TIPE 2024

Apprendre à une intelligence artificielle à jouer à Snake en utilisant un algorithme génétique

Marilou Bernard de Courville

N° SCEI 41188

16 octobre 2023

Introduction

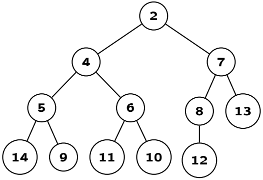
Problématique et pertinence au regard du thème de l'année

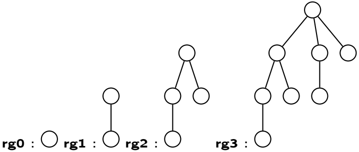
- **Objectif :** parcourir efficacement la ville dans un but touristique.
- **Recherche opérationnelle :** permet d'atteindre l'objectif en réalisant des optimisations sous contraintes [1].
- **Deux cas** envisagés :
  1. optimiser un trajet en métro via l'algorithme de Dijkstra et différentes structures de données codées en OCaml [3] ;
  2. optimiser un trajet soit en métro ou à pied en passant par plusieurs types de chemins en utilisant un solveur linéaire (CPLEX d'IBM) et un code en Python.
- Ces deux cas sont modélisés à l'aide de **graphes orientés**.

Trajet le plus court en métro - modélisation

Modèles de données utilisés (types)

- Un tas est une représentation organisée en arbre des stations en fonction d'une priorité (distance du chemin).
- Représentation en tas (heap) efficace en complexité pour accéder au nœud de priorité minimum et mettre à jour les priorités.
- Trois structures étudiées : tas non modifiables (*immutable heaps*), tas modifiables, et tarbres (*treap*).
- Programmes réalisés en OCaml utilisent seulement le module `Hashtbl` de la bibliothèque standard pour manipuler les tas.





2	4	7	5	6	8	13	14	9	11	10	12

Élément	2	4	5	6	7	8	9	10	11	12	13	14
Position	0	1	3	4	2	5	8	10	9	11	6	7

Marilou Bernard de Courville

Trajet le plus court en métro - algorithme

Algorithme de Dijkstra et influence des structures de données sur la complexité

Require: Un graphe $G = (V, A)$, V sommets, A arêtes
Require: Un noeud source s
Ensure: d tableau des plus court chemins de s vers $v \in V$
  for all $v \in V[G]$ **do**
    $d[v] \leftarrow +\infty$, $\text{père}[v] \leftarrow \text{None}$
  end for
  $d[s] \leftarrow 0$, $S \leftarrow \emptyset$, $Q \leftarrow V[G]$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow \text{Extrait}_{\text{Min}}(Q)$
    $S \leftarrow S \cup \{u\}$
    for all arête (u, v) d'origine u **do**
      if $d[u] + w(u, v) < d[v]$ **then**
        $d[v] \leftarrow d[u] + w(u, v)$,
  $\text{père}[v] := u$
      end if
    end for
  end while

Table – Dijkstra : complexité (opérations)

Implantation	Complexity
Naif	$\mathcal{O}\left(V^2 + A\right)$
Tas	$\mathcal{O}\left((V + A)\log V\right)$

Table – Simulations : temps exécution pour toutes les stations du métro

Type	Temps exécution
Naif	960ms
Tas non mutable v1	312ms
Tas mutable	191ms
Tarbre	1018ms
Tas non mutable v2	145ms

Plus compliqué : plus petit chemin passant par toutes les lignes du métro

Étude de la résolution à l'aide d'un solveur linéaire

- Sujet traité par Florian Sikora [7] par étude de graphe coloré pour le réseau du métro ;
- Sommet : station, arête : trajet entre deux stations connexes, arête colorée par la couleur de la ligne reliant les stations
- Problème du "Generalised directed rural postman" [2] ;
- Problème NP-difficile : pas de solution en temps polynomial [6] ;
- Résolution requiert l'utilisation d'un solveur linéaire (CPLEX d'IBM) pour "integer linear programming" (ILP) [9] ;
- Fait intervenir une matrice de contraintes (MIP) de `1270x1847, 6999 coeffs non nuls.`

Formalisation du problème : variables

- Ensemble V des sommets, A des arêtes, C des couleurs. $(u, v) \in V^2$, $u \xrightarrow[l \in C]{} v \in A$.
- $x_{u,v,l} \in \{0, 1\}$: variable binaire pour chaque arc $u \underset{l}{\rightarrow} v$ (sur ligne l), avec $x = 1$ si l'arc est considéré, $x = 0$ sinon.
- $w_{u,v,l} \in \mathbb{N}$: est le temps pour parcourir l'arête $u \underset{l}{\rightarrow} v$
- $(u, v) \in V^2, f_{u,v,l}, y_v \in \mathbb{N}$ sont les flots des arcs/sommets : positifs si l'arc/sommet est sur le chemin considéré.
- s, t sont les points de départ/arrivée fictifs (temps nul pour rejoindre tout sommet).
- $\forall((u, v, l_1), (v, w, l_2)) \in A^2, z_{u,v,w,l_1,l_2} \in \{0, 1\}$ indique si deux arêtes sont utilisées consécutivement $u \xrightarrow[l_1]{} v \xrightarrow[l_2]{} w$.

Formalisation du problème : optimisation

Objectif : minimiser distance $\sum_{(u,v,l)\in A} w_{u,v,l} \times x_{u,v,l}$ sous contraintes.

Autant de chemins qui entrent sur un sommet et qui en sortent :	$\forall v \in V\backslash\{s,t\}, \sum_{(u,v,l)\in A} x_{u,v,l} = \sum_{(v,w,l)\in A} x_{v,w,l}$	(1)
Unique chemin de la source et vers la cible :	$\sum_{(s,v,l)\in A} x_{s,v,l} = \sum_{(u,t,l)\in A} x_{u,t,l} = 1$	(2)
Pour chaque ligne au moins 1 arc de cette ligne sélectionné :	$\forall l \in C, \sum_{(u,v,l)\in A} x_{u,v,l} \geq 1$	(3)
Evite solutions disjointes : le flot est décroissant pour solution connectée	$\forall (u,v,l) \in A, \mid V \mid x_{u,v,l} \geq f_{u,v,l}$	(4)
Tous les sommets perdent du flot sauf la source :	$\forall v \in V\backslash\{s\}, \sum_{(u,v,l)\in A} f_{u,v,l} - \sum_{(v,w,l)\in A} f_{v,w,l} \geq y_v$	(5)
Le flot au niveau du sommet est positif s'il est dans la solution :	$\forall v \in V, y_v - \sum_{(u,v,l)\in A} x_{u,v,l} - \sum_{(v,w,l)\in A} f_{v,w,l} \geq 0$	(6)
Evite de prendre deux fois le même sommet :	$y_v \geq 2$	(7)
Chemin consécutif :	$x_{u,v,l_1} + x_{v,w,l_2} \leq z_{u,v,w,l_1,l_2} + 1$	(8)
Evite de reprendre deux fois la même ligne :	$\forall l \in C, \sum_{(u,v,l_1),(v,w,l_2) \text{ / } l=l_1 \text{ ou } l=l_2} z_{u,v,w,l_1,l_2} = 2$	(9)

Conclusion

- Deux problématiques urbaines traitées :
  - optimisation d'un trajet en métro
  - parcours touristique efficace d'une ville en empruntant différents types de chemins
- Pertinence de la modélisation des problèmes urbains par des graphes pour les résoudre.
- Application de la recherche opérationnelle pour trouver une solution.
  - Optimisation fait intervenir un grand nombre de contraintes résultant en des problèmes combinatoires complexes sans solution analytique.
  - Importance du choix des algorithmes et structures de données pour obtenir des solutions pratiques efficaces.