Vers de nouvelles dynamiques de localisation des ménages et des activités dans les territoires urbains.

Félix Voituret
Master 2 Ingénérie logicielle

24 juin 2013

Table des matières

1 Introduction **3**

2 Problématique et enjeux **4**
2.1 Problème de localisation d'activité 5
2.2 Problème de dimensionnement de réseau de transport 5
2.3 Problème combiné 5

3 MATSim **6**
3.1 Demande initiale 6
3.2 Simulation 7
3.3 Scoring 7
3.4 Replanning 9

4 Notre approche **10**
4.1 Modélisation de notre problème 10
4.2 Exploitation avec MATSim 10
4.3 Clustering 11
4.4 Agrégation 12
4.5 Optimisation 12
4.5.1 T-linearisation 13
4.5.2 Résolution par Heuristique 16
4.6 Implémentation 19
4.7 Expérimentation et Résultats 19
4.8 Test sur la ville de Lyon 20

5 Un outil d'aide à la décision pour la construction des territoires urbains **22**
5.1 MATSim2GS 22
5.2 Avancement 23

6 Conclusion et perspective **24**

1 Introduction

Au regard de mon parcours en licence et de mon projet professionnel, l'idée de l'alternance est venue très rapidement dans le but de découvrir le monde de la recherche et de commencer à travailler sur des projets en optimisation combinatoire, domaine dans lequel je souhaiterais m'orienter par la suite lors d'une thèse.

Suite à une mission de deux mois réalisée durant l'été 2011 dans le cadre de travaux introductifs, j'ai démarré mon projet d'alternance au sein du *Laboratoire Informatique d'Avignon* plus communément appelé *LIA* [1], en septembre 2011. Rattaché à l'infrastructure dédiée à l'enseignement informatique à l'*Université d'Avignon et des Pays de Vaucluse*, ce laboratoire emploi chercheurs, mais également doctorant ainsi que stagiaire / alternant Master, et de nombreux projets y sont réalisés que ce soit pour le domaine public ou le domaine privé.

Les axes de recherches sont orientés autour de plusieurs domaines à savoir :

- Réseau et télécommunication
- Recherche opérationnelle
- Traitement du langage écrit
- Traitement du langage oral
- Indexation et classification de document

Mon projet s'inscrit dans les travaux réalisés au sein de l'équipe de recherche opérationnelle et optimisation combinatoire. Il est supervisé par *Serigne Gueye* [2] et *Philippe Michelon* [3].

Ce document décrit l'ensemble des travaux réalisés durant cette mission. Après une présentation des enjeux et du contexte économique du projet, une introduction du système de simulation multi-agent *MATSim* sera faite, cet outil étant au cœur des travaux effectués. Ensuite, une explication détaillée de notre approche ainsi que de la solution développée sera fournie, avant de finir sur les résultats obtenus ainsi que les perspectives du projet.

1. http ://lia.univ-avignon.fr
2. Maître de conférence, serigne.gueye@univ-avignon.fr
3. Professeur, philippe.michelon@univ-avignon.fr

2 Problématique et enjeux

Le projet DAMA [4], est un projet pluridisciplinaire financé par l'organisme PREDIT ANR[1], à hauteur d'environ 100000 euros sur trois ans, et faisant intervenir les entités suivantes :

- Laboratoire Informatique d'Avignon.
- UMR Espace d'Avignon.
- Laboratoire d'économie des transports de Lyon.

Son objectif est l'étude des différentes interactions urbaines, ainsi que leurs impacts vis-à-vis de la dépendance automobile dans les réseaux de transports. En effet, plusieurs études ont démontrés la nécessité de considérer la topologie des territoires urbains autrement, afin de favoriser les activités [5] de proximités. L'une d'entre elle, basée sur la théorie de la métrique lente, met en évidence que les structures actuelles impliquent que plus les déplacements sont grands, plus l'usager gagne du temps et de l'argent, ce qui entraîne naturellement le fait que les activités de proximités soient défavorisées.

Ainsi depuis des décennies, les villes se sont étalées au détriment de la mobilité, la conception du territoire étant établi sur la conviction que peu importe où une infrastructure sera localisée, elle restera accessible du moment qu'elle est desservie par un voie routière. Cela induit une ubiquité de l'automobile dans les modes de fonctionnement urbain, car en effet il n'existe aucune alternative viable à l'heure actuelle. Or les différents facteurs économiques, comme le prix de l'essence, ou encore les considérations environnementales impliquant une mobilité toujours plus réduite remettent en cause ces fonctionnements, au même titre que la raréfaction des financements publiques pour le déploiement de nouvelles infrastructures.

Les questions suivantes ce posent alors :

- Vaut - il mieux, au regard de l'accessibilité produite, créer de l'infrastructure ou relocaliser les activités ?
- Quelles serait la localisation optimale des activités actuelles à réseau constant ou en projetant des évolutions du réseau routier ?

La dernière question est en particulier spécifique à la ville de *Lyon*. Une des finalités étant de déterminer si à horizon 2030, la construction d'une nouvelle rocade serait pertinente ou non.

Le *Laboratoire Informatique d'Avignon* s'intéressent alors à deux problèmes d'optimisation associés à cette thématique : le problème de localisation d'activité et le problème de dimensionnement de réseau, leurs descriptions font l'objet des chapitres suivants.

4. Découpler Accessibilité et Mobilité Automobile dans les territoires urbains
5. Par activité sera sous-entendu une entité géographique qui crée ou reçoit des flux de déplacement

2.1 Problème de localisation d'activité

Considérant un ensemble d'activité générant des flots dit *Origine-Destination*, ainsi qu'un ensemble de localisation géographique, l'objectif du problème de localisation consiste à affecter chaque activité à une localisation de sorte à minimiser le produit des flots et des distances. Ce problème possède un ensemble d'application illimité, disposition de bâtiment dans un complexe scolaire, de service dans un centre hospitalier, de composants électronique sur un circuit et bien d'autre. Par rapport à notre thématique, il s'agit de pouvoir localiser des activités au sein d'un réseau de transport, de sorte à les rendre les plus accessibles possibles.

2.2 Problème de dimensionnement de réseau de transport

Ici il s'agit de définir un réseau de transport ou ses évolutions, en considérant un ensemble de route potentielle, disposant d'un coût de construction et d'utilisation, en fonction des flux observés. Différentes approches sont possibles pour évaluer l'utilisation d'une route. En effet nous pouvons nous baser sur la simple logique qu'un usager utilisera le plus court chemin (selon une métrique définie telle que la distance ou le temps de trajet) pour aller d'un point à un autre. Mais il est également possible et plus réaliste d'utiliser des *modèles d'équilibre* qui prennent en compte de la saturation d'un arc du réseau pour déterminer la route qu'un usager empruntera pour effectuer son trajet.

2.3 Problème combiné

Pour finir et afin d'obtenir un modèle le plus complet possible, un problème combiné sera étudié. Il s'agira d'un problème de localisation d'activité ou le réseau de transport n'est pas encore conçu [6].

6. Ou du moins partiellement ...

3 MATSim

MATSim[7] est un framework open-source développé en Java permettant de réaliser de la simulation multi-agent sur des réseaux de transport. L'objectif d'un tel système est de pouvoir mesurer différents indicateurs relatif aux systèmes de transports terrestre urbain, à l'aide d'une approximation du comportement des individus d'une population, ou *agent*. Cet outil, très utilisé au sein de grandes cités telles que Berlin, Singapour, ou encore Toronto, a l'avantage d'être très performant même sur des quantités de données énormes[8]. Le processus itératif employé est composé de trois étapes majeures, comme le décrit le workflow figure 1. L'objectif étant au fil des itérations, d'améliorer le fonctionnement du système en tentant de maximiser un score dit d'utilité pour l'ensemble des *agents*. Développé par une équipe de quarante chercheurs issus de plusieurs laboratoire d'Europe depuis plus de dix ans, *MATSim* demeurent un outil des plus performant de sa catégorie.

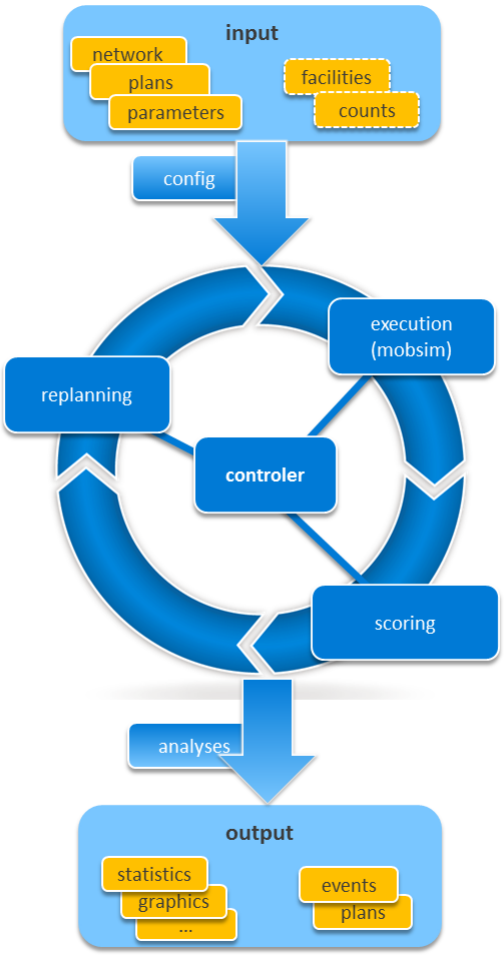

FIGURE 1 – MATSim workflow

Le détail des différentes étapes fait l'œuvre des paragraphes suivant.

3.1 Demande initiale

Les données d'entrée, ou *demande initiale*, sont constituées de différents fichiers XML, contenant les informations suivantes :

- **config.xml** : Ensemble des variables de configuration du système.
- **network.xml** : Contient le réseau de transport, via un ensemble de nœuds et de liens ainsi que des informations relatives aux limitations de vitesses.
- **plans.xml** : Contient les plans de chaque *agent*, un plan étant une succession d'activité et de déplacement.
- **facilities.xml** : Contient les informations sur les activités, à savoir leur horaire d'ouverture ainsi que leurs catégories parmi cinq types : Résidentielle, Emploi, Commerce, Éducation et Loisirs.

A l'issue du processus, les résultats sont restitués sous la même forme.

7. Multi Agent Transportation Simulation toolkit
8. A titre indicatif, *MATSim* peut théoriquement simulé la population entière de Suisse

3.2 Simulation

Ici, le comportement des *agents* est simulé suivant leurs plans, sur une période d'une journée complète. Différents mécanismes sont employés afin d'obtenir une image du système de transport la plus réaliste possible.

A noter que le système est multi-modal, c'est-à-dire qu'un agent peut tout aussi bien utilisé une automobile, qu'un transport en commun ou encore se rendre à destination en marchant. En revanche, seul les trajets automobiles dispose d'une représentation du trafic dynamique, les temps de trajet avec les transports publics ainsi que la marche sont calculées instantanément par évaluation de la distance à parcourir et d'une vitesse moyenne, indépendamment du réseau de transport.

3.3 Scoring

A l'issue du processus de simulation, un score va être calculé pour chaque *agent*. Ce score est calculé comme la somme des utilités des déplacements ainsi que des activités exécutées par notre *agent* le tout pondéré par divers coefficients. Cette utilité est monétisée, ainsi le coût d'un déplacement aura un impact sur le gain de satisfaction obtenue par l'activité cible.

De manière plus pratique, une fonction économique appelée *Charyparnagel*, est utilisé. Elle est calculée ainsi :

$$F = \sum_{i=1}^{n} U_{act}(\text{type}_i, \text{start}_i, \text{dur}_i) + \sum_{i=2}^{n} U_{trav}(\text{loc}_{i-1}, \text{loc}_i)$$

L'utilité (ou desutilité) d'un trajet étant obtenu par une simple pondération du temps de transport :

$$U_{trav}(t_{trav}) = \beta_{trav}.t_{trav}$$

Et l'utilité d'une activité modélisée par la somme de cinq utilités distinctes :

$$U_{act,i} = U_{dur,i} + U_{wait,i} + U_{late.ar,i} + U_{early.dp,i} + U_{short.dur,i}$$

Ces utilités, pour quatre d'entre elle, agissent comme des pénalités liées aux phénomènes suivants :

- Le fait d'arriver trop tôt et d'attendre.
- Le fait d'arriver en retard.
- Le fait de partir trop tôt.
- Le fait de ne pas passer assez de temps.

Ainsi nous obtenons les expressions suivantes :

$$U_{wait}(t_{wait}) = \beta_{wait}.t_{wait}$$

$$U_{late.ar}(t_{start}) = \begin{cases} \beta_{late.ar}(t_{start} - t_{latest.ar}) & \text{si } t_{start} > t_{latest.ar} \\ 0 & \text{sinon} \end{cases}$$

$$U_{early.dp}(t_{end}) = \begin{cases} \beta_{early.dp}(t_{earliest.dp} - t_{end}) & \text{si } t_{end} < t_{earliest.dp} \\ 0 & \text{sinon} \end{cases}$$

$$U_{short.dur}(t_{start}, t_{end}) = \begin{cases} \beta_{short.dur}(t_{short.dur} - (t_{end} - t_{start})) & \text{si } t_{end} < t_{short.dur} \\ 0 & \text{sinon} \end{cases}$$

Enfin, l'utilité associée au temps passé sur une activité U_{dur}, agi comme l'utilité concrète d'une activité et s'exprime comme une utilité logarithmique :

$$U_{dur}(t_{dur}) = \beta_{dur}.t^*.\ln(t_{dur}/t_0)$$

Avec :

- β_{dur} : Constante.
- t^* : Valeur pour laquelle l'utilité marginal vaut β_{dur} (donc $t^* = t_{dur}$
- t_0 : Valeur pour laquelle l'utilité est positive.

Cette fonction, soulève cependant un certain nombre de problème comme nous pourrons le voir lors de la présentation des résultats.

3.4 Replanning

Pour finir, afin de modifier le score dans le but de l'améliorer, un module est chargé de reconstruire les plans. Bien évidemment, l'intégralité des plans originaux sont conservées, mais différentes opérations sont effectuées afin de réduire les temps de transports, tel que des changements d'itinéraire, changements modaux, variations d'horaire selon un écart tolérable, etc.

Cela est résolu par un algorithme génétique, sur une variante d'un problème standard d'optimisation de ramassages-livraisons (Pickup and Delivery Problem with Time Windows). Le problème résolu est défini par les trois sous problèmes suivants :

- Activity pattern generation : Selection d'un sous ensemble d'activité et ordonnancement de celui ci.
- Location selection : Localisation des activités effectués.
- Time allocation : Selection des horaires d'execution.

Les derniers plans obtenus pour chaque agent étant conservé en mémoire afin d'éviter d'être bloqué dans un optimum local.

C'est trois étapes sont répétées sur un nombre fini d'itération, l'objectif étant d'atteindre un équilibre du système où l'ensemble des *agents* ont une satisfaction maximum [9]. Nous avons ainsi un outil complet et robuste, bénéficiant d'une décennie de conception et qui permet d'obtenir une visualisation des différents impacts (en l'évaluation des scores) sur un territoire urbain en cas de modification. Il s'agit maintenant d'interagir de manière efficace et automatisé avec ce framework afin de pouvoir analyser l'effet de certains stimuli sur le système territorial.

9. Il n'y a cependant aucune garantie d'atteindre un tel équilibre

4 Notre approche

Le but recherché par le *LIA* est de mettre au point des méthodes de localisation d'activité optimale et d'analyser leur impact sur des données concrètes issus du partenariat avec le *laboratoire d'économie des transports de Lyon.*

4.1 Modélisation de notre problème

Dans un premier temps, il convient d'avoir une modélisation de notre problème d'optimisation combinatoire. Le problème de dimensionnement de réseau n'étant pas encore intégré à notre problématique, le choix de la modélisation se résume à un simple *problème d'affectation quadratique*. Il s'agit d'un problème dont la première formulation fut défini en 1957 par *Koopman et Beckmann* comme un programme quadratique en variable booléenne :

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij}.d_{kl}.x_{ik}.x_{jl} \qquad \text{(QAP)} \\ \text{sujet à} \quad & \sum_{i=1}^{n} x_{ik} = 1 \ \forall k \in]n] \\ & \sum_{k=1}^{n} x_{ik} = 1 \ \forall i \in]n] \\ & x \in \{0,1\}^n \end{aligned}$$

Avec :

- f_{ij}, le flot entre l'activité i et l'activité j.
- d_{kl}, la distance entre la localisation i et la localisation j.
- x_{ik}, une variable booléenne valant 1 si l'activité i est affecté à la localisation k, 0 sinon.

Différentes méthodes existent pour résoudre un tel problème, de manière approchée (méthode dite heuristique) ou exacte via des méthodes énumératives comme la séparation et évaluation progressive. Cependant, l'énorme complexité du problème dû à la fonction objectif qui est quadratique, rendent l'utilisation de méthodes exactes très difficiles, en effet, pour une instance de taille n, il y existe un ensemble de $n^2!$ solution réalisable.

4.2 Exploitation avec MATSim

L'approche adoptée consiste à utiliser les données issues de la simulation afin de résoudre, le problème de localisation assimilé. Ce procédé est répété itérativement comme le montre la figure 2. Deux étapes de traitements sont mises en place entre la simulation et l'optimisation afin de permettre l'extraction, la réduction et le formatage des données. Nous disposerons alors une solution logicielle qui nous permet de résoudre des problèmes de localisation [10] d'activité

10. ou relocalisation

sur des données réelles, tenant compte d'un trafic dynamique issu de simulation.

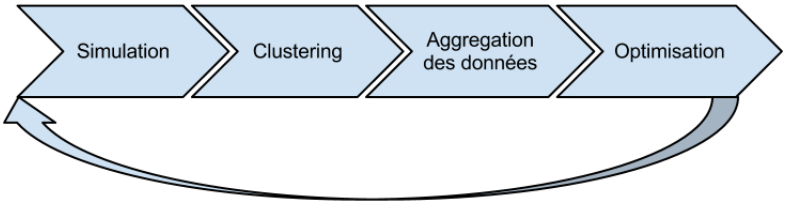

FIGURE 2 – Workflow

4.3 Clustering

Afin de réduire le volume de données considéré, une méthode de type clustering est alors employée. Le but étant de regrouper des activités proches de même catégorie comme une seule activité, un algorithme de classification hiérarchique ascendante a donc été mis en place.

Considérant un ensemble d'activité fini, des classes ou clusters d'activités sont définis itérativement à l'aide d'un seuil donné en paramètre.

Algorithm 1 Algorithme de classification hiérarchique ascendante

$C \leftarrow \emptyset$
$A \leftarrow \emptyset$
for $i = 1 \rightarrow n$ **do**
  if $i \notin A$ **then**
    $c \leftarrow \{i\}$
    $p \leftarrow (x_i, y_i)$
    for $j = i + 1 \rightarrow n$ **do**
      if $(j \notin A) \wedge (d(j, p) \leq \Delta)$ **then**
        $c \leftarrow c \cup \{j\}$
        $p \leftarrow (\bar{x_k}, \bar{y_k}) \mid k \in c$
        $A \leftarrow A \cup \{j\}$
      end if
    end for
    $A \leftarrow A \cup \{i\}$
    $C \leftarrow C \cup c$
  end if
end for

Cette approche de complexité en $O(n^2)$ donne des résultats de bonne qualité très rapidement, comme le témoigne la figure 3. Nous avons ainsi réduit un ensemble d'environ 7000 activités

résidentielles sur la ville de *Zurich* (identifié par les points rouges sur la carte) à un ensemble de taille inférieure à 1000 clusters (identifié par les points bleus sur la carte).

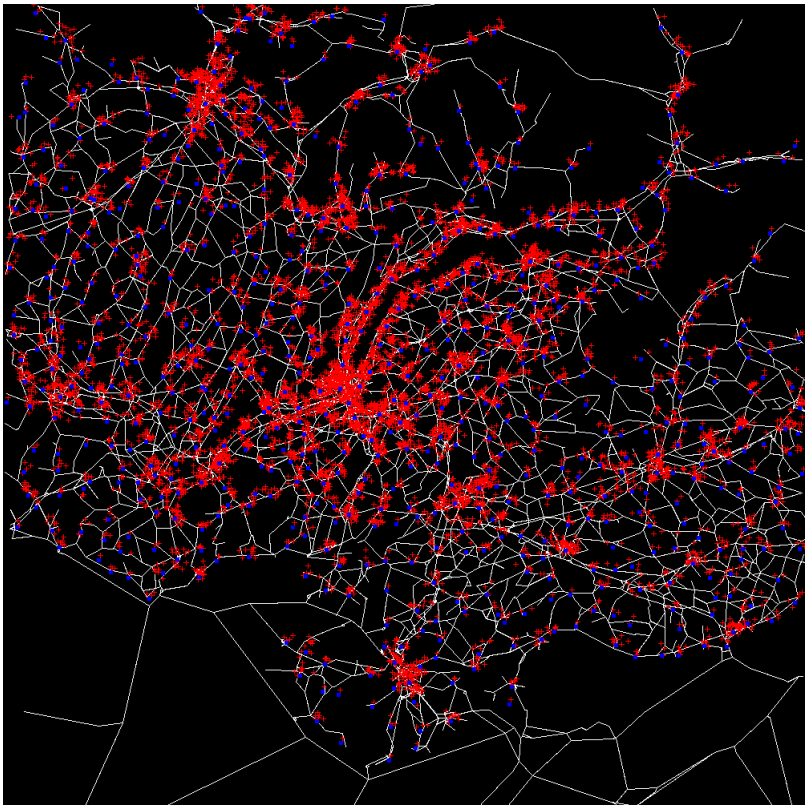

FIGURE 3 – Classification sur un ensemble de 7000 résidences avec un seuil de 2000

4.4 Agrégation

Une fois l'ensemble des activités réduit, l'extraction des données nécessaires à la résolution d'un problème de localisation d'activité peut avoir lieu, pour cela, les résultats de la simulation effectuée précédemment sont utilisés. Les flots sont obtenus par l'évaluation du trafic d'une activité à une autre et les distances entre ces activités sont modélisées par les temps de transports. L'estimation des flots origine-destination est réalisée par le calcul de la moyenne du trafic constaté par heure de la journée et les temps de transports sont aussi normalisés de la même façon afin d'avoir des données génériques. Les clusters d'activités obtenues précédemment sont bien évidemment pris en compte comme des activités uniques agrégeant les flots de chaque activité qu'ils contiennent.

4.5 Optimisation

Nous avons désormais les données nécessaires afin de traiter notre problème de localisation, il ne nous reste plus qu'à mettre en place une méthode de résolution pour notre *problème d'affectation quadratique* assimilé. Il s'agit d'un problème dit *NP-Complet* ce qui signifie qu'il n'existe à l'heure actuelle aucun algorithme ayant une complexité autre qu'exponentielle pour le résoudre de manière exacte. Une approche a été cependant testée, à savoir la *T-linearisation*.

4.5.1 T-linearisation

Il s'agit d'une méthode de linéarisation utilisée afin de calculer une borne inférieure pour le coût de la solution optimale, qui nous demeure encore inconnu. Cette borne sera par la suite exploitée par un algorithme de type *Branch-and-Bound*, qui consiste à explorer l'arbre des solutions tout en élaguant les branches de celui-ci pour réduire l'espace de recherche. La qualité de cette borne est donc un élément fondamental pour assurer une résolution rapide par l'élagage du plus grand nombre de branche possible.

Modélisation

Nous partons du principe que toute instance de problème est symétrique et que le cas échéant il est possible de la rendre symétrique. La formulation initiale étant très contraignante dans le cadre d'une T-linéarisation, la modélisation du problème a été transformée, afin notamment de la rendre la plus proche de la formulation du problème de sac à dos quadratique, sur lequel cette méthode a été initialement définie et ainsi adapter notre schéma plus aisément.

Nous allons ainsi utiliser une indexation de couple, en considérant la bijection $N : \mathbb{N}^2 \mapsto N$ telle que :

$$N(i,k) = (i * n) + k$$

Le problème est alors réécrit de la façon suivante :

$$\begin{aligned}
\min \quad & \sum_{r=1}^{n^2}\sum_{p=1}^{n^2} Q_{rp}.X_r.X_p \\
\text{sujet à} \quad & \sum_{i=1}^{n} X_{N(i,k)} = 1 \ \forall k \in]n] && (1)\\
& \sum_{k=1}^{n} X_{N(i,k)} = 1 \ \forall i \in]n] && (2)\\
& Q_{N(i,k)N(j,l)} = f_{ij}.f_{kl} && (3)\\
& X \in \{0,1\}^n && (4)
\end{aligned}$$

Cette formulation sera ainsi utilisée tout au long de l'application de cette méthode, en tant que formulation de référence.

Recherche de borne inférieure

La *T-linearisation* consiste à remplacer le terme quadratique de la fonction objectif par une nouvelle variable réelle t. L'ajout de cette variable implique alors la description du domaine suivant :

$$Y = \left\{ (x,t) \in \mathbb{R}^{n^2+1} \quad | \quad t \geq \sum_{r=1}^{n^2}\sum_{p=1}^{n^2} Q_{rp}.X_r.X_p, (1), (2), (3), (4) \right\}$$

et Y est alors approximée par les inégalités valides suivantes :

$$t \leq \sum_{r=1}^{n^2} \alpha_{\pi(r)}.X_{\pi(r)} \quad \forall \pi \in S_{n^2}$$

Notre problème se réécrit finalement sous cette forme :

$$\begin{aligned} \min \quad & t \\ \text{sujet à} \quad & t \leq \sum_{r=1}^{n^2} \alpha_{\pi(r)}.X_{\pi(r)} \quad \forall \pi \in S_{n^2} \\ & (1), (2), (3), (4) \end{aligned}$$

où $\alpha_{\pi(r)}$ est un sous-problème d'affectation linéaire défini comme ci-dessous :

$$\begin{aligned} \min \quad & \sum_{p=1}^{r-1} 2.Q_{\pi(r)\pi(p)}.X_p \\ \text{sujet à} \quad & X_{\pi(r)} = 1 \\ & (1), (2), (3), (4) \end{aligned}$$

Cependant, générer l'ensemble de ces contraintes est impossible, en effet, pour une instance de taille n et en considérant le nombre de permutation possible, nous avons un total de $n^2!$ contraintes. Afin de palier à ce problème, l'algorithme de génération de contrainte, telle que défini dans [3] a été utilisé. Chaque itération ajoutant une nouvelle contrainte tant que cette dernière est violée, ce qui est résolu par le problème de séparation. Le problème de séparation consiste à trouver une nouvelle contrainte, par un critère défini, qui viole la contrainte précédente et ainsi assure qu'il existe une meilleure solution au problème. Dans notre cas, nous utilisons un tri sur le vecteur solution afin de déterminer une nouvelle permutation à partir de la permutation l'identité pour ce tri donnée. Sans contrainte sur un problème donné, il a été prouvée que cette méthode garantie de déterminer la meilleure coupe [11] à ajouter, or dans notre problème intégrant des contraintes d'affectations, son efficacité n'est pas assurée et devient une simple heuristique.

Ceci nous donne ainsi une nouvelle coupe qui sera intégré au problème maître et permettra d'améliorer la solution courante, le problème étant résolu sur un domaine continu durant l'algorithme et ceux jusqu'à ce que l'on ne puisse plus trouver de coupe via cette méthode.

Résolution des sous problème linéaire

Chaque itération de notre algorithme nécessite de résoudre n^2 problèmes d'affectation linéaire. Il s'agit d'un problème relativement facile, pouvant être résolu par différentes méthodes. Deux choix ont été cependant retenus et testés :

11. Une coupe permet d'élaguer une partie de l'espace des solutions réalisables.

– Résolution à l'aide de l'algorithme de la *Méthode Hongroise*.
– Résolution en temps que simple problème linéaire à l'aide de l'algorithme du *simplex* par exemple.

La première solution à savoir utiliser la *méthode Hongroise*, consiste à rechercher une correspondance parfaite (ou *Perfect Matching*), dans un graphe biparti, ce qui peut être réalisé avec une complexité algorithmique en $O(n^3)$. Une première implémentation a été réalisée, avant d'attaquer la *T-linéarisation* elle - même.

Cependant, dans l'application des travaux suivants c'est une résolution par programmation linéaire qui a été utilisé, car plus facile à intégrer et dans la continuité des outils utilisés. Les résultats actuels étant ciblés sur la qualité et non la vitesse.

Résultats obtenus

Plusieurs implémentations ont été écrites en *C++* à l'aide du framework de programmation mathématique *ILOG Cplex*[12]. Et pour chaque version, la méthode a été testée sur un jeu d'instance issu de la QAPLIB[13] et comparée avec une borne de référence dénommée borne de Gilmore-Lawler [4] comme le montre le tableau 1. Afin d'améliorer la qualité de la borne obtenue, différents procédés ont été testés à savoir :

– L'utilisation à une itération donnée des coûts réduits calculés à l'itération précédente.
– L'utilisation de coefficients négatifs.

Les résultats présentés concerne la méthode combinant la *T-linéarisation* ainsi que les deux méthodes cités, qui donne les meilleurs résultats.

Taille	GLB - Gap (%)	T-linéarisation - Gap (%)	Différence de gap
12	14.86	12.80	2.06
14	12.25	11.47	0.78
15	21.26	19.03	2.23
16	38.69	37.93	0.76
17	19.86	19.41	0.45
18	17.31	15.72	1.59
19	30.45	22.18	8.27
20	16.80	15.67	1.13

TABLE 1 – Résultat obtenus pour une T-linéarisation

Le gap correspond à l'écart relatif à la valeur optimal en pourcentage. Il est calculé ainsi :

12. http ://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
13. Site regroupant l'ensemble des travaux sur le problème d'affectation quadratique ainsi qu'un grand nombre d'instance et des meilleures solutions trouvées

$$\text{gap} = ((\text{optimal} - \text{bound})/(\text{optimal})) * 100$$

Un gap nulle indiquant que nous avons une valeur optimale, l'objectif est d'obtenir un écart le plus proche de zero. Bien qu'une amélioration des écarts relatifs soit constatée, les résultats obtenus demeurent encore trop faible pour être exploités dans une résolution par séparation et évaluation progressive. La pertinence du traitement du problème de séparation par un simple tri sans contrainte est alors très vite remis en cause. En effet, de nombreuses permutations dans le voisinage de la dernière permutation calculés sont effectivement violée comme l'ont montré des tests sur un simple voisinage 2OPT [14] après arrêt de notre algorithme sur différentes instances. D'autres approches sont utilisées telles que l'intégration des contraintes sur le tri effectuée, malheureusement dans le cas où les coûts réduits sont utilisés, cela implique à chaque itération de tri des calculs trop nombreux, ce qui rend la méthode beaucoup trop longue et inexploitable. Cette approche a donc été mise en suspend au profit d'autre méthode, comme la linéarisation standard.

Linéarisation standard

Cette fois il s'agit de linéariser le problème tout entier, en augmentant considérablement le nombre de variable, une par produit possible. Notre fonction économique est alors réécrite ainsi :

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij}.d_{kl}.X_{ikjl}$$

Et les contraintes correspondantes sont intégrées afin d'assurer la cohérence du modèle linéarisé avec le modèle original. Cependant, cette méthode est bien trop gourmande en ressource et le nombre de variable crée est colossal, ce qui rend la résolution impossible.

Étant donné que le volume d'activité à traiter sera très grand (des milliers d'activités à relocaliser), l'utilisation de méthode heuristique a finalement été retenu.

4.5.2 Résolution par Heuristique

Les méthodes *heuristiques* sont des méthodes de résolution dite approchées, car elle n'assure pas l'obtention d'une solution optimale. De nombreuses techniques existent pour obtenir de telle solutions réalisables ayant le meilleur coût possible, tel que des *algorithmes gloutons*, des *algorithmes génétiques*, ou encore des méthodes basées sur l'*exploration de voisinage* telle que les *recherches tabous* ou le *recuit simulé*.

Pour la résolution d'un *problème d'affectation quadratique*, de nombreuses heuristiques ont été mis au point, certaines combinant plusieurs philosophies pour obtenir des résultats plus performant [15]. Dans le cadre de nos travaux, une méthode gloutonne suivi d'une exploration de voisinage a été mise en place.

14. Le voisinage dit 2OPT consiste à permuter deux entités d'une solution
15. De telles méthodes sont dite hybride

Heuristique gloutonne

Il s'agit de construire une solution réalisable, en sélectionnant à chaque itération une affectation à réaliser et ceux à l'aide d'un critère dit *glouton*. Ce procédé est ainsi répété jusqu'à ce que l'ensemble des activités soit affectées.

Différents critères ont été testés, la plupart basée sur l'évaluation d'un coût d'interaction entre les activités déjà affectées et celle que l'on projette de mettre en place.

Algorithm 2 Heuristique gloutonne pour le problème d'affectation quadratique

$\pi \leftarrow \{1\}^n$
$i, j, k, l \leftarrow \text{argmin}_{i,j,k,l}\{f_{ij}.d_{kl}\}$
$\pi_i \leftarrow k$
$\pi_j \leftarrow l$
$\Gamma \leftarrow \{i, j\}$
for $i = 1 \rightarrow n - 2$ **do**
  $a, b \leftarrow \text{argmin}_{a,b}\{\sum_{j\in\Gamma} f_{aj}.d_{b\pi_i} \mid a \notin \Gamma \wedge b \notin \pi\}$
  $\pi_a \leftarrow b$
  $\Gamma \leftarrow \Gamma \bigcup \{a\}$
end for

Dans cet algorithme, on cherche toujours à affecter une paire activité / localisation ayant le coût d'affectation le plus faible par rapport à la solution courante. Cette méthode s'exécute de manière quasi-instantané et donne les résultats suivants sur les instances de type Nugent [16].

Instance	Solution optimal	Solution obtenus	Gap (%)
nug12.dat	578	924	37.44
nug14.dat	1014	1500	32.40
nug15.dat	1150	1750	34.28
nug16a.dat	1610	2350	31.49
nug16b.dat	1240	1760	29.54
nug17.dat	1732	2434	28.84
nug18.dat	1930	2894	33.31
nug20.dat	2570	3508	26.74
nug21.dat	2438	4018	39.32
nug22.dat	3596	6010	40.17
nug24.dat	3488	5224	33.23
nug25.dat	3744	5432	31.07
nug27.dat	5234	8186	36.06
nug28.dat	5166	7736	33.22
nug30.dat	6124	8824	30.60

TABLE 2 – Résultat obtenus pour l'heuristique gloutonne

Nous avons ainsi un gap moyen de 33.18 %, ce qui est loin d'être correcte. Afin d'améliorer ce score nous avons appliqué par la suite une méthode d'exploration de voisinage.

16. Les instances de type Nugent sont les instances considérées comme les plus complexes à résoudre et sont souvent employées à fin de démonstration.

Local search : Amélioration par exploration de voisinage

L'exploration de voisinage (plus connu sous le nom de local search) consiste à construire le voisinage d'une solution réalisable pour trouver une solution de meilleur qualité, si elle existe. La notion de voisinage d'une solution étant définie comme l'ensemble des solutions obtenues en ne modifiant qu'une partie de cette dernière. Ainsi différents voisinages sont possibles comme le montre la figure 4, où de la solution x dérivent plusieurs voisinages plus ou moins grand. Il convient donc de trouver un voisinage permettant d'obtenir un espace de recherche le plus petit possible et de converger rapidement. Une application directe sur l'heuristique précédente utilisant un voisinage 2OPT permet ainsi de passer d'un gap de 33.18 % en moyenne à un gap de 4.33 %.

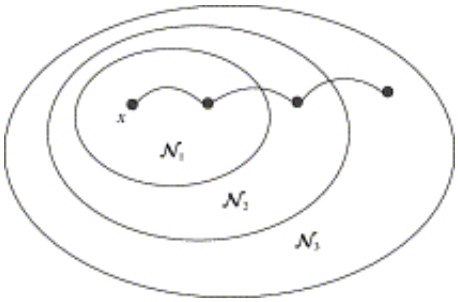

FIGURE 4 – Exemple de voisinage

Afin de trouver des résultats encore meilleurs, une petite modification a été réalisée. En effet les deux premières affectations orientent complètement la construction de la solution finale, ainsi que l'exploration du voisinage qui suit. Ainsi afin d'augmenter les chances de partir sur la meilleure affectation de départ, toutes les paires sont testées, on obtient ainsi pour une instance de taille n, n^4 solutions construite et amélioré. La solution ayant le coût le plus faible est alors conservée et cela donne des résultats similaires à l'heuristique $GRASP$[17], comme le montre la figure 5 avec un gap de 0.11 % en moyenne. Cependant, les performances en temps perdent en qualité et le temps d'exécution explose proportionnellement à la taille d'instance.

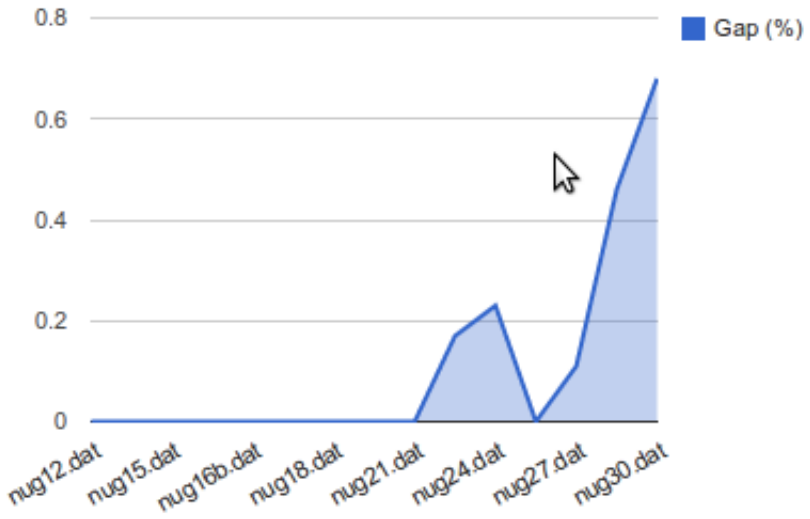

FIGURE 5 – Résultat obtenus sur les instances de type *Nugent*

17. *Greedy Randomized Adaptive Search Procedure*, heuristique hybride combinant construction gloutonne aléatoire et exploration de voisinage

4.6 Implémentation

Les différentes parties de notre système étant définies (y compris les méthodes d'optimisation pour le processus de relocalisation), il ne nous reste plus qu'à mettre en place notre architecture logicielle. Après la définition de notre architecture applicative, une phase d'analyse et de design a été effectué pour chaque sous-système avant d'écrire leur implémentation. Des batteries de tests unitaires et fonctionnels ont également été écrits afin de s'assurer du bon fonctionnement de nos différents algorithmes et structure de données.

4.7 Expérimentation et Résultats

Les différents tests ont été effectués sur des données portant sur la ville de *Zurich*. Ces données distribuées librement, ont l'avantage d'être déjà formatée pour le framework *MATSim* et donc demeurent facile à exploiter. Elles sont constituées de 1 % de la population à travers 8760 *agents* et 10281 *activités*.

Plusieurs types d'expérimentation ont ainsi été faite. Les premières, d'ordre purement techniques permis de s'assurer de la cohérence de notre algorithme et de la restitution des données. Par la suite, des études ont été menées, afin notamment de mettre en évidence une convergence théorique, ou encore de mettre en valeur différents indicateurs mathématique. Un certain nombre de paramètres ont alors été défini, à savoir les activités que nous relocalisons, car en effet, bien que le modèle proposé semble cohérent, de nombreuses données ne sont pas prises en compte : gestion des surfaces du bâti, ou encore la gestion des capacités d'accueil des infrastructures.

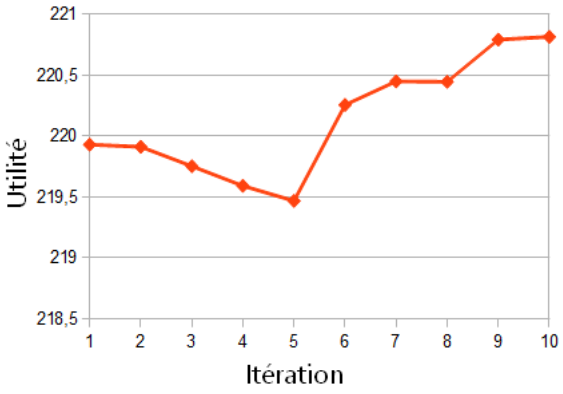

FIGURE 6 – Utilité moyenne sur 10 itérations

Finalement, le choix des activités relocalisables se fit sur les commerces qui sont les activités les plus apte à être déporté de manière réaliste et ceux sur une portion réduite. Seulement 33 % des commerces fut désigné comme relocalisable [18]. Après exécution de notre algorithme, l'évolution de la satisfaction de l'usager, des temps de trajet effectués ainsi que leurs distances ont été évalués à l'aide de mesure statistique, à savoir la moyenne, la variance, ainsi que l'écart-type.

Cependant, un certain nombre de tendance sont récurrentes et soulève des questions quant à la pertinence de la fonction d'utilité employée par *MATSim*. En effet comme nous pouvons le voir sur la figure 7, qui représente l'évolution de l'utilité moyenne sur 300 itérations, plus on relocalise, plus l'utilité augmente avant de se stabiliser :

18. Le choix de ces 33 % ayant été fait de manière arbitraire.

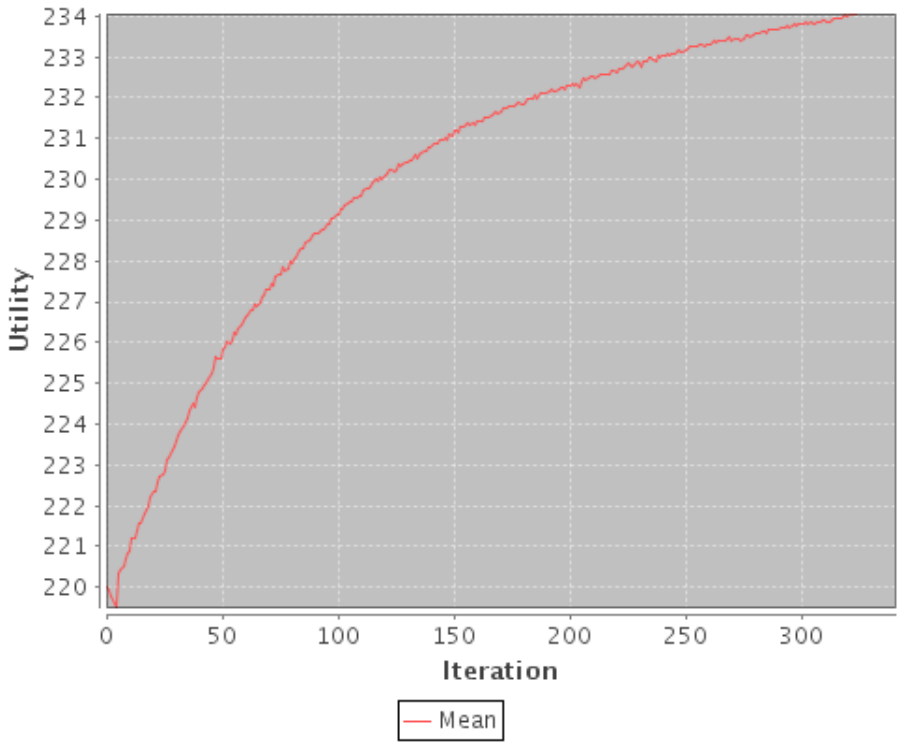

FIGURE 7 – Utilité moyenne sur 300 itérations.

Ce qui, considérant l'optimisation effectuée devrait signifier que nous avons effectivement une baisse des temps de transport. Or ce n'est pas le cas, comme nous pouvons le voir sur la figure 8, qui montre l'évolution des temps de transports moyen.

Un certain nombre de recherche et de tests ont été effectués afin de comprendre ces résultats, nous avons donc vérifié le seul facteur qui pouvait influencer la fonction d'utilité : le temps consommé sur une activité. Et en effet, nous avons pu observer des variations tout au long des itérations sur le temps passé sur une activité, ce qui pourrait expliquer l'incohérence des deux courbes. D'autres tests sont cependant en cours pour déterminer de manière plus précise, la cause de ce phénomène.

D'autres mesures ont également été effectué, notamment l'évolution des distances moyennes parcourues, qui elle, diminuent fortement sur le long terme, comme le montre la figure 9.

4.8 Test sur la ville de Lyon

Un certain nombre de tests sur la ville de Lyon sont également prévu. L'ensemble des données issu d'une enquête ménage / déplacement ont été convertis afin de pouvoir être exploité par notre outil. Les résultats pourront ainsi accompagné la prise de décision sur la construction d'une nouvelle rocade bouclant le périphérique Lyonnais, à horizon 2030.

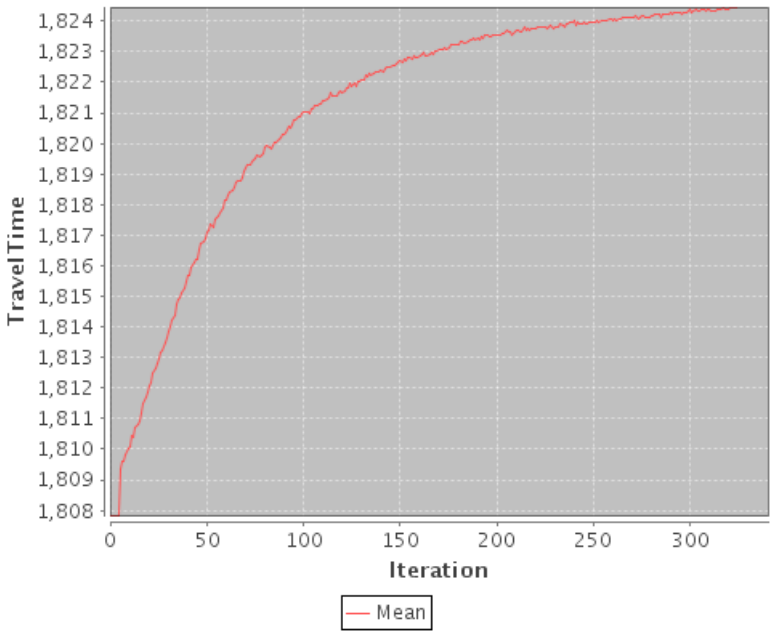

FIGURE 8 – Temps de transport moyen sur 300 itérations.

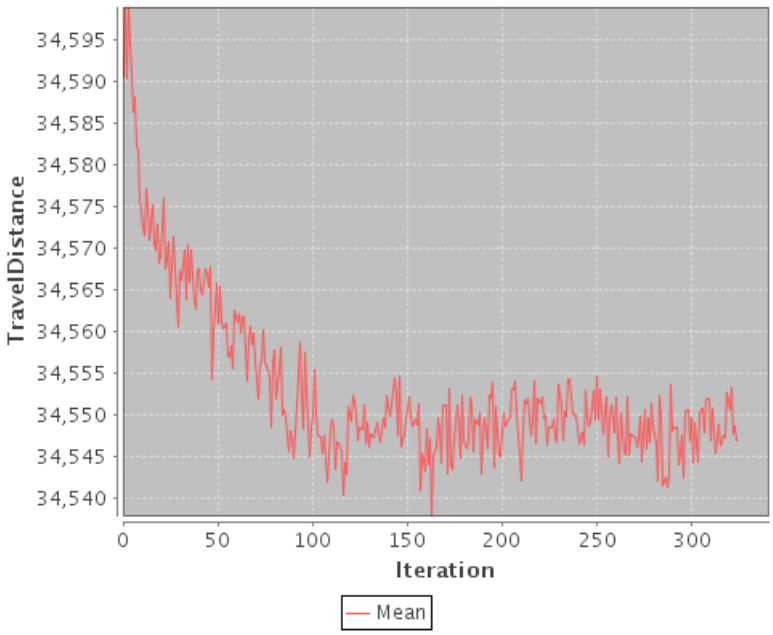

FIGURE 9 – Distances moyennes parcourus sur 300 itérations.

5 Un outil d'aide à la décision pour la construction des territoires urbains

Notre solution logicielle bien que fonctionnelle, nécessite une interface comprenant :

- Un outil de visualisation graphique.
- Un ensemble d'indicateur.

, et ceux afin de pouvoir analyser correctement et rapidement les résultats fournis par notre algorithme. Cela permet également une valorisation des travaux de recherche effectué en proposant un environnement complet pour le design urbain ou autre application d'ordre industrielle.

5.1 MATSim2GS

Ce composant Java a été écrit afin de pouvoir fournir une interface entre la structure de réseau employé par *MATSim* et celle exigée par la libraire de génération de graphe *GraphStream*[19] Ainsi nous pouvons obtenir un rendu graphique du réseau de transport comme le montre la figure 10. Ce qui nous permettra une visualisation des impacts et des relocalisations effectives.

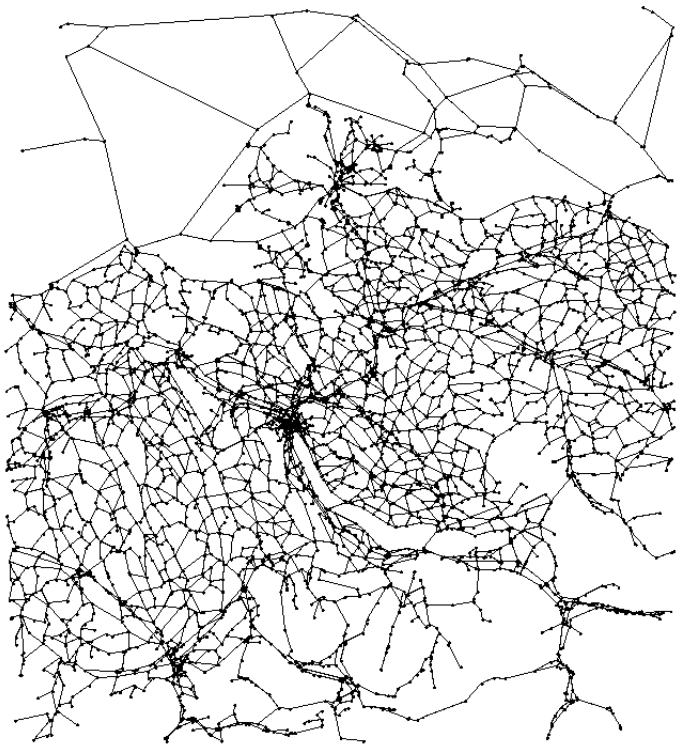

FIGURE 10 – Rendu généré pour la ville de *Zurich*

19. Bibliothèque graphique conçu par l'équipe du *Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes* situé au Havre. http ://graphstream-project.org/

5.2 Avancement

L'écriture d'un composant de visualisation étant terminé, la mise en place de la génération d'indicateur comme ceux utilisé pour les tests reste à faire. Une étape de packaging et de documentation sera alors réalisée pour permettre une utilisation aisée de notre outil, comme le montre la figure 11.

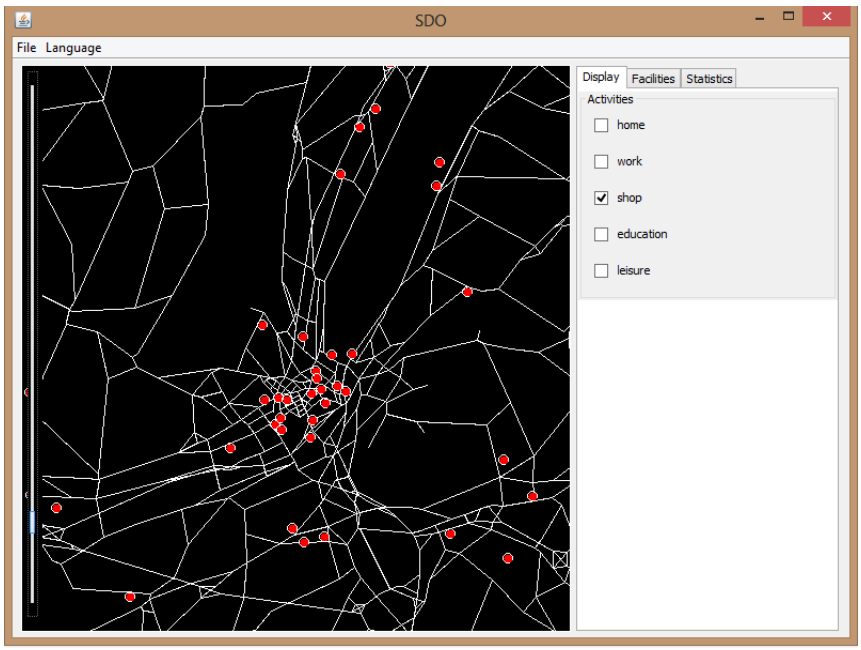

FIGURE 11 – Capture d'écran

Cependant, son utilisation reste limité par les capacités mémoires d'une machine de bureau standard. En effet sur de très grand réseaux de transports, il est impossible de généré un tel rendu.

6 Conclusion et perspective

Les travaux effectués ont permis d'appliquer de nombreux concepts de génie logiciel abordés au sein de ma formation au monde de la recherche. L'étroite collaboration avec l'équipe de l'*UMR Espace* m'a également permis de découvrir l'apport des géographes au domaine de la recherche opérationnelle, ainsi que l'implication dans un projet pluridisciplinaire.

Nous avons désormais une application complète pour notre étude sur les interactions urbaines, ainsi qu'un outil d'aide à la décision de support pour les autres équipes impliqués. Un certain nombre de tâches restent cependant à faire, à savoir :

- Suite de tests sur la ville de Lyon.
- Suite de tests afin de pouvoir expliquer l'incohérence des résultats obtenus sur Zurich.
- Valorisation des travaux effectués.

Références

[1] Programme de recherche et d'innovation dans les transports terrestres, http ://www.predit.prd.fr

[2] Interpretation of the logarithmic "utility of performing", the so-called *"Charypar-Nagel scoring function"* http ://www.matsim.org/node/651

[3] Rodrigues, D. Quadri, Michelon Philippe, Gueye Serigne, *A t-linearization to exactly solve 0-1 quadratic knapsack problems*, 2010 Actes de EWMINP, pp. 251-260.

[4] P.C. GILMORE. *Optimal and suboptimal algorithms for the quadratic assignment problem. SIAM Journal on Applied Mathematics*, 10 :305-31, 1962.