Projet de Modélisation

Problèmes de cheminement dans les graphes

Roméo Florian Webert Chris

Mars 2018

Table des matières

1 Problème

1.1 Données

On dispose d'un graphe orienté donné par l'utilisateur. Il peut le créer manuellement via l'interface en ligne de commande, ou bien donner un fichier *GPR*.

Chaque sommet est identifié par une chaîne de caractère unique et contient les bornes inférieures et supérieures de la fenêtre.

Chaque arête est également identifiée par une chaîne de caractère unique et contient deux valuations strictement positives. On considère que la première représente un coût et la seconde la durée.

1.2 Traitements

Le programme doit être capable de donner plusieurs informations sur le graphe :

- détecter l'existence d'un chemin quelconque entre deux sommets donnés ;
- détecter l'existence de circuits dans le graphe ;
- donner les composantes fortement connexes ;
- donner les numérotations préfixes, suffixes et l'ordre topologique de chaque sommet ;
- donner le meilleur chemin en fonction :
  - du meilleur coût ;
  - de la meilleure durée ;
  - du meilleur coût avec une durée limitée ;
  - de la meilleure durée avec un coût limité

La plupart des traitements ci-dessus doivent être visualisés graphiquement pour le confort de l'utilisateur.

2 Programmation

2.1 Algorithmes de traitement

Tous les algorithmes de traitement ont été implémentés de manière itérative, permettant de traiter des graphes beaucoup plus volumineux.

2.1.1 Parcours de graphe

Le parcours en profondeur *DFS (Depth First Search)* a été choisi ici. Il est utilisé pour la numérotation des sommets, l'existence de chemins entre deux sommets, la détection de circuits et des composantes fortement connexes.

2.1.2 Recherche de meilleur chemins

La recherche du meilleur chemin avec seulement le meilleur coût ou seulement le meilleur temps se fait à l'aide de l'algorithme de Dijkstra. Il a été choisi car les valuations des arcs sont strictement positives.

Cette recherche demande en paramètre une fonction externe qui renvoie la valuation à utiliser. Cela permet de ne pas avoir à dupliquer du code pour chaque nouvelle valuation ajoutée à un arc.

La recherche du meilleur chemin en tenant compte des fenêtres de temps utilise l'algorithme à correction d'étiquettes. Nous avons choisi cet algorithme car les corrigés des séances de TD nous ont permit de vérifier son bon fonctionnement.

2.1.3 Structure

Le diagramme de classe est visible sur la figure 2.1 page 5. La classe *Search* est la base de tout parcours itératif. Elle contient une file des prochains sommets à explorer et appelle les classes filles pour chaque sommet a explorer.

La classe *DisconnectedGraphSearch* permet de parcourir des graphes non connexes. Si demandé, elle vérifiera à la fin du parcours si tous les sommets ont été fermés. Si ce n'est pas le cas, un nouveau parcours sera lancé sur le premier sommet non fermé trouvé.

Les classes *DFS*, *PCC*, *PCCFT* implémentent respectivement les algorithmes de parcours en profondeur, de Dijkstra, et à correction d'étiquette.

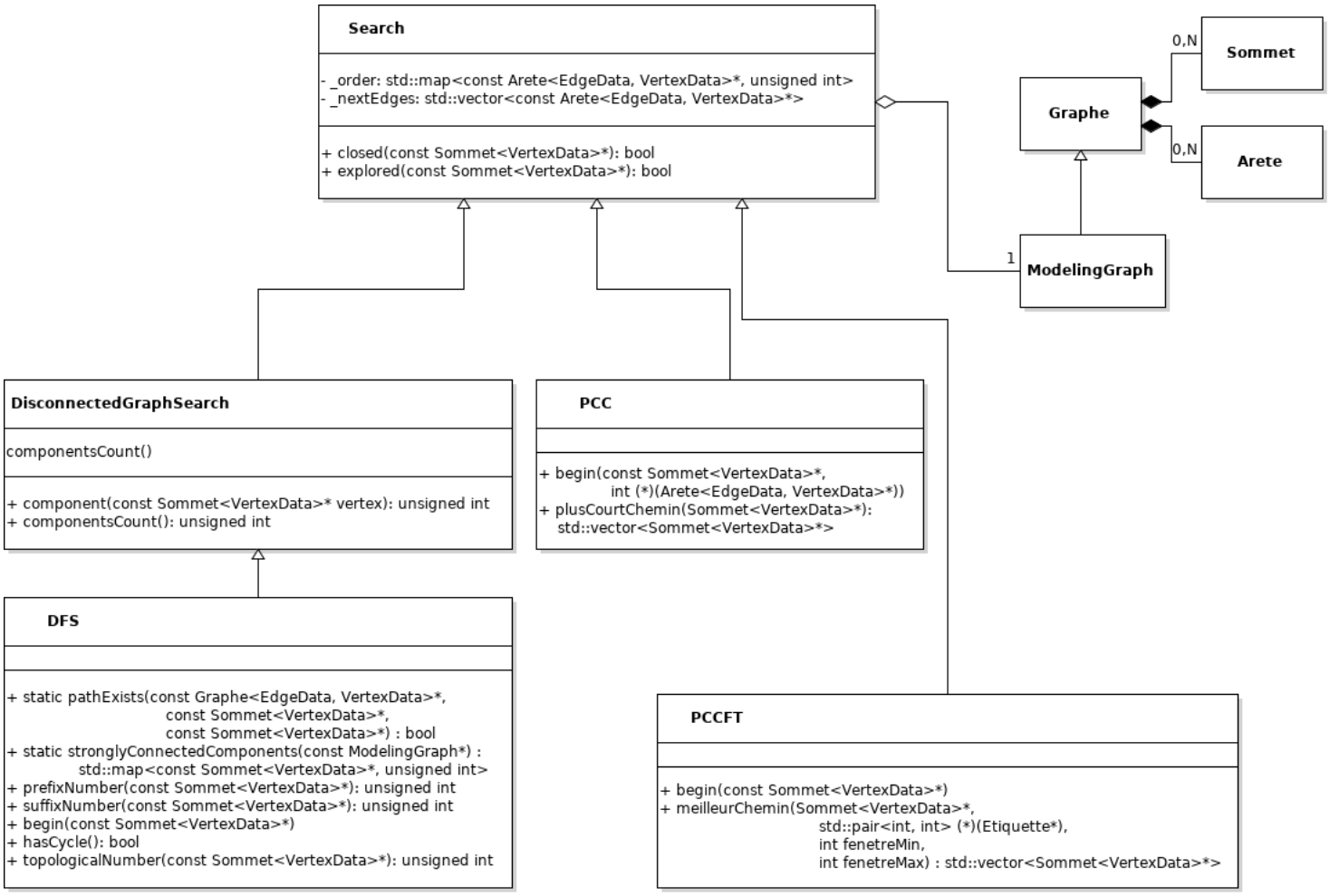

FIGURE 2.1 – Diagramme de classes des algorithmes

2.2 Lecture de fichiers GPR

Le diagramme de classe est visible sur la figure 2.2 page 6. Les fichiers *GPR* sont lus par la classe *GPRParser*, associée à une chaîne de responsabilité. Chaque ligne sera lue par *GPRParser*, puis envoyée à la chaîne de responsabilité qui la traitera, si possible.

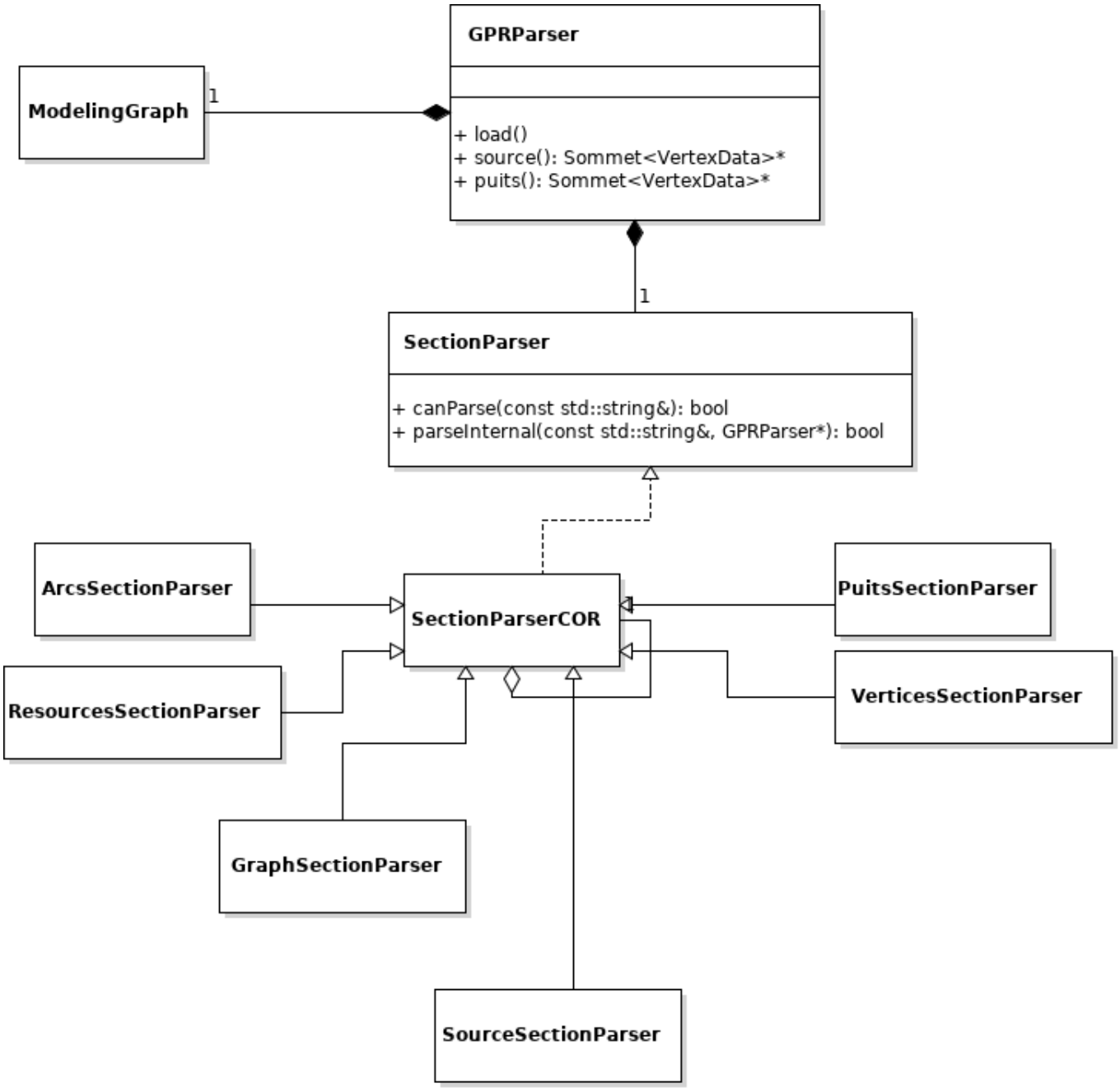

FIGURE 2.2 – Diagramme de classes du lecteur *GPR*

2.3 Interface utilisateur

L'interface utilisateur a été conçue de façon à ne pas avoir de dépendances sur le projet. Elle utilise la console pour interagir avec l'utilisateur, et crée des fichiers au format *DOT*[1].

1. `http://www.graphviz.org/pdf/dotguide.pdf`

3 Résultats et performances obtenus

3.1 Performances

Instance *(nb sommets/nb arcs)*	10/36	20/122	50/1086	100/1694	160/2815
Ouverture (μs)	430	1011	5576	8860	13114
Visualisation (μs)	524	990	6737	8815	13235
Parcours DFS (μs)	319	493	2469	3460	9261
Detection CFC (μs)	447	1205	7052	13196	23009
PCC (μs)	113	429	11503	18236	157206
PPC-FT (μs)	122	1068	27489	21389	42926

TABLE 3.1 – Résultats des instances *GPR* fournies

Tous les parcours DFS partent de la source, et les recherches du meilleur chemin se font de la source au puits.

3.2 Résultats graphiques

3.2.1 DFS

Le rendu est visible sur la figure 3.1 page 8.

Chaque nœud contient :

— son identifiant ;

— P : son numéro préfixe ;

— S : son numéro suffixe

Les arcs parcourus sont pleinement affichés avec leur ordre de parcours.

Les nœuds $T_1 \dots T_n$ correspondent à l'ordre topologique et sont alignés horizontalement avec le nœud du graphe correspondant.

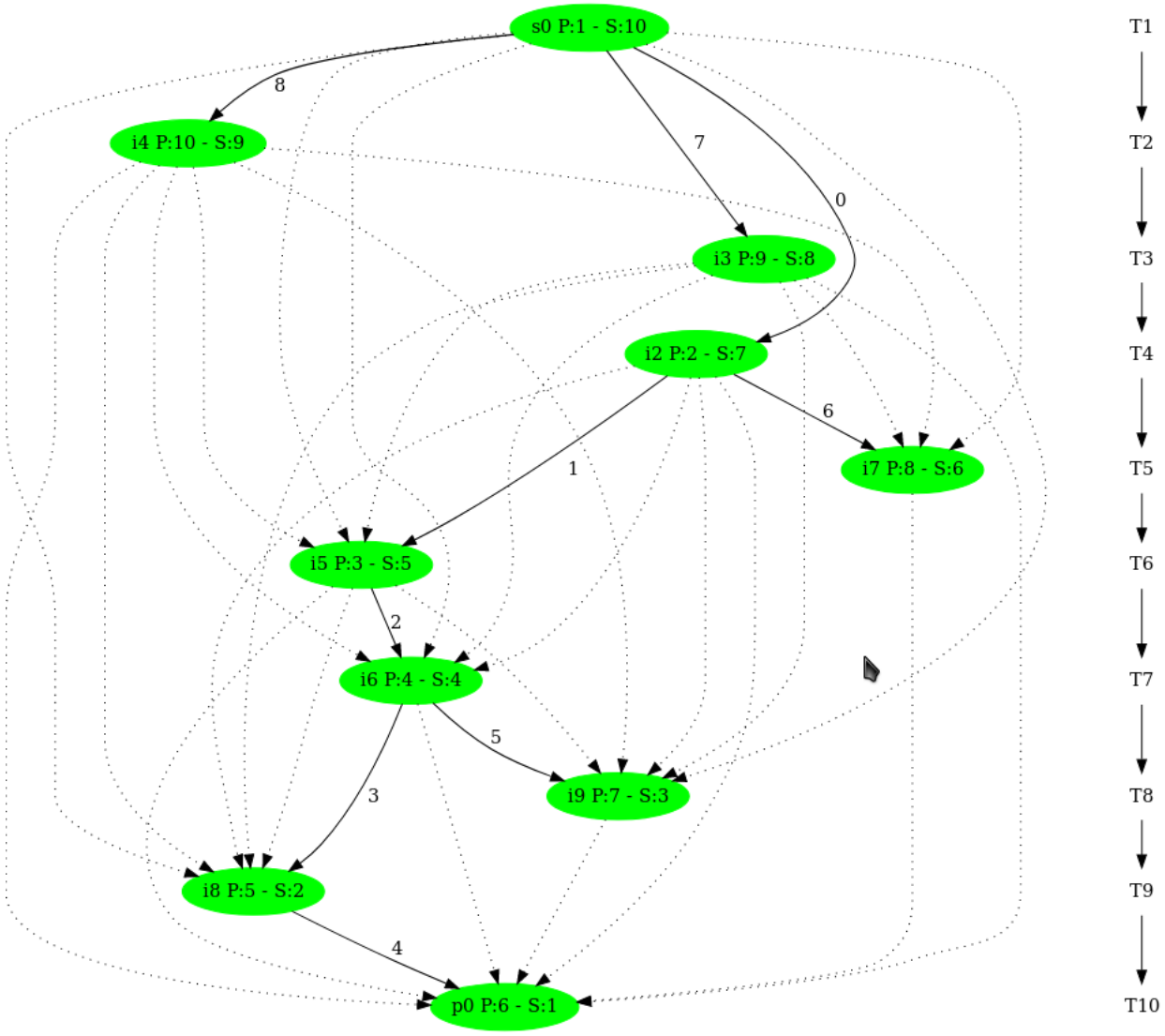

FIGURE 3.1 – Rendu du parcours *DFS* de `data_VRPTW_10.gpr`

3.2.2 Composantes fortement connexes

Le rendu est visible sur la figure 3.2 page 9.

Chaque nœud contient son identifiant, et est inclus dans le cadre correspondant à une composante fortement connexe.

Les arcs sont affichés pleinement s'ils font partie de la même composante connexe, en pointillés sinon.

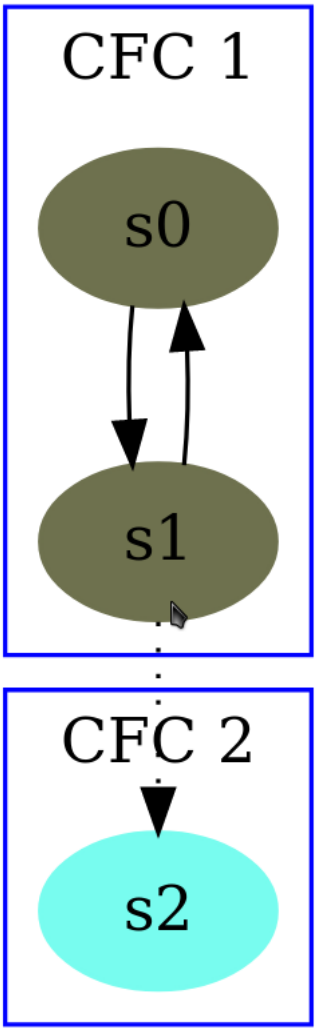

FIGURE 3.2 – Rendu de la recherche de composantes fortement connexes dans `data_scc.gpr`

3.2.3 Plus court chemin

Le rendu est visible sur la figure 3.3 page 10.

Chaque nœud contient son identifiant, et est affiché en vert s'il fait partie du plus court chemin. Les arcs en faisant partie sont pleinement affichés.

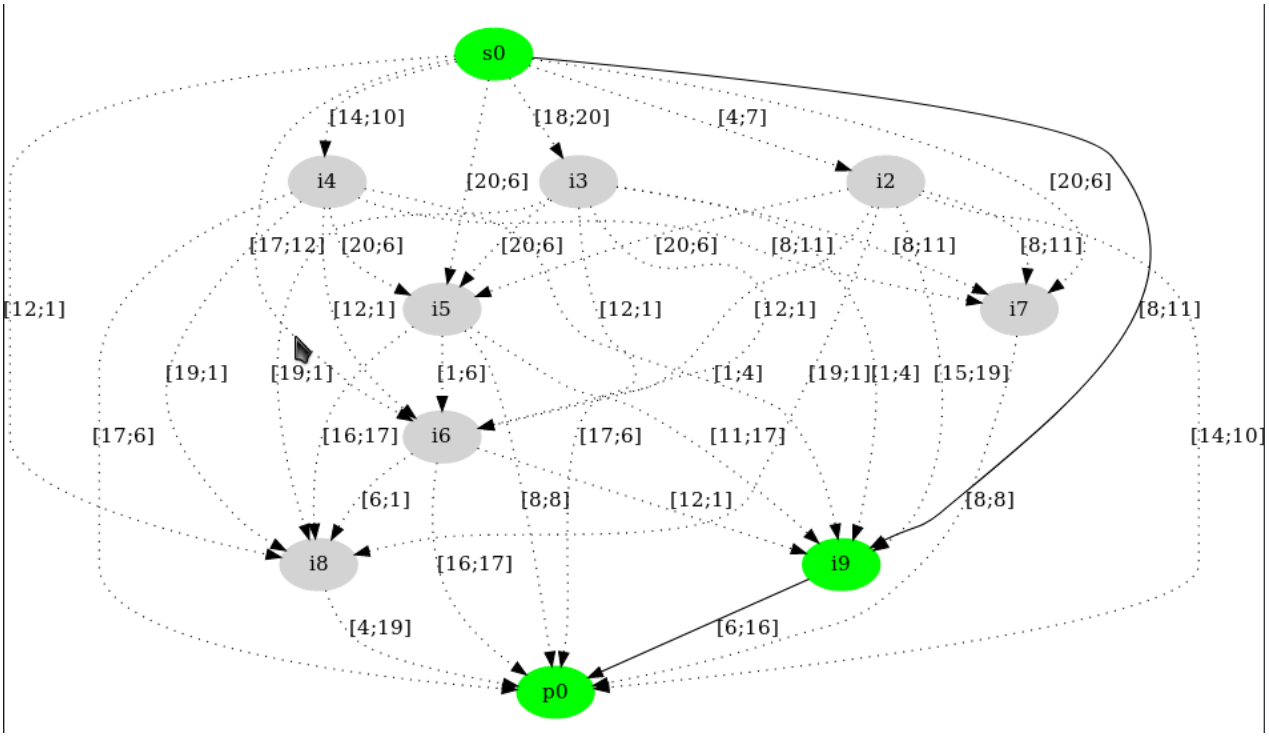

FIGURE 3.3 – Rendu de la recherche du plus court chemin de `data_VRPTW_10.gpr`