Affectation et transport

optimisation linéaire

Optimisation linéaire en variables réelles

Rappel (cours ING1) :

Un problème d'optimisation linéaire en variables réelles consiste à optimiser (minimiser ou maximiser) un objectif **linéaire** sur *n* **variables réelles** sous un ensemble de *m* contraintes **linéaires**.

opt $c_1x_1+c_2x_2+\ldots+c_nx_n$ avec $(x_1,x_2,\ldots,x_n) \in R^n$

s.c. $a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n \leq b_1$
$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n \leq b_2$
…
$a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n \leq b_m$

Optimisation linéaire

Rappel (cours ING1) :

On **sait résoudre** les problèmes d'optimisation linéaires en **variables réelles** à la fois en pratique (l'algorithme du simplexe est très efficace sur la quasi-totalité des instances) et en théorie (les algorithmes de point intérieur garantissent un temps d'exécution polynomial pour toutes les instances).

Par contre, on **ne sait pas résoudre** les problèmes d'optimisation linéaire en **variables entières** par un algorithme à la fois **général**, **exact** et **polynomial**.

Affectation (*assignment*) : modélisation

On dispose d'un ensemble de *n* employés et d'un autre de *n* tâches. Pour chaque couple (employé,tâche), on connaît le coût d'affectation de cet employé à cette tâche. Comment minimiser le coût total d'affectation sachant que ni les employés ni les tâches ne sont fractionnables (*i.e.* chaque employé doit être affecté à exactement une tâche) ?

Démarche

Approche exhaustive :

Chaque affectation étant une permutation, il y en a $n!$. En fixant un seuil à 10^9 configurations, on ne peut pas résoudre le problème dès que n est plus grand que 13.

Approche gloutonne :

Quel que soit l'algorithme glouton proposé, le dernier couple (employé,tâche) peut avoir un prix arbitrairement grand, donc on ne peut pas résoudre le problème ainsi.

Affectation : méthode hongroise (ou de Kuhn-Munkres)

1. réduction de la matrice

   soustraire à chaque ligne le plus petit élément de chaque ligne
   soustraire à chaque colonne le plus petit élément de chaque colonne

2. recherche d'un couplage parfait (*perfect matching*)

   construire un graphe orienté dont les sommets sont les employés et les tâches.
   ajouter une source et un puits.
   relier la source à chacun des employés et chacune des tâches au puits.
   pour chaque zéro de la matrice, créer un arc de l'employé vers la tâche.
   donner à chaque arc un poids de 1.

   déterminer un flot maximal dans ce graphe (par exemple par la l'algorithme de Ford-Fulkerson)

Affectation : exemple

1. réduction de la matrice

17	15	9	5	12	-5
16	16	10	5	10	-5
12	15	14	11	5	-5
4	8	14	17	13	-4
13	9	8	12	17	-8

12	10	4	0	7
11	11	5	0	5
7	10	9	6	0
0	4	11	13	9
5	1	0	4	9
-0	-1	-0	-0	-0

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	11	13	9
5	0	0	4	9

Affectation : exemple

2.1- On marque les lignes sans zéro sélectionné (sans vert)

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	11	13	9
5	0	0	4	9

x

2.2- On marque toutes les colonnes ayant un zéro non sélectionné (rouge) se trouvant sur une ligne marquée

x

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	11	13	9
5	0	0	4	9

x

2.3- On marque toutes les lignes ayant un zéro sélectionné dans une colonne marquée.

On répète 2.2 et 2.3 jusqu'à ce que cela ne soit plus possible

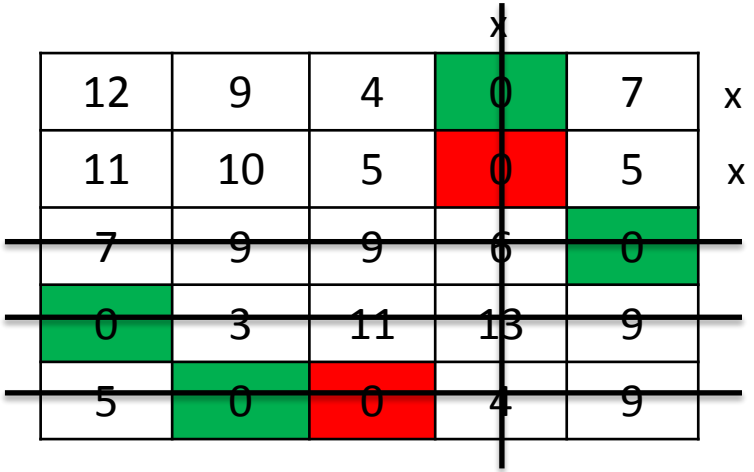
x

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	11	13	9
5	0	0	4	9

x
x

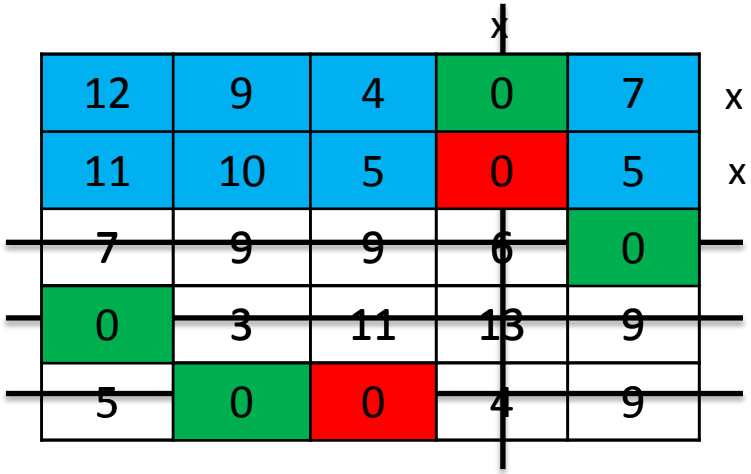
Affectation : exemple (suite)

3.1- On barre toutes les lignes non marquées,
et toutes les colonnes marquées.



Affectation : exemple (suite)

3.2- La zone bleue représente une sous-matrice. On sélectionne le minimum de la sous-matrice, ici 4.



Affectation : exemple (suite)

3.3- On enlève 4 à toute la sous-matrice.
On ajoute 4 aux croisements des lignes et colonnes rayées.

			x	
8	5	0	0	3	x
7	6	1	0	1	x
7	9	9	10	0	
0	3	11	17	9	
5	0	0	8	9	

On réitère les étapes 1, 2 et 3.

Affectation : exemple (suite)

8	5	0	0	3
7	6	1	0	1
7	9	9	10	0
0	3	11	17	9
5	0	0	8	9

On regarde où sont les zéros sélectionnés dans le tableau initial.

On somme pour avoir le coût de minimisation.

Minimisation
= 9 + 5 + 5 + 4 + 9
= 32

17	15	9	5	12
16	16	10	5	10
12	15	14	11	5
4	8	14	17	13
13	9	8	12	17

Historique

Interview d'Harold Kuhn

A tale of 3 eras: the discovery and rediscovery of the hungarian method

- 1950s : Harold Kuhn (1925-2014) et James Munkres (1930-)
- 1930s : Denes König (1884-1944) et Jeno Egervary (1891-1958)
- : Jacobi (1804-1851)

Transport (*transportation*) : description

Définition (*Précis de recherche opérationnelle* Faure, Lemaire et Picouleau)

Un problème de transport peut être modélisé par un graphe valué biparti entre *m* origines (offres) et *n* destinations (demandes). Dans ce graphe, on peut tracer *nm* arcs qui symbolisent les liaisons que l'on peut employer pour transporter des marchandises de chaque origine vers chaque destination. Chaque arc est valué par le coût **unitaire** du transport sur la liaison qu'il représente.

L'objectif du transport est d'acheminer l'ensemble des marchandises depuis les origines vers les destinations pour un coût minimal. On se restreindra (sans perte de généralité) au cas où l'offre est égale à la demande.

Démarche

Ce problème étant une généralisation du précédent il n'a pas non plus de solution avec un algorithme naïf.

On suit pour le résoudre une démarche "à la simplexe" avec :
- **une phase d'initialisation** qui peut être effectuée par (au moins) deux algorithmes :
    - coin nord-ouest : très rapide mais ne tient pas compte des coûts.
    - Balas-Hammer : plus complexe mais donne un candidat de 'bonne' qualité.
- **une phase d'optimisation** proprement dite :
    algorithme de stepping stone (suite gloutonnes d'améliorations)

Historique

Gaspard Monge : théorie des remblais et déblais

Cédric Villani : Optimal transport old and new

Transport : modélisation

Soient o_i les quantités disponibles, d_i les quantités demandées et c_{ij} les coûts de transport unitaires. La solution du problème revient à trouver les valeurs numériques des *mn* nombres positifs ou nuls x_{ij} qui représentent la quantité livrée depuis l'origine *i* à la destination *j*, tels que :

fonction objectif à minimiser : $z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$

contrainte globale : $\sum_{i=1}^{m} o_i = \sum_{j=1}^{n} d_j$
contrainte sur les offres : $\sum_{j=1}^{n} x_{ij} = o_i \ (i = 1,2,\ldots,m)$
contrainte sur demandes : $\sum_{i=1}^{m} x_{ij} = d_j \ (j = 1,2,\ldots,n)$

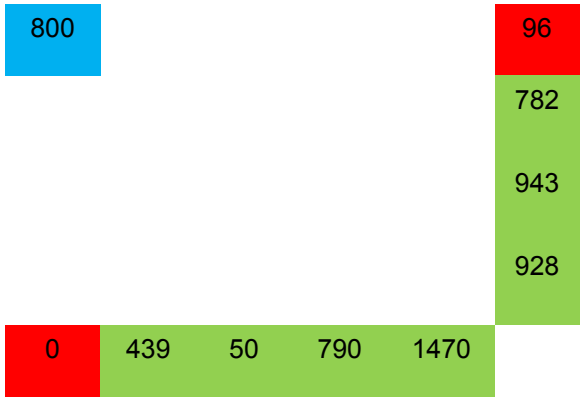
Transport : coin nord-ouest

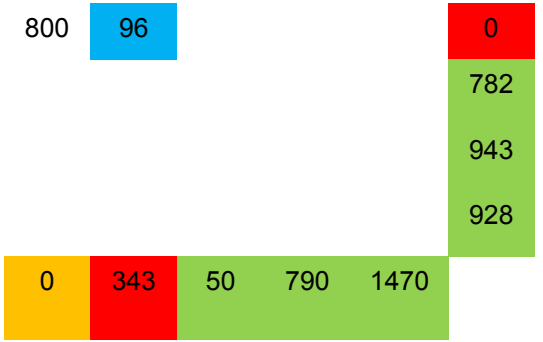
Algorithme :

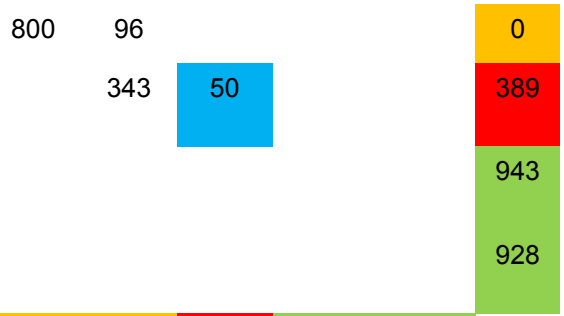
Transporter en premier sur la relation x_{11} (coin nord-ouest de la représentation matricielle du problème), la quantité maximale possible
Transporter le surplus (si il y en a) sur la relation x_{12} puis compléter si nécessaire avec la relation x_{22}
Recommencer jusque avoir sélectionner en tout $m+n-1$ relations si solution non dégénérée

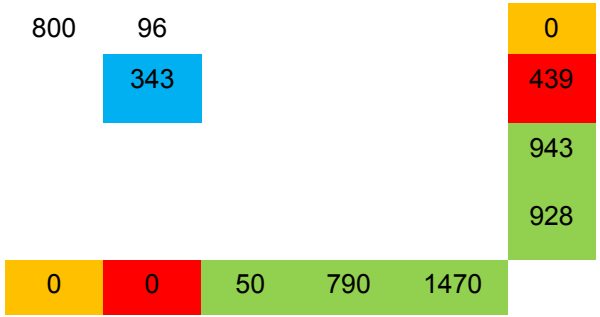
Transport : coin nord-ouest (exemple)

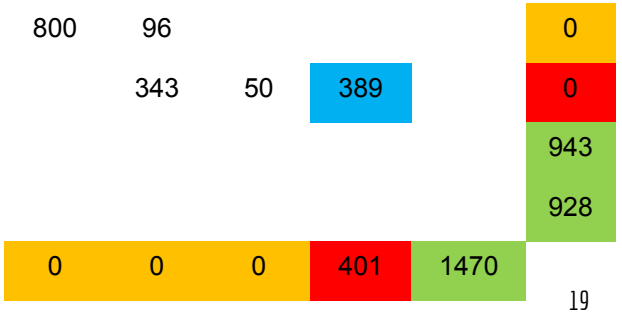
21	11	84	49	13	896
27	52	43	29	42	782
11	47	14	80	93	943
52	94	76	74	54	928
800	439	50	790	1470	



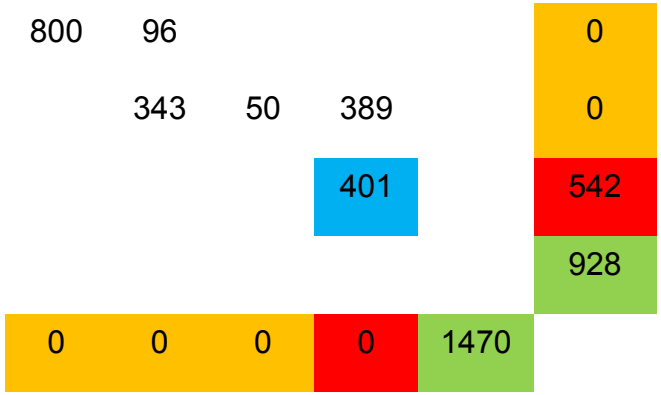


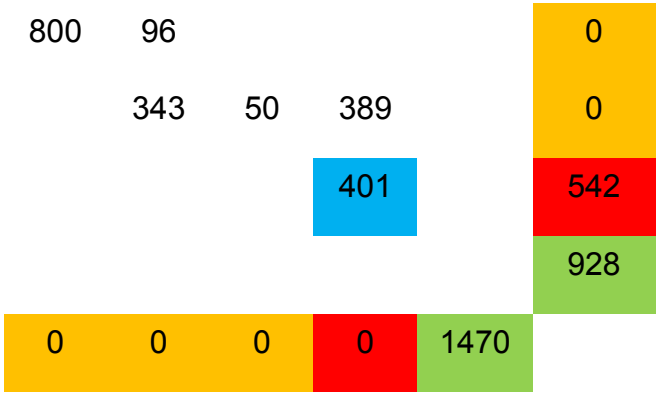






Transport : coin nord-ouest (exemple fin)





Le coût est de 181721:
800x21 + 96x11 + 343x52 + 50x43 + 389x29 + 401x80 + 542x93 + 928x54

Balas Hammer : algorithme

1- Regret : différence des deux minimums (ce qui peut faire 0 en cas d'égalité)
2- Sélectionner le maximum des regrets sur l'ensemble des lignes et des colonnes :
    vider le stock ou remplir la demande sur l'arc le moins cher.
3- Supprimer la ligne ou la colonne dont le stock est vide ou la demande satisfaite
4- Réitérer 1,2 et 3 jusqu'à la fin

Balas Hammer : exemple

21	11	84	49	13	896
27	52	43	29	42	782
11	47	14	80	93	943
52	94	76	74	54	928
800	439	50	790	1470	

					2
					2
					3
					2
10	36	29	20	29	

21	11	84	49	13	457
27	52	43	29	42	782
11	47	14	80	93	943
52	94	76	74	54	928
800	0	50	790	1470	

	439				8
					2
					3
					2
10		29	20	29	

Balas Hammer : exemple suite

21	11	84	49	13	457
27	52	43	29	42	782
11	47	14	80	93	893
52	94	76	74	54	928
800	0	0	790	1470	

	439				8
					2
		50			69
					2
10			20	29	

21	11	84	49	13	457
27	52	43	29	42	782
11	47	14	80	93	93
52	94	76	74	54	928
0	0	0	790	1470	

	439				36
					13
800		50			13
					20
			20	29	

Balas Hammer : exemple suite

21	11	84	49	13	457
27	52	43	29	42	782
11	47	14	80	93	93
52	94	76	74	54	928
0	0	0	790	1470	

	439			457	
					13
800		50			13
					20
			45	12	

21	11	84	49	13	0
27	52	43	29	42	0
11	47	14	80	93	93
52	94	76	74	54	928
0	0	0	8	1013	

	439			457	
			782		
800		50			13
					20
			6	29	

Balas Hammer : exemple fin

21	11	84	49	13	457
27	52	43	29	42	0
11	47	14	80	93	93
52	94	76	74	54	0
0	0	0	8	85	

	439			457	
			782		
800		50			13
				928	
			?	?	

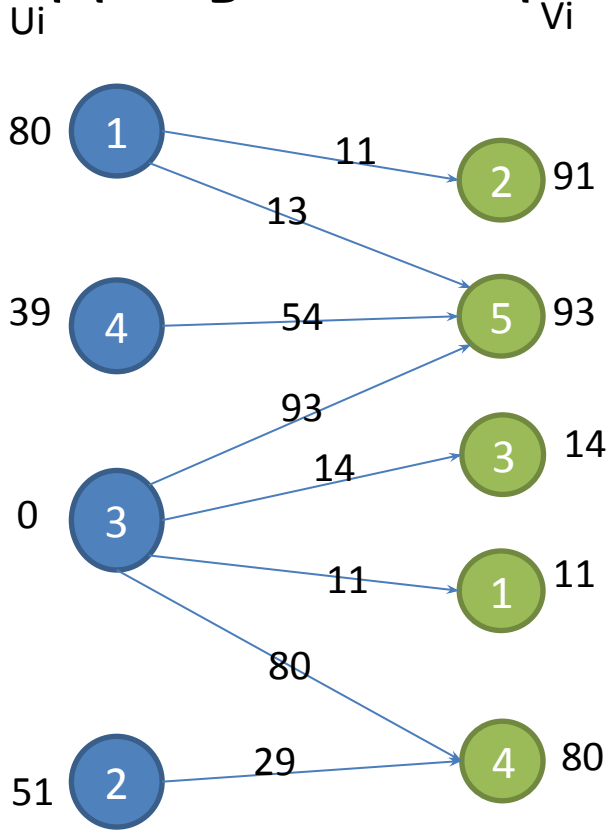
21	11	84	49	13	457
27	52	43	29	42	0
11	47	14	80	93	0
52	94	76	74	54	0
0	0	0	0	0	

	439			457	
			782		
800		50	8	85	
				928	

Le coût est de 101605 :

439x11 + 457x13 + 782x29 + 800x11 + 50x14 + 8x80 + 85x93 + 928x54

Stepping stone : potentiels



21	11	84	49	13	896
27	52	43	29	42	782
11	47	14	80	93	943
52	94	76	74	54	928
800	439	50	790	1470	

	439			457	
			782		
800		50	8	85	
				928	

Pour calculer les potentiels, on essaye d'avoir que du positif pour faciliter les calculs.

Stepping stone : algorithme

On reprend la matrice avec les distributions

On remplit les Ui et les Vi calculés précédemment

						Ui
		439			457	80
				782		51
	800		50	8	85	0
					928	39
Vj	11	91	14	80	93	

On calcule les Delta(ij)

Dij=Ui+Cij-Vj

Ui le potentiel des stocks

Vj le potentiel des demandes

Cij le coût unitaire

On cherche un cycle pour combler le delta négatif.

Ici Rouge -> 85 -> 457 -> 439 -> Rouge

Rouge => 85 (+85)

85 => 0 (-85)

457 => 542 (+85)

439 => 354 (-85)

						Ui
	>0	439	>0	>0	457	80
	>0	>0	>0	782	=0	51
	800	<0	50	8	85	0
	>0	>0	>0	>0	928	39
Vj	11	91	14	80	93	

On refait le graphe des potentiels, le tableau.

On calcule les deltas jusqu'à ce qu'il n'y ait plus de négatif

Math park

Mariages stables, prix équitables et transport de sable

par Filippo Santambrogio

Matrices totalement unimodulaires

En fait, la formulation de ces deux problèmes **en variables réelles** donne systématiquement des solutions **en variables entières** (résultat bien entendu complètement faux dans le cas général). On peut donc les résoudre avec l'algorithme du simplexe.

C'est la théorie des matrices totalement unimodulaires.