Graphes et IA

Section 2 : Bases des graphes

Présentation de **Kevin TRANCHO**

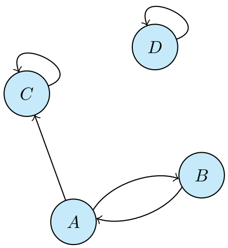
dispensé en classe de seconde année

à l'**ESGI** Paris

école supérieure de génie informatique

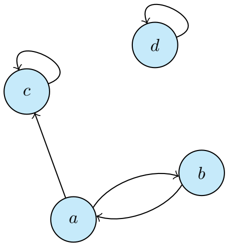
Concept

Schématiquement : un graphe est ensemble de nœuds reliés par des flèches.



Mathématiquement : un graphe $\mathcal{G}\,(S, A)$ est modélisé par un ensemble de *sommets* S et un ensemble d'*arcs* $A \subseteq S \times S$.

Concept



Ici, le graphe $\mathcal{G}\,(S, A)$ est tel que :

$$S = \{a, b, c, d\}$$

$$A = \{(a, b), (a, c), (b, a), (c, c), (d, d)\}$$

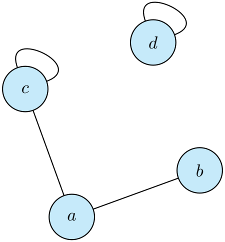
Arcs

Pour un graphe $\mathcal{G}\left(S, A\right)$, si $a = (d, f) \in A$:

- a est l'arc partant de $d \in S$ jusqu'à $f \in S$.
- f est un *successeur* de s dans $\mathcal{G}$.
- si $s = t$, alors a correspond à une boucle sur a.

Si pour tout $(s, t) \in A$, nous avons $(t, s) \in A$. Alors $\mathcal{G}$ est un graphe *non-orienté*.

Exemple de graphe non-orienté



Ici, le graphe non-orienté $\mathcal{G}\left(S, A\right)$ est tel que :

$$S = \{a, b, c, d\}$$

$$A = \{(a, b), (a, c), (b, a), (c, a), (c, c), (d, d)\}$$

Voisinage et pondération

Pour un graphe $\mathcal{G}(S, A)$, nous pouvons définir le voisinage V de chaque sommet $s \in S$ comme la liste de ses successeurs :

$$\text{out}(s) = V(s) = \{t \mid t \in S \ / \ (s, t) \in A\}$$

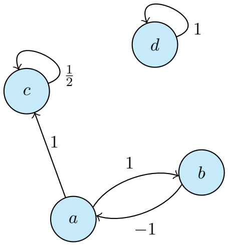
Nous pouvons définir un graphe pondéré $\mathcal{G}(S, A, p)$ tel que pour tout arc $a \in A$, l'application p associe à l'arc a un poids ou transition w :

$$p(a) = w$$

Il existe plusieurs stratégies pour sauvegarder l'application $p : A \to \mathbb{K}$.

Exemple de graphe pondéré

Soit le graphe $\mathcal{G}\,(S, A, p)$ illustré.



Nous avons l'application p telle que :

$$
\begin{array}{rrcl}
p & : \qquad A & \rightarrow & \mathbb{R} \\
& (a,b) & \mapsto & 1 \\
& (a,c) & \mapsto & 1 \\
& (b,a) & \mapsto & -1 \\
& (c,c) & \mapsto & \frac{1}{2} \\
& (d,d) & \mapsto & 1
\end{array}
$$

Principales implémentations des successeurs

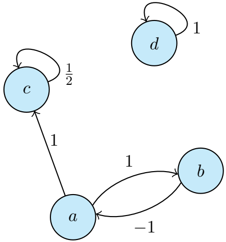
- Depuis modèle théorique : ensemble des arcs dans une table de hachage.
- Matrice d'adjacences : les transitions $(v_i, v_j) \in A$ peuvent se modéliser par les coefficients d'une matrice M :

$$M_{i,j} = p\left((v_i, v_j)\right)$$

- Listes d'incidence : pour chaque sommet, nous pouvons construire une liste (ou table de hachage) de ses transitions.

Matrice d'adjacence

Soit le graphe $\mathcal{G}\left(S, A, p\right)$ illustré.

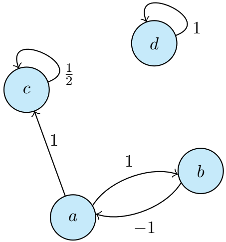


p peut se modéliser par la matrice d'adjacence suivante :

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Listes d'incidence

Soit le graphe $\mathcal{G}(S, A, p)$ illustré.



p peut se modéliser par les listes d'incidence suivantes :

$$
\begin{array}{cccl}
p & : & a & \rightarrow & \{b \rightarrow 1, \\
 & & & & \ c \rightarrow 1\} \\
 & & b & \rightarrow & \{a \rightarrow -1\} \\
 & & c & \rightarrow & \left\{c \rightarrow \frac{1}{2}\right\} \\
 & & d & \rightarrow & \{d \rightarrow 1\}
\end{array}
$$

Temps de pratique

Codez les 3 modèles d'implémentation de graphes en python (voir lab 01 associé).

Diffusion de flot dans un graphe

Dans un graphe pondéré $\mathcal{G}(S, A, p)$, le flot de diffusion se calcule :

- Depuis un vecteur $v \in \mathbb{R}^{|S|}$ (d'initialisation) correspondant aux quantités actuelles (ou initiales) dans chaque nœud.
- Le flot $w \in \mathbb{R}^{|S|}$ se calcule comme la somme pondérées des quantités des prédécesseurs par leurs transitions :

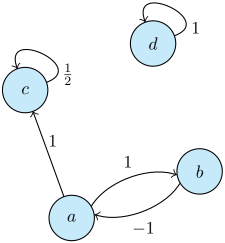
$$\forall i \in [|S|], w_i = \sum_{j \in [|S|], (S_j, S_i) \in A} v_j p(v_j)$$

- À noter que ceci peut se modéliser par une multiplication par la matrice d'adjacence M correspond aux transitions de $\mathcal{G}$:

$$w = vM$$

Diffusion de flot dans un graphe

Soit le graphe $\mathcal{G}\left(S, A, p\right)$ illustré.

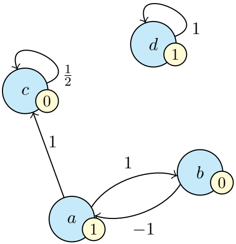


Soit M sa matrice d'adjacence :

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Diffusion de flot dans un graphe

Soit le graphe $\mathcal{G}\left(S, A, p\right)$ illustré pour $v = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$:

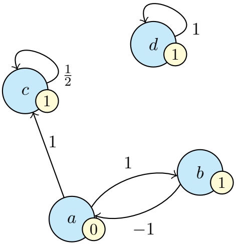


Le flot au rang suivant :

$$\begin{aligned} vM &= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix} \end{aligned}$$

Diffusion de flot dans un graphe

Soit le graphe $\mathcal{G}(S, A, p)$ illustré pour $v = \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}$:

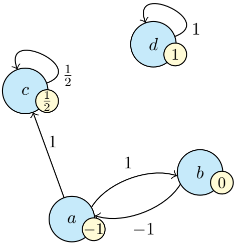


Le flot au rang suivant :

$$
\begin{aligned}
vM^2 &= \begin{pmatrix}0\\1\\1\\1\end{pmatrix}^T \begin{pmatrix}0 & 1 & 1 & 0\\ -1 & 0 & 0 & 0\\ 0 & 0 & \frac{1}{2} & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\\
&= \begin{pmatrix}-1 & 0 & \frac{1}{2} & 1\end{pmatrix}
\end{aligned}
$$

Diffusion de flot dans un graphe

Soit le graphe $\mathcal{G}\left(S, A, p\right)$ illustré pour $v = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$:

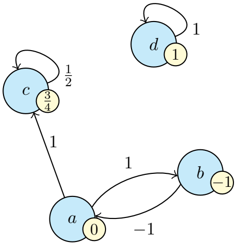


Le flot au rang suivant :

$$
\begin{aligned}
vM^3 &= \begin{pmatrix} -1 \\ 0 \\ \frac{1}{2} \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 0 & -1 & \frac{3}{4} & 1 \end{pmatrix}
\end{aligned}
$$

Diffusion de flot dans un graphe

Soit le graphe $\mathcal{G}(S, A, p)$ illustré pour $v = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$:



Le flot au rang suivant :

$$
\begin{aligned}
vM^4 &= \begin{pmatrix} 0 \\ -1 \\ \frac{3}{4} \\ 1 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & \frac{3}{8} & 1 \end{pmatrix}
\end{aligned}
$$

Chemins

Pour un graphe $\mathcal{G}\left(S, A\right)$,

- Un chemin de $d \in S$ vers $f \in S$ est une suite $(a_1, a_2, \cdots, a_n) \in S^n$ avec $a_1 = d$ et $a_n = f$ telle que pour tout $i \in [n-1]$, nous avons $(a_i, a_{i+1}) \in A$.

Parcours

Programme 1.1: Parcours d'un graphe

entrée : G : graphe (S : sommets, A : arcs)
entrée : départ $\in S$
sortie : ordre : suite de sommets

```
ordre ← ⟨⟩
visités ← ∅
à faire ← {s}
tant que |à faire| > 0 faire
    s ← à faire.extraire()
    si s ∈ visités alors
        relancer la boucle
    ordre.ajouter (s)
    visités ← visités ∪ {s}
    à faire ← à faire ∪ G.voisins (s)
```

Parcours en profondeur

Programme 1.2: Parcours en profondeur d'un graphe

fonction *ParcoursProfondeur* $(G, s, \text{ordre}, \text{visités})$
- **si** $s \in$ visités **alors**
  - S'arrêter.
- visités $\leftarrow$ visités $\cup \{s\}$
- ordre.ajouter (s)
- **pour** $t \in G.$*voisins* (s) **faire**
  - ParcoursProfondeur $(G, t, \text{ordre}, \text{visités})$

entrée : G : graphe (S : sommets, A : arcs)
entrée : départ $\in S$
sortie : ordre : suite de sommets
ordre $\leftarrow \langle\rangle$
visités $\leftarrow \emptyset$
ParcoursProfondeur $(G, \text{départ}, \text{ordre}, \text{visités})$

Parcours en largeur

Programme 1.3: Parcours en largeur d'un graphe

entrée : G : graphe $(S$: sommets, A : arcs$)$
entrée : départ $\in S$
sortie : ordre : suite de sommets
ordre $\leftarrow \langle\rangle$
visités $\leftarrow \emptyset$
à faire $\leftarrow$ deque (s)
tant que $|$à faire$| > 0$ **faire**
    $s \leftarrow$ à faire.extraireDébut$()$
    si $s \in$ visités **alors**
        relancer la boucle
    ordre.ajouterFin (s)
    visités $\leftarrow$ visités $\cup \{s\}$
    à faire $\leftarrow$ à faire $\cup$ G.voisins (s)

Graphe transposé

Pour un graphe $\mathcal{G}(S, A)$, son graphe transposé $\mathcal{G}^T\left(S, A^T\right)$ est tel que :

$$A^T = \{(t, s) \mid (s, t) \in A\}$$

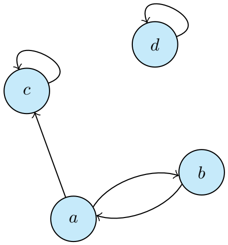
Sa matrice d'adjacence est la matrice transposée de la matrice d'adjacence de $\mathcal{G}$.

Composantes fortement connexes

Pour un graphe $\mathcal{G}(S, A)$,

- $\mathcal{G}$ est fortement connexe si pour tous $(s, t) \in S^2$ il existe un chemin entre s et t.
- Un graphe peut se partitionner en composantes fortement connexes.

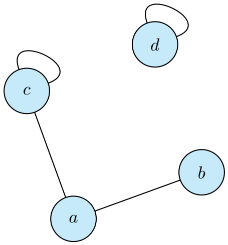
Composantes fortement connexes



Les composantes fortement connexes du graphe $\mathcal{G}(S, A)$ illustré ci-dessus sont données par la partition C :

$$C = \{\{a, b\}, \{c\}, \{d\}\}$$

Composantes fortement connexes



Les composantes fortement connexes du graphe non-orienté $\mathcal{G}\left(S, A\right)$ illustré ci-dessus sont données par la partition C :

$$C = \left\{\left\{a, b, c\right\}, \left\{d\right\}\right\}$$

Détection de cycle

Programme 1.4: Détection d'un cycle dans un graphe

fonction *ParcoursCycles* $(G, s, \text{états})$
  états $[s] \leftarrow$ 'en cours'
  pour $t \in G.$*voisins* (s) **faire**
    si états $[s] =$ *'en cours'* **alors**
      retourner ***Vrai***
    si états $[s] =$ *'à visiter'* $\wedge$ *ParcoursCycles* $(G, t, \text{états})$ **alors**
      retourner ***Vrai***
  états $[s] \leftarrow$ 'fait'
  retourner ***Faux***

entrée : G : graphe $(S$: sommets, A : arcs$)$
sortie : cycle : booléen
états $\leftarrow \{s \rightarrow$ 'à visiter' $\mid s \in S\}$
cycle $\leftarrow$ **Faux**
pour $s \in S$ **faire**
  si *ParcoursCycles* $(G, s, \text{états})$ **alors**
    cycle $\leftarrow$ **Vrai**

Tri topologique

Programme 1.5: Réaliser un tri topologique dans un graphe acyclique

entrée : G : graphe (S : sommets, A : arcs)
sortie : ordre : suite de sommets
$G\,(S, A) \leftarrow G$.transpose()
ordre $\leftarrow \langle\rangle$
tant que $|S| > 0$ **faire**
    $choix \leftarrow \{s \mid s \in S \;/\; |G.\text{voisins}\,(s)| = 0\}$ .extraire()
    ordre.ajouter $(choix)$
    G.supprimer $(choix)$

Questions

Avez-vous des questions ?