



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](http://www.polytech.univ-tours.fr)

**Département Informatique**  
**5<sup>e</sup> année**  
**2009 - 2010**

**Projet de Fin d'Etude**



**Nouveaux critères pour du calcul  
d'itinéraire vélo**

**Encadrants**

Gaël Sauvanet  
[gael.sauvanet@univ-tours.fr](mailto:gael.sauvanet@univ-tours.fr)  
Emmanuel Néron  
[emmanuel.neron@univ-tours.fr](mailto:emmanuel.neron@univ-tours.fr)

**Etudiant**

Ke SHANG  
[ke.shang@etu.univ-tours.fr](mailto:ke.shang@etu.univ-tours.fr)  
DI5 2009 - 2010

Université François-Rabelais, Tours

Version du 18 avril 2010



# Table des matières

---

<b>1 Présentation du projet</b>	<b>10</b>
1.1 Contexte . . . . .	10
1.2 Description de l'existant . . . . .	10
1.3 Objectifs du projet . . . . .	12
1.4 Définition des Besoins . . . . .	12
1.4.1 Besoins fonctionnels . . . . .	12
1.4.2 Besoins non fonctionnels . . . . .	14
1.5 Environnement du projet . . . . .	14
1.6 Présentation du contexte technique . . . . .	14
1.6.1 Client . . . . .	15
1.6.2 Serveur . . . . .	15
1.6.3 XML-RPC . . . . .	15
1.6.4 Base de données . . . . .	15
<b>2 Méthodes</b>	<b>16</b>
2.1 Présentation du problème . . . . .	16
2.2 Méthodes utilisées . . . . .	17
2.2.1 Algorithme de Dijkstra . . . . .	17
2.2.2 Optimisation multi-objectif . . . . .	19
2.2.3 Algorithme de BCA* . . . . .	21
<b>3 Réalisation des besoins</b>	<b>26</b>
3.1 Implémentation de l'algorithme BCA* . . . . .	27
3.1.1 Structure de données . . . . .	27
3.1.2 Réalisation . . . . .	28
3.2 Implémentation des nouveaux critères . . . . .	33
3.2.1 Structure de la base de données . . . . .	33
3.2.2 Réalisation de critère d'effort . . . . .	36
3.2.3 Réalisation de critère de sécurité . . . . .	39
3.3 Amélioration . . . . .	39
3.3.1 Amélioration de l'algorithme . . . . .	39
3.3.2 Amélioration de temps du calcul . . . . .	41
3.4 Modification de l'interface . . . . .	44
<b>4 Conclusion</b>	<b>46</b>
<b>A Résultat de tests de valeur <math>\varepsilon</math> différent</b>	<b>47</b>
<b>B Guide - Installation et configuration</b>	<b>50</b>
B.1 Installation des packages nécessaires . . . . .	50
B.2 Configuration de la base de données . . . . .	50
B.3 Configuration du site web Géovélo . . . . .	52
B.4 Configuration de LEDA . . . . .	52



<b>C Cahier des Charges - Introduction</b>	<b>56</b>
<b>D Contexte de la réalisation</b>	<b>57</b>
D.1 Contexte . . . . .	57
D.2 Objectifs . . . . .	57
<b>E Description générale</b>	<b>58</b>
E.1 Environnement du projet . . . . .	58
E.2 Caractéristiques des utilisateurs . . . . .	58
E.3 Fonctionnalités et structure générale du système . . . . .	59
E.4 Contraintes de développement, d'exploitation et de maintenance . . . . .	60
<b>F Architecture générale du système</b>	<b>62</b>
<b>G Description des Besoins</b>	<b>64</b>
G.1 Besoins fonctionnels . . . . .	64
G.2 Besoins non fonctionnels . . . . .	65
<b>H Découpage du projet en tâches</b>	<b>66</b>
H.1 Tâche 1 : La familiarisation avec les algorithmes . . . . .	66
H.2 Tâche 2 : La familiarisation de le fonctionnement de Géovélo . . . . .	66
H.3 Tâche 3 : Configuration de l'environnement . . . . .	67
H.4 Tâche 4 : Rédaction du guide . . . . .	67
H.5 Tâche 5 : Étude de l'existant . . . . .	67
H.6 Tâche 6 : Implémentation du nouvel algorithme . . . . .	67
H.7 Tâche 7 : Prise en main PostGIS . . . . .	67
H.8 Tâche 8 : Modéliser les critères . . . . .	68
H.9 Tâche 9 : Implémentation des nouveaux critères . . . . .	68
H.10 Tâche 10 : Prise en main AJAX . . . . .	68
H.11 Tâche 11 : Modification de l'interface du site web . . . . .	68
H.12 Tâche 12 : Rédaction du rapport . . . . .	69
<b>I Planning</b>	<b>70</b>
<b>Bibliographie</b>	<b>73</b>

# Table des figures

---

1.1	Le site Géovélo . . . . .	10
1.2	Le diagramme de cas d'utilisation de Géovélo . . . . .	11
1.3	Le modèle de vos préférencea . . . . .	14
2.1	Solutions entre les critères . . . . .	16
2.2	Exemple de l'algorithme de Dijkstra . . . . .	18
2.3	Résultat d'algorithme de Dijkstra . . . . .	19
2.4	Pareto frontier . . . . .	20
2.5	Relation de dominance . . . . .	21
2.6	Un graphe à traiter . . . . .	22
2.7	Calcul de Heristique . . . . .	23
2.8	Distance de Tchebycheff . . . . .	23
2.9	Algorithme de BCA* . . . . .	24
2.10	Calcul de valeur de chaque critère . . . . .	24
3.1	Architecture principale de Géovélo . . . . .	26
3.2	Tâches de chaque partie . . . . .	27
3.3	Etapes pour implémenter BCA* . . . . .	30
3.4	Zone originale de vos préférences . . . . .	31
3.5	Architecture du site Géovélo . . . . .	32
3.6	Zone modifiée de vos préférences . . . . .	33
3.7	Cas particuliers . . . . .	33
3.8	Calcul d'effort . . . . .	37
3.9	Zone d'effort . . . . .	38
3.10	Amélioration de l'algorithme . . . . .	40
3.11	Exemple d'un itinéraire . . . . .	41
3.12	Interface originale . . . . .	44
3.13	Interface améliorée . . . . .	45
B.1	Ajouter un serveur dans l'outil pgAdmin . . . . .	51
B.2	Linker settings dans l'outil codeblocs . . . . .	53
B.3	Search directories dans l'outil codeblocs . . . . .	53
B.4	Complier and debugger settings . . . . .	54
B.5	ServerXmlRpc . . . . .	54
E.1	Le site Géovélo . . . . .	59
E.2	Le diagrammes de cas d'utilisation de Géovélo . . . . .	60
F.1	L'architecture principale de Géovélo . . . . .	62
F.2	L'architecture du site Géovélo . . . . .	63
G.1	Le modèle de vos préférence . . . . .	64
I.1	Diagramme de gantt du projet . . . . .	71



I.2    Diagramme de gantt du projet . . . . .	72
---	----

# Liste des tableaux

---

1.1	Le composant de l'effort . . . . .	13
2.1	Exemple de bi-critère . . . . .	17
2.2	Le résultat de calcul . . . . .	18
3.1	Domaine de $\varepsilon$ . . . . .	42
3.2	Tests d'itinéraire différent . . . . .	43
A.1	Tests de critère de distance . . . . .	47
A.2	Tests de critère de sécurité . . . . .	48
A.3	Tests de critère d'effort . . . . .	49

# Remerciements

---

Je souhaite tout d'abord remercier beaucoup Gaël Sauvanet pour m'avoir donné toutes les informations nécessaires et tous les conseils. Il m'a aidé tout au long du projet de fin d'étude. Son point de vue m'a été d'une grande utilité pour la compréhension du fonctionnement du système existant. Il m'a expliqué clairement et m'a guidé pour les réflexions concernant les algorithmes.

Un merci à Emmanuel Néron pour ses instructions et ses connaissances. Il m'a proposé des bonnes idées concernant ce projet.

Enfin, merci à tous ceux qui m'ont aidé durant la période du projet de fin d'étude.

# Introduction

---

Lors de la cinquième année du cycle d'ingénieur de l'École Polytechnique de l'Université de Tours Département Informatique. Nous devons effectuer un Projet de Fin d'Étude (PFE) du 10/2009 au 5/2010. Le but est de familiariser les étudiants avec les compétences acquises en cours et d'en acquérir de nouvelles.

Le sujet du projet de fin d'étude est "Nouveaux critères pour du calcul d'itinéraire vélo". Ce projet est basé sur des travaux de mon encadrant Gaël Sauvanet et l'étudiante Juan Zhang dans le cadre du projet Géovélo réalisé par l'association Autour du Train. L'objectif de ce projet de mettre en place un nouvel algorithme et implémenter des nouveaux critères.

Dans ce rapport, je vais d'abord présenter mon projet. Et puis je vais détailler les méthodes et les algorithmes utilisées, la spécification de la base de données. Enfin ce sera la réalisation de ce projet de chaque étape, je vais aussi parler les difficultés rencontrées.

# Présentation du projet

## 1.1 Contexte

Géovélo est un site qui permet de calculer des itinéraires adaptés au réseau cyclable. Nous pouvons choisir le départ et la destination. Entre les deux points Géovélo nous fournit des itinéraires en tenant compte de différents critères. L'utilisateur peut choisir un itinéraire selon ses préférences de distance et de sécurité. Géovélo est une première en France. L'interface cartographique est basée sur API GOOGLE MAPS. Pour l'instant, il n'est disponible que pour la ville de Tours.

## 1.2 Description de l'existant

Pour l'instant, le site Géovélo fonctionne bien pour trouver l'itinéraire en fonction du point de départ et un point de destination en considérant deux critères (distance et sécurité). Il utilise une version multi-critère de DIJKSTRA ALGORITHM pour calculer l'itinéraire.

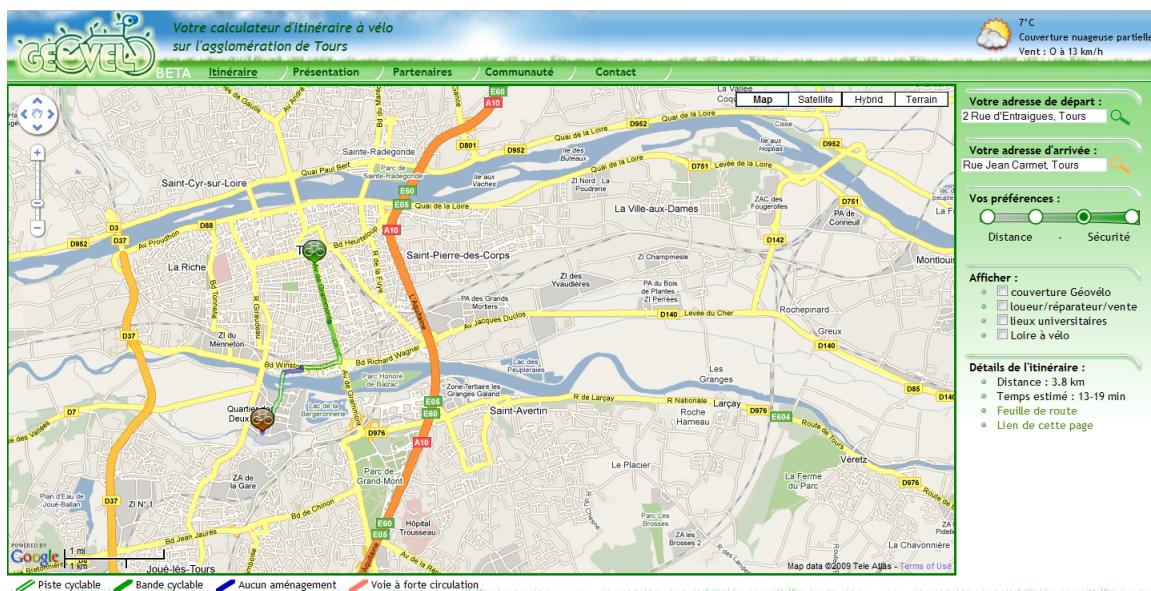


FIGURE 1.1 – Le site Géovélo

L'image ci-dessus est triée du site web Géovélo actuel. Sur ce site, les fonctionnalités suivantes sont possibles :

### Sélection d'un itinéraire

Pour saisir un itinéraire, il existe 3 méthodes :

- Saisir une adresse de départ ou d'arrivée puis cliquer sur la loupe ;
- Déplacer les curseurs de départ et d'arrivée directement sur la carte à l'aide de la souris ;
- Utiliser le clic droit de la souris sur la carte.

## Sélection d'une préférence

Permet de choisir un itinéraire selon ses préférences, de distance et de sécurité.

## Affichage d'informations géographiques

- Couverture Géovélo : Afficher un cadre qui permet de visualiser la zone géographique couverte par Géovélo ;
- Loueur/réparateur/vente : Afficher l'ensemble des professionnels du vélo ;
- Lieux universitaires : Afficher les principaux lieux universitaires sur l'agglomération. On peut cliquer sur chaque point d'intérêt pour obtenir des informations supplémentaires ;
- Loire à vélo : Afficher le tracé de la Loire à vélo.

## Affichage des détails de l'itinéraire

Après avoir saisi le départ et la destination, Géovélo nous donne la solution avec la distance et le temps estimé. Nous pouvons cliquer sur la feuille de route pour obtenir des informations détaillées. Nous pouvons cliquer sur "lien de cette page" pour obtenir un lien vers un itinéraire de manière à pouvoir le partager par mail ou autre.

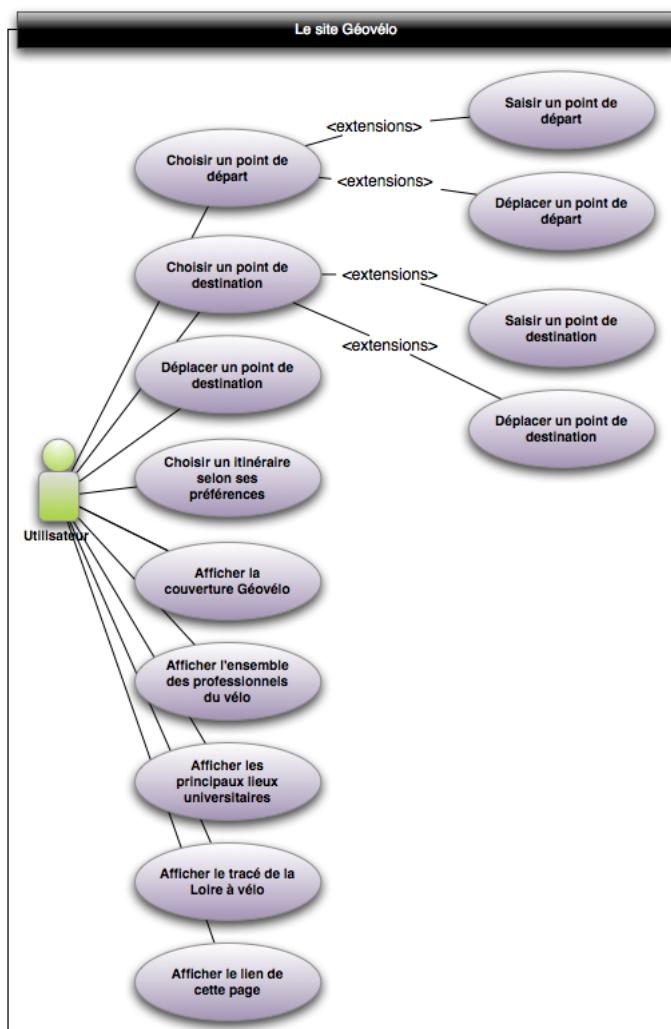


FIGURE 1.2 – Le diagramme de cas d'utilisation de Géovélo

## 1.3 Objectifs du projet

Pour les objectifs de mon projet de fin d'étude, il contient deux missions principalement :

### - Mettre en place un nouvel algorithme BCA\*

L'algorithme de ce projet est développé en utilisant le plus court chemin (une version multi-critère de l'algorithme de Dijkstra) [Dij59]. Évidemment cet algorithme fonctionne bien mais il perd beaucoup de temps du calcul.

Pour l'améliorer mon encadrant propose un nouveau algorithme : BEST COMPROMISE SOLUTION A\* (BCA\* - une généralisation d'A\* pour la recherche de solutions de compromis dans des problèmes d'optimisation multi-objectifs) [FP00]. C'est un algorithme qui est spécialement conçu pour la recherche de solutions de compromis. L'intérêt est que nous n'avons pas besoin de calculer tous les résultats, le mécanisme de BCA\* permet d'enlever des points qui ne nous intéressent pas pour trouver une solution plus rapidement en réduisant l'espace de recherche.

### - Implémenter des nouveaux critères

Pour l'instant, le site Géovélo fonctionne bien en considérant seulement deux critères (distance et sécurité). L'objectif de ce projet est d'implémenter les nouveaux critères qui contiennent trois critères principaux (effort, sécurité, tourisme) :

**Le critère d'effort** Il existe de nombreuses définitions de l'effort, l'idée à retenir est que nous cherchons à évaluer la difficulté d'un trajet. Nous pouvons prendre en compte plusieurs critères comme la pente, la distance, la météo... Il y a aussi tout l'aspect psychologique : la difficulté n'est pas la même suivant les personnes. Par exemple, si quelqu'un est motivé, il pourra surmonter plus facilement une côte. Il est important de prendre en compte l'effort car c'est un élément de l'information que nous le devons fournir à un cycliste. Nous voulons favoriser ce mode de transport, il faut donc lui apporter de nombreux services. Connaître le parcours le plus simple ou "mesurer" la difficulté de son parcours est donc utile pour un cycliste.

**Le critère de sécurité** Le critère de sécurité consiste à prendre en compte l'itinéraire en différents sécurités. Il permet l'utilisateur de choisir l'itinéraire conseillé pour le vélo.

**Le critère de tourisme** Le critère de tourisme consiste à trouver un itinéraire. Dans cet itinéraire, l'ensemble de points contiennent le point de départ, les points d'intérêts tels que les musées, les jardins, les monuments... et le point de destination. Un touriste devra commencer par le point de départ jusqu'au point de destination en passant obligatoirement par tous les points d'intérêts sélectionnés.

Selon le temps restant, nous devrons aussi modifier l'interface du site Géovélo.

## 1.4 Définition des Besoins

### 1.4.1 Besoins fonctionnels

Nous avons identifié les besoins fonctionnels suivants. Ces besoins identifient le travail à réaliser dans le cadre du projet.

- Besoin 1 : Mettre en place BCA\* en C++ sous la forme d'un web service ;  
Il consiste à lire d'abord des sources et comprendre la fonctionnalité d'algorithme existant, et puis implémenter BCA\* pour deux critères (distance et sécurité), enfin tester la fonctionnalité et la compatibilité ;

BCA\* est une méthode multi-objective. Cet algorithme prend en paramètre les préférences de l'utilisateur (poids sur chaque critère), après il retourne l'itinéraire le plus adapté aux préférences de l'utilisateur. Cet algorithme explore en priorité les labels les plus "prometteurs" et éliminent ceux qui ne sont pas intéressants.

- Besoin 2 : Implémenter des nouveaux critères ;

Cette tâche consiste à modéliser de nouveaux critères (effort, sécurité, tourisme) en collaboration avec 3 étudiants du DA dont leur sujet porte sur l'effort, la sécurité et le tourisme :

### **Le critère d'effort**

Nous pouvons voir la figure ci-dessous fournie par Simon Laporte (un étudiant du DA), le critère d'effort contient beaucoup de paramètres que nous devons considérer, cette tâche sera divisée en deux parties. Une partie pour Simon, il devra modéliser ce critère et essayer de résumer d'une formule permettant de considérer tous les paramètres. Et pour moi, je devrai implémenter cette formule et afficher la valeur d'effort calculé par cette formule. Nous nous répéterons jusqu'à ce que nous trouverons la formule optimale.

Sujet	Thèmes	Critères	Sous-critères
Efforts	Tâche	Motif	Domicile-Travail Loisir Sport Objectifs Distance Dénivelé
			Vitesse
		Climat	Température Hygrométrie Vent
	Environnement	Revêtement	
		Intersections	Nombre
		Matériel	Poids Roues
		Participants	
Individus	Physique		
			Age Sexe Aptitudes
	Psychologique		Difficulté perçue
			Effort consenti

TABLE 1.1 – Le composant de l'effort

### **Le critère de sécurité**

Le site Géovélo existe déjà le critère de sécurité, il suffit de récupérer d'abord les données améliorées par l'étudiant du DA, et puis nous devrons mettre à jour la base de données.

Quand nous allons finir implémenter des nouveaux critères. Nous pourrons utiliser BCA\* pour tester la fonctionnalité et la compatibilité.

- Besoin 3 : Modifier l'interface du site web.

Nous allons implémenter un nouvel algorithme et ajouter les nouveaux critères pour le site Géovélo, donc il failliras aussi utiliser le langage PHP/XAJAX/CSS pour modifier la zone de vos préférences.

Le modèle est comme la figure ci-dessous :

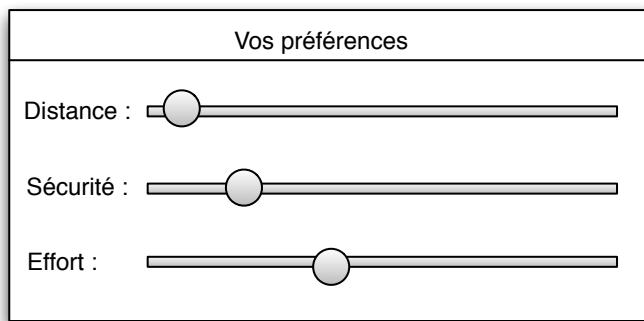


FIGURE 1.3 – Le modèle de vos préférences

Parce que nous utilisons nouvel algorithme pour calculer l'itinéraire, donc il faut modifier un peu sur le site Géovélo. Nous devrons d'abord ajouter notre nouveau critère pour la zone de vos préférences. Ensuite le modèle de sélection n'est pas très évident pour l'utilisateur, mon encadrant propose une idée en utilisant Slider à la place.

#### 1.4.2 Besoins non fonctionnels

- Besoin 4 : En terme de fiabilité, le site Géovélo devra être stable et devra supporter une charge d'environ 5000 visiteurs/mois ;
- Besoin 5 : La documentation d'installation et de configuration doit être rédigée ;
- Besoin 6 : Les commentaires du code source doivent être lisibles et compréhensibles ;
- Besoin 7 : Le temps de calcul d'un itinéraire doit être inférieur à 3 secondes ;
- Besoin 8 : L'interface du site Géovélo doit être conviviale et ergonomique.

### 1.5 Environnement du projet

Ce PFE est effectué sous le système d'exploitation UBUNTU8.

Le codage doit être réalisé en C++. Sous UBUNTU8, nous installons l'outil CODEBLOCKS comme environnement de développement et les paquets concernant la librairie de communication XMLRPC.

Nous utilisons POSTGRESQL comme SGBD et son module géographique POSTGIS. Nous utilisons l'outil PGADMIN3 pour gérer la base de données et l'outil qgis pour visualiser et traiter les données géographiques.

Nous utilisons le serveur APACHE2 et son module PHP5 comme serveur web.

### 1.6 Présentation du contexte technique

Pendant mon projet de fin d'étude, il est nécessaire d'utiliser différents langages et outils. Pour la partie de calcul d'itinéraires, nous utilisons C++ et LEDA. Pour la partie de site Géovélo, nous utilisons HTML, CSS, JAVASCRIPT, PHP, XAJAX, API GOOGLE MAPS. Pour la partie de base de données, nous utilisons POSTGRESQL et POSTGIS. Je vais les diviser en quatre parties (Client, Serveur, XMLRPC, Base de données) pour expliquer leur utilisations et fonctionnalités.

### 1.6.1 Client

- **AJAX** : Une technique de développement web. Il permet de créer des applications web interactives. Le plus gros avantage est qu'avec l'utilisation de AJAX, il est possible de maintenir des données sans besoin de rafraîchir la page toute entière. Les applications web peuvent rendre une réponse à l'utilisateur plus rapidement et éviter d'envoyer les informations qui n'ont pas besoin d'être modifiées.
- **XAJAX** : Un framework open source utilisant le langage PHP, il permet d'utiliser le langage PHP pour développer des fonctionnalités AJAX sans connaître le langage JavaScript.
- **API GOOGLE MAPS** : Nous savons que Google Maps est un outil utilisé pour suivre l'emplacement actuel, les conditions en temps réel du trafic, mais il offre aussi un outil de recherche, et l'API Google Maps permet d'intégrer Google Maps dans son propre site en utilisant JAVASCRIPT. Ensuite il est possible de développer avec JAVASCRIPT pour manipuler la carte.

### 1.6.2 Serveur

- **LEDA** : Une librairie C++ qui offre une grande variété d'algorithmes et de structures adaptés aux graphes et aux calculs géométriques. Dans mon projet il est nécessaire de calculer un itinéraire sur la carte, donc c'est plus facile de calculer l'itinéraire en utilisant cette librairie existante.

### 1.6.3 XML-RPC

Un protocole RPC (Remote procedure call), un ensemble de spécifications qui permettent à des applications tournant dans des environnements différents de communiquer facile. Cette technique qui permet à deux machines de communiquer ou d'échanger des données au format XML. Il est utilisé pour échanger des informations entre des pages du site et le serveur de calcul d'itinéraire. Nous pouvons l'utiliser pour transmettre des structures de données complexes. Il existe huit types de données : array, base64, boolean, date/time, double, integer, string, nil.

### 1.6.4 Base de données

- **POSTGRESQL** : Un système de gestion de base de données relationnelles. Actuellement, il est un des systèmes les plus puissants, le plus riche en fonctionnalités et le plus complexe des bases de données du logiciel libre. Il fonctionne principalement sous le système UNIX (Linux, FreeBSD, Ubuntu...).
- **POSTGIS** : Une extension qui ajoute la capacité de gestion des données spatiales sur le système PostgreSQL. Il offre des fonctionnalités comme les objets spatiaux, l'indexation spatial, les fonctions de manipulation spatiales et les opérateurs. En résumé il permet de manipuler des données géographiques facilement.

# Méthodes

---

Pour calculer l'itinéraire, nous devons utiliser plusieurs méthodes et algorithmes. Dans ce chapitre, nous allons détailler les méthodes et algorithmes que nous utilisons.

## 2.1 Présentation du problème

Il existe déjà beaucoup d'études concernant des problèmes de plus court chemin avec l'objectif unique en utilisant DIJKSTRA ALGORITHME. L'objectif est de trouver un chemin en minimisant la somme du coût de ses arcs entre le départ et la destination. Mais l'optimisation d'un objectif unique n'est pas suffisante, car il existe plusieurs objectifs que nous devons considérer dans le problème véritable.

Par exemple, l'objectif de site Géovélo ne consiste pas seulement à trouver un itinéraire plus court, il faut considérer plusieurs critères (distance, sécurité, effort, tourisme) en même temps. Les différents critères sont toujours indépendants sur chaque itinéraire, il n'existe pas de relation entre eux, donc nous ne pouvons pas considérer qu'une critère. Géovélo permet de fournir la solution bien équilibrée en considérant le poids de chaque critère que l'utilisateur choisit.

Par exemple, nous pouvons considérer que deux critères (distance et sécurité) :

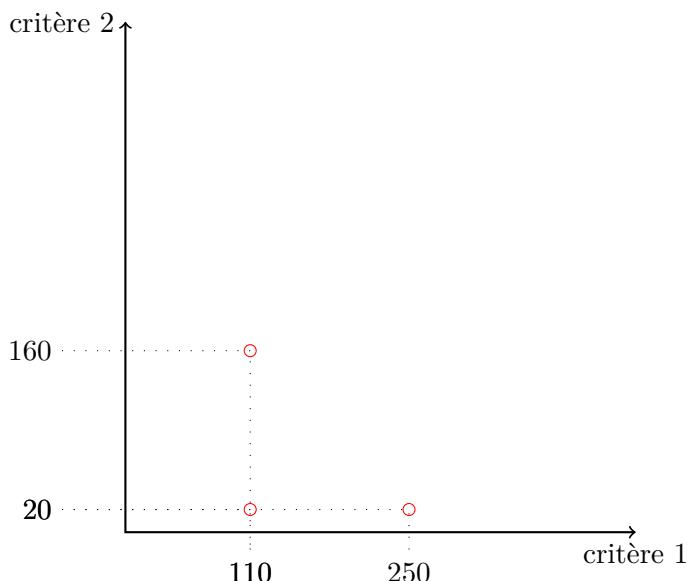


FIGURE 2.1 – Solutions entre les critères

Solution	Distance	Insécurité
1	400	50
2	300	100
3	250	150
4	200	200
5	100	250

TABLE 2.1 – Exemple de bi-critère

Nous pouvons voir l'exemple ci-dessus, évidement nous ne pouvons pas déterminer laquelle est la meilleure. Ca dépend la sélection de l'utilisateur. Ce type de problème est appelé **MULTIOBJECTIVE PROBLEM** [Wika].

Pour l'instant, nous pouvons utiliser le calcul d'itinéraire bi-critère existant qui se base sur DIJKSTRA ALGORITHM pour calculer un itinéraire intermédiaire entre deux points adjacents. Cet algorithme est différent de l'algorithme classique DIJKSTRA ALGORITHMME. Nous obtenons qu'une solution concernant le plus court chemin en utilisant l'algorithme classique. Mais cet algorithme existant nous fournit plusieurs solutions intermédiaires entre deux points adjacents. Chaque solution est bien représentée par un couple distance, insécurité correspondant aux deux critères.

Pour calculer un itinéraire entre le départ et la destination selon différents critères, la mécanisme de l'algorithme existant est de trouver toutes les solutions au premier, et puis parmi les résultats il retient quelques itinéraires satisfaisants les besoins de l'utilisateur. Evidement, cet algorithme fonctionne bien, mais toutes les solutions de calcul pourra prendre beaucoup de temps, de plus elles ne sont pas forcément utile si l'utilisateur a besoin d'obtenir qu'une solution de compromis.

## 2.2 Méthodes utilisées

Dans cette partie, je vais détailler les différentes méthodes utilisées comprenant DIJKSTRA ALGORITHMME, BCA\*, l'optimisation multi-objectif.

### 2.2.1 Algorithme de Dijkstra

DIJKSTRA ALGORITHMME a été publie par l'informaticien néerlandais Edsger Dijkstra en 1959. Il sert à résoudre le problème du plus court chemin. Cet algorithme permet de trouver une chaîne de longueur minimale entre deux sommets d'un graphe orienté ou non. Dans ce projet, nous pouvons utiliser cet algorithme pour trouver le plus court chemin de chaque critère.

#### Principe de l'algorithme de Dijkstra

Nous considérons un graphe  $G = [x, U]$  qui contient éventuellement des circuits et où toutes les longueurs des arcs sont positives :  $\forall u \in U, l(u) \geq 0$  ;

Nous cherchons le plus court chemin du sommet 1 à tous les autres ;

Nous notons  $\Pi(i)$  : un potentiel associé au sommet i, qui représente la longueur du plus court chemin de 1 à i actuelle ;

Nous notons  $\bar{\Pi}(i)$  : le potentiel final associé à i, donc la longueur du plus court chemin de 1 à i ;

Nous notons  $S$  : l'ensemble du sommets pour lesquels nous connaissons le potentiel final.

### Algorithm 1 DIJKSTRA

1.  $\bar{\Pi}(1) = 0$   
 $\forall u/(1, j) \in U, \Pi(j) = l(1, j)$   
 $\forall u/(1, j) \notin U, \Pi(j) = \infty$   
 $S = 1$
2. TQ  $S \neq X$  Faire  
 Sélectionner  $k \in X \setminus S / \min_{i \in X \setminus S} \Pi(i)$   
 $\bar{\Pi}(k) = \Pi(k)$   
Pour tout  $l \in X \setminus S, (k, l) \in U$  Faire  
 $\Pi(l) = \min(\pi(l), \Pi(k) + l(k, l))$   
FinPour  
 $S \leftarrow S \cup k$   
FTQ

### Exemple de l'algorithme de Dijkstra

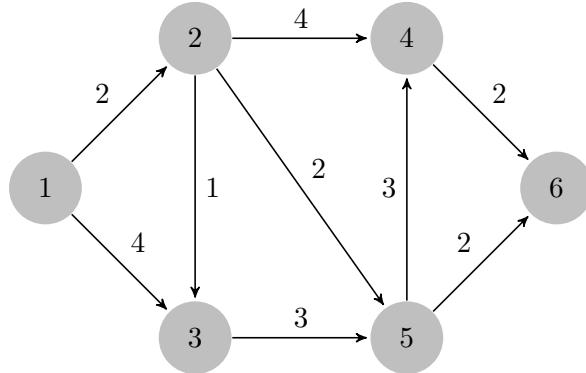


FIGURE 2.2 – Exemple de l'algorithme de Dijkstra

Nous pouvons utiliser la table ci-dessous pour montrer le résultat du calcul.

	1	2	3	4	5	6
$\Pi(0)$	0	2	4	$+\infty$	$+\infty$	$+\infty$
$\Pi(1)$		2	3	6	4	$+\infty$
$\Pi(2)$			3	6	4	$+\infty$
$\Pi(3)$				6	4	6
$\Pi(4)$				6		6
$\Pi(5)$						6

TABLE 2.2 – Le résultat de calcul

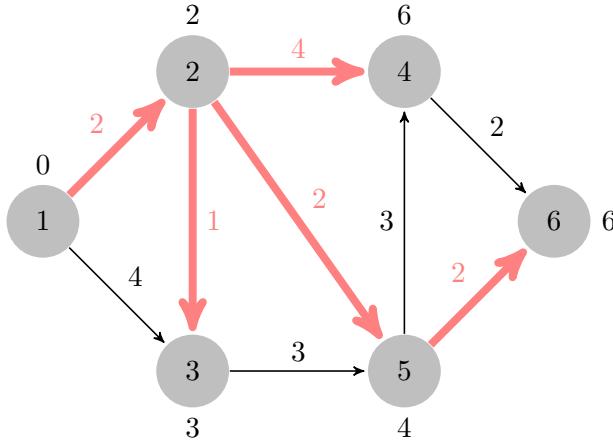


FIGURE 2.3 – Résultat d’algorithme de Dijkstra

Nous pouvons trouver les plus courts chemins entre le sommet de départ (sommet 0) et des autres sommets.

### 2.2.2 Optimisation multi-objectif

L’optimisation multi-objectif est une branche de l’optimisation combinatoire, elle est utilisée fréquemment dans nombreux secteurs de l’industrie concernés (Télécommunications, Transport, Environnement, Mécanique, Aéronautique, ...). Comme le suggère le nom, un problème d’optimisation multi-critère consiste à optimiser plusieurs fonctions objectif simultanément. C’est difficile de résumer des objectif en un objectif unique, car il existe des conflits entre ces objectif.

En général, Les objectifs de l’optimisation sont contradictoires, c'est-à-dire que l'objectif est de trouver des bons compromis parmi toutes les solutions. Ces problèmes peuvent être NP-complets.

Dans la partie suivante, je vais présenter la définition mathématique d’un problème d’optimisation multi-objectif.

#### Définition : Problème multi-objectif

Etant donné  $X = (x_1, x_2, \dots, x_n)$  est le vecteur des variables de décision. Il satisfait les contraintes ci-dessous :

$$g_i(X) \geq 0, i = 1, 2, \dots, k \quad (1)$$

$$h_i(X) = 0, i = 1, 2, \dots, l \quad (2)$$

Supposons r objectifs, et tous ces objectif sont conflictuels, la fonction optimale est :

$$f(X) = (f_1(X), f_2(X), \dots, f_r(X))$$

Trouver  $X^* = (x_1^*, x_2^*, \dots, x_n^*)$ , optimiser  $f(X^*)$  sous les contraintes (1) et (2). Les solutions qu’on obtient sont les solutions PARETO EFFICIENCY. [Wikib]

#### Définition : Pareto efficiency

Pareto efficiency est un concept en économie. Ce terme est nommé d’après Vilfredo Pareto, un économiste italien qui a utilisé le concept dans ses études d’efficacité économique et la répartition des revenus. Un optimum de Pareto est un état dans lequel nous ne pouvons pas améliorer le bien-être d’un individu sans détériorer d’un autre.

Nous pouvons dire que le vecteur  $X^* \in F$  est un optimum de Pareto, si pour chaque  $X \in F$ , soit  $\forall i \in I(f_i(X) = f_i(X^*))$ ,  $I = \{1, 2, \dots, r\}$ , soit  $\exists j \in I, f_j(X) > f_j(X^*)$ .

Sachant que  $F = \{X \in R^n | g_i(X) \geq 0, i = 1, 2, \dots, k; h_j(X) = 0, j = 1, 2, \dots, l\}$

En général, l'optimisation multi-objectif donne un ensemble de solutions, plus d'une solution. Dans un ensemble de solutions P, on sélectionne un ensemble de solutions non dominées. Dans l'espace de recherche, les solutions non dominées sont les solutions Pareto optimal. L'ensemble de solutions non dominées forme PARETO FRONTIER.

L'objectif d'optimisation multi-objectif est de trouver un ensemble de solutions non dominées et proche du PARETO FRONTIER.

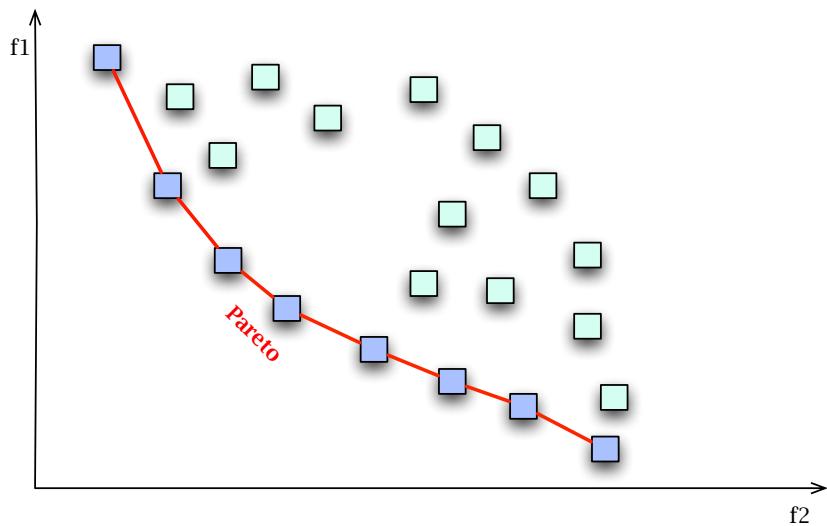


FIGURE 2.4 – Pareto frontier

### Définition : Relation de dominance

P est un ensemble de solutions, il existe k objectifs :  $f_1(), f_2(), \dots, f_k()$ . Pour chaque individu dans l'ensemble P. Il y a trois relations entre les deux solutions x et y. Pour x et y, soit x domine y, soit y domine x, soit x et y ne se dominent pas leur même. Nous pouvons donc définir la relation ci-dessous :

#### – La dominance :

Une solution x et une solution y appartiennent à l'ensemble P, x domine y si et seulement si :  $\forall i \in \{1, 2, \dots, k\} : f_i(x) \leq f_i(y)$  et  $\exists j \in \{1, 2, \dots, k\} : f_j(x) < f_j(y)$ . Nous notons  $y \prec x$ .

Si la solution x domine la solution y, nous disons que x est la solution non dominée et y est la solution dominée par x. Le symbole  $\prec$  représente la relation dominée.

#### – Non relation :

Une solution x et une solution y appartiennent à l'ensemble P, nous disons que la solution x et la solution y sont non relation si et seulement s'il n'existe pas la relation dominée entre elles.

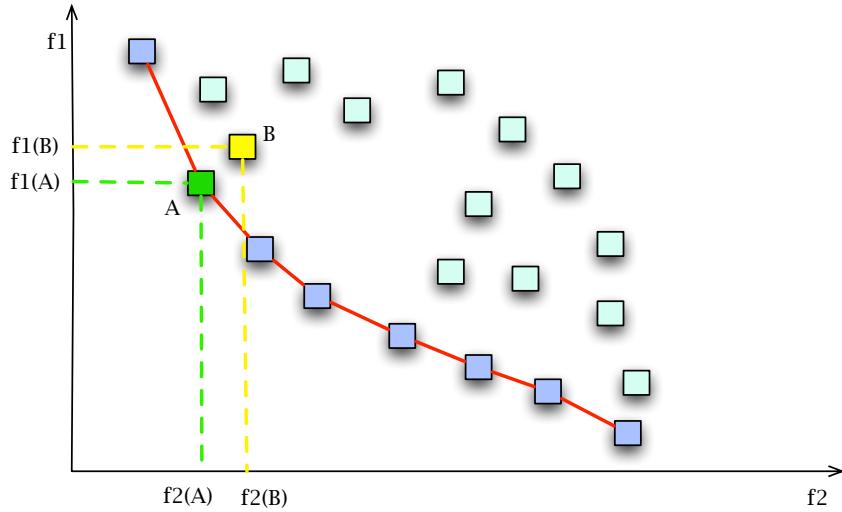


FIGURE 2.5 – Relation de dominance

Dans la figure ci-dessus, on peut savoir  $f_1(B) > f_1(A)$  et  $f_2(B) > f_2(A)$ . En fonction de la définition de dominance, le point A domine le point B. Nous notons  $B \prec A$ . Donc le point A est une solution non dominée et le point B est une solution dominée. Le point B n'appartient pas à l'ensemble de PARETO FRONTIER, parce qu'il est dominée par le point A.

### 2.2.3 Algorithme de BCA\*

Pour résoudre le problème de l'algorithme existant, mon encadrant propose un nouvel algorithme BCA\* qui permet de rechercher un meilleur compromis. Cet algorithme se base sur une énumération implicite de solutions non-dominées, à la manière de MOA\* [SW91], en se focalisant, durant la recherche, sur l'obtention d'un compromis adapté aux préférences de l'utilisateur.

#### Principe de l'algorithme de BCA\*

Pour chaque critère  $i$ , utiliser l'algorithme DIJKSTRA pour trouver les valeurs suivantes :

- $\alpha_i$  : La valeur minimale de critère  $i$  ;
- $\beta_i$  : La valeur maximale de critère  $i$ .

Pour chaque critère  $i$ , calculer les valeurs suivantes :

- $w_i$  : Le poids normé de critère  $i$ , avec l'aide des paramètres  $\alpha_i$ ,  $\beta_i$  et  $\delta_i$  (Le poids attribué au critère), il est calculé par l'équation :  

$$w_i = \delta_i / (\beta_i - \alpha_i)$$
- $\lambda$  : La valeur minimale des optimisations mono critère, elle est obtenu par l'équation :  

$$\min \{ w_i * (\beta_i - \alpha_i) \}.$$

Pour faire fonctionner l'algorithme BCA\* :

- $g\_label$  : La liste qui contient des éléments à explorer.
- $label$  : La liste qui contient des solution de chaque noeud.

---

### Algorithm 2 BCA\*

---

1. La fonction aStarBi :

TantQue  $g\_label$  n'est pas vide Faire

    récupérer le label le plus petit

Si  $\lambda >$  la valeur de ce label Faire

Si il ne s'agit pas du noeud destinataire Faire

Pour tous les arcs sortants Faire

                calculer le vecteur de cout du label

                calculer la priorité d'exploration du label

Si la valeur de priorité  $< \lambda$  Faire

                    tester si les solutions enregistrées dans ce noeud v dominent

Si le label n'est pas dominé en v Faire

                        ajouter le label dans  $g\_label$  et  $label(v)$

FinSi

FinSi

FinPour

SiNon il s'agit du noeud destinataire Faire

                ajouter la solution

                calculer de la priorité d'exploration du label

FinSi

FinSi

FinTantQue

---

### Exemple de l'algorithme de BCA\*

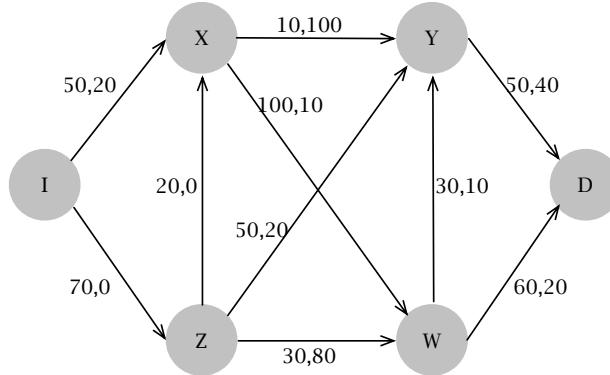


FIGURE 2.6 – Un graphe à traiter

Nous pouvons voir la figure ci-dessus, c'est un graphe contenant 4 sommets. Le noeud I est un sommet du départ et le noeud D est un sommet de la destination. Cet exemple considère que deux critères (distance et sécurité).

Nous pouvons d'abord utiliser DIJKSTRA ALGORITHM qui permet d'obtenir le plus court chemin en considérant respectivement le critère de la distance et le critère de la sécurité.

- Tout d'abord, nous pouvons calculer Heristique  $H_i()$ . On utilise un tableau (L'étiquette jaune sur la figure ci-dessus) pour stocker deux valeurs, cela signifie que combien ça couté optimal à atteindre le noeud D pour chaque critère.

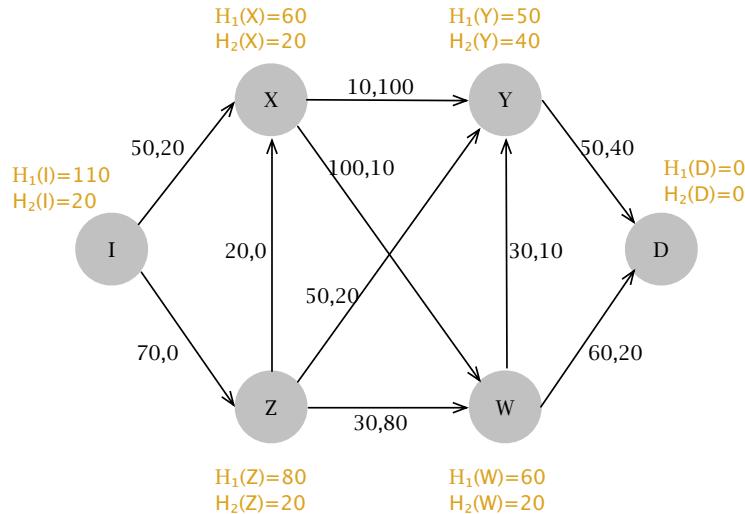


FIGURE 2.7 – Calcul de Heristique

- Ensuite, nous pouvons calculer les 4 valeurs :  $\alpha$ ,  $\beta$ ,  $\delta$  et  $w$  pour chaque critère :

	$\alpha_i$	$\beta_i$	$\delta_i$	$w_i$
critère 1	110	250	1	1 / 140
critère 2	20	160	1	1 / 140

- Et puis, nous pouvons calculer la distance de TCHEBYCHEFF.

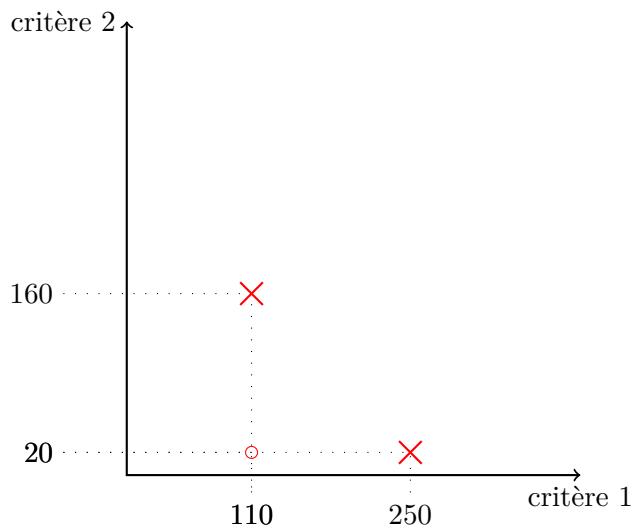


FIGURE 2.8 – Distance de Tchebycheff

Nous pouvons calculer la paramètre :  $\lambda = \min \{ 250 - 110, 160 - 20 \} = 140$

Nous pouvons initialiser deux tables :

- `g_label` : Une liste qui consiste à stocker la priorité qui nous permet de savoir dans quel ordre explorer les noeuds. Il y a 4 valeurs dans la liste `g_label` :
  - le nom de noeud ;
  - la valeur de critère distance ;
  - la valeur de critère sécurité ;
  - la priorité.
- `label(X)` : Une liste qui consiste à stocker la valeur de chaque noeud X.

Nous pouvons insérer le noeud de départ par défaut. Pour cet exemple, nous insérons `{I, 0, 0, 0}`.

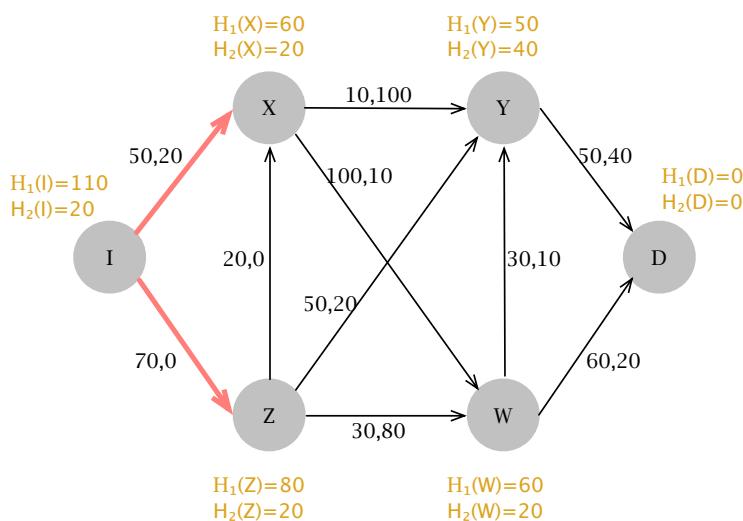


FIGURE 2.9 – Algorithme de BCA\*

4. Nous récupérons le noeud vient de la liste `g_label`, pour l'instant, la liste `g_label` contient qu'un noeud I, alors nous commençons à explorer le noeud I, avec l'aide la figure ci-dessus, nous savons que le noeud I a deux noeuds adjacents X et Z, alors nous calculons d'abord des valeurs pour X et Z en ajoutant le poids d'arc, et puis nous ajoutons le heuristique  $H_i()$  pour chaque noeud.

$$\left\{ \begin{array}{l}
 (I, 0, 0) \xrightarrow{+H(X)} (X, 110, 40) \longrightarrow S = 20 \\
 (I, 0, 0) \xrightarrow{+H(Z)} (Z, 150, 20) \longrightarrow S = 40
 \end{array} \right.$$

FIGURE 2.10 – Calcul de valeur de chaque critère

Nous pouvons voir la figure ci-dessus, le noeud X devient  $(110, 40)$  et Z devient  $(150, 20)$ . Et puis, nous devons calculer la distance de Tchebycheff pour le noeud X et le noeud Z :

- Pour le noeud X,  $S = \min \{ |110 - 110|, |40 - 20| \} = 20$  ;
- Pour le noeud Z,  $Z = \min \{ |150 - 110|, |20 - 20| \} = 40$ .

Nous devons mettre à jour deux liste. En ce moment, ils sont devenus :

- $g\_label = \{ (X, 50, 20, 20), (Z, 70, 0, 40) \}$  ;
- $label(X) = \{ (X, 50, 20) \}$  ;
- $label(Z) = \{ (Z, 70, 0) \}$ .

Apres, nous pouvons refaire dans l'étape 4 jusqu'à la liste  $g\_label$  devient vide.

# Réalisation des besoins

---

Dans cette partie, nous allons maintenant présenter les besoins réalisés. Pour chacun de ces besoins, nous commencerons par faire un point sur ce qui a été fait, en détaillant exactement le travail.

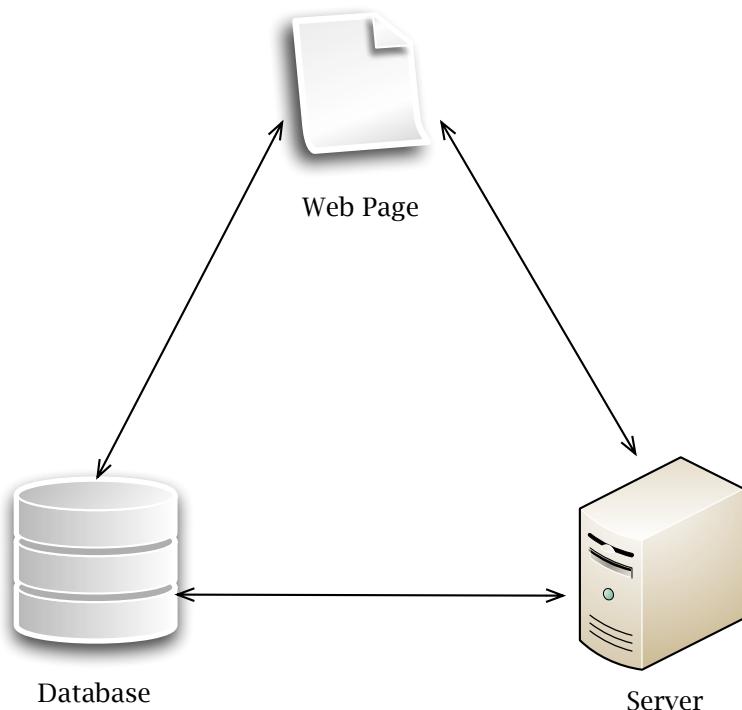


FIGURE 3.1 – Architecture principale de Géovélo

Voici, la figure ci-dessus est une architecture de Géovélo qui contient 3 parties principalement (Web Page, Database, Serveur) :

Un client peut utiliser le site Géovélo pour faire une recherche. Il existe deux types de recherche :

- Recherche statique : Ce type de recherche signifie qu'un client consulte des données statiques (Couverture Géovélo, Loueur/réparateur/vente, Lieux universitaires, Loire à vélo...) qui n'a pas besoin de lancer le serveur pour faire un calcul. Le site Géovélo peut récupérer des coordonnées de la base de données directement ;
- Recherche dynamique : Ce type de recherche signifie qu'un client demande un calcul pour un itinéraire. Le site Géovélo peut transmettre des coordonnées et des préférences au serveur en format XML (Grâce à la technique XML-RPC). Quand la serveur reçoit cette requête, elle récupère des coordonnées nécessaires de la base de donnée et lance un calcul en utilisant DIJKSTRA ALGORITHM et BCA\* ALGORITHM. Quand elle trouve un itinéraire correspondant le besoin de client, elle va transmettre ce résultat au site Géovélo qui permet de tracer cet itinéraire et l'afficher.

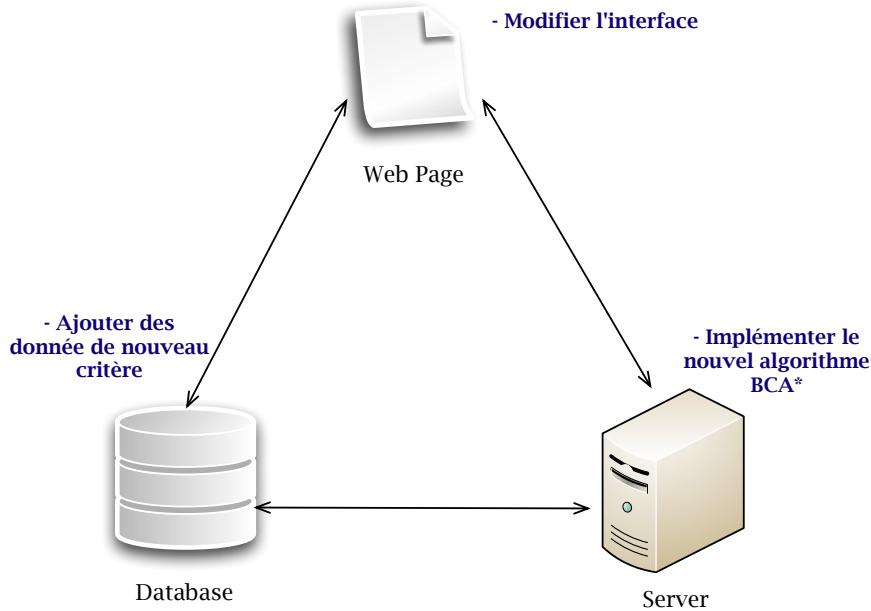


FIGURE 3.2 – Tâches de chaque partie

Maintenant, nous allons présenter les tâches à faire pour chaque partie :

- Server : Il faut implémenter ce nouvel algorithme BCA\* et tester cet algorithme fonctionne bien pour deux critères (distance et sécurité). Ensuite, modifier l'interface de Web Page pour adapter cet algorithme ;
- Database : Nous devons modéliser un nouveau critère et ajouter les données nécessaires dans la base de données ;
- Web Page : Selon le temps restant, il faut essayer d'améliorer l'interface.

Tout d'abord, nous devons se focaliser sur la partie de serveur pour implémenter BCA\* ALGORITHM.

### 3.1 Implémentation de l'algorithme BCA\*

Comme nous l'avons mentionné plus tôt, l'implémentation de l'algorithme BCA\* contient 2 étapes : nous devons d'abord utiliser DIJKSTRA ALGORITHM pour calculer la valeur minimale et maximale pour chaque critère. Et puis, nous utilisons BCA\* ALGORITHM pour trouver un itinéraire.

#### 3.1.1 Structure de données

Nous allons d'abord présenter la structure de données que nous avons utilisé :

```

struct structDynaNode {
    double mono[NB_OBJECTIVE];
    double comb;
    edge edgeBack;
    sortseq<label,label> solutions3;
    dictionary<label,p_queue<int,label> : :item> D;
    LEDA_MEMORY(structDynaNode)
} ;

```

Cette structure de données contient les informations dynamiques spécifiques à un noeud. Elle permet d'enregistrer les informations temporaires quand nous utilisons DIJKSTRA ALGORITHM pour chercher le plus court chemin, les information s'agit de la distance, la sécurité, l'effort, les arcs qui arrivent à ce noeud...

```

struct structStatNode {
    unsigned int idNode;
    int elevation;
    float lon;
    float lat;
    double alpha[NB_OBJECTIVE];
    double beta[NB_OBJECTIVE];
    float comb[NB_COMB][NB_OBJECTIVE];
    LEDA_MEMORY(structStatEdge)
};

}

```

Cette structure de données contient les informations statiques spécifiques à un noeud. Elle enregistre l'identifiant d'un noeud, la valeur minimale de chaque critère, valeur maximale de chaque critère... Les informations devront être utilisées pendant BCA\* ALGORITHM.

```

struct structStatEdge {
    int idTroncon;
    float distance;
    float bike_duration;
    float car_duration;
    float foot_duration;
    float elevation;
    float bike_comfort;
    int categorie;
    int tranquilité;
    bool bike;
    list <two_tuple<double,double>> listePoints;
    leda::string nom_rue;
    LEDA_MEMORY(structStatEdge)
};

}

```

Cette structure de données contient les informations statiques spécifiques à un arc. Elle enregistre l'identifiant, la longueur, la sécurité, la catégorie, la tranquillité d'un tronçon et toutes les paires de coordonnées comme {longitude, latitude} pour les points qui composent le tronçon. La catégorie et la tranquillité sont utilisées pour distinguer les différents types de tronçon.

Il existe aussi une classe label qui est utilisée pour enregistrer les solutions de chaque noeud. Dans cette classe, il y a 4 attributs importants :

- array<double> values : Le coût de noeud de départ au noeud courant ;
- node n : L'identifiant de noeud ;
- edge e : L'identifiant d'arc précédent ;
- double c : La valeur de priorité.

### 3.1.2 Réalisation

Pour l'instant, il a déjà existé une version d'algorithme BCA\* écrit par mon encadrant, il suffit de modifier les sources originaux pour adapter ce nouvel algorithme. Pour faire fonctionner l'algorithme BCA\*, nous devons modifier deux parties de ressources :

## Partie de Serveur

Dans la partie de serveur, nous récupérons 9 paramètres de site Géovélo et transmettons 6 paramètres au site Géovélo par défaut :

- nodeDepart1 : L'identifiant du noeud source de l'arc où se trouve le point de départ ;
- nodeDepart2 : L'identifiant du noeud destination de l'arc où se trouve le point de départ ;
- nodeArrivee1 : L'identifiant du noeud source de l'arc où se trouve le point d'arrivée ;
- nodeArrivee2 : L'identifiant du noeud destination de l'arc où se trouve le point d'arrivée ;
- deltaDepart : La position du point de départ par rapport à son arc (en mètres) ;
- deltaArrivee : La position du point d'arrivée par rapport à son arc (en mètres) ;
- idtrDepart : L'identifiant du tronçon où se trouve le point de départ ;
- idtrArrivee : L'identifiant du tronçon où se trouve le point d'arrivée ;
- coeff : Le coefficient de compromis entre les critères.

En ce moment, nous avons seulement besoin d'utiliser nodeDepart1 et nodeArrivee1. Pour adapter ce nouvel algorithme, nous devons ajouter 2 paramètres :

- cdistance : Le poids de critère de distance ;
- csecurtie : Le poids de critère de sécurité.

L'algorithme BCA\* a besoin de récupérer les 2 paramètres qui permettent de calculer l'itinéraire en considérant les attributs fournis par le client.

- distance\_char : La longueur d'itinéraire ;
- liste\_coeff : La liste d'itinéraire choisi ;
- best\_coeff : L'itinéraire choisi ;
- nb\_sols : Le nombre de solutions ;
- liste\_point : Les coordonnées de points d'itinéraire ;
- feuille\_route : Les coordonnées de la feuille de route.

Pour la sortie, nous devons ajouter 2 paramètres qui permettent d'enregistrer les poids de critère choisi par client, ils sont utilisés pour mettre à jour les poids de critère sur la zone de vos préférences dans la partie de site Géovélo.

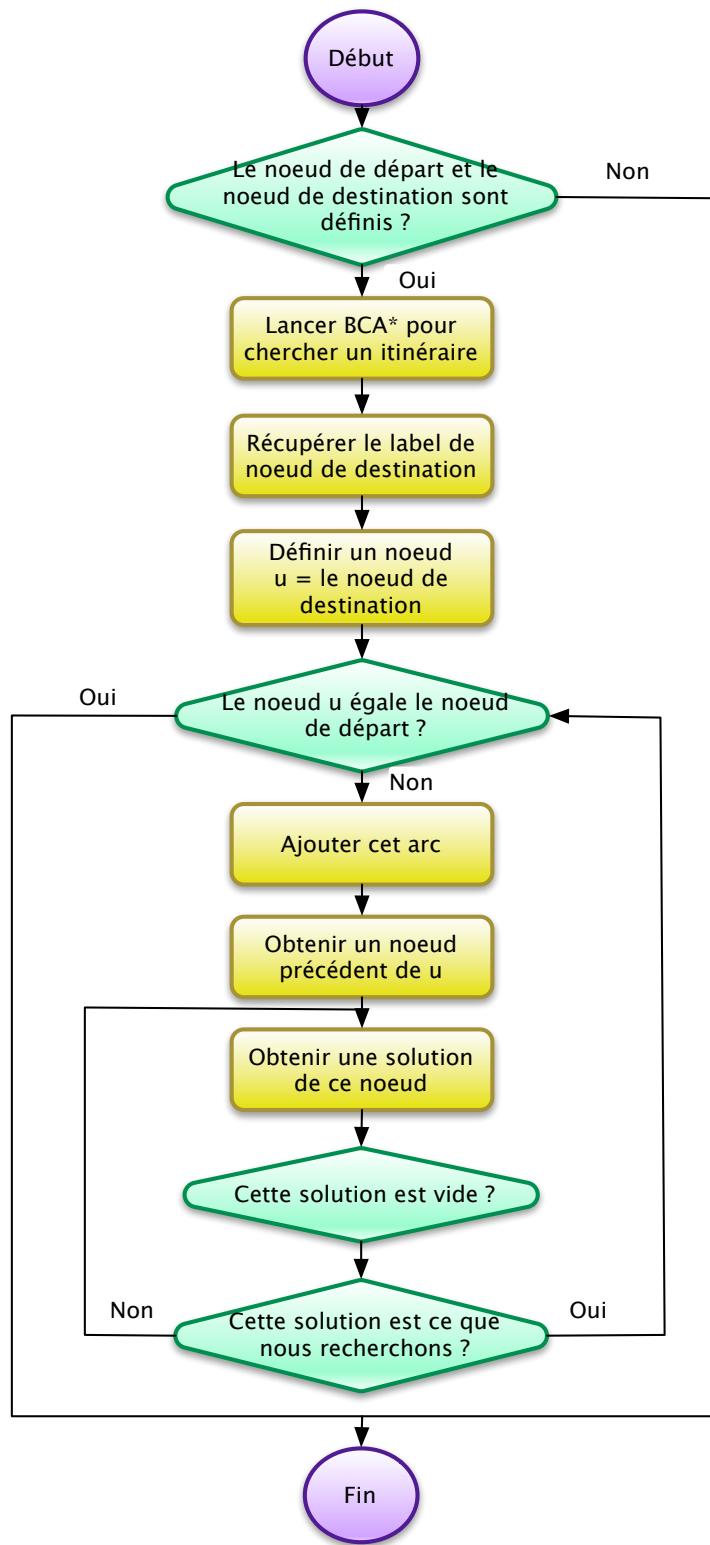


FIGURE 3.3 – Etapes pour implémenter BCA\*

La figure ci-dessus est une étape principale pour implémenter l'algorithme BCA\* :

1. Nous récupérons le point de départ et le point de destination, nous devons d'abord vérifier ses existences ;
2. Nous passons 3 paramètres (le point de départ, le point de destination et les poids de critères) à la fonction BCA\* qui permet de calculer un itinéraire selon la préférence d'utilisateur ;
3. Nous pouvons trouver qu'une solution dans la liste de noeud de destination. Cette solution est un objet de classe `label`, donc elle contient deux attributs importants que nous pouvons utiliser par la suite : le cout de chaque critère (du départ à la destination) et l'arc précédent ;
4. Nous commençons parcourir le noeud de la destination au départ. Chaque fois nous devons d'abord enregistrer l'arc trouvé. Ensuite nous pouvons obtenir le noeud précédent avec l'aide de cet arc ;
5. Dans la liste de solution de ce noeud, elle contient plusieurs chemins de départ à ce noeud, alors nous devons vérifier quelle solution est ce que nous recherchons. Pour faire ça, il suffit de trouver une solution qui peut répondre à la formule suivante :

$$\text{Le cout de ce noeud suivant} - \text{Le cout d'arc} = \text{Le cout de ce noeud}$$

Si cette formule est correcte pour tous les critères, ça signifie que nous trouvons cette solution ;

6. Nous pouvons mettre à jour le noeud u courant et l'arc courant. Et puis, nous revenons à l'étape 4.

## Partie de Web Page

Nous avons bien réalisé la partie de Serveur, il faut ensuite modifier l'interface pour adapter ce nouvel algorithme.

Nous savons que algorithme de ce projet est développé en utilisant le plus court chemin (une version multi-critère de l'algorithme de Dijkstra). Nous pouvons voir la figure suivante, quand un client a choisi un itinéraire, cet algorithme peut renvoyer une liste de solution de compromis entre le critère de distance et le critère de sécurité.

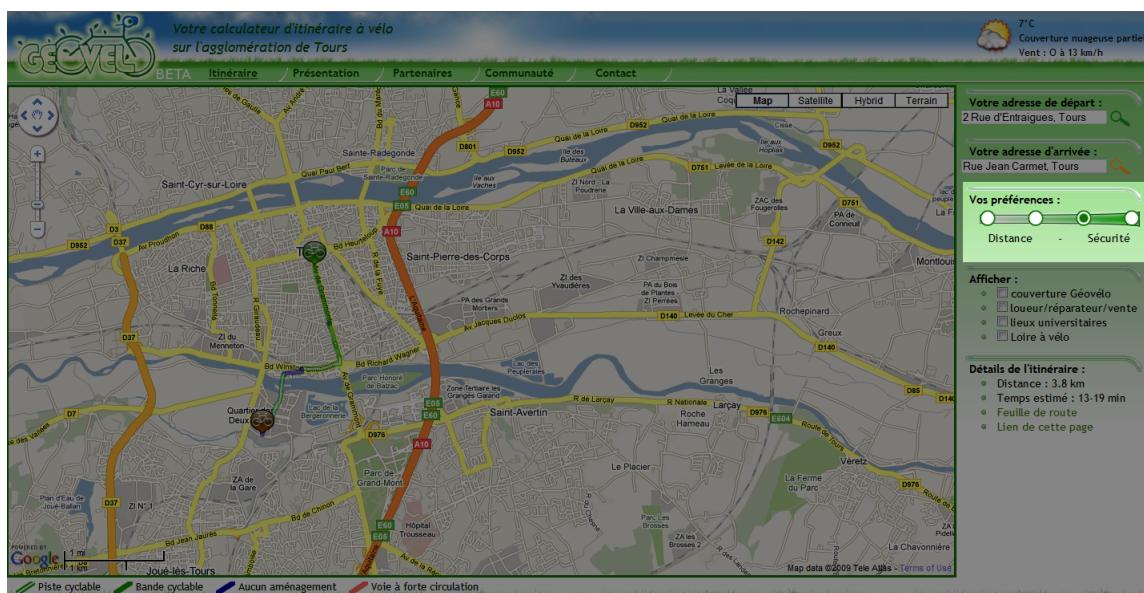


FIGURE 3.4 – Zone originale de vos préférences

Mais ce nouvel algorithme BCA\* renvoie qu'une solution de meilleure compromis. Evidement, le modèle de vos préférences fonctionne plus pour l'algorithme BCA\*, nous devons modifier la zone de vos préférences. Pour faire ça, il faut d'abord comprendre l'architecture du site Géovélo.

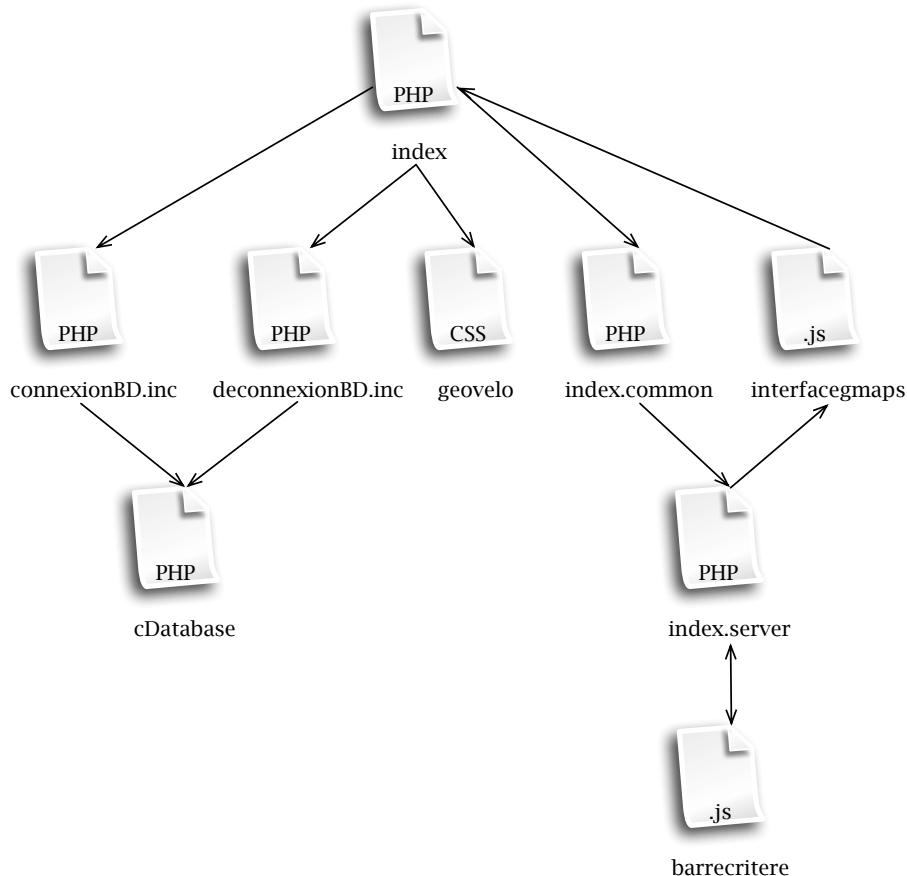


FIGURE 3.5 – Architecture du site Géovélo

La figure ci-dessus est une architecture principale de site Géovélo, nous allons présenter le fonction de chaque fichier :

- `index.php` : La page d'accueil de site Géovélo qui affiche les éléments principaux et appelle les autres pages.
- `connexionBD.inc.php` : La page consiste à se connecter à la base de données.
- `deconnexionBD.inc.php` : La page consiste à se déconnecter de la base de données.
- `cDatabase.php` : La page consiste à créer une classe de gestion d'une connexion. Cette classe permet de gérer toutes les connexions vers la base de données
- `index.common.php` : La page consiste à déclarer des librairies, instancier des objets XAJAX.
- `interfacegmaps.js` : La page consiste à gérer des codes JavaScript pour l'interface Google maps.
- `index.server.php` : La page consiste à créer des fonctions XAJAX de calcul d'itinéraire, appeler la fonction XMLRPC du serveur pour calculer un ensemble de solutions.
- `barrecritere.js` : La page s'agit des fonctions de javascript qui charge d'afficher la zone de vos préférences.

Le fichier `barrecritere.js` et le fichier `index.server.php` sont très importants, la plupart du temps nous travaillons sur ces deux fichiers.

Pour modifier la zone de vos préférences, nous devons focaliser sur le fichier `index.server.php` et le fichier `barrecritere.js`. L'entrée d'algorithme BCA\* a besoin de poids de chaque critères, alors nous devons concevoir une zone qui permet de choisir le poids pour chaque critère.

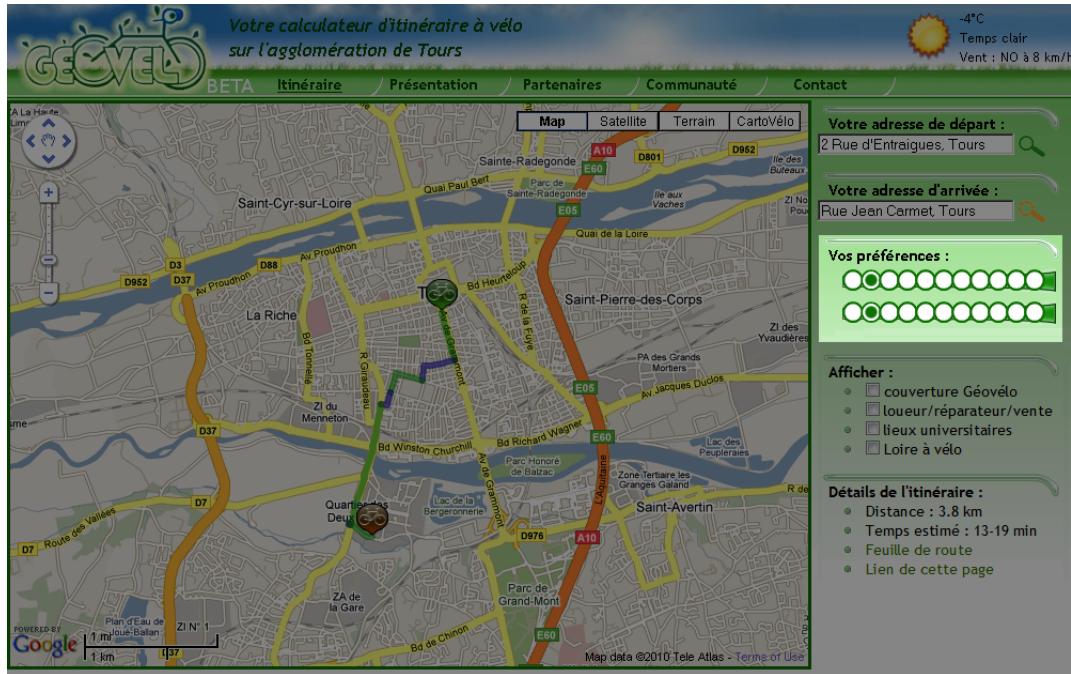


FIGURE 3.6 – Zone modifiée de vos préférences

Voici la figure ci-dessus est la zone modifiée. Chaque critère a 11 poids qui peuvent être choisi. chaque poids correspond à une chiffre (de 0 à 10). Plus la chiffre est grande, plus le client considère ce critère qui est important. Il existe aussi deux cas particuliers :

- Si le client choisit le dernier poids d'un critère, la couleur de poids va devenir rouge. Ca signifie que nous considérons que ce critère et le problème va devenir simplement le plus court chemin ;
- Si le client choisit le premier poids d'un critère, la couleur de poids va devenir gris. Ca signifie que nous considérons plus ce critère. Pour l'instant, nous considérons que deux critères, alors il semble inutile.

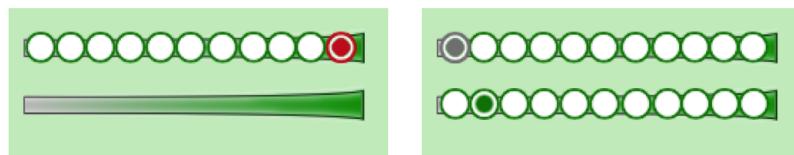


FIGURE 3.7 – Cas particuliers

## 3.2 Implémentation des nouveaux critères

Quand Nous avons fini d'implémenter l'algorithme BCA\*, nous commençons de modéliser et implémenter des nouveaux critères. Nous allons d'abord présenter la structure de la base de données.

### 3.2.1 Structure de la base de données

Pour trouver un itinéraire, il faut récupérer les informations géométriques comme les coordonnées de départ, de destination, et des tronçons, etc. Pour l'instant, il existe déjà les 7 tables :

- agglo\_teleatlas ;
- edge\_ta ;

- geometry\_columns ;
- loueur\_reparateur ;
- node\_ta ;
- poi ;
- spatial\_ref\_sys.

Dans la partie suivantes, je vais présenter la fonctionnalité de chaque table et expliquer des attributs de chaque table.

```
TABLE agglo_teleatlas
(
    gid serial NOT NULL,
    id bigint DEFAULT (-1),
    "name" text,
    oneway character varying(2),
    categorie1 smallint DEFAULT 0,
    categorie2 smallint DEFAULT 0,
    tranquilite1 smallint DEFAULT 0,
    tranquilite2 smallint DEFAULT 0,
    desserte smallint DEFAULT 0,
    paysage smallint DEFAULT 0,
    x double precision DEFAULT 0,
    y double precision DEFAULT 0,
    angle double precision DEFAULT 0,
    the_geom geometry,
    the_geom2 geometry,
    revetement smallint DEFAULT 0,
    f_elev smallint DEFAULT 0,
    t_elev smallint DEFAULT 0,
    loire_a_velo smallint DEFAULT 0,
    the_geom3 geometry,
    type_route smallint DEFAULT 0,
)
```

La table agglo\_teleatlas s'agit des données brutes qui contiennent toutes les informations.

```
CREATE TABLE edge_ta
(
    gid serial NOT NULL,
    source integer NOT NULL,
    destination integer NOT NULL,
    id_tr integer NOT NULL,
    difficulte double precision,
    longueur double precision,
    categorie1 smallint DEFAULT 0,
    tranquilite smallint DEFAULT 0,
    desserte integer DEFAULT 0,
    sens_creation smallint DEFAULT 0,
)
```

La table edge\_ta consiste à stocker toutes les informations concernant des tronçons. Les attributs utilisés sont suivants :

- source : Identificateur de départ ;
- destination : Identificateur de destination ;
- id\_tr : Identificateur de tronçons ;
- longueur : Longueur de tronçons ;
- categorie1 : Type d'aménagement vélo :
  1. non renseigne ;

- 2. piste cyclable ;
  - 3. bande cyclable ;
  - 4. mixte (pieton / pas pieton) ;
  - 5. rien.
- **tranquilité** : Tronçon est conseillé pour le vélo ou non :
1. conseillé ;
  2. non conseillé.
- **desserte** : Tronçon est un tronçon de desserte ou non :
1. pas desserte ;
  2. desserte.
- **sens\_creation** : Sens de tronçons.

Pour l'instant, nous considérons que deux critères (distance et sécurité), alors nous pouvons consulter la table `edge_ta` pour obtenir le cout de distance et le cout de sécurité :

- **Distance** : il suffit de récupérer l'attribut `longueur` ;
- **Sécurité** : il n'existe pas de valeur fixe pour le critère sécurité, les attributs `categorie1` et `tranquilité` sont utilisés pour calculer la sécurité :

```
if (categorie1 == 1)
    securite = 1;
else
if (categorie1 == 2)
    securite = 1.5;
else
if (categorie1 == 3)
    securite = 2;
else
if (categorie1 == 4)
    securite = 3;
else
    securite = 3;
if (tranquilité == 1)
    securite += 2.5;
```

```
CREATE TABLE geometry_columns
(
    f_table_catalog character varying(256) NOT NULL,
    f_table_schema character varying(256) NOT NULL,
    f_table_name character varying(256) NOT NULL,
    f_geometry_column character varying(256) NOT NULL,
    coord_dimension integer NOT NULL,
    srid integer NOT NULL,
    "type" character varying(30) NOT NULL,
)
```

La table `geometry_columns` contient toutes les informations géométriques.

```
CREATE TABLE loueur_reparateur
(
    gid serial NOT NULL,
    nom character varying(100),
    loueur smallint DEFAULT 0,
    reparateur smallint DEFAULT 0,
    vente smallint DEFAULT 0,
```

```

adresse character varying(250),
code_postal integer DEFAULT 0,
ville character varying(200),
site_internet character varying(250),
horaires character varying(250),
photo character varying(250),
the_geom geometry,
the_geom2 geometry,
)

```

La table loueur\_reparateur consiste à stocker toutes les informations concernant des lieux de loueur et de réparateur.

```

CREATE TABLE node_ta
(
    gid serial NOT NULL,
    the_geom geometry,
    the_geom2 geometry,
    elev smallint DEFAULT 0,
)

```

La table node\_ta consiste à stocker toutes les informations concernant des noeuds. Les attributs utilisés sont suivants :

- gid : Identificateur de noeud ;
- the\_geom : Coordonnées de noeud ;
- id\_tr : Identificateur de tronçons.

```

CREATE TABLE poi
(
    gid serial NOT NULL,
    categorie smallint DEFAULT 0,
    souscategorie smallint DEFAULT 0,
    nom character varying(100),
    adresse character varying(250),
    code_postal integer DEFAULT 0,
    ville character varying(200),
    site_internet character varying(250),
    horaires character varying(250),
    photo character varying(250),
    the_geom geometry,
    the_geom2 geometry,
    description character varying(300),
)

```

La table poi consiste à stocker toutes les informations concernant des lieux universitaires.

### 3.2.2 Réalisation de critère d'effort

Dans la partie de réalisation, nous allons présenter ce que nous faisons pour le critère d'effort.

#### Partie de Database

```

CREATE TABLE altitude
(
    pos bigint NOT NULL,
    alt integer,
)

```

```
the_geom geometry,
gid integer NOT NULL,
)
```

La table altitude qui permet de stocker l'altitude de chaque point sur la carte. En ce moment, chaque point est un argument similaire. C'est impossible que nous stockons tous les altitude sur la carte, parce que les données sont trop grandes. Pour résoudre ce problème, nous attribuons chaque zone à une altitude moyenne.

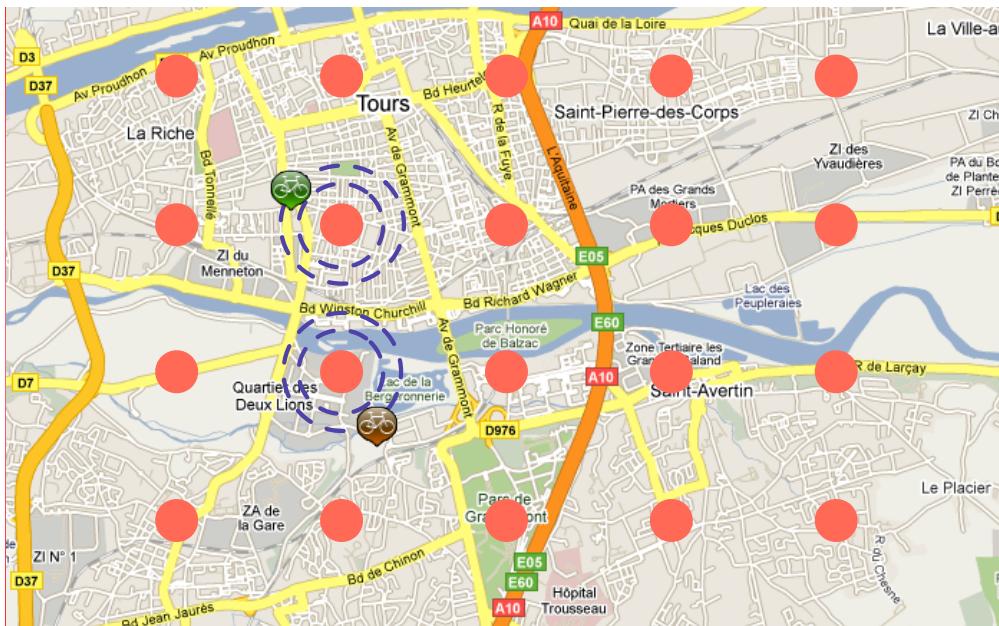


FIGURE 3.8 – Calcul d'effort

Par exemple, nous pouvons voir la figure ci-dessus. Chaque point rond représente une altitude moyenne. Pour trouver l'altitude de point de départ et l'altitude de point de destination, l'idée est que nous devons agrandir le diamètre progressivement jusqu'au moment où nous trouvons des points.

En ce moment, quand nous allons calculer une itinéraire sur la carte, ça signifie que nous devons effectuer une opération de base de données pour chaque point d'une itinéraire, évidemment ce n'est pas très efficace. Pour éviter ce problème, c'est mieux que nous calculons d'abord l'altitude de tous les points. Pour faire ça, nous créons un attribut altitude dans la table node\_ta et mettons à jour les altitudes en utilisant la commande suivante :

```
UPDATE node_ta N e
SET altitude = (SELECT A.alt as altitude FROM altitude_idf A WHERE A.the_geom &&
    Expand(N.the_geom2 ,0.01)
ORDER BY N.the_geom2 <-> A.the_geom LIMIT 1);
```

Pour l'instant, nous pouvons obtenir l'altitude de chaque point, il faut aussi créer une fonction getaltitude(integer, integer) permettant de calculer l'altitude de chaque tronçons.

```
DECLARE
  s integer ;
  d integer ;
  alt double precision ;
BEGIN
  SELECT N.altitude INTO s FROM node_ta N WHERE N.gid = source ;
  SELECT N.altitude INTO d FROM node_ta N WHERE N.gid = destination ;
  alt = d - s ;
```

```

IF alt < 0 THEN
    alt = 0 ;
END IF ;
RETURN alt ;
END

```

### Partie de Serveur

Nous devons d'abord ajouter deux paramètres : une paramètre d'entrée qui se charge de récupérer le poids de critère d'effort et une paramètre de sortie qui permet de renvoyer ce poids de critère d'effort. Et puis, pour chaque tronçons nous utilisons l'équation suivante pour calculer la valeur d'effort :

$$\text{Effort} = (\text{altitude de tronçons} * 100 + 1) * \text{longueur de tronçons} / 1000 * \text{matériel} / (\text{âge} + \text{sex})$$

Nous pouvons utiliser la fonction précédente `getAltitude(integer, integer)` pour calculer une altitude de tronçons. Pour les 3 paramètres (matériel, âge et sexe), ils sont fournis par la zone de vos préférences.

### Partie de Web Page

Nous devons d'abord ajouter une barre de critère d'effort. Ensuite concevoir une zone qui permet de saisir les 3 paramètres (matériel, âge et sexe), elle est utile pour l'étudiant de DA pour trouver une bonne valeur. Selon la demande de l'étudiant de DA, nous devons ajouter une liste qui permet d'afficher l'altitude de chaque point d'une itinéraire. Enfin dans la zone de détails de l'itinéraire, nous ajoutons un label qui affiche la valeur d'effort.

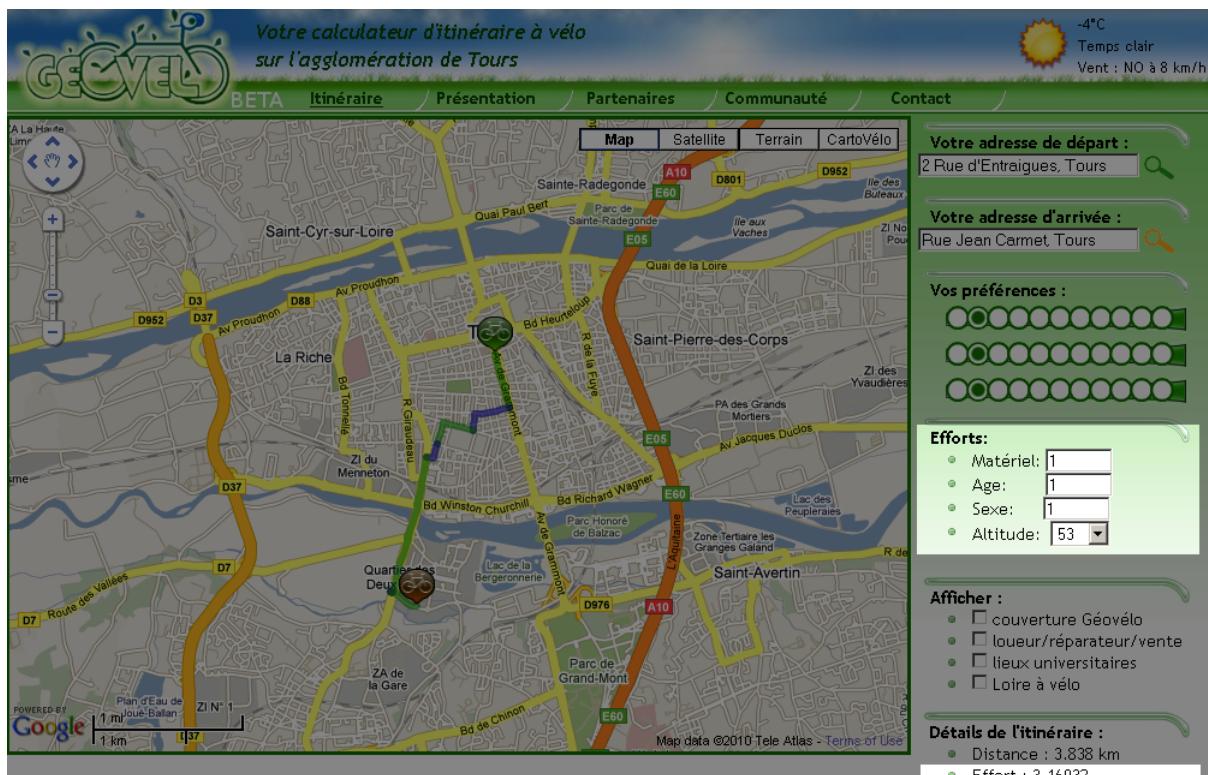


FIGURE 3.9 – Zone d'effort

### 3.2.3 Réalisation de critère de sécurité

Dans la partie de réalisation, nous allons présenter ce que nous faisons pour le critère de sécurité.

#### Partie de Database

```
CREATE TABLE vitesse
(
    id bigint,
    kph smallint
)
```

Nous ajoutons une table `vitesse` qui s'agit la vitesse de chaque tronçons.

#### Partie de Serveur

La domaine de vitesse est entre 0 et 120. Le plus vitesse est grande, le plus la tronçons est en insécurité. Nous pouvons mettre à jour le critère de sécurité en tenant compte de vitesse :

```
if (vitesse > 0 && vitesse <= 20)
    securite += 0.5;
else
if (vitesse > 20 && vitesse <= 40)
    securite += 1;
else
if (vitesse > 40 && vitesse <= 60)
    securite += 1.5;
else
if (vitesse > 60 && vitesse <= 80)
    securite += 2;
else
if (vitesse > 80 && vitesse <= 100)
    securite += 2.5;
else
if (vitesse > 100 && vitesse <= 120)
    securite += 3;
```

## 3.3 Amélioration

Dans cette partie, nous allons présenter la méthode pour améliorer l'algorithme BCA\* et temps du calcul.

### 3.3.1 Amélioration de l'algorithme

En ce moment, le site Géovélo est disponible pour le client en considérant 3 critères (distance, sécurité, effort) en même temps. Si un client choisit un itinéraire en considérant qu'un critère, le problème devient simplement un plus court chemin, il n'a pas besoin de lancer BCA\* ALGORITHM pour chercher un itinéraire. L'idée est d'utiliser directement DIJKSTRA ALGORITHM pour résoudre ce problème.

Nous savons que la fonction BCA\* permet d'utiliser DIJKSTRA ALGORITHM pour calculer le plus court chemin de chaque critère avant elle lance BCA\* ALGORITHM. L'idée est que nous devons vérifier le coût de chaque critère calcule par DIJKSTRA ALGORITHM. Si tous les coûts sont même, ça signifie qu'il y a le même résultat en considérant le différent critère, il suffit de retourner directement la solution calculée par DIJKSTRA ALGORITHM.

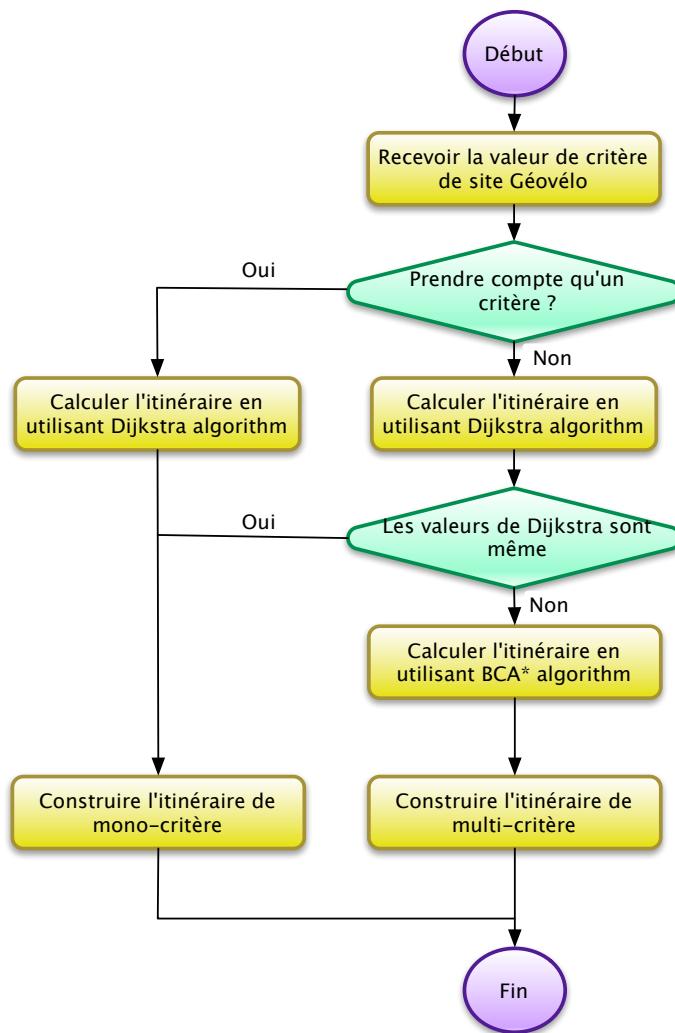


FIGURE 3.10 – Amélioration de l'algorithme

La figure ci-dessus est une étape principale pour améliorer l'algorithme :

1. Dans la partie de Serveur, nous récupérons la valeur de chaque critère de site Géovélo. Pour le moment, nous obtenons 3 valeurs (distance, sécurité, effort) ;
2. Nous vérifions s'il existe une valeur égale 10 qui signifie que nous prenons compte que ce critère. Si la réponse est oui, nous pouvons utiliser directement DIJKSTRA ALGORITHM pour calculer l'itinéraire ;
3. Sinon nous utilisons aussi DIJKSTRA ALGORITHM pour faire la même chose, mais cette fois, nous devons vérifier les valeurs minimaux et maximaux de chaque critère sont même. Si la réponse est oui, il n'a pas besoin de lancer BCA\* ALGORITHM, nous pouvons directement atteindre l'étape de construction de l'itinéraire ;
4. Si le client a choisi plusieurs critères, de plus les valeurs de Dijkstra ne sont pas même. Dans ce cas, nous calculons l'itinéraire en utilisant BCA\* ALGORITHM ;
5. Dans l'étape de construction de l'itinéraire, le mono-critère utilisé par DIJKSTRA ALGORITHM construit une itinéraire du départ à la destination, et le multi-critère construit une itinéraire de la destination au départ.

### 3.3.2 Amélioration de temps du calcul

Pour l'instant, le nouvel algorithme fonctionne bien, mais il existe aussi le problème : Si un itinéraire est longe, ça pourra prendre beaucoup de temps du calcul.



FIGURE 3.11 – Exemple d'un itinéraire

Par exemple, nous choisissons le point de départ 103 LES CAVES POUPIÈRES, LUYNES et le point de destination 26 AVENUE DU MARÉCHAL D'ORNANO, CHAMBRAY-LÈS-TOURS. Ca prend environ 10 minutes pour calculer cet itinéraire. Evidement, un client ne peut pas attendre 10 minutes pour qu'un itinéraire.

Pour éviter ce problème, une idée est de modifier la paramètre  $\varepsilon$  qui sont dans la fonction `testDomination`. Ce paramètre permet de déterminer le nombre de solution. Le plus ce paramètre est grand, le moins de nombre dans la liste de solution. Le moins de nombre signifie que nous pouvons une solution plus rapide, mais elle va perdre la précision.

Donc c'est aussi un problème de compromis, pour trouver une bonne valeur de  $\varepsilon$ , nous décidons tester les différentes valeurs de  $\varepsilon$  parmi 200 itinéraires différents.

C'est mieux que nous enregistrons tous les informations dans la base de données, parce que c'est plus facile d'utiliser le langage SQL pour comparer les différentes solutions. Nous décidons créer deux tables dans la base de données :

```
CREATE TABLE instance
(
    id serial NOT NULL,
    depart integer,
    destination integer,
    distance double precision[],
    securite double precision[],
    effort double precision[],
)
```

La table `instance` consiste à stocker 200 itinéraires différents. Pour chaque critère, nous affectons la valeur de  $\varepsilon$  à 0 par défaut. Les attributs utilisés sont suivants :

- `id` : Identifiant d'itinéraire ;
- `depart` : Identifiant de noeud de départ ;
- `destination` : Identifiant de noeud de destination ;
- `distance` : Cout de critère de distance ;

- **securite** : Cout de critère de sécurité ;
- **effort** :Cout de critère d'effort.

```
CREATE TABLE solution
(
    id serial NOT NULL,
    id_instance integer,
    poids integer[],
    epsilon double precision[],
    noeud integer,
    temps_init double precision,
    temps_bca double precision,
    valeur double precision,
    meilleure double precision[],
)
```

La table solution permet de stocker la solution de  $\varepsilon$  différent. Les attributs utilisés sont suivants :

- **id** : Identifiant de solution ;
- **id\_instance** : Identifiant de d'itinéraire ;
- **poids** : Poids de chaque critère ;
- **epsilon** :  $\varepsilon$  de chaque critère ;
- **noeud** : Nombre de noeud à explorer ;
- **temps\_init** : Temps de calcul de DIJKSTRA ALGORITHM ;
- **temps\_bca** : Temps de calcul de BCA\* ALGORITHM ;
- **valeur** : Valeur de  $\lambda$  ;
- **meilleure** : Solution trouvée.

Quand nous avons fini de créer deux tables dans la base de données, nous crérons un SCRIPT qui permet de lancer le programme 200 fois pour calculer l'itinéraire, chaque fois nous prenons un départ et une destination aléatoirement.

Et puis nous créerons aussi un SCRIPT qui permet de calculer 200 itinéraires différents en utilisant  $\varepsilon$  différent. Pour l'instant, nous considérons qu'un le cas que les poids sont égaux à 1.

$\varepsilon$									
0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	
0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	

TABLE 3.1 – Domaine de  $\varepsilon$

Nous décidons de tester chaque critère en utilisant la valeur de  $\varepsilon$  ci-dessus. Pour chaque valeur de  $\varepsilon$ , nous devons comparer la différence avec  $\varepsilon$  originale. Il existe 6 options que nous pourrons comparer :

- **moyenne** : Moyenne de différence de temps de calcul en utilisant BCA\* ALGORITHM ;
- **max** : Valeur maximale de différence de temps de calcul en utilisant BCA\* ALGORITHM ;
- **min** : Valeur minimale de différence de temps de calcul en utilisant BCA\* ALGORITHM ;
- **même** : Précision de solution (sur 200 instances) ;
- **valeur** :
- **temps** : Moyenne de temps de calcul en utilisant BCA\* ALGORITHM.

$\varepsilon$	moyenne	max	min	même	valeur	temps
{ 0.001, 0, 0 }	6.37384975	621.100033	-0.020039	198	2.31709120738146e-05	0.455050065
{ 0.002, 0, 0 }	6.43944979	623.53003	-0.040025	198	2.31709120738146e-05	0.389450025
{ 0.003, 0, 0 }	6.488849645	625.290032	-0.029999	196	0.000270881646140671	0.34005017
{ 0.004, 0, 0 }	6.51764975	626.250029	-0.029999	195	0.00031878629079344	0.311250065
{ 0.005, 0, 0 }	6.536900025	626.980033	-0.030003	194	0.000319831042505924	0.29199979
{ 0.006, 0, 0 }	6.563749895	628.370031	-0.030021	193	0.000365332561392438	0.26514992
{ 0.007, 0, 0 }	6.58659998	628.93003	-0.020023	192	0.000380013176455837	0.242299835
{ 0.008, 0, 0 }	6.60585008	629.440031	-0.020037	190	0.000746068150417435	0.223049735
{ 0.009, 0, 0 }	6.617399765	629.850033	-0.029998	190	0.000769480322391908	0.21150005
{ 0.01, 0, 0 }	6.63454969	630.460034	-0.020032	189	0.00101670637814947	0.194350125
{ 0.02, 0, 0 }	6.638899835	631.900033	-0.020035	184	0.00306715715603449	0.18999998
{ 0.03, 0, 0 }	6.653199795	632.510033	-0.020026	184	0.0031399691675285	0.17570002
{ 0.04, 0, 0 }	6.655749855	632.700033	-0.020024	184	0.0031399691675285	0.17314996
{ 0.05, 0, 0 }	6.659099745	632.69003	-0.029999	184	0.00314024111357441	0.16980007
{ 0.06, 0, 0 }	6.65384983	632.640034	-0.020041	183	0.00355953090701858	0.175049985
{ 0.07, 0, 0 }	6.65669989	632.690034	-0.030021	183	0.00355953090701858	0.172199925
{ 0.08, 0, 0 }	6.656799855	632.710032	-0.020037	183	0.00355953090701858	0.17209996
{ 0.09, 0, 0 }	6.65424977	632.380032	-0.020037	183	0.00355953090701858	0.174650045

TABLE 3.2 – Tests d’itinéraire différent

La table ci-dessous est un résultat pour le critère de distance. Nous savons que le temps de calcul en utilisant BCA\* ALGORITHM est 6.828899815 par défaut.

Evidement,  $\varepsilon$  nouvelle est plus rapide que  $\varepsilon$  original. Les valeurs de  $\varepsilon$  nouvelle restent 0.2 en moyenne. Mais avec  $\varepsilon$  accrue, la précision de solution diminue.

Pour les autres critères, ils montrent la même situation. Donc c'est mieux que nous fixer la valeur de  $\varepsilon$  entre 0.001 et 0.002.

### 3.4 Modification de l'interface

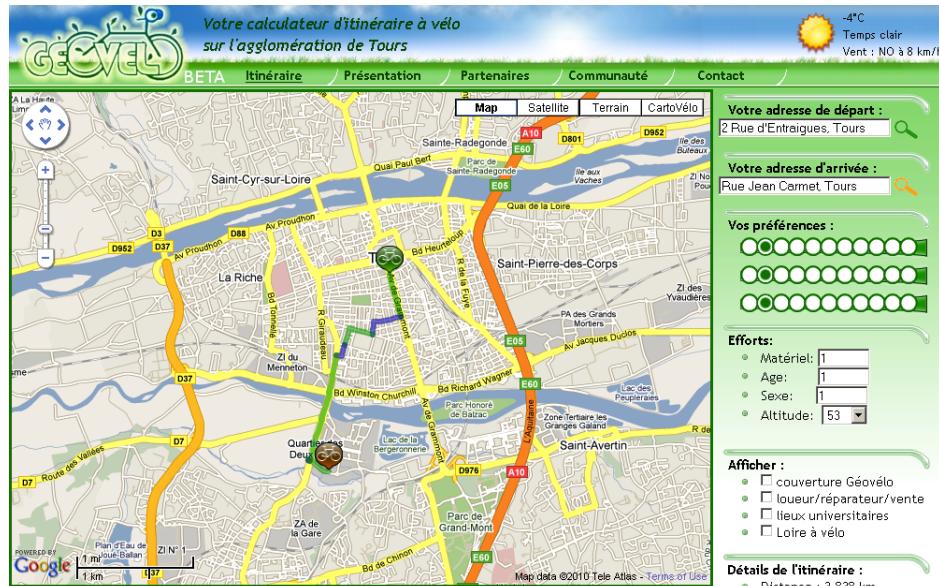


FIGURE 3.12 – Interface originale

Nous pouvons voir la figure ci-dessus, auparavant nous modifions la zone de vos préférences afin de tester la fonctionnalité de nouvel algorithme, elle affiche que 3 barres. Evidement ce n'est pas très claire pour distinguer chaque critère.

Dans la zone de vos préférences, chaque critère a 11 points ronds, donc il y a déjà 33 points ronds pour 3 critère. Le client peut ressentir de la confusion. Pour éviter ça, mon encadrant propose que nous pouvons utiliser SLIDE.

Nous décidons d'utiliser SLIDE de JQUERY. JQUERY est une bibliothèque Javascript libre qui se charge de simplifier le document HTML traversant, la gestion des événements, l'animation, et les interactions Ajax pour le développement web rapide. Il est conçu pour changer la façon dont nous écrivons JAVASCRIPT.

Dans la zone d'efforts, les éléments sont seulement utilisés pour aider l'étudiant de DA pour trouver des bonnes valeurs. A la place, mon encadrant trouve une bibliothèque FLOT pour afficher le critère d'effort.

FLOT est une bibliothèque de traçage pure JAVASCRIPT pour JQUERY. Selon les données fournies, elle peut produire des tracés graphiques. Pour le critère d'effort, le client peut voir la changement d'altitude d'un itinéraire clairement.

## Modification de l'interface

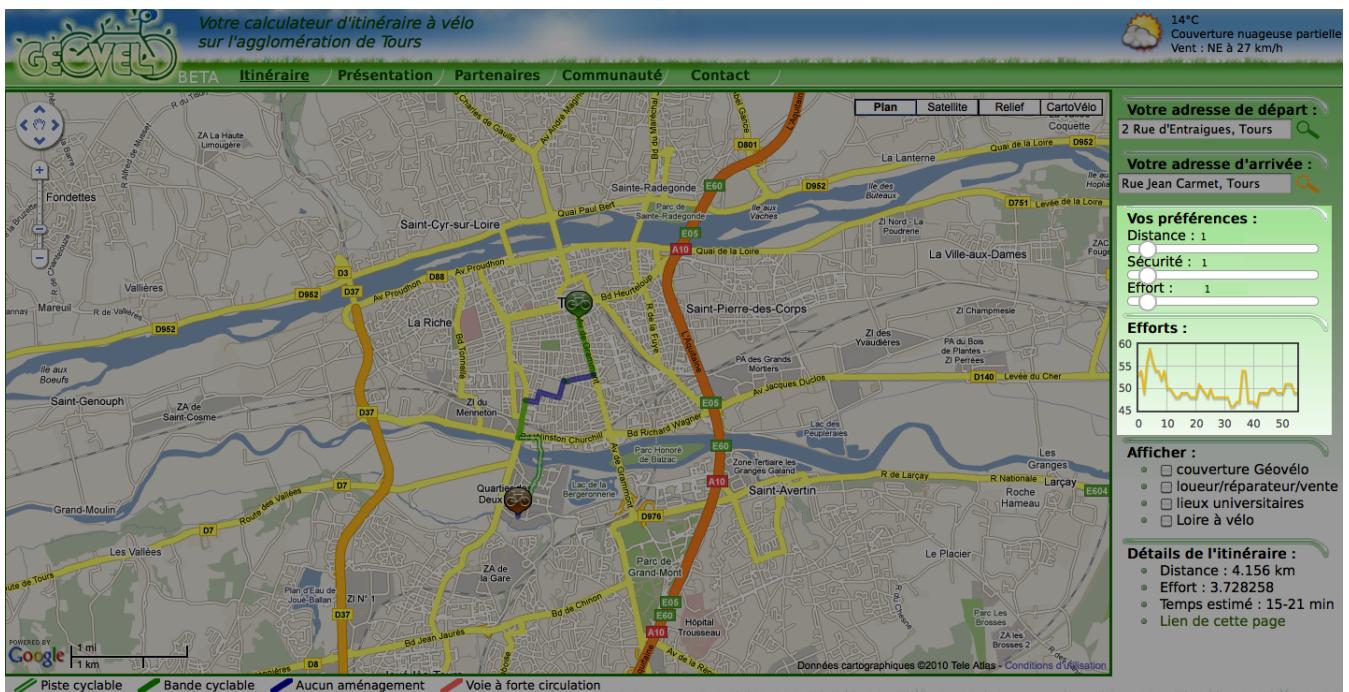


FIGURE 3.13 – Interface améliorée

# Conclusion

---

L'origine de ce projet est réalisé par l'étudiante Juan Zhang (DI5 d'année dernière). Evidement, ça fonctionne bien, mais il perde beaucoup de temps à cause d'utilisation d'une version multi-critère de l'algorithme de Dijkstra. De plus, il considère que deux critères (distance et sécurité).

C'est intéressant d'améliorer le temps du calcul et implémenter les nouveaux critères. Cette année, mon encadrant Gaël Sauvanet a proposé un nouvel algorithme BCA\* qui permet de calculer l'itinéraire plus rapide.

Ce sujet est très intéressant pour moi, parce qu'il concerne des plusieurs domaines (algorithme, communication, base de données, réseau, carte, etc), aussi il m'a permis de découvrir les connaissances dans nombreux technologies tels que LEDA, XmlRPC, Xajax, Javascript, API Google maps.

De plus, pour l'instant le site Géovélo est disponible, chaque jour nombreux de gens peuvent consulter des itinéraires sur ce site. Donc le temps d'exécution et de réponse sont très importants et ont toujours été considérés pendant mon PFE.

L'objectif principale a été atteint, j'ai implémenté le nouvel algorithme BCA\* qui permet de chercher l'itinéraire plus rapide. J'ai ajouté un nouveau critère d'effort et amélioré le critère de sécurité. J'ai modifié l'interface pour adapter le nouvel algorithme et le nouveau critère.

La suite de ce projet consiste à tester toujours pour l'algorithme BCA\* et l'interface. Ensuite modéliser et implémenter le critère de tourisme. Enfin mettre en place le projet sur une interface nouvelle de site Géovélo. C'est une interface plus jolie, mais elle utilise encore l'algorithme original.

A l'issue de ce projet, il me reste donc à remercier tous ceux qui ont permis sa réalisation et sa concrétisation, notamment Gaël Sauvanet, pour l'idée et l'encadrement, et aussi Emmanuel Néron, Nicolas Ragot, pour l'aide de modification de Cahier des Charges.

# Résultat de tests de valeur $\varepsilon$ différent

---

Pour le poids de critère : { 1, 1, 1 } et tests de critère de distance :

epsilon	moyenne	max	min	même	valeur	temps
{ 0.001, 0, 0 }	6.37384975	621.100033	-0.020039	198	2.31709120738146e-05	0.455050065
{ 0.002, 0, 0 }	6.43944979	623.53003	-0.040025	198	2.31709120738146e-05	0.389450025
{ 0.003, 0, 0 }	6.488849645	625.290032	-0.029999	196	0.000270881646140671	0.34005017
{ 0.004, 0, 0 }	6.51764975	626.250029	-0.029999	195	0.00031878629079344	0.311250065
{ 0.005, 0, 0 }	6.536900025	626.980033	-0.030003	194	0.000319831042505924	0.29199979
{ 0.006, 0, 0 }	6.563749895	628.370031	-0.030021	193	0.000365332561392438	0.26514992
{ 0.007, 0, 0 }	6.58659998	628.93003	-0.020023	192	0.000380013176455837	0.242299835
{ 0.008, 0, 0 }	6.60585008	629.440031	-0.020037	190	0.000746068150417435	0.223049735
{ 0.009, 0, 0 }	6.617399765	629.850033	-0.029998	190	0.000769480322391908	0.21150005
{ 0.01, 0, 0 }	6.63454969	630.460034	-0.020032	189	0.00101670637814947	0.194350125
{ 0.02, 0, 0 }	6.638899835	631.900033	-0.020035	184	0.00306715715603449	0.18999998
{ 0.03, 0, 0 }	6.653199795	632.510033	-0.020026	184	0.0031399691675285	0.17570002
{ 0.04, 0, 0 }	6.655749855	632.700033	-0.020024	184	0.0031399691675285	0.17314996
{ 0.05, 0, 0 }	6.659099745	632.69003	-0.029999	184	0.00314024111357441	0.16980007
{ 0.06, 0, 0 }	6.65384983	632.640034	-0.020041	183	0.00355953090701858	0.175049985
{ 0.07, 0, 0 }	6.65669989	632.690034	-0.030021	183	0.00355953090701858	0.172199925
{ 0.08, 0, 0 }	6.656799855	632.710032	-0.020037	183	0.00355953090701858	0.17209996
{ 0.09, 0, 0 }	6.65424977	632.380032	-0.020037	183	0.00355953090701858	0.174650045

TABLE A.1 – Tests de critère de distance

Pour le poids de critère : { 1, 1, 1 } et tests de critère de sécurité :

epsilon	moyenne	max	min	même	valeur	temps
{ 0, 0.001, 0 }	6.50934983	624.360031	-0.029989	194	0.000126151250766353	0.319549985
{ 0, 0.002, 0 }	6.556999615	625.950031	-0.020024	186	0.000517676031018095	0.2719002
{ 0, 0.003, 0 }	6.58639986	626.35003	-0.029997	182	0.000871452785213192	0.242499955
{ 0, 0.004, 0 }	6.621099625	627.710031	-0.019996	180	0.00122308538629602	0.20780019
{ 0, 0.005, 0 }	6.62044978	625.770032	-0.019998	171	0.00215850605318717	0.208450035
{ 0, 0.006, 0 }	6.642649745	626.360033	-0.020036	170	0.00221459807514725	0.18625007
{ 0, 0.007, 0 }	6.66434985	629.070032	-0.019996	166	0.00281554737571807	0.164549965
{ 0, 0.008, 0 }	6.679899935	629.630031	-0.019997	164	0.00304027272225735	0.14899988
{ 0, 0.009, 0 }	6.687199775	630.14003	-0.020057	162	0.00359215734888643	0.14170004
{ 0, 0.01, 0 }	6.693799775	630.83003	-0.010035	161	0.00365988749781361	0.13510004
{ 0, 0.02, 0 }	6.728599895	632.110032	-0.010034	144	0.0123228014545173	0.10029992
{ 0, 0.03, 0 }	6.724649755	632.54003	-0.010033	134	0.0185907225683646	0.10425006
{ 0, 0.04, 0 }	6.73254984	632.690033	-0.010035	128	0.0208145593693717	0.096349975
{ 0, 0.05, 0 }	6.733699865	632.920032	-0.010048	122	0.0245964468618368	0.09519995
{ 0, 0.06, 0 }	6.72994992	632.98003	-0.250006	120	0.0307796223928548	0.098949895
{ 0, 0.07, 0 }	6.716149735	632.940031	-0.150003	116	0.0321668879948513	0.11275008
{ 0, 0.08, 0 }	6.71435002	632.970033	-0.250004	115	0.0337801785934477	0.114549795
{ 0, 0.09, 0 }	6.71269973	632.970032	-0.150006	114	0.0340247791984773	0.116200085

TABLE A.2 – Tests de critère de sécurité

Pour le poids de critère : { 1, 1, 1 } et tests de critère d'effort :

epsilon	moyenne	max	min	même	valeur	temps
{ 0, 0, 0.001 }	6.27789951	618.680031	-0.079998	197	4.26646989132748e-05	0.551000305
{ 0, 0, 0.002 }	6.30009985	618.280028	-0.059998	197	5.31157233170869e-05	0.528799965
{ 0, 0, 0.003 }	6.351699785	619.670035	-0.059999	196	0.000271140959543902	0.47720003
{ 0, 0, 0.004 }	6.41399967	620.370033	-0.039992	195	0.000348708770756573	0.414900145
{ 0, 0, 0.005 }	6.46669997	622.130032	-0.039988	192	0.000587683808172608	0.362199845
{ 0, 0, 0.006 }	6.49714978	623.800034	-0.08999	192	0.000729577201174014	0.331750035
{ 0, 0, 0.007 }	6.524349835	625.630034	-0.12	189	0.00142242780830924	0.30454998
{ 0, 0, 0.008 }	6.541499755	625.710033	-0.039992	189	0.00142242780830924	0.28740006
{ 0, 0, 0.009 }	6.545599945	626.530033	-0.029989	187	0.00188740831131182	0.28329987
{ 0, 0, 0.01 }	6.553299725	626.410031	-0.039996	187	0.00188740831131182	0.27560009
{ 0, 0, 0.02 }	6.625199655	628.620032	-0.019995	179	0.00977916889062763	0.20370016
{ 0, 0, 0.03 }	6.65999986	630.40003	-0.020022	174	0.0127204406233615	0.168899955
{ 0, 0, 0.04 }	6.67749989	630.500032	-0.010045	172	0.0150281496598562	0.151399925
{ 0, 0, 0.05 }	6.683699905	631.020032	-0.059989	172	0.0157710817158272	0.14519991
{ 0, 0, 0.06 }	6.69124971	631.020033	-0.010036	172	0.0157710817158272	0.137650105
{ 0, 0, 0.07 }	6.692799845	631.300031	-0.010036	172	0.0157710817158272	0.13609997
{ 0, 0, 0.08 }	6.69719978	631.290032	-0.020005	172	0.0157710817158272	0.131700035
{ 0, 0, 0.09 }	6.696599735	631.540033	-0.010037	171	0.015933305206054	0.13230008

TABLE A.3 – Tests de critère d'effort

# Guide - Installation et configuration

---

## B.1 Installation des packages nécessaires

Nous travaillons sous le système de Ubuntu. Pour installer les paquets et les outils ci-dessous, nous pouvons utiliser deux façons :

1. Utiliser l'outil Gestionnaire de paquets Synaptic
2. Utiliser la ligne de commande sudo apt-get install dans le terminal.

Nom	Paquets
Apache	apache2, apache2.2-common, apache2-mpm-prefork, apache2-utils
PHP	libapache2-mod-php5, php5, php5-common
PostgreSQL	php5-psql, postgresql-8.3-postgis, libpq5, libpq-dev
XMLRPC	libxmlrpc++0, libxmlrpc++-dev
pgAdmin	pgadmin3
CodeBlocks	codeblocks, codeblocks-contrib, libcodeblocks0, libwxsmithlib0
QGIS	qgis, libqgis1-dev

Si votre version de Ubuntu est supérieure à 8.04, vous devez installer deux paquets supplémentaires gcc-4.2 et g++-4.2, parce que LEDA ne peut s'exécuter seulement qu'en version 4.2.

## B.2 Configuration de la base de données

PostgreSQL est un serveur qui permet de se connecter à différentes bases de données, par défaut il n'existe qu'un utilisateur postgres qui peut se connecter, pour créer un nouvel utilisateur et une nouvelle base de données, nous devons utiliser ce compte.

```
$ sudo su postgres
$ psql
postgres=# create user 'nom_utilisateur';
postgres=# alter role 'nom_utilisateur' with createdb;
postgres=# create database 'nom_utilisateur';
postgres=# alter user 'nom_utilisateur' with encrypted password 'mot_de_passe';
postgres=# \q
postgres@ubuntu :~$ exit
nom_utilisateur@ubuntu :~\$ psql
```

Nous exécutons les commandes ci-dessus, après nous voyons que la ligne de commande est devenue nom\_utilisateur=>, cela signifie que nous ne sommes pas un superutilisateur, pour changer le droit d'utilisateur, nous utilisons la commande ci-dessous.

```
postgres=# alter user 'nom_utilisateur' superuser;
```

Pour configurer notre base de données, nous utilisons les commandes ci-dessous. La commande createlang permet d'ajouter le langage de procédures PL/pgSQL sur la base de données.

```
$ cd /usr/share/postgresql-8.3-postgis
$ createlang plpgsql 'nom_base_de_données'
```

**PL/psql** : (Procedural Language/PostgreSQL Structured Query Language) est un langage procédural soutenu par le PostgreSQL. C'est facile à utiliser. Nous pouvons l'utiliser pour augmenter la structure de contrôle pour le langage SQL ; effectuer des calculs complexes ; hériter de tous les types définis par utilisateurs, les fonctions et opérateurs. De plus, nous pouvons utiliser toutes les choses que nous définissons dans la fonction de langage C.

Le fichier `lwpostgis.sql` contient des divers types spatiaux et fonctions. Nous utilisons la commande ci-dessous pour l'ajoutons à notre base de données.

```
$ psql -d 'nom_base_de_données' -f lwpostgis.sql
```

Le fichier `spatial_ref_sys.sql` contient des coordonnées spatiales. Nous utilisons la commande ci-dessous pour l'ajoutons à notre base de données.

```
$ psql -d 'nom_base_de_données' -f spatial_ref_sys.sql
```

Nous utilisons les commandes ci-dessous pour autoriser l'utilisateur à utiliser et modifier toutes les tables dans la base de données.

```
# alter table spatial_ref_sys owner to 'nom_utilisateur';
'superutilisateur'=# alter table geometry_columns owner to 'nom_utilisateur';
```

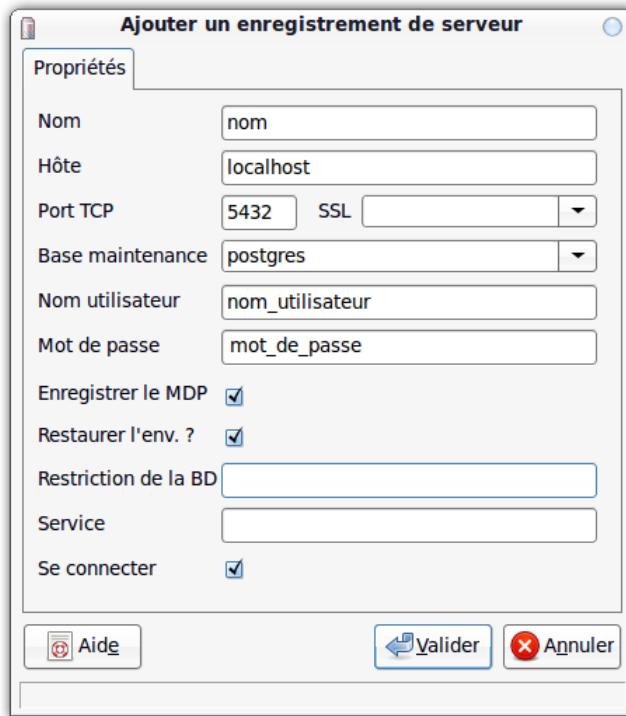


FIGURE B.1 – Ajouter un serveur dans l'outil pgAdmin

Nous ouvrons l'outil pgAdmin3 et cliquons sur "Fichier -> Ajouter un serveur" pour se connecter au nouveau serveur.

Nous importons nos bases de données existantes, s'il existe des erreurs, nous pouvons changer le nom d'utilisateur précédent en notre propre nom.

## B.3 Configuration du site web Géovélo

Nous entrons "http ://localhost/" dans un navigateur, il affiche "It works!". Cela veut dire que notre serveur fonctionne. Nous pouvons copier le répertoire du site web Géovélo "geovelo" dans le répertoire "/var/www/".

```
$ sudo cp -a geovelo /var/www/
```

Dans le répertoire "/var/www/", nous ajoutons tous les droits à le répertoire "geovelo".

```
$ sudo chmod 777 geovelo  
$ sudo chmod 777 -R geovelo  
$ cd geovelo/  
$ sudo chmod 777 *
```

Dans le répertoire "/var/www/geovelo/acces\_donnees/", nous changeons la configuration de fichier "cDatabase.php" en fonction de notre configuration. Et puis, nous utilisons "http ://localhost/geovelo/" pour consulter notre site web.

En ce moment nous avons bien configuré notre base de données et site web, nous nous préparerons à lancer le serveur XMLRPC.

## B.4 Configuration de LEDA

Nous téléchargeons le paquet de LEDA depuis le site officiel, ensuite décompressons ce fichier dans le répertoire "/home/'nom\_utilisateur'/LEDA".

**LEDA** : (The Library of Efficient Data types and Algorithms) est une librairie C++ qui offre une grande variété d'algorithmes et de structures adaptés aux graphes et aux calculs géométriques. Dans mon projet il est nécessaire de calculer un itinéraire sur la carte, donc c'est plus facile de calculer l'itinéraire en utilisant cette librairie existante.

```
$ gedit .bashrc
```

Ajoutons "export LEDAROOT=/home/'nom\_utilisateur'/LEDA" après la ligne "export HISTCONTROL=ignoreboth".

Nous ouvrons notre projet existant ServerXmlRpc.cbp en utilisant l'outil CodeBlocs. A la ligne 35 de fichier loadData.cpp, changeons la configuration de la base de données. Utilisons le clic droit de la souris sur le projet "ServerXmlRpc", choisissons "Build options..." et cliquons sur "ServerXmlRpc" pour modifier le répertoire dans le "Linker settings" et "Search directories".

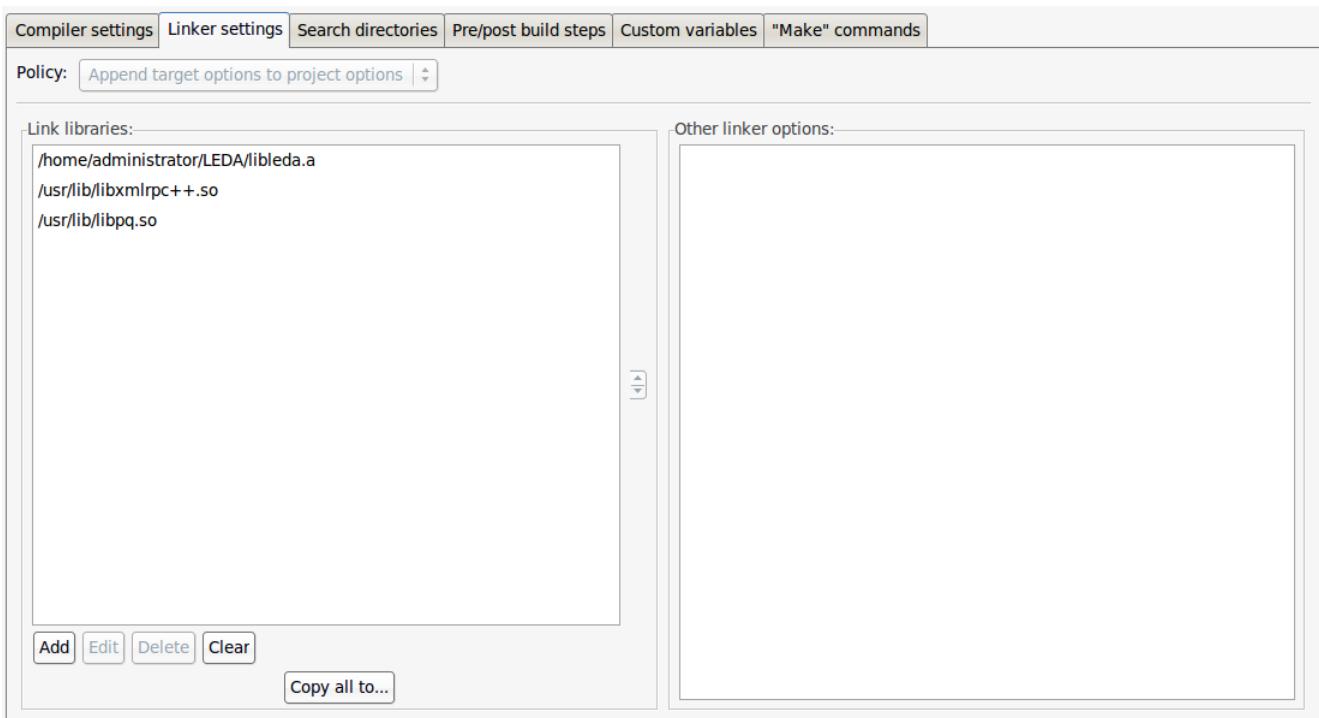


FIGURE B.2 – Linker settings dans l’outil codeblocs

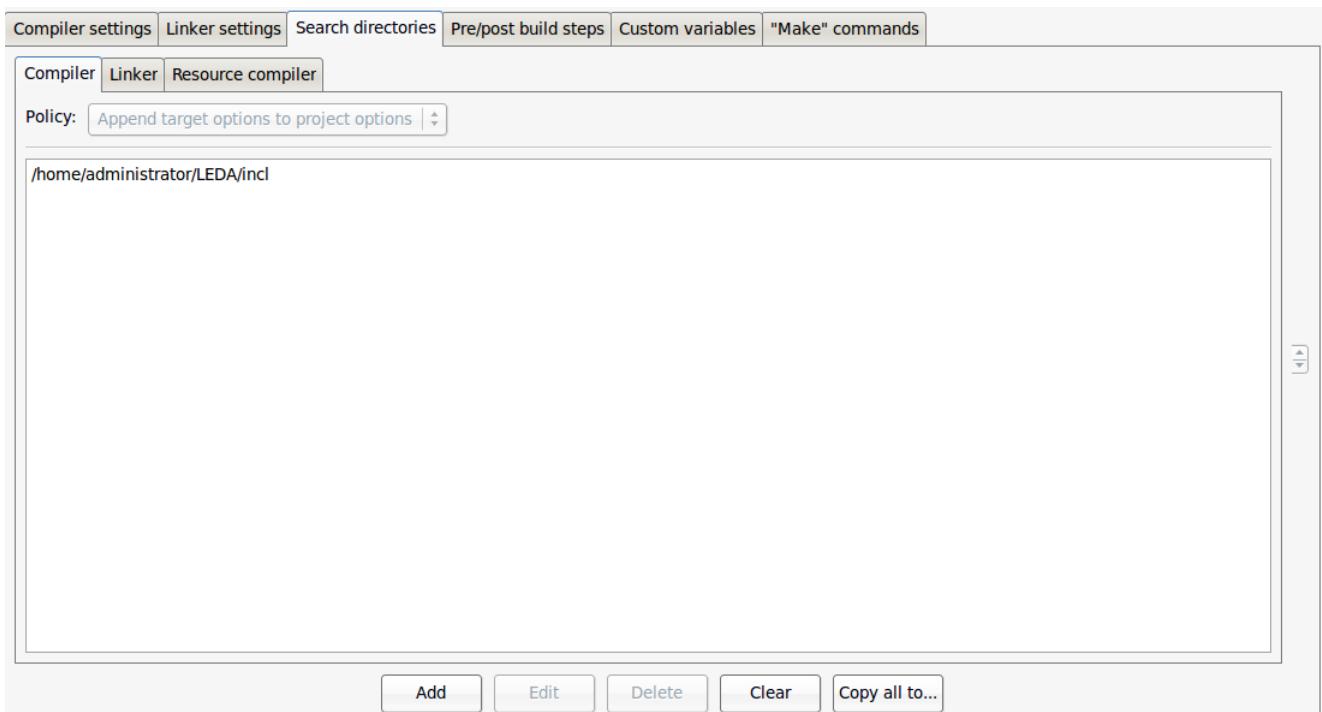


FIGURE B.3 – Search directories dans l’outil codeblocs

Ensuite, Cliquons sur "settings", choisissons "Compiler and debugger" pour définir notre version de g++ par défaut. Nous pouvons le modifier avec l'aide de l'image ci-dessous.

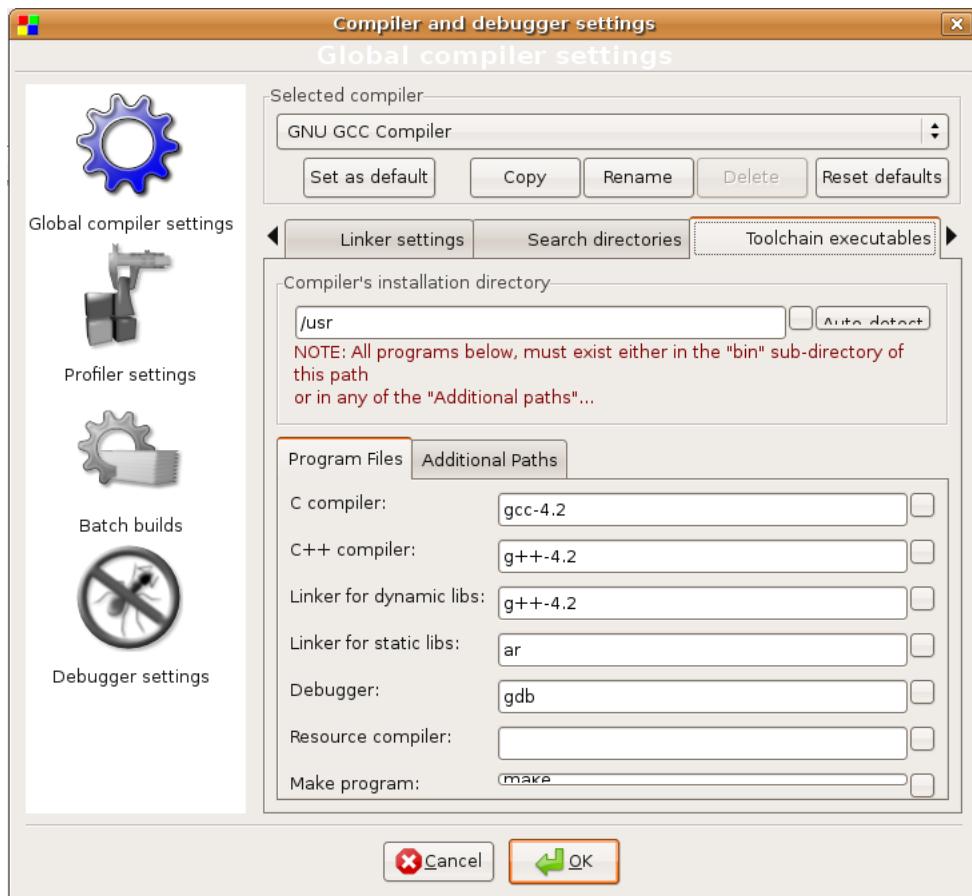


FIGURE B.4 – Complier and debugger settings

Enfin nous lançons le serveur XMLRPC, quand le terminal affiche l'image ci-dessous, ça signifie que notre serveur marche bien, en ce moment nous avons réussi à installer Géovélo.

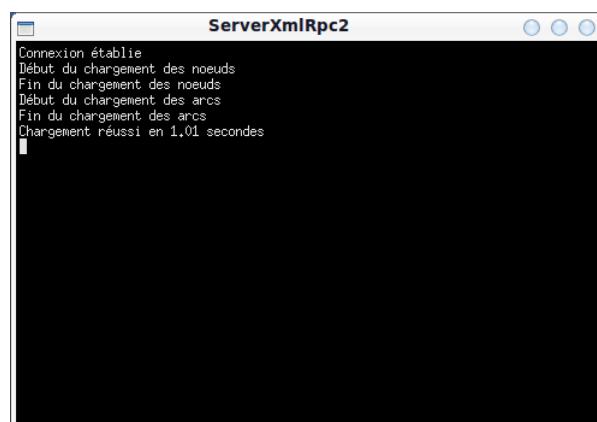


FIGURE B.5 – ServerXmlRpc

Pour le serveur à distance, nous pouvons créer un script pour compiler et exécuter notre programme. Par exemple, nous créons un script `script.sh`.

```
g++-4.1 -Wall -O2 -Wall -I/home/nobody/LEDA-6/incl -c TestBCAstar.cpp  
        -o obj/Release/TestBCAstar.o  
  
g++-4.1 -Wall -O2 -Wall -I/home/nobody/LEDA-6/incl -c AStarBi.cpp  
        -o obj/Release/AStarBi.o  
  
g++-4.1 -Wall -O2 -Wall -I/home/nobody/LEDA-6/incl -c loadData.cpp  
        -o obj/Release/loadData.o  
  
g++-4.1 -L/usr/lib -o bin/Release/TestBCAstar obj/Release/AStarBi.o  
        obj/Release/loadData.o obj/Release/TestBCAstar.o  
        -s /home/nobody/LEDA-6/libleda.a /usr/lib/libpq.so /usr/lib/libxmlrpc  
        ++.so
```

Ensuite, nous pouvons utiliser la commande `nohup` qui permet de lancer un processus qui restera actif même après la déconnexion de l'utilisateur l'ayant initiée. Par exemple, nous sommes dans le répertoire `"./bin/Release"`, nous pouvons utiliser la commande suivante pour lancer notre serveur.

```
$ nohup ./TestBCAstar &
```

# Cahier des Charges - Introduction

---

Lors de la cinquième année du cycle d'ingénieur de l'École Polytechnique de l'Université de Tours Département Informatique. Nous devons effectuer un Projet de Fin d'Étude (PFE) du 10/2009 au 5/2010. Le but est de familiariser les étudiants avec les compétences acquises en cours et d'en acquérir de nouvelles.

Le sujet du projet est "Nouveaux critères pour du calcul d'itinéraire vélo". Ce projet est basé sur des travaux de mon encadrant Gaël Sauvanet et l'étudiante Juan Zhang dans le cadre du projet Géovélo réalisé par l'association Autour du Train. La première étape de ce PFE est d'établir le cahier des charges pour délimiter la réalisation du projet.

# Contexte de la réalisation

---

## D.1 Contexte

Géovélo est un site qui permet de calculer des itinéraires adaptés au réseau cyclable. Nous pouvons choisir le départ et la destination. Entre les deux points Géovélo nous fournit des itinéraires en tenant compte de différents critères. L'utilisateur peut choisir un itinéraire selon ses préférences de distance et de sécurité. Géovélo est une première en France. L'interface cartographique est basée sur API GOOGLE MAPS. Pour l'instant, il n'est disponible que pour la ville de Tours.

## D.2 Objectifs

L'objectif du projet sera d'améliorer l'algorithme, nous allons mettre en place BCA\* ALGORITHM[FP00] pour chercher des itinéraires plus rapidement. Et puis nous devrons implémenter les nouveaux critères avec l'aide de l'outil POSTGIS. Selon le temps restant, nous devrons modifier l'interface du site web.

L'algorithme de ce projet est développé en utilisant le plus court chemin (une version multi-critère de l'algorithme de Dijkstra). Évidemment cet algorithme fonctionne bien mais il perd beaucoup de temps. Pour calculer un itinéraire entre le départ et la destination selon différents critères, il a besoin de trouver toutes les solutions au premier, et puis parmi les résultats il retient quelques itinéraires satisfaisants les besoins de l'utilisateur. Pour l'améliorer mon encadrant propose un nouveau algorithme : BCA\* (une généralisation d'A\* pour la recherche de solutions de compromis dans des problèmes d'optimisation multi-objectifs). C'est un algorithme qui est spécialement conçu pour la recherche de solutions de compromis. L'intérêt est que nous n'avons pas besoin de calculer tous les résultats, le mécanisme de BCA\* permet d'enlever des points qui ne nous intéressent pas pour trouver une solution plus rapidement en réduisant l'espace de recherche.

# Description générale

---

## E.1 Environnement du projet

Ce PFE est effectué sous le système d'exploitation Ubuntu8.

Le codage doit être réalisé en C++. Sous Ubuntu8, on installe l'outil codeblocks comme environnement de développement et les paquets concernant la librairie de communication XMLRPC.

On utilise POSTGRESQL comme SGBD et son module géographique Postgis. On utilise l'outil PGADMIN3 pour gérer la base de données et l'outil qgis pour visualiser et traiter les données géographiques.

On utilise le serveur APACHE2 et son module PHP5 comme serveur web.

## E.2 Caractéristiques des utilisateurs

Le site Géovélo réalisé au cours de ce projet est destiné en premier lieu aux utilisateurs du site.

Un utilisateur du site du Géovélo possède des caractéristiques :

- Connaissance ou non de l'informatique : Non ;
- Expérience de l'application : Oui ;
- Utilisateurs réguliers et/ou occasionnels : Occasionnel ;
- Droits d'accès utilisateurs : Tous les droits sur le site Géovélo .

## E.3 Fonctionnalités et structure générale du système

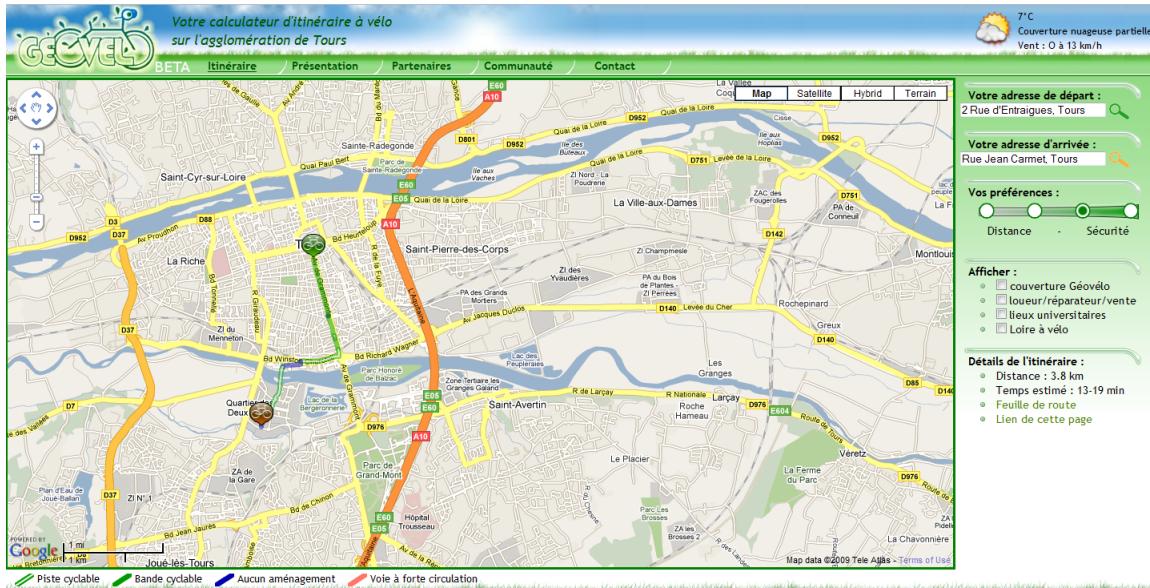


FIGURE E.1 – Le site Géovélo

L'image ci-dessus est triée du site web Géovélo actuel. Sur ce site, les fonctionnalités suivantes sont possibles :

### Sélection d'un itinéraire

Pour saisir un itinéraire, il existe 3 méthodes :

- Saisir une adresse de départ ou d'arrivée puis cliquer sur la loupe ;
- Déplacer les curseurs de départ et d'arrivée directement sur la carte à l'aide de la souris ;
- Utiliser le clic droit de la souris sur la carte.

### Sélection d'une préférence

Permet de choisir un itinéraire selon ses préférences, de distance et de sécurité.

### Affichage d'informations géographiques

- Couverture Géovélo : Afficher un cadre qui permet de visualiser la zone géographique couverte par Géovélo ;
- Loueur/réparateur/vente : Afficher l'ensemble des professionnels du vélo ;
- Lieux universitaires : Afficher les principaux lieux universitaires sur l'agglomération. On peut cliquer sur chaque point d'intérêt pour obtenir des informations supplémentaires ;
- Loire à vélo : Afficher le tracé de la Loire à vélo.

### Affichage des détails de l'itinéraire

Après avoir saisi le départ et la destination, Géovélo nous donne la solution avec la distance et le temps estimé. Nous pouvons cliquer sur la feuille de route pour obtenir des informations détaillées. Nous pouvons cliquer sur "lien de cette page" pour obtenir un lien vers un itinéraire de manière à pouvoir le partager par mail ou autre.

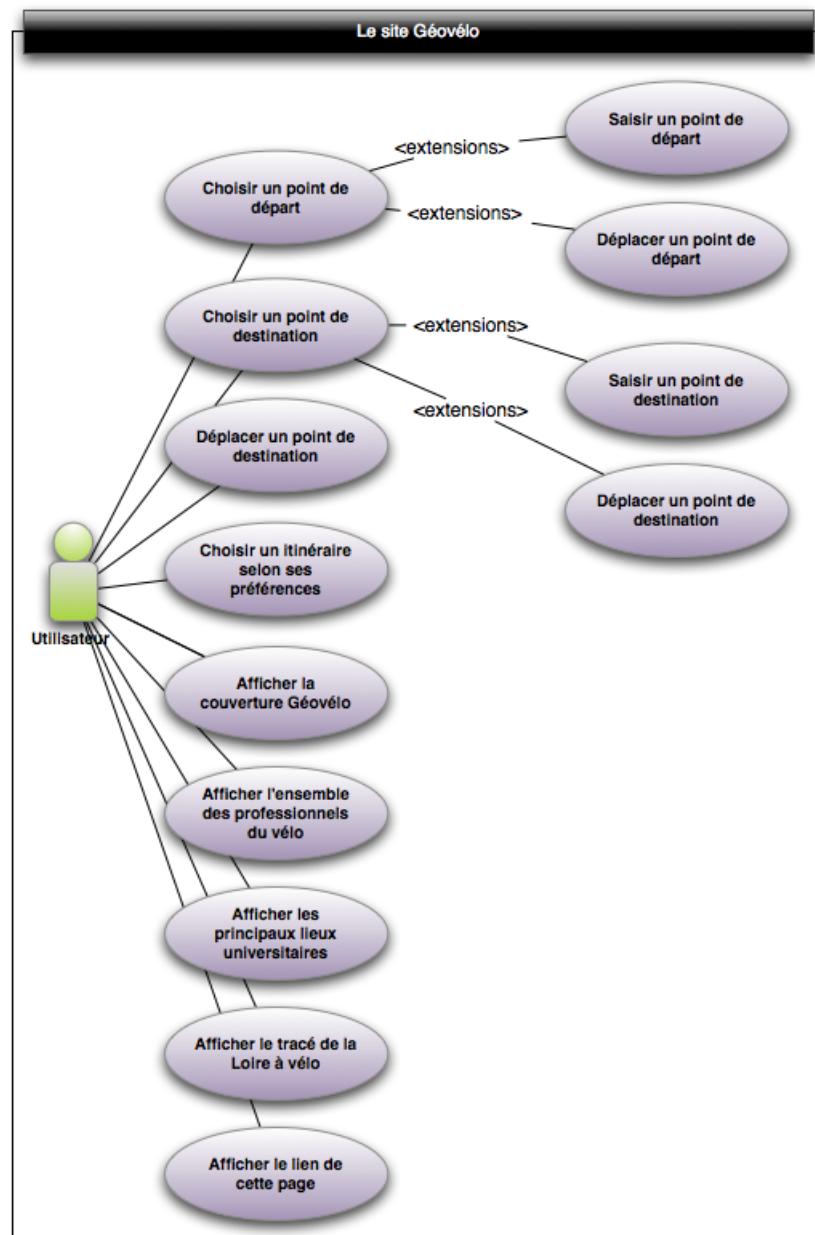


FIGURE E.2 – Le diagrammes de cas d'utilisation de Géovélo

## E.4 Contraintes de développement, d'exploitation et de maintenance

### Contraintes de développement

Pendant ce projet de fin d'étude, il est nécessaire d'utiliser différents langages et outils :

Calcul d'itinéraires : C++, LEDA

Site web : HTML, CSS, PHP, AJAX, API GOOGLE MAPS

Base de données : POSTGRESQL, POSTGIS

Parmi ces technologies, je ne connais pas encore LEDA, AJAX, API GOOGLE MAPS, POSTGRESQL,

POSTGIS, donc je vais essentiellement les présenter ci-dessous.

- LEDA : LEDA (The Library of Efficient Data types and Algorithms) est une librairie C++ qui offre une grande variété d'algorithmes et de structures adaptés aux graphes et aux calculs géométriques. Dans mon projet il est nécessaire de calculer un itinéraire sur la carte, donc c'est plus facile de calculer l'itinéraire en utilisant cette librairie existante.
- AJAX : AJAX (Asynchronous JavaScript and XML) est une technique de développement web. Il permet de créer des applications web interactives. Le plus gros avantage est qu'avec l'utilisation de AJAX, il est possible de maintenir des données sans besoin de rafraîchir la page toute entière. Les applications web peuvent rendre une réponse à l'utilisateur plus rapidement et éviter d'envoyer les informations qui n'ont pas besoin d'être modifiées.
- XAJAX : XAJAX est un framework open source utilisant le langage PHP, il permet d'utiliser le langage PHP pour développer des fonctionnalités AJAX sans connaître le langage JavaScript.
- API GOOGLE MAPS : Nous savons que Google Maps est un outil utilisé pour suivre l'emplacement actuel, les conditions en temps réel du trafic, mais il offre aussi un outil de recherche, et l'API Google Maps permet d'intégrer Google Maps dans son propre site en utilisant JavaScript. Ensuite il est possible de développer avec JavaScript pour manipuler la carte.
- POSTGRESQL : PostgreSQL est un système de gestion de base de données relationnelles. Actuellement, il est un des systèmes les plus puissants, le plus riche en fonctionnalités et le plus complexe des bases de données du logiciel libre. Il fonctionne principalement sous le système UNIX (Linux, FreeBSD, Ubuntu...).
- POSTGIS : PostGIS est une extension qui ajoute la capacité de gestion des données spatiales sur le système PostgreSQL. Il offre des fonctionnalités comme les objets spatiaux, l'indexation spatial, les fonctions de manipulation spatiales et les opérateurs. En résumé il permet de manipuler des données géographiques facilement.

# Architecture générale du système

---

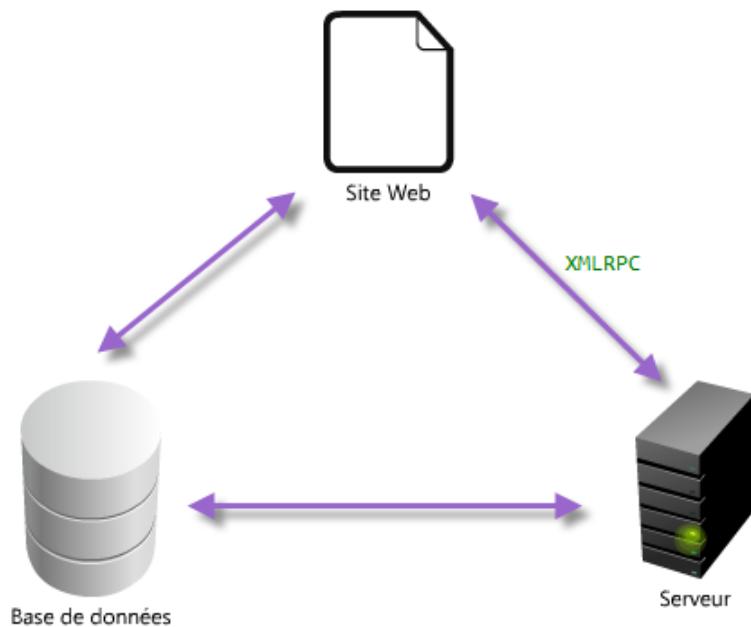


FIGURE F.1 – L'architecture principale de Géovélo

L'image ci-dessus est l'architecture principale de Géovélo. Sur le site web Géovélo, quand l'utilisateur consulte des informations statiques qui n'ont pas besoin de calculer d'itinéraires (par exemple affichage de la couverture, affichage de la Loire à vélo etc), le site web Géovélo peut interroger directement la base de données.

Quand l'utilisateur demande un itinéraire, le site web Géovélo doit adopter la deuxième méthode : il va utiliser le protocole RPC pour communiquer avec le serveur de calcul d'itinéraire. Le serveur reçoit les requêtes, et puis il traite les données et retourne le résultat au site web.

XMLRPC : RPC (Remote procedure call) est une technique qui permet à deux machines de communiquer ou d'échanger des données au format XML. Il est utilisé pour échanger des informations entre des pages du site et le serveur de calcul d'itinéraire.

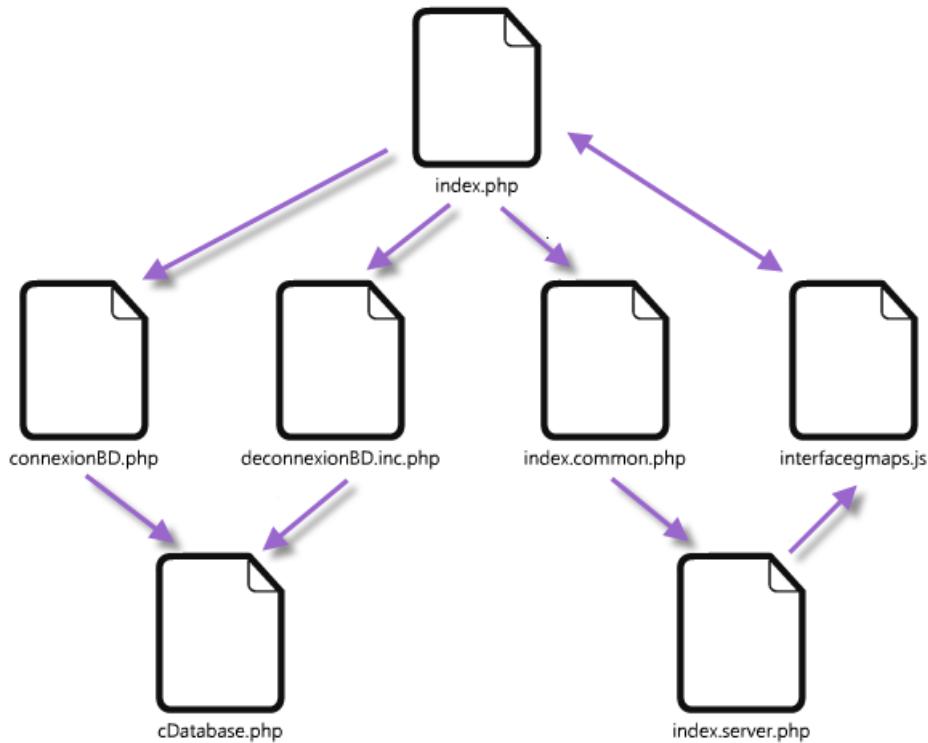


FIGURE F.2 – L'architecture du site Géovélo

Les fichiers principaux sont :

- `index.php` : La page d'accueil de site Géovélo qui affiche les éléments principaux et appelle les autres pages ;
- `connexionBD.inc.php` : La page consiste à se connecter à la base de données ;
- `deconnexionBD.inc.php` : La page consiste à se déconnecter de la base de données ;
- `cDatabase.php` : La page consiste à créer une classe de gestion d'une connexion. Cette classe permet de gérer toutes les connexions vers la base de données ;
- `index.common.php` : La page consiste à déclarer des librairies, instancier des objets XAJAX ;
- `interfacegMaps.js` : La page consiste à gérer des codes JAVASCRIPT pour l'interface Google maps ;
- `index.server.php` : La page consiste à créer des fonctions XAJAX de calcul d'itinéraire, appeler la fonction XMLRPC du serveur pour calculer un ensemble de solutions.

# Description des Besoins

---

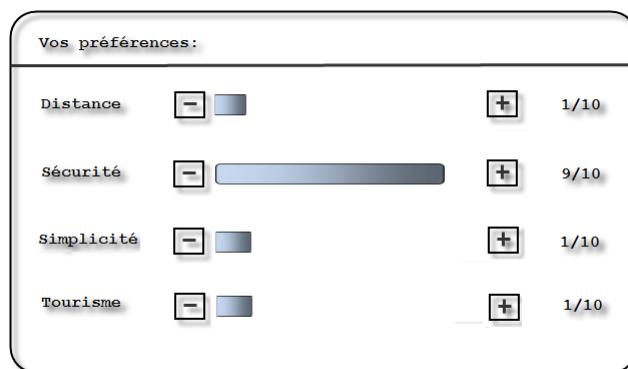
## G.1 Besoins fonctionnels

Nous avons identifié les besoins fonctionnels suivants. Ces besoins identifient le travail à réaliser dans le cadre du projet .

- Besoin 1 : Implémenter BCA\* ALGORITHM en C++ sous la forme d'un web service ;  
Il consiste à lire d'abord des sources et comprendre la fonctionnalité d'algorithme existant, et puis implémenter BCA\* ALGORITHM pour deux critères (distance et sécurité), enfin tester la fonctionnalité et la compatibilité ;

BCA\* ALGORITHM est une méthode multi-objective. Cet algorithme prend en paramètre les préférences de l'utilisateur (poids sur chaque critère), après il retourne l'itinéraire le plus adapté aux préférences de l'utilisateur. Cet algorithme explore en priorité les labels les plus "prometteurs" et éliminent ceux qui ne sont pas intéressants.

- Besoin 2 : Ajouter des nouveaux critères ;  
Cette tâche consiste à modéliser de nouveaux critères (difficulté, simplicité, attrait touristique) en collaboration avec 3 étudiants du DA dont leur sujet porte sur la sécurité, l'effort et le tourisme, après on va récupérer les données et ajouter les attributs nécessaires sur la base de données avec l'aide de l'outil POSTGIS, enfin nous utilisons BCA\* ALGORITHM tester la fonctionnalité et la compatibilité ;
- Besoin 3 : Modifier l'interface du site web.  
Nous allons implémenter un nouvel algorithme et ajouter les nouveaux critères pour le site web Géovélo, donc il faut utiliser le langage PHP/AJAX/CSS pour modifier la zone des préférences. Le modèle est comme l'image ci-dessous ;



Le formulaire intitulé "Vos préférences:" contient quatre séries de sliders pour ajuster le poids de différents critères. Chaque critère est accompagné d'un curseur horizontal, d'un bouton "-" à gauche et d'un bouton "+" à droite, ainsi que d'une valeur numérique à sa droite.

Critère	Min	Max	Valeur
Distance	-	+	1/10
Sécurité	-	+	9/10
Simplicité	-	+	1/10
Tourisme	-	+	1/10

FIGURE G.1 – Le modèle de vos préférence

## G.2 Besoins non fonctionnels

- Besoin 4 : En terme de fiabilité, le site web Géovélo devra être stable et devra supporter une charge d'environ 5000 visiteurs / mois ;
- Besoin 5 : La documentation d'installation et de configuration doit être rédigée ;
- Besoin 6 : Les commentaires du code source doivent être lisibles et compréhensibles ;
- Besoin 7 : Le temps de calcul d'un itinéraire doit être inférieur à 3 secondes ;
- Besoin 8 : L'interface du site web Géovélo doit être conviviale et ergonomique .

# Découpage du projet en tâches

---

Ce projet sera séparé en 10 tâches :

- Tâche 1 : La familiarisation avec les algorithmes ;
- Tâche 2 : La familiarisation avec le fonctionnement de Géovélo ;
- Tâche 3 : Configuration de l'environnement ;
- Tâche 4 : Rédaction du guide ;
- Tâche 5 : Étude de l'existant ;
- Tâche 6 : Implémentation du BCA\* ALGORITHM ;
- Tâche 7 : Prise en main POSTGIS/POSTGRESQL ;
- Tâche 8 : Modéliser les critères ;
- Tâche 9 : Implémentation des nouveaux critères ;
- Tâche 10 : Prise en main AJAX ;
- Tâche 11 : Modification de l'interface site web ;
- Tâche 12 : Rédaction du rapport .

## H.1 Tâche 1 : La familiarisation avec les algorithmes

### Description de la tâche

Il s'agit de comprendre les algorithmes du projet. Le projet précédent est développé en utilisant le plus court chemin (une version multi-critère de l'algorithme de Dijkstra [Dij59]). Tout d'abord, je dois bien comprendre le mécanisme de DIJKSTRA ALGORITHM. Et puis, je dois lire le document de l'algorithme de BCA\*[FP00] pour implémenter le nouvel algorithme pour ce projet. Cette partie est la plus importante pour faciliter les tâches suivantes .

### Estimation de charge

Cette partie est la plus importante, dont je passerai beaucoup de temps. Pour cette partie, de 23-09-09 à 25-10-09 .

## H.2 Tâche 2 : La familiarisation de le fonctionnement de Géovélo

### Description de la tâche

Cette tâche consiste à comprendre le fonctionnement de Géovélo. Lire le rapport de PFE de l'année dernière[ ?], comprendre l'architecture et les fonctionnalités.

### Estimation de charge

De 09-10-09 à 25-10-09

### H.3 Tâche 3 : Configuration de l'environnement

#### Description de la tâche

Parce que le projet fonctionne sous Ubuntu, cette tâche consiste à configurer l'environnement de Ubuntu et à installer les outils sous Ubuntu.

#### Estimation de charge

De 01-10-09 à 08-10-09

### H.4 Tâche 4 : Rédaction du guide

#### Description de la tâche

Cette tâche consiste à rédiger la documentation pour expliquer comment mettre en place le projet sous Ubuntu.

#### Estimation de charge

De 05-10-09 à 08-10-09

### H.5 Tâche 5 : Étude de l'existant

#### Description de la tâche

Cette tâche consiste à lire les sources et à comprendre le fonctionnement de chaque fichier.

#### Estimation de charge

De 09-10-09 à 25-10-09

### H.6 Tâche 6 : Implémentation du nouvel algorithme

#### Description de la tâche

C'est la partie de codage. On va d'abord implémenter BCA\* ALGORITHM pour deux critères (distance et sécurité), et puis tester la fonctionnalité de ce nouvel algorithme, on va aussi comparer les deux algorithmes.

Par exemple, on va créer un programme simple qui pourra lancer 1000 tests pour calculer des itinéraires différents en l'algorithme original et l'algorithme BCA\*, puis on pourra mettre les résultats dans un fichier. Il contient des données des itinéraires et des temps de calcul. Alors, on pourra utiliser des données pour vérifier la exactitude et vérifier si cet algorithme est meilleur que l'algorithme original.

#### Estimation de charge

De 26-10-09 à 15-12-09

### H.7 Tâche 7 : Prise en main PostGIS

#### Description de la tâche

Cette tâche consiste à prendre en main l'outil PostGIS.

**Estimation de charge**

De 25-12-09 à 30-12-09

## H.8 Tâche 8 : Modéliser les critères

**Description de la tâche**

Cette tâche consiste à modéliser de nouveaux critères en collaboration avec 3 étudiants du DA dont leur sujet porte sur la sécurité, l'effort et le tourisme. Et puis on va ajouter les nouveaux critères dans la base de données avec l'aide de l'outil PostGIS. Cela implique de récupérer et d'intégrer de nouvelles données.

**Estimation de charge**

De 25-12-09 à 30-12-09

## H.9 Tâche 9 : Implémentation des nouveaux critères

**Description de la tâche**

Cette tâche consiste à modifier les sources de la méthode BCA\* pour adapter les nouveaux critères.

**Estimation de charge**

De 31-12-09 à 19-02-10

## H.10 Tâche 10 : Prise en main AJAX

**Description de la tâche**

Dans cette partie, on va commencer par modifier l'interface du site web Géovélo. Il faut connaître les langages HTML/CSS/PHP/AJAX. Parmi ces langages, je ne connais pas le langage AJAX, donc je dois prendre en main AJAX.

**Estimation de charge**

De 26-02-10 à 02-03-10

## H.11 Tâche 11 : Modification de l'interface du site web

**Description de la tâche**

Dans ce projet, on va ajouter les nouveaux critères pour le site web Géovélo, donc on doit modifier la zone des préférences. Elle consiste à ajouter des champs et boutons pour les nouveaux critères. Ensuite on devra tester ce site sur les différents navigateurs existants (Firefox, InternetExplorer, Opera, Safari...).

**Estimation de charge**

De 03-03-10 à 15-03-10

## H.12 Tâche 12 : Rédaction du rapport

### Description de la tâche

Cette tâche consiste à écrire le rapport et le guide d'installation .

### Estimation de charge

De 09-04-10 à 22-04-10

# Planning

---

Afin de réaliser ce projet, il faut respecter les échéanciers suivants.

### Gantt : Gantt Chart : Project

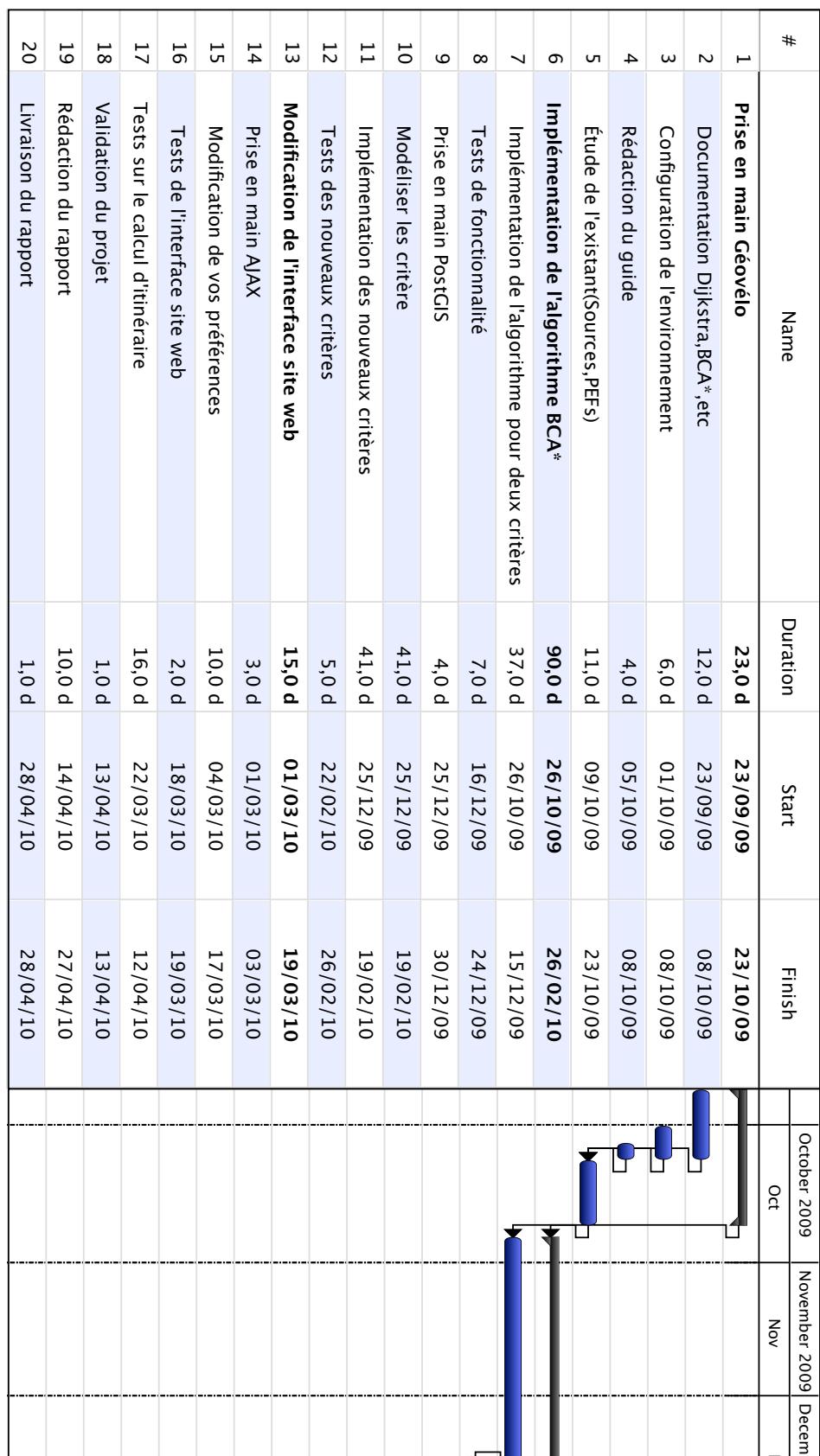
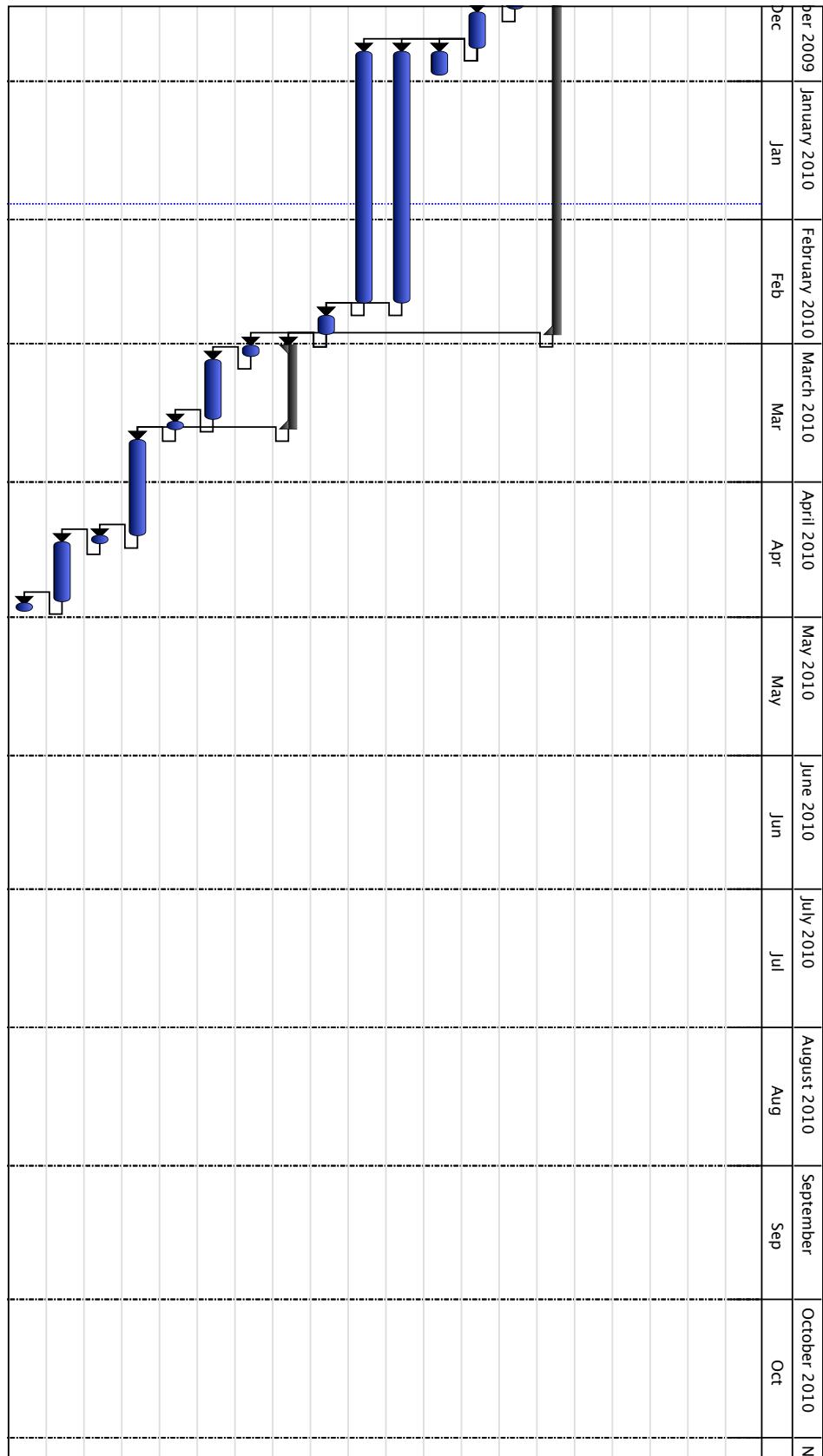


FIGURE I.1 – Diagramme de gantt du projet

**Gantt : Gantt Chart : Project**

**FIGURE I.2 – Diagramme de gantt du projet**

# Bibliographie

---

- [Dij59] E.W. DIJKSTRA. « Two problems in connection with graphs ». *Numerische Mathematik*, 269–271, 1959.
- [FP00] M. FUTTERSACK et P. PERNY. « Une généralisation d'A\* pour la recherche de solutions de compromis dans des problèmes d'optimisation multi-objectifs ». In *Actes de la conférence RFIA 2000*, 3 :377–386, 2000.
- [SW91] B. S. STEWART et C.C. WHITE. Multiobjective A\*. The Association for Computing Machinery, 1991.
- [Wika] WIKIPEDIA. Multiobjective optimization [online]. Disponible sur : [http://en.wikipedia.org/wiki/Multiobjective\\_optimization](http://en.wikipedia.org/wiki/Multiobjective_optimization).
- [Wikb] WIKIPEDIA. Pareto efficiency [online]. Disponible sur : [http://en.wikipedia.org/wiki/Pareto\\_efficiency](http://en.wikipedia.org/wiki/Pareto_efficiency).

# Nouveaux critères pour du calcul d'itinéraire vélo

---

Département Informatique

5<sup>e</sup> année

2009 - 2010

Projet de Fin d'Etude

**Résumé :** Le projet de fin d'étude se déroule durant l'année 2009-2010. L'objectif de projet de fin d'étude est d'implémenter le nouvel algorithme BCA\*, mettre en place les nouveaux critères et modifier l'interface pour adapter l'algorithme et les critères. Ce rapport comprend plusieurs phases et tâches du projet : le cahier des charges, l'installation et la configuration, la recherche des méthodes et des algorithmes, les réalisations.

**Mots clefs :** Géovélo, itinéraire, critère, BCA\*, C++, LEDA, HTML, CSS, PHP, AJAX, XAJAX API Google maps, PostgreSQL, PostGIS, XMLRPC

**Abstract :** The final project takes place during 2009-2010. The project objective is to implement the new algorithm BCA\*, establish new criteria and modify the interface to adapt the algorithm and criteria. This report includes several phases and tasks of the project : the specification, installation and configuration, research methods and algorithms, achievements.

**Keywords :** Géovélo, path, criterion, BCA\*, C++, LEDA, HTML, CSS, PHP, AJAX, XAJAX, API Google Maps, PostgreSQL, PostGIS, XMLRPC

---

## Encadrants

Gaël Sauvanet

[gael.sauvanet@univ-tours.fr](mailto:gael.sauvanet@univ-tours.fr)

Emmanuel Néron

[emmanuel.neron@univ-tours.fr](mailto:emmanuel.neron@univ-tours.fr)

## Etudiant

Ke SHANG

[ke.shang@etu.univ-tours.fr](mailto:ke.shang@etu.univ-tours.fr)

DI5 2009 - 2010

Université François-Rabelais, Tours