Google Hash Code

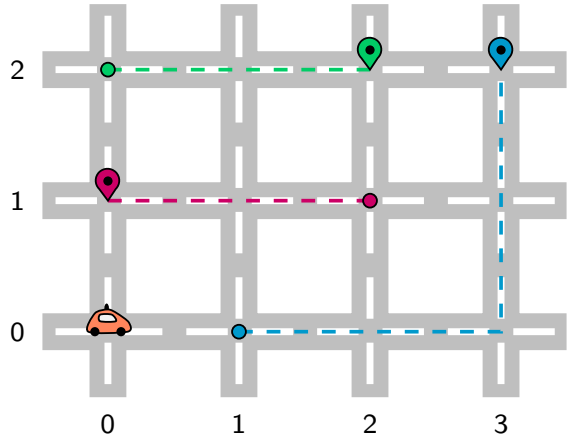
Self-driving rides

Hash Code 2018, Online Qualification Round

Notes

Énoncé

Représentation du problème



Notes

Énoncé

Représentation du problème

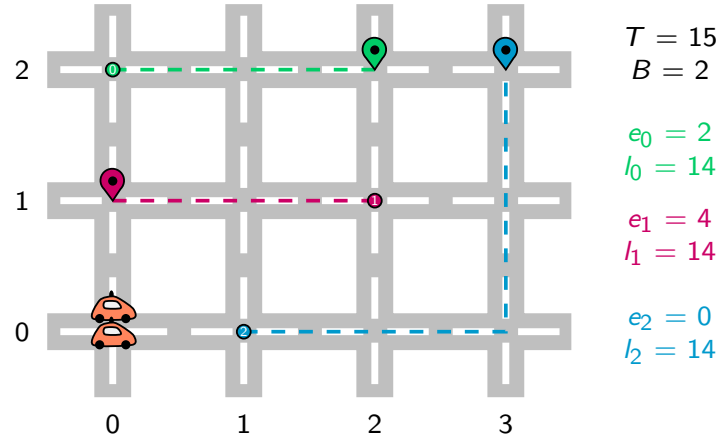
- R, C nombre de lignes et de colonnes de la grille
- F véhicules
- N courses
  - $\forall r \in [1, N], s_r, f_r$: le point de début et le point d'arrivée de la course
  - $\forall r \in [1, N], e_r, l_r$: le temps au plus tôt de début et le temps au plus tard de fin de la course
- B bonus par course commençant à l'heure
- T horizon de temps
- Score d'une course : distance de la course plus un éventuel bonus si elle est commencée à l'heure au plus tôt

Objectif : Maximiser le score de toutes les courses effectuées

Notes

Exemple

Exemple



$T = 15$
$B = 2$

$e_0 = 2$
$l_0 = 14$

$e_1 = 4$
$l_1 = 14$

$e_2 = 0$
$l_2 = 14$

Notes

Exemple

Exemple

- Grille de 3 lignes et 4 colonnes
- 2 véhicules
- 3 courses
  - $s_0 = (0, 2), f_0 = (2, 2), e_0 = 2, l_0 = 14$
  - $s_1 = (2, 1), f_1 = (0, 1), e_1 = 4, l_1 = 14$
  - $s_2 = (1, 0), f_2 = (3, 2), e_2 = 0, l_2 = 14$
- Bonus de 2
- Horizon de 15 pas de temps

Notes

Énoncé

Les variables ?

- Les courses affectées aux véhicules
  - $\forall v \in [0, F-1], L_v$: la liste des courses affectées au véhicule v

Notes

Recherche locale

Principe

- On part d'une solution initiale
- À chaque étape, on modifie la solution
  - en essayant de l'améliorer
  - en espérant obtenir un résultat optimum global
- Approche locale
  - suivant les problèmes pas de garantie d'optimalité (heuristique)
  - peu coûteuse

Solution initiale

- Solution " vide "
- Solution aléatoire
- Solution d'un algorithme glouton

Notes

Recherche locale

Principe

- On part d'une solution initiale
- À chaque étape, on modifie la solution
  - en essayant de l'améliorer
  - en espérant obtenir un résultat optimum global
- Approche locale
  - suivant les problèmes pas de garantie d'optimalité (heuristique)
  - peu coûteuse

Modifications

- Ajout d'une course à un véhicule
- Suppression d'une course à un véhicule
- Échange de courses pour un véhicule
- Échange de courses entre 2 véhicules

Notes

Recherche locale

Principe

- On part d'une solution initiale
- À chaque étape, on modifie la solution
  - en essayant de l'améliorer
  - en espérant obtenir un résultat optimum global
- Approche locale
  - suivant les problèmes pas de garantie d'optimalité (heuristique)
  - peu coûteuse

Amélioration du score

Il faut une fonction qui calcule le score

Notes

Recherche locale

Principe

- On part d'une solution initiale
- À chaque étape, on modifie la solution
  - en essayant de l'améliorer
  - en espérant obtenir un résultat optimum global
- Approche locale
  - suivant les problèmes pas de garantie d'optimalité (heuristique)
  - peu coûteuse

1. Marche aléatoire
2. Algorithme de la descente
3. Recherche Tabou

Self-driving rides | Hash Code 2018 | Online Qualification Round | 5 / 10

Notes

Recherche locale

Voisinage

Pour une solution, l'ensemble des solutions à une modification près

Exemple

- Grille de 3 lignes et 4 colonnes
- 2 véhicules
- 3 courses
  - $s_0 = (0, 2), f_0 = (2, 2), e_0 = 2, l_0 = 14$
  - $s_1 = (2, 1), f_1 = (0, 1), e_1 = 4, l_1 = 14$
  - $s_2 = (1, 0), f_2 = (3, 2), e_2 = 0, l_2 = 14$
- Bonus de 2
- Horizon de 15 pas de temps

Notes

Recherche locale

Voisinage

Pour une solution, l'ensemble des solutions à une modification près

Exemple

- $s_0 = (0, 2), f_0 = (2, 2), e_0 = 2, l_0 = 14$
- $s_1 = (2, 1), f_1 = (0, 1), e_1 = 4, l_1 = 14$
- $s_2 = (1, 0), f_2 = (3, 2), e_2 = 0, l_2 = 14$

$L_0 = [], L_1 = []$ score : 0

$L_0 = [0]$ (4, (2, 2))	$L_1 = []$ (0, (0, 0))	score : 4
$L_0 = []$ (0, (0, 0))	$L_1 = [0]$ (4, (2, 2))	score : 4
$L_0 = [1]$ (6, (0, 1))	$L_1 = []$ (0, (0, 0))	score : 4
$L_0 = []$ (0, (0, 0))	$L_1 = [1]$ (6, (0, 1))	score : 4
$L_0 = [2]$ (5, (3, 2))	$L_1 = []$ (0, (0, 0))	score : 4
$L_0 = []$ (0, (0, 0))	$L_1 = [2]$ (5, (3, 2))	score : 4

Self-driving rides | Hash Code 2018 | Online Qualification Round | 6 / 10

Notes

Recherche locale

Voisinage

Pour une solution, l'ensemble des solutions à une modification près

Exemple

- $s_0 = (0, 2), f_0 = (2, 2), e_0 = 2, l_0 = 14$
- $s_1 = (2, 1), f_1 = (0, 1), e_1 = 4, l_1 = 14$
- $s_2 = (1, 0), f_2 = (3, 2), e_2 = 0, l_2 = 14$

$L_0 = [0], L_1 = []$ score : 4

$L_0 = []$	(0, (0, 0))	$L_1 = []$	(0, (0, 0))	score : 0
$L_0 = []$	(0, (0, 0))	$L_1 = [0]$	(4, (2, 2))	score : 4
$L_0 = [0, 1]$	(7, (0, 1))	$L_1 = []$	(0, (0, 0))	score : 6
$L_0 = [0]$	(4, (2, 2))	$L_1 = [1]$	(6, (0, 1))	score : 8
$L_0 = [0, 2]$	(11, (3, 2))	$L_1 = []$	(0, (0, 0))	score : 8
$L_0 = [0]$	(4, (2, 2))	$L_1 = [2]$	(5, (3, 2))	score : 8

Notes

Recherche locale

Quel voisin choisir ?

- le meilleur
- un parmi ceux améliorant

Algorithme de la descente

- On part d'une solution
- On se déplace vers une solution du voisinage améliorant strictement l'objectif
- On peut rester bloquer dans des minimum locaux

$\Rightarrow$ On recommence à partir d'une autre solution

Notes

Recherche locale

Restarts

- Solution aléatoire
- Solution " vide ", dans laquelle on fixe un certain pourcentage de courses comme dans la meilleure solution trouvée jusqu'ici
  - 5%, 10%, 20%

Pas d'amélioration

- On se déplace vers une solution du voisinage sans améliorer l'objectif
  - Il ne faut pas être un poisson rouge

Notes

Recherche Tabou

Principe

- On part d'une solution s
- On se déplace vers la meilleure solution du voisinage qui ne soit pas interdite
- On ajoute s aux solutions interdites pour les m itérations suivantes

Mémoire

- Interdire des solutions peut être coûteux en mémoire
- À la place on interdit des mouvements
  - Si m trop faible, tabou peu efficace
  - Si m trop grand, risque de rater des solutions

Notes

Recherche Tabou

$m = 3$

$L_0 = [0], L_1 = []$ score : 0
t = [sup 0]

$L_0 = [0]$ (4, (2, 2))	$L_1 = []$ (0, (0, 0))	score : 4
$L_0 = []$ (0, (0, 0))	$L_1 = [0]$ (4, (2, 2))	score : 4
$L_0 = [1]$ (6, (0, 1))	$L_1 = []$ (0, (0, 0))	score : 4
$L_0 = []$ (0, (0, 0))	$L_1 = [1]$ (6, (0, 1))	score : 4
$L_0 = [2]$ (5, (3, 2))	$L_1 = []$ (0, (0, 0))	score : 4
$L_0 = []$ (0, (0, 0))	$L_1 = [2]$ (5, (3, 2))	score : 4

Notes

Recherche Tabou

$m = 3$

$L_0 = [0](4, (2, 2)), L_1 = [2](5, (3, 2))$ score : 8
t = [sup 0, sup 2]

$L_0 = []$ (0, (0, 0))	$L_1 = [0]$ (4, (2, 2))	score : 4
$L_0 = [0, 1]$ (7, (0, 1))	$L_1 = []$ (0, (0, 0))	score : 6
$L_0 = [0]$ (4, (2, 2))	$L_1 = [1]$ (6, (0, 1))	score : 8
$L_0 = [0, 2]$ (11, (3, 2))	$L_1 = []$ (0, (0, 0))	score : 8
$L_0 = [0]$ (4, (2, 2))	$L_1 = [2]$ (5, (3, 2))	score : 8

Notes