MASTER 1 IA

LES MÉTAHEURISTIQUES EN INTELLIGENCE ARTIFICIELLE

—

Par Alexis AREKION

INTRODUCTION

Nous traiterons ici de deux méthodes de recherche de solution en particulier. A savoir le Hill Climber et le Recuit Simulé. Pour ce faire nous allons les implémenter sur le problème du sac à dos multidimensionnel. Nous testerons également plusieurs méthodes de pénalisation sur ces algorithmes. Et enfin nous analyserons les résultats de ces différentes expériences.

Le Problème du sac a dos multi-dimensionnel

Variante du célèbre problème du sac à dos, le problème du sac à dos multi-dimensionnel est relativement simple à comprendre :

-On a un sac à dos à n dimensions avec une contrainte de taille sur chaque dimension

-On a un nombre x d'objets avec également n dimensions et un score

On veut le sac à dos avec le score maximal (le score du sac étant la somme des scores des objets qu'il contient).

Toute la difficulté du problème repose sur la multi-dimensionnalité des objets et du sac puisque l'on doit respecter une taille limite sur chaque dimension.

1. Présentation de notre sac à dos

Dans notre cas nous utiliserons la base nommée B50 comme base de notre sac à dos.

De ce fait nous aurons :

- 5 dimensions pour nos objets ainsi que notre sac (avec 5 contraintes de taille donc)
- 15 objets à disposition pour remplir notre sac soit 15! compositions de sac possible (ce nombre étant bien trop grand on ne peut bien sur pas parcourir l'ensemble des solutions possibles)

2. Observations sur notre sac à dos

En analysant les données de notre problème on remarque que les contraintes sont plus grandes que la somme des tailles de tous les objets ainsi même en mettant l'ensemble des objets à notre disposition dans notre sac on veille à ne jamais dépasser les contraintes du sac.

On peut donc en déduire la meilleure solution possible pour ce problème; en effet, les scores étant tous positifs il suffit donc de mettre l'ensemble des objets à notre disposition pour avoir le meilleur sac possible.

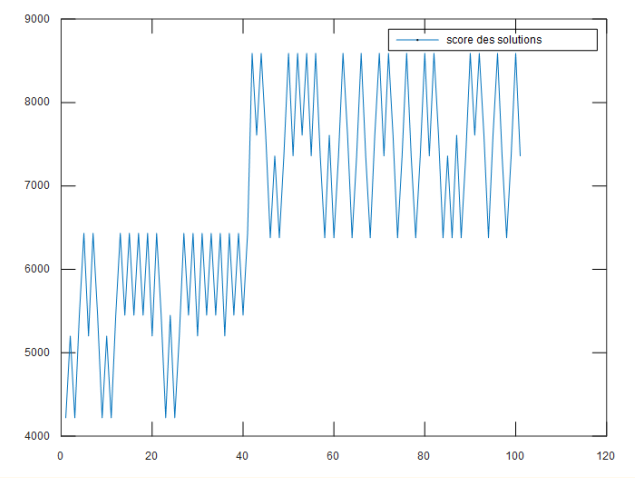
A noter qu'afin de pouvoir tester nos algorithmes efficacement nous les ferons tourner sur ce sac ainsi qu'un sac identique mais avec des contraintes de taille suffisamment basses pour qu'il ne soit pas obligatoire de mettre tous les objets dans le sac pour avoir le meilleur sac possible.

Recherche de solution vs construction de solution

Contrairement aux méthodes que nous utilisions jusque là le Hill Climber ainsi que le recuit simulé ne s'appuient pas sur la construction de solution.

En effet si traditionnellement pour trouver la meilleure solution à notre problème nous aurions rajouté petit à petit des objets dans notre sac jusqu'à avoir le meilleur sac possible. Ici on prend le contrepied de cette logique et on parcours directement l'ensemble des sacs possibles jusqu'à trouver le meilleur sac. (Le nombre de sacs possibles étant bien trop grand on ne va parcourir qu'une partie des solutions possibles à chaque fois bien entendu)

Exemple d'un parcours de 40 solutions en partant d'une solution aléatoire :



De plus dans les 2 cas on explorera les solutions de la même manière à savoir :

- Partir d'une solution aléatoire

- Observer aléatoirement un de ses voisins
- En suivant l'algorithme soit on ignorera ce voisin soit on ira sur ce voisin
- On observe de nouveau les voisins et on recommence jusqu'à atteindre la condition d'arrêt
- On observe les résultats de l'algorithme

Le Hill Climber

Premier algorithme que nous étudierons le Hill Climber bien qu'il soit un excellent algorithme de recherche de solution possède un "gros défaut" : il ne peut trouver que des solutions locales.

Cependant ce "désavantage" varie énormément en fonction des problèmes, en effet plus la plage de solutions locales de la solution globale est grande proportionnellement à l'ensemble des solutions plus le Hill Climber aura de chance de trouver la solution globale.

1. Mise en place du Hill Climber sans pénalisation

Tout d'abord on doit trouver un moyen de modéliser nos différentes solutions pour celà on va créer 3 choses :

-1 matrice nommée "s" de 15 éléments (1 par objet) ne comprenant que des 1 et des 0 (1 on prend l'objet 0 on ne le prend pas)

-1 seconde matrice nommée "contraintes" contenant la somme des poids de tous les objets qu'on a pris sur chacune des dimensions (5 poids au total donc)

-le score de la solution dans une variable note

Ensuite on devra déterminer plusieurs éléments :

-La condition d'arrêt de notre algorithme, dans notre cas une variable "aléa" qui correspond au nombre de voisins qu'on a parcouru sans changer de solution (on la règle à 40 pour être sur d'avoir exploré tous les voisins de notre solution si aucun n'est meilleur cela signifie qu'on a atteint un

optimum local et donc que notre Hill Climber ne bougera plus) A noter que l'on peut rajouter d'autres conditions d'arrêt (comme par exemple un nombre limité d'itération ou de temps de calcul).

-Un moyen de récupérer plusieurs informations :

-Le nombre d'itérations, ici un simple compteur suffit

-Les notes des différentes solutions que l'on a parcouru, on les stocke à chaque déplacement dans une matrice "repartscore"

-La meilleure solution, dans ce cas il s'agit de la dernière solution stockée dans l'algorithme puisque le Hill Climber ne se déplace que si la solution voisine est meilleure que sa solution actuelle

Ensuite pour l'algorithme le déroulement est simple :

-On choisit aléatoirement une solution

-On lance une boucle réglée sur notre condition d'arrêt dans laquelle :

-On recherche un voisin aléatoirement que l'on stockera dans une matrice "si" ("si" = "s" sauf pour un élément choisi aléatoirement qu'on a "inversé" (1 -> 0 et 0 -> 1))

-On calcule le score du voisin

-Si le score de "si" est supérieur au score de "s" alors on passe a "si" (s = si) et on stocke le score de la nouvelle solution "s" dans la matrice contenant l'ensemble des scores des solutions parcourues.

-Sinon notre solution "s" ne bouge pas et on relance la boucle si l'on a pas atteint la condition d'arrêt

-On affiche les résultats de l'algorithme

2. Evaluation du Hill Climber sans pénalisation

Comme précisé dans la présentation du problème les contraintes de taille de notre sac sont bien trop grandes et ne peuvent en l'état pas être dépassées, ainsi on peut donc aisément implémenter notre sac dans un Hill Climber sans pénalisation.

Etant donné qu'il s'agit d'un algorithme stochastique on le testera sur 1 000 lancers avec à chaque fois une nouvelle solution de départ choisie aléatoirement.

On en sortira une matrice contenant le meilleur score de chacun des 1 000 lancers qu'on utilisera pour avoir la moyenne du pourcentage de réussite / la variance / le minimum ainsi que le maximum avec un graphique contenant l'ensemble des scores et bien sur un histogramme de ceux ci.

On en sortira également une matrice contenant le nombre d'itérations de chacun des 1 000 tirages.

Afin d'avoir un élément de comparaison sur lequel se baser on va créer un algorithme de recherche aléatoire celui-ci cherchera aléatoirement la solution optimale sur le même nombre d'itérations que notre Hill Climber.

De plus les contraintes étant trop grandes on connaît à l'avance le score du meilleur sac qui est de 11 356.

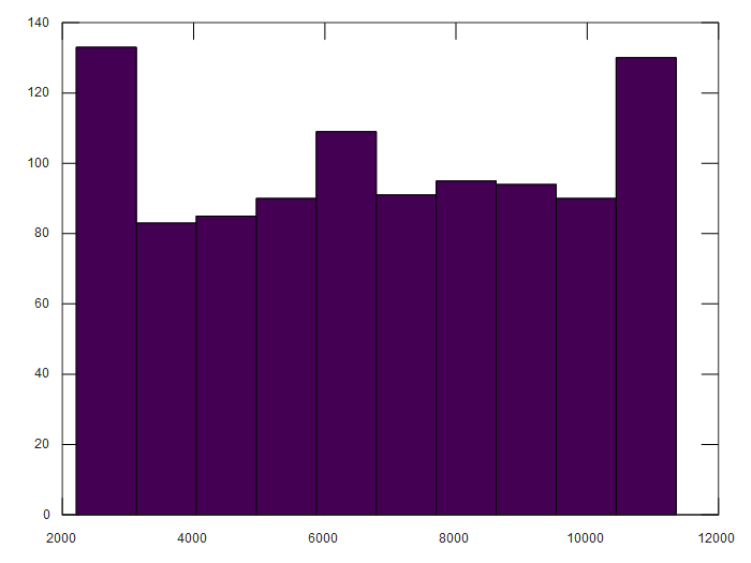
3. Résultats du Hill Climber

Sur les 1 000 lancers ont à un score qui :

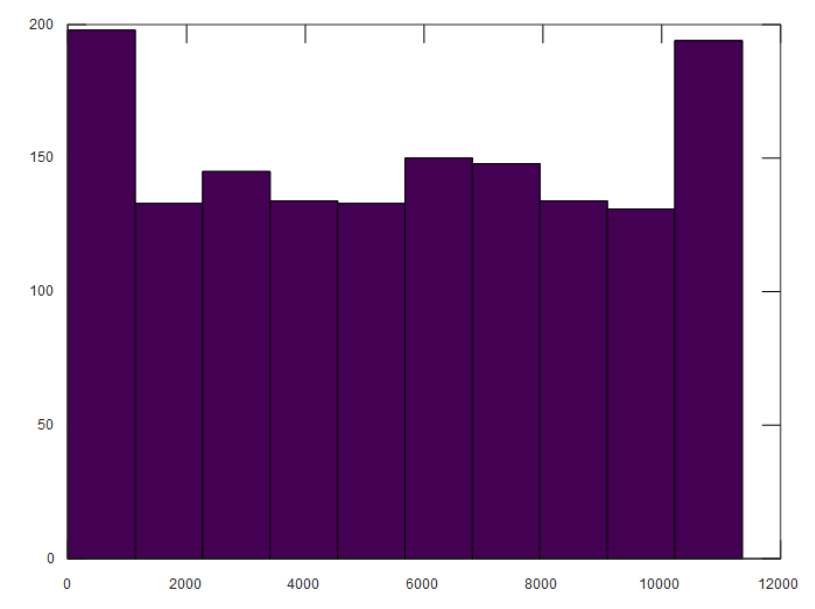
- En Moyenne est égal à 6855.6 pour le Hill Climber contre `5758.6 pour la recherche aléatoire`
- Un minimum de `2213` contre 0 pour la recherche aléatoire
- Un maximum de 11356 ce qui implique que selon le sac de départ le Hill Climber est capable de trouver la meilleure solution, ce résultat est cependant à nuancer puisque l'algorithme de recherche aléatoire y arrive également.

On remarque que les résultats du Hill Climber ne sont pas significativement meilleurs qu'un algorithme de recherche aléatoire sur beaucoup d'itérations, puisqu'on ne gagne en moyenne qu'à peine 10% sur le score. J'ai donc cherché à vérifier si avec un nombre d'itérations relativement faibles le Hill Climber avait tous des résultats significativement meilleurs que la recherche aléatoire, et la réponse est non.

Histogramme des scores du Hill Climber sur 1 000 tentatives



Histogramme des scores de la recherche aléatoire sur le même nombre d'itérations



On remarque tout de suite que la répartition des scores des 2 algorithmes sont quasiment identiques à la différence près que le Hill Climber semble avoir légèrement améliorer l'ensemble des résultats puisqu'au lieu d'être répartis entre 0 et 11 356 ils sont répartis entre 2 200 et 11 356.

Cette ressemblance peut s'expliquer en partie du fait que le Hill Climber se base sur une solution aléatoire à chaque lancer.

En conclusion en l'absence de contraintes le Hill Climber semble n'avoir que peu d'intérêt.

Les méthodes de pénalisation

Dans cette seconde partie nous soumettrons notre sac à des contraintes bien plus strictes afin que la solution soient beaucoup plus difficile à déterminer, et pouvoir tester l'efficacité du Hill Climber dans ce genre de situation.

On testera alors 4 méthodes de pénalisation, en testant également l'algorithme de recherche aléatoire en parallèle afin d'avoir une idée de la pertinence de notre Hill Climber.

1. Only Feasible

Cette méthode de pénalisation est certainement la plus simple, ici on ne prend en compte que les solutions possibles, ainsi si une solution dépasse les contraintes alors son score sera automatiquement égale à 0.

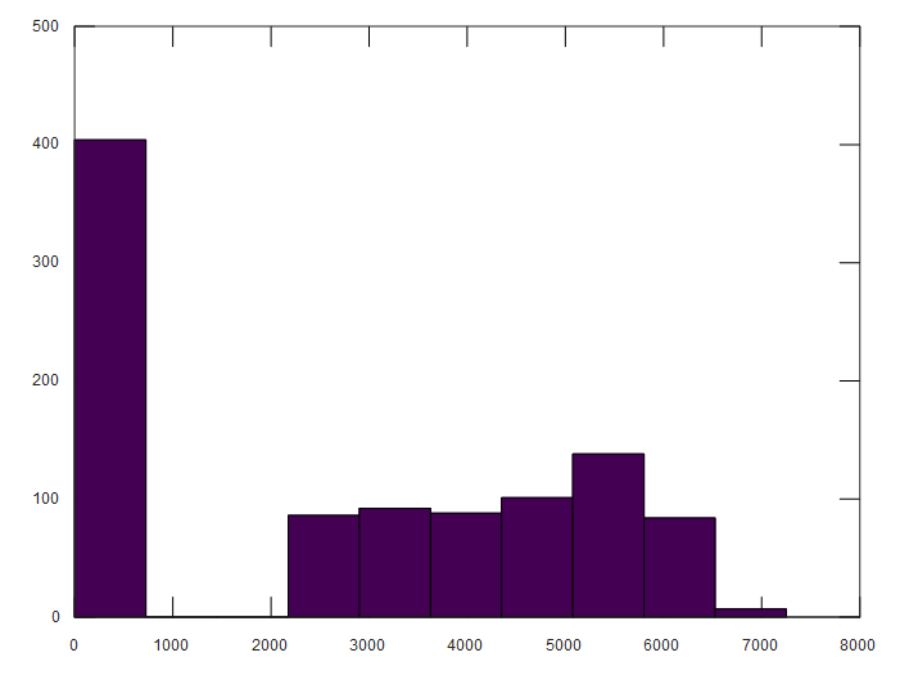
1. Résultats du Hill Climber avec Only Feasible

Sur les 1 000 lancers ont à un score qui :

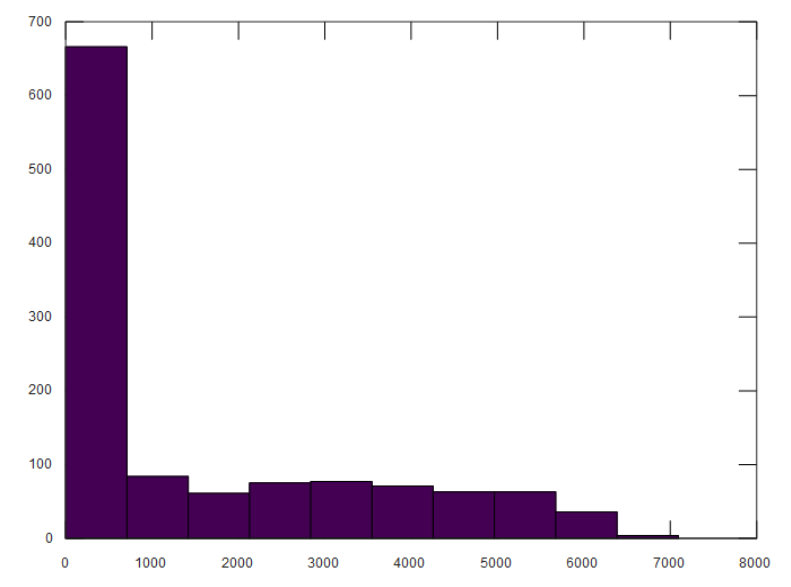
- En Moyenne est égal à `2629.7` pour le Hill Climber contre `1485.6 pour la recherche aléatoire`
- Un minimum de `0` contre 0 pour la recherche aléatoire
- Un maximum de `7251` contre `7092 pour la recherche aléatoire`
- Le Hill climber n'as pas trouvé de solution possible 400 fois contre 680 pour la recherche aléatoire

Ici on constate une nette différence entre le Hill Climber et la recherche aléatoire, on ne gagne plus 10% mais quasiment 90% sur le score moyen, et même la meilleure solution du Hill Climber reste préférable à la meilleure solution aléatoire, il trouve également plus souvent des solutions possibles que la recherche aléatoire.

Histogramme des scores du Hill Climber sur 1 000 tentatives

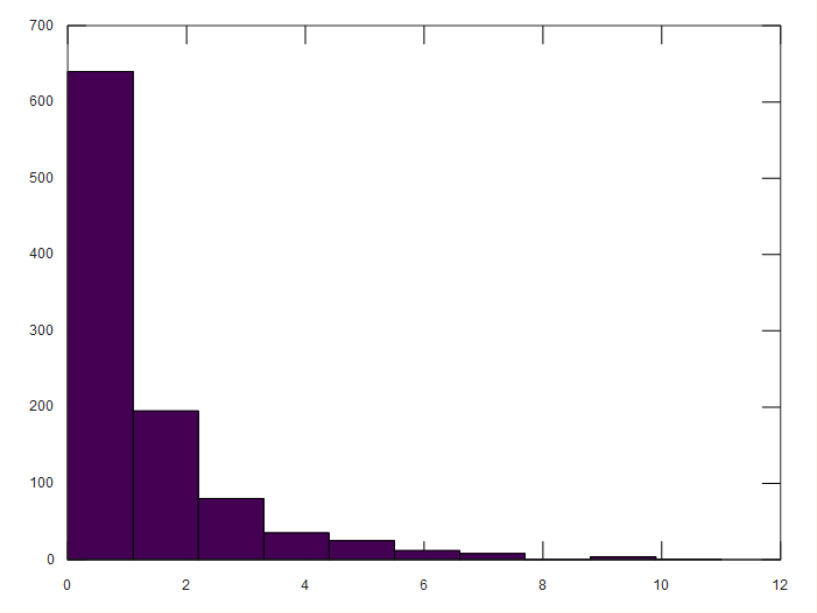


Histogramme des scores de la recherche aléatoire sur le même nombre d'itérations



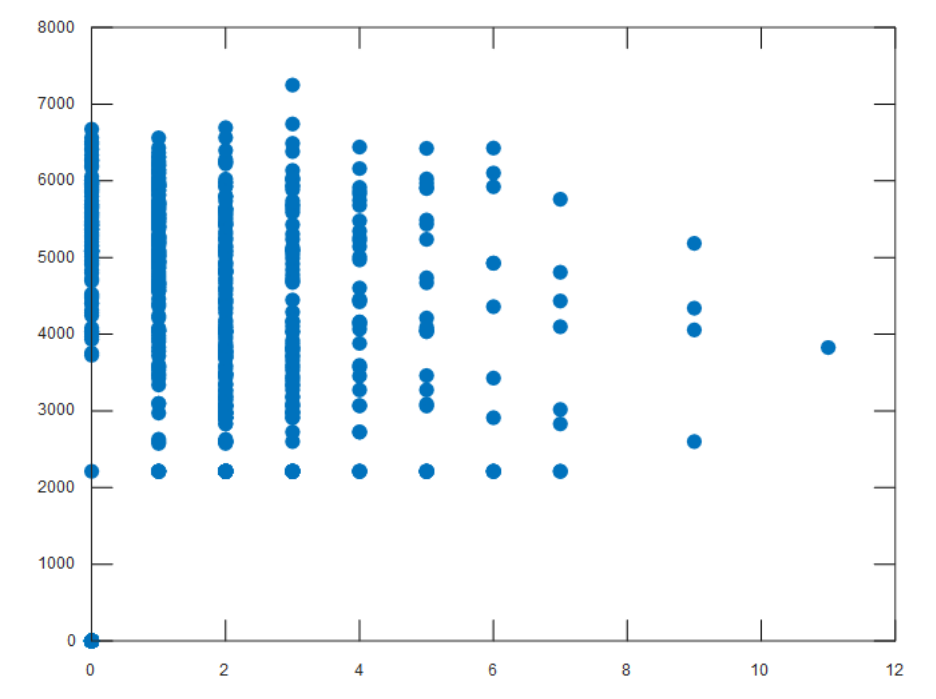
On remarque le même comportement que la recherche sans contrainte à l'exception du grand nombre de scores très bas qui s'explique par l'impossibilité de nombreuses solutions ne respectant pas les contraintes et donc mise à 0 du fait de la méthode de pénalisation.

Histogramme du nombre d'itération du Hill Climber



On remarque un nombre d'itérations réellement bas de plus dans la grande majorité des cas on a même aucun déplacement de la part de notre Hill Climber.

Nombre d'itérations en fonction des scores obtenus



On ne remarque pas de corrélation entre le score et le nombre d'itérations du Hill Climber

Seconde méthode de pénalisation : weight penalty

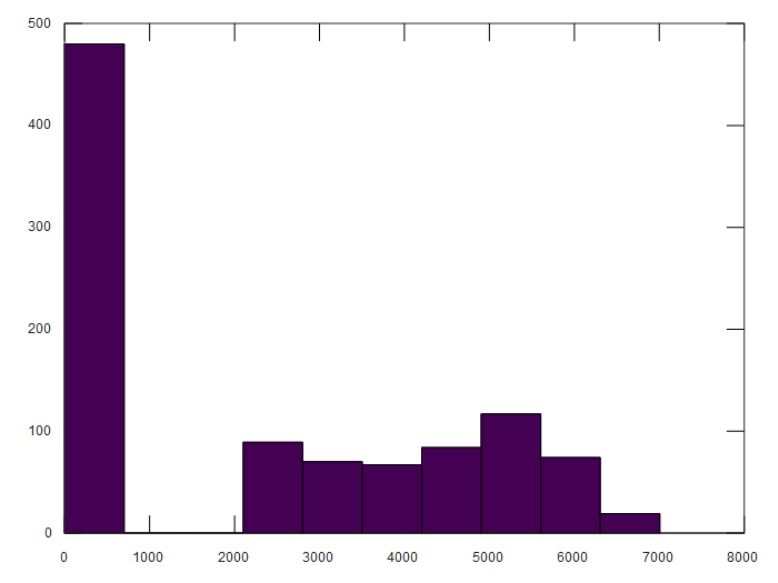
Ici contrairement à la méthode précédente on permettra au Hill Climber de passer par des solutions impossibles. Solution qui verront leur note diminuer proportionnellement au dépassement des contraintes. Cependant étant donné que seules les solutions possibles nous intéressent dans les résultats, on mettra la note des solutions impossibles à 0.

Sur les 1 000 lancers ont à un score qui :

- En Moyenne est égal à `2248.9`
- Un minimum de `0`
- Un maximum de `7006`
- On ne trouve pas de solutions possibles dans `480 lancers`

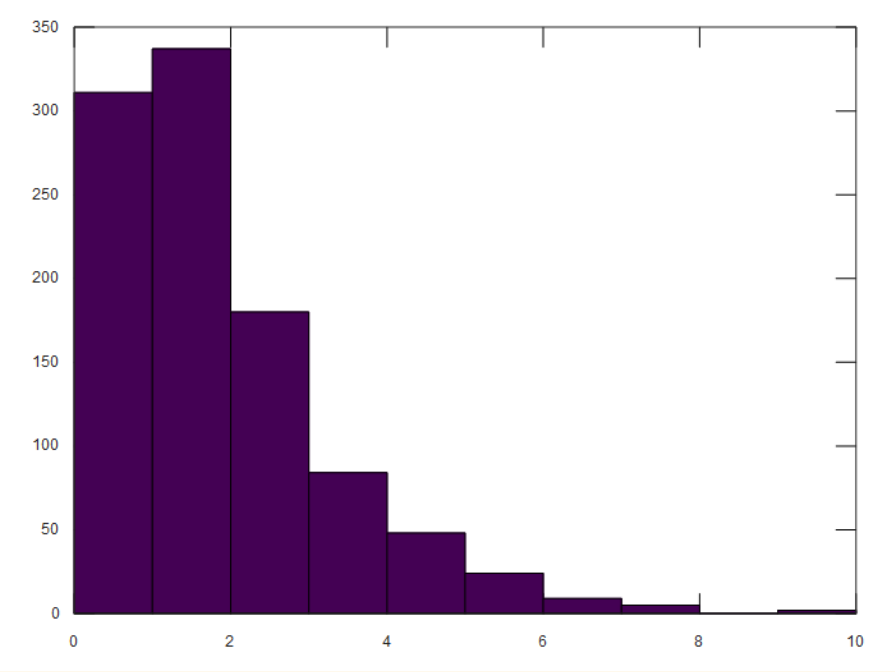
Ici on constate que les résultats sont légèrement moins bons que ceux du only feasible.

Histogramme des scores sur 1000 tentatives



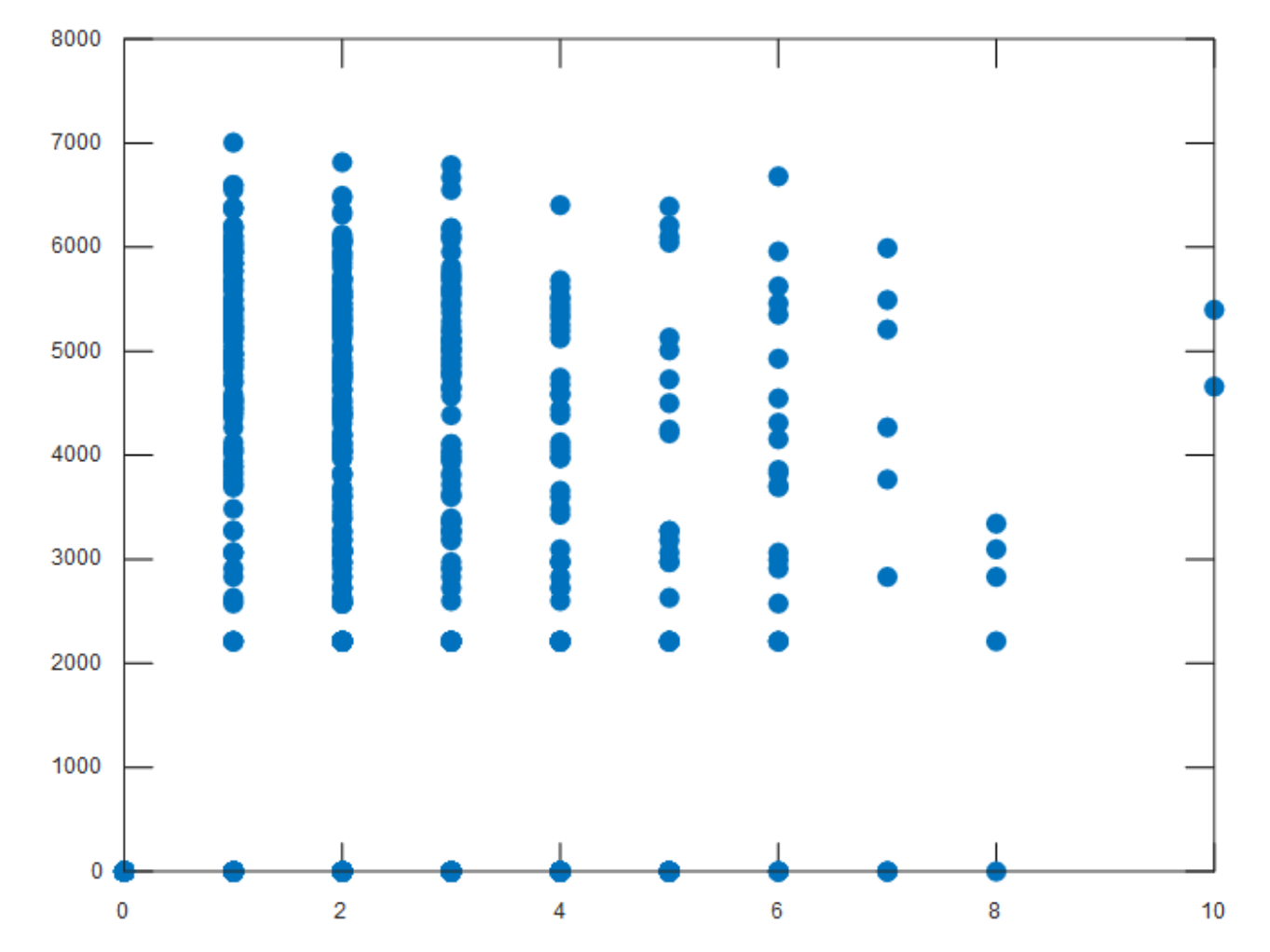
La répartition des scores semble similaire à celle du only feasible

Histogramme du nombre d’itération du Hill Climber sur 1 000 tentatives



On constate que bien que le nombre d'itérations reste relativement bas celui-ci est légèrement supérieur à ceux du only feasible puisque désormais la majorité des Hill Climbers ont au moins 1 itération.

Les scores en fonction du nombre d'itérations



La seule corrélation qu'on peut observer est que toutes les solutions possibles ont au moins 1 itération.

En conclusion bien que ses résultats soient légèrement moins bons , comme on pouvait s'y attendre cette méthode de pénalisation a permis à notre Hill Climber de pouvoir explorer un peu plus notre espace de solutions.

Troisième méthode de pénalisation : dimensional scoring

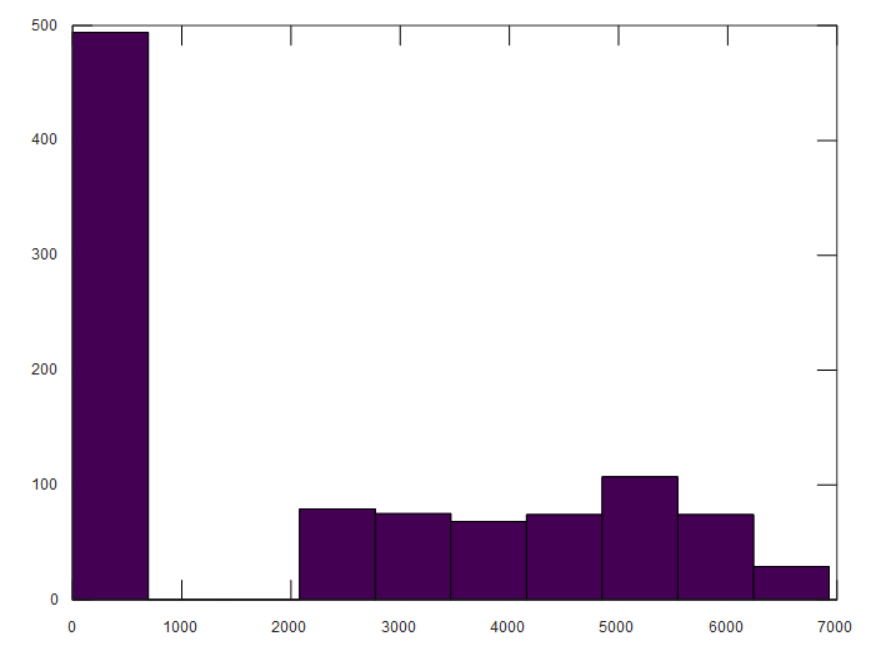
Dans cette méthode de pénalisation on retirera simplement 1/5ème de la note pour chaque dimension dont la contrainte n'a pas été respecté.

Sur les 1 000 lancers ont à un score qui :

- En Moyenne est égal à `2179.5`
- Un minimum de `0`
- Un maximum de `6929`
- On ne trouve pas de solutions possibles dans `494 lancers`

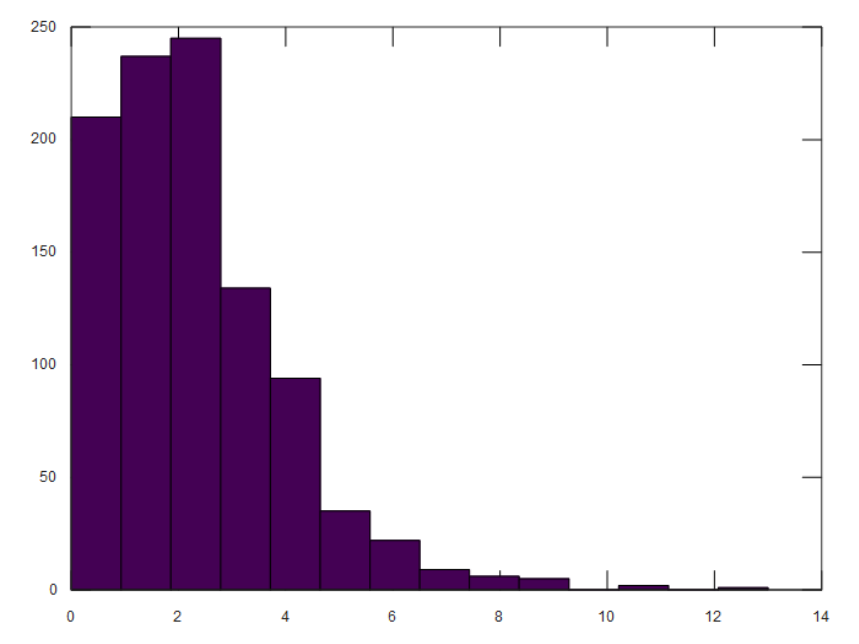
Ici on constate que les résultats sont légèrement moins bons que les 2 méthodes de pénalisations utilisées précédemment.

Histogramme des scores



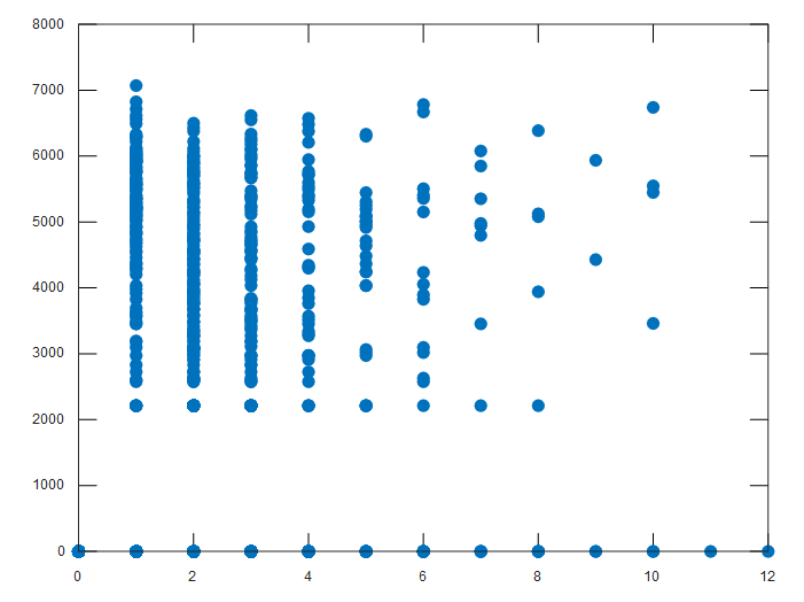
La répartition des scores semble très similaire aux 2 autres méthodes de pénalisation

Histogramme du nombre d'itérations



On constate que bien que le nombre d'itérations reste relativement bas celui-ci est légèrement supérieur à ceux du weigth penality puisque désormais la majorité des Hill Climbers on au moins 2 itérations.

Les scores en fonction du nombre d'itérations



Ici encore à l'exception du fait que les solutions possibles aient au moins 1 iteration on ne trouve pas de corrélation entre les scores et le nombre d'itérations

Troisième méthode de pénalisation: dimensional scoring with weight penalty

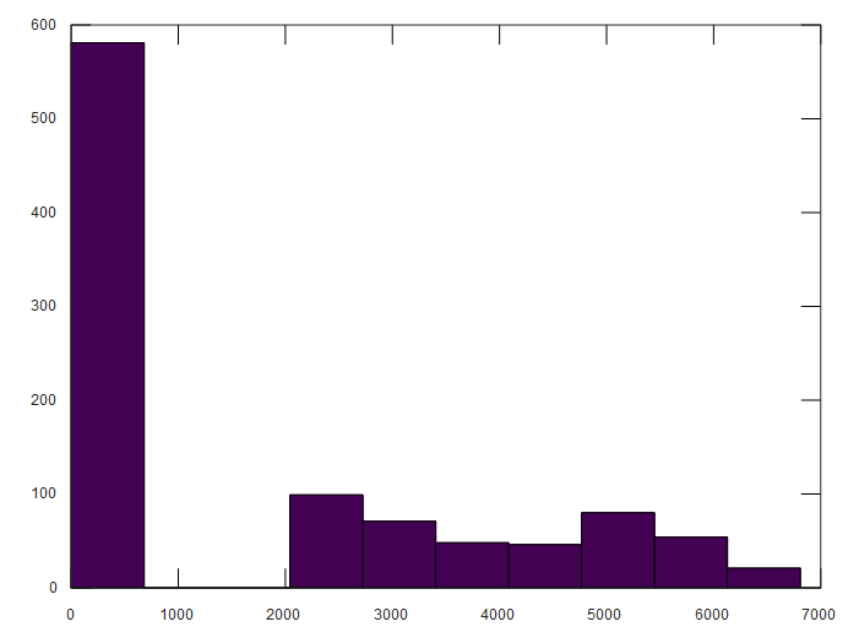
Cette méthode de pénalisation étant la combinaison des 2 méthodes précédentes. Pour chaque dimension, un score est calculé comme avec weight penalty, le score de la solution est la moyenne des scores sur chaque dimension.

Sur les 1 000 lancers on à un score qui :

- En moyenne est égal à 1688.9
- Un minimum de `0`
- Un maximum de `6810`
- On ne trouve pas de solutions possibles dans `581 lancers`

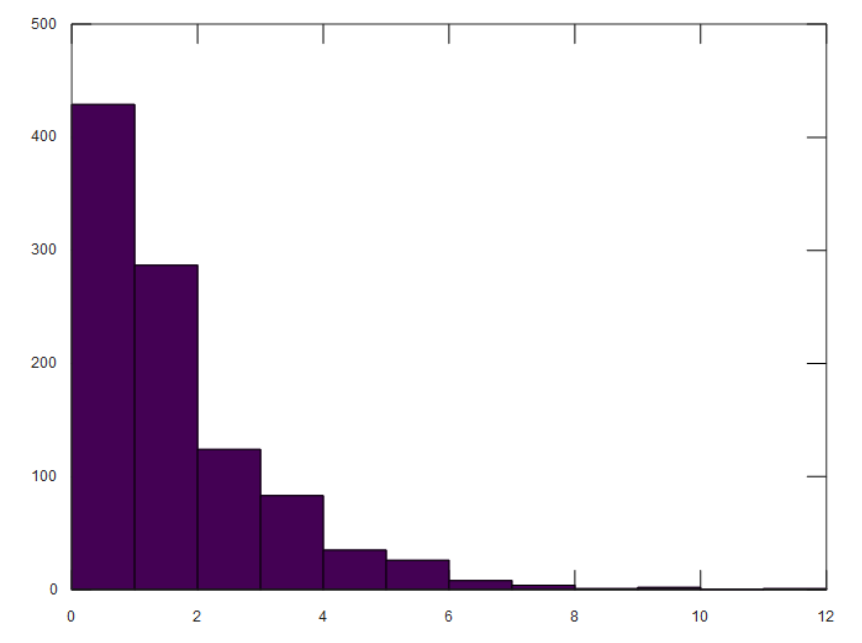
Ici on constate que les résultats sont moins bons que les 3 méthodes de pénalisations utilisées précédemment.

Histogramme des scores



La répartition des scores semble très similaire aux 3 autres méthodes de pénalisation

Histogramme du nombre d'itérations



Ici l'algorithme semble faire moins d'itération que les 2 méthodes dont il est issu, ainsi la très grande majorité des lancers n'ont aucune itération ou une seule itération.

En conclusion cette méthode de pénalisation semble être la plus mauvaise des 4. Contrairement à la méthode only feasible qui semble être la meilleure puisqu'elle a à la fois le score moyen le plus élevé sur 1 000 lancers et elle prend également moins d'itérations que ses consoeurs pour arriver à ses solutions.

Le recuit simulé

Le recuit simulé est contrairement au Hill Climber une méthode de recherche de solution locale. En réalité elle se rapproche assez d'un Hill Climber auquel on accorderait plus de liberté lui permettant de sortir de son espace de solutions locales pour explorer les espaces de solutions voisins.

Le fonctionnement du recuit simulé se rapproche beaucoup de celui du Hill Climber à l'exception d'une chose, quand la note de la solution voisine est inférieure à celle de la solution actuelle il peut modulo un nombre aléatoire et une "Température"(plus la température est haute plus il a de chances de le faire) explorer cette solution, de plus à chaque fois qu'il prendra une solution moins bonne que sa solution actuelle la température va baisser jusqu'à être trop basse pour permettre a l'algorithme de changer d'espace de solution (il adoptera alors exactement le même comportement qu'un Hill Climber).

Dans cette partie contrairement à la précédente on ne testera que la méthode de pénalisation "Only Feasible" mais sur plusieurs températures initiales différentes nous permettant par la même occasion de déterminer l'impact de la température initiale sur les performances du recuit simulé.

Sachant que : $\tau 0 = e\ \Delta 0\ /\ T0$

Et qu'un $\tau 0 = 0.50$ est démarrage à haute température (de qualité médiocre) et qu'un $\tau 0 = 0.20$ est démarrage à basse température (de bonne qualité). En effet plus $\tau 0$ est haut plus il peut explorer de solutions et donc moins ces solutions seront de bonne qualité en moyenne et à l'inverse plus $\tau 0$ seras bas et plus elles seront bonnes en moyenne et plus son fonctionnement se rapprochera de celui d'un Hill Climber.

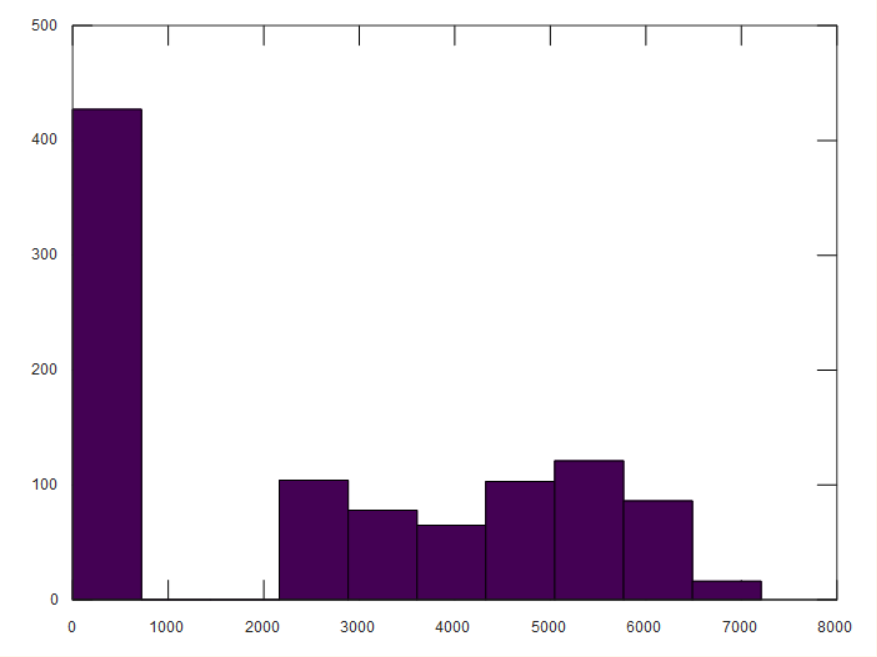
Ainsi on s'arrangera pour avoir des températures initiales qui feront varier $\tau 0$ entre 0.2 et 0.5.

Résultats pour $\tau 0 = 0.5$

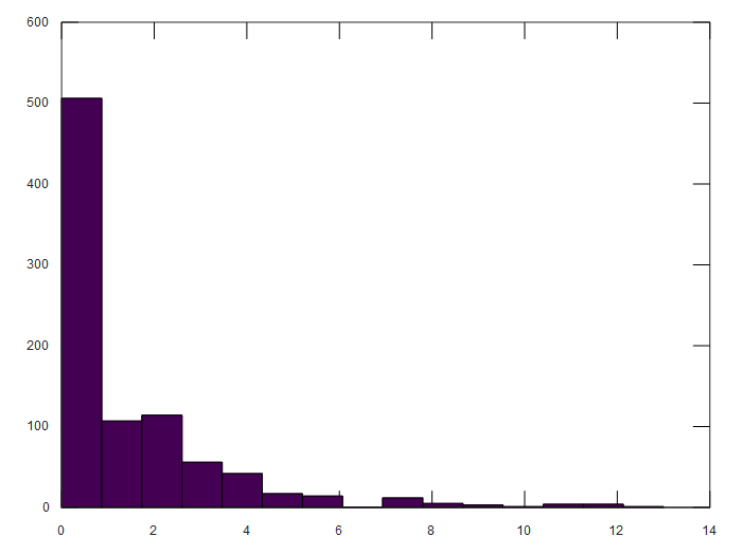
Sur les 1 000 lancers ont à un score qui :

- En moyenne est égal à `2515.4`
- Un minimum de `0`
- Un maximum de `7208`
- On ne trouve pas de solutions possibles dans `427 lancers`

Histogramme des scores



Histogramme du nombre d'itérations



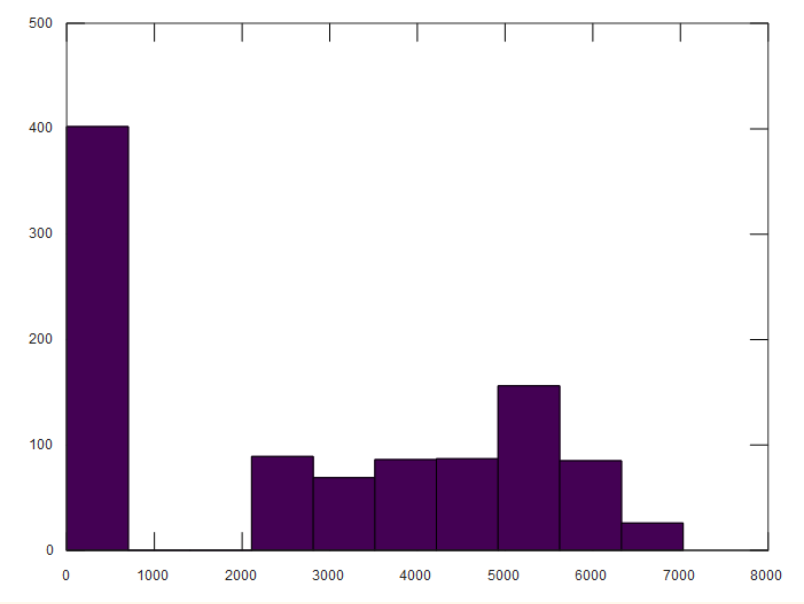
Pour un total de 6389

Résultats pour $\tau 0 = 0.4$

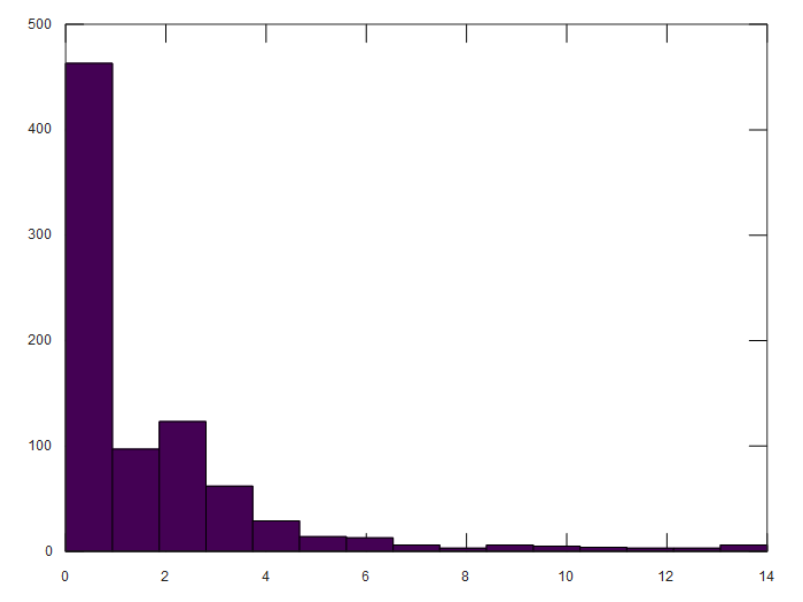
Sur les 1 000 lancers on a un score qui :

- En moyenne est égal à `2644.7`
- Un minimum de 0
- Un maximum de 7030
- On ne trouve pas de solutions possibles dans 430 lancers

Histogramme des scores



Histogramme du nombre d'itérations



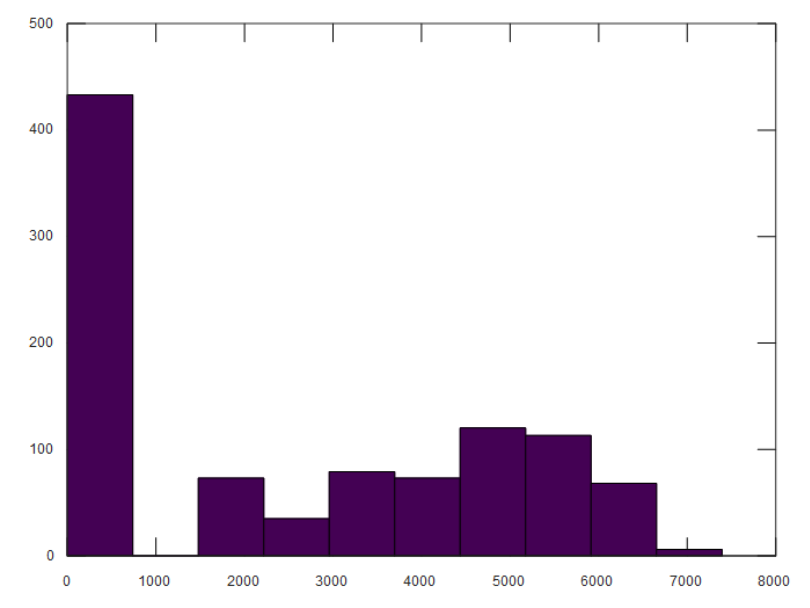
Pour un total de 8376

Résultats pour $\tau 0 = 0.2$

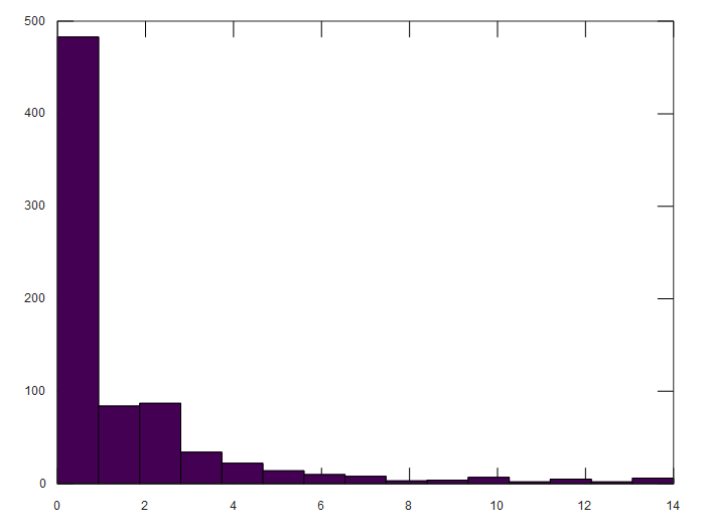
Sur les 1 000 lancers ont à un score qui :

- En Moyenne est égal à 2473.7
- Un minimum de 0
- Un maximum de 7393
- On ne trouve pas de solutions possibles dans 433 lancers

Histogramme des scores



Histogramme du nombre d'itérations



Pour un total de 14250

En conclusion on remarque plusieurs choses :

-La température ne semble pas avoir d'influence sur la répartition des scores ou du nombre d'itérations

-Plus la température est basse moins les scores du recuit simulé sont bons

-Plus la température est basse plus le recuit simulé aura d'itérations.