ENSG Géomatique

ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES

LAREG

Documentation

Algorithme Mouv'Ant

issu du projet recherche du cycle ingénieur

École Nationale des Sciences Géographiques

Jourdana Sylvain
Messal Loïc
Élèves-Ingénieurs de l'ENSG

17 Février 2016

« Rien ne résiste à un acharnement de fourmi. »
Les travailleurs de la mer (1866)
Victor Hugo

Table des matières

Table des figures

1 Présentation du projet

1.1 Définitions et contexte

Depuis 2008, le Laboratoire de Recherche en Géodésie utilise des algorithmes d'optimisation stochastique pour résoudre des problèmes combinatoires posés par les recherches en géodésie spatiale et non solubles par les méthodes classiques. Ces problèmes concernent la recherche de sous-réseaux de stations d'observation au sol, la recherche d'orbites optimales pour un satellite ou pour une constellation de satellites, etc. Pour tous les problèmes considérés jusqu'à présent, le LAREG utilise des algorithmes génétiques dont les performances ne sont pas optimales.

L'algorithme MouvAnt apporte une nouvelle méthode de résolution pour ces problèmes. MouvAnt est fondé sur le principe de l'optimisation par colonie de fourmis [1]. Il s'agit en bref de reproduire l'intelligence collective mise en œuvre par une fourmilière lors de la recherche de nourriture.

1.2 Spécifications

Bien que le principe puisse s'appliquer à chacun de ces problèmes, MouvAnt a été développé pour résoudre la détermination de meilleurs sous-réseaux de stations de télémétrie laser utilisés pour calculer les paramètres de rotations de la Terre.

1.3 Prérequis

L'algorithme a été développé avec le langage Python dans sa version 3 et fait appel à une bibliothèque codée en C. Il est alors nécessaire de disposer

— d'un interpréteur Python pour éxecuter l'algorithme (plus d'informations sur `https://wiki.python.org/moin/BeginnersGuide/Download`),

— et d'un compilateur de code C (plus d'informations sur `https://gcc.gnu.org/install/index.html`).

Le script Python fait appel à la bibliothèque numpy [2] et la bibliothèque standard.

1. Aussi appelé ACO pour Ant Colony Optimization
2. `https://docs.scipy.org/doc/numpy-dev/user/quickstart.html`

Le script C n'a de dépendance qu'avec la bibliothèque standard.

L'algorithme a également besoin de données en entrée. L'utilisateur de MouvAnt devra donc posséder son jeu de données spécifique à la semaine de télémétrie laser. Ces fichiers doivent respecter le format des fichiers grgs.pos+eop.#####.v60_MODIF_para.dat et grgs.pos+eop.#####.v60_MODIF_tran.dat fournis par le LAREG, où ##### correspond au numéro de la semaine considérée.

2 Principes généraux de l'algorithme

Comme il a été dit dans les pages précédentes, MouvAnt est un algorithme d'optimisation par colonie de fourmis spécifique au problème de détermination de sous-réseaux de stations au sol. La sélection du meilleur sous-réseau se base sur celui qui minimise « au mieux » [3] les 3 critères (résultants du calcul de système de référence). De ce fait, MouvAnt est un algorithme multicritère.

2.1 Quelques définitions

Stations

Il s'agit d'un objet représentant les stations de télémétrie laser. Des probabilités d'être choisie par les fourmis y sont associées, ainsi qu'une quantité de phéromones déposée afin d'orienter le tirage de cette station. Ces stations sont numérotées pour les distinguer.

Datum

Il s'agit d'une chaîne de longueur égale au nombre de station. Chaque caractère est un nombre binaire traduisant la sélection de la station dans le sous-réseau mis en jeu.

Sigma

Les sigmas sont les écarts-types suivant les 3 axes de rotation de la Terre issus du calcul des effets de système de référence pour un sous-réseau donné. Ils sont donc stockés dans les variables `sigmax`, `sigmay` et `sigmaz`. Ce sont ces sigmas que MouvAnt cherche à minimiser pour déterminer les meilleurs sous-réseaux de stations.

3. Cette notion revient à prendre le sous-réseau le plus proche de l'origine dans l'espace des 3 critères.

2.2 Les grandes étapes

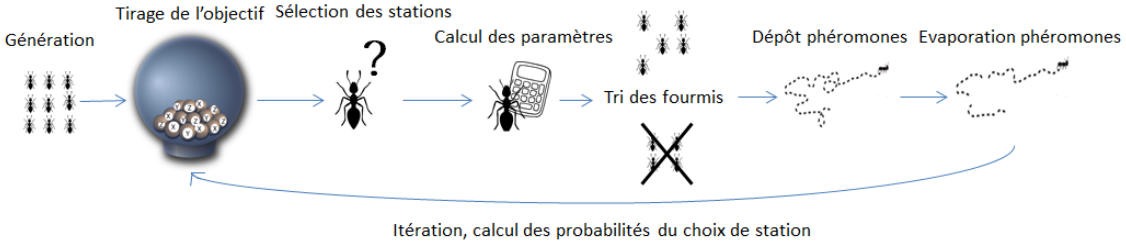

FIGURE 1 – Schéma du fonctionnement de MouvAnt

Génération des fourmis

Les fourmis sont des objets possédant des attributs permettant de leur fixer le nombre de stations qu'elles doivent prendre en compte, et de fait leur datum. Les valeurs des sigmas leur sont également associées. Étant les objectifs à minimiser, les sigmas sont préalablement initialisés à une valeur extrême (l'infini). L'objectif sélectionné est également un attribut de ces objets.

Tirages aléatoires

Ces tirages aléatoires présentés ci-dessous vont conditionner les fourmis dans la recherche de solutions au problème.

L'objectif L'aspect multicritère de MouvAnt repose ici. Chacune des fourmis va sélectionner de manière aléatoire et conserver parmi ses attributs le critère qu'elle va chercher à minimiser (soit `sigmax`, soit `sigmay`, soit `sigmaz`). Elle cherchera alors à minimiser cet objectif. Le problème devient similaire à une optimisation d'un seul critère.

Le nombre de stations Chaque fourmi tire au sort un nombre de stations qu'elle doit considérer pour établir son sous-réseau. Ce nombre est borné. Le nombre de stations minimal peut être modifié par l'utilisateur de MouvAnt. Le nombre maximal est quant à lui égal au nombre de stations de télémétrie laser qui ont fourni des données pour la semaine étudiée.

Sélection des stations

Arrivé ici, chaque fourmi possède un critère à minimiser et le nombre de stations `NbStation` à sélectionner. Chacune d'entre elles va choisir aléatoirement `NbStation` stations. Ce choix est pondéré selon les probabilités pour le critère considéré attribuées à chaque station. Leur datum est alors généré et enregistré dans un fichier.

Calcul des sigmas

L'algorithme va parcourir le fichier contenant les datums. Pour chacun d'eux, MouvAnt va calculer les sigmas correspondants (c'est ici qu'est utilisée la bibliothèque codée en C pour des questions de performance). Les valeurs des sigmas pour chaque datum sont alors attribuées aux fourmis correspondantes.

Dépôt de phéromones

Le dépôt de phéromones est l'étape qui ajoute l'aspect d'intelligence collective à MouvAnt. Elle consiste à orienter les fourmis de l'itération suivante dans leur choix des stations. Les phéromones sont déposées en deux fois en raison de l'aspect multicritère.

MouvAnt parcourt tout d'abord les fourmis de l'itération en cours. L'algorithme ajoute des phéromones aux stations pour celles qui minimisent un critère. Par exemple, si a, b et c sont trois fourmis, avec a et c qui ont tiré le critère `sigmax` alors que b a tiré `sigmaz`. MouvAnt compare a et c et ajoute des phéromones aux stations de c si `c.sigmax < a.sigmax` et inversement. Comme b est la seule fourmi à optimiser `sigmaz` (et donc nécessairement celle qui minimise ce critère), MouvAnt dépose également des phéromones sur les stations de b.

Ensuite, MouvAnt accumule les solutions de l'itération en cours avec celles des itérations précédentes. Cet ensemble de solutions est alors trié. Cette étape est détaillée ci-dessous. Les meilleurs fourmis reçoivent alors une récompense en terme de phéromone : MouvAnt dépose des phéromones sur les stations qu'elles ont tiré.

Tri des fourmis Le tri consiste à extraire les solutions qui participent à minimiser les 3 sigmas à la fois. MouvAnt utilise la relation de Pareto pour déterminer ces solutions dites « non dominées ». Cela consiste à comparer deux fourmis par rapport à une liste d'objectifs, par exemple les fourmis a et b. Pour chaque objectif, la relation va comparer la valeur de cet objectif pour la fourmi a et pour la fourmi b. Si `a.objectif1 < b.objectif1` alors a domine au moins une fois b. Cette information est enregistrée dans la variable `flag1`. Maintenant si `a.objectif2 > b.objectif2` alors b domine au moins une fois a et est stocké dans la variable `flag2`. Il y a alors trois cas possibles :

- `flag1` est le seul levé. Cela veut dire que pour tous les objectifs dans la liste, la fourmi a a dominée la fourmi b. Donc a domine b.
- `flag2` est le seul levé. Cela veut dire que pour tous les objectifs dans la liste, la fourmi a a été dominée par la fourmi b. Donc b domine a.
- `flag1 et flag2` ont été levés. Alors la fourmi a a dominée b au moins une fois et la fourmi a a également été dominée par la fourmi b au moins une fois. En d'autres termes, les deux fourmis sont équivalentes au sens de Pareto.

En comparant toutes les fourmis deux par deux, pour déterminer les meilleures fourmis, il suffit alors de conserver celles qui n'ont jamais été dominées avec le `flag2` uniquement. Cet ensemble porte le nom de solutions non dominées, puisque ce sont les fourmis qui dominent les autres et qui sont équivalentes à elles-mêmes, et donc ne sont jamais dominées.

Évaporation des phéromones

Les phéromones déposées s'évaporent d'un certain pourcentage. Cela permet à MouvAnt de se libérer d'un minimum local.

Surtout au début de l'algorithme, des stations tirées au hasard peuvent faire parti des solutions non dominées mais ne contribuent pas à la construction des bonnes solutions. Ce tirage est alors une erreur. Réduire la quantité de phéromones d'un certain pourcentage pour toutes les stations permet de réduire les chances associées à une station d'être tirée. Ainsi, cette station peut revenir à un niveau de phéromones égales à une station qui n'aurait pas été exploitée mais qui contribue à la construction des bonnes solutions. Ainsi, l'évaporation permet à MouvAnt d'exclure certaines pistes et favorise la recherche d'autres solutions qui sont potentiellement meilleures.

Itérations & mise à jour des probabilités

L'ensemble des solutions non dominées doit converger. Pour cela, il est nécessaire de faire plusieurs itérations afin que le tirage pondéré, le dépôt de phéromones et l'évaporation produisent leurs effets. Avant de passer à l'itération suivante, il faut recalculer les probabilités qu'une station soit tirée. La probabilité associé à une station pour un critère donnée est la proportion de phéromones pour ce critère par rapport à la somme des phéromones déposées rapportées à ce critère.

3 Paramétrage

L'algorithme MouvAnt s'adapte automatiquement au nombre de stations au sol étant considérées pour une semaine donnée. Certains paramètres sont toutefois personnalisables.

3.1 Paramètres gérés automatiquement

Noms des fichiers en sortie

Tous les fichiers en sortie de l'algorithme commenceront par le radical des fichiers d'entrée. Seules leurs extensions changent selon leur contenu.

Nombre de fourmis

La variable `nb_fourmis` désigne le nombre de fourmis qui vont être déployées dans l'algorithme. Les valeurs attribuées dépendent uniquement du nombre de stations au sol de la semaine considérée, et sont semblables au nombre d'individus dans l'algorithme epsilon-MOGA.

Nombre d'itérations

La variable `max_it` désigne le nombre d'itérations effectuées. Les valeurs attribuées dépendent uniquement du nombre de stations au sol de la semaine considérée, et sont semblables au nombre de générations dans l'algorithme epsilon-MOGA.

3.2 Paramètres personnalisables

Lors de sa conception, les développeurs de MouvAnt ont décidé de fixer des valeurs par défaut à ces paramètres. Ces valeurs arbitraires ont permis d'obtenir de très bons résultats sur plus de dix années de données. Les variables présentées ci-dessous ont alors été mises sous silence. Vous pouvez les modifier à votre guise, mais ni le fonctionnement de l'algorithme ni la qualité du résultat ne pourraient être garantis.

Nombre de phéromones déposées

La variable `k` désigne la quantité de phéromones déposées. Elle sert surtout à déposer des phéromones lors de l'initialisation de l'algorithme. On a alors : `pheromone_initiale=k`.

Évaporation

A chaque itération, les phéromones déposées s'évaporent pour que MouvAnt soit capable de se libérer d'un minimum local. Cette quantité stockée dans la variable `p` est alors égale à $k \times 0.1$.

Contraintes

Afin de calculer les effets de systèmes de références (dont les écarts types servent à qualifier le choix des stations), il est nécessaire de tirer au moins 3 stations distinctes. La variable `NbStation_min` $= 3$ considère cette contrainte.

Bornes des probabilités

Chacune des fourmis choisit les stations qu'elle considère avec un tirage aléatoire. Ce tirage est conditionné par les deux paramètres décrits ci-dessous.

Probabilité minimale La variable `proba_min`, fixée à 0.01 par défaut, fixe la probabilité minimale qu'une station soit choisie. Cela permet de « laisser une chance » à la station d'être tirée, alors qu'elle ne semble pas être un bon candidat pour la solution optimale. Par comparaison aux algorithmes génétiques, il s'agit de conserver la capacité de mutation des chromosomes. MouvAnt est alors capable de se libérer d'un minimum local.

Probabilité maximale De même, la variable `proba_max`, fixée à 0.99 par défaut, fixe la probabilité maximale qu'une station soit choisie. Cela permet aux fourmis de choisir une autre station bien que la station ayant sa probabilité égale à `proba_max` soit un bon candidat pour la solution optimale. Par comparaison aux

algorithmes génétiques, il s'agit de conserver la capacité de mutation des chromosomes. MouvAnt est alors capable de se libérer d'un minimum local.

4 Compilation

Le code C comprend toutes les fonctions qui vont permettre de calculer les écarts-types à partir des datums. Le code source doit être compilé. Pour cela, il suffit d'executer dans une console la commande :

```
gcc -shared -Wl,-soname,mouvant -o mouvant.so -fPIC mouvant.c
```

Cette commande va créer un fichier `mouvant.so` qui est une bibliothèque partagée compilée pour la machine. Elle sera exploitée par le code Python `mouvant.py`.

5 Exécution

Les fichiers suivants doivent être placés dans un même répertoire :
— `mouvant.py` ;
— `global.h` ;
— `mouvant.c` ;
— `mouvant.so` ;
— les fichiers de données de la même semaine :
  — `grgs.pos+eop.#####.v60_MODIF_para.dat`
  — `grgs.pos+eop.#####.v60_MODIF_tran.dat`

Pour lancer MouvAnt, il suffit alors :

1. D'ouvrir une console ;
2. Se placer dans le répertoire contenant les fichiers listés ci-dessus ;
3. Exécuter la commande suivante :

```
python3 mouvant.py grgs.pos+eop.#####.v60_MODIF_para.dat grgs.pos+
    eop.#####.v60_MODIF_tran.dat
```

Dans la section 8 page 20 est détaillé l'outil `multiProcess` qui a été adapté par nos soins pour lancer MouvAnt en parallèle sur un nombre défini de processeurs. Cela nous a permis de traiter la centaine de fichiers de données en une seule ligne de commande.

6 Résultats

Les résultats obtenus au cours et en fin d'algorithme sont stockés dans cinq différents fichiers.

6.1 Datum

Ce fichier contient chaque datum correspondant au trajet d'une fourmi. Pour une colonie de 200 fourmis par exemple il contiendra 200 lignes, chacune attestant donc du choix de stations d'une fourmi. C'est ce fichier qui sera entré en paramètre du code en C.

6.2 Ecart-type

Il s'agit du fichier créé en sortie du code en C. Il contient les valeurs de chaque écart-type calculées pour chaque datum passé en entrée. Ce fichier va alors être renvoyé dans le code en python afin de mettre à jour le taux de phéromones de chaque station.

6.3 Solutions non-dominées

Ce fichier contient l'ensemble des solutions non-dominées au sens de l'ensemble de Pareto. Il est créé après la dernière itération et permet notamment d'utiliser certains outils comme le calcul des statistiques.

6.4 Meilleur datum

Il s'agit d'un fichier contenant le meilleur datum trouvé par l'algorithme. Ce dernier est calculé parmi l'ensemble des solutions non-dominées obtenu après la dernière itération de l'algorithme.

7 Perspectives d'évolution

Bien que fournissant actuellement de très bons résultats, l'algorithme MouvAnt développé semble posséder des perspectives d'amélioration. Ces probables améliorations suivent trois axes de recherche : le choix du nombre de stations de chaque fourmi, le choix de l'objectif et les valeurs des paramètres fixés initialement.

7.1 Nombre de stations

Le choix du nombre de stations que chaque fourmi va choisir est donc généré aléatoirement lors de la création de celle-ci. Or ce choix semble limiter la recherche d'un sous-réseau optimal. En effet, lorsqu'une fourmi va se rapprocher d'un sous-réseau optimal, elle pourrait être bridée. Cela au sens où il est possible qu'elle soit obligée de choisir de nouvelles stations afin de remplir son quota ou bien inversement qu'elle ne puisse plus choisir de nouvelles stations alors que si la fourmi avait choisi ne serait-ce qu'une seule station supplémentaire, le sous-réseau aurait peut-être été optimal. Pour se faire une idée du résultat obtenu avec une méthode qui ne « briderait » pas les fourmi, il semblerait intéressant de regarder si les sous-réseaux proches (peu de stations différentes, datums comprenant un nombre de stations choisies proche) d'une solution optimale sont presque optimaux. Une solution envisageable de mise en place d'une telle méthode serait non pas de seulement tirer aléatoirement le nombre de stations que la fourmi va choisir mais aussi de guider la fourmi sur son parcours. Lorsqu'il est question de guidage il peut s'agir d'une mise à jour du nombre de stations restant à choisir par la fourmi au cours de son trajet. Pour ce faire, il peut être question d'une prise en compte des probabilités associées à chaque station restante : si les probabilités restantes sont élevées (supérieures à un seuil) la fourmi continue à tirer des stations sinon elle s'arrête. Attention toutefois avec une telle méthode à ne pas s'enfermer rapidement dans un minimum local qui serait une solution proche des solutions optimales sans nécessairement en être une. Il conviendrait alors de fixer un seuil empiriquement ce qui reviendrait une nouvelle fois à faire varier un paramètre dont la portée n'est pas réellement connue.

7.2 Choix de l'objectif

Le choix de l'objectif est lui aussi tiré aléatoirement par chaque fourmi. Il pourrait être préférable d'équirépartir les quantités de chaque objectif sur l'ensemble

des fourmis de la colonie. Cela peut se faire dès l'initialisation ou bien au fur et à mesure des itérations en réadaptant les quantités de fourmi par objectif selon que certains aient été tirés plus souvent. Aussi, il pourrait être préférable d'augmenter le nombre de fourmis s'intéressant aux objectifs sigma x et sigma y afin de leur donner une plus grande importance si c'est ce qui est souhaité car ils semblent avoir une plus grande influence sur le choix du sous-réseau optimal.

7.3 Paramètres d'entrée

Les travaux effectués ont été réalisés en fixant les paramètres personnalisables par défaut ce qui a fournit de bons résultats. Il pourrait néanmoins être intéressant d'effectuer une batterie de tests sur l'influence de ces différents paramètres sur le résultat final. Aussi, il est fort probable que ces paramètres soient propres à chaque problème d'optimisation et il faudra alors fixer ces paramètres empiriquement suivant le problème qui est étudié.

8 Outils divers

8.1 Parallélisation

Présentation de multiProcess

MultiProcess est un outil qui permet d'exécuter MouvAnt sur tous les fichiers de données contenus dans le répertoire où se trouve MouvAnt. Bien entendu, le fichier `multiProcess2.sh` doit s'y trouver également. Il a l'avantage d'exécuter MouvAnt en parallèle sur un nombre défini de processeurs. Cela permet en outre d'exploiter au maximum les machines de calculs, tout en laissant des processeurs disponibles pour d'autres programmes. Pour modifier le nombre de processeur à utiliser, il suffit d'éditer le script et modifier la variable `MAX_NPROC` à votre convenance. Pour exécuter multiprocess, il suffit de lancer les commandes suivantes :

— Pour autoriser l'execution de multiprocess :

```
sudo chmod +x multiProcess2.sh
```

— Pour exécuter multiprocess en étant dans le dossier de données :

```
./multiProcess2.sh
```

Sur machine distante

Il arrive parfois d'avoir besoin de lancer les calculs sur une machine distant. La première étape consiste à placer les données dans un dossier de cette machine. Sous Linux, il suffit de se connecter en SFTP [4] à l'aide de la commande :

```
sftp nomUtilisateur@nomMachine
```

Une fois connectée, il faut jongler entre deux machines. Voici une liste de commandes pratiques :

— `cd [..|nomDossier]` : permet de se déplacer dans l'arborescence de fichier de la machine distante
— `lcd [..|nomDossier]` : permet de se déplacer dans l'arborescence de fichier de la machine locale
— `mkdir [nomDossier]` : permet de créer le dossier nomDossier sur la machine distante

4. Secure File Transfert Protocol

— `rm -r nomDossier` : permet de supprimer le dossier nomDossier sur la machine distante
— `get [*|file]` : permet de transférer tous les fichiers ou le fichier file de la machine distante sur la machine locale
— `put [*|file]` : permet de transférer tous les fichiers ou le fichier file de la machine locale sur la machine distante
— `bye` : permet de quitter la connexion

L'objectif du SFTP est de créer l'arborescence décrite dans la partie 5 page 16, en n'oubliant pas d'y rajouter le fichier `multiProcess2.sh`.

Tout est prêt pour lancer l'exécution de multiprocess. En utilisant le protocole SSH [5], il est possible d'exécuter des commandes sur une machine distante, comme s'il s'agissait de la machine locale. Pour cela, il suffit d'exécuter la commande suivante :

```
ssh -X nomUtilisateur@nomMachine
```

Les calculs avec MouvAnt sur plusieurs années de données durent plusieurs heures. Si la connexion entre les deux machines se coupe, ou si le terminal qui est connecté en SSH se ferme, les scripts en cours d'exécution lancés à partir de la machine locale se terminent. Il est alors préférable de dissocier le terminal de la commande exécutée au moyen de la commande `nohup`. Finalement, pour exécuter multiprocess sur une machine distante, il suffira de lancer :

```
nohup␣./multiProcess2.sh␣>␣/dev/null␣&
```

La connexion peut alors être fermée avec la commande `exit` sans que les calculs ne s'arrêtent.

8.2 Regrouper les meilleures solutions

Les meilleurs datums pour chaque semaine sont enregistrés dans le fichier portant l'extension `.meilleurDatum`. Pour certaines applications, il peut être intéressant de concaténer les meilleurs datums par ordre chronologique. La commande ci-dessous permet réaliser la concaténation et créer le fichier `fullData.txt` contenant tous les datums.

```
for f in *.meilleurDatum;
do
(
```

5. Secure SHell

```
        cat $f;
        echo ''
) >> fullData.txt;
done
```

8.3 Calcul de statistiques sur les résultats

Pour estimer la convergence de l'algorithme, il a fallu comparer l'ensemble des solutions non dominées issu d'une semaine et déterminé par MouvAnt à l'ensemble des solutions non dominées, calculé pour toute les combinaisons possibles. Le fichier script `statistiques.py` génère le fichier de statistiques contenant une batterie de tests pour tous les fichiers d'un dossier.

8.4 Représentation des meilleures solutions

Solutions non dominées

L'outil ParetoTotal.py permet de représenter graphiquement les écarts-types. Il s'agit juste de lire un fichier contenant les écarts-types, et de tracer `sigmaY` = f(`sigmaX`), `sigmaZ` = f(`sigmaX`) et `sigmaZ` = f(`sigmaY`).

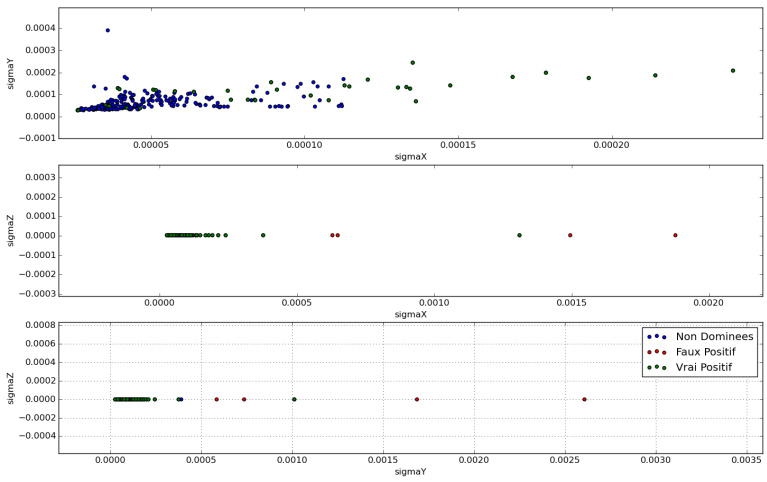

FIGURE 2 – Graphique des écarts-types

Cartographie des stations

L'outil de visualisation des solutions non dominées utilise l'API Leaflet pour afficher des marqueurs sur les stations choisies (lecture d'un datum) sur une carte. Il suffit d'ouvrir le fichier html sur un serveur php pour faire fonctionner cet outil.

FIGURE 3 – Répartition des stations pour une solution non dominée trouvées par MouvAnt