2016

UFR Sciences Angers
Master 1 Informatique

université angers

Placement de routes

Urbanisme

David Florian

Sous la direction de M. Da Mota Benoit et M. Goëffon Adrien

Membres du jury

Soutenu publiquement le :
23 juin 2016

université angers

L'auteur du présent document vous autorise à le partager, reproduire, distribuer et communiquer selon les conditions suivantes :

- Vous devez le citer en l'attribuant de la manière indiquée par l'auteur (mais pas d'une manière qui suggérerait qu'il approuve votre utilisation de l'œuvre).

- Vous n'avez pas le droit d'utiliser ce document à des fins commerciales.

- Vous n'avez pas le droit de le modifier, de le transformer ou de l'adapter.

Consulter la licence creative commons complète en français :
http://creativecommons.org/licences/by-nc-nd/2.0/fr/

Ces conditions d'utilisation (attribution, pas d'utilisation commerciale, pas de modification) sont symbolisées par les icônes positionnées en pied de page.

REMERCIEMENTS

Je tiens à remercier mes enseignants référents, Adrien Goëffon et Benoit Da Mota, pour leurs conseils et leur accompagnement.

Il me faut également remercier mes camarades de promotion, plus particulièrement ceux présents à l'Université au cours de ces deux mois et demi de travail. Parmis eux, les présences fréquentes de Jérôme Fourmond, Morgane Troysi et Ugo Rayer ont été particulièrement appréciables et appréciées.

Table des matières

Introduction

1 Présentation du contexte

Dans le cadre du Master Informatique, il nous est demandé de réaliser un projet de recherche sur 10 semaines, afin de valider nos acquis, d'approfondir certaines connaissances et compétence et d'avoir une première approche d'un travail sur un projet de recherche, réalisé en quasi autonomie, puisque encadré par nos enseignants-chercheurs.

2 Présentation du projet

Ce projet a été proposé par deux enseignants : Benoit Da Mota et Adrien Goëffon. Il utilise principalement des principes étudiés dans le cadres de l'Optimisation Combinatoire, et de l'option Résolution de Problèmes. La combinaison de ces deux domaines, et compte tenu de la complexité du sujet, nécessite un bon niveau en programmation, dans un langage orienté vers la performance d'exécution. De plus, la visualisation d'une solution permettant de mieux rendre compte des résultat, la maîtrise de bibliothèques graphiques pouvait être un avantage.

3 Introduction au sujet

Le but est de pouvoir fournir à un utilisateur des dispositions possibles de routes sur une surface. Elles doivent respecter plusieurs contraintes. La première stipule que toutes les routes portions de routes doivent être connectées entre elles, par l'intermédiaire d'autres routes, et aux entrées-sorties. La deuxième, que, dans un premier temps, la surface étudiée et les cellules la composant seront de forme rectangulaire ; D'autres formes de cellules pourront être envisagés par la suite. La troisième définit ce qu'est une parcelle exploitable : est exploitable, toute parcelle ayant au moins une route dans son voisinage, à une distance maximale définie dans les paramètres du problème.

A partir de ces règles, on souhaite offrir à l'utilisateur des solutions maximisant deux objectifs : le premier est un objectif de densité, c'est le nombre de parcelles exploitables de la surface. Le second est l'accessibilité, ou circulabilité, globale, entre les parcelles exploitables. On souhaite que les déplacement entre les parcelles, en passant par les routes, soient les plus courts possible, de n'importe quelle position, vers n'importe quelle autre. Pour ceci, on considère l'écart proportionnel entre la distance « parfaite », distance sans obstacles entre deux parcelles, et la distance par les routes. Cela nous donne un ratio associé aux deux cellules concernées.

La distance « parfaite », que l'on appellera par la suite distance directe, est le nombre de cellules d'écart entre deux parcelles. Pour le moment, il s'agit de la distance Manhattan qui est utilisée. La distance Manhattan est calculée en comptant le nombre de cellules se trouvant sur deux lignes droites maximum, reliant deux parcelles. On pourrait choisir d'utiliser la distance euclidienne, ou distance « à vol

d'oiseau », si on le souhaitait. Il suffirait d'utiliser le calcule de distance euclidienne à la place du Manhattan. La distance euclidienne peut alors être une valeur décimale, au lieu d'entiers avec la distance Manhattan.

Si le premier objectif est plutôt simple à évaluer et maximiser, le second est bien plus complexe dans son approche et sa conception et son optimisation. Dans l'exemple ci-dessous, le ratio associé aux parcelles A et B est de 26 pour 1. Or, la moyenne des ratios pour toutes la surface est de 1.40137, ce qui est une valeur relativement correcte. En effet, du fait du placement des routes sur cette surface, deux cellules qui ne sont pas du même côté de la surface, auront toujours un ratio très proche de 1, et ces cas représentent environ la moitié des ratios calculés. On voit donc donc l'écart entre le ratio maximum, qui est élevé, et la moyenne des ratios, qui est tout à fait correcte.

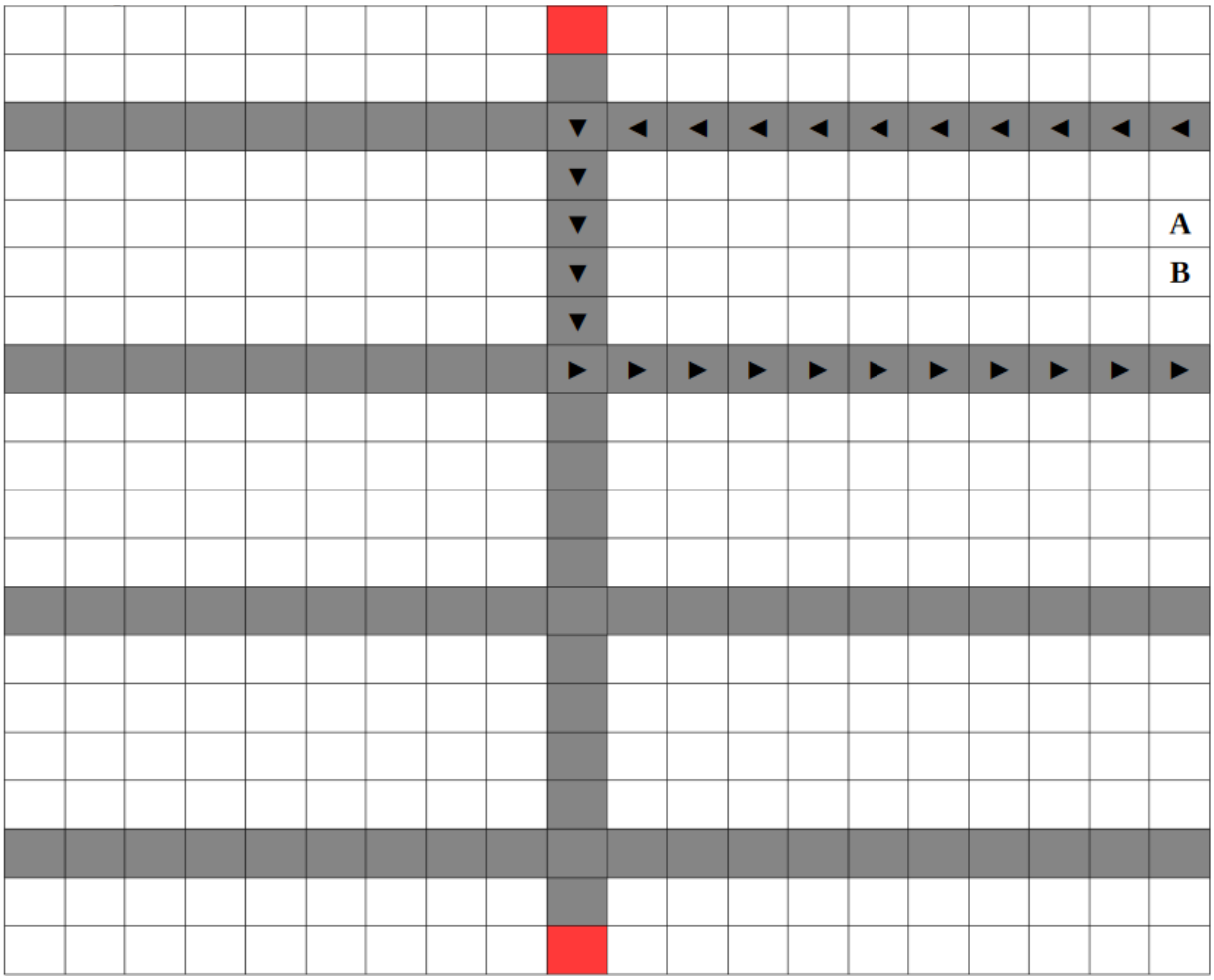

Illustration 1: Surface d'exemple.

4 Contextualisation

Le sujet peut être appliqué sur différents domaines. Une des applications est l'organisation et l'accès à des parcelles d'une exploitation agricole, selon le placement des routes et chemins desservant

les parcelles. Des ajouts pour se rapprocher de ce modèle pourrait être l'ajout de la gestion des entrées et sorties entres les parcelles et les accès ou des tailles de parcelles différentes, au lieu d'une unité fixe dans le cas présent (une route a la même dimension qu'une parcelle).

Une autre application serait la planification de l'urbanisation d'une ville, afin d'optimiser les emplacements pour habitations et bâtiments, tout en conservant une bonne circulation générale sur l'ensemble du réseau routier. Pour améliorer ce modèle, on pourrait ajouter des priorités sur certains bâtiments. Ou encore, travailler sur une ville déjà existante, que l'on pourrait modifier, mais avec comme contrainte de ne pas utiliser certaines zones, car contenant des infrastructures. On ne pourrait alors plus créer et planifier l'organisation en partant « de zéro », mais on serait contraint par l'environnement pré-existant.

On peut également imaginer appliquer cela au choix de lignes de bus. On ne pourrait alors placer des « routes de bus », seulement sur des routes au sens large, déjà présentes. Il faudrait alors définir des cellules pouvant devenir des « routes », dans la représentation existante du problème, et celles qui sont non modifiables, les habitations, parcs, magasins, etc.

D'autres cas d'utilisation peuvent être réfléchis, il suffit de modifier l'application afin d'ajouter les nouvelles contraintes du problèmes que l'on veut étudier.

5 Etudes préliminaires

1. Réflexions sur les applications et enjeux du problème
2. Recherche de travaux similaires
3. Réflexion sur les langages, outils et approches du problème

6 Choix des outils

Développement orienté multiplate-formes, j'ai donc choisi des outils fortement universels :

Langage C++, avec *Kdevelopp*, *Cmake* et *Make*, puis *Qt* pour l'interface graphique, avec *Qt Creator*, davantage adapté à ce framework. Des tests unitaires ont été réalisés avec la bibliothèque *CppUnit*. *Gnuplot* a été utilisé afin de créer des images de fronts Pareto do solutions. Son utilisation a été facilitée grâce à l'interface C++ *gnuplot-cpp*, développé par Jeremy Conlin (jeremit0@gmail.com), avec quelques corrections, pour éviter les « warning » de compilation et corriger la documentation, pour qu'elle soit reconnue valide par *Doxygen*.

Les outils de débogage utilisés sont *Valgrind* et ceux intégrés à *Kdevelopp* et *QtCreator*. *Valgrind* a également été utilisé afin de supprimer les fuites mémoires. De plus, couplé à *Callgrind*, il permit de générer des fichiers lisibles par *Kcachegrind*, pour l'analyse des performances de l'application. J'ai également utilisé l'outil *CppCheck* afin d'effectuer certaines micro-optimisations.

Le versionnage est géré à l'aide d'un dépôt Git, hébergé sur github.com. La documentation est créée grâce à *Doxygen*.

Mes plate-formes de développement et de test ont été Linux Mint et Windows 10.

Démarche

1 Organisation

J'ai décidé de séparer le programme et ses fichiers sources en plusieurs parties. Ceci a pour but de pouvoir exécuter le programme sans utiliser obligatoirement une interface graphique, d'améliorer sa portabilité avec d'autres outils ou plate forme de destination. Cela améliorant également la maintenabilité et la poursuite éventuelle du programme par d'autres personnes.

Ainsi, le code du programme est divisé en trois parties, chacune représentant un niveau de fonctionnalités.

- Le cœur de l'application est appelé le moteur (« Engine »), il est chargé de représenter et de manipuler les données représentant les solutions ainsi que les paramètres d'une instance du problème. De plus, une classe « Coordinates » permet de pouvoir stocker et comparer des coordonnées lors des manipulations d'une surface.
- La partie algorithmique du projet et placée à la racine de celui-ci, car c'est elle qui est chargée de résoudre les instances du problèmes, de chercher des solutions, de les stocker et de évaluer pour les comparer. Nous avons donc une classe représentant une surface et son évaluation, « FieldEvaluation ». L'évaluation de la surface doit être à tout moment à jour, à partir du moment ou la solution et réalisable. Il y a ensuite une classe permettant de trouver des voisins d'une solutions, « LocalSearch » ; Il s'agit donc d'une recherche locale. Il serait envisageable de créer et d'utiliser d'autres méthodes de recherche de solutions telles que des algorithmes génétiques ou de la programmation par contrainte. Enfin, puisque le problème est multi-objectifs, il nous faut stocker et mettre à jour les solutions ayant une évaluation non dominée. Ceci est pris en charge pas la classe « Resolution » qui est chargée de gérer quelle est la solutions sur laquelle on travaille et de gérer les solutions trouvées précédemment, ainsi que d'exporter les résultats sous la forme d'un front Pareto.
- La dernière partie est la partie interface graphique. Elle comprend l'affichage et les interactions pouvant être effectuées par l'utilisateur de l'application.

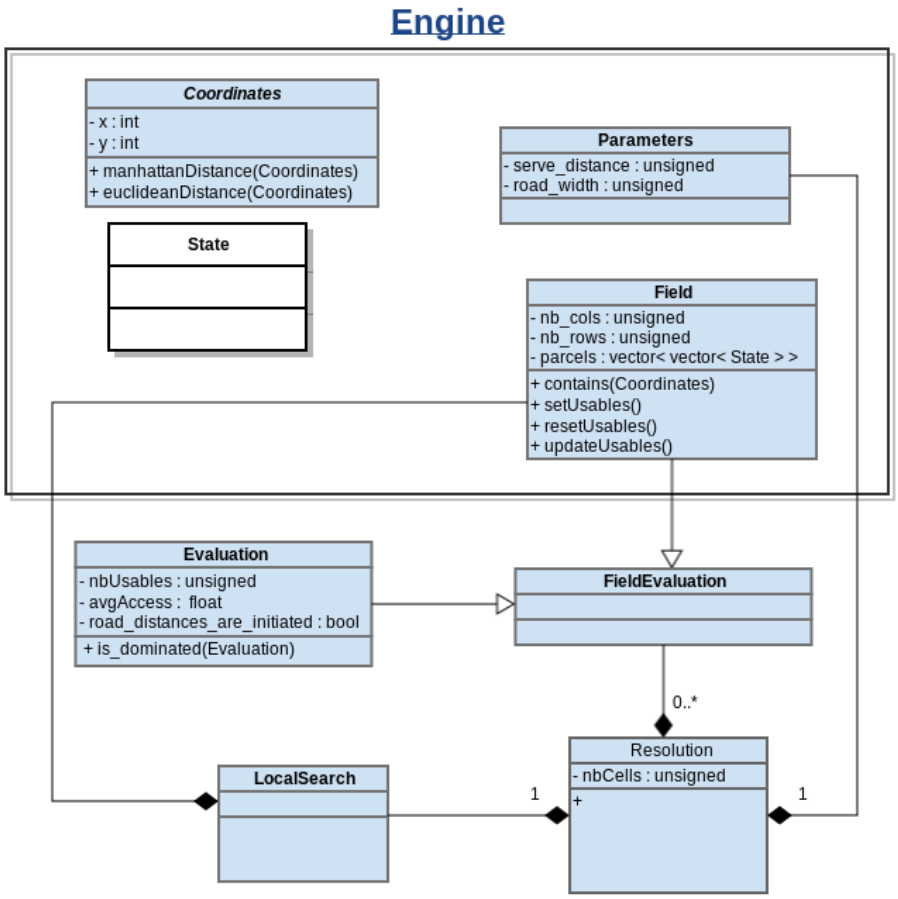

Illustration 2: Diagramme de classes du projet

2 Représentation des données du problème

La surface du problème étant un rectangle de cases rectangulaires, la structure de donnée la plus évidente est de la stocker sous forme de matrice à deux dimensions. C'est le choix qui a été fait, afin de faciliter la représentation du problème, et donc sa compréhension et la réflexion sur ses spécificités et enjeux. Chaque cellule est donc représentée par un état dans la matrice.

Cependant, s'est posée la question de l'intérêt de l'utilisation d'une représentation à base de graphes, plutôt que matricielle. Cette solution paraît davantage indiquée dans le cas de l'utilisation d'un algorithme génétique, du fait du croisement possible de « branches » du graphe. De plus, cela aurait permis de plus facilement changer la forme des cellules, avec une forme hexagonale par exemple. Malgré cela, cette représentation pouvait apporter une certaines confusion dans ce problème déjà complexe et vaste ; c'est pourquoi cette possibilité n'a pas été retenue.

3 Evaluation des solutions

3.1. Nombre de parcelles exploitables

Il s'agit de vérifier qu'elles sont les parcelles accessibles, et donc exploitable. On applique cette règle : *Si une cellule, qui n'est pas une route, a une route à une distance inférieure ou égale à D cellules (D le paramètre de desserte, en distance Manhattan), elle est alors exploitable.* On compte ensuite le nombre de parcelles exploitables total.

3.2. Calcul du ratio entre distance directe et distance par les routes

Le ratio moyen permettant d'évaluer l'accessibilité est plus difficile à calculer. En effet, il s'agit de trouver, pour chaque parcelle exploitable, on cherche la distance par les routes pour aller à chaque autre parcelle exploitable, pour pouvoir diviser leur distance directe – Manhattan ou euclidienne – le ratio entre ces deux parcelles. Il faut ensuite faire la moyenne des tous les ratios obtenus ainsi. Il y a donc N x (N-1) ratios à calculer, N étant le nombre de parcelles exploitables, pour obtenir le ratio total.

Or, pour calculer ce ratio, il nous faut, pour chaque parcelle, la distance vers chaque autre parcelle. Pour les obtenir, il nous faut donc trouver les chemins les plus courts pour aller d'une parcelle aux autres. Il s'agit donc d'un « pathfinding », opération coûteuse, surtout lorsque l'on travaille avec les valeurs exactes et minimales, comme c'est le cas ici.

De plus, on était amené à évaluer très fréquemment de nouvelles solutions, avec de légers changements. Une évaluation aussi peut performante aurait donc compromis la bonne progression du développement. La solution a été d'utiliser le concept de programmation dynamique, vu cette année. En effet, lors du calcul des plus courts chemin, il arrive fréquemment que l'on calcul plusieurs fois le chemin minimal entre deux mêmes routes. Cela est dû fait que si deux routes sont coller, il y a de forte chances que le chemin le plus court vers une troisième route soit le même, à une route près.

Cette amélioration a permis de grandement améliorer les performances de l'application. L'évaluation reste malgré tout une part importante des ressources utilisées par l'application. La mise en place d'une potentielle évaluation incrémentale, pour calculer l'accessibilité d'une solution, dérivée voisine d'une autre dont on connaît l'accessibilité, s'est avéré bien plus restreinte et difficile a mettre en place qu'escompté. Les tentatives sur ce sujet n'ont pas abouti. Cela vient du fait que l'ajout d'un chemin entre deux routes peut modifier la distance entre deux autres routes, qu'elles soient proches ou non du nouveau chemin.

4 Recherche de solutions améliorant un objectif

4.1. Initialisation d'une solution

Si la solution initiale présente sur l'affichage de l'application n'est pas réalisable, on peut décider de relier les entrées-sorties (E/S) entre-elles, ce qui va la rendre réalisable. Le placement des routes

permettant de relier deux E/S est effectué par un algorithme. Il vérifie d'abord si les deux cellules sont en face l'une de l'autre. Dans ce cas, une bonne méthode de placement des routes est de créer

4.2. Maximisation des objectifs par recherche locale

Le choix du type d'algorithme lors de la recherche de solutions améliorantes s'est porté sur la recherche locale. Ce choix est justifié par le fait que la structure d'une surface simplifiait ce procédé. De plus, il ne restreindrait pas l'application future des évolutions évoquées.

4.2.1. Maximisation du nombre de parcelles exploitables

La première étape consiste à maximiser le nombre de parcelles exploitables. Pour ceci, un algorithme glouton a été utilisé. Pour chaque route ajoutée, on cherche celle qui augmentera le plus le nombre de routes exploitables en la plaçant. En revanche, les routes qui auraient plusieurs autres routes voisines seront défavorisées. Il n'y a donc pas de stratégie de placement sur plusieurs routes à placer. Ainsi, l'ajout d'une route peut diminuer l'intérêt d'une route précédemment placée. Ce placement n'est donc pas optimal, mais puisqu'on est sur un problème multi-objectifs, il ne serait pas judicieux de ce concentrer sur un seul des ces objectifs.

4.2.2. Maximisation de la circulabilité

La seconde étape va donc chercher à maximiser le second objectif : l'accessibilité. On va chercher à ajouter des « chemins » de routes entre deux routes existantes, afin de créer des passages améliorant globalement la circulation, à l'aide d'un changement local.

...

On peut alterner les étapes en fournissant un nombre de routes à ajouter pour le nombre d'exploitables – sachant que la valeur 0 permet d'ajouter des routes tant qu'on améliore l'objectif – et le nombre de chemins pour l'accessibilité.

5 Persistance des solutions non dominées

Lors de l'évaluation d'importantes quantités de données sont générées lors de la recherche de chemin. Les valeurs de distance entre toutes les routes et toutes les autres routes sont stockées dans une matrices à quatre dimensions (2 pour les coordonnées de départ et deux pour les coordonnées d'arrivé). Puisqu'on utilise la programmation dynamique, il faut stocker toutes les valeurs calculées afin de les réutiliser pour calculer les suivantes. On pensait également les utiliser lors d'une mise à jour incrémentale des données lors de l'ajout de routes, pour ne pas les recalculer entièrement lors de la nouvelle évaluation, mais il s'est avéré qu'elles ne pouvaient pas être utilisées. On peut alors se poser la question de l'intérêt des les conserver lors du stockage d'une solution en vue de la réutilisation future.

Une solution serait de stocker tout le chemin en plus de la valeur de distance entre chaque case, afin de mettre à jour seulement les chemins qui peuvent être affectés par l'ajout de routes. Mais cela

entraîne une quantité bien plus importante de données, et la faisabilité et la pertinence de cette solution n'ayant pas été démontré, cette possibilité n'a pas été mise en place.

6 Affichage et Interactions

6.1. Affichage

L'affichage est donc géré avec le framework Qt. Une surface est affichable à l'aide d'un widget, représenté par une classe. Cette classe nécessite la surface à afficher et son évaluation, afin de pouvoir dessiner une « Hotmap ». Les E/S sont rouges, les routes grises, les parcelles non exploitables bleu clair et les exploitables blanches. Cet affichage doit être mis à jour lors d'une interaction par l'utilisateur et à la suite de l'exécution d'un algorithme de résolution. La solution affichée par l'application est la dernière solution trouvée.

Un affichage alternatif est possible. En effet, une fois qu'une solution est réalisable, l'utilisateur peut a la possibilité d'afficher une « Hotmap » de celle-ci, afin de le guider dans ses prises de décisions. Ce terme désigne l'affichage de couleurs, allant de vert à rouge, selon l'écart avec la moyenne de l'accessibilité de la parcelle. Cet affichage permet de repérer les zones qui sont les moins accessibles, et donc d'essayer de placer des routes afin d'améliorer leur accessibilité, dans le but d'améliorer la circulabilité. Plus une région contient de parcelles d'une couleur proche du rouge, moins son accessibilité est bonne. On peut donc concentrer nos efforts à l'améliorer manuellement.

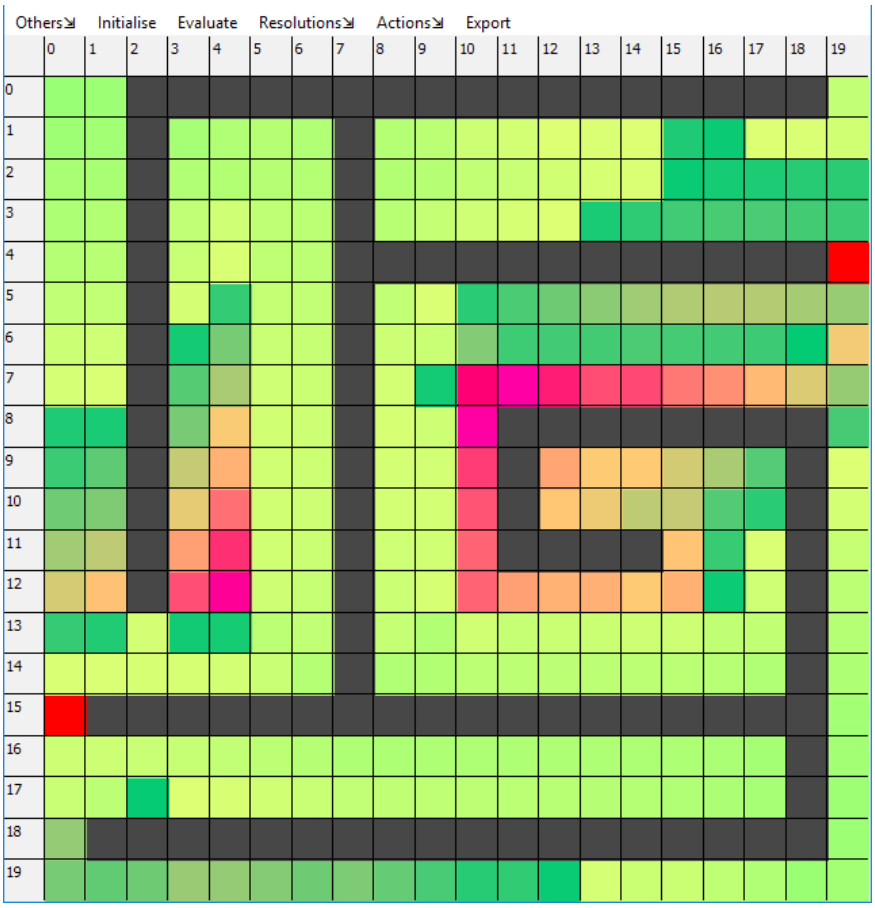

Illustration 3: Exemple de « Hotmap »

6.2. Interactions

Le widget d'affichage d'une surface permet non seulement de l'afficher, mais également, grâce aux fonctionnalités du framework Qt, de recevoir et de traiter des entrées de la part de l'utilisateur. L'application étant conçue comme une aide à la décision, elle gère différentes interactions avec celui-ci.

Au lancement du logiciel, l'utilisateur peut ajouter ou supprimer des E/S et des routes, en cliquant sur les zones concernées, avec le pointeur de la souris. Un clic gauche va alors placer une route, tandis qu'un clic droit ajoutera une E/S. Il peut choisir de créer une solution réalisable manuellement, puis de l'évaluer. Sinon, il peut choisir de placer uniquement des E/S et demander à l'application de les relier automatiquement, afin de créer une solution réalisable automatiquement. Il a ensuite la possibilité de lancer une résolution sur une solution initiale complétée automatiquement ou manuellement. Pour chaque action du bouton « Initialise », une E/S et la suivante sont reliées. Ainsi, si le nombre N d'E/S est supérieur à deux, on est sur de toutes les relier avec N clics sur le bouton « Initialise ».

Chaque Comme indiqué plus haut, on peut choisir de d'abord maximiser le nombre d'exploitables, avant d'essayer de « corriger » la solution trouvée. Une autre possibilité est de procéder par petites itérations, en indiquant un nombre de routes à ajouter à chaque étape de maximisation du

nombre d'exploitables et un nombre de chemins à chaque étape de maximisation de la circulabilité. Pour exécuter une résolution, on clique sur « Resolution » dans la barre des menus, puis on choisit « Usable » ou « Access », pour augmenter respectivement le nombre d'exploitables ou la circulabilité. On pourrait ajouter une option, qui permettrait de maximiser les deux objectifs en une seule étape. L'utilisateur devrait alors indiquer une valuation pour chaque objectif, ce qui transformerait temporairement le problème en un problème mono-objectif. En effet, fonction objectif, donnant l'évaluation d'une solution, serait alors de la forme :

$$\max f(x) = \alpha \times \frac{nbexploitables}{nbtotal} + \beta \times \frac{1}{accessibilite}$$

7 Création du front Pareto

La dernière étape pour l'utilisateur est de choisir une des solutions qui lui convient, afin de pouvoir la mettre en place s'il le désire. Pour effectuer cela, il peut voir les évaluations de toutes les solutions trouvées non dominées, qui sont stockées en mémoire, placées sur un front Pareto. Ainsi, il n'a qu'a choisir le compromis entre les deux objectifs qui lui convient le mieux.

8 Fonctionnalités possibles, mais non implémentées

Largeur routes ?? La largeur des routes n'est prise en compte dans aucun calcul, cela entraîne une lenteur importante lors du pathfinding entre parcelles, par les routes, si on utilise une largeur supérieure à 1.

Déploiement

J'ai décidé de donner un caractère professionnel à l'application. Se trouvait donc sur le site Github de celle-ci, un « readme » au format markdown, décrivant son but et détaillant les fonctionnalités à venir, celles envisagées, et celles déjà réalisées, mise à jour selon l'avancement du développement. Les menus de l'application sont en anglais. Le code est en anglais, en dehors des commentaires qui sont intégralement en français.

Une balise indiquant si la dernière version du projet compile correctement est également présente, grâce à l'intégration du service Travis. Ensuite, un léger site web, dédié à l'application a été mis au point, à partir du contenu du readme. Une documentation a été crée, à l'aide de *Doxygen*, et ajoutée au site web. Toutes les classes, fonctions et attributs public sont donc documentés, en plus d'une description succincte du but du programme. Afin de protéger mon travail, une licence GPLv3, dernière version en date de la licence GPL, a été appliquée au projet, et l'ajout d'une balise dans le readme permet d'indiquer clairement cette licence.

Une fois le développement bien avancé, des outils de profilage, de vérification de code et de fuites mémoires ont été utilisé. Ainsi, *valgrind* a permis de « chasser » les fuites mémoires, même si

certaines persistent, principalement à cause de l'utilisation de Qt et de ses éléments graphiques. *Callgrind* est un outil intégré à *valgrind*, a permis de générer des fichiers lisibles par *Kcachegrind*, un utilitaire graphique permettant de visualiser quelles sont les fonctions les plus appelées et les plus coûteuses. J'ai tenté d'utiliser *gprof*, mais, n'étant pas un outil graphique, il était plus difficile de repérer les « points chauds » du programme qu'en utilisant *Kcachegrind*, il lui a donc été préféré. L'outil *Cppcheck*, utilisé sous Windows, a lui permis d'effectuer quelques micro-optimisations du code, principalement des portées de variables qui pouvaient être réduites.

Tout au long du développement, les tests ont été exécutés, afin de vérifier que les fonctionnalités du moteur qui étaient correctes le soient toujours. De plus, dans l'éventualité de reprise du programme, afin de continuer son développement ou sa maintenance, par une autre personne ou non, cela permet de s'assurer que les modifications apportées n'empêchent pas le fonctionnement des anciennes fonctions.

Conclusion

Le sujet demandait des compétences dans divers domaines. Ceux ci sont : l'optimisation combinatoire, la résolution de problèmes, la bonne utilisation – organisation et optimisation – du développement dans un langage adapté et performant, et la création d'une interface graphique complète.

Le développement de l'application a fait appel à plusieurs notions, types d'algorithmes et approches, étudiées durant l'année du Master 1 :

- La programmation dynamique
- La recherche de voisins grâce à une recherche locale
- Une utilisation importante de structures de données complexes : matrices à 4 dimensions, création et utilisation de graphes, listes d'éléments contenant d'importantes données calculées
- Gestion du partage des données, contenues dans les structures, entres les différents niveaux de l'application : affichage, interactions, résolution, export et sauvegarde …

Il à été nécessaire de mettre en place et respecter certaines méthodologies et de faire des choix : déterminer l'intérêt des algorithmes génétiques dans ce cas précis, gérer le multi-objectifs et la création d'un front Pareto à partir d'une évaluation des solutions, l'utilisation d'une ancienne solution, sans devoir recalculer ses données, gestion de la mémoire, mettre à jour les données lorsque c'est pertinent et seulement lorsque c'est pertinent pour éviter d'effectuer des longs calculs s'ils ne sont pas utiles dans certains cas.

Il aurait été vain de tenter de traiter l'ensemble de l'étendue du problème. Il a donc fallu déterminer les éléments primordiaux à mettre en place et sélectionner ceux à traiter lors de la résolution. Il est donc possible de continuer le projet afin de comparer différents algorithmes et implémentations, ajouter des interactions et informations fournies à l'utilisateur.

La mise en place de tests et devoir maintenir un code clair et commenté était indispensable pour le bon déroulement du projet sur une période de temps plus étendue qu'à l'accoutumée.

Sitographie

Table des illustrations

Tableau 1: Inscrire une légende

mots-clés : Entrée-sortie ; Solution réalisable

Présidence de l'université
40 rue de rennes – BP 73532
49035 Angers cedex
Tél. 02 41 96 23 23 | Fax 02 41 96 23 00

université angers

ENGAGEMENT DE NON PLAGIAT

Je, soussigné(e) .....
déclare être pleinement conscient(e) que le plagiat de documents ou d'une partie d'un document publiée sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.
En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour écrire ce rapport ou mémoire.

signé par l'étudiant(e) le

Cet engagement de non plagiat doit être signé et joint à tous les rapports, dossiers, mémoires.

Présidence de l'université
40 rue de rennes – BP 73532
49035 Angers cedex
Tél. 02 41 96 23 23 | Fax 02 41 96 23 00

université angers