François Gueguen
Master 1 IL

Rapport de projet – Recuit Simulé

Sommaire

Introduction

L'objectif est de comprendre le fonctionnement et l'utilisation du recuit simulé. Il s'agit ici de minimiser des fonctions objectifs. La mise en oeuvre de l'algorithme et l'effet des différents paramètres seront étudiés.

Concept du recuit simulé

Pour des problèmes NP complets d'optimisation (comme le problème du voyageur de commerce), on ne connait pas d'algorithme polynomial permettant une résolution de façon optimale. On va donc chercher une solution approchée de cette optimum en utilisant des heuristiques. Le recuit simulé et un algorithme basé sur une heuristique permetant la recherche de solutions à un problème donnée. Il permet notament d'éviter les minima locaux mais nécessite un réglage minutieux des paramètres.

Fonctions mathématiques...

La recherche de solutions optimales a été effectuée sur 2 fonctions mathématiques :

Fonction sans minimum local

La fonction étudiée est :
$f(x)=x^2$

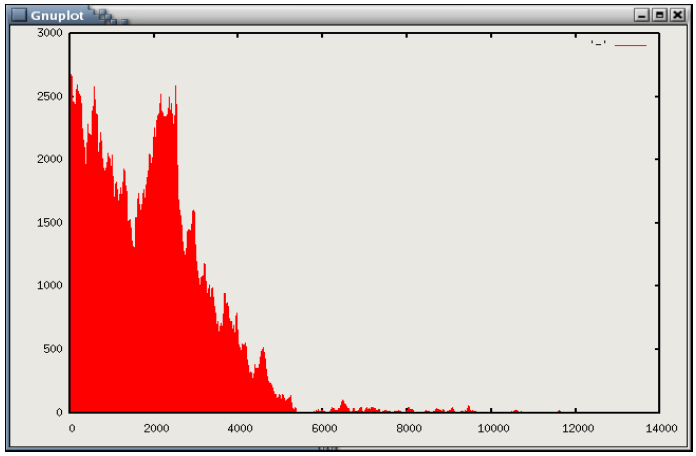
Pour une telle fonction, le minimum optimal est facilement trouvé ou approché car il est impossible de rester bloqué à un minimum local.

Avec les arguments suivants :

```
Etat initial = 50
TInit = 1000
TFin = 1
Alpha = 0.99
Ampli = 1
NMaxEAcc = 20
NMaxRep = 20000
```

Le résultat optimal est trouvé quasiment à tous les essais.

Voici un graphe correspondant à ces arguments :



On s'apperçoit que le résultat minimal est atteint. On peut aussi le constater dans le fichier de coûts généré.

Fonction avec des minima locaux

La fonction étudiée est :

$f(x)=x^2$ tant que $x <= 3$
$f(x)=(x - 10)^2 * 5/49 + 4$ tant que $x > 3$

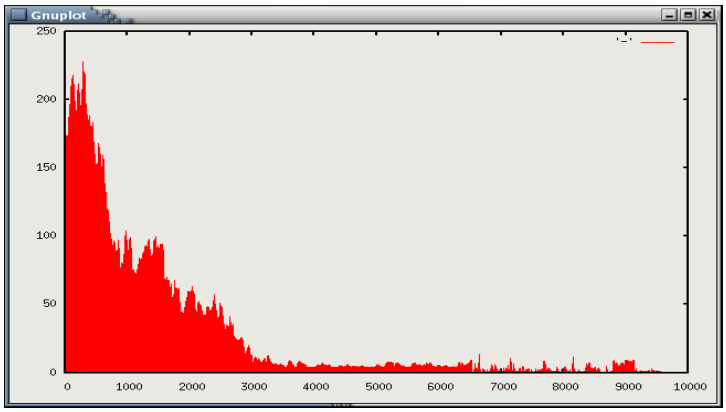
Pour une telle fonction les arguments ont plus d'importance que pour la précédente car il existe un minimum local où l'algorithme peut se bloquer.

Avec les arguments suivants (les mêmes que pour la fonction précédente) :

```
Etat initial = 50
TInit = 1000
TFin = 1
Alpha = 0.99
Ampli = 1
NMaxEAcc = 20
NMaxRep = 20000
```

Le résultat optimal est souvent trouvé. Il arrive cependant que la fonction se bloque à un minimum local. Étant donné que l'algorithme dépend d'une condition aléatoire, il est impossible de garantir de ne pas être bloqué à un minimum local (à moins d'une chute de température infiniment lente) et donc on peut considérer que ce résultat est satisfaisant.

Voici un graphe correspondant à ces arguments :



On s'apperçoit que même si au final on a trouvé le résultat optimal (qui est de 0), l'algorithme a été bloqué dans un minimum local pendant un bon moment. Il s'en est sorti vers l'itération 6500, y est retourné autour de 9000 puis s'en est ressorti au final.

Si l'on veut rester bloqué au minimum local, on peut utiliser des arguments tels que :

```
Etat initial = 200
TInit = 1
TFin = 0.001
Alpha = 0.95
Ampli = 1
NMaxEAcc = 20
NMaxRep = 20000
```

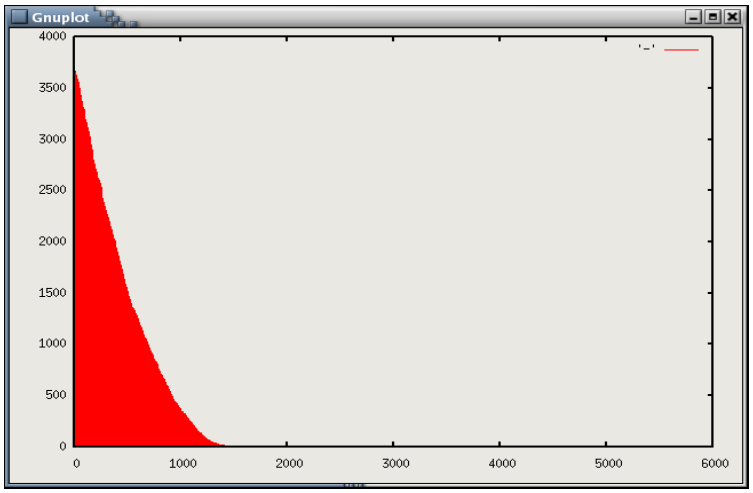
La différence vient des températures : étant donné que la température est toujours inférieure ou égale à 1, la remontée vers une solution un peu moins bonne ne sera quasiment jamais autorisée et donc il sera presque impossible de se sortir de minima locaux.

```
if (myRandom01() <= exp((fx-fy)/T)) {...}
```

On voit clairement qu'étant donné qu'*fy* est, lorsque le programme atteint cette condition, nécessairement supérieur ou égal à *fx*. On a donc une valeur négative de plus en plus importante

dans l'exponentielle, qui tend donc vers 0. En clair, plus la température baisse, moins il est probable qu'une valeur générée aléatoirement entre 0 et 1 soit inférieure à cette exponentielle, et donc assez rapidement il est improbable qu'une remontée intervienne.

On obtient un graphe du genre :



Avec une valeur finale de 4, soit le minimum local recherché.

Voyageur de commerce

Un voyageur de commerce doit visiter une seule fois un ensemble de villes et doit revenir chez lui en minimisant la distance parcourue.

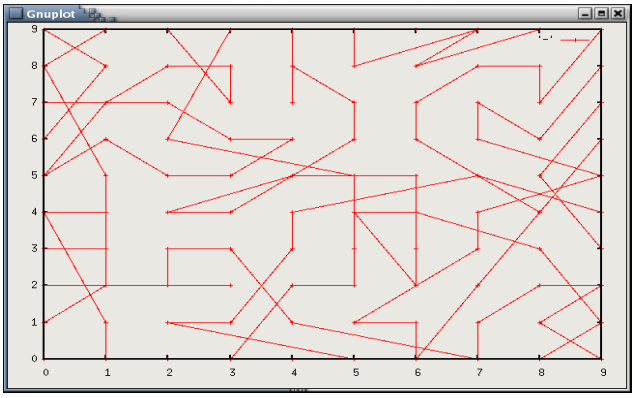
Lorsque le nombre de villes et la taille de la carte ne sera pas précisée, 100 villes sur une carte 10*10 seront utilisés.

Influence des arguments

On s'aperçoit qu'avec les arguments utilisés précédemment avec les fonctions, on obtient de mauvais résultats :

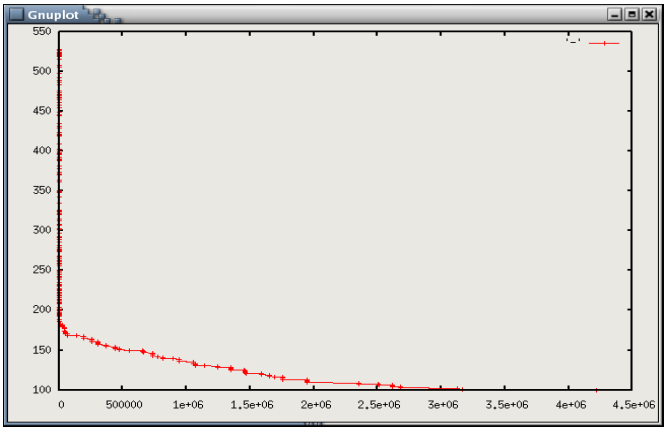
```
TInit = 1000
TFin = 1
Alpha = 0.99
Ampli = 1
NMaxEAcc = 20
NMaxRep = 20000
```

donnent un tracé tel que :



qui n'est clairement pas optimal, si même très proche d'une solution optimale.

La courbe de l'évolution des coûts est :



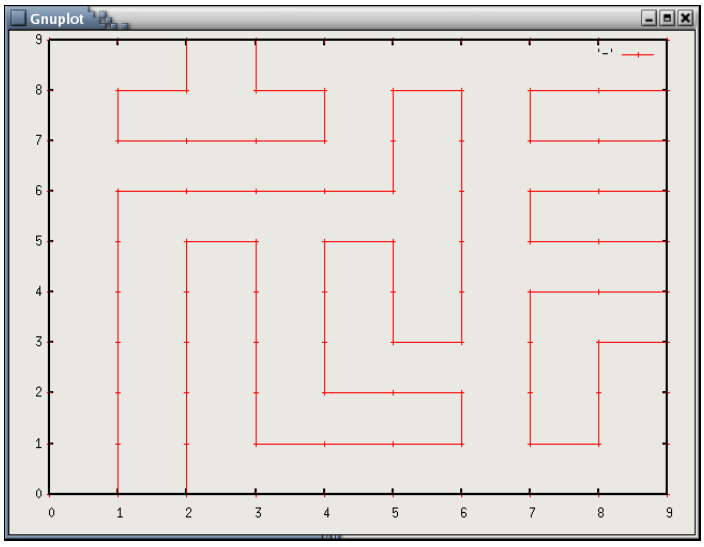
On peut supposer que la mauvaise qualité du résultat est due en grande partie à la température initiale (et à la finale) utilisée, comme vu avec la recherche d'un minimum local avec la deuxième fonction mathématique utilisée.

La faible baisse des coûts au début de l'algorithme vient aussi de la température élevée qui accepte très régulièrement les remontées vers des solutions moins bonnes. On peut donc baisser la température initiale car il est tout de même très improbable de se bloquer dans une solution locale au tout début d'un tel calcul.

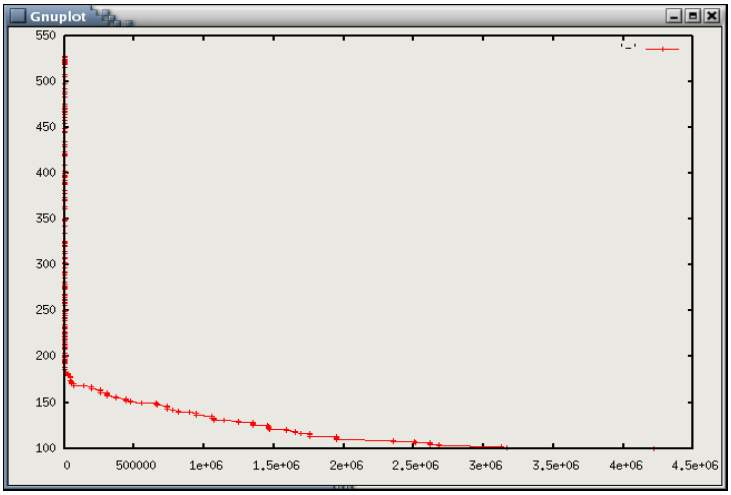
En ne modifiant que la température de la manière suivante :

```
TInit = 1
TFin = 0,001
Alpha = 0.99
Ampli = 1
NMaxEAcc = 20
NMaxRep = 20000
```

donnent un tracé tel que :



Ce qui est un résultat (supposé) optimal, avec une évolution des coûts de la forme :



On se rend compte que le choix des températures est extrèmement important.

Concernant les autres paramètres :

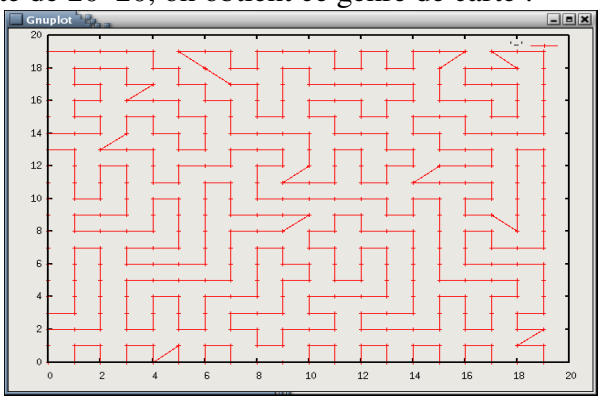
- NMaxEAcc ne sert à rien car pas utilisé dans le code
- Alpha sert à la rapidité de descente de la température, et donc n'influence pas les calculs en eux même mais leur nombre (et donc la précision souhaitée)
- NMaxRep est le nombre de calculs par seuil de température. Si les températures sont bien choisies, on constate que son influence est moindre.
- Ampli est l'importance de la modification à effectuer à chaque calcul. Le problème du voyageur de commerce étant très complexe, trop de modifications pourraient empécher d'atteindre un bon résultat au final.

Qualité des résultats

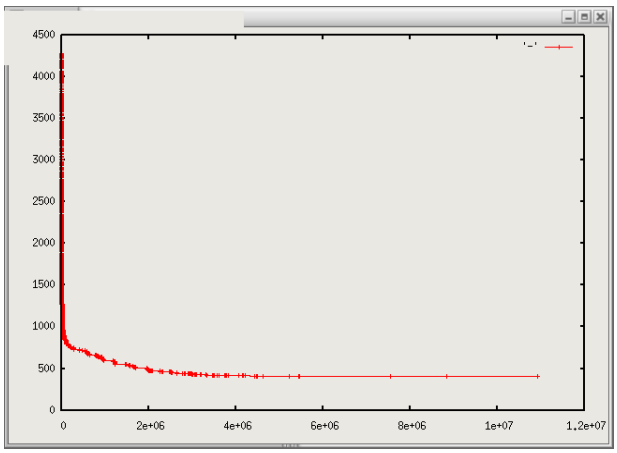
On se rend compte qu'avec les arguments :

```
TInit = 1
TFin = 0,001
Alpha = 0.99
Ampli = 1
NMaxEAcc = 20
NMaxRep = 20000
```

on obtient de très bons résultats sur la carte de 100 sur 10*10.
Si les villes étaient réparties de manière irrégulière sur la carte, il aurait fallu augmenter la température initiale car des minima locaux seraient apparus.
Enfin, on peut tester ces mêmes arguments sur une carte plus grande pour en voir la performance : avec 400 villes sur une carte de 20*20, on obtient ce genre de carte :



avec ce genre de courbes d'évolution des coûts :



Ce qui est très acceptable...

Conclusion

En conclusion, on peut dire que le recuit simulé est une bonne solution pour trouver des solutions acceptables à certains problèmes NP complets, en particulier le problème du voyageur de commerce.