# Mode d’emploi

Simulateur pour prédire le développement d’une ville.

Aout 2019

Tristan Cordier

# Sommaire

# Présentation du projet

Une ville est un système très complexe et de nombreux facteurs influent son développement, situation géographique, contexte politique, culture de habitants etc… Ce simulateur se concentre sur la modélisation des comportements des ménages suivant certain critère comme l'accès aux transports et aux équipements. C'est un outil qui a pour but d'aider à l'aménagement du territoire.

## Contexte

À la suite de la collaboration entre le laboratoire I3S, l'Université de Los Andes (Bogota) et les géographes du laboratoire ESPACE de l'Université de Sophia Antipolis, une première version du simulateur à été réalisé. Les deux principaux objectifs du simulateur sont actuellement d'analyser des données socio-spatiales représentant le mode de vie des ménages et, d'identifier les dynamiques de regroupement géographique des familles dans les villes.

## Description

La simulation est basée sur des agents : entité discrète avec un certain comportement et ses propres objectifs. Les agents sont indépendants et peuvent êtres différents les uns des autres dans leur comportement. Ils ont une capacité d'autonomie et par conséquent sont capable de s'adapter et de modifier leur comportement. Il est alors possible de modéliser et de prédire les échanges entre agents, leurs changements de comportement ainsi que leur capacité d'adaptation.

Le simulateur utilise trois types d'agents, les promoteurs (promoter), les investisseur (investor) et les ménage (household). Les promoteurs ont pour but d'acheter les terrains à disposition en fonction de leur budget et de la valeur du terrain. Les investisseurs ont pour but d'acheter les propriétés et les louer. Les ménages occupent les propriétés mais peuvent aussi devenir investisseur.

Le simulateur manipule donc aussi des terrains (land) et des propriétés (property, lands achetés par des promoters). Pendant une étape de la simulation toutes ces classes sont mises à jour. Quand un household est mis à jour, les autres households prennent en compte ce changement, et en tiennent compte dans leurs décisions.

# Fonctionnement du projet

Dans cette partie nous verrons comment s'approprier le projet, l'installer et s'en servir.

## Technologies utilisées et mise en place

Avant de pouvoir lancer des simulations, il y a quelques éléments à mettre en place. Le projet s'appuie sur une base de données PostgreSQL avec l'extension PostGIS (qui permet de stocker des information spatiales). Il faut donc installer ces deux éléments. Il faut aussi s'assurer que PostgreSQL est bien configuré sur le port 5432 ou modifier le port en question dans le fichier ressource/application.properties.

Il faut alors créer une base de données « tomsa » (avec extension postgis), avec mot de passe « tomsa » et un utilisateur « tomsa ». Une fois ceci effectué, il faut lancer le script tomsa.sql (situé dans le dossier « db »)

Il est aussi possible de stocker les résultats dans une base de données MongoDB qu'il faut installer si besoin. Il peut être nécessaire d'installer la commande curl si elle ne l'est pas déjà.

## Utilisation des modules

Le simulateur est découpé en trois modules : simulator, data-extractor et interface. Nous allons voir comment utiliser chacun.

**Simulator :**

Ce module représente le cœur du simulateur, C'est ce module qui permet de lancer des simulations et de stocker les résultats. Les résultats d'une simulation se stabilisent généralement aux environs de 10 tours. Avant de lancer une simulation il faut installer les packages et dépendances avec la commande suivante dans un terminal dans le dossier « simulator »:

mvn install

Une fois terminé on peut lancer le module avec (il faut s'assurer que le serveur de la base de données est lancé sur le port 5432) :

mvn spring-boot:run

Le module est lancé, il écoute sur le port 9090 (modifiable dans le fichier ressource/application.properties) en l'attente d'une requête http qui lancera la simulation. Il y a 4 types de requêtes possibles qui peuvent se lancer dans un terminal à l'aide de la commande *curl*.

La requête « state » permet de lancer une seule simulation :

curl -H "Content-Type: application/json" -X POST localhost:9090/state -d "{\"storageType\": 1, \"nbrHousehold\": 100, \"nbrInvestor\": 100, \"nbrPromoter\": 100, \"num\":10, \"fileHousehold\" : \"householdAgent.apl\", \"fileInvestor\" : \"investorAgent.apl\, \"filePromoter\" : \"promoterAgent.apl\"}"

Il s'agit d'une requête post qui contient plusieurs paramètres.

- StorageType : 0 pour stocker les résultats sur MongoDB et autre pour stocker sur Postgres
- fileInvestor / fileHousehold / filePromoter : c'est le nom du fichier .apl de l'agent concerné, il doit se trouver dans le dossier "docs" du simulateur
- num : c'est le nombre de tours que va accomplir la simulation
- listOfEquipment : c'est la liste d'équipements qui vont être pris en compte lors de la simulation, la liste correspond à la liste des « codigo_upz » (code de district) des équipements (si ce paramètre n'est pas présent, la simulation se lance avec tous les équipements)
- listOfNetwork : même chose que pour équipements sauf qu'il s'agit de la liste des « id » des transports en base de données

La requête « statetab » permet de lancer plusieurs simulations en précisant les paramètres de chacune.

curl -H "Content-Type: application/json" -X POST localhost:9090/statetab -d "{\"simulationMessageList\":{\"storageType\": 1, \"nbrHousehold\": 50, \"nbrInvestor\":510, \"nbrPromoter\": 50, \"num\":10, \"listOfEquipment\":[85,81], \"listOfTransport\":[176,794], \"fileHousehold\" : \"householdAgent.apl\", \"fileInvestor\" : \"investorAgent.apl\", \"filePromoter\" : \"promoterAgent.apl\"},

{\"storageType\": 1, \"nbrHousehold\": 100, \"nbrInvestor\": 100, \"nbrPromoter\": 100, \"num\":15, \"listOfEquipment\":[85,81], \"listOfTransport\":[176,794], \"fileHousehold\" : \"householdAgent.apl\", \"fileInvestor\" : \"investorAgent.apl\", \"filePromoter\" : \"promoterAgent.apl\"}}"

Ici deux simulations vont être lancées en fonction de leurs paramètres.

La requête « batch » permet de lancer un grand nombre de simulations en combinant toutes les possibilités.

curl -H "Content-Type: application/json" -X POST localhost:9090/batch -d "{\"storageType\": 1, \"nbrHouseholdRange\": [50,60],\"nbrHouseholdStep\": 5, \"nbrInvestorRange\": [50,60],\"nbrInvestorStep\": 5, \"nbrPromoterRange\": [50,60],\"nbrPromoterStep\": 5, \"numRange\":[10], \"numStep\":1, \"listOfEquipment\":[85,81,90,105], \"listOfTransport\":[176,794,254], \"fileHousehold\" : \"householdAgent.apl\", \"fileInvestor\" : \"investorAgent.apl\", \"filePromoter\" : \"promoterAgent.apl\"}"

Voici les paramètres possibles.

- fileInvestor / fileHousehold / filePromoter : c'est le nom du fichier .apl de l'agent concerné, il doit se trouver dans le dossier "docs" du simulateur
- numRange, nbrHouseholdRange, nbrInvestorRange, nbrPromoterRange : correspond au domaine de variation de ces paramètres. Toujours [min, max] avec max inclu, il est aussi possible d'entrer une seule valeur qui ne variera pas (Défault 50). Exemple : "nbrHouseholdRange":[50,60] , le nombre sera compris entre 50 et 60
- numStep, nbrHouseholdStep, nbrInvestorStep, nbrPromoterStep : correspond au pas de variation dans la range (Défault 1). Exemple "nbrHouseholdRange":[50,60], "nbrHouseholdStep": 5 , household prendra les valeurs 50,55 et 60.
- listOfEquipment : c'est la liste d'équipements qui vont être pris en compte lors de la simulation (si ce paramètre n'est pas présent, la simulation se lance avec tous les équipements)
- listOfNetwork : même chose que pour équipements.
- StorageType : 0 pour stocker les résultats sur MongoDB et autre pour stocker sur Postgres

La requête « stop » permet l'arrêt de la simulation.

curl -H "Content-Type: application/json" -X POST localhost:9090/stop -d "{}"

**Data-extractor :**

Ce module a pour objectif l'extraction des données sous forme de csv. Il y a aussi un script R servant à visualiser les résultats des simulations (seulement les simulations stockées dans Postgres). De même que pour simulator, il faut installer les packages et lancer le module.

mvn install

mvn spring-boot:run

Il n'y a qu'une seule requête pour extraire les données. Le serveur de la base de données à extraire doit être lancé pour pouvoir extraire (server PostgreSQL ou MongoDB).

curl -H "Content-Type: application/json" -X POST localhost:9091/extract -d "{\"entity\":[\"household\",\"investor\"], \"idSimulationRange\":[10,20], \"storageType\": 1}"

Voici la signification des paramètres :

- StorageType : 0 pour récupérer les résultats dans MongoDB et autre pour récupérer sur PostgreSQL
- Entity : Tableau contenant les informations que nous voulons extraire. Peut prendre les valeurs, « household, configuration, property, investor, promoter, land ».
- idSimulationRange : [min,max] des id simulation à extraire, max inclus. Le pas est de 1.

La commande ci-dessus va extraire toutes les données concernant les houshold et investor concernant les simulations de 10 à 20. Deux fichiers CSV seront créés, un pour les household et un pour les investor.

**Interface :**

Ce module permet de passer par une interface web pour envoyer les requêtes http (remplace la commande *curl*). Pour installer et lancer le module exécuter les commandes suivantes :

mvn install

mvn appengine :devserver

Une fois effectué, ouvrir un navigateur et rentrer les URL suivantes :

- "localhost:8080/one" : pour l'interface de lancement d'une seule simulation (Interface pour la requête "localhost:9090/state" vu précédemment)
- "localhost:8080/multi" : pour l'interface de lancement de multiple simulations (Interface pour la requête "localhost:9090/statetab")
- "localhost:8080/extract" : pour l'interface d'une extraction (Interface pour la requête "localhost:9091/extract")

Les modules correspondants doivent être lancé pour pouvoir répondre à la requête, data-extractor ou simulator.

## Où le modifier

Pour effectuer certaines simulations, il est nécessaire de modifier certaines parties directement dans le code. Il peut être utile de changer la fonction de satisfaction des households, cette fonction « getSatifaction » est ligne 85 de la classe urbanSimulation\entities\agents\Household.java. On peut par exemple la rendre aléatoire pour voir comment se comportent les agents.

Une autre possibilité serait de modifier la valeur d'un terrain, par exemple en ne prenant en compte que les réseaux de transport sans les équipements. Pour ce faire il faut modifier à cinq endroits le code. Il faut changer les paramètres de la fonction setUtility().

purchasable.setUtility()

Ces cinq lignes se trouve : dans la classe urbanSimulation\entities\agents\Household.java, la méthode « invest » (ligne 107), la méthode « rentProperty » (ligne 137), la méthode « buyProperty » (ligne 281), dans la classe urbanSimulation\entities\agents\Promoter.java la méthode « buyLand » (ligne 122) et dans la classe urbanSimulation\entities\agents\Investor.java la méthode « invest » (ligne 130).