Julien PELEGRI 29/04/2019

PRPD - TP : ENVELOPPE CONVEXE

Objet de ce rapport :

Nous avons étudié le problème des enveloppes convexes en implémentant en Python 3.7 les méthodes de résolutions suivantes : Exhaustive, Graham, Jarvis, Eddy Floyd et Shamos.

Ce compte rendu a pour but de comparer ces méthodes d'un point de vue de la complexité temporelle en s'appuyant sur différentes simulations. Dans nos programmes il n'y a pas de différences notables d'un point de vu complexité spatiale.

Plan :

Remarque :

Pour les simulations 1, 2.1, 2.2 nous avons, pour un nombre de points fixé, testé des cas particuliers de répartition de points. Cela a permis de mettre en avant les écarts d'efficacité entre Shamos, Graham, Jarvis et Eddy Floyd.

Dans la partie 2.3, l'objectif est de comparer les complexités asymptotiques. Pour ce faire on réalisera un grand nombre de simulations sur un nombre important de points générés aléatoirement.

Les algorithmes sont exécutés à chaque fois sans aucun « print » (résultats intermédiaires ou final) pour pouvoir comparer au mieux les complexités.

Les calculs ont été effectués sur mon ordinateur personnel, les temps de calculs étaient différents de ceux des ordinateurs disponibles lors des TP.

1/ Tests de fonctionnement des différentes méthodes :

On test le fonctionnement de nos algorithmes sur 15 points :

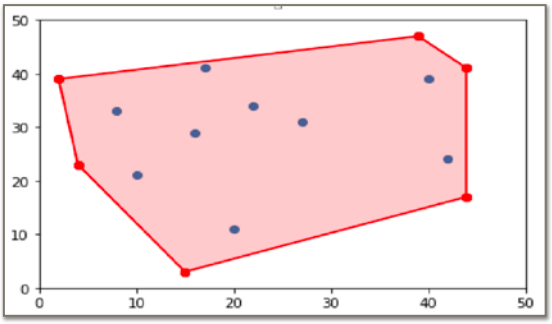


Illustration d'une solution pour une répartition particulière de 15 points

METHODE	TEMPS (en secondes)	NOMBRE DE SIMULATIONS
Exhaustive	**2,5*10^4**	**10 (ça prend déjà bcp de temps…)**
Jarvis	**0,06**	**200 000**
Graham	**0,07**	**1 000**
Eddy Floyd	**0.11**	**200 000**
Shamos	**0.18**	**100 000**

Remarque : les temps ci-dessus sont issus de la moyenne des exécutions. Cela permet de réduire l'incertitude puisque la résolution est faite sur un ensemble de points aléatoires à chaque fois.

On constate qu'une méthode exhaustive n'est pas envisageable pour un problème de grande taille, nous abandonnons donc cette solution pour la suite.

Les autres méthodes ont, elles, des résultats trop proches pour pouvoir conclure quant à leurs différences de complexités.

2/ Étude comparative des différentes méthodes :

2.1/ Étude pour 50 points judicieusement reparti et une fenêtre de taille [[-100,100],[-100,100]] :

2.1.1/ Répartition aléatoire dans toute l'enveloppe :

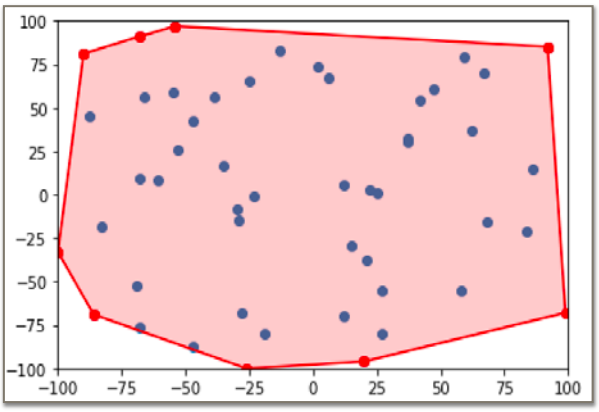


Illustration d'une solution pour un ensemble de 50 points

METHODE	TEMPS (en millisecondes)	NOMBRE DE SIMULATIONS
Jarvis	**0.27**	**10 000**
Eddy Floyd	**0.33**	**10 000**
Graham	**0.29**	**1 000**

METHODE	TEMPS (en millisecondes)	NOMBRE DE SIMULATIONS
Shamos	**0.79**	**10 000**
Exhaustive	**> 10 minutes**	

On remarque que les trois premières méthodes semblent être équivalentes.

2.1.2/ Répartition des points très resserrée :

Les points générés aléatoirement sont compris entre [[0,50],[0,50]]

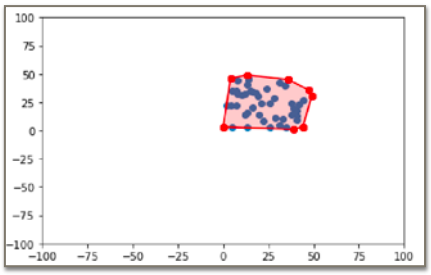


Illustration d'une solution pour un ensemble de points resserrés

METHODE	TEMPS (en mili secondes)	NOMBRE DE SIMULATION
Jarvis	**0.27**	**10 000**
Eddy Floyd	**0.33**	**10 000**
Graham	**0.31**	**200**
Shamos	**0.78**	**10 000**

2.1.3/ Répartition des points excentrée :

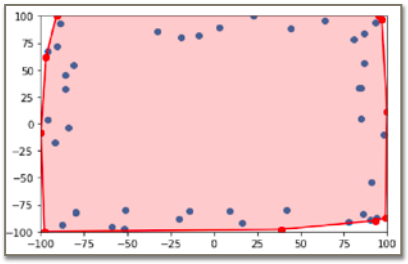


Illustration d'une solution pour un ensemble de points excentrés

Ici les points sont générés pour être sur les bords (à 0,8 fois le minimum ou le maximum) de façon aléatoire.

METHODE	TEMPS (en mili secondes)	NOMBRE DE SIMULATION
Jarvis	**0.35**	**1 000**

METHODE	TEMPS (en mili secondes)	NOMBRE DE SIMULATION
Eddy Floyd	**0.33**	**10 000**
Graham	**0.28**	**200**
Shamos	**0.82**	**10 000**

On se rend compte avec cette seconde simulation que le choix du meilleur algorithme dépend de la répartition des points. Ici Jarvis qui était la méthode la plus rapide avant n'est plus que la troisième plus rapide.

Cela est cohérent avec les complexités asymptotiques des méthodes.

Remarque :

Pour cette simulation les fluctuations de temps sont plus importantes. Les algorithmes ayant plus de difficultés à s'exécuter, nous avons réalisé 10 000 simulations à 10 reprises pour en obtenir 100 000 (on a bien sûr fait la moyenne des temps).

Le nombre de simulations pour Graham est moindre à cause de l'erreur « *maximum recursion depth exceeded while calling a Python object* », même en essayant d'augmenter le nombre d'itération dans le programme le problème n'est pas résolu.

2.2/ Étude du temps de l'exécution en fonction du nombre de points générés :

Ici on a fait des simulation pour un nombre a de points dans une grille [[-10*a,10*a],[-10*a,10*a]].

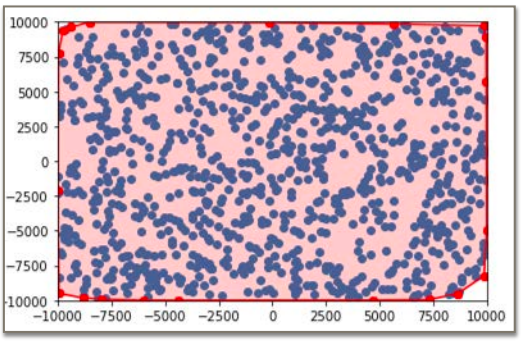
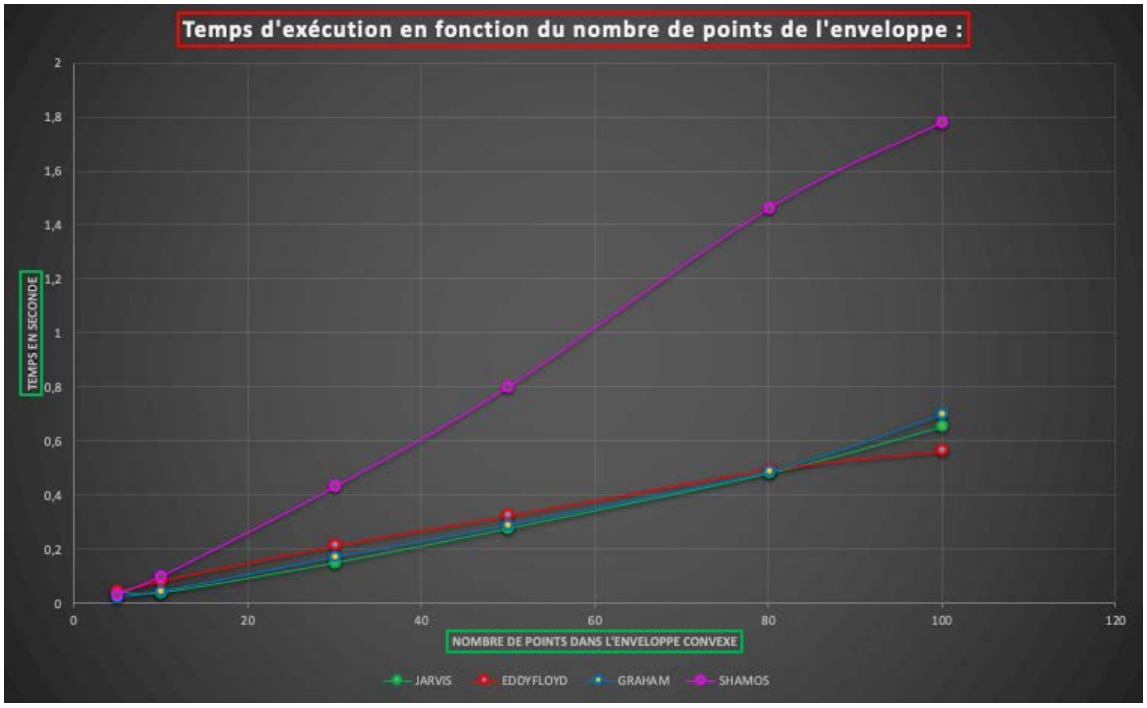


Illustration d'une solution pour a=50

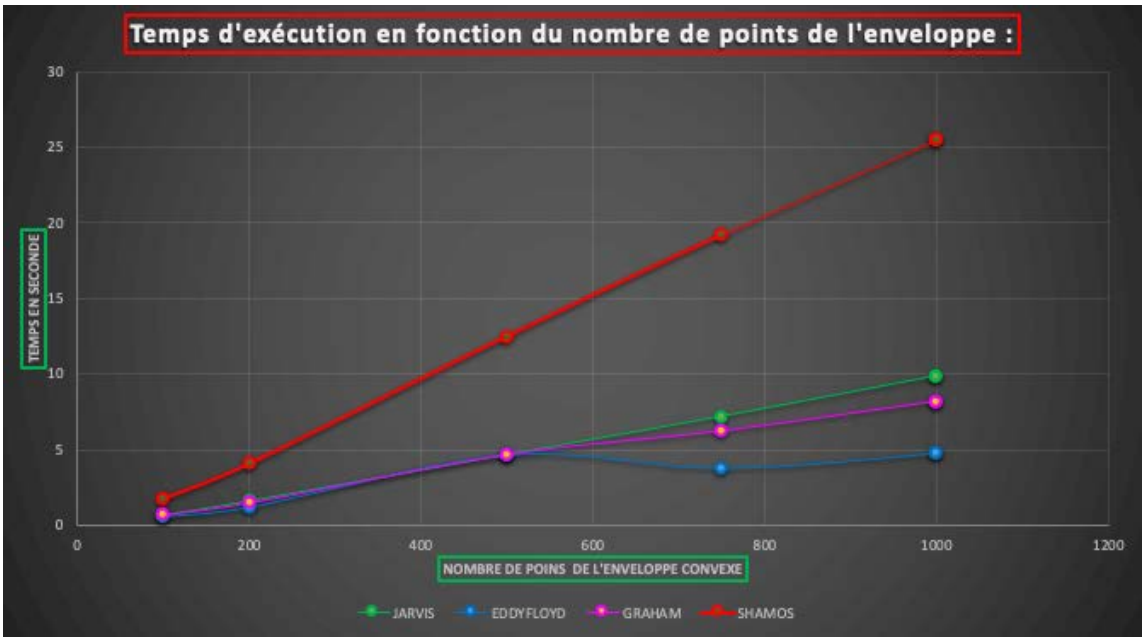
La première courbe correspond à un faible nombre de points, la deuxième à un nombre important de points.

Courbes pour une variation de « a » entre [0, 100]



Les méthodes sont censés être de même complexité, néanmoins Shamos présente un écart important (pour 100 points on triple le temps).

Courbes pour une variation de « a » entre [100, 1000]



Le faible écart entre Eddy Floyd, Graham, Jarvis peut s'expliquer par le fait que se sont les complexités asymptotiques qui sont les mêmes, on a donc des différences légères.

3/ Comparaison de nos résultats expérimentaux avec la théorie :

Complexités des algorithmes :

METHODE	COMPLEXITÉ ASYMPTOTIQUE
Exhaustive	**Exponentiel : O(n^3)**
Eddy Floyd	**O(n*log(n))**
Graham	**O(n*log(n))**
Shamos	**O(n*log(n))**
Jarvis	**O(n*h)**

On a noté n le nombre total de points et h le nombre de points contenus dans l'enveloppe convexe.

Conclusion :

En général les algorithmes les plus efficace sont Graham, Eddy Floyd et Jarvis, Shamos est légèrement moins performant. Jarvis est le plus efficace dans le cas particulier où le nombre de points est élevé et le nombre de points de l'enveloppe convexe est faible.

Cependant dans le pire des cas la complexité de Jarvis est quadratique !