Parcours génériques dans un graphe

Michel HABIB

26 janvier 2001

Chapitre 1

Introduction

On considère un graphe $G = (X, U)$ et l'on suppose les arcs munis d'étiquettes, i.e. une valuation $\omega : U \to T$, où T est un ensemble muni d'une relation d'ordre total $\ll$, et de deux éléments distingués $\top$ (resp. $\bot$) un unique élément maximal (resp. minimal).

On étend cette valuation aux chemins comme suit:

$\mu = [x_1, \ldots, x_k]$, $\omega(\mu) = \oplus_{i=0}^{i=k-1} \omega(x_i, x_{i+1})$.

La relation $\oplus$ binaire associative sur T, étant interprétée suivant les exmeples, comme:

- l'addition dans le cas usuel où T est l'ensemble des entiers ou des réels,
- le maximum
- un produit
- ...

La relation d'ordre total $\ll$ s'interprétant comme:

- l'ordre usuel quand T est l'ensemble des entiers,
- l'ordre lexicographique lorsque T est un langage,
- ...

1.1 L'algorithme

Algorithme de Parcours Générique
Données: un graphe orienté $G = (X, U)$, une fonction de coût $\omega : U \to T$
Résultat: une arborescence des plus courts chemins issus de x
$OUVERTS \leftarrow \{x\}$
$FERMES \leftarrow \emptyset$
$Parent(x) \leftarrow NIL$
$\forall y \neq x\ ,\ Parent(y) \leftarrow y$
$d(x) \leftarrow \perp$ (l'élément minmal de T)
$\forall y \neq x\ ,\ d(y) \leftarrow \top$ (l'élément maximal de T)
tant que $OUVERTS \neq \emptyset$ **faire**
  $z \leftarrow Choix(OUVERTS)$
  $Ajout(z, FERMES)$
  Explorer(z)
  $Virer(z, OUVERTS)$

Explorer(z)
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
    si $d(z) \oplus \omega(z, y) < d(y)$ **alors**
      $Parent(y) \leftarrow z$
      $d(y) \leftarrow d(z) \oplus \omega(z, y)$
  sinon
    $Ajout(y, OUVERTS)$
    $Parent(y) \leftarrow z$
    $d(y) \leftarrow d(z) \oplus \omega(z, y)$

À l'issue de l'algorithme on dispose d'une arborescence de racine x, définie à l'aide de la fonction Parent et de la fonction $d : X \to T$ qui associe à chaque sommet y, la distance d'un plus court chemin de x à y.

Pour mettre en œuvre cet algorithme, il nous faut étendre de manière naturelle la relation $\oplus$ à $T \cup \{\infty\}$.

1.2 Instanciations de cet algorithme

1.2.1 Algorithmes de plus courts chemins

Algorithme de Dijkstra

Si T est l'ensemble des réels positifs,
si $\ll$, resp. $\oplus$ sont respectivement l'ordre naturel et l'addition sur les réels,
et si la fonction choix fournit un sommet z vérifiant:
$d(z) = min_{y \in OUVERTS}\{d(y)\}$
l'algorithme devient alors, l'algorithme classique de Dijkstra.

L'algorithme A*

Si T est l'ensemble des réels positifs et
si $\ll$, resp. $\oplus$ sont respectivement l'ordre naturel et l'addition sur les réels,
et si la fonction choix fournit un sommet z vérifiant:
$d(z) + h(z) = min_{y \in OUVERTS}\{d(y) + h(y)\}$ où h(y) est une information "heuristique" sur la distance qui reste à parcourir.
et si l'instruction : "Ne rien faire" est remplacée par:

si $d(z) \oplus \omega(z, y) < d(y)$ **alors**
  $Parent(y) \leftarrow z$
  $d(y) \leftarrow d(z) \oplus \omega(z, y)$
  $Ajout(y, OUVERTS)$

Dans les deux instanciations précédentes, le goulot d'étranglement de complexité provient de la gestion de l'ensembles des OUVERTS. Il faut utiliser une bonne structure de données qui permette le calcul du minimum en $O(logn)$ ou mieux. Difficile à mettre en œuvre en Java ...

1.2.2 Autres Algorithmes

Parcours en largeur

Il suffit de mettre un coût constant unitaire sur les arcs et de prendre pour T l'ensemble des entiers positifs et de gérer les OUVERTS comme une file (premier

entré, premier sorti).

Parcours en largeur lexicographique

Il suffit de prendre $T = \{1, 2, \ldots, n\}^*$ (l'ensemble des mots construits avec les nombres 1,2 jusqu'à n), et d'interpréter l'ordre $\ll$ par l'ordre lexicographique sur ces mots.

La fonction Explorer devenant:

Explorer(z)
$numero(z) \rightarrow numerocourant - 1$
$numerocourant \rightarrow numerocourant - 1$
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
    $d(y) \rightarrow d(y) \bullet numero(z)$
  sinon
    $d(y) \rightarrow d(y) \bullet numero(z)$
    $Ajout(y, OUVERTS)$

le symbole "$\bullet$" représentant l'opération de concaténation, on étiquette ainsi les sommets du graphe avec des mots sur $\{1, 2, \ldots, n\}^*$.

La variable globale numerocourant doit être initialisée à n dans les initialisations de la procédure principale.

Cerises sur le gateau

Changeons la distance:
si T est l'ensemble des réels positifs,
si $\ll$, resp. $\oplus$ sont respectivement l'ordre naturel et le **maximum** sur les réels,
et si la fonction choix fournit un sommet z vérifiant:
$d(z) = min_{y \in OUVERTS}\{d(y)\}$
l'algorithme permet alors, de calculer l'arborescence des distances minimum (suivant la distance du max) issue de x.

On peut même obtenir un arbre de poids minimum avec cet algorithme, car l'algorithme de Prim rentre dans ce moule!