Programmation par Contraintes et Recherche Opérationnelle

Hybridation des techniques pour l'optimisation combinatoire

Sophie Demassey

sophie.demassey@emn.fr

Équipe Contraintes Discrètes, EMN/LINA

Plan général

- Préliminaires
- Techniques de PPC / Techniques de RO
- Comparaison des approches
- Hybridations: quelques illustrations

Préliminaires

Préliminaires

- Recherche Opérationnelle
- Contextes d’hybridation PPC/RO
- Cadre de la présentation

Recherche Opérationnelle

- RO : étude des problèmes d'optimisation liés aux organisations du monde réel
- Champs d'application :
  - administration et services (ex: emploi du temps),
  - production (ex: ordonnancement),
  - réseaux et transport (ex: tournées de distribution),
  - informatique (ex: systèmes de fichiers),
  - biologie (séquences ADN),...
- Discipline historique : Pascal (1650), Monge (1781, déblais et remblais), Harris (1913, gestion de stocks)
- RO moderne débute durant la 2nde guerre mondiale (O comme Opérations militaires) : implantation de radars,...

Recherche Opérationnelle

- Fournir des outils réutilisables d'analyse et de résolution
- Techniques issues des domaines:
  - Mathématiques appliquées : analyse fonctionnelle, algèbre linéaire, probabilités, graphes, jeux,...
  - Informatique : algorithmique, complexité, visualisation,...
- Principales méthodologies :
  - Algorithmes de graphes
  - Programmation linéaire, en nombres entiers
  - Programmation mathématique (continue, quadratique,...)
  - Programmation stochastique
  - Programmation dynamique
  - Meta-heuristiques
  - Optimisation multi-objectif et aide à la décision

Contextes d'hybridation PPC/RO

$\longrightarrow$ RO et PPC : différence d'échelle.

$\longrightarrow$ Hybridation dans le contexte (restreint au sens de la RO) de la résolution des problèmes combinatoires

Types d'hybridation principalement étudiés:

- Algorithmes de graphes et PPC ($\sim$1994)
- Programmation linéaire et PPC ($\sim$1995)
- Meta-heuristiques et PPC ($\sim$1998)

Cadre de la présentation

- Problèmes combinatoires : ensemble discret de solutions identifiable à $E \subseteq \mathbb{Z}^n$ (n entier positif possiblement grand) et défini implicitement par un ensemble de contraintes à satisfaire. Ex. (partition équilibrée):

$$E = \{(x_1, \ldots, x_n) \in \mathbb{Z}^n \text{ tq. } \sum_k x_{2k} = \sum_k x_{2k+1}\}$$

- Modèles déterministes : les paramètres sont connus avec exactitude
- Problèmes d'existence : trouver un élément de E (i.e de $\mathbb{Z}^n$ satisfaisant toutes les contraintes) ou prouver que $E = \emptyset$
- Problèmes d'optimisation : trouver un élément de E en lequel la restriction de $f : \mathbb{Z}^n \longrightarrow \mathbb{R}$ à E atteint sa valeur minimale (/maximale)
- Méthodes de résolution complètes des problèmes d'optimisation ($\neq$ méthodes approchées)

Techniques de PPC et techniques de RO

- Algorithmes de graphes
- Satisfaction de contraintes
  - Modélisation
  - Résolution
  - Optimisation
- Programmation linéaire (en nombres entiers)
  - Modélisation
  - Relaxation
  - Dualité
  - Branch-and-bound
  - Méthodes de décomposition

RO et théorie des graphes

- Une majorité de problèmes combinatoires ont un graphe comme support
  - Ordonnancement (chemins dans le graphe des précédences, couplages tâches-machines)
  - Transport (chemins dans le graphe des liaisons)
  - Emplois du temps (stables du graphe des incompatibilités)
  - Réseaux (flots et couvertures dans le graphe des liaisons)
- Des algorithmes de graphes (polynomiaux) peuvent être employés pour résoudre des sous-problèmes (relaxations) de problèmes combinatoires plus complexes
- Inversement, des applications réelles font apparaître de nouveaux problèmes de graphes.

Exemples d’algorithmes de graphes

- plus court chemin : Dijkstra (1959), Bellman (1958), Floyd (1962) $\leq O(n^3)$
- exploration (recherche de chemins, composantes connexes,...) $O(m)$
- flot max/coupe min : Ford-Fulkerson
- recouvrement par des arbres
- couplages
- recherche de cliques
- ...

Complexité des problèmes de graphes

- Tous les problèmes de graphes ne peuvent pas être résolus par des algorithmes de graphes :
  - Trouver un plus court chemin dans un graphe : $O(n^2)$, $O(nm)$
  - Trouver un plus court chemin passant par tous les sommets du graphe exactement une fois (Voyageur de commerce, TSP): *NP-difficile* $n!$ possibilités
  - Taille maximum résolue :
    - années 60: $n = 20$
    - années 80: $n = 1000$
    - années 00: $n = 1000000$
  - Progrès théoriques (PL et coupes), algorithmiques (algorithmes de résolution du PL, structures de données appropriées), technologiques (vitesse et mémoide des machines)...

Satisfaction de Contraintes (CSP)

- Modélisation
- Résolution
- Optimisation

CSP: Modélisation

- $X_1, \ldots, X_n$ Variables (entités du problème)
- $D_1, \ldots, D_n$ Domaines (valeurs possibles)
- $C_1, \ldots, C_m$ Contraintes (relations sur les variables)

- solution : instanciation complète des variables à une valeur de leur domaine satisfaisant toutes les contraintes

CSP: Contraintes

- Contraintes binaires mathématiques $=, >, <, \neq, \geq, \leq$
  propagation par cohérence d'arcs et cohérence aux bornes

- Contraintes n-aires, propagation par
  - cohérence d'arcs généralisée [Bessière&Regin99]
  - algorithmes spécialisés [Beldiceanu94]
    contraintes globales :
    `alldifferent`$([X_1, \ldots, X_m])$
    `element`$(n, [X_1, \ldots, X_m], v)$
    `cumulative`$([S_1, \ldots, S_m], [D_1, \ldots, D_n], [R_1, \ldots, R_n], L)$

- Contraintes logiques (disjonctives) $\vee, \Rightarrow, \Leftrightarrow$

CSP: Résolution

- Notion de cohérence
- Propagation de contraintes :
  - supprime des domaines les valeurs incohérentes
  - infère de nouvelles contraintes
- Recherche systématique: backtracking
  - stratégies de branchement (ordres de sélection variables, valeurs)
  - recherche/mémorisation des conflits (sous-instanciation incohérente)

CSP: Optimisation

- Recherche d'une instanciation complète optimale pour un critère donné $\min f : D_1 \times \cdots \times D_n \longrightarrow \mathbb{R}$
- Modélisation à l'aide d'une variable de coût Z de domaine intervalle dans $\mathbb{R}$ et de contraintes modélisant la relation $Z \leq f(X_1, \ldots, X_n)$
- Branch-and-bound: Chaque fois qu'une solution est trouvée, avec un coût z^*, une contrainte $Z < z^*$ est ajoutée au modèle et la recherche se poursuit dans l'espace restant.

Programmation linéaire

- Modélisation (PLNE)
- Relaxations
- Branch-and-bound
- Dualité
- Méthodes de décomposition

Programmation linéaire en Nombres Entiers (PLNE)

Formulation linéaire d'un problème d'optimisation combinatoire

$$
\begin{aligned}
z = \min \quad & \sum_j c_j x_j & \\
s.t. \quad & \sum_j a_{ij} x_j \geq b_i & i = 1..n \\
& x_j \geq 0 & j = 1..m \\
& x_j \in \mathbb{Z} & j = 1..m.
\end{aligned}
$$

Variantes :

- Programme Linéaire en Variables Binaires ($x_j \in \{0, 1\}$)
- Programme Linéaire Mixte (sous-ensemble de variables fractionnaires)

PLNE: Relaxation Linéaire ou Continue

$$\begin{aligned}
z = \min \quad & \sum_j c_j x_j & \\
s.t. \quad & \sum_j a_{ij} x_j \geq b_i & i = 1..n \\
& x_j \geq 0 & j = 1..m \\
& \boxed{x_j \in \mathbb{Z}} & j = 1..m.
\end{aligned}$$

- La PLNE est NP-difficile dans le cas général

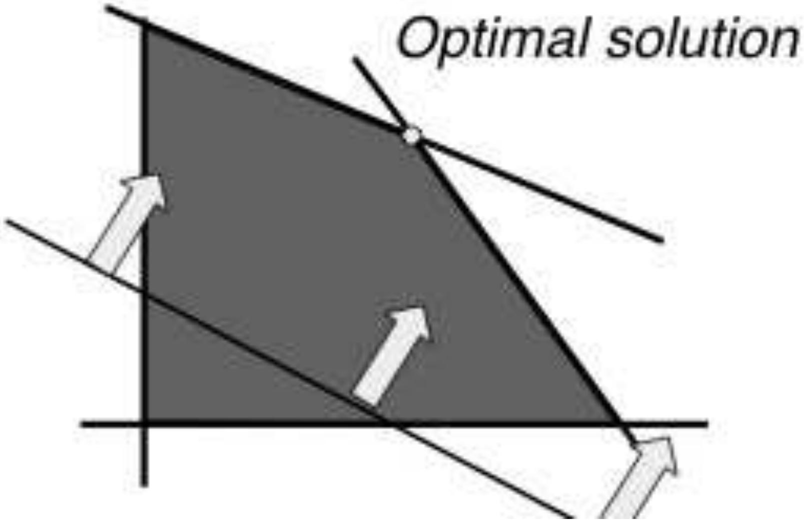
PLNE: Relaxation Linéaire ou Continue

$$
\begin{aligned}
\bar{z} = \min \quad & \sum_j c_j x_j & \\
s.t. \quad & \sum_j a_{ij} x_j \geq b_i & i = 1..n \\
& x_j \geq 0 & j = 1..m
\end{aligned}
$$

- La PLNE est NP-difficile dans le cas général
- La PL continue est polynomiale, algorithmes de résolution :
  – Simplexe [Dantzig47] (exponentiel dans le pire cas)
  – Points intérieurs [Karmarkar84] (primal-dual à barrière log)

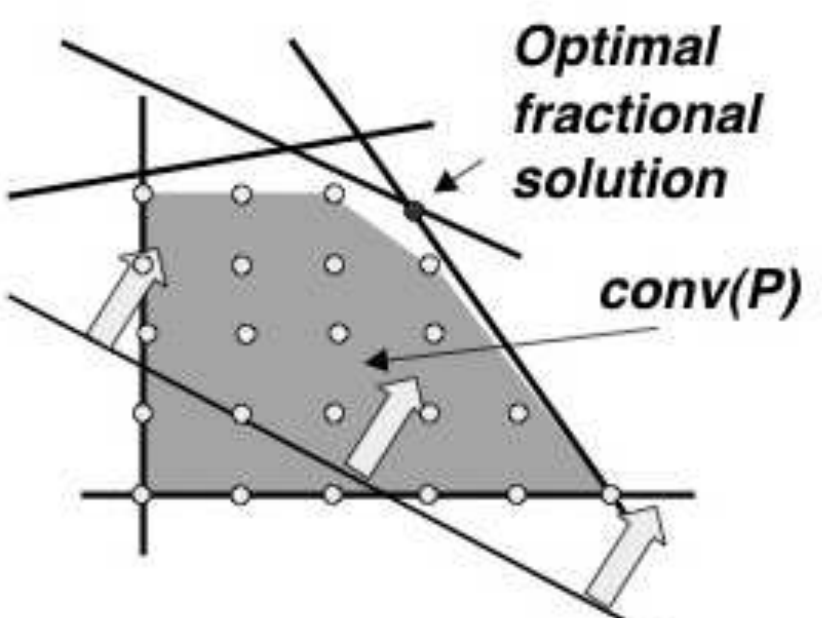
PL: Propriété géométrique

- L'ensemble des solutions d'un PL forme un polytope convexe
- S'il en existe, une solution optimale est située sur un de ses sommets
- Le simplexe est un algorithme itératif fini parcourant les sommets de proche en proche, en suivant la décroissance du coût



PLNE: Relaxation Continue

- La solution optimale de la relaxation continue est une affectation des variables à des valeurs fractionnaires satisfaisant les inégalités linéaires et minimisant la fonction objectif
- mais elle viole généralement les contraintes d'intégrité du PLNE (jamais si unimodularité)
- La valeur optimale $\bar{z}$ de la relaxation continue est une borne inférieure de la valeur optimale z du PLNE: $\bar{z} \leq z$



PLNE: Branch-and-bound

- Espace de recherche initial $\bar{E}$: ensemble des solutions de la relaxation du PLNE ($E \subseteq \bar{E}$)
- Recherche arborescente en séparant progressivement $\bar{E}$
- Dans chaque sous-espace, on cherche une solution optimale (relâchée). Son coût $\bar{z}$ est une estimation par défaut de la meilleure solution (réalisable) contenue dans le sous-espace.
- Le sous-espace est supprimé de la recherche si $\bar{z}$ est supérieure à la valeur z^* de la meilleure solution connue à présent (rencontrée à une feuille de l'arbre, ou calculée de manière heuristique).
- La séparation de l'espace se fait, par exemple, sur la variable la plus fractionnaire $\bar{x}_j$ dans la solution relâchée. On ajoute une contrainte linéaire dans les PL définissant chacun des deux sous-espaces : $x_j \leq \lfloor \bar{x}_j \rfloor$ (ss-espace gauche) et $x_j \geq \lceil \bar{x}_j \rceil$ (ss-espace droit)

PL: Dualité

Primal (P)	Dual (D)
$\bar{z} = \min \sum_j c_j x_j$	$\bar{z}' = \max \sum_i b_i y_i$
$s.t. \quad \sum_j a_{ij} x_j \geq b_i \qquad i = 1..n$	$s.t. \quad \sum_i a_{ij} y_i \leq c_j \qquad j = 1..m$
$x_j \geq 0 \qquad j = 1..m$	$y_i \geq 0 \qquad i = 1..n$

- théorème de dualité (si optima finis): $\bar{z} = \bar{z}'$.
- $\bar{x}$ et $\bar{y}$ solutions optimales de (P) et (D).
  $\bar{y}_i$ est une valeur duale associée à la contrainte $\sum_j a_{ij} x_j \geq b_i$ de (P)

Écarts complémentaires : $\begin{cases} \sum_j a_{ij}\bar{x}_j < b_i \Rightarrow \bar{y}_i = 0 \\ \bar{y}_i > 0 \Rightarrow \sum_j a_{ij}\bar{x}_j = b_i \end{cases}$

PLVB: Variable fixing

- variables 0-1
- Le coût réduit d'une variable x_i est l'écart de satisfaction, par la solution optimale $\bar{y}$ du dual, de la contrainte duale:

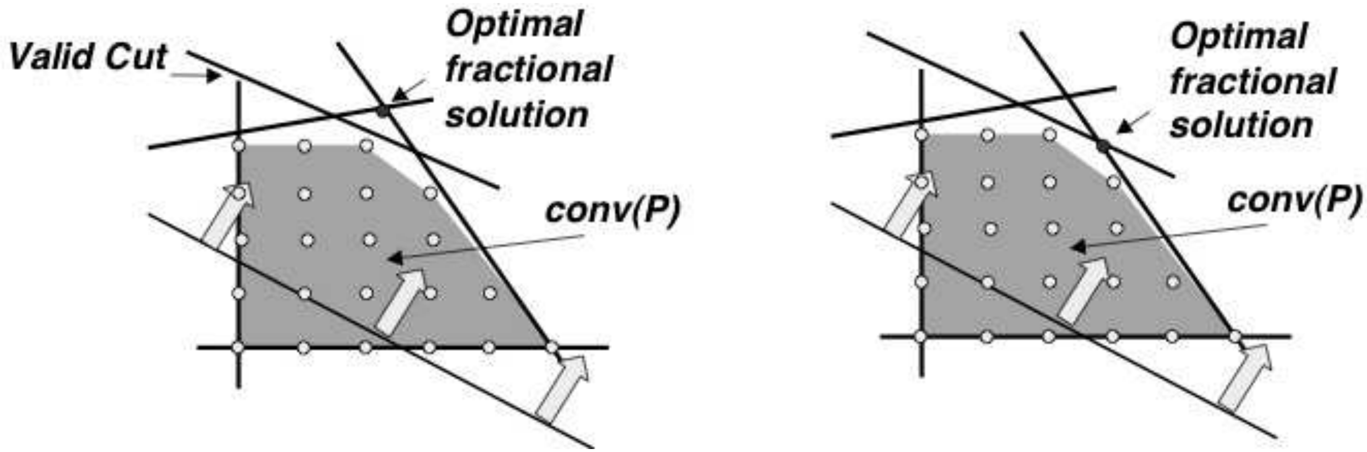
$$\bar{c}_i = c_j - \sum_i a_{ij}\bar{y}_i$$

- $\bar{c}_i$ mesure le coût supplémentaire à payer si $x_i = 1$
- raisonnement sur les coûts réduits permet de fixer des variables d'un PLVB sans avoir à brancher:
  - Soient z^* une borne supérieure au problème et $\bar{z}$ la valeur optimale relâchée
  - Si $\bar{z} + \bar{c}_i > z^*$ alors x_i peut être fixée à 0

PLNE: Relaxations et décompositions

- Relaxation $\bar{P}$: sous-problème obtenu par suppression de contraintes
  - $E \subseteq \bar{E}$
  - $\bar{f}(x) \leq f(x)$ pour tout $x \in E$
- Décomposition : séparation en plusieurs sous-problèmes
  + techniques itératives de recomposition
  - Relaxation continue et génération de coupes
  - Relaxation lagrangienne et résolution du dual lagrangien
  - Décomposition de Danzig-Wolfe et génération de colonnes
  - Décomposition de Benders

PLNE: Génération de coupes



- inégalité valide : inégalité linéaire redondante avec les contraintes initiales du PLNE
- coupe : inégalité valide violée par la solution optimale de la relaxation
- coupes polyedriques et facettes : caractérisation de l'enveloppe convexe de l'ensemble des solutions du PLNE
  - coupes génériques pour la relaxation continue (ex: Chvátal-Gomory)
  - coupes renforçant la structure (relâchée) de sous-problèmes (ex: suppression des cycles dans les TSP)

PLNE: Génération de coupes

- la procédure itérative (relaxation + coupe de la solution courante) permet pour certains types de coupes (ex: Chvátal-Gomory) de recomposer entièrement le PLNE initial en un nombre fini d'étapes :
  conv$(\bar{P}_{+coupes})$ = conv$(P) \quad \Rightarrow \quad \bar{z}_{+coupes} = z$

- la convergence de ces procédures est généralement trop lente pour être appliquée jusqu'au terme

- une recherche arborescente est alors nécessaire pour compléter la recherche. branch-and-cut : génération de coupes à chaque noeud de l'arbre de recherche, valides pour l'espace entier ou pour le sous-espace courant.

- la génération de coupes est une méthode d'inférence basée sur le coût, pour réduire l'espace de recherche et donner une meilleure estimation de la borne: $\bar{z} \leq \bar{z}_{+coupes} \leq z$

- l'ajout d'un nombre excessif d'inégalités valides peut ralentir la résolution de la relaxation linéaire

PLNE: Relaxation lagrangienne

- suppression des contraintes linéaires difficiles et pénalisation de leur violation dans l'objectif

$$
\begin{aligned}
z_\lambda = \min \quad & \sum_j c_j x_j + \sum_i \lambda_i (b'_i - \sum_j a'_{ij} x_j) & \\
s.t. \quad & \sum_j a_{ij} x_j \geq b_i & i = 1..n \\
& x_j \geq 0 & j = 1..m \\
& x_j \in \mathbb{Z} & j = 1..m.
\end{aligned}
$$

- résolution du dual lagrangien : $z_{\text{Lag}} = \max\{z_\lambda \mid \lambda \geq 0\}$ (optimisation continue) par des méthodes itératives : algorithme du sous-gradient, algorithme des faisceaux, génération de coupes.

- $\bar{z} \leq z_{\text{Lag}} \leq z$

PL: Génération de colonnes (Dantzig-Wolfe)

- résolution des PL (fractionnaires) avec un nombre excessif de variables (la plupart étant nulles dans toute solution optimale)
- résoud le PL restreint à un sous-ensemble de variables ($\bar{z}_0 \geq \bar{z}$)
- génère itérativement les variables x_j/colonnes $(c_j, a_{1j}, \ldots, a_{nj})$ manquantes
- correspondantes à des contraintes du dual violées par la solution duale courante (relâchée)

$$\sum_i a_{ij}\bar{y}_i > c_j$$

- la procédure s'arrête avec $\bar{z}_{\text{DW}} = \bar{z}$ sans que toutes les colonnes n'aient besoin d'être générées
- PLNE: génération de colonnes à chaque noeud de l'arbre de recherche (branch-and-price)

PLNE: Décomposition de Benders

- s'applique aux PLNE identifiant deux ensembles distincts de variables: x difficiles et y faciles
- le sous-problème obtenu en instanciant toutes les variables x est facile à résoudre
- la résolution de ce sous-problème permet de générer une coupe éliminant l'instanciation courante de x, ainsi que, par dualité, d'autres instanciation de x ne menant pas à une meilleure solution.
- la décomposition de Benders permet la résolution exacte des PLNE en un nombre fini d'étapes $z_{\text{Ben}} = z$.
- si la convergence est trop lente, la procédure peut être arrêtée prématurément et fournit alors un encadrement de l'optimum (borne inférieure et une solution réalisable).

Comparaison des approches PLNE/PPC

Comparaison des approches PLNE/PPC

- Modélisation
- Recherche arborescente
- Relaxations et décompositions
- Inférences

$\longrightarrow$ Voies d'intégration

Comparaison PLNE/PPC: modélisation

- Tout problème combinatoire peut être modélisé comme un CSP ou comme un PLNE, et souvent de plusieurs manières possibles.
- Certaines modélisations nécessitent l'introduction d'un nombre fini MAIS exponentiel de variables ou de contraintes.
  $\longrightarrow$ techniques de résolution adaptées.
- PPC: formulation déclarative des problèmes. Les contraintes globales encapsulent les sous-structures du problème.
  $\longrightarrow$ Modélisation modulaire donc plus flexible.
  $\longrightarrow$ Hormis dans les contraintes globales d'optimisation, les sous-structures sont modélisées indépendamment du critère d'optimalité.
- PLNE: la structure d'un problème est modélisée de manière répartie sur quasiment l'ensemble de contraintes du PLNE.
  $\longrightarrow$ Cette représentation n'offre pas la même flexibilité qu'un CSP.
  $\longrightarrow$ La relaxation d'une contrainte, linéaire ou d'intégralité, peut détruire la structure.

PLNE/PPC: recherche systématique de l'optimum

- branch-and-bound = relaxation + réduction + séparation
- stratégies de séparation
  - PL: séparation sur les variables mais aussi sur des combinaisons linéaires de variables
  - PL: recherche guidée par la solution optimale relâchée (ex: var la plus fractionnaire)
- recherche et réparation de conflits (explications/nogoods)
  - PPC: identification aisée des conflits. Réparation/mémorisation: backjumping, dynamic backtracking,...
  - PL: Benders (pour des modèles particuliers). Identification par dualité.

Comparaison PL/PPC: relaxations

Relaxation = sous-problème

- PPC:
  - produit des domaines
  - contraintes globales: filtrage spécifique à la structure modélisée
  - pas ou peu de back-propagation: filtrage basé sur le coût
- PL:
  - relaxations: continue, lagrangienne,...
  - perte de la structure caractérisant la réalisabilité
  - méthodes de décomposition: séparation en sous-problèmes tous liés au coût

Comparaison PL/PPC: inférences

inférence = générer des contraintes *faciles* à partir des contraintes *difficiles* (notions différentes selon l'approche) pour resserer la relaxation

- PPC:
  - propagation de contraintes
  - contraintes faciles: contraintes de domaines
  - inférence par raisonnement logique, mais aussi calculatoire dans le cas des contraintes globales
- PL:
  - génération de coupes linéaires
  - contraintes difficiles: contraintes d'intégralité
  - inférences basée davantage sur le coût que sur la réalisabilité (ex: coupes de Chvatal-Gomory)
  - raisonnement basé sur la dualité pour recombiner les sous-problèmes dans les méthodes de décomposition

Comparaison PLNE/PPC: voies d'intégration

- dans les 2 approches: séparation en sous-problèmes traités de manière indépendante
  - algorithmes de RO comme techniques de filtrage des contraintes globales en PPC
  - résolution de sous-problèmes par PPC dans les méthodes de décomposition de la PL
- contraintes globales d'optimisation pour un traitement plus efficace du coût en PPC
- raisonnement sur la réalisabilité pour la génération de coupes en PL

Hybridations des techniques RO/PPC : quelques illustrations

Exemples d'intégration des techniques RO/PPC

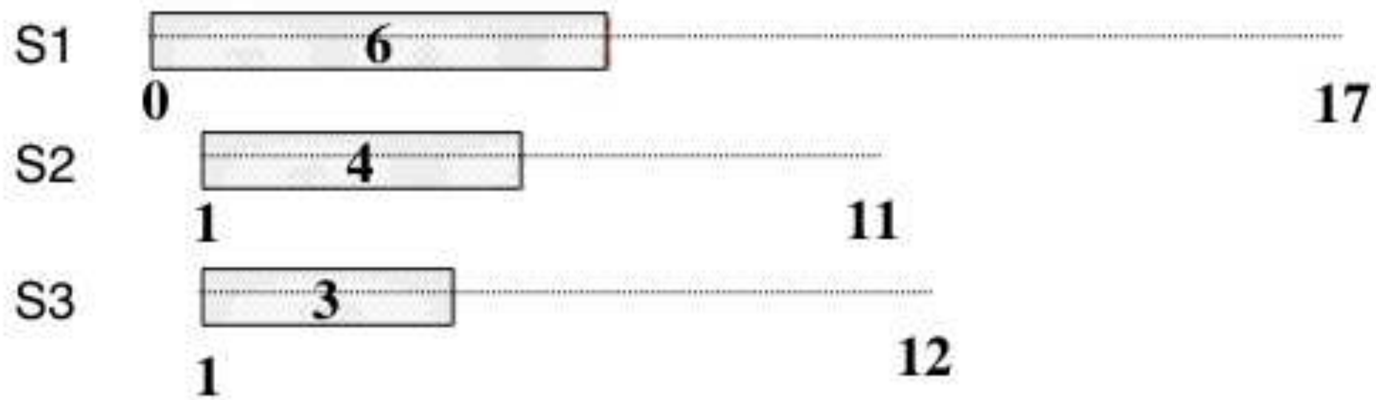
- RO pour les contraintes globales
  - `disjunctive` (edge-finding)
  - `global-cardinality` (algorithme de flot max)
- linéarisation pour les contraintes globales d'optimisation
  - `all-different` (filtrage par PL)
- Modèles mixtes PL/PPC et décompositions hybrides
  - RCPSP (coupes linéaires inférées par PPC)
  - Emploi du temps (génération de colonnes hybrides)

RO: filtrage pour les contraintes globales

- emploi d'algorithmes issues de la RO de manière transparente pour l'utilisateur d'un solveur PPC
- edge-finding: technique d'ordonnancement pour la contrainte globale `disjunctive`
- flot max: algorithme de graphe pour la contrainte globale `global-cardinality`

contrainte globale `disjunctive`

- Problème d'ordonnancement : un ensemble de tâches $a_1, \ldots, a_n$ à exécuter séquentiellement sur une machine
- une tâche a_i est exécutée entre les instants S_i (variable) et $S_i + p_i$
- `disjunctive`$((S_i)_n, (p_i)_n)$ est satisfaite si tous les intervalles $[S_i, S_i + p_i]$ sont disjoints.
- $[ES_i, LS_i]$: domaine de la variable S_i



Edge-finding

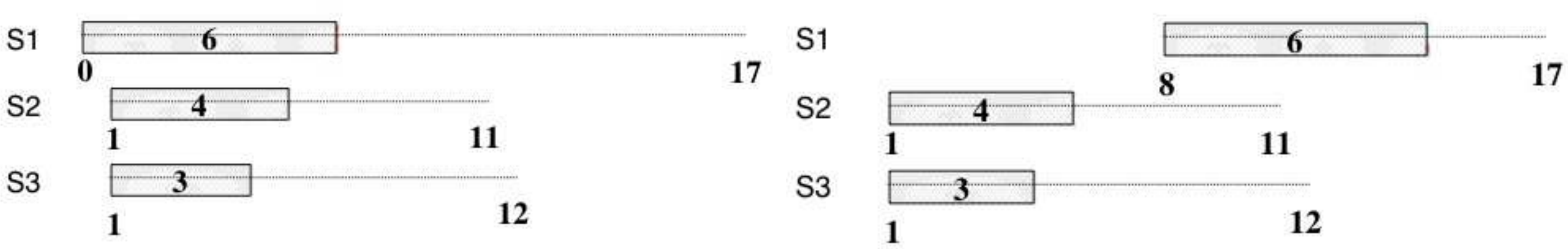
- Règle primale du edge-finding: Comme

$$\min(ES_1, ES_2, ES_3) + p_1 + p_2 + p_3 > \max(LS_2 + p_2, LS_3 + p_3)$$

alors la tâche a_1 est placée après l'ensemble de tâches $\{a_2, a_3\}$ dans tout ordonnancement réalisable. On en déduit l'ajustement du domaine de S_1:

$$ES_1 \leftarrow \max(ES_2 + p_2, ES_3 + p_3, \min(ES_2, ES_3) + p_2 + p_3).$$

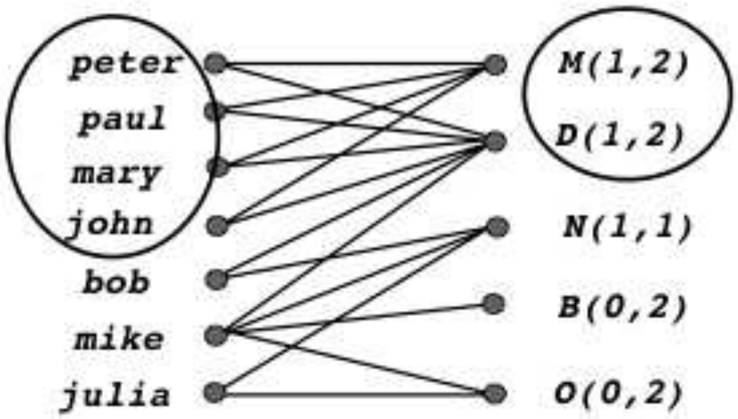
- Algorithme de détection et de filtrage en $O(n \log n)$ [Carlier&Pinson95]

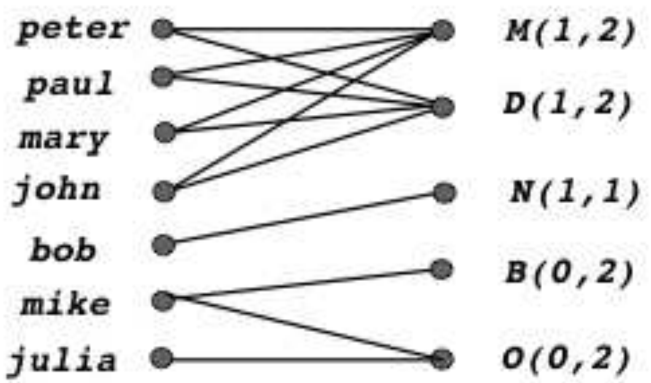


contrainte globale `global-cardinality`

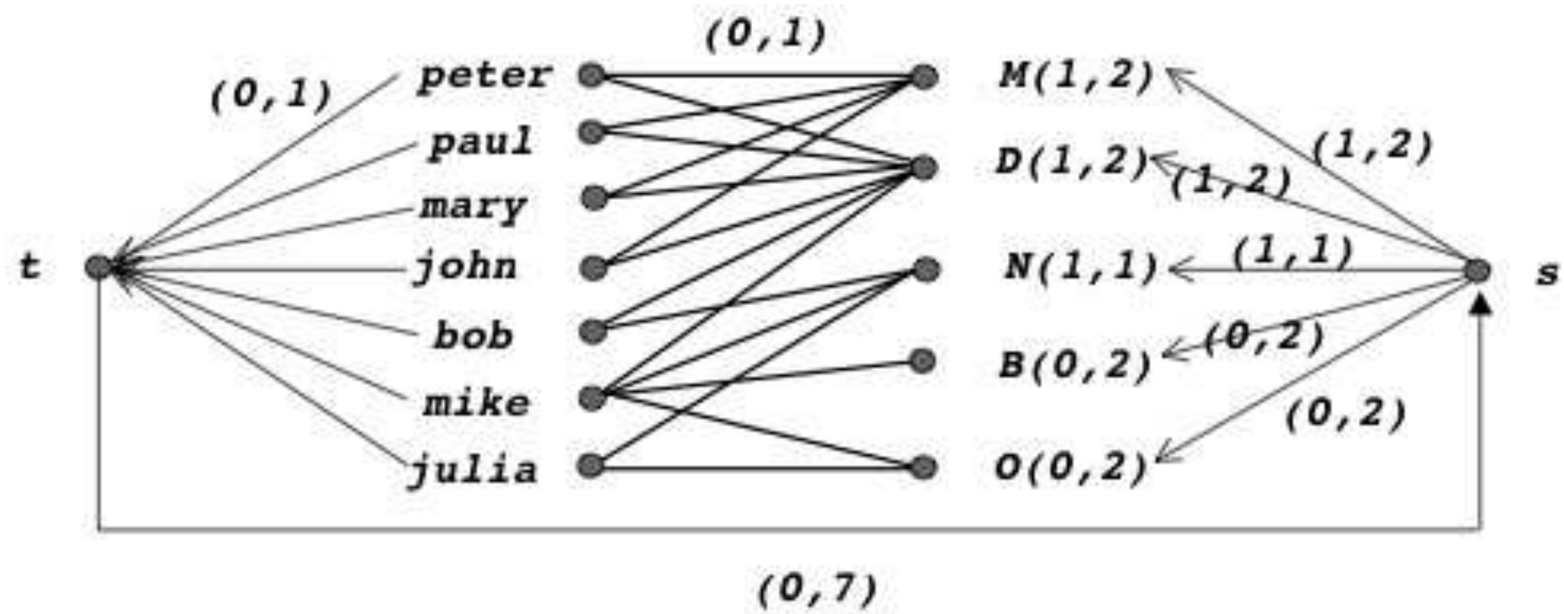
- `global-cardinality`$((X_i)_n, (v_k)_m, (N_k)_m)$ [Régin96] est satisfaite si, pour tout k, N_k correspond au nombre de variables $(X_i)_n$ instanciées à la valeur v_k.

- la contrainte est cohérente s'il existe un flot max de valeur n dans un graphe biparti





Filtrage par calcul du flot max dans un graphe biparti



- algorithme de flot max basé sur le graphe résiduel associé à un flot f: $res(u, v) = \max(ub(u, v) - f(u, v), f(u, v) - lb(u, v))$
- Soit f un flot max, une valeur a de X est incohérente ssi $f(a, X) = 0$ et si a et X n'appartiennent pas à la même composante fortement connexe dans le graphe résiduel.

PL pour le filtrage en optimisation

- Contraintes globales d'optimisation
- Linéarisation
- `cost-all-different`

$\longrightarrow$ utiliser le coût pour filtrer
$\longrightarrow$ solution optimale pour guider

Contraintes globales d'optimisation

- Un coût c_{ij} est associé à chaque instanciation valeur/variable $X_i = j$
- problème: déterminer une instanciation complète réalisable qui minimise la somme des coûts
- le coût total représenté par une variable telle que $Z = \sum_i c_{iX_i}$
- variante d'optimisation d'une contrainte globale :
  - assure le filtrage des variables X au sens de la contrainte globale
  - assure le filtrage de Z à partir des variables X
  - assure le filtrage des variables X à partir de Z (back-propagation)
  - retourne une instanciation des X qui minimise Z
    $\longrightarrow$ pour guider la recherche
  - notion de regret $r_{ij} =$ le surcoût si $X_i = j$

Linéarisation

- modéliser l'ensemble des solutions d'une contrainte globale d'optimisation (ou de tout un CSOP) en un PLNE
- linéarisation automatique en un PLVB [Refalo00]:
  - $X_i = j \longrightarrow x_{ij} = 1$
  - $X_i \in D_i \longrightarrow \sum_{j \in D_i} x_{ij} = 1$
- linéarisations ad-hocs
- L'ensemble des solutions réalisables du modèle linéaire peut être identique à celui de la contrainte ou bien l'inclure strictement (relaxation).

`all-different` avec coûts

- `all-different`$((X_i)_n, (c_{ij})_{n\times n}, Z)$: satisfaite ssi toutes les valeurs des varaibles X_i dans $D = \{1, \ldots, n\}$ sont différentes et si $Z = \sum_i c_{iX_i}$.
- linéarisation (problème d'affectation):

$$
\begin{array}{rll}
z = \min & \displaystyle\sum_i \sum_j c_{ij} x_{ij} & \\
s.t. & \displaystyle\sum_j x_{ij} = 1 & i = 1..n \\
 & \displaystyle\sum_i x_{ij} = 1 & j = 1..n \\
 & x_{ij} \in \{0, 1\} & i, j = 1..n
\end{array}
$$

`all-different` avec coûts

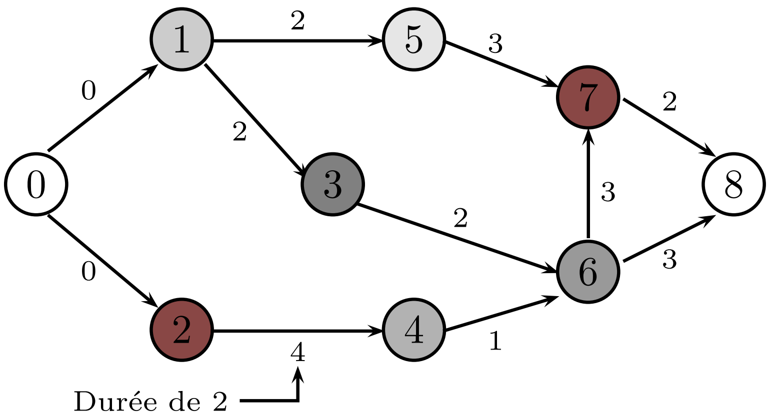
- matrice unimodulaire $\Rightarrow$ solution de la relaxation continue est entière
- PLVB polynomial
- Algorithme Hongrois ($O(n^3)$, incrémental $O(n^2)$)
- retourne le coût réduit $\bar{c}_{ij}$ associé aux variables x_{ij}
  - filtrage de la borne inférieure de Z avec la valeur optimale du PL
  - back-propagation par variable-fixing:
    si $lb(Z) + \bar{c}_{ij} > ub(Z)$ alors $X_i \neq j$

Double modélisation PL/PPC

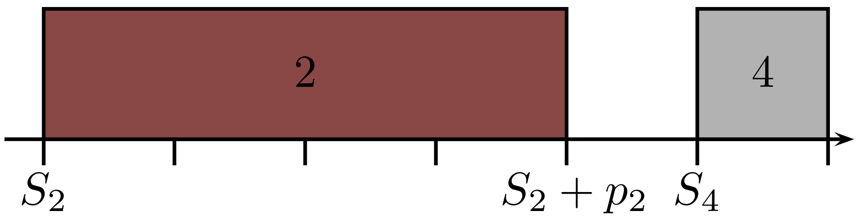
- RCPSP un problème général d'ordonnancement
- bonnes modélisations PL ?
- modélisation PPC
- inférer du modèle PPC vers le modèle PL:
  - prétraitement du modèle linéaire
  - génération de coupes linéaires

Ordonnancement de projet à contraintes de ressources

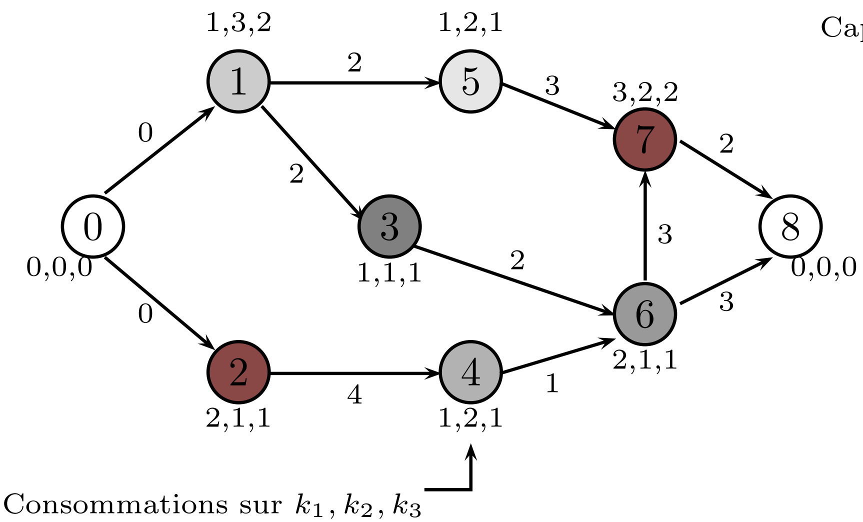
RCPSP (Resource Constrained Project Scheduling Problem)



- $\mathcal{A} = \{1, \ldots, n\}$ activités,
  0 et $n+1$ activités fictives
- p_i durée, S_i date de début
- contraintes de précédence :
  $(i, j) \in E \Rightarrow S_j \geq S_i + p_i$



Ordonnancement de projet à contraintes de ressources

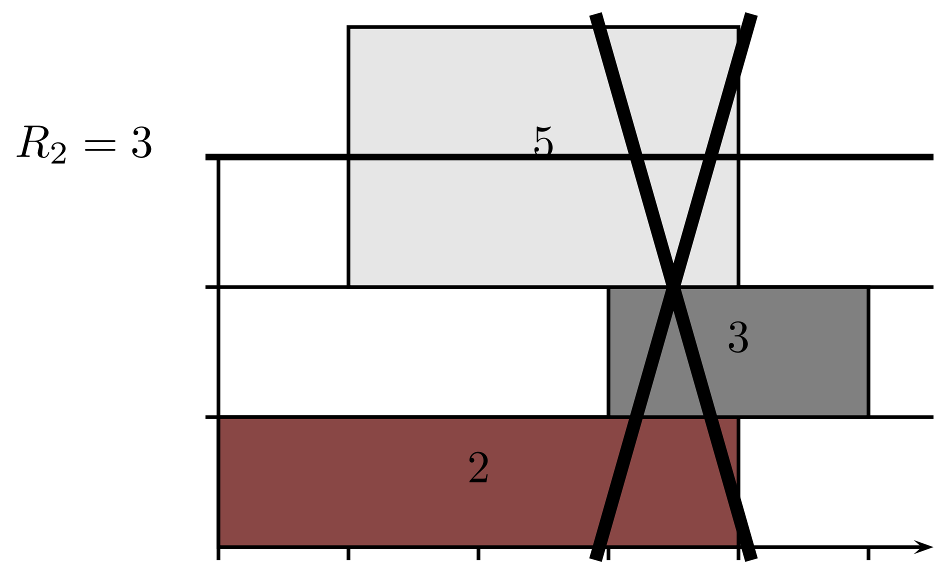


Capacités des ressources k_1, k_2, k_3 :

$R_1 = 3$

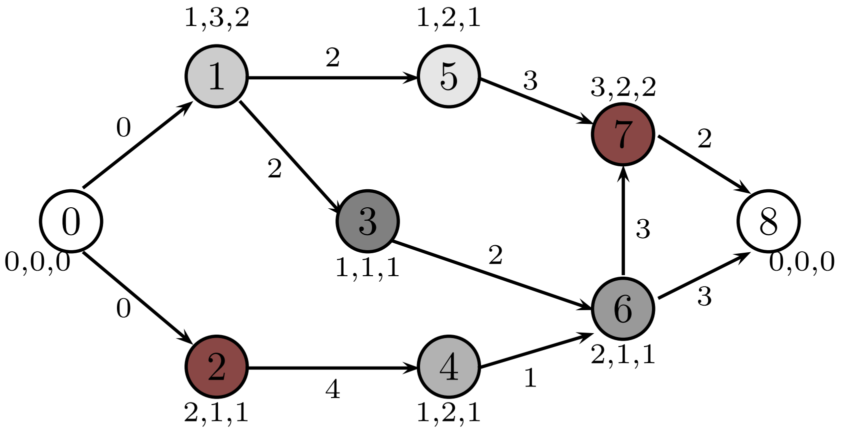
$R_2 = 3$

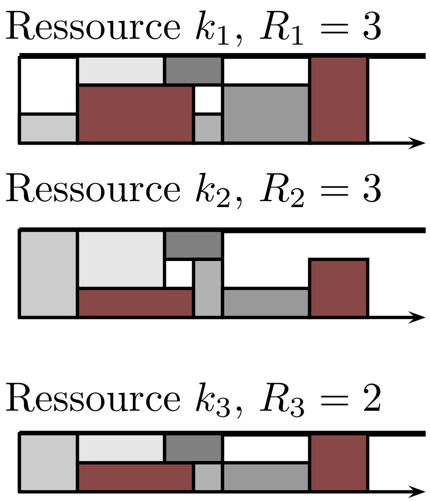
$R_3 = 2$



- $\mathcal{R} = \{1, \ldots, m\}$ ressources
- R_k capacité, r_{ik} consommation
- contraintes de ressources : $\sum_{i \in \mathcal{A}_t} r_{ik} \leq R_k$

Ordonnancement de projet à contraintes de ressources





$S = (0, 0, 2, 5, 6, 2, 7, 10)$

- $S = (S_0, \dots, S_{n+1})$ ordonnancement réalisable
- durée totale S_{n+1} minimale

RCPSP: 1er modèle linéaire

$$\min \sum_{t \in T} t y_{(n+1)t} \tag{1}$$

sujet à :

$$\sum_{t=0}^{T} y_{it} = 1 \qquad \forall\, i \in \mathcal{A} \tag{2}$$

$$\sum_{t=0}^{T} t(y_{jt} - y_{it}) \geq p_i \qquad \forall\, (i,j) \in E \tag{3}$$

$$\sum_{i \in \mathcal{A}} r_{ik} \sum_{\tau = t - p_i + 1}^{t} y_{i\tau} \leq R_k \qquad \forall\, k \in \mathcal{R}, \forall\, t \in T \tag{4}$$

$$y_{it} \in \{0,1\} \qquad \forall\, i \in \mathcal{A}, \forall\, t \in T$$

- $y_{it} = 1$ si i débute au temps t: $S_i = t$
- défaut: nombreuses variables 0-1 dans chaque contraintes

RCPSP: 2nd modèle linéaire

$$\min \sum_{t=0}^{T} t y_{(n+1)t} \tag{1}$$

sujet à :(2), (3),

$$\sum_{l \in \mathcal{F}_i} \sum_{t=0}^{T} x_{lt} = p_i \qquad \forall\, i \in \mathcal{A} \tag{5}$$

$$\sum_{l \in \mathcal{F}} x_{lt} \leq 1 \qquad \forall\, t \in T \tag{6}$$

$$y_{it} \geq \sum_{l \in \mathcal{F}_i} x_{lt} - \sum_{l \in \mathcal{F}_i} x_{lt-1} \qquad \forall\, t \in T, \forall\, i \in \mathcal{A} \tag{7}$$

$$\boxed{x_{lt} \in \{0,1\}},\ x_{l(-1)} = 0 \qquad \forall\, l \in \mathcal{F}, \forall\, t \in T \tag{8}$$

$$y_{it} \in \{0,1\}, \qquad \forall\, i \in \mathcal{A}, \forall\, t \in T \tag{9}$$

- $x_{lt} = 1$ si l est l'ensemble admissible en cours d'exécution au temps t.
- nombre exponentiel de variables x_{lt} !

RCPSP: 3eme modèle linéaire

$$\min S_{n+1} \tag{10}$$

sujet à :

$$
\begin{aligned}
& x_{ij} = 1 && \forall\, (i,j) \in E && (11)\\
& x_{ij} + x_{ji} \leq 1 && \forall\, (i,j) \in \mathcal{A}^2, i < j && (12)\\
& x_{ik} \geq x_{ij} + x_{jk} - 1 && \forall\, (i,j,k) \in \mathcal{A}^3 && (13)\\
& S_j - S_i \geq -M + (p_i + M)x_{ij} && \forall\, (i,j) \in \mathcal{A}^2 && (14)\\
& \boxed{\sum_{(i,j)\in C^2} x_{ij} \geq 1} && \forall\, C \in \mathcal{C}_m && (15)\\
& x_{ij} \in \{0,1\},\ x_{ii} = 0 && \forall\, (i,j) \in \mathcal{A}^2 &&\\
& S_i \geq 0 && \forall i \in \mathcal{A} &&
\end{aligned}
$$

- $x_{ij} = 1$ si $i \longrightarrow j$
- nombre exponentiel de contraintes (15) !!

RCPSP: modèle CSP

- contraintes de précédence :
  modélisation basée sur la distance minimale : $S_j - S_i \geq b_{ij}$
  variables $X_{ij} = S_j - S_i$, domaines $[b_{ij}, -b_{ji}]$

RCPSP: modèle CSP

- contraintes de précédence :
  modélisation basée sur la distance minimale : $S_j - S_i \geq b_{ij}$
  variables $X_{ij} = S_j - S_i$, domaines $[b_{ij}, -b_{ji}]$
- contraintes de ressources : `cumulative` pour chaque ressource

RCPSP: modèle CSP

- contraintes de précédence :
  modélisation basée sur la distance minimale : $S_j - S_i \geq b_{ij}$
  variables $X_{ij} = S_j - S_i$, domaines $[b_{ij}, -b_{ji}]$
- contraintes de ressources : `cumulative` pour chaque ressource
- propagation de contraintes : augmentation des b_{ij}
  ou déduction de séquencements obligatoires

$$i \longrightarrow j \;\Rightarrow\; b_{ij} = \max(b_{ij}, p_i)$$

  - $i \longrightarrow j$, i précède j
  - $i \| j$, i en parallèle à j
  - $i - j$, i en disjonction avec j

RCPSP: Propagation de contraintes

- contraintes de précédence :
  3-cohérence aux bornes par plus court chemin $O(n^3)$
  $$b_{ik} = \max(b_{ik}, b_{ij} + b_{jk})$$

RCPSP: Propagation de contraintes

- contraintes de précédence :
  3-cohérence aux bornes par plus court chemin $O(n^3)$
  $$b_{ik} = \max(b_{ik}, b_{ij} + b_{jk})$$
- contraintes de ressources :
  - paires disjonctives : $i - j$ et $b_{ji} > -p_i \;\Rightarrow\; i \rightarrow j$
  - edge-finding sur les ensembles disjonctifs

RCPSP: Propagation de contraintes

- contraintes de précédence :
  3-cohérence aux bornes par plus court chemin $O(n^3)$
  $$b_{ik} = \max(b_{ik}, b_{ij} + b_{jk})$$
- contraintes de ressources :
  - paires disjonctives : $i - j$ et $b_{ji} > -p_i \ \Rightarrow \ i \rightarrow j$
  - edge-finding sur les ensembles disjonctifs
- shaving sur les séquencements :
  - réfutation d'une contrainte $i \rightarrow j$, $i \| j$ ou $j \rightarrow i$ après propagation :
    $i \rightarrow j \Rightarrow S_l - S_h \geq b_{hl}^{i \rightarrow j}$
  - $[b_{hl}^{i \rightarrow j}, -b_{lh}^{i \rightarrow j}] = \emptyset \ \Rightarrow \ \neg(i \rightarrow j)$
  - $b_{hl} = \max(b_{hl}^{i \rightarrow j}, b_{hl}^{i \| j}, b_{hl}^{j \rightarrow i})$

Prétraitement du PL par PPC

- fixation de variables :

$$
\begin{aligned}
x_{ij} &= 1 \qquad \text{si } b_{ij} \geq p_i \\
x_{ij} &= 0 \qquad \text{si } b_{ji} \geq 1 - p_i
\end{aligned}
$$

Prétraitement du PL par PPC

- fixation de variables :

$$
\begin{aligned}
x_{ij} &= 1 \qquad \text{si } b_{ij} \geq p_i \\
x_{ij} &= 0 \qquad \text{si } b_{ji} \geq 1 - p_i
\end{aligned}
$$

- resserement des contraintes de précédence:

$$S_j - S_i \geq b_{ij} \qquad \text{si } b_{ij} \geq p_i \tag{19}$$

$$S_j - S_i \geq b_{ij} + (p_i - b_{ij})x_{ij} \qquad \text{si } 1 - p_j \leq b_{ij} < p_i \tag{20}$$

$$S_j - S_i \geq (1 - p_j) + (p_i + p_j - 1)x_{ij} + (b_{ij} + p_j - 1)x_{ji} \qquad \text{si } b_{ij} < 1 - p_j. \tag{21}$$

Coupes linéaires inférées par PPC

relaxation des contraintes de ressources $\sum_{(i,j)\in C} x_{ij} \geq 1$ considérées en partie comme des coupes :

- basées sur le shaving
  - si $b_{hl} < p_h \leq b_{hl}^{i\rightarrow j}$, alors $x_{hl} \geq x_{ij}$
  - $S_l - S_h \geq b_{hl}^{i\|j} + (b_{hl}^{i\rightarrow j} - b_{hl}^{i\|j})x_{ij} + (b_{hl}^{j\rightarrow i} - b_{hl}^{i\|j})x_{ji}$

Coupes linéaires inférées par PPC

relaxation des contraintes de ressources $\sum_{(i,j)\in C} x_{ij} \geq 1$ considérées en partie comme des coupes :

- basées sur le shaving
  - si $b_{hl} < p_h \leq b_{hl}^{i\rightarrow j}$, alors $x_{hl} \geq x_{ij}$
  - $S_l - S_h \geq b_{hl}^{i\|j} + (b_{hl}^{i\rightarrow j} - b_{hl}^{i\|j})x_{ij} + (b_{hl}^{j\rightarrow i} - b_{hl}^{i\|j})x_{ji}$
- traduction du edge-finding

$$S_j \geq S_l + \sum_{i\in C\setminus\{j\}} p_i x_{ij} + \sum_{i\in C\setminus\{l\}} b_{li} x_{il} \quad \forall j, l \in C$$

Génération de colonnes hybride

- Un problème d’emploi du temps
- Décomposition: 2 sous-problèmes à résoudre
- Quelle approche choisir ?
- Recomposition par génération de colonnes

Un problème d'emploi du temps

- Une journée de travail (7h-19h) découpée en 24 demi-heures $t = 1, 2, \ldots, 24$
- $e \in E$ les employés, $a \in A$ les activités à effectuer :
  - r_{at}: nombre minimal d'employés requis à la période t
  - c^e_{at}: coût d'affecter l'employé e à a à la période t
- Créer l'emploi du temps des employés pour la journée tel que :
  1. la charge de travail est couverte
  2. le coût total est minimal

ETP: modèle linéaire du pb de couverture optimale

Problème de couverture à coût minimal (optimisation):

- NP-difficile mais se modélise et se résoud aisément en PL.
- Variables binaires:
  $x^e_{at} = 1$ si l'employé e est affecté à l'activité a au temps t, $x^e_{at} = 0$ sinon.

$$\min \quad \sum_{e \in E} \sum_{a \in A} \sum_{t \in T} c_{at} x^e_{at} \tag{22}$$

$$s.t. \quad \sum_{e \in E} x^e_{at} \geq r_{at} \qquad \forall\, a \in A, \forall\, t \in T, \tag{23}$$

$$\sum_{a \in A} x^e_{at} \geq 1 \qquad \forall\, e \in E, \forall\, t \in T, \tag{24}$$

$$x^e_{at} \in \{0, 1\} \qquad \forall\, e \in E, \forall\, a \in A, \forall\, t \in T \tag{25}$$

$\longrightarrow$ les journées des employés ne respectent pas les contraintes légales et contractuelles de l'entreprise.

Ex: $-11 - - - 2121 - -2 - 1 - 12222211$

ETP: horaires légaux

Problème de génération d'horaires (satisfaction):

- Un horaire est une affectation des activités aux périodes
  $s : T \rightarrow A \cup \{\text{pause}, \text{repas}, \text{repos}, ...\}$, $s(t) =$l'activité effectuée au temps t
  Ex: $- - -11111p2222rr222222 - --$
- Déterminer les horaires possibles pour un employé e satisfaisant les contraintes légales :
  - e ne peut travailler moins de 3h et plus de 8h par jour
  - Un repas de 1h est obligatoire si e travaille plus de 5h
  - Les repas ne sont permis qu'entre 12h et 14h
  - e ne peut effectué une même tâche moins de 2h et plus de 4h d'affilée
  - ...
- nombreuses contraintes, variées, amenées à changer souvent,...

$\longrightarrow$ programmation dynamique ou programmation par contraintes

ETP: décomposition

- résoudre le problème de génération d'horaires pour chaque employé: S^e l'ensemble des journées possibles pour e
- résoudre une seconde formulation linéaire du problème de couverture: Variables binaires: $x_s^e = 1$ si l'employé e est affecté à la journée $s \in S^e$, $x_s^e = 0$ sinon.

$$\min \quad \sum_{e \in E} \sum_{s \in S^e} c^s x_s^e \tag{26}$$

$$s.t. \quad \sum_{e \in E} \sum_{s \in S^e} \delta_{at}^s x_s^e \geq r_{at} \qquad \forall\, a \in A, \forall\, t \in T, \tag{27}$$

$$\sum_{s \in S^e} x_s^e = 1 \qquad \forall\, e \in E, \tag{28}$$

$$x_s^e \in \{0, 1\} \qquad \forall\, e \in E, \forall\, s \in S^e. \tag{29}$$

avec $\delta_{at}^s = 1$ si $s(t) = a$ et $\delta_{at}^s = 0$ sinon.

ETP: décomposition

- résoudre le problème de génération d'horaires pour chaque employé: S^e l'ensemble des journées possibles pour e (PPC)
- résoudre une seconde formulation linéaire du problème de couverture: Variables binaires: $x_s^e = 1$ si l'employé e est affecté à la journée $s \in S^e$, $x_s^e = 0$ sinon.

$$\begin{aligned}
\min \quad & \sum_{e \in E} \sum_{s \in S^e} c^s x_s^e & & (30)\\
s.t. \quad & \sum_{e \in E} \sum_{s \in S^e} \delta_{at}^s x_s^e \geq r_{at} & \forall\, a \in A, \forall\, t \in T, \quad & (31)\\
& \sum_{s \in S^e} x_s^e = 1 & \forall\, e \in E, \quad & (32)\\
& x_s^e \in \{0, 1\} & \forall\, e \in E, \forall\, s \in S^e. \quad & (33)
\end{aligned}$$

avec $\delta_{at}^s = 1$ si $s(t) = a$ et $\delta_{at}^s = 0$ sinon. (PL)

- décomposition du problème

ETP: génération de colonnes hybride

programme maître (PL): $\mathcal{S}' \subset \mathcal{S}$

$$\begin{aligned} z' = \min \sum_{s \in \mathcal{S}'} c^s x_s & \\ s.t. \sum_{s \in \mathcal{S}'} \delta^s_{at} x_s \geq r_{at} & \qquad \forall\, a \in A, \forall\, t \in [1..T], \\ x_s \geq 0 & \qquad \forall\, s \in \mathcal{S}'. \end{aligned}$$

valeurs duales λ_{at} $\downarrow$ $\qquad$ $\uparrow$ colonnes ajoutés à $\mathcal{S}'$

sous-problème (PPC):

Find $s \in \mathcal{S}$ s.t. $\sum_{t=1}^{T} (c_{s(t)t} - \lambda_{s(t)t}) < 0$ $\quad \begin{cases} \text{add to } \mathcal{S}' & \text{if exist,} \\ \text{STOP} & \text{otherwise.} \end{cases}$