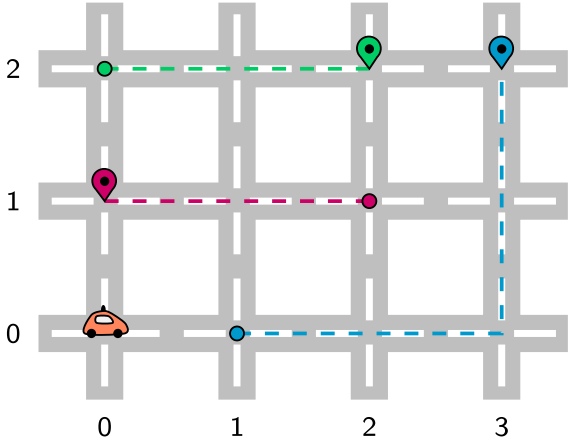
Google Hash Code

Self-driving rides

Hash Code 2018, Online Qualification Round

Problem Statement

Problem Representation



Problem Statement

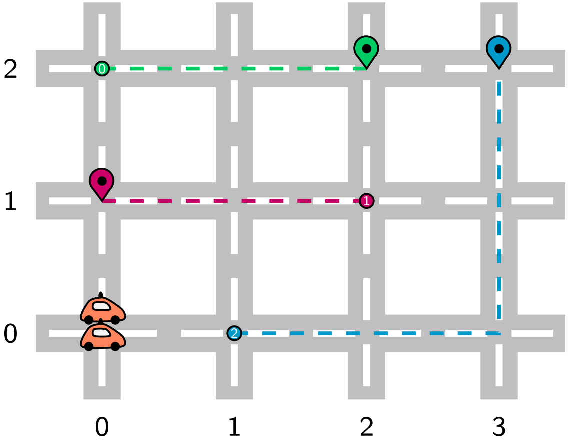
Problem Representation

- R, C number of rows and columns in the grid
- F size of the fleet (number of vehicles)
- N number of rides
  - $\forall r \in [1, N], s_r, f_r$: starting and ending points of the ride
  - $\forall r \in [1, N], e_r, l_r$: earliest start time and latest end time of the ride
- B bonus for rides that start on time
- T time horizon
- Score for a ride: distance of the ride plus a potential bonus if it starts on time

  Objective: Maximize the score for all completed rides

Example

Exemple



$T = 15$
$B = 2$

$e_0 = 2$
$l_0 = 14$

$e_1 = 4$
$l_1 = 14$

$e_2 = 0$
$l_2 = 14$

Example

Exemple

- Grid with 3 rows and 4 columns
- 2 vehicles
- 3 rides
  - $s_0 = (0, 2), f_0 = (2, 2), e_0 = 2, l_0 = 14$
  - $s_1 = (2, 1), f_1 = (0, 1), e_1 = 4, l_1 = 14$
  - $s_2 = (1, 0), f_2 = (3, 2), e_2 = 0, l_2 = 14$
- Bonus: 2
- Time horizon: 15 time steps

Problem Statement

Variables?

- The rides assigned to the vehicles
  - $\forall v \in [0, F-1], L_v$: the list of rides assigned to vehicle v

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
  - trying to improve the value of the objective function
  - hoping to achieve the global optimum
- Local approach
  - depending on the problem, no guarantee of optimality (heuristic)
  - low cost

Initial solution

- "Empty" solution
- Random solution
- Solution from a greedy algorithm

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
  - trying to improve the value of the objective function
  - hoping to achieve the global optimum
- Local approach
  - depending on the problem, no guarantee of optimality (heuristic)
  - low cost

Modifications

- Add a ride to a vehicle
- Remove a ride from a vehicle
- Swap rides within a vehicle
- Swap rides between 2 vehicles

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
  - trying to improve the value of the objective function
  - hoping to achieve the global optimum
- Local approach
  - depending on the problem, no guarantee of optimality (heuristic)
  - low cost

Improving the score

Need for a function computing the score

Local Search

Principle

- Start from an initial solution
- At each step, modify the solution
  - trying to improve the value of the objective function
  - hoping to achieve the global optimum
- Local approach
  - depending on the problem, no guarantee of optimality (heuristic)
  - low cost

1. Random walk
2. Gradient descent
3. Tabu Search

Local Search

Neighborhood

For a solution, the set of solutions with one modification

Exemple

- Grid of 3 rows and 4 columns
- 2 vehicles
- Bonus of 2
- Time horizon of 15 time steps
- 3 rides
- $s_0 = (0, 2), f_0 = (2, 2), e_0 = 2, l_0 = 14$
- $s_1 = (2, 1), f_1 = (0, 1), e_1 = 4, l_1 = 14$
- $s_2 = (1, 0), f_2 = (3, 2), e_2 = 0, l_2 = 14$

Local Search

Neighborhood

For a solution, the set of solutions with one modification

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

Local Search

Neighborhood

For a solution, the set of solutions with one modification

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

Local search

Which neighbor to choose?

- Randomly
- The best
- One of the best

Gradient descent

- We start with an initial solution
- At each step, we move towards a solution in the neighborhood **strictly improving** the objective
- You can get stuck in local minima

$\Rightarrow$ Start again from another solution

Local search

Restarts

- Random solution
- "Empty" solution, in which a certain percentage of variables is fixed as in the best solution found so far
  - 5%, 10%, 20%

No improvement

- We move towards a solution in the neighborhood **without improving** the objective
  $\Rightarrow$ Don't be a goldfish

Tabu Search

Principle

- We start from a solution s.
- We move towards **the best** solution in the neighbourhood which is not **forbidden**
- Add s to the forbidden solutions for the next m iterations

Memory

- Prohibiting solutions can be memory-intensive
- Instead we forbid movements
  - If m too small $\Rightarrow$ blocking search around a local optimum
  - If m too large $\Rightarrow$ risk of missing solutions

Tabu Search

$m = 3$

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

Tabu Search

$m = 3$

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

t = [del 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

Tabu Search

$m = 3$

$L_0 = [0](4, (2, 2)), L_1 = [](0, (0, 0))$ score: 4

t = [del 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

Tabu Search

$m = 3$

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

t = [del 0, del 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

Tabu Search

$m = 3$

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

t = [del 0, del 2]

$L_0 = []$	(0, (0, 0))	$L_1 = [2, 0]$	(10, (2, 2))	score: 6
$L_0 = [0, 2]$	(11, (3, 2))	$L_1 = []$	(0, (0, 0))	score: 8
$L_0 = [0, 1]$	(7, (0, 1))	$L_1 = [2]$	(5, (3, 2))	score: 10
$L_0 = [0]$	(4, (2, 2))	$L_1 = [2, 1]$	(9, (0, 1))	score: 10

Tabu Search

$m = 3$

$L_0 = [0, 1](7, (0, 1)), L_1 = [2](5, (3, 2))$ score: 10

t = [del 0, del 2, del 1]

$L_0 = []$	(0, (0, 0))	$L_1 = [2, 0]$	(10, (2, 2))	score: 6
$L_0 = [0, 2]$	(11, (3, 2))	$L_1 = []$	(0, (0, 0))	score: 8
$L_0 = [0, 1]$	(7, (0, 1))	$L_1 = [2]$	(5, (3, 2))	score: 10
$L_0 = [0]$	(4, (2, 2))	$L_1 = [2, 1]$	(9, (0, 1))	score: 10

Tabu Search

$m = 3$

$L_0 = [0, 1](7, (0, 1)), L_1 = [2](5, (3, 2))$ score: 10

t = [del 0, del 2, del 1]

$L_0 = [1, 0]$	(9, (2, 2))	$L_1 = [2]$	(5, (3, 2))	score: 10
$L_0 = [1]$	(6, (0, 1))	$L_1 = [2, 0]$	(10, (2, 2))	score: 10
$L_0 = [0]$	(4, (2, 2))	$L_1 = [2, 1]$	(9, (0, 1))	score: 10
$L_0 = [0, 1, 2]$	(13, (3, 2))	$L_1 = []$	(0, (0, 0))	score: 10

Tabu Search

$m = 3$

$L_0 = [0](4, (2, 2)), L_1 = [2, 1](9, (0, 1))$ score: 10

t = [del 2, del 1, swap 1]

$L_0 = [1, 0]$	(9, (2, 2))	$L_1 = [2]$	(5, (3, 2))	score: 10
$L_0 = [1]$	(6, (0, 1))	$L_1 = [2, 0]$	(10, (2, 2))	score: 10
$L_0 = [0]$	(4, (2, 2))	$L_1 = [2, 1]$	(9, (0, 1))	score: 10
$L_0 = [0, 1, 2]$	(13, (3, 2))	$L_1 = []$	(0, (0, 0))	score: 10

Tabu Search

$m = 3$

$L_0 = [0](4,(2,2)), L_1 = [2,1](9,(0,1))$ score: 10

t $=$ [del 2, del 1, swap 1]

$L_0 = []$	(0, (0, 0))	$L_1 = [2,1]$	(9, (0, 1))	score: 6
$L_0 = []$	(0, (0, 0))	$L_1 = [2,1,0]$	(12, (2, 2))	score: 8
$L_0 = [0]$	(4, (2, 2))	$L_1 = [1,2]$	(12, (3, 2))	score: 12
$L_0 = [0,2]$	(11, (3, 2))	$L_1 = [1]$	(6, (0, 1))	score: 12

Tabu Search

$m = 3$

$L_0 = [0](4, (2, 2)), L_1 = [1, 2](12, (3, 2))$ score: 12

t = [del 1, swap 1, swap 2]

$L_0 = []$	(0, (0, 0))	$L_1 = [2, 1]$	(9, (0, 1))	score: 6
$L_0 = []$	(0, (0, 0))	$L_1 = [2, 1, 0]$	(12, (2, 2))	score: 8
$L_0 = [0]$	(4, (2, 2))	$L_1 = [1, 2]$	(12, (3, 2))	score: 12
$L_0 = [0, 2]$	(11, (3, 2))	$L_1 = [1]$	(6, (0, 1))	score: 12

Tabu Search

$m = 3$

$L_0 = [0](4, (2, 2)), L_1 = [1, 2](12, (3, 2))$ score: 12

t = [del 1, swap 1, swap 2]

$L_0 = []$	(0, (0, 0))	$L_1 = [1, 2]$	(12, (3, 2))	score: 8
$L_0 = [0]$	(4, (2, 2))	$L_1 = [1]$	(6, (0, 1))	score: 8
$L_0 = []$	(4, (2, 2))	$L_1 = [1, 2, 0]$	(12, (3, 2))	score: 8