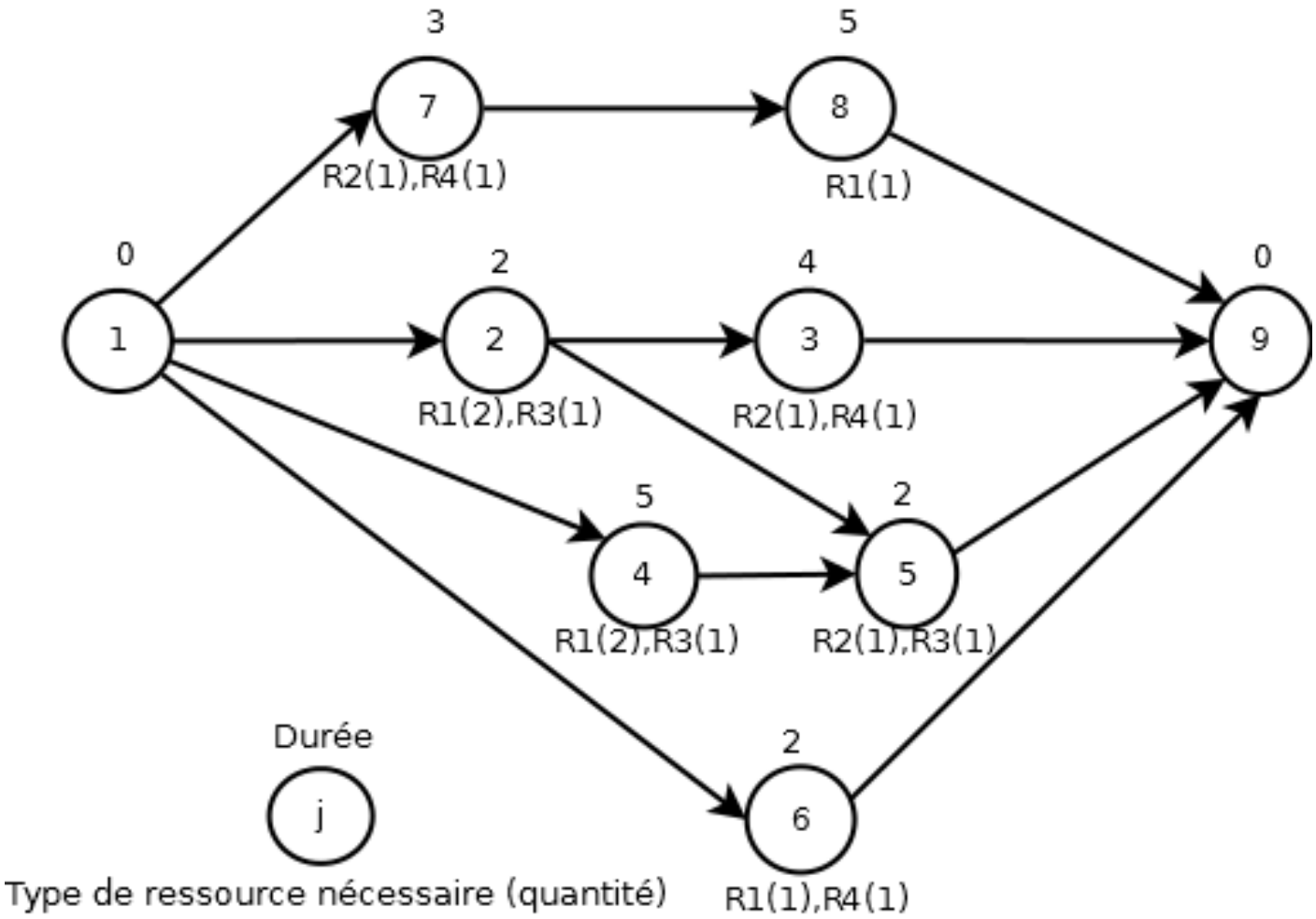
- 7 tâches
- 4 types de ressources

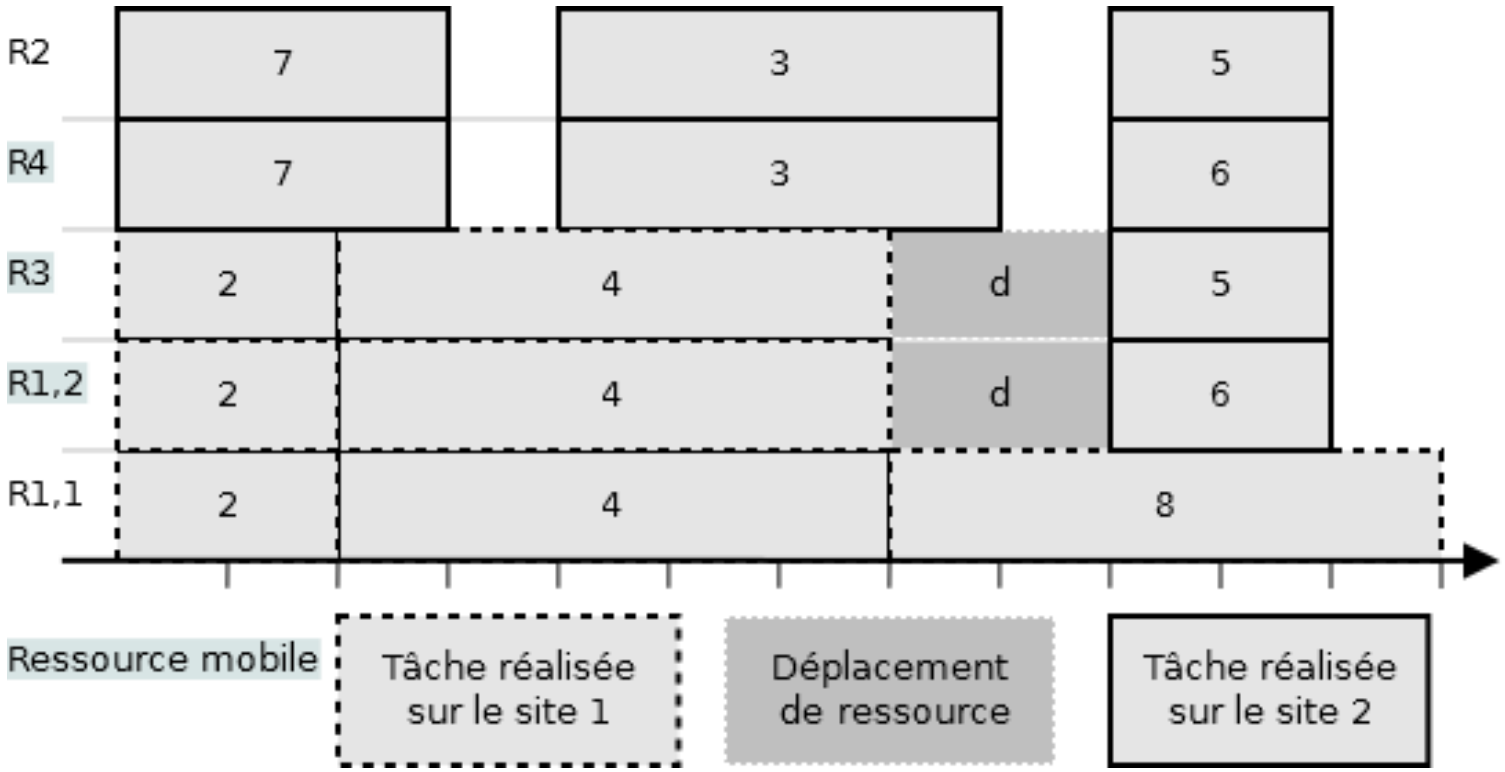
	Type	Mobilité	Site
R1,1	1	Fixe	1
R1,2	1	Fixe	1
R2	2	Fixe	2
R3	3	Mobile	X
R4	4	Mobile	X

Table : Liste des ressources disponibles

- 2 sites
  - Les sites sont distant d'une durée de transport de 2 périodes

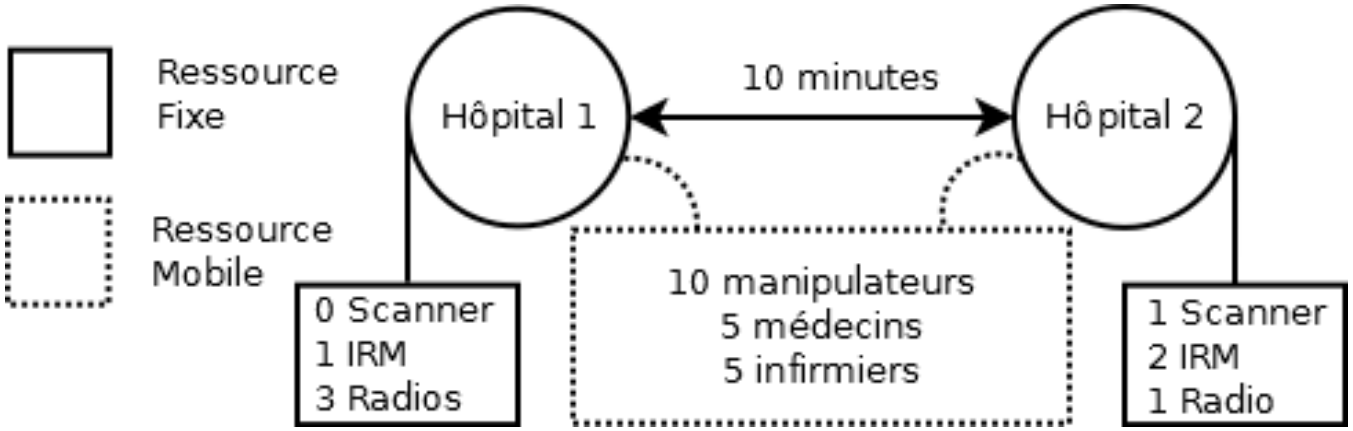
Exemple


3
7
R2(1),R4(1)
5
8
R1(1)
0
1
2
2
R1(2),R3(1)
4
3
R2(1),R4(1)
0
9
5
4
R1(2),R3(1)
2
5
R2(1),R3(1)
Durée
j
Type de ressource nécessaire (quantité)
2
6
R1(1),R4(1)


R2
R4
R3
R1,2
R1,1
7
3
5
7
3
6
2
4
d
5
2
4
d
6
2
4
8
Ressource mobile
Tâche réalisée sur le site 1
Déplacement de ressource
Tâche réalisée sur le site 2

Un exemple d'application : la CHT

La Communauté Hospitalière de Territoire (CHT)



- Une liste de patients

But

Ordonnancer toutes les opérations et affecter les ressources

Sommaire

Éléments des codages et systèmes de voisinage

- Une liste ordonnée σ de tâches
- Une liste l d'affectations de chaque tâche à un site
- Une matrice d'affectation a des ressources aux tâches

- Le codage σ
- Le codage σ, l
- Le codage σ, l, a

Schedule Generation Scheme

SGS

A chaque codage correspond un SGS (Schedule Generation Scheme). Le principe de base est d'ordonnancer au plus tôt chaque tâche dans l'ordre σ

- Détermine la date de début des tâches pour le codage σ, I, a
- Détermine l'affectation des ressources et la date de début des tâches pour le codage σ, I
- Détermine l'affectation des ressources, la localisation des tâches et la date de début des tâches pour le codage σ

Les trois différents systèmes de voisinage

Les systèmes de voisinage seront donc :

- Un mouvement V1 pour modifier σ :
  Insertion d'une tâche dans σ
- Un mouvement V2 pour modifier I :
  Modification d'une affectation d'un site pour une tâche
- Un mouvement V3 pour modifier a :
  Modification d'une affectation d'une ressource pour une tâche

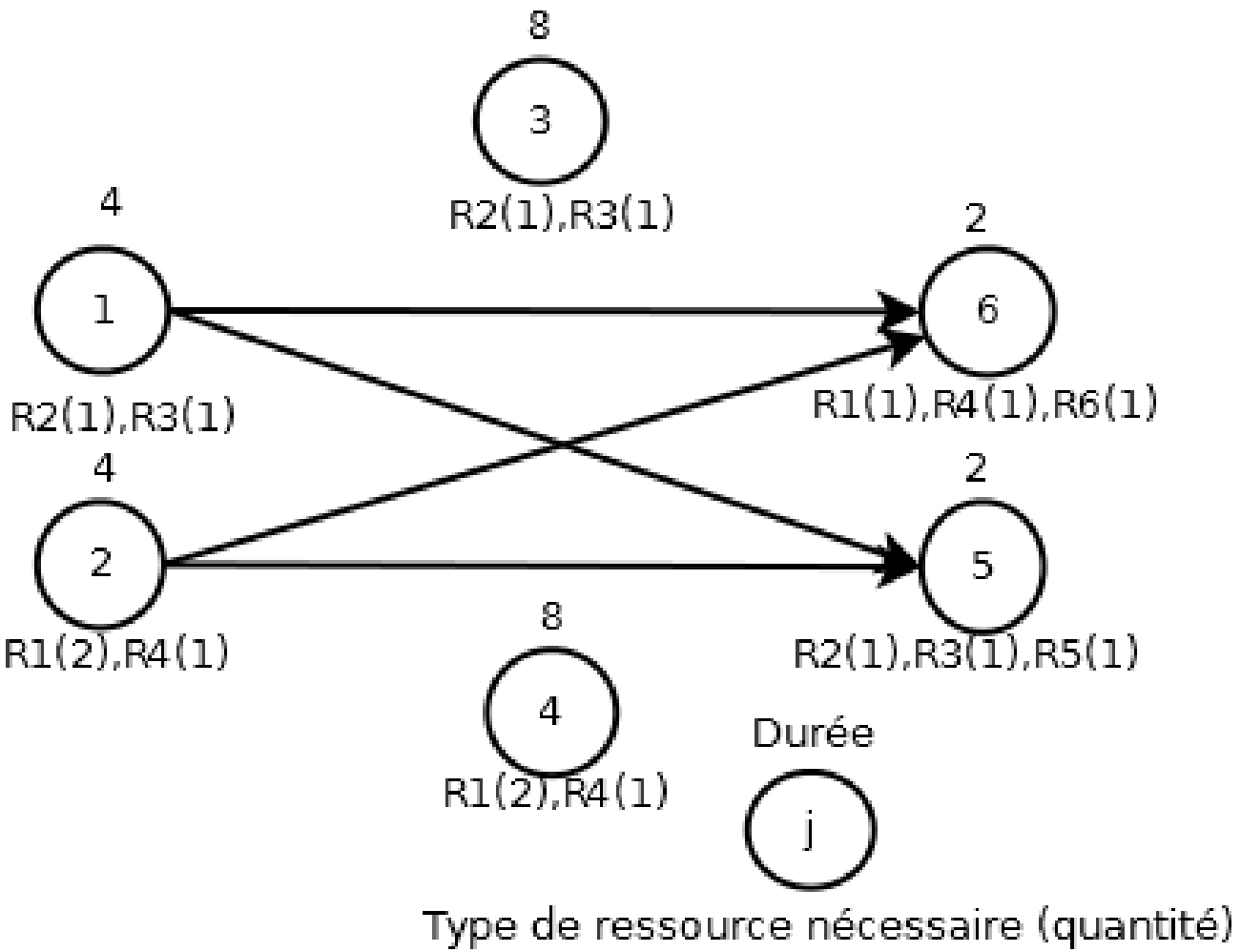
Exemple

- 6 tâches

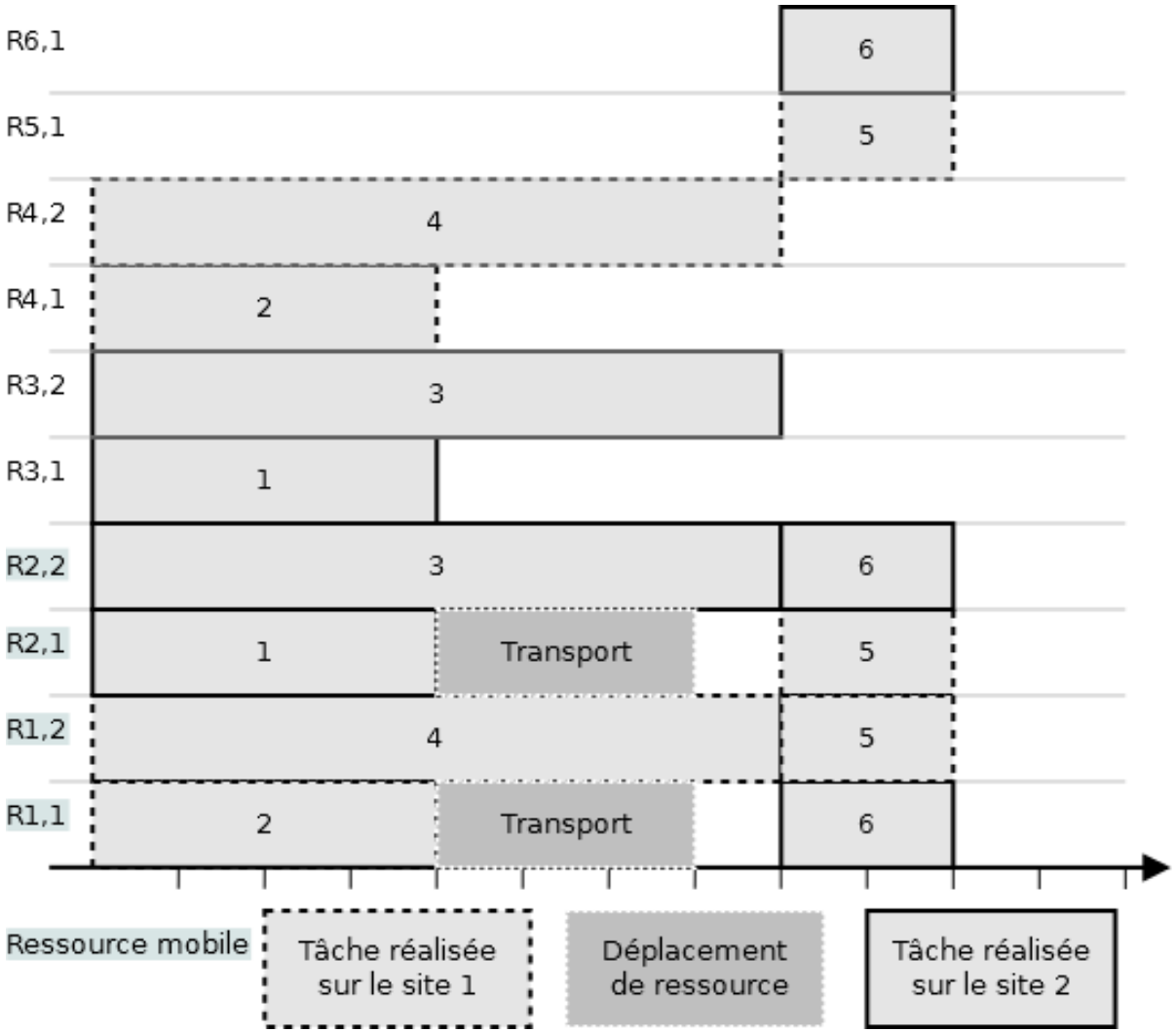
	Type	Mobilité	Site
R1,1 et R1,2	1	Mobile	X
R2,1 et R2,2	2	Mobile	X
R3,1 et R3,2	3	Fixe	2
R4,1 et R4,2	4	Fixe	1
R5	5	Fixe	1
R6	6	Fixe	2

- 2 sites
  - Les sites sont éloigné d'une durée de transport de 3 périodes

Graphe de précédence



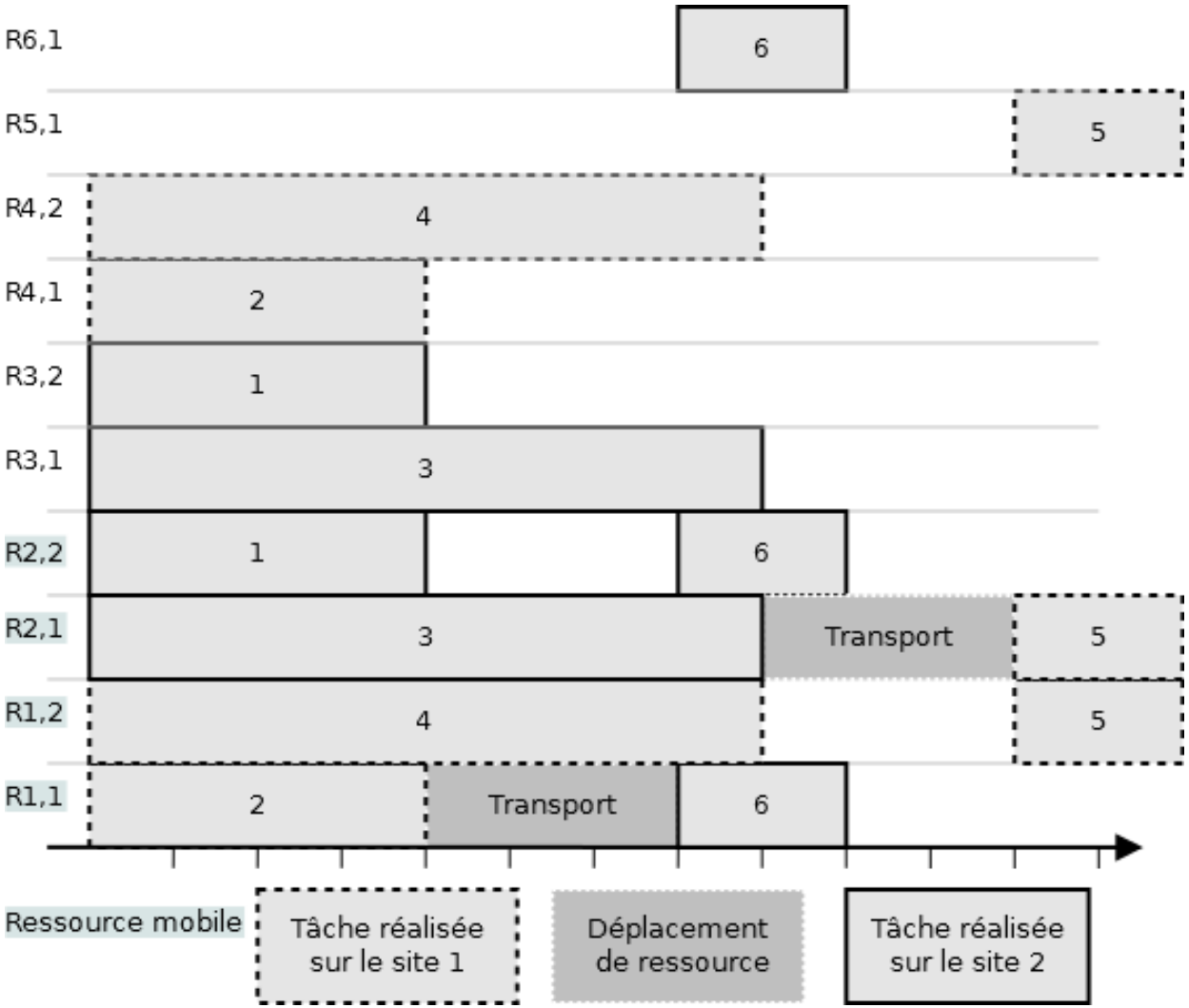
Une solution optimale



Un codage pour cette solution

	Position dans σ	I	a
Tâche 1	1	Site 2	R2,1 R3,1
Tâche 2	2	Site 1	R1,1 R4,1
Tâche 3	3	Site 2	R2,2 R3,2
Tâche 4	4	Site 1	R1,2 R4,2
Tâche 5	6	Site 1	R1,2 R2,1 R5,1
Tâche 6	5	Site 2	R1,1 R2,2 R6,1

Une solution optimale pour le codage σ, I



Un codage pour cette solution

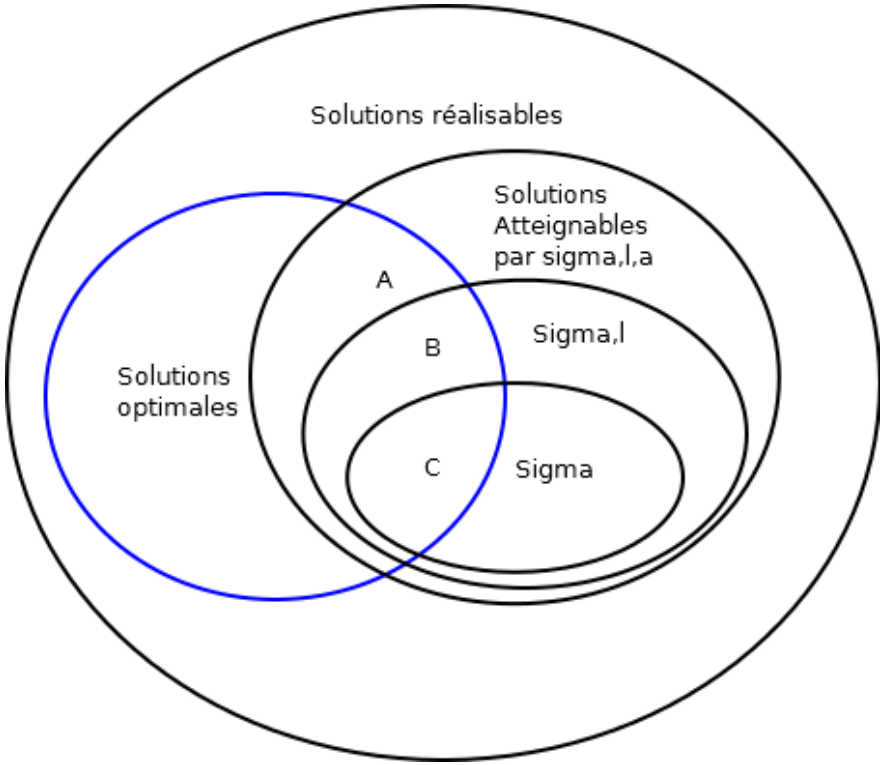
	Position dans σ	I
Tâche 1	1	Site 2
Tâche 2	2	Site 1
Tâche 3	3	Site 2
Tâche 4	4	Site 1
Tâche 5	6	Site 1
Tâche 6	5	Site 2

$\Rightarrow$ Le vecteur σ et I de cette solution sont les mêmes que pour la solution optimale de l'instance vu précédemment.

Accessibilité

Théorème

Les codages σ et σ, l ne respectent **pas toujours** l'accessibilité à une solution optimale. $B, C = \oslash$



Sommaire

La recherche locale itérée

Algorithme 6 : Algorithme de principe de la recherche locale itérée

Entrées : X_0 : Solution initiale aléatoire et réalisable;
Variables : X^* : Meilleure solution trouvée;
X' : Solution voisine;
Initialisation : X^* := Recherche locale sur X_0;

```
1 Debut
2     Tant que Test d'arrêt est faux faire
3         X' := Perturbation de X* ;
          X' := Recherche locale sur X';
          X* := Critère d'acceptation de X' par rapport à X* en prenant en
          compte l'historique ;
4     Fintq
5     Retourner X*
6 Fin
```

Les paramètres

- Le test d'arrêt : Le temps
- Le critère d'acceptation : Type recuit simulé
- Critère d'arrêt de la recherche locale : Nombre maximum d'itération
- Le voisinage utilisé sera l'application équiprobable d'un des différents mouvements possibles
- Perturbation : Application de 4 fois le système de voisinage

Sommaire

Les instances

- Instances de la littérature de la PSPLIB
- 480 instances par nombre de tâches différent (30, 60, 90, 120)
- Un grand nombre des instances où la solution optimale est connue
- La solution optimale des instances du RCPSP est une borne inférieure pour notre problème

On ajoute à ces instances les caractéristiques suivantes :

- 1, 2, 3 sites
- Une ressource a une probabilité de 50% d'être fixe.
- La distance entre deux sites varie de 1 à 10 périodes
- Affectation des ressources fixes aux sites aléatoirement

Sommaire

Protocole expérimental

- On utilise le meilleur paramétrage pour chaque codage
- On résout les instances avec les trois codages en limitant la résolution à 30 minutes
- On réplique la résolution 10 fois par instances

Résultat sur une instances (30 tâches)

Résultat obtenus sur 30 minutes d'exécution

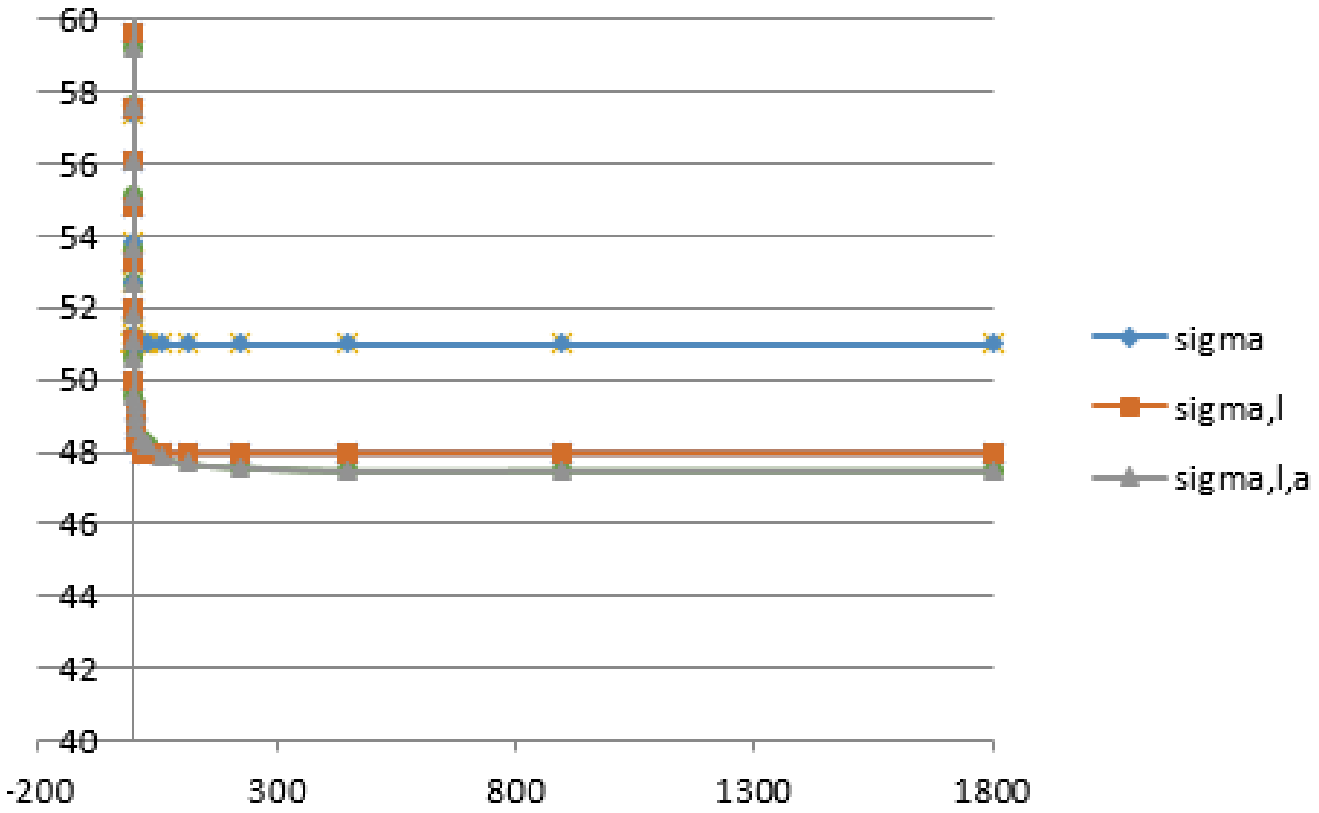
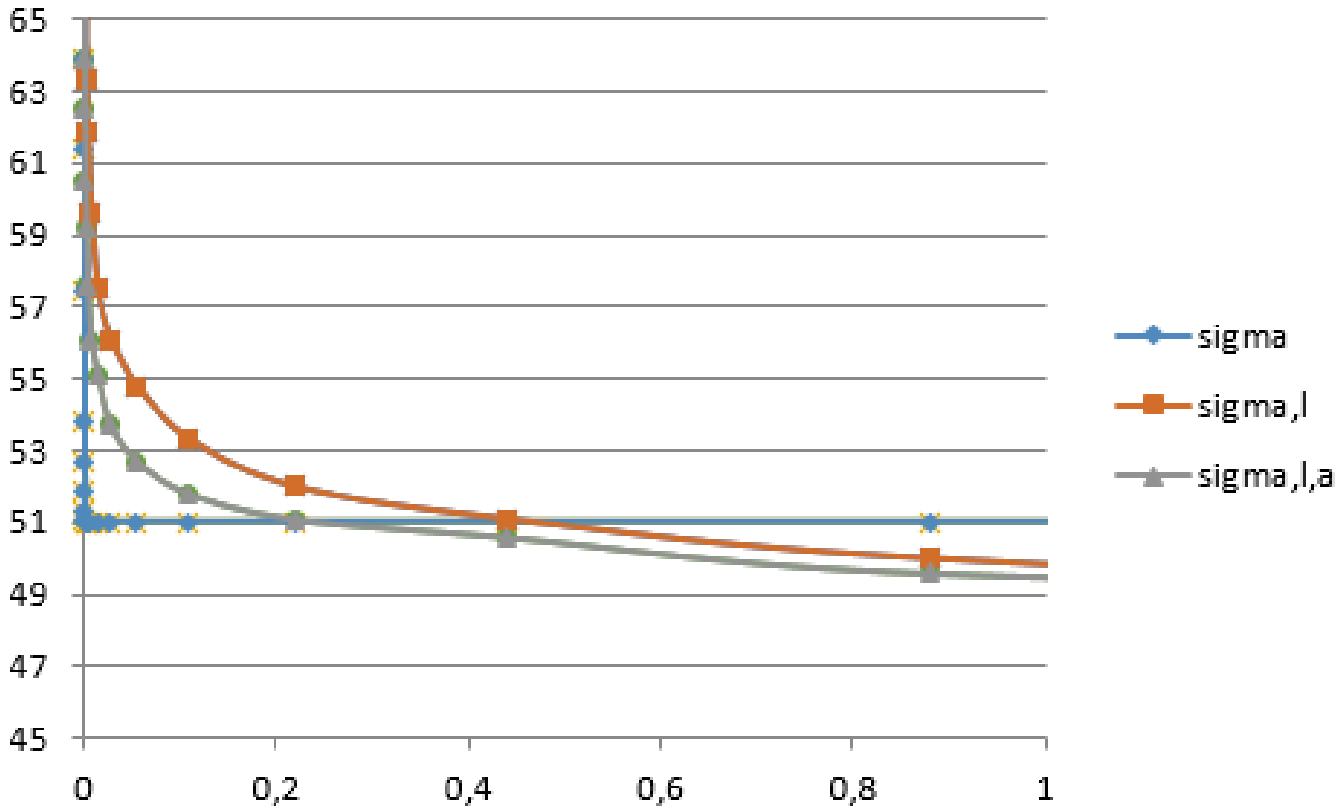


Figure : Makespan moyen obtenu dans le temps(s) pour chaque codage

Résultat sur une instances : Intersection σ,l/σ

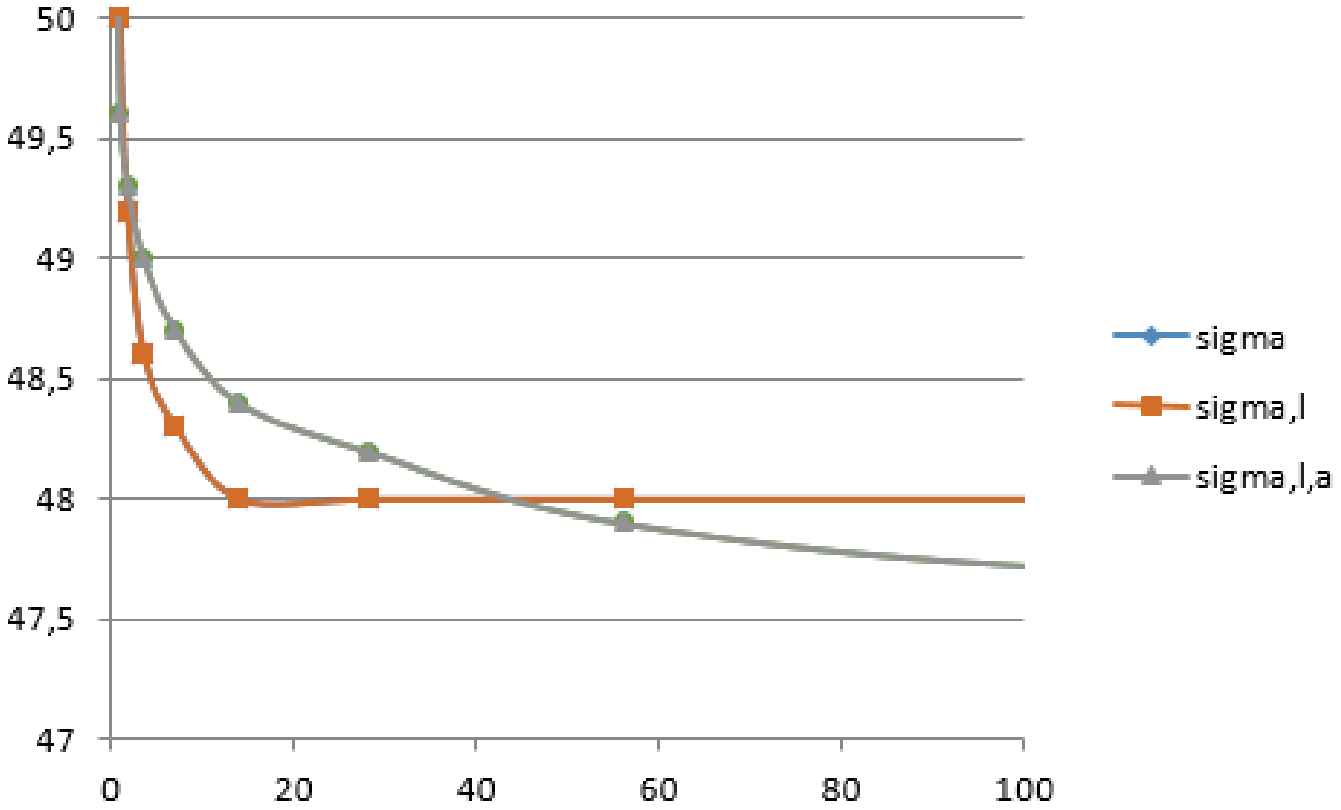
Zoom sur la première seconde d'exécution



$\Rightarrow$ au delà d'une seconde d'exécution le codage σ ne donne plus les meilleurs résultats.

Résultat sur une instances : Intersection σ,I/σ,I,a

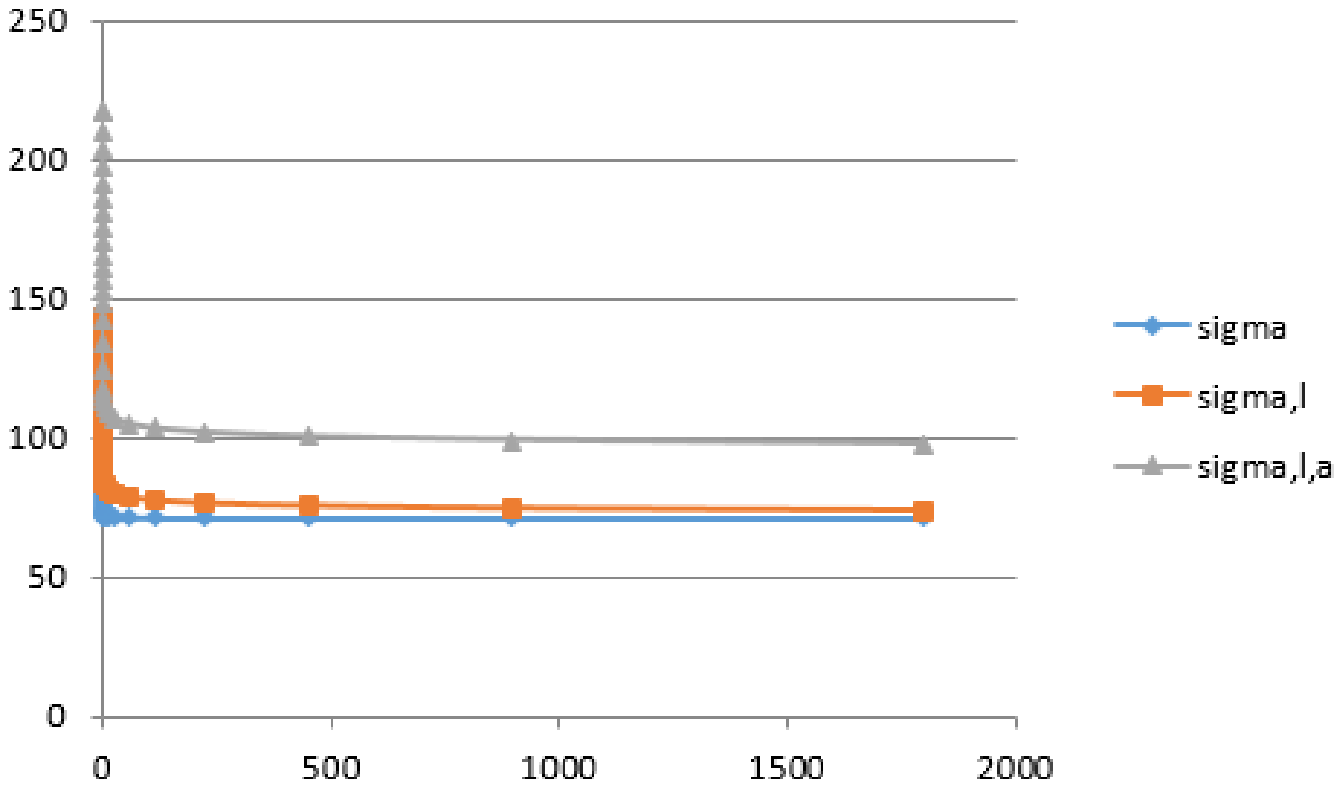
Zoom sur les 100 première secondes d'exécution



$\Rightarrow$ au delà de 100 secondes d'exécution le codage σ, *I* ne donne plus les meilleurs résultats.

Résultat sur les 480 instances de 30 tâches

Résultats obtenus sur les 30 minutes d'exécution



$\Rightarrow$ Pas de convergence en 30 minutes. Le codage σ donne les meilleurs résultats en moyenne, même après 30 minutes de résolution.

Sommaire

Conclusion et perspectives

- Proposition de trois méthodes approchées
- Production d'une librairie d'instances basée sur la PSPLIB
- Comparaison des trois codages et de leur intérêt dans le temps

Objectifs

- Comparer nos résultats avec ceux de la littérature comme borne inférieure
- Travailler sur le problème dans un contexte stochastique

Conclusion

Je vous remercie de votre attention, avez vous des questions ?