École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
3e année
2010 - 2011

Rapport Projet Algorithme-C

Tournée d'un véhicule multicritéres

Encadrants
Emmanuel Neron
emmanuel.neron@univ-tours.fr

Université François-Rabelais, Tours

Étudiants
Cyrille PICARD
cyrille.picard@etu.univ-tours.fr
Michael PURET
michael.puret@etu.univ-tours.fr

DI3 2010 - 2011

Version du 4 juin 2011

Table des matières

1 Introduction **6**

I Spécification **7**

2 Cahier des charges **8**

2.1 Besoins 8

3 Modélisation du problème **9**

3.1 Variables 9

3.2 Contraintes 9

3.3 Fonctions Objectifs 9

4 Spécification **10**

4.1 Délimitation système/environnement 10

4.2 Définition du programme à réaliser 11

4.3 Lien entre les différentes partie 11

II Algorithimique **12**

5 Réalisation de la structure de données **13**

5.1 Introduction 13

5.2 Informations utile 13

5.3 disposition en mémoire 13

6 Algo de tri rapide multicritères **15**

7 Conclusion **18**

Annexes **20**

A Fiche de suivi de projet **20**

Table des figures

1.1 Représentation simplifiée de la configuration 6

4.1 Schéma représentant la délimitation système/environnement 10

Liste des tableaux

1. Introduction

Pour ce projet nous intéressons à réaliser un programme qui permet en fonction d'une liste de lieux de ressortir la liste des différents parcours possible pour réaliser cette tournée entre les différents lieux. Ce problème est un peu similaire à celui du voyageur de commerce. Le principe de l'application qu'on essaye de développer est de permettre à partir d'une Base de données représentant la configuration d'une ville. C'est à dire un ensemble de lieux relié entre par des arcs, de calculer un parcours pour passer par tout les lieux en fonction de leur intérêt pour l'utilisateur. La difficulté majeur est de retourner des solutions dans une période très courte.

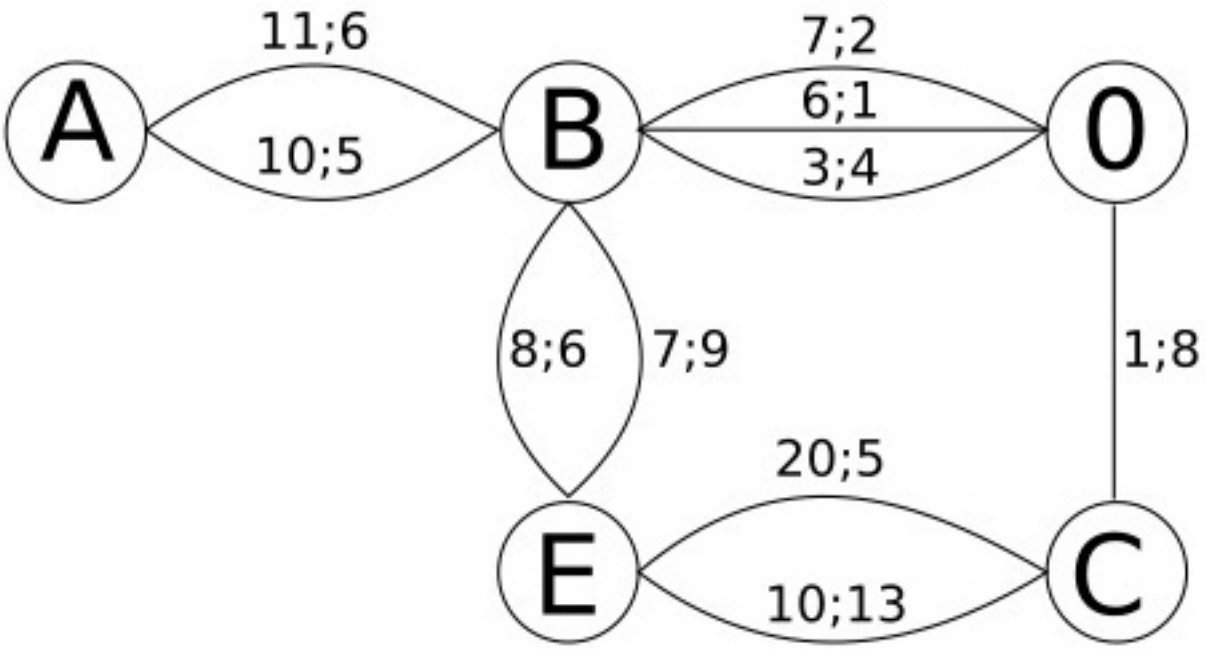

Figure 1.1 – Représentation simplifiée de la configuration

Première partie

Spécification

2. Cahier des charges

Suite à la première rencontre avec notre encadrant un premier cahier des charges a été évoquer. Le logiciel doit retourner la liste des différents parcours possible pour visiter une série de lieux dans l'ordre désiré par l'utilisateur. Le but est de calculer un parcours qui minimise l'insécurité et la distance à parcourir tout en maximisant l'intérêt des lieux.

2.1 Besoins

- Le programme devra retourner une solution de parcours viable rapidement,
- Le logiciel permettra de retourner une nouvelle solution si on interverti des lieux ou/et si on change l'ordre des lieux à visiter.
- Le programme doit être simple à utiliser
- Le programme peut retourner un certain nombre de parcours calculé dans le temps d'exécution impartie.

3. Modélisation du problème

3.1 Variables

Il y a des paramètres sur la configuration de la ville qui vont être importants à prendre en compte car ils vont influencer les résultats ce sont les variables. Un variable qui dépend de l'utilisateur c'est l'intérêt des différends lieux de la ville. Par exemple pour un même lieu deux utilisateurs vont pas lui attribuer obligatoirement le même intérêt. Les autres variables dont il faut tenir compte sont les caractéristiques des arcs qui sont la distance et l'insécurité.

3.2 Contraintes

La principale contrainte de ce problème est la liaisons entre les différents lieux, c'est à dire si il existe un ou plusieurs arc entre deux lieux pour pouvoir aller d'un lieu dit source à un lieu dit destination. Si on reprend la configuration présenter dans la figure 1.1 on voit que pour aller du lieu de départ (0) au lieu A ou E il faut passer par le lieu B. En d'autre termes pour réaliser le parcourt on peut se déplacer d'un lieu à un autre si il existe au moins un arc reliant les deux lieux en question.

3.3 Fonctions Objectifs

Pour notre problème on peu considérer plusieurs fonctions objectifs qui sont les suivantes :
1re fonction : Il faut que l'ensemble des lieux soit visités en essayant de maximiser l'intérêt
2e fonction : Le parcourt doit minimiser la distance
3e fonction : Le parcourt doit aussi minimiser l'insécurité

4. Spécification

Maintenant que nous avons réaliser la définition et la modélisation du problème a été effectué, on va chercher à spécifier le programme qui permettra de répondre au problème et à définir de manière concise la structure général de ce programme.

4.1 Délimitation système/environnement

On peu représenter cette délimitation à l'aide du schéma ci-dessous

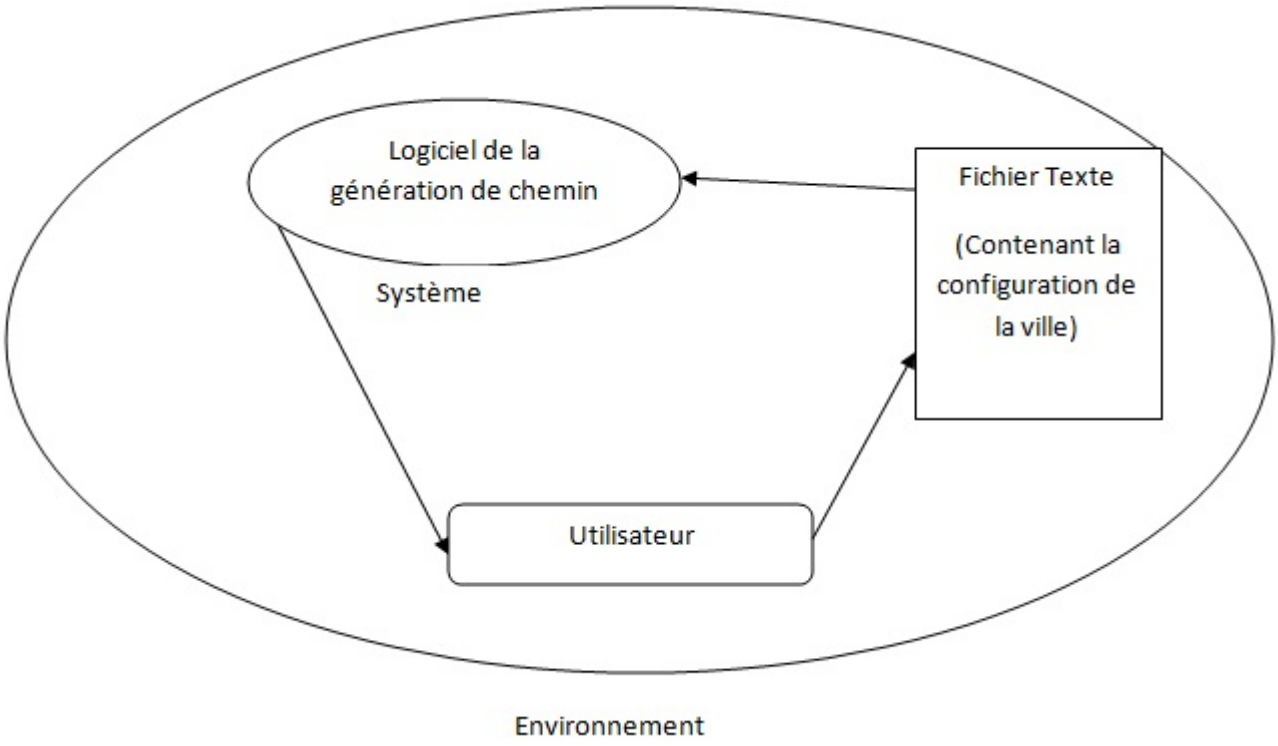

FIGURE 4.1 – Schéma représentant la délimitation système/environnement

En plus de distinguer le système de l'environnement le schéma précèdent nous permet de voir en même temps les flux d'informations entre les différentes identités de l'environnement.

- La flèche entre l'utilisateur et le fichier texte représente le processus par lequel l'utilisateur rentre la configuration de la ville, c'est à dire la position des lieux avec leur intérêt pour l'utilisateur et les arcs avec leurs caractéristiques.
- Le fichier texte une fois remplis par l'utilisateur sert d'entré au programme pour créer la structure de données

– Une fois que le programme a finis de tourner il retourne à l'utilisateur la liste des parcours possible par rapport à la ville qui a été rentré dans la fichier texte.

4.2 Définition du programme à réaliser

Pour réaliser un programme qui permet de répondre au mieux au besoins nous avons séparer le problème en 4-5 partie :

1. Pour gérer la configuration des lieux qui provient du fichier texte prévu comme entrée, il faut un ensemble de structure de données qui permet de stocker et d'utiliser les différents paramètre lier à la configuration, c'est à dire : l'intérêt des lieux, l'intercommunication entre les différents lieux, les arcs qui permette cette communication et les caractéristiques de ces arc (Distance, Insécurité).
2. Générer un chemin de base
3. Générer de nouveaux chemin en réalisant des permutations dans le chemin de base en fonction des permutations possible entre les lieux.
4. Avoir un structure permettant de retourner les chemins générer

On peut représenter de manière simplifier la structure générale du programme qui répondra au problème posé :

4.3 Lien entre les différentes partie

diagramme de classe

Deuxième partie

Algorithimique

5. Réalisation de la structure de données

5.1 Introduction

5.2 Informations utile

Le solveur reçoit en paramètre un document texte découpé en trois parties :
- Parametres (le nombre de lieux à visiter ; les temps de recherche en seconde ; le nombre de lieux ; le nombre d'arcs)
- Lieux (identifiant ; intérêt ; nom du lieu)
- Arcs (identifiant du lieu de départ ; identifiant du lieu d'arrivée ; distance ; insécurité)

voici un exemple de ville : donner le .txt correspondant au graphe

5.3 disposition en mémoire

L'ensemble des données sont stockées dans un jeu de structures et de tableaux alloués sur le tas.

Il existe une structure principal nome "donnée" qui contient l'ensemble des informations.

Donnee :
- temps_execution : Entier ; *temps imparti pour la recherche de solutions.*
- nb_lieux_total : Entier ; *nombre de lieux intéressant.*
- nb_arcs : Entier ; *nombre d'arcs total.*
- *table_interet : Interet_lieu ; *tableau des intérêts décroissant avec les correspondances des lieux.*
- *lieux : Lieu ; *pointeur sur un tableau contenant l'ensemble des références des lieux.*
- ***index_lieu : Index_arc ; *pointeur sur une table d'index pour l'utilisation de map.*
- ***map : Arc ; *pointeur sur un tableau contenant pour chaque lieu l'ensemble des arcs disponible. Ces arcs sont triés par internet décroissant, distance et insécurité croissante.*
- *solution : Caracteristique ; *pointeur sur un tableau contenant l'ensemble des solutions acceptable.*

L'ensemble des lieux et de leurs informations sont contenus dans un tableau à une dimension de type "Lieu" :

Lieu :
- id : Entier ; *numéro du lieu.*
- interet : Entier ; *intérêt du lieu.*
- *nom : Chaine de caractéres ; *nom du lieu.*
- nb_arc : Entier ; *nombre d'arcs sortant.*

Les arcs sont alloués sur le tas indépendamment les uns des autres, la structure qui les définis est du type "Arc" :

Arc :

– distance : Entier ; *distance de l'arc.*
– insecurite : Entier ; *insécurité de l'arc.*
– *depart : Lieu ; *lieu de départ de l'arc.*
– *destination : Lieu ; *lieu de destination de l'arc.*

C'est pour se déplacer d'un lieu à un autre que le tableau à trois dimensions "map" à été mi en place. Sur sa première dimension, on retrouve le lieu de départ, sa deuxième dimension contient un tableau de pointeurs sur les arcs alloués sur le tas.

Donner le tableau en fonction de l'exemple.

Comme il existe un nombre variable d'arcs menant à la même destination pour un même point de départ, le tableau carré "index_lieu" de type "index_arc" se charge de faire la correspondance entre le départ et la destination. La structure index_arc contient deux entiers ; id_arc est l'indice du premier arc disponible, nb_arc indique combien d'arcs sont disponibles.

Index_arc :

– id_arc : Entier ; *identifiant de l'arc*
– nb_arc : Entier ; *nombre d'arcs disponible*

Donner le tableau en fonction de l'exemple.

Le chemin de référence est construit de manière à maximiser dès le départ un des trois critères. De ce fait une table "liste_lieu", de type "index_lieu" contiens les lieux triés en fonction des choix de l'utilisateur.**à faire**. Ainsi le constructeur de chemins de référence n'aura plus qu'à utiliser cette liste sans faire de recherche. Donner le tableau en fonction de l'exemple.

Le dernier tableau contenu par la structure de donnée est celui des solutions, il grandit avec l'avancement du programme. Il est nommé "solution" et est de type parcours. Il contient une structure de type "caractéristique" qui définit le chemin, un tableau trajet pointant les arcs utilisés et un tableau itinéraire qui stocke les lieux visités.

Parcourt :

– *carac : Caracteristique ; *pointeur sur les caractéristiques de cette solution.*
– *trajet : Arc ; *tableau contenant les arcs utilisés dans cette solution.*
– itineraire : interet_lieu ; *tableau contenant les lieux déjà visités et le nombre de fois.*

6. Algo de tri rapide multicritères

Algorithme 1 Swap

Précondition:

Entrée : X, Y pointeur sur des Arcs

$XetY$ sont des pointeurs valides

Postcondition:

Sortie :$\emptyset$

$X = Y'$ et $Y{=}X'$

1: Allouer(tmp)
2: tmp $\leftarrow \uparrow$X
3: $\uparrow$X $\leftarrow \uparrow$Y
4: $\uparrow$Y $\leftarrow$ tmp

Algorithme 2 Quicksort_map

Précondition:

Entrée :

- data pointeur sur la structure Donner
- id_Lieu entier, identifiant du lieu a trier
- m entier borne droit du tri
- n entier borne gauche du tri

Toutes les données sont valides

Postcondition:

Sortie : $\emptyset$

Postcondition : data → map[id_lieu] est trier par :

- IntérÕt décroissant
- Distance croissant
- Insécurité croissant

```
1:  si (m < n) alors
2:      k ← (m+n)/2
3:      swap (map[m],map[n])
4:      i ← m + 1
5:      j ← n
6:      tantque i ≤ j faire
7:          tantque ((i ≤ n) & position(data, id_lieu, i, m) faire
8:              i ← i + 1
9:          fin tantque
10:         tantque ((j ≤ n) & position(data, id_lieu, j, m) faire
11:             j ← j + 1
12:         fin tantque
13:         si (i < j) alors
14:             swap (map[i],map[j])
15:         fin si
16:     fin tantque
17:     swap(map[m],map[j])
18:     quicksort_map(data,id_lieu,m,j-1)
19:     quicksort_map(data,id_lieu,j+1,n)
20: fin si
```

Algorithme 3 Position

Précondition:

Entrée :

- data pointeur sur la structure donnée
- id_lieu entier, identifiant du lieu a trier
- id_arc entier, identifiant de l'arc
- id_key entier, identifiant de la clef

Toutes les données doivent Õtre valides

Postcondition:

Sortie : posi est un booléen

- posi = 0 : id_arc avant id_key
- posi = 1 : id_arc après id_key

```
1: Key_interet ← interet_destination (data,id_lieu,id_key)
2: Key_distance ← distance_arc (data,id_lieu,id_key)
3: Key_insecurite ← insecurite_arc (data,id_lieu,id_key)
4: arc_interet ← insecurite_arc (data,id_lieu,id_arc)
5: arc_distance ← insecurite_arc (data,id_lieu,id_arc)
6: arc_insecurite ← insecurite_arc (data,id_lieu,id_arc)
7: posi = 1
8: si (arc_interet > key_interet) alors
9:         posi = 0
10: fin si
11: si (posi ≠ 0) et (arc_interet = key_interet) alors
12:         si (arc_distance < key_distance) alors
13:                 posi = 0
14:         fin si
15:         si (posi ≠ 0) et (arc_distance = key_distance) alors
16:                 si (arc_insecurite ≤ key_insecurite) alors
17:                         posi = 0
18:                 fin si
19:         fin si
20: fin si
21: retourner(posi)
```

7. Conclusion

Annexes

A. Fiche de suivi de projet

17/01/2011	1ére rencontre avec Emmanuel Néron pour prendre une explication approfondi du sujet ainsi que le premier objectif à réaliser qui est le choix d'une structure de donnée pour gérer les villes.

20/02/2011 au 04/03/2011	Réflexion sur les méthodes possible pour la structure de donnée à mettre en place, ainsi que les algorithmes à utiliser pour parcourir les différentes structures de données

04/03/2011 au 29/03/2011	Réalisation de la structure de données pour gérer la configuration de la ville, mise en place d'algorithme de tri pour les arcs.

07/04/2011	Finalisation de la structure de données, vérification des fonctions permettant d'interroger la structure de données pour les algorithmes permettant de créé les trajets, première réalisation d'un algorithme pour générer le trajet de référence.

11/04/2011	Réunion avec notre encadrant pour lui présenter la structure de données et l'esquisse de l'algorithme pour générer le trajets de référence.

11/04/2011 au 05/05/2011	Réalisation et implémentation de l'algorithme pour générer un chemin de base et résolution d'un problème par rapport au tri des arc entre les lieux.

05/05/2011 au 01/06/2011	validation du planning, installation et évaluation du logiciel XX, essais sur le jeu de données fourni, présentation orale des résultats validation du planning, installation et évaluation du logiciel XX, essais sur le jeu de données fourni, présentation orale des résultats.

Tournée d'un véhicule multicritéres

Département Informatique
3e année
2010 - 2011

Rapport Projet Algorithme-C

Résumé : Description en français

Mots clefs : Mots clés français

Abstract: Description en anglais

Keywords: Mots clés en anglais

Encadrants
Emmanuel Neron
emmanuel.neron@univ-tours.fr

Université François-Rabelais, Tours

Étudiants
Cyrille PICARD
cyrille.picard@etu.univ-tours.fr
Michael PURET
michael.puret@etu.univ-tours.fr

DI3 2010 - 2011