4TC-IAT, Optimisation combinatoire

Tristan Roussillon

INSA Lyon, TC

2023

Problème d'optimisation combinatoire

$$(\mathrm{P}) \left\{ \begin{array}{c} \min \ f(s) \text{ tel que :} \\ s \in S \text{ et } A(s) \end{array} \right.$$

- S est un ensemble discret fini de solutions,
- $f : S \mapsto \mathbb{R}$ est la fonction objectif,
- $A : S \mapsto \{\text{vrai}, \text{faux}\}$ est le critère d'acceptation d'une solution.

Ensemble des solutions acceptables

L'ensemble des solutions acceptables n'est généralement pas décrit par une liste exhaustive, car

- il peut être difficile ou long de décider si une solution est acceptable ou non,
- et la liste pourrait être si longue qu'elle soit impossible à compléter en un temps raisonnable.

Elle est plus souvent décrite implicitement par des propriétés.

De nombreux problèmes se ramènent à un problème d'optimisation combinatoire

- problème d'affectation (assignment problem)
- problème de coloration (graph coloring)
- problème des surveillants de musée (art gallery problem)
- problème du sac à dos (knapsack problem)
- problème du voyageur de commerce (travelling salesman problem)
- problème de séquençage de tâches (job sequencing)
- ...

Ex. 1 : problème d'affectation

- 4 individus $\{1, 2, 3, 4\}$ et 4 tâches $\{1, 2, 3, 4\}$
- 1 matrice de coût $C := \begin{pmatrix} 8 & 3 & 1 & 5 \\ 11 & 7 & 1 & 6 \\ 7 & 8 & 6 & 8 \\ 11 & 6 & 4 & 9 \end{pmatrix}$
  par ex. affecter l'individu 4 à la tâche 3 coûte $c_{4,3} = 4$
- Trouver l'affectation de coût minimal
  par ex. l'identité a un coût de $8 + 7 + 6 + 9 = 30$.

Ex. 1 : problème d'affectation

$$(Aff) \begin{cases} \min \ f(s) \text{ tel que :} \\ \qquad s \in S \end{cases}$$

- S est l'ensemble des permutations de $(1, 2, 3, 4)$.
- $f : S \mapsto \mathbb{R}$ est définie telle que, $\forall s \in S$, $f(s) = \sum_{i=1}^{4} c_{k,t_k}$, où $s = (t_1, t_2, t_3, t_4)$.

C'est un problème facile car

- toutes les solutions sont acceptables,
- il n'y a que $4! = 24$ solutions possibles.

Ex. 1 : problème d'affectation

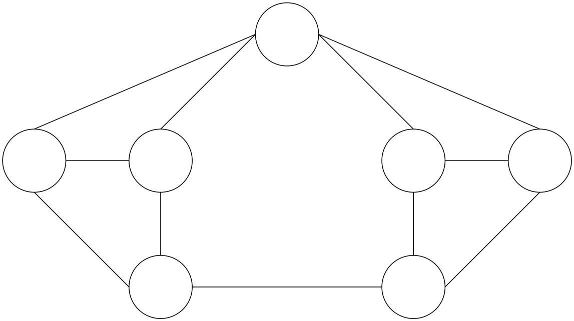
S'il y a n individus et n tâches, il y a $n!$ solutions possibles.

Par ex. si $n = 23$, $n! \approx 5 \cdot 10^{22}$, alors qu'une année ne compte qu'environ $3,16 \cdot 10^{13}$ microsecondes.

Heureusement, il existe des algorithmes plus efficaces que la recherche exhaustive :

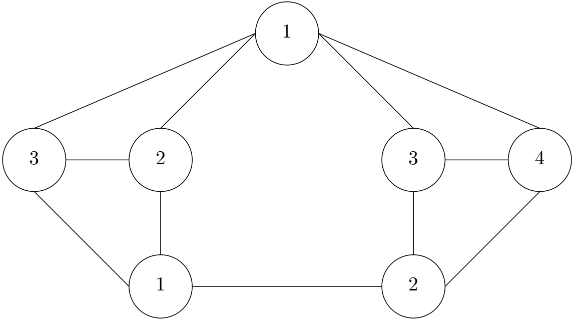
- algorithme hongrois en $O(n^3)$,
- algorithme des enchères,
- push-relabel, preflow-push, ...

Ex. 2 : coloration de graphe



- Soit le graphe ci-dessus (graphe de Moser).
- Trouver sa coloration avec le plus petit nombre de couleurs.

Ex. 2 : coloration de graphe



- Soit le graphe ci-dessus (graphe de Moser).
- Trouver sa coloration avec le plus petit nombre de couleurs.

Ex. 2 : coloration de graphe

$$(Col) \begin{cases} \min\ f(s) \text{ tel que :} \\ s \in S \text{ et } A(s) \end{cases}$$

- S est l'ensemble des associations noeuds/couleurs,
- $\forall s \in S$, $A(s)$ si pour toute arête, les couleurs, selon s, des noeuds reliés par l'arête sont différentes,
- $f : S \mapsto \mathbb{R}$ est définie telle que, $\forall s \in S$, $f(s)$ est le nombre de couleurs présentes dans s.

Une *coloration* s est une association noeuds/couleurs telle que $A(s)$.

Il est très difficile d'énumérer l'ensemble des colorations.

Complexité des problèmes de coloration de graphe

Il y a trois problèmes associés :

- décision : existe-t-il une coloration en k couleurs ?
- recherche : exhiber une telle coloration.
- optimisation : trouver la coloration avec le plus petit k.

Pour $k > 2$,

- il n'y a pas d'algorithmes connus pour résoudre ces problèmes en temps polynomial.
- Le problème de décision est dans la classe NP (NP-complet).

Description des solutions

$$(\mathrm{P}) \left\{ \begin{array}{c} \min \ f(s) \text{ tel que :} \\ s \in S \text{ et } A(s) \end{array} \right.$$

S est généralement décrit par *l'affectation* de *valeurs*, parmi un ensemble de valeurs possibles, à chacune des *variables* du problème. $A(S)$ ajoute des *contraintes*.

Ex.

- (Aff) : variables = individus, valeurs = tâches
- (Col) : variables = noeuds, valeurs = couleurs

Sommaire

Principe de l'algorithme glouton

Réaliser un choix localement optimal à chaque étape en espérant atteindre l'optimal global à la fin.

La plupart du temps, on n'atteint pas l'optimum global, mais pour certains problèmes si, et pour d'autres, on peut garantir une certaine proximité.

L'algorithme est souvent facile à implémenter et s'exécute souvent rapidement.

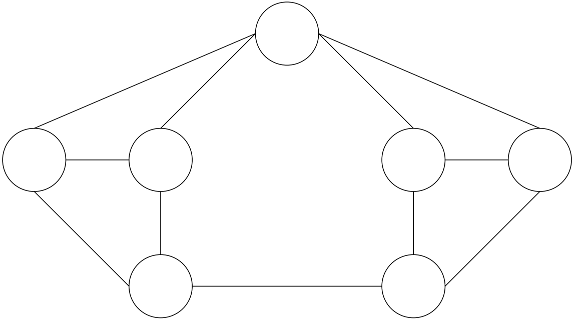
Résolution gloutonne du problème d'affectation

$$C := \begin{pmatrix} 8 & 3 & 1 & 5 \\ 11 & 7 & 1 & 6 \\ 7 & 8 & 6 & 8 \\ 11 & 6 & 4 & 9 \end{pmatrix}$$

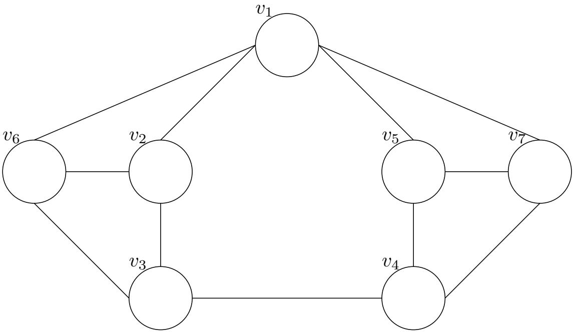
- on traite les individus dans un certain ordre ;
- pour chaque individu, on lui affecte la tâche encore disponible de moindre coût.

- $c_{1,3} = 1$, $c_{2,4} = 6$, $c_{3,1} = 7$, $c_{4,2} = 6$.
- coût total : $1 + 6 + 7 + 6 = 20$. Optimal ?

Résolution gloutonne de la coloration

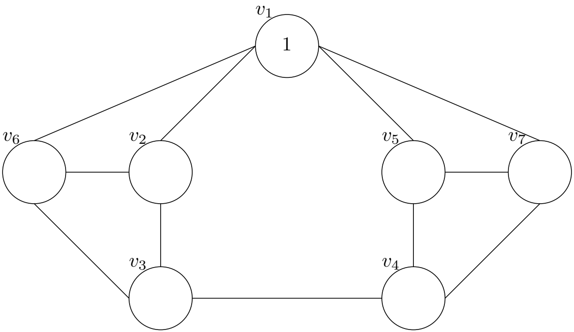


Résolution gloutonne de la coloration



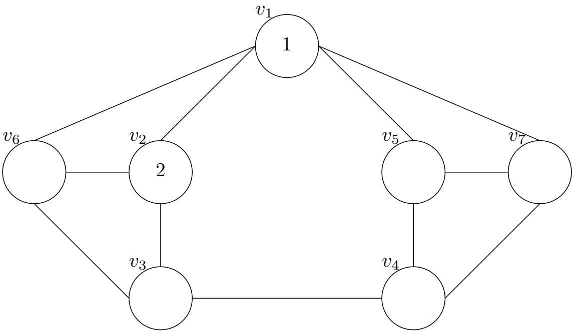
- on traite les noeuds dans un certain ordre ;

Résolution gloutonne de la coloration



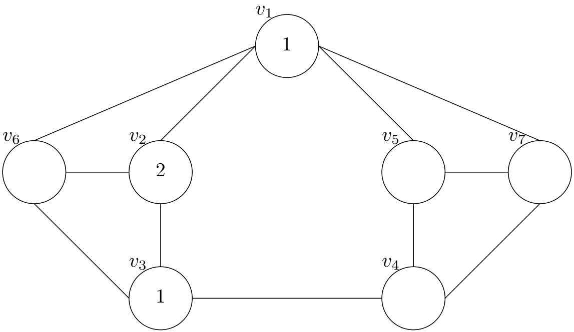
- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

Résolution gloutonne de la coloration



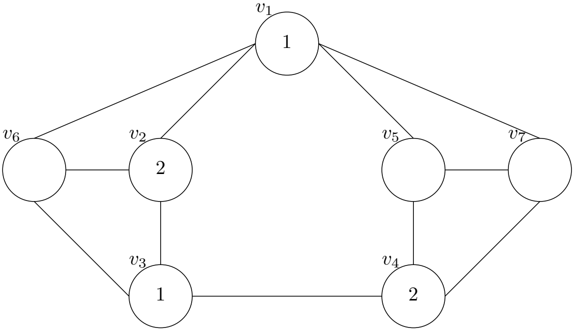
- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

Résolution gloutonne de la coloration



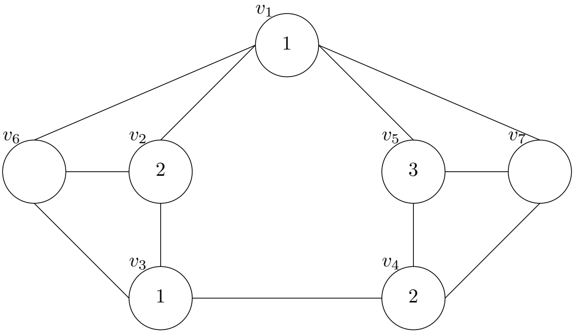
- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

Résolution gloutonne de la coloration



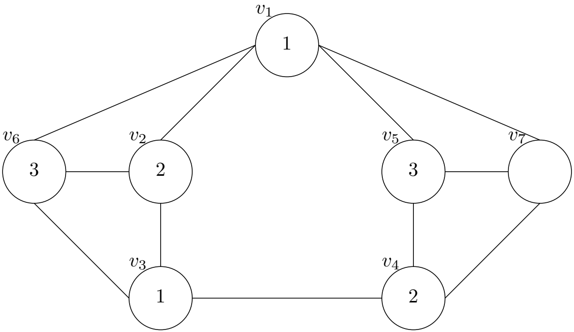
- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

Résolution gloutonne de la coloration



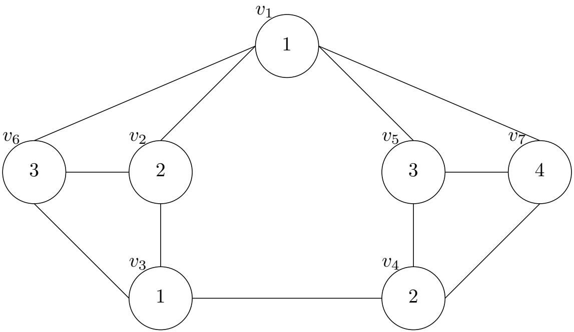
- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

Résolution gloutonne de la coloration



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

Résolution gloutonne de la coloration



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on lui affecte la couleur possible de plus petit indice.

- 4 couleurs au total. Optimal ?

Ordres possibles

- random : noeuds triés aléatoirement.
- largest first : noeuds triés par ordre de degré non croissant.
- smallest last : L'ordre $v_1, v_2, \ldots, v_n$ est tel que v_i a le plus petit degré dans le graphe ne contenant plus que les sommets $v_1, v_2, \ldots, v_i$.
- DSATUR : l'ordre est construit en choisissant à chaque étape le noeud v qui maximise la saturation (nombre de couleurs différentes déjà affectées aux voisins).
- ...

Algorithme, complexité linéaire

Backtracking

Méthode constructive pour résoudre un problème de décision :

- on traite les variables du problème dans un certain ordre ;
- pour une l'affectation possible d'une valeur à une variable, on teste récursivement si une solution valide peut être construite à partir de cette affectation. Si ce n'est pas possible, on abandonne et on revient sur les affectations qui auraient été faites pour des variables précédentes (backtracking).
- on répond non si on a tout tenté sans succès, oui si on a trouvé une solution valide.

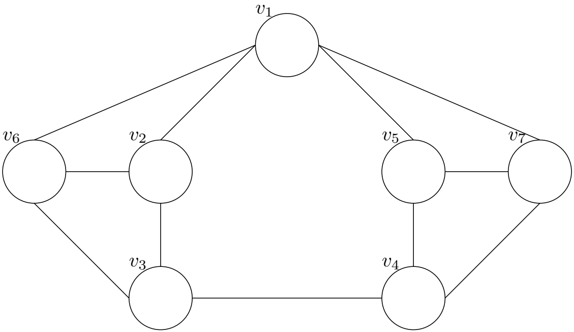
Avertissement

On ne résoud pas un problème d'optimisation, mais le problème de *décision* ou de *recherche* associé.

Dans le cas d'une fonction objectif discrète, le backtracking peut être utilisé pour voir si on peut faire mieux qu'une première approximation.

Par ex. sur le problème de coloration précédent on a trouvé une solution à 4 couleurs avec l'algorithme glouton. Existe-t-il une coloration à 3 couleurs ?

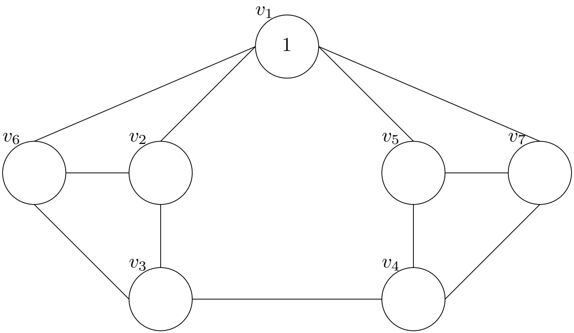
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte OK !

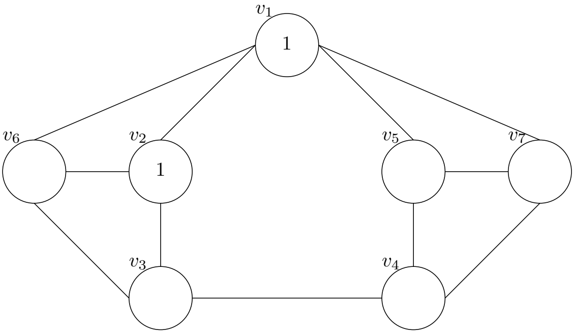
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte OK !

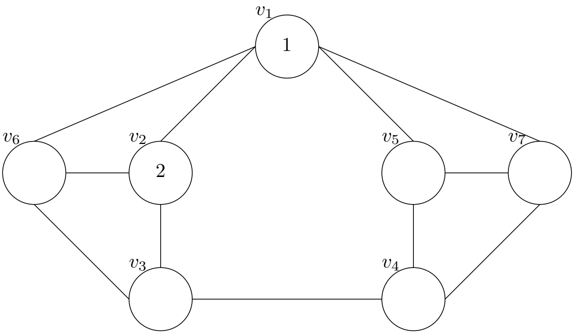
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte KO !

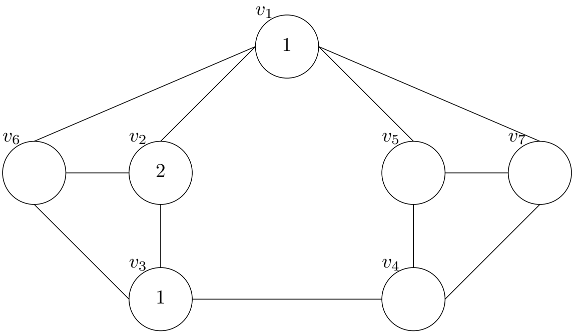
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte OK !

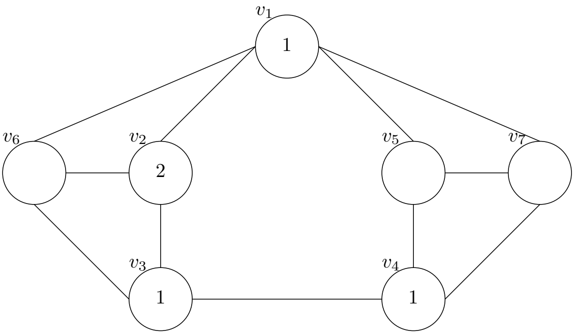
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte OK !

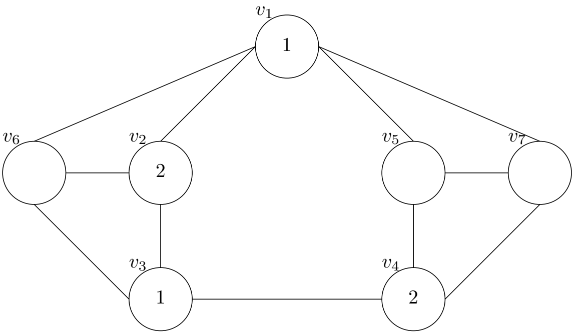
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte KO !

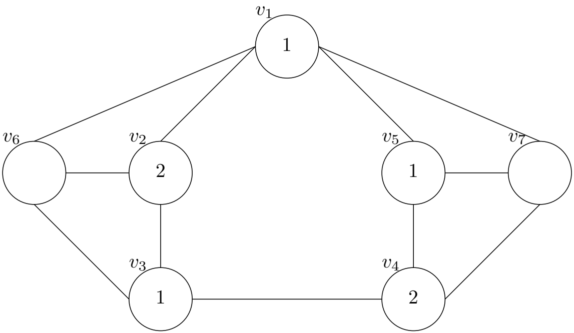
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte OK !

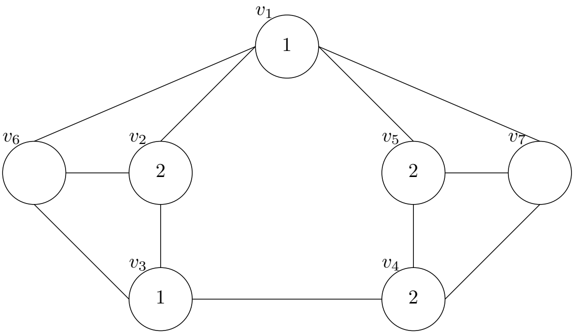
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte KO !

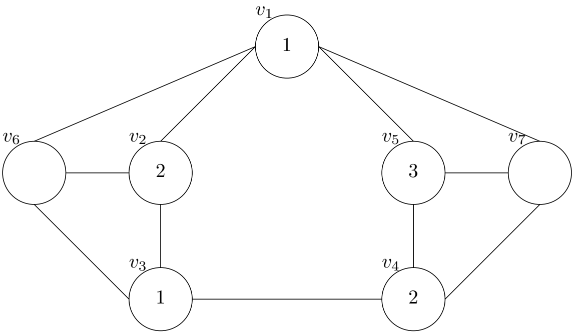
Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte KO !

Coloration avec 3 couleurs ?



- on traite les noeuds dans un certain ordre ;
- pour chaque noeud, on teste les couleurs 1, 2 puis 3 ;
- on revient en arrière quand les contraintes sont violées.

contrainte OK ! ensuite ?

Algorithme, complexité exponentielle

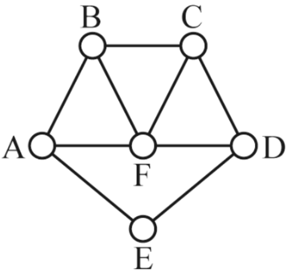
TODO

Intermède

Alain Hertz, *L'agrapheur, Intrigues policières à saveur mathématique*, Presses internationales Polytechnique, 2010, 258 pp.

Intermède

					5	7		6
	3		1		A	B	9	F
							5	C
	1	9	5		E		4	D
					3			7
					6			
					7			1
								3



Les cases A, B, C, F ne peuvent recevoir que 2, 4, ou 8 ;
Les cases E et D, que 2 ou 8.
Peut-on affecter un seul chiffre aux cases A et C ?

Programmation dynamique

Il s'agit de décomposer le problème en sous-problèmes, puis résoudre les sous-problèmes, des plus petits aux plus grands en stockant les résultats intermédiaires.

Cette approche est théorisée et nommée “programmation dynamique” (pour faire bien) par Richard Bellman dans les années cinquantes. Elle repose sur la simple observation qu'un chemin optimal est formé de sous-chemins optimaux.

Retour sur le problème d'affectation

Il y a 4 individus et tâches. Pour $k \leq 4$,

$$(Aff_k) \begin{cases} \min \ f_k(s) \text{ tel que :} \\ \qquad s \in S_k \end{cases}$$

- S_k est l'ensemble des arrangements de k tâches parmi 4.
- $f_k : S_k \mapsto \mathbb{R}$ est le coût de l'affectation de k tâches aux k premiers individus.

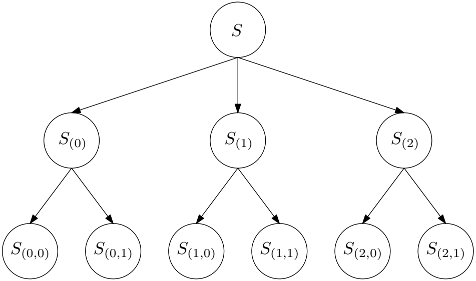
Formule de récurrence

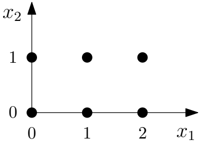
$$\min_{s \in S_k} f_k(s) = c_k + \min_{s' \in S_{k-1}} f_{k-1}(s'),$$

où on note c_k le coût minimal d'affectation à l'individu k de l'une des tâches non présentes dans la solution s.

Représentation arborescente et principe de séparation

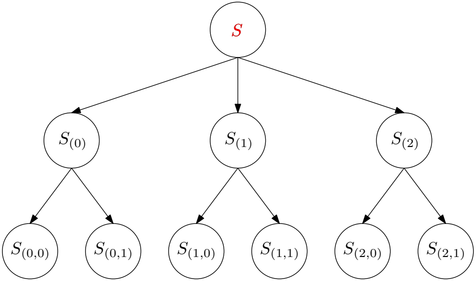
- $S \supseteq X$ est un ensemble de vecteurs possibles
- On partitionne S en sous-ensembles
  $S_{(v)} := \{(x_1, \ldots, x_n) \in S \mid x_1 = v\}$
  selon les valeurs prises par x_1
- Pareil selon x_2 et ainsi de suite.

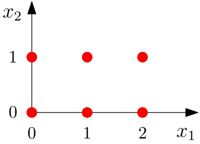




Représentation arborescente et principe de séparation

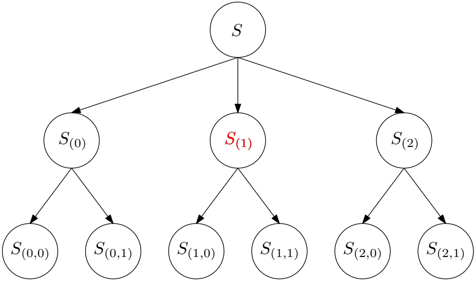
- $S \supseteq X$ est un ensemble de vecteurs possibles
- On partitionne S en sous-ensembles
  $S_{(v)} := \{(x_1, \ldots, x_n) \in S \mid x_1 = v\}$
  selon les valeurs prises par x_1
- Pareil selon x_2 et ainsi de suite.

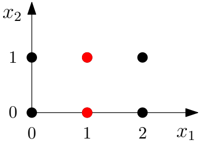




Représentation arborescente et principe de séparation

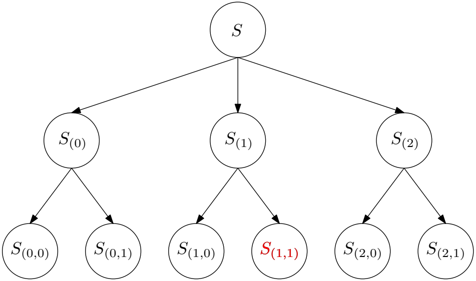
- $S \supseteq X$ est un ensemble de vecteurs possibles
- On partitionne S en sous-ensembles
  $S_{(v)} := \{(x_1, \ldots, x_n) \in S \mid x_1 = v\}$
  selon les valeurs prises par x_1
- Pareil selon x_2 et ainsi de suite.

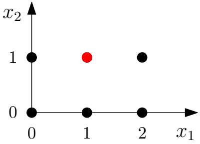




Représentation arborescente et principe de séparation

- $S \supseteq X$ est un ensemble de vecteurs possibles
- On partitionne S en sous-ensembles
  $S_{(v)} := \{(x_1, \ldots, x_n) \in S \mid x_1 = v\}$
  selon les valeurs prises par x_1
- Pareil selon x_2 et ainsi de suite.





Principe d'évaluation

- fonction d'évaluation e : pour tout ensemble T descendant de S, on choisit e telle que $e(T) \leq \min_{x \in T} f(x)$
- si on connait une solution $\bar{x} \in S$ et qu'au noeud T on a

$$f(\bar{x}) < e(T) \leq \min_{x \in T} f(x)$$

- alors T ne contient aucune solution optimale et ses descendants ne seront pas énumérés

Séparation-évaluation

On résoud (exactement) de nombreux problèmes en combinant ces deux principes (*Branch and Bound*).
Il y a d'innombrables variantes selon :

- la fonction d'évaluation
- le choix du noeud à explorer (et donc de la valeur à fixer)
- le choix de la variable à partir de laquelle on partitionne le noeud choisi

Exemple d'un problème d'affectation

- 4 individus $\{1, 2, 3, 4\}$ et 4 tâches $\{1, 2, 3, 4\}$
- 1 matrice de coût $C := \begin{pmatrix} 8 & 3 & 1 & 5 \\ 11 & 7 & 1 & 6 \\ 7 & 8 & 6 & 8 \\ 11 & 6 & 4 & 9 \end{pmatrix}$
  par ex. affecter l'individu 4 à la tâche 3 coûte $c_{4,3} = 4$
- Trouver l'affectation de coût minimal

Exemple d'un problème d'affectation

- On peut noter les inconnues et contraintes $x^T = (x_1, x_2, x_3, x_4)$, telles que $\forall i \in \{1, 2, 3, 4\},\ x_i \in \{1, 2, 3, 4\}$ (x_i contient le numéro de tâche affectée à l'individu i).
- Pour exprimer la fonction objectif, on peut écrire pour tout i $x_i := \sum_{j=1}^{4} j d_{i,j} = d_{i,1} + 2d_{i,2} + 3d_{i,3} + 4d_{i,4}$ ($d_{i,j} = 1$ si la tâche j est affectée à l'individu i, $d_{i,j} = 0$ sinon).
- Le problème s'exprime donc comme un (PNE)

$$\begin{cases} \min \ \sum_{i,j} c_{i,j} d_{i,j} \text{ tel que :} \\ \forall i \in \{1, 2, 3, 4\},\ x_i \in \{1, 2, 3, 4\} \text{ et } x_i = \sum_{j=1}^{4} j d_{i,j} \end{cases}$$

Résolution par séparation-évaluation

- ensemble des affectations possibles : S ($|S| = 4! = 24$)
- fonction d'évaluation e : on somme les coûts minimaux des tâches non affectées, aux coûts des tâches déjà affectées
- choix de l'individu selon lequel on sépare : arbitraire
- choix de la tâche à fixer en priorité : celle qui minimise e

Déroulement de la méthode

$$\begin{pmatrix} 8 & 3 & 1 & 5 \\ 11 & 7 & 1 & 6 \\ 7 & 8 & 6 & 8 \\ 11 & 6 & 4 & 9 \end{pmatrix}$$

$$e(S) = (7 + 3 + 1 + 5) = 16$$

- $\Rightarrow$ 16 minorant du coût minimum
- choix de l'individu : 1
- choix de la tâche à affecter…

Déroulement de la méthode

$$\text{individu } 1 \leftarrow \text{tâche } 1$$

$$\begin{pmatrix} \cancel{8} & \cancel{3} & \cancel{1} & \cancel{5} \\ \cancel{11} & 7 & 1 & 6 \\ \cancel{7} & 8 & 6 & 8 \\ \cancel{11} & 6 & 4 & 9 \end{pmatrix}$$

- $e(S_{(1)}) = 8 + (6 + 1 + 6) = 21$

Déroulement de la méthode

individu 1 $\leftarrow$ tâche 2

$$\begin{pmatrix} \cancel{8} & \cancel{3} & \cancel{1} & \cancel{5} \\ 11 & \cancel{7} & {\color{red}1} & {\color{red}6} \\ {\color{red}7} & \cancel{8} & 6 & 8 \\ 11 & \cancel{6} & 4 & 9 \end{pmatrix}$$

- $e(S_{(1)}) = 8 + (6 + 1 + 6) = 21$
- $e(S_{(2)}) = 3 + (7 + 1 + 6) = 17$

Déroulement de la méthode

individu 1 $\leftarrow$ tâche 3

$$\begin{pmatrix} \cancel{8} & \cancel{3} & \cancel{1} & \cancel{5} \\ 11 & 7 & \cancel{1} & 6 \\ 7 & 8 & \cancel{6} & 8 \\ 11 & 6 & \cancel{4} & 9 \end{pmatrix}$$

- $e(S_{(1)}) = 8 + (6 + 1 + 6) = 21$
- $e(S_{(2)}) = 3 + (7 + 1 + 6) = 17$
- $e(S_{(3)}) = 1 + (7 + 6 + 6) = 20$

Déroulement de la méthode

individu 1 $\leftarrow$ tâche 4

$$\begin{pmatrix} \cancel{8} & \cancel{3} & \cancel{1} & \cancel{5} \\ 11 & 7 & \color{red}{1} & \cancel{6} \\ \color{red}{7} & 8 & 6 & \cancel{8} \\ 11 & \color{red}{6} & 4 & \cancel{9} \end{pmatrix}$$

- $e(S_{(1)}) = 8 + (6 + 1 + 6) = 21$
- $e(S_{(2)}) = 3 + (7 + 1 + 6) = 17$
- $e(S_{(3)}) = 1 + (7 + 6 + 6) = 20$
- $e(S_{(4)}) = 5 + (7 + 6 + 1) = 19$

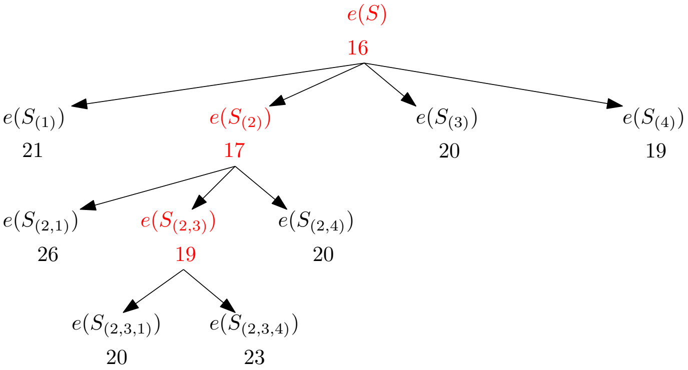
Déroulement de la méthode

individu 1 $\leftarrow$ tâche 2

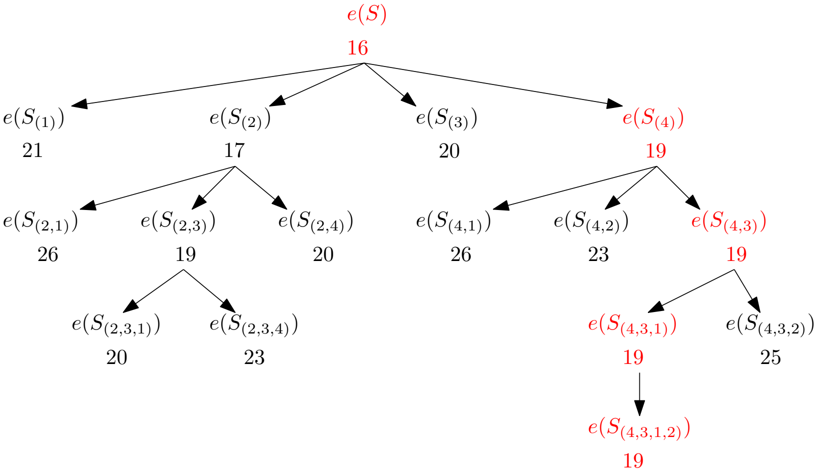
$$\begin{pmatrix} \cancel{8} & \cancel{3} & \cancel{1} & \cancel{5} \\ 11 & \cancel{7} & 1 & 6 \\ 7 & \cancel{8} & 6 & 8 \\ 11 & \cancel{6} & 4 & 9 \end{pmatrix}$$

- $e(S_{(1)}) = 8 + (6 + 1 + 6) = 21$
- $e(S_{(2)}) = 3 + (7 + 1 + 6) = 17$
- $e(S_{(3)}) = 1 + (7 + 6 + 6) = 20$
- $e(S_{(4)}) = 5 + (7 + 6 + 1) = 19$

Trace sous forme arborescente



Trace sous forme arborescente



Résultat

$$e(S_{(4,3,1,2)}) = 5 + 1 + 7 + 6 = 19$$

$$\begin{pmatrix} 8 & 3 & 1 & \color{red}{5} \\ 11 & 7 & \color{red}{1} & 6 \\ \color{red}{7} & 8 & 6 & 8 \\ 11 & \color{red}{6} & 4 & 9 \end{pmatrix}$$

Minimum global

- individu 1 $\leftarrow$ tâche 4
- individu 2 $\leftarrow$ tâche 3
- individu 3 $\leftarrow$ tâche 1
- individu 4 $\leftarrow$ tâche 2

Méthodes de résolution

- recherche exhaustive (très-très petits problèmes)
- *Branch and Bound* (séparation-évaluation)
- relaxation continue dans certains cas particuliers
- méthode des coupes (intégrales, mixtes)
- *Branch and Bound* + coupes = *Branch and cut*
- programmation par contraintes
- méta-heuristiques :
  glouton, meilleur voisin, tabou, recuit-simulé, algorithme génétique, colonie de fourmis, ...