Encyclopédie de cours
-
Théorie des Graphes

Introduction

Graphe : G = (X,A) avec X ensemble de sommets et A partie symétrique d'arêtes a = (x,y).
(a incidente en x et y, x et y voisins)

Graphe sans boucle : Pas d'arêtes (x,x)

Ordre du graphe : nombre de sommets

Graphe simple (ou 1-graphe) : ni boucles ni arêtes doubles

Graphe orienté (ou digraphe) : les arêtes ont une origine et un extrémité (donc un sens, ce sont des arcs) (sommets adjacents)

Γ(x) : ensemble des voisins de x

Γ : Application multivoque

$Γ^{-1}$: Application multivoque réciproque (ensemble des prédécesseurs de x, seulement dans le cas de graphe orienté

Cas d'un 1-graphe : G = (X, Γ)

Degré sortant $d_s(x)$ (ou demi-degré extérieur) : nombre d'arcs sortant de x (sans y boucler)

Degré sortant $d_e(x)$ (ou demi-degré intérieur) : nombre d'arcs entrant dans x (sans y boucler)

Degré de x (ou valence) : $d_s(x)$ + $d_e(x)$

Puits : Sommet avec que des arcs entrants

Source : Sommet avec que des arcs sortants

Sommet isolé : Sommets sans arc incident (degré 0)

Graphe réflexif : au moins une boucle sur chaque sommet

Graphe symétrique : graphe orienté mais 2 arcs entre chaque sommets (1 dans chaque sens)

Graphe antisymétrique : graphe orienté avec un seul sens d'arc entre chaque sommet

Graphe transitif : (x,y) et (y,z) appartiennent à A => (x,z) appartient à A

Graphe complet : au moins un arc entre chaque paire de sommet

Clique : sous-ensemble de sommets complet

Sous-graphe : graphe engendré par un sous-ensemble de sommets

Graphe partiel : graphe engendré par un sous-ensemble d'arêtes

Mode de représentation d'un graphe

Représentation machine : liste des prédécesseurs / successeurs de chaque nœud

Matrice d'adjacence : indice ligne ⇔ n° du nœud de départ, indice colonne = n° d'arrivée, 1 si nœuds liés, 0 sinon

Matrice d'incidence : indice ligne = n° du nœud, indice colonne = n° de l'arc, -1 si nœud source, 1 si nœud destination, 0 sinon

Etude de la connexité

Chaîne: succession de nœuds et d'arêtes

Extrémités : nœuds au bout de la chaîne

Longueur : nbr d'arêtes de la chaîne

Chaîne élémentaire : chaque nœud apparaît une seule fois max

Chaîne simple : chaque arête apparaît une fois max

Cycle : les 2 extrémités coïncident

Cycle élémentaire : idem chaîne => ne contient aucun autre cycle

Chemin : chaîne mais orientée

Chemin élémentaire/simple : idem que chaîne

Circuit : cycle orienté

Circuit élémentaire : rencontre jamais 2 fois le même sommet

Graphe connexe : Toute paire de sommets est reliée par une chaîne

Nombre de connexité : =1 si graphe connexe

Composante connexe : sous-graphe connexe d'un graphe

Point d'articulation : coupe graphe en 2 si supprimé (augmente nombre de composantes connexes)

Isthme : arête, idem que point d'articulation

Ensemble d'articulation : ensemble de sommets dont la suppression rend le graphe non-connexe

Graphe (orienté) fortement connexe : Toute paire de sommets est reliée par un chemin

Composantes fortement connexes : idem que composantes connexes mais avec chemin

Graphe réduit : graphe divisé par relation de forte connexité = toute composante connexe assimilée à un sommet. Graphe sans circuit

Soit μ un cycle. μ+ = nombre d'arcs orientés dans le sens de parcours, μ- dans le sens opposé.
Vecteur μ : colonnes = arêtes du graphe, 1 si appartient à μ+, -1 si à μ-, 0 sinon

Cycles dépendants : Il existe (λ1, …, λi) non-nul tq λ1μ1+…+λiμi = 0

Cycles indépendants : λ1μ1+…+λiμi = 0 => (λ1, …, λi) =(0,…,0)

Base de cycles : ensemble de cycles dont les vecteurs permettent par combinaison linéaire de créer tous les autres

Nombre cyclomatique : dimension de la base de cycles

Parcours eulériens et hamiltoniens

Chaîne eulérienne : emprunte une seule fois toutes les arêtes

Cycle eulérien : chaîne eulérienne avec extrémités qui coïncident

Graphe eulérien : possède un cycle eulérien

Th : graphe non-orienté connexe possède chaîne eulérienne ⇔ 0 ou 2 sommets de degré impair

Th : graphe possède cycle eulérien ⇔ tout sommet de degré pair
Chemin eulérien : chaîne eulérienne orientée

Circuit eulérien : Cycle eulérien orienté

Graphe eulérien : possède un circuit eulérien

Th : graphe orienté connexe admet chemin eulérien ⇔ 1 sommet a un arc entrant de plus, 1 sommet un arc sortant de plus, les autres le même nombre d'e/s

Th : graphe orienté connexe admet circuit eulérien ⇔ tout sommet a autant d'entrées que de sorties

Pb : postier chinois : parcourir toutes les rues d'une ville avec le moins de distance possible

Th : graphe non-orienté admet cycle chinois ⇔ graphe connexe

Th : graphe orienté admet circuit chinois ⇔ graphe fortement connexe

Chaîne/Chemin hamiltonien : passe une seule fois pas chaque sommet (graphe connexe d'ordre n) => chemin/chaîne de longueur n-1

Cycle/Circuit hamiltonien : passe une seule fois par chaque sommet

Graphe hamiltonien : contient un cycle/circuit hamiltonien

Pb : voyageur de commerce : parcourir n sommets en revenant au départ, en connaissant les poids des arcs -> chercher un cycle/chaîne hamiltonien de longueur minimale

Cycle/circuit pré-hamiltonien : passe au moins une fois par chaque sommet

Th : Graphe pré-hamiltonien ⇔ graphe connexe / fortement connexe

Méthodes de recherche de chemins

G1, G2, G3, G4 de matrices d'adjacence A, B, C, D ; ⊕=ou logique ; ⊗=et logique
C = A⊕B : Cij = Aij⊕Bij => G3 constitué des arcs de G1 et de G2
D = A⊗B : Dij = (Ai1⊗B1j) ⊕ (Ai2⊗B2j) ⊕ … (Ain⊗Bnj) => G4 constitué d'arcs (i,j) ou il existe k tq (i,k) arc de G1 et (k,j) arc de G2
U2 = U ⊗ U ➔ G', (i,j) ϵ G' ⇔ ∃chemin de longueur 2 entre i et j dans G
Up : existence des chemins de longueur p (démo par récurrence)

Pb : plus court chemin : graphe orienté, l(a) = lij = longueur de l'arc a, trouver le chemin entre i et j μ(i,j) le plus court

Matrice des plus courts chemins : L=(lij) avec lij longueur de l'arc (i,j) (infini si par d'arc)
L^(k)=(l^(k)ij) avec l^(k)ij le plus court chemin dei vers j, en passant uniquement par des sommets compris entre 1 et k. L^(0) = L

Algo de Dijkstra :
-On calcule la distance pour aller à chaque successeur (longueur de l'arc + distance déjà parcouru depuis le nœud actuel = 0 si premier nœud), et on le marque avec On note aussi le nœud actuel comme prédécesseur. Si il a déjà été visité avec une distance plus courte, ignore, si plus longue, remplace (distance et prédécesseur)
-On place les successeur dans une pile, celui ayant la plus courte distance en premier
-On recommence avec le premier nœud de la pile, en l'en retirant

Algo de Floyd :
Pour k de 1 à n :
  Pour i de 1 à n :
    Pour j de 1 à n :
      lij = min(lij, lik + lkj)

Arbres et arborescences

Arbre : graphe connexe sans cycle (donc graphe simple) ⇔ une unique chaîne entre 2 sommets quelconques

Forêt : graphe non-connexe sans cycle

Arbre couvrant (ou maximum) : arbre incluant tous les sommets du graphe

Forêt maximale : ajout d'une arête créé forcément un cycle

Th : G possède n sommets, p compo connexe ⇔ forêt max de G contient n-p arcs

Th : T forêt max de G => T et G même nombre de connexité

Th : T forêt max de G => pour tout arc de G n'appartenant pas à T, il existe un unique cycle dont tous les arcs appartiennent à G. L'ensemble de ces cycles forme une base de cycles de G, de dimension le nombre cyclomatique de G

Arborescence : Graphe possédant un sommet racine relié à chaque autre sommet par un chemin unique (partant de lui)

Arborescence couvrante : idem que arbre couvrant mais avec une racine

Arbre couvrant de poids minimal : arbre de poids le plus petit parmi tous les arbres possibles (poids = somme de la longueur des arêtes)

Algo construction forêt maximale :
Les segments formants la forêt sont rouges (et forment Gr), les autres verts (le tout forme Gc, graphe "colorié")
-On examine chaque arc : si il connecte 2 sommets déjà connectés par des arcs rouges (et ferme donc un cycle), on le colorie en vert (nbr de connexité de Gc/Gr conservé)
-Sinon, on le colorie en rouge (nbr de connexité de Gr/Gc diminue de 1)

Algo de Prim :
-Choix de la racine (arbitraire)
-greffe de l'arête de plus faible poids qui permette de maintenir un état d'arbre
-Si connexe, arrêt sur arbre couvrant, sinon répéter sur chaque compo connexe pour créer forêt

Algo de Kruskal :
-On supprimer l'arête de plus petit poids, et on la note comme faisant partie de l'arbre
-On répète jusqu'à ce qu'il ne reste plus qu'un sommet

Réseaux, réseaux de transport et flots

Réseau : graphe fortement connexe, sans boucle, avec plus d'un sommet

Nœud : sommet avec plus de 2 arcs incidents (autres = anti-nœud)

Branche : chemin dont seules les extrémités sont des nœuds

Capacité de l'arc (noté C(a)) : flot maximal pouvant circuler dans l'arc a

Flot (noté phi(a)) : quantité transportée par chaque arc, flot entrant = flot sortant (loi de Kirchhoff), doit être < C(a)

Réseau de transport : graphe orienté, antisymétrique, sans boucle, possédant un sommet sans prédécesseur (entrée), un sans successeur (sortie), et au moins un chemin reliant les deux

Valeur du flot : flot juste après l'entrée ou juste avant la sortie

Arc saturé : Phi(a) = C(a)

Flot complet : si tout chemin reliant l'entrée à la sortie du réseau de transport contient au moins un arc saturé

Graphe d'écart (réseau résiduel) : graphe composé des arcs non saturés (de capacité égale au reste avant saturation) et des arcs inverses

Th : flot maximal ⇔ pas de chemin entre entrée et sortie dans le graphe d'écart

Algo de recherche flot complet :
- On cherche sur le graphe partiel des arcs non saturés un chemin allant d'entrée à sortie
- On augmente le flot de chacun de ses arcs de 1

Algo amélioration du flot :
- On marque "+" le premier sommet

Pour tout sommet x :
- On marque "+x" tout successeur de flot non-maximal
- On marque "-x" tout prédécesseur de flot non-nul

Si on arrive au dernier sommet, il existe une chaîne dont le flot peut être augmenté de 1

Algo de Ford-Fulkerson :
- On définit une flot phi^0 à (0, 0, …, 0)
- On cherche une chemin de l'entrée à la sortie sur le graphe résiduel
- On ajoute (ou soustrait) le flot minimal du chemin, puis on passe à phi^1, ainsi de suite jusqu'à plus de chemin possible à l'étape 2

Coût d'un arc (noté w(a)) : valeur réelle associée

Coût total d'un flot : somme des coûts des arcs

Algo de Busaker-Gowen :
Soit G' le graphe d'écart relatif à phi^k (associé aux w'(a) et c'(a), phi^k supposé de coût minimal parmi ceux de même valeur)
-si phi(a) < c(a) : w'(a) = w(a) et c'(a) = c(a) - phi(a)
-si phi(a) > 0 : w'(a) = -w(a) et c'(a) = phi(a)
Si µk est un chemin de coût minimal relativement aux coûts w' de entrée à sortie sur G', on note ek la capacité résiduelle de ce chemin.
On applique alors Ford-fulkerson pour trouver phi^{k+1} (= ek*µ)

Couplages

Graphe biparti : dont les sommets peuvent êtres classés en deux ensembles disjoints

Couplage : sous-ensemble d'arêtes non-adjacentes deux à deux

Sommet saturé par un couplage : possède une arête incidente appartenant au couplage

Couplage parfait : sature tous les sommets du graphe

Couplage maximal : de cardinalité maximale

Chaîne alternée (relativement à un couplage): chaîne élémentaire dont une arête sur 2 appartient au couplage

Chaîne alternée augmentante (ou améliorante) : joint deux sommets insaturés (se termine par des arêtes n'appartenant pas au couplage)

Transfert (le long d'une chaîne alternée) : inversion des arêtes couplées et non-couplées, donne encore un couplage.

Sommet pendant : sommet relié à un seul autre

Th : couplage est maximal ⇔ il n'existe pas de chaîne augmentante relative à ce couplage

Algo construction couplage max :
- Trouver une chaîne améliorante
- Transfert
- Garder le nouveau couplage obtenu et recommencer

Algo construction chaîne améliorante (par construction d'un arbre alterné) :
- racine = sommet insaturé
- On construit l'arbre en ajoutant les sommets adjacents reliés par une arête non-couplée, puis de même avec les arêtes couplées
- Qd on arrive à insérer un sommet insaturé, on a notre chaîne améliorante (si on finit l'arbre avant, c'est qu'il n'en existe pas d'autres

Algo construction couplage max (cas biparti) :
-créer un sommet relié à tous les sommets de d'une partie, idem pour la deuxième
-on cherche le flot maximal entre ces deux sommets virtuels (tous les arcs ont une capacité de 1

Poids d'une arête : valeur réelle associée (noté w)

Poids d'un couplage : somme des poids des arêtes couplées

Couplage de poids maximal : explicite

Chaîne alternée réductrice : inverse de chaîne améliorante (arêtes aux extrémités sont couplées)

Chaîne alternée conservative : une extrémité couplée, l'autre non

Coût réduit d'une chaîne alternée : poids des arêtes n'appartenant pas au couplage - poids des arêtes y appartenant (noté δ)

Cycle alterné pair : chaîne paire dont extrémités coïncident (transfert ne change pas sa cardinalité)

Th : coplage de poids max ⇔ existe pas de chaîne / cycle alterné pair de coût réduit > 0

Th : couplage de cardinalité p et chaîne améliorante de coût réduit maximal => transfert donne couplage de poids maximal parmi ceux de cardinalité p+1

Pb : n tâches i à réaliser par n machines j avec un coût Cij, trouver permutation avec coût minimal

Algo hongrois:
- sur matrice des coûts C : enlever à chaque élément de chaque colonne le plus petit élément de la colonne, puis faire de même pour chaque ligne
- pour la ligne ayant le moins de zéro : en choisir un, l'encadrer, et barrer tous les autres zéros de sa ligne ET de sa colonne. Répéter pour toutes les lignes
- on marque les lignes avec aucun zéro encadré, les colonnes avec un zéro barré appartenant à une ligne marquée et les lignes avec un zero encadré appartenant à une colonne marquée (faire tourner jusqu'à ce qu'il n'y ai plus de marquage possible)
- on barre les colonnes marquées et les lignes NON-marquées
- on cherche le plus petit élément non-barré
- on le soustrait aux colonnes NON-barrées, puis aux lignes barrées
- on répète la deuxième étape
- Les zéros encadrés représentent les cases de la matrice de départ à choisir pour la solution optimale

Problèmes d'ordonnancement

Pb : tâches en nombre défini, caractérisée par leur temps d'exécution, et liée par des contraintes de postérité

Graphe potentiel-tâche : tache i ⇔ sommet i / arc de i à j ⇔ i doit précéder i, longueur de l'arc ⇔ durée de tâche i / α et ω, sommets extrémités, tâches fictives de début et fin, de durée 0

Date au plus tôt (notée ti) : longueur du plus long chemin de α à i

Durée minimale du projet : Date au plus tôt de ω

Date au plus tard (notée Ti) : longueur du plus long chemin de i à ω

Marge totale (notée Mi) : Ti – ti
Tâche critique : tâche dont Mi = 0

Marge libre (mi) : délai possible sans décaler date au plus tôt des tâches suivantes

Graphe potentiel-étape (ou PERT) : tâche ⇔ arc / durée de tâche ⇔ longueur de l'arc / étape du projet ⇔ sommet / tâches doivent se suivre ⇔ arcs se suivent => possibilité d'ajouter des tâches fictives de durée 0 pour établir contrainte de postériorité

Ordonnancement : programme fixant les ti

Ordonnancement optimal : ordonnancement faisable et minimisant tω

Chemin critique : plus long chemin de α à ω

Durée minimale : longueur du chemin critique

Graphes planaires

Graphe planaire : graphe réalisable dans le plan (sans arcs qui se croisent)

Graphes isomorphes : identiques si on déforme le plan (de manière élastique)

Face : surface entourée par des arêtes

Frontière : ensembles des arêtes délimitant une face

Faces adjacentes : ont une arête commune dans leurs frontières

Contour (de face) : cycle élémentaire constitué de la frontière

Face infinie : face unique autour du graphe

Graphe déduit par contraction : obtenu par répétition de :
-suppression des sommets de degré 1
-remplacement des sommets de degré 2 par arêtes reliant ses 2 voisins

Th (Kuratowski) : graphe planaire ⇔ G n'admet pas de sous-graphe partiel se contractant en un graphe isomorphe à K5(pentagramme) ou K3,3 (6 sommets, chacuns reliés à 3 voisins)

Coloration d'un graphe

Coloration des sommets : affectation d'une couleur à un sommet, aucun sommet adjacent n'a la même

Coloration des arêtes : idem que sommets, mais avec arêtes

Graphe p-chromatique : p couleurs utilisée pour colorier le graphe

Nombre chromatique (noté γ) : nbr min de couleurs nécessaire à la coloration des sommets

Indice chromatique (noté q) : idem que nbr chromatique mais pour coloration des arêtes

Ensemble stable : ne contient que des sommets non-adjacents deux à deux (sommets de même couleur = ensemble stable)

Nombre de stabilité (noté α) : cardinal maximal d'un ensemble stable

Th : γ >= N / α, avec N nombre de sommets du graphe

Complément :
Pour un graphe à n sommets et m arêtes, on a :
γ(G) ≥n / (n-dmin) avec dmin degre minimum des sommets de G
γ(G) ≥ cardinal de la plus grande clique de G
γ(G) ≥n2 / (n2- 2m)
γ(G) £n + 1 -a(G) avec a(G) nombre de stabilite du graphe G
γ(G) £dmax + 1 avec dmax degre maximum des sommets de G

Algo de Welsh Powell :
-prendre la matrice d'adjacence du graphe, sommets rangés par ordre de degré décroissant
-colorier d'une couleur la première ligne non coloriée, et la colonne de même indice. On considère alors la matrice composée des lignes non-coloriées ayant un 0 dans les colonnes coloriées.
-recommencer avec la même couleur jusqu'à ce que matrice considérée vide, puis recommencer avec une autre couleur jusqu'à ce que tout soit coloré