Optimisation et système multi-agents

Etienne Krönert, Christopher Renaud Chan, Louis Maestrati, Alexandre Monlouis, Martial Morisse, Clément Meunier

Introduction:

Il s'agit ici de prolonger le travail déjà effectué lors de notre premier travail à rendre. Le problème du PVC étant un problème NP-Complet, notre première démarche a été de lui appliquer des démarches d'optimisation approché, ce fut fait au travers de 3 principaux algorithmes: l'algorithme génétique, le tabou et le recuit, dont leur démarche s'inspire de principe physique courant. L'idée est ici de combiner ces trois algorithmes en établissant un échange et dialogue entre eux. Nous avons utilisés pour cela des agents contrôleurs qui permettent d'effectuer les algorithmes à la chaîne un grand nombre de fois. Plusieurs démarches (3) ont été envisagé: celle de la méthode RGT, la méthode linéaire et enfin la méthode simultanée.

A) Etude des Algorithmes pour des problèmes de grandes tailles

a) Présentation des algorithmes

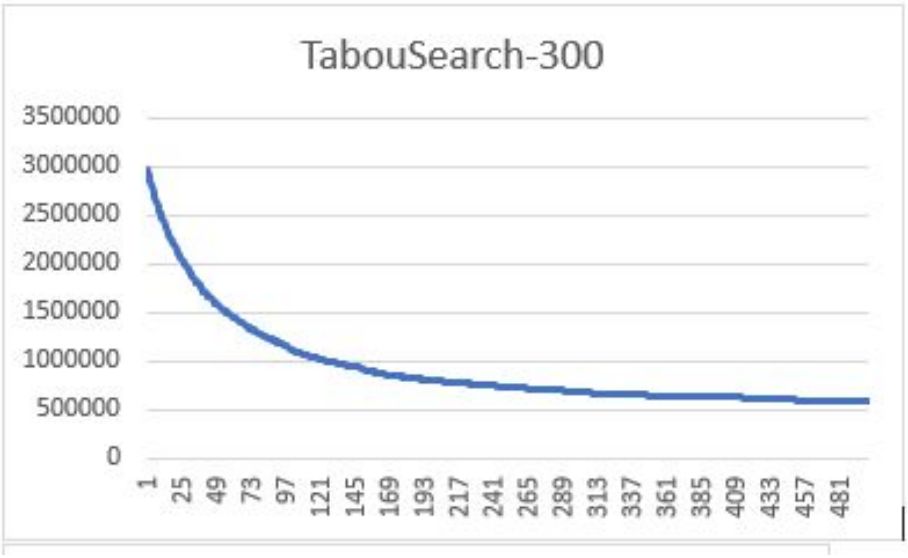
Algorithme Tabou

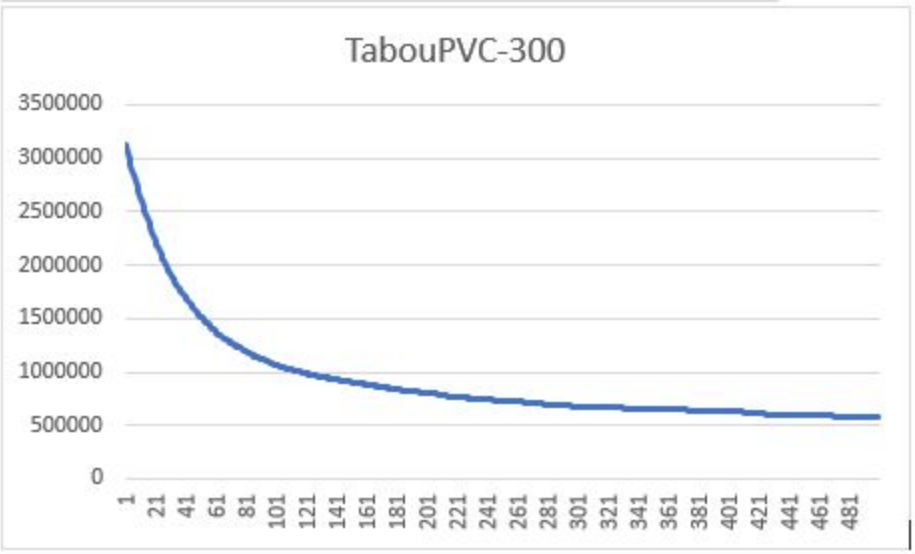
L'algorithme d'optimisation inspiré de celui de la plus forte pente, en optimisation continue, consiste à choisir à chaque itération le meilleur voisin de la solution. Toutefois, un tel algorithme, se retrouve facilement piégé dans un optimum local. Pour y remédier l'algorithme du Tabou tient à jour une liste des dernières solutions visitée, on évite ainsi de revisiter les nœuds étudiés récemment.

Implémentation:

TabouSearch et Tabou PVC

A chaque itération, on parcourt l'ensemble des voisins (permutation), se place sur meilleurs voisin parmis ceux qui ne sont pas dans la liste des mouvement tabou, sauf si c'est la meilleure solution que l'on a visité (aspiration). Après un certain nombre d'itérations l'algorithme s'arrête.





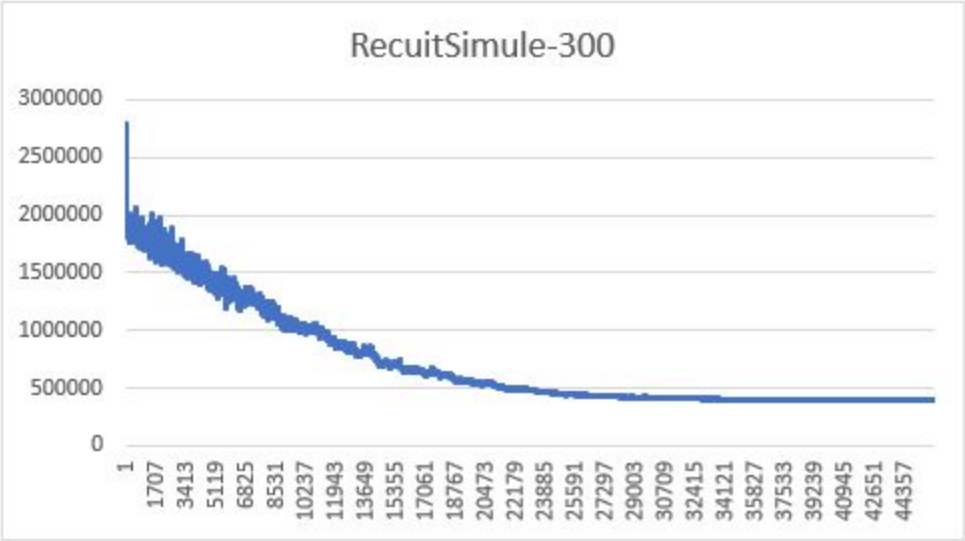
Algorithme du Recuit Simulé

L'algorithme du Recuit Simulé (RS) est un algorithme d'optimisation inspiré du recuit des métaux. La fonction que l'on cherche à minimiser est appelée Energie. Celle-ci évolue de façon similaire à l'énergie d'un métal en refroidissement.

Implémentation:

RecuitSimulé

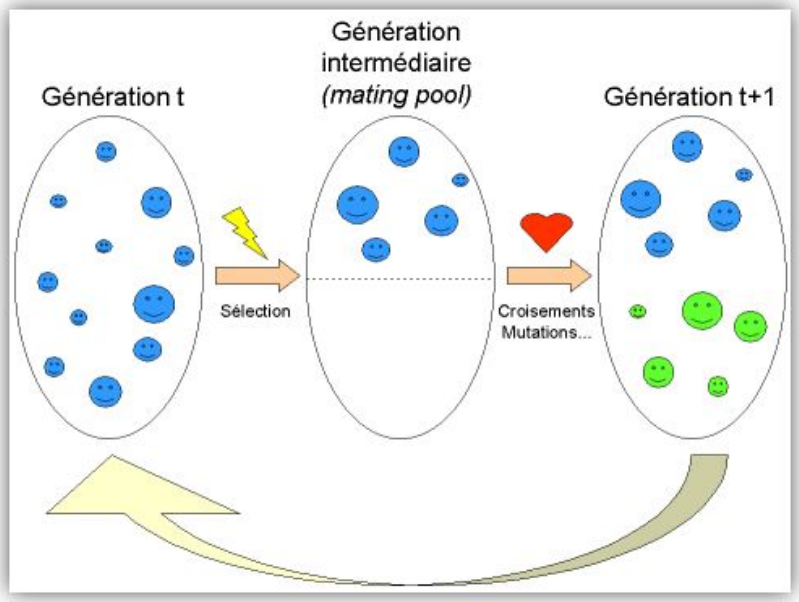
A chaque itération on considère un chemin voisin du chemin courant (par une permutation), on calcule alors l'énergie associée. On pose dE = Ec - Ev
(Ec : énergie du chemin courant, Ev: énergie du chemin voisin)
Si dE<0, on choisi le nouveau chemin.
Si dE>0, on choisi le nouveau chemin avec une probabilité de exp(-dE/k*T).
Au bout d'un certain nombre d'itérations, on baisse la température. Quand la température atteint la température finale on arrête l'algorithme.



On remarque qu'au début de l'algorithme, on choisit plus souvent d'augmenter l'énergie du système, car la température est plus élevée. Quand la température diminue, l'énergie ne fait que diminuer.

Algorithmes Génétiques :

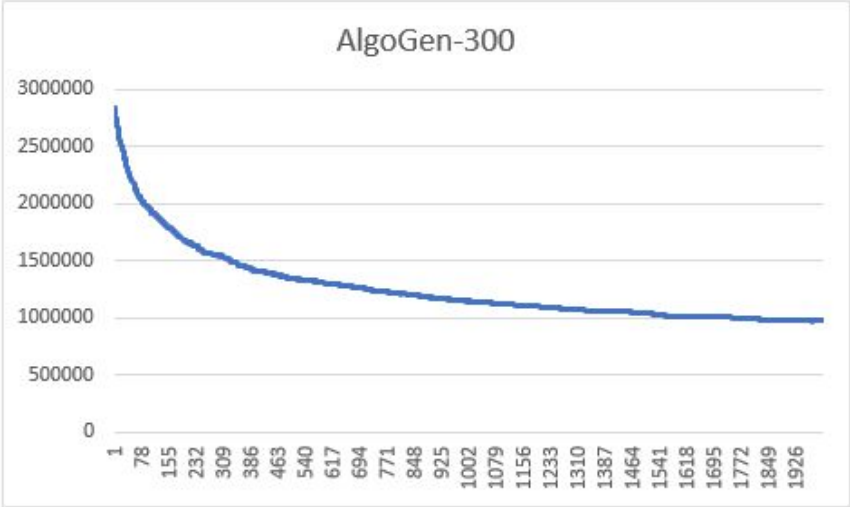
Les algorithmes génétiques sont des algorithmes d'optimisation inspirés de la théorie de l'Evolution : une population de solution évolue par sélection, croisement et mutation de façon à optimiser une fonction fitness choisie.



On a deux implémentations d'algorithmes génétique:

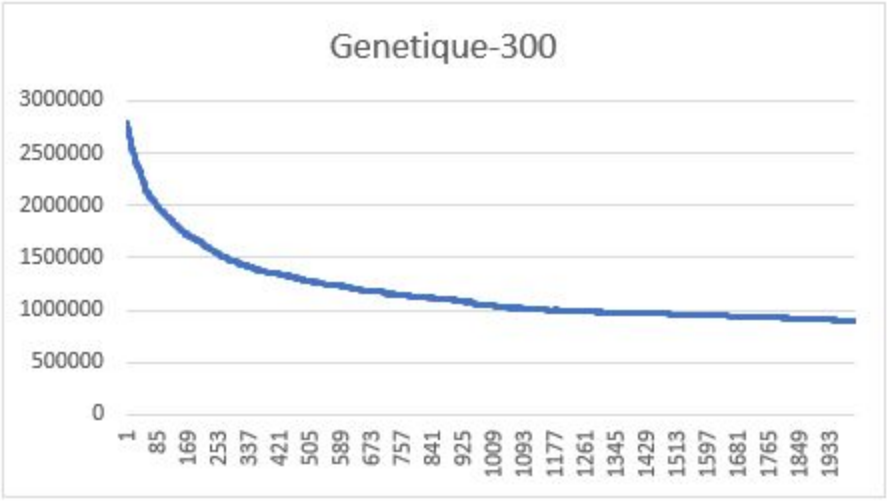
-AlgoGen:

C'est un algorithme génétique qui utilise des tableau d'entier pour représenter les chemins. L'opérateur de sélection consiste à choisir les n meilleurs individus de la population grâce à un algorithme de trie à insertion. Les mutations correspondent à des permutations. Le croisement est un double PMX avec une correction sur la chaîne centrale.



- Genetique :

C'est un algorithme génétique qui utilise des tableaux d'entiers pour représenter les chemins. L'opérateur de sélection est un opérateur déterministe qui choisit les N meilleurs individus en classant selon la valeur de fitness (ici la longueur du chemin). Ici la mutation utilisée est une permutation de place de deux entiers. Le croisement est un croisement CX (croisement cyclique).



b) Etude Comparative :

Cette étude comparative des différents algorithmes permet de comparer les performances des algorithmes entre eux et servira de base à l'évaluation des algorithmes d'optimisation distribués.

Hyperparamètres utilisés pour les simulations :

AlgoGen :
- Population : 30
- ProbMut : 0.7
- NbSelection : 8
- NbItération :2 000

Genetique :
- Population : 512
- ProbMut : 0.5
- NbSelection : 128
- NbItération :2 000

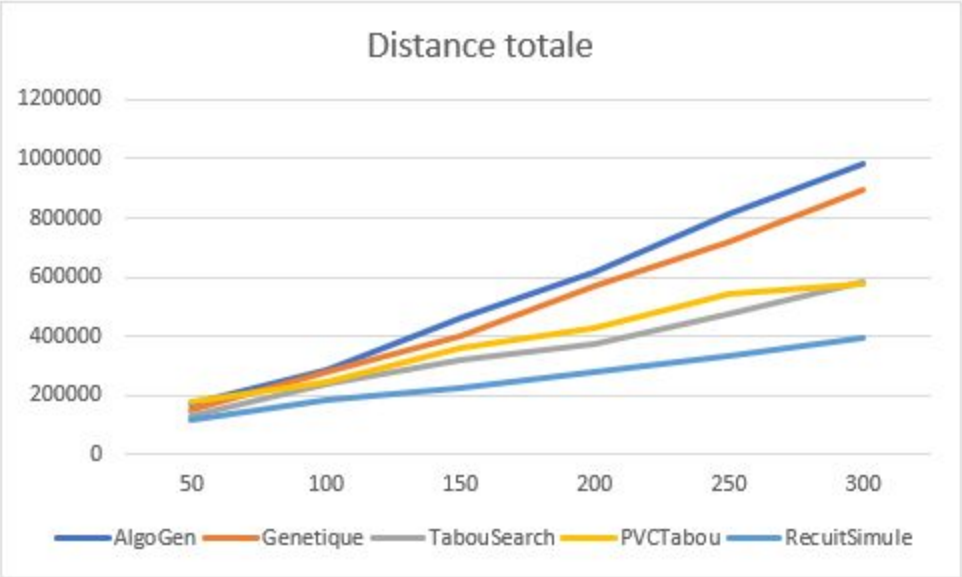
TabouSearch :
- Longueur Tabou :100
- NbItération :500

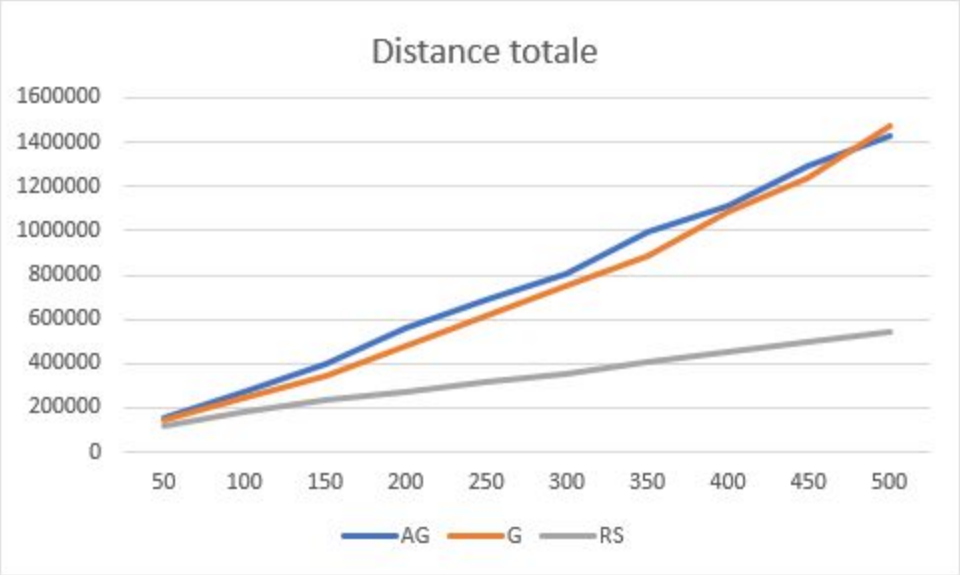
PVCTabou :
- Longueur Tabou :100
- NbItération :500

Recuit Simulé :
- Facteur de refroidissement : 0.999
- Temperature initiale : 1000
- Temperature Finale :10
- Iterations avant équilibre :70

Solutions :

On observe la meilleure solution obtenue à la fin de l'exécution de l'algorithme en fonction de la taille du problème, c'est-à-dire du nombre de villes





On remarque que la taille du chemin trouvé est linéaire en fonction du nombre de villes, ce qui est logique si on considère que la distance moyenne entre deux villes est constante si on augmente le nombre de villes.

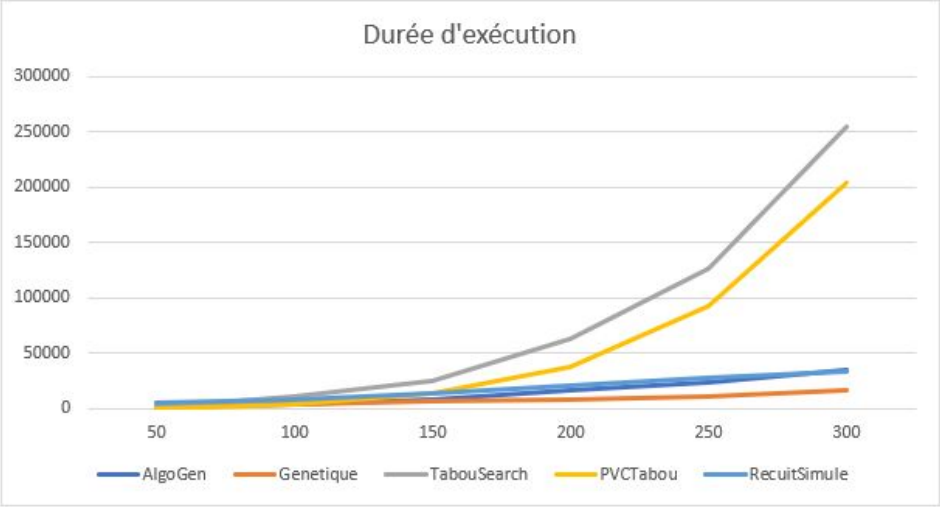
On remarque que les AG (respectivement Tabou) ont des performances similaires en termes de solutions trouvées, malgré des hyperparamètres très différents pour les AG.

Les algorithmes génétiques ont des performances moindres que les algorithmes Tabou. Ceci est contrebalancé par la vitesse lente des algorithmes Tabou, qui empêche leur utilisation sur des problème de plus grande taille.

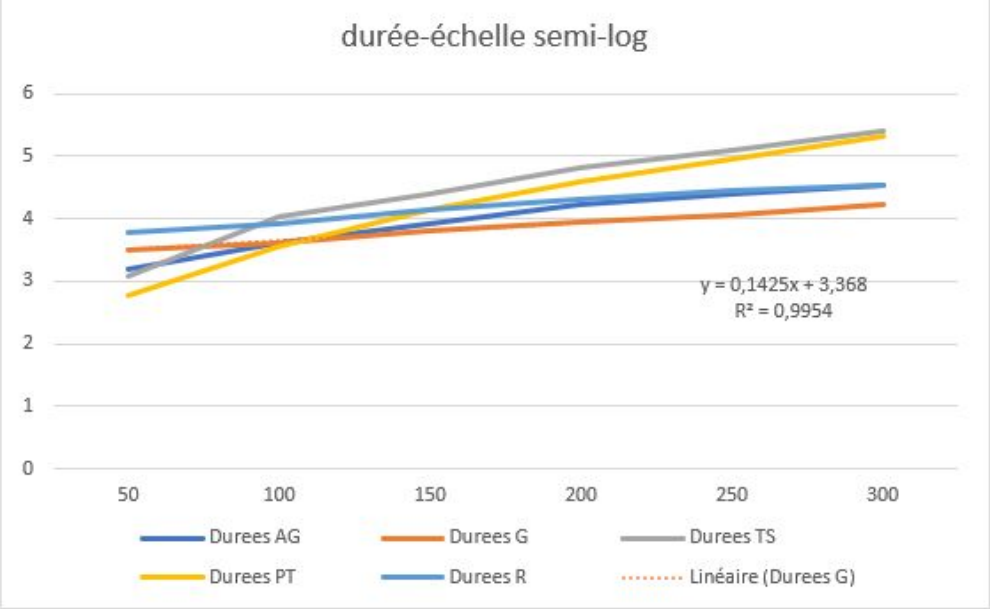
L'algorithme du recuit a les meilleures performances. On peut l'expliquer par le nombre d'itérations de l'algorithme : de l'ordre de 106 contre 2000 itérations pour le génétique et 500 pour le Tabou. Ceci est permis par la vitesse d'exécution d'une itération dans le Recuit Simulé.

Durées :

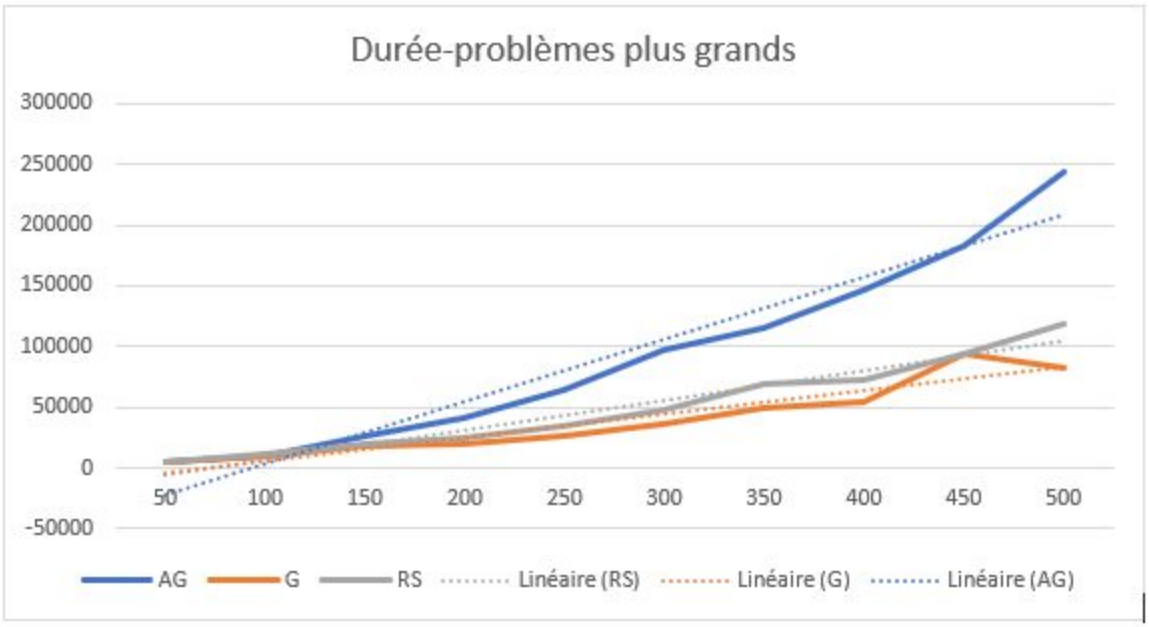
On observe le temps d'exécution des algorithmes pour différentes taille de problèmes.



La complexité des algorithmes de Tabou semble être exponentielle, ce qui est conforté par le graphique en échelle semi-log.



Les Algorithmes Génétiques et Recuit Simulé ont une complexité plutôt linéaire (même si AlgoGen est plus ambiguë).



La différence de complexité s'explique par le fait que le Tabou explore tous ses voisins, ainsi quand le nombre de villes augmente, on augmente le nombre de chemin voisins.

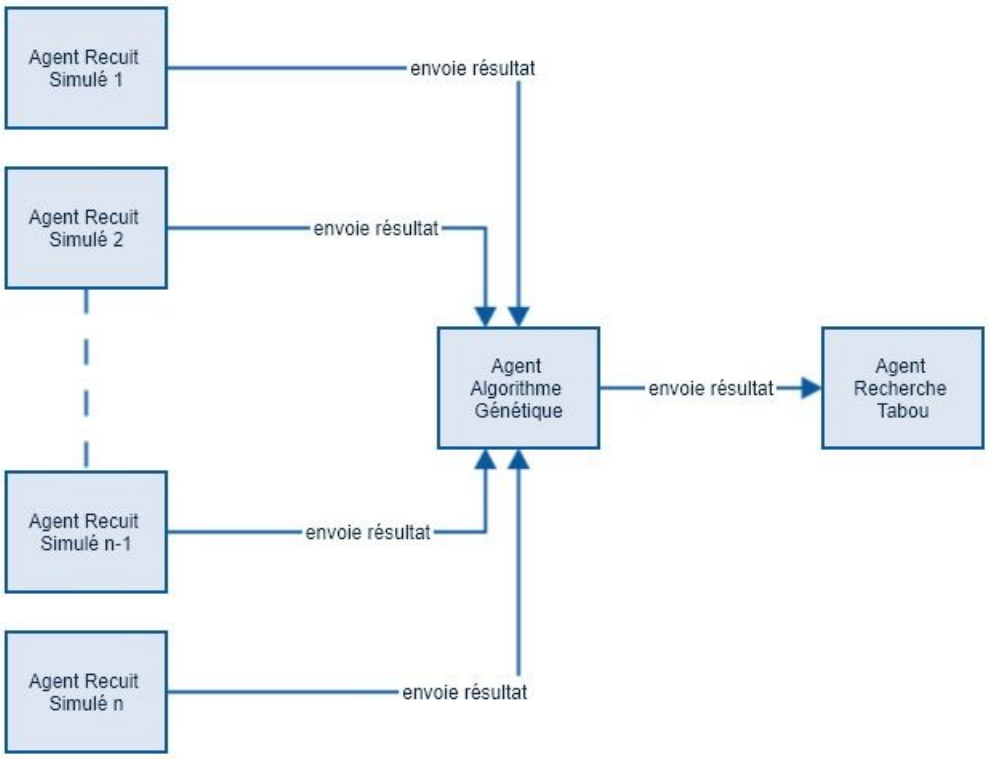
Au contraire les AG et RS visitent un nombre de voisin constant par itérations.

Bien qu'étant tout les deux des AG, on remarque que la vitesse d'exécution de l'algorithme Génétique est deux fois plus rapide que celle de AlgoGen.

B) Les protocoles SMA:

I- Protocole RGT :

Ce protocole d'interaction est un protocole de type coopération entre les différents agents d'optimisation. Ici nous avons un agent chef qui déclenche l'algorithme, en envoyant un message à N agents d'optimisation de type Recuit Simulé. Une fois qu'ils ont terminé leurs optimisation ils envoient leur résultats à l'agent d'optimisation de type Algorithme Génétique qui utilise les résultats pour remplir la population au départ de son optimisation. Une fois qu'il a fini son optimisation, l'agent de type Algorithme Génétique envoie son résultat à l'agent d'optimisation de type Recherche Tabou qui à l'issue de son optimisation donne le résultat final du protocole.



Quelques tests:

resultatRGT1(RS, Genetique, TabouSearch)
resultatRGT2(RS, AlgoGen, TabouSearch)
resultatRGT3(RS,Genetique, TabouPVC)
resultatRGT4(RS,AlgoGen, TabouPVC)

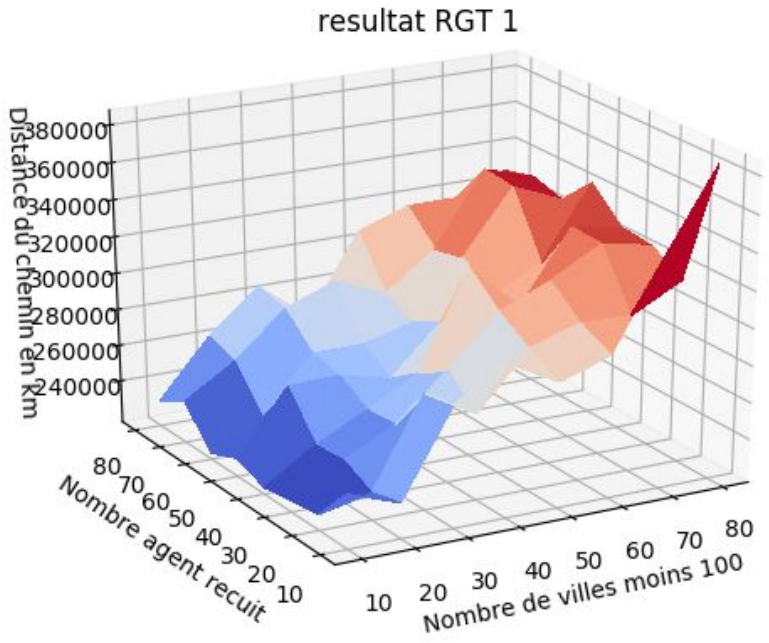
Nombre d'itérations algorithme génétique: 1024

Cas RGT1 et RGT2

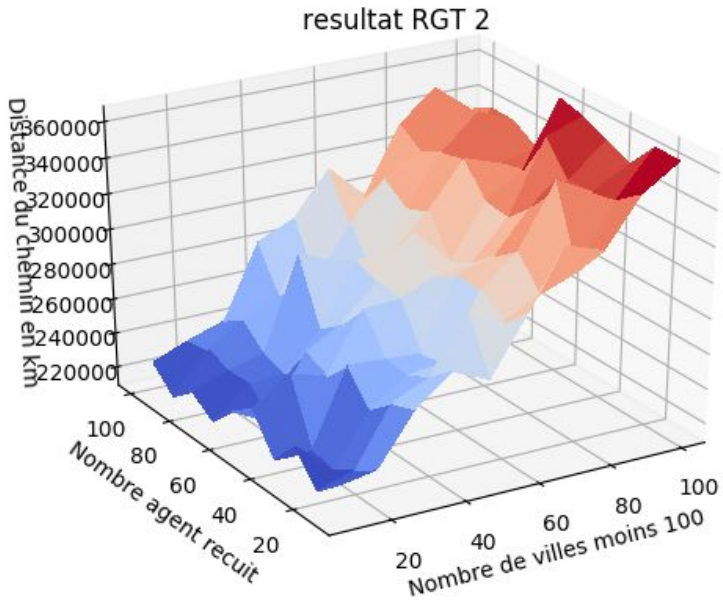
L'influence la plus prédominante est celle du nombre de villes, comme attendu. Les résultats sont évidemment plus faible suivant la taille du problème.

Cependant l'influence du nombre d'agents recuit est assez contre-intuitif. En effet le "creu de la vague" se situe, dans les deux cas, aux alentours de 60-70 agents , valeurs pour lesquels on trouve les meilleurs résultats. En effet diminuer trop fortement le nombre d'agents

recuit ne permet pas un parcours assez profond du problème; la population fournis à l'AG, trop limité en taille, se retrouve coincé dans un optimum locale qui peut s'avérer être mauvais. A l'inverse, augmenter trop fortement le nombre d'agents recuits augmente la taille de la population fournis à l'AG et ne lui permet pas un parcours tenant suffisamment compte des différents "génotypes" qui lui sont fournis, 1000 itérations ne permettant pas un brassage suffisant des allèles dans ce cas.

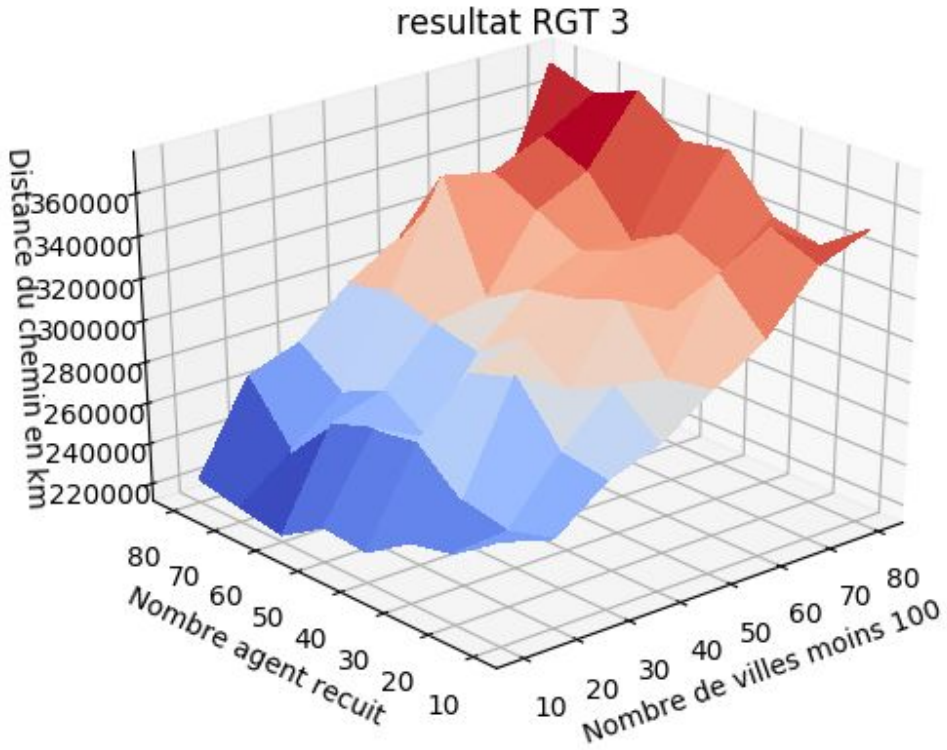


On remarque dans le cas RGT2 la présence de plus nombreux pics dans ce cas si. AlgoGen semble plus instable que le premier algorithme génétique.



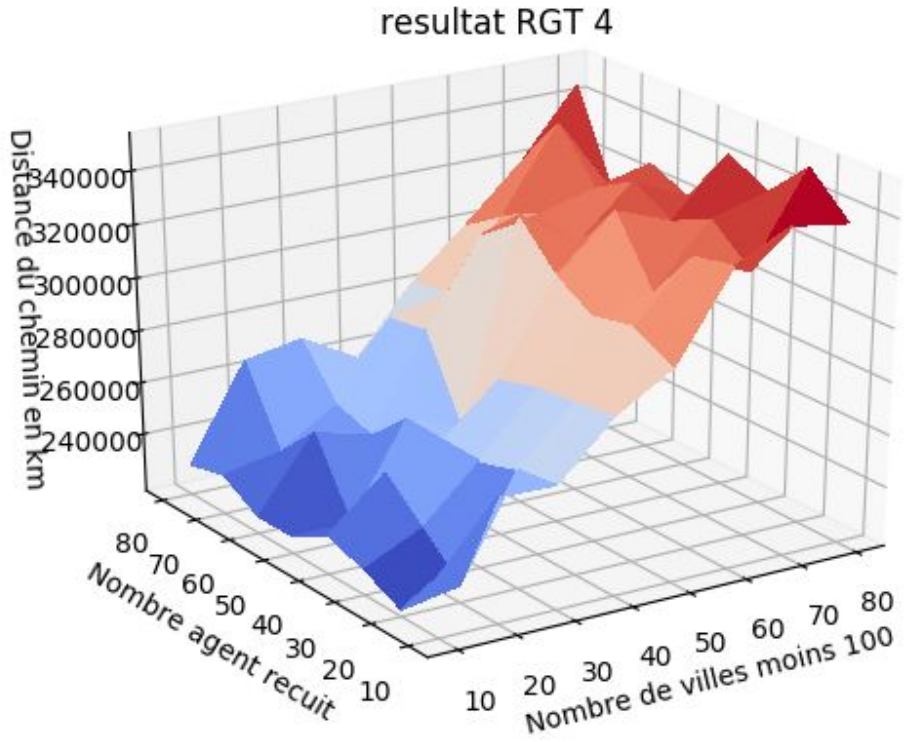
Cas RGT3

A la différence du précédent, le changement d'algorithme génétique et d'agent tabou introduit une forte dépendance sur le nombre d'agent recuit. L'augmentation de ceux-ci semble négatif. Cette combinaison correspond au deux algorithmes génétique/tabou les plus performants, un grand nombre d'agent recuit ne fait que complexifier la taille du problème et introduire des cas résiduelles.



Cas RGT4

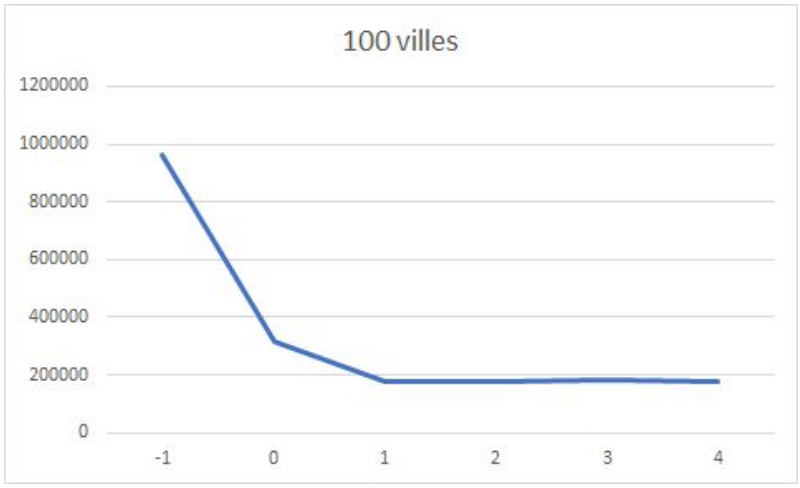
L'algorithme AlgoGen semble fonctionner de manière optimale aux alentours d'une population de 50, l'efficacité du TabouPVC rend primordiale la taille de la population assignée à l'algorithme génétique: ici 50.



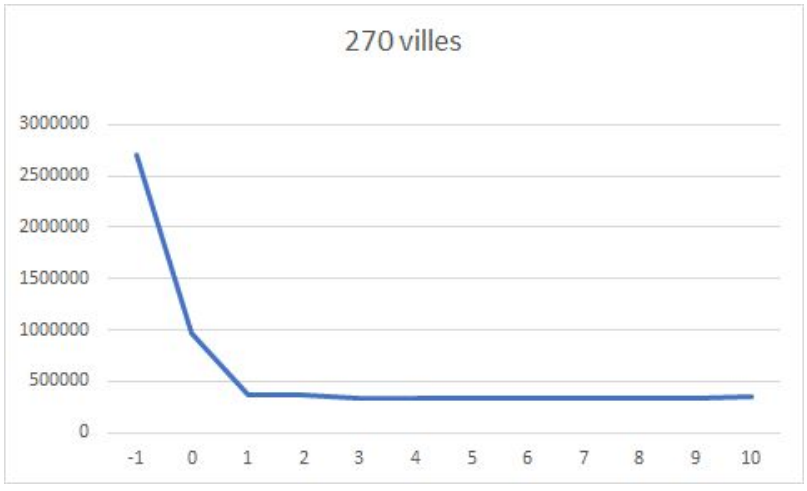
II- Protocole linéaire:

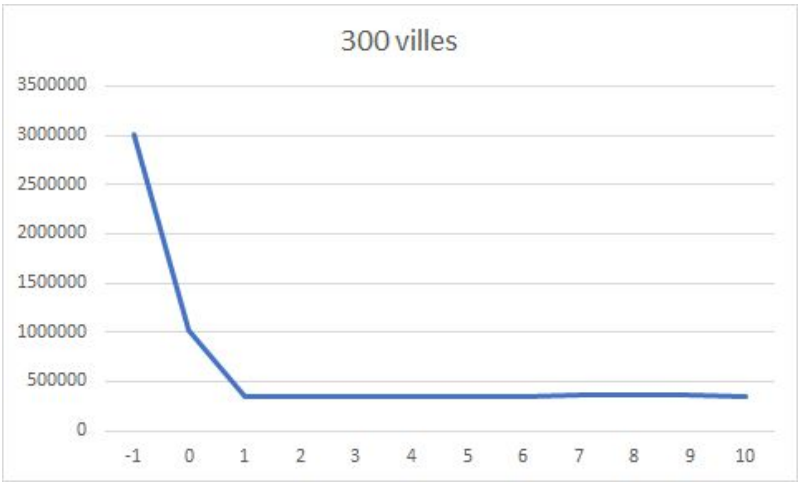
Après une première passe, l'agent génétique fournit une solution à l'agent recuit,
Qui lui même envoie à l'agent tabou sa solution ,et les agents recuits et tabous ainsi de suite s'échangent leurs solutions .

L'abscisse -1 correspond au chemin initial à partir duquel l'agent génétique va débuter.
L'abscisse 0 correspond au chemin proposé par le premier agent génétique.
Les abscisses suivants correspondent aux solutions proposés par les différents agents , au fur et à mesure du temps.

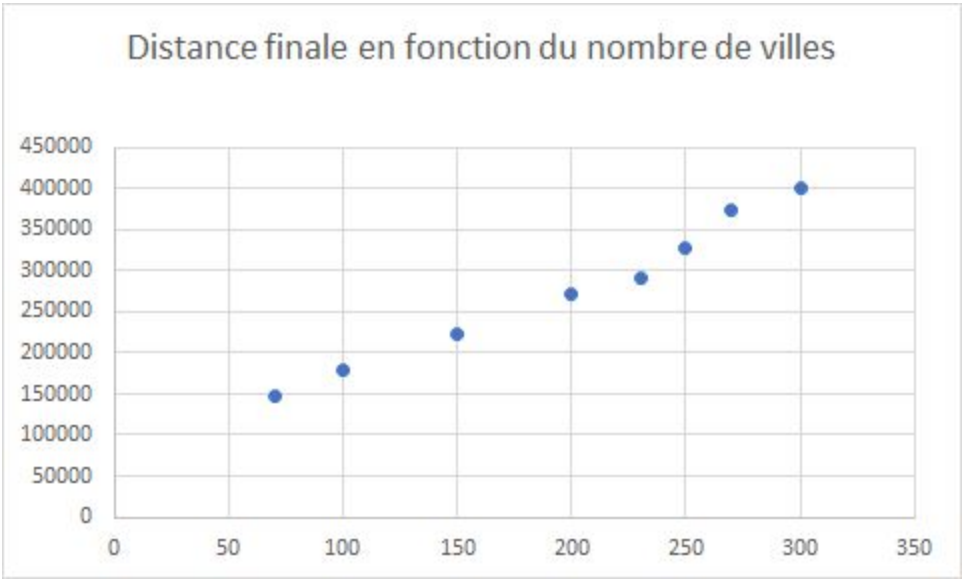

100 villes
1200000
1000000
800000
600000
400000
200000
0
-1
0
1
2
3
4


250 villes
3000000
2500000
2000000
1500000
1000000
500000
0
-1
0
1
2
3
4
5
6


270 villes
3000000
2500000
2000000
1500000
1000000
500000
0
-1
0
1
2
3
4
5
6
7
8
9
10



Les courbes montrent que l'agent génétique, après le première passe, fournit une solution proche de la meilleure solution trouvée. La convergence est très rapide , la solution proposée par le premier agent tabou est optimale.



La distance augmente linéairement avec le nombre de villes, ce qui est cohérent car la distance moyenne est linéaire .

Hyperparamètres utilisés pour les simulations :

AlgoGen :

- Population : 30
- ProbMut : 0.7
- NbSelection : 8
- NbItération :2 000

TabouSearch :

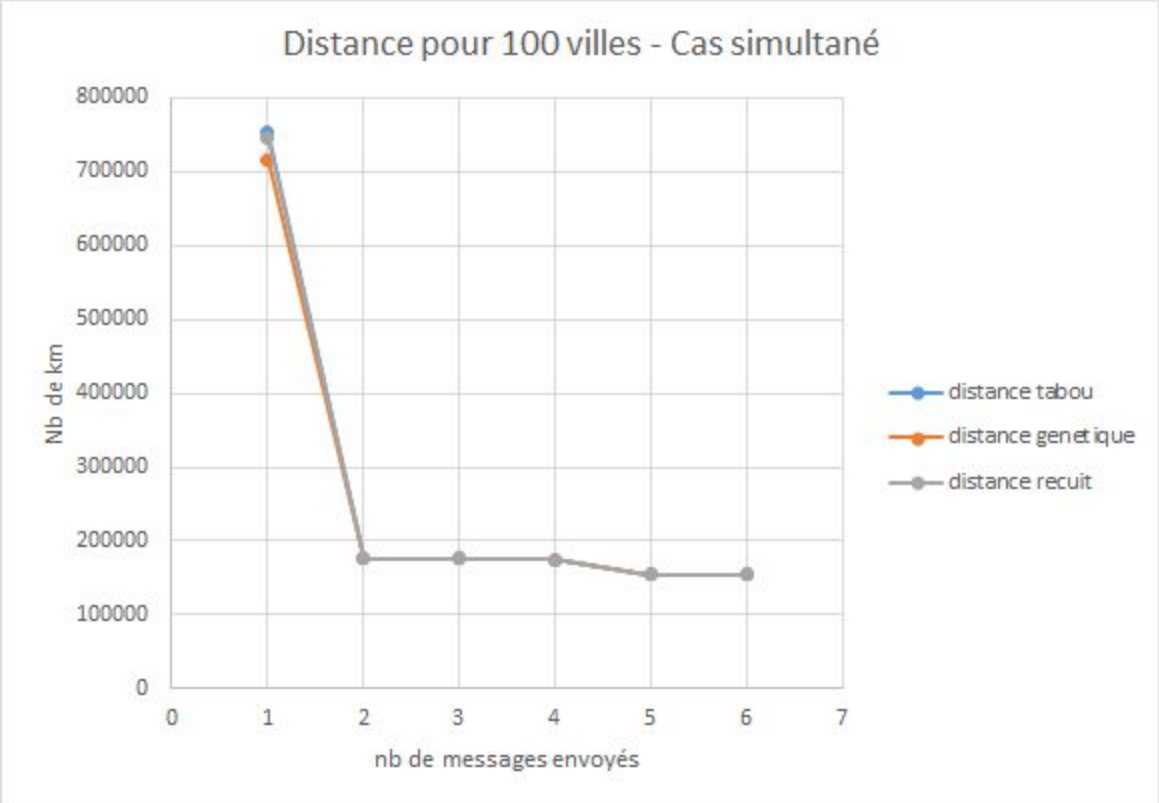
- Longueur Tabou :100
- NbItération :500

Recuit Simulé :

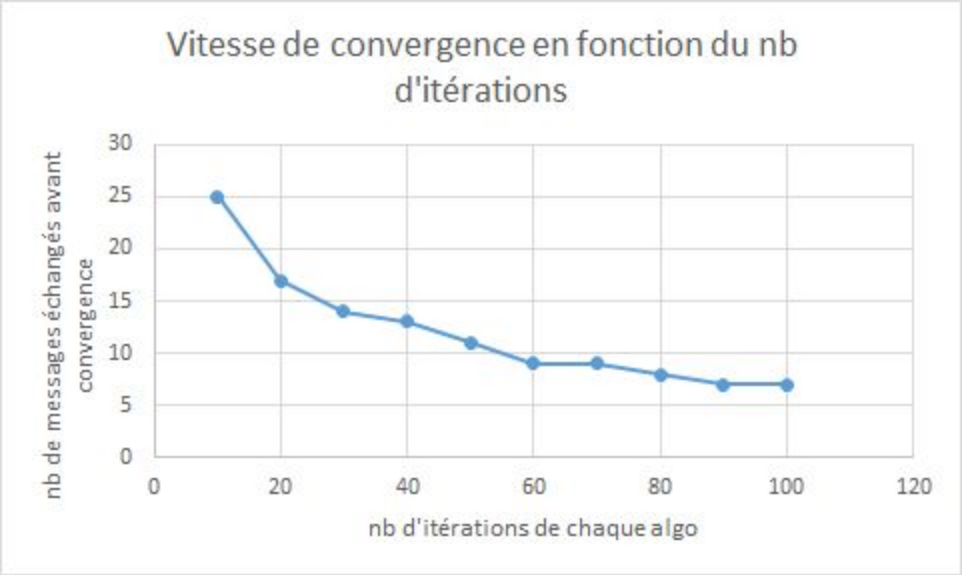
- Facteur de refroidissement : 0.999
- Temperature initiale : 1000
- Temperature Finale :10
- Iterations avant équilibre :70

III- Protocole Simultané:

Le protocole simultané consiste à réutiliser la meilleure solution trouvée par l'ensembles des agents lors du dernier message envoyé. Ainsi, à chaque message envoyé, les trois agents mobilisés ne retiennent que la meilleure distance trouvée et relancent leur algorithme -Tabou, génétique, recuit- à partir de la solution initiale optimale par rapport aux derniers messages. On propose ci -dessous le graphe des distances renvoyées par chacun des trois agents. On voit que le nombre d'itérations des algorithmes tabou, génétique et recuit sont assez élevés pour permettre aux trois agents d'obtenir les mêmes résultats à partir du deuxième message.

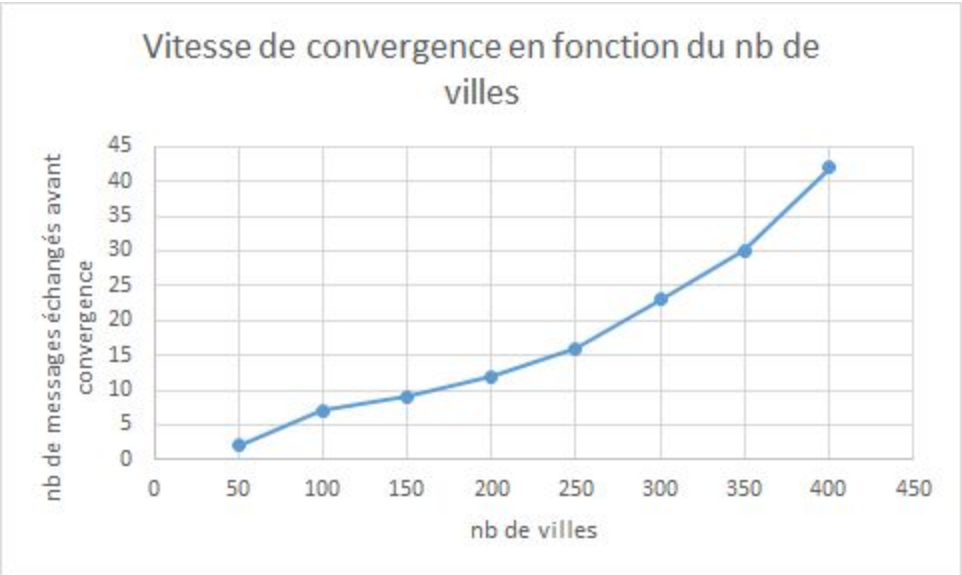


En réduisant le nombre d'itérations des trois algorithmes, on obtient des résultats différents: On retrouves les résultats liés aux performances individuelles des algorithmes (cf b) Etudes Comparative), et la convergence ne se fait qu'après un plus grand nombre de messages:



On constate donc qu'il est plus efficace dans ce cadre d'utiliser un nombre plus faible d'itération pour chaque algorithme afin d'optimiser la vitesse de calcul, car on ne rajoute qu'un ou deux messages supplémentaires en réduisant de 30 le nombre d'itérations de chaque algorithmes.

Il peut enfin être pertinent de s'intéresser à l'augmentation du nombre de messages à envoyer pour obtenir un résultat optimal lorsque le nombre de ville augmente:



On constate une dépendance à tendance exponentielle, ce qui justifie la difficulté pour ce protocole de résoudre des problèmes extrêmement volumineux, le dernier test pour le cas de 400 villes ayant par ailleurs nécessité une dizaine de minutes de calcul sur ordinateur.

La prochaine étape, qui permettrait véritablement d'optimiser ce processus, serait de déterminer la configuration optimale en terme de nombre d'itérations pour un nombre de villes fixée: il s'agirait alors de calculer le nombre de messages nécessaire à la convergence pour tous les triplets (i,j,k), i correspondant au nombre d'itérations du tabou, j à celui du génétique et k celui du recuit. Cette recherche étant extrêmement longue en terme de temps de calcul machine, elle n'est ici suggérée que si elle devait être mise en place dans un cadre commercial.