Rapport final du projet de Graphes et Recherche opérationnelle

Maxence Ahlouche Maxime Arthaud Korantin Auguste
Martin Carton Thomas Forgione Thomas Wagner

Enseeiht

17 décembre 2013

Introduction

todo ?

Voyageur de commerce – Énoncé

Chercher un chemin passant par tous les sommets, de longueur minimale.

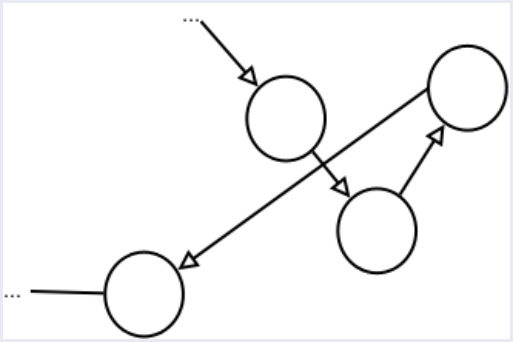
- cycle hamiltonien de coût minimal
- NP-complet
- méthodes approchées

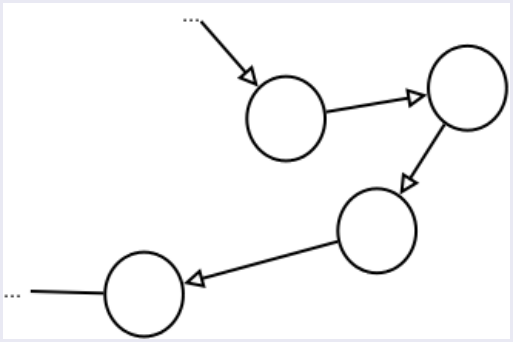
Voyageur de commerce – Résolution approchée

Heuristiques

Aller sur le nœud le plus près

Recherche locale





Voyageur de commerce – Métaheuristiques

- Recherche locale itérée
- Recherche tabou
- Recuit simulé
- Algorithmes génétiques
- Colonies de fourmis

Sac à dos – Énoncé

Remplir un sac pour maximiser la valeur des objets, sans dépasser une certaine masse.

Sac à dos – Résolution

- Résolution exacte :
  - Programmation dynamique.
  - Masses entières uniquement.
  - $O(nW)$
- Résolution dynamique :
  - Algorithme glouton.
  - Masses quelconques.
  - $O(n \log n)$

Sac à dos – Résultats

Nombre d'objets/ Amplitudes des prix et masses/ Masse maximale autorisée	Résultat optimum	Prix le plus élevé	Masse la plus faible	Meilleur ratio prix/masse
50/25/20	85	49/42.4%	67/21.2%	81/4.7%
500/25/500	2016	1125/44.2%	1725/14.4%	1983/1.6%
5000/25/500	5540	1175/79%	4577/17.4%	5540/0%
50000/25/500	11195	1175/90%	6684/40.3%	11195/0%
50000/1000/500	118260	5959/95%	101857/13.9%	118147/0.1%
50000/5000/100	100847	14931/85.2%	93532/7.3%	100282/0.6%

Simplexe – Présentation du problème

$$\max_{\substack{x \in \mathbb{R}^n \\ Ax \leqslant b}} f(x)$$

Exemple

	P_1	P_2	P_3	P_4	stock
R_A	2	4	5	7	72
R_B	1	1	2	2	17
R_C	1	2	3	3	24
bénéfice	7	9	18	17	

Transformation des contraintes d'inégalité en égalité.

Exemple

	P_1	P_2	P_3	P_4	x_1	x_2	x_3	stock
R_A	2	4	5	7	1	0	0	72
R_B	1	1	2	2	0	1	0	17
R_C	1	2	3	3	0	0	1	24
bénéfice	7	9	18	17	0	0	0	

Simplexe – Algorithme

Algorithme

Tant qu'il y a un élément strictement positif sur la première ligne

à_ajouter = indice de la colonne dont le gain est maximal

à_retirer = $\underset{i}{\text{argmin}} \frac{matrice[i,stock]}{matrice[i,\text{à\_ajouter}]}$

mettre à_ajouter dans la base et retirer à_retirer de la base

mettre à jour le reste de la matrice

Fin Tant que

Cas difficiles

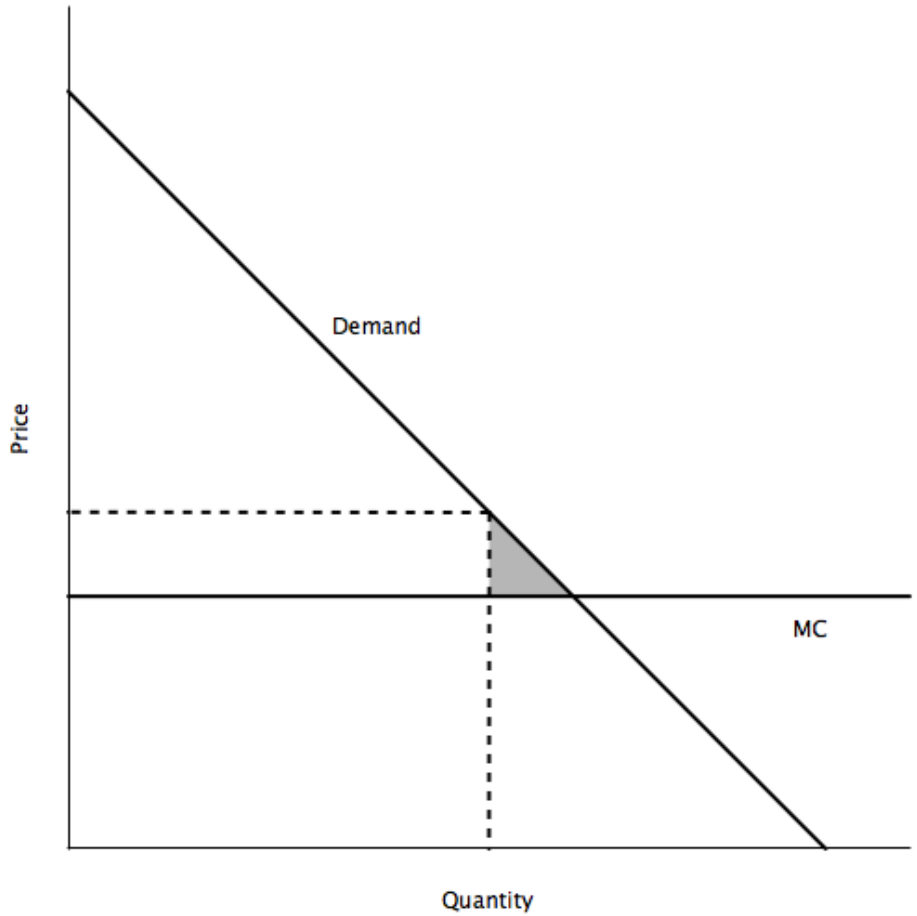
- Cas où l'ensemble de départ vide
- Cas où $(0,0) \notin C$
- Cas de dégénérescence

Équilibre de Nash

Équilibre de Nash : jouer de manière aléatoire.

- Chaines de Markov : bat aisément un humain qui joue « normalement ».
- Variantes : reviennent au Shifumi classique si le nombre d'éléments est impair.

Duopole – principe



Nos stratégies :

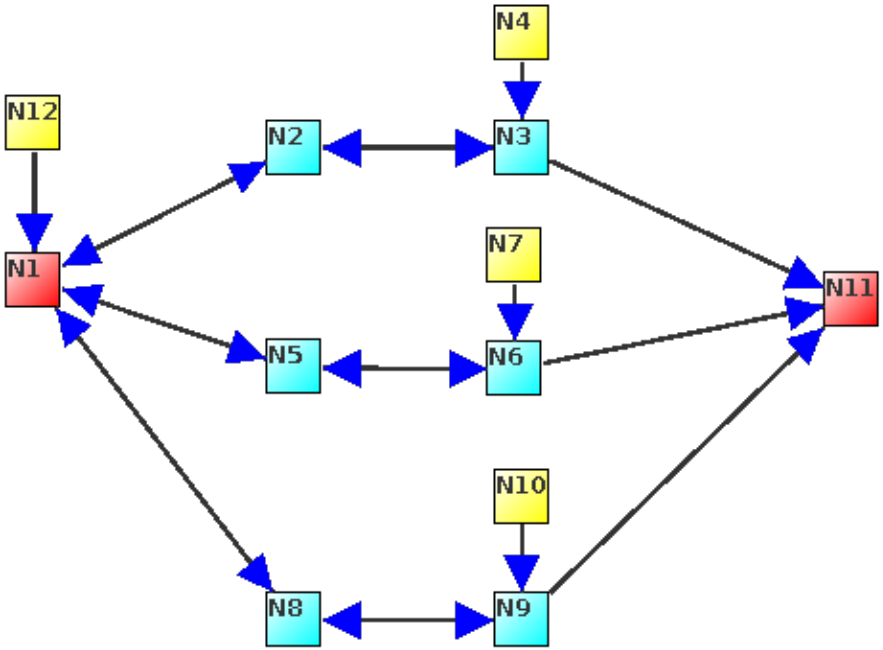
- Stackelberg en moyenne
  $x = \frac{3-y}{2}$
- Stratégie pénalisante
- Stratégie évolutive
- Stratégie polynomiale
  $f(0) = 1.125$
  $f(0.75) = 0.75$
  $f(1.5) = 0.75$

Duopole – résultats

Stratégie	Gain minimal	Gain moyen	Gain maximum
cooperatif*	561.56	978.59	1123.88
noncooperatif*	380.79	877.10	1317.08
stackelberg*	498.00	797.42	1132.44
palkeo	694.54	986.94	1123.88
Pénalise	419.12	860.12	1124.70
Pénalise variante	421.22	896.10	1123.88
Stackelberg en moyenne	492.11	923.94	1123.88
Stackelberg en moyenne (variante)	531.85	800.61	1262.25
gklmjbse	561.56	832.41	1135.33
poly	561.82	1011.24	1123.88
killer**	0.00	773.86	1133.09
cooperatifmixte**	698.67	990.58	1123.88
agressivemieux**	3.15	750.64	1126.18
best_strategie**	322.67	881.00	1262.81

TABLE : Résultats des différentes stratégies sur 1000 tours

Gare de péage



- $x[12] = random() < p_{cb}$
- $x[1] = random() < lambda$
- $x[2] = (x[2] > 0) * (x[2] - 1 + d_{32}) + d_{12}$
- $d_{12} = x[1] * d_{121} * (d_{21} <= d_{81})$
- $d_{23} = (x[2] > 0)$
- $d_{32} = x[3] * (1 - d_{43})$
- $x[3] = d_{23} + x[3] * (1 - d_{23}) * (1 - d_{43})$
- $d_{311} = x[3] * d_{43}$
- $x[10] = random() < \frac{1}{p_{cb}/\mu_{cb}+(1-p_{cb})/\mu_{ncb}}$

UA 5 Robots

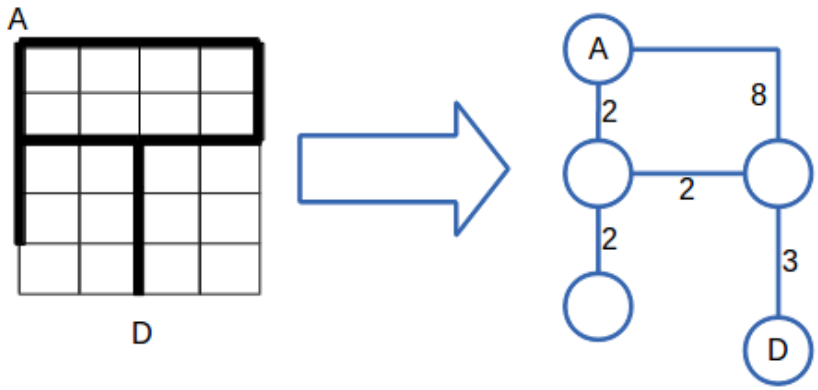
- Suivi de mur
- Algorithme de Dijkstra
- Algorithme A*

Suivi de mur

Un capteur ultrasonique à gauche.
Un capteur ultrasonique frontale.

- Si distance frontale $<$ minima : on pivote à droite.
- Sinon :
  - Si distance latérale $>$ distance voulue $+$ marge : coupe le moteur de gauche
    Sinon il est actif
  - Si distance latérale $<$ distance voulue - marge : coupe le moteur de droite
    Sinon il est actif

Algorithme de Dijkstra



Si le robot connaît le plan du labyrinthe l'algorithme de dijkstra suffit.