Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

Recherche des skylines à base de l'algorithme branch & bound

Réalisé par :

- **M elle. Benmostefa Nor El Houda**
- **Mr. Sekkak Oussama**

Présenté le 27 mai 2015 devant la commission d'examinassions composée de MM.

- Mr. Hadjila Fethellah (Encadreur)
- Mr. Etchiali.A (Examinateur)
- Mr. Benziane.Y (Examinateur)

Année universitaire: 2014-2015

بسم الله الرحمن الرحيم

Remerciements

Remerciements

A l'aide de DIEU tout puissant, qui nous a aidé et nous a donné la patience, le courage la force et qui trace le chemin de notre vie, nous avons pu arriver à réaliser ce modeste travail.

Nous tenons à remercier tout particulièrement monsieur HADJILA.F enseignant à l'université Abou Bekr Belkaid notre encadreur de mémoire pour son aide, son soutien, ses conseils, sa patience et sa générosité. Son ouverture d'esprit, sa disponibilité et ses analyses pertinentes ont contribué à rendre cette étude agréable et enrichissante.

Nos remerciements s'adressent également à :

- Mr ETCHIALI.A
- Mr BENZIANE.Y

Pour avoir porter un intérêt à ce modeste travail, et d'avoir accepter de l'examiner.

Merci aussi à tous nos enseignants pendant tout le parcours scolaire.

Enfin, merci à tous ceux qui ont contribué de prés ou de loin à l'aboutissement de ce travail, par leur confiance et leurs encouragements.

Merci aussi à internet ☺ .

Sincèrement...

الحمد لله.

DEDICACE

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que : **Je dédie cette thèse de Licence à :**

A Ma tendre Mère : Vous représentez pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Vous avais fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études.

A Mon très cher Père : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail et le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation le long de ces années.

A mon cher frère : Moncif que j'aime beaucoup.

A ma chère sœur : Wassila et mes adorables nièces Selma&Sara que dieu les protège sans oublié son mari Boumedyen.

A mes chéres grands parents, que Dieu me les garde incha'allah.

A tous mes oncles, mes tantes, cousins et cousines et tous les membres de ma famille.

A mon binôme Houda et tout sa famille.

A tous mes amis et mes collègues.

A tous mes enseignants depuis mes premières années d'études.

Aussi à tous ceux qui me sens chers et que j'ai omis de citer.

Oussama

DEDICACE

Merci ALLAH de m'avoir donner la capacité, la force, la patience d'aller sur le droit chemin tout au long de nos années d'étude jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel et de dire" Ya Rab".

Je dédie ce modeste travail à celle qui m'a mis au monde, le symbole de tendresse et d'amour. Quoi que je fasse, je ne pourrais jamais te récompenser pour les grands sacrifices que tu as fait et continue de faire pour mon bonheur et ma réussite.

Ma Mère, qui a toujours cru en moi et a encouragé.

Mon Père, école de mon enfance, qui a été mon ombre durant toutes les années des études, et qui a veillé tout au long de ma vie à m'encourager, à me donner l'aide et à me protéger.

Que dieu les garde et les protège.

A mes très chères sœurs Latifa, Aya et mes adorables neveux.

A mes frères Kheireddine et Ilyes que j'aime beaucoup.

A mes grands parents pour toutes leurs prières, que Dieu me les garde en très bonne santé.

A mes oncles, mes tantes, cousins et cousines et tous les membres de ma famille, petits et grands.

Aussi a mon binôme Oussama et tout sa famille et spécialement ces mignonnes nièces.

A tous mes amis et mes collègues.

A tous ceux qui me sont chères.

A tous ceux qui m'aiment.

A tous ceux que j'aime.

Je dédie ce travail.

Nor El Houda

Liste de matières :

Liste des figures :

Liste des tables :

Résumé :

L'étude et le traitement des requêtes skylines a reçu une grande importance , durant ces dernières années, l'objectif de notre travail est de concevoir et implementer un algorithme de recherche des skylines, appelé branch & bound skyline.

Cette approche est considérée comme une amélioration de NN, en particulier, elle permet l'optimisation du temps requis pour extraire les points skylines.

Cet algorithme se base sur trois idées clés : l'adoption du principe du plus proche voisin pour localiser les skylines, l'utilisation de la structure des R-Tree pour accélérer l'accès aux points, l'exploitation du B&B pour élaguer l'espace de recherche. Les résultats obtenus confirment l'efficacité de l'algorithme.

Mots cles :

Requêtes des skylines, Algorithme avec index, BBS (Branch and Bound Skyline),index spaciale, R-Tree.

Abstract :

The study and the treatment of the skyline requests has received a great importance, in the last few years, the objective of our work is to conceive and implement an algorithm that retrieves the skylines points. To this end we develop a solution called branch & bound skyline.

This approach is an improvement of NN algorithm, in particular, it allows the optimization of necessary time to extract the points skylines.

This algorithm is based on three key ideas: the adoption of the principe of nearest neighbor to locate the skylines, the use of the structure of R-Tree to acccelerate the points access, the exploitation of the B&B principle to prune the search space. The obtained results confirm the effectiveness of the algorithm.

keywords :

skylines's queries, Algorithm with index, BBS (Branch and Bound Skyline), spacial index, R-Tree.

ملخص :

إن دراســة ومعالجــة « skyline » حقق اهتماما بالغا في الآونة الأخـيرة نظرا لايجابياته المتعـددة في مجـــال تحسـين مطـالب الحلول الممكنة لإشكالية ما, فهدفنا في هذا الموضوع هو استخدام و تجسيد خوارزمية من بين عدة خوارزميات البحث عن الحل الأمثل و الذي يسمى « Branch and bound skyline ».

هذا المنهج يعتبر تحديث و تطوير لخوارزمية « Nearest Neighbor » بالتحديد أنه يمكن من التقليل وخفض مدة التنفيذ المطلوبة لاستخراج جميع إحداثيات نقط الحلول المثلى حسب معايير مختلفة ومتعددة .

إن هذه الخوارزمية تستند على ثلاثة أفكار رئيسية, أولا الاعتماد على مبدأ أقرب جار ممكن من أجل تحديد وحصر نقط الحلول, ثانيا استعمال بنية R-tree من أجل تعجيل الوصول إلى النقط, وثالثا وأخيرا تشغيل مبدأ التجزئة و الربط بغية تقليص مساحة البحث.

النتائج المحققة في النهاية و طريقة الوصول إليها تدل على أهمية وكفاءة هذه الخوارزمية .

Resumen *:*

El estudio y el tratamiento de las consultas "skylines" ha recibido una gran importancia ,durante estos últimos años,el objetivo de nuestro trabajo es diseñar y poner en práctica un algoritmo de búsqueda de "skyline" llamado " branch and bound".

Este enfoque se considera como una mejora del algoritmo NN ,en particular permite la optimización del tiempo requerido para extraer los puntos "skyline" .

Este algoritmo se basa en tres ideas fundamentales : la adopción del principio del vecino más cercano para localizar los puntos "skyline", el uso de la estructura de la R-Tree para acelerar el acceso a los puntos , funcionamiento del B&B a podar el espacio de búsqueda.

Los resultados confirman la eficacia del algoritmo.

palabra clave :

Consultas de skylines , Algoritmo con índice, BBS (Branch and Bound Skyline), índice espacial, R- Árbol.

Chapitre I
Introduction generale

I.1 Contexte du travail :

La majorité des problèmes de recherche nécessitent une phase d'optimisation (la selection de la meilleure solution), ce problème devient de plus en plus difficile lorseque le volume des données est toujours grand.

Dans ce contexte et afin de trouver un mécanisme d'optimisation multicritères , la communauté de recherche a introduit le concepte de « Skylines » autrement dit une solution paréto optimale du problème d'optimisation multi-critéres.

I.2 Problématique :

Dans ce sujet on traite la problématique de recherche des éléments paréto optimaux. Vu que la majorité des algorithmes posent des problèmes d'inéfficacité (en terme de leurs temps d'exécution), il est donc impératif de trouver des approches apportant des remèdes à ces lacunes.

I.3 Contribution :

Dans ce travail on propose L'adoption du principe du plus proche voisin, pour l'extraction des skylines, cette approche se base sur une distance donnée (ex. Distance de Manhattan, …), Pour localiser les points les plus intéressants (ceux qui sont proches par rapport à l'origine du repère).

Pour accélérer la recherche des plus proches voisins, on implémente une structure appelée R-Tree, cette dernière est un arbre équilibré de degré fixé n ayant les caractéristiques suivante:

- Le nœud racine possède entre 2 et **n** fils.
- Tout nœud interne a entre **m** et **n** fils avec **m** un entier inferieur a **n**/2. Cet arbre sera parcouru plusieurs fois à chaque itération de l'algorithme B&B.

L'exploitation de l'algorithme permet d'éliminer les régions de l'espace à travers des heuristiques bien déterminées, par conséquent il suffit de rechercher uniquement dans les régions susceptibles de contenir d'autres points candidats.

I.4 Plan du mémoire :

Le reste de ce mémoire est organisé comme suit:

Le deuxième chapitre :

Est consacré a définir le concept de Skyline ; on présentera les différents algorithmes de recherche des skylines.

Le troisième chapitre :

Introduit l'algorithme multicritère « BBS» .On présente aussi le prototype et les résultats obtenus.

La conclusion générale :

Résume les résultats de notre travail, et présente les perspectives que nous souhaitons réaliser dans le futur

Chapitre II
Recherche Des Skylines

II.1Introduction :

Le concept de requêtes skylines a été introduit pour formuler des recherches multi-critères. Lorsque ces critères sont conflictuels, l'opérateur skyline retourne les meilleurs compromis entre plusieurs critères. ces résultats sont incomparables entre eux. généralement les skylines répondent efficacement au problème de recherche pour une base de données ayant un nombre limité de critères (typique en e-commerce), mais lorsque le nombre de dimension augmente, ces approches rencontrent beaucoup de difficuléts . en effet, la probabilité d'objets incomparables augmente avec le nombre de dimension et entraine une surabondance de résultats conflictuels.

II.2 La dominance :

Les requètes skylines sont basées sur la relation de dominance.

L'idée d'utiliser la dominance au sens de Paréto a été proposée par Goldberg [Goldberg ,1989] pour résoudre les problémes proposés par Schaffer [Schaffer , 1985]. L'auteur suggère d'utiliser le concepte d'optimalité de Paréto pour respecter l'intégralité de chaque critère au lieu de comparer a priori les valeurs de différentes critères .L'utilisation d'une selection basée sur la notion de dominance de pareto entraine la convergence de la population vers un ensemble de solutions éfficaces. Ce concept ne permet pas de choisir une alternative plutôt qu'une autre mais il apporte une aide précieuse au décideur .[1]

II.3 Notion de dominance :

Un point **X** domine le point **Y** sur un domaine **D** noté **X≺DY**. Si et seulement si :

$\forall i \in [1, d] : p_i \leq q_i$ et $\exists j \in [1, d] : p_j < q_j$. [3]

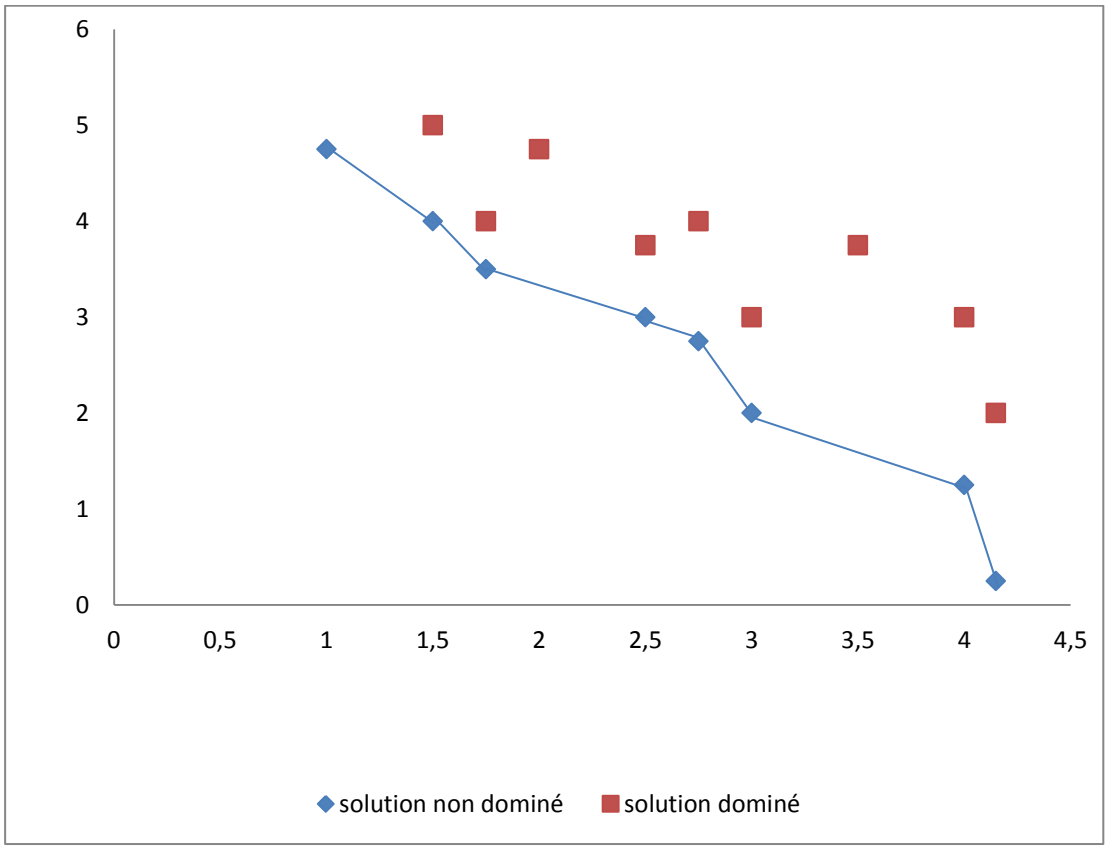


Figure II.1: Exemple de Front de Paréto dans un espace à deux critères.

II.4 Préférence :

L'utilisation de préférences permet de personnaliser la recherche multicritères et d'accroître la pertinence du résultat. L'exemple le plus connu est celui des requêtes skylines, basées sur le concept de dominance défini par Pareto. Ces requêtes permettent d'éliminer les n-uplets dominés par d'autres n-uplets. L'utilisateur pourra alors choisir parmi les n-uplets qui ne sont pas dominés, que l'on peut considérer comme les meilleurs choix. Cependant, l'un des principaux problèmes des requêtes skylines est l'augmentation trop importante de la taille du résultat lorsque le nombre de dimensions, ou critères, augmente, rendant le choix par l'utilisateur difficile**.**

L'idée générale est d'étendre les relations de dominances en introduisant des critères plus flexibles et personnalisés pour comparer les n-uplets, puis de les combiner progressivement afin de satisfaire au mieux les besoins de l'utilisateur. Des extensions ont été apportées à l'opérateur Skyline afin d'offrir à l'utilisateur la possibilité de classer les n-uplets de choisir la meilleure sélection ou encore de sélectionner les **k** meilleures solutions. L'utilisateur peut ainsi utiliser successivement plusieurs relations de préférences en les ordonnant afin de prendre en compte les priorités ou niveau de fiabilité qu'il attribue à chacune. Les algorithmes sont détaillés ainsi que l'expérimentation permettant de valider nos approches.

Exemple :

Soit un site d'un club de football qui fait des statistiques d'évaluation de ses joueurs pour trouver le meilleur joueur de la phases d'aller du championnat.

Cette recherche est basée sur plusieurs critères correspondants à chaque joueur qui a participé avec l'équipe durant cette période.

Joueurs	Passes complets /matche	passes décisifs/matche	But(s)	Fautes commises/matche	Ballons Perdus/matche
a	19	4	12	1	9
b	22	2	3	2	10
c	20	8	5	4	8
d	10	0	8	6	7
e	12	4	1	9	10
f	25	0	0	7	5
g	23	3	5	5	6
h	8	0	4	4	3
i	14	0	2	3	2
k	18	5	1	9	1
l	19	1	0	10	4
m	10	0	1	6	2
n	23	1	2	8	3

Table II.1 :L'évaluation des joueurs

II.5 Définitions :

II.5.1Problème multi-objectif (multi-critère) :

a) Un problème multi-objectif ou multicritère peut être défini comme un problème dont on recherche l'action qui satisfait un ensemble de contrainte et optimise un vecteur de fontion objectif.

Par la suite nous allons voir que les problèmes d'optimisation ont en général plusieurs solutions car la définition d'un optimum ne peut pas être établi dans les problème **multi-objectif** .[5]

b) Une action (ou un vecteur de décision) sera notée : $\chi = (\chi_1 , \chi_2 , \dots , \chi_n)$

Avec χ_i les variables de probleme et **n** le nombre de variables.

c) Les contraintes :

Les contraintes seront notées :

$g_i(\chi) = 0$ avec i= 1 …. , I , et $\mathbf{h_j}(\chi) <= 0$ $avec$ $\mathbf{j} = 1, \dots . . , J$.[5]

d) Les vecteur de fonction objectif

Sera noté f : $f(x) = (f_1(x) , f_2(x) , \dots . , f_k(x))$ Avec f_i les objectifs ou critères de décisions et k le nombre d'objectifs sont des fonctions de minimisations.

$\mathcal{M}in\, f(x) = (f_1(x) , f_2(x) , \dots . , f_k(x))$, Avec :

x= $(x_1, x_2, \dots , x_n)$,

$g_i(\chi) = 0$ i $= 1,2, \dots , I,$

$h_j(\chi) <= 0$, $\mathbf{j} = 1,2, \dots , J,$ [5]

e) Un problème d'optimisation recherche l'action $\chi *$ telle que les contraintes $g_i(\chi *)$ soit satisfaite pour **i** = 1,… ,m et qui optimise la fonction :$f(x^*)=(f1(x^*), f2(x^*),\dots, fk(x^*))$

L'union de domaines de définition de chaque variable et les contraintes définies en équation forment un ensemble **E** que nous appelons l'Ensemble des action réalisable .

Nous appellerons **F** l'ensemble des objectifs réalisables. [5]

II.5.5 Skyline :

Un point **p** est un skyline sur un domaine **D** si et seulement si aucun autre point **q** (≠p) ne domine pas **p** dans **D**. soit un ensemble de données **S** sur **D**,un skyline est un ensemble des points optimaux noté :

Sky (D, E) P = {p ∈ E| (∀ q ∈ E, ¬ (q ≺D p))}. Si P =∅, Sky (D, E) P = E. [1]

II.6 Les algorithmes de recherche des skylines :

II.6.1 Les critères d'évaluations :

Avant de présenter les différents algorithmes, il nous faut pouvoir les comparer sur des critères identiques afin de mettre en évidence leurs forces et leurs faiblesses.

En proposent une liste sur laquelle nous nous appuyons sans la suivre strictement car certains points sont contradictoires et d'autres peu pertinents dans le cadre fixe. Voici la liste des critères choisis : [8]

II.6.1.1 Limitations :0

Ce critère a trait à l'efficacité mais surtout au passage à l'échelle des algorithmes. Il s'agit de donner les limites, par exemple au nombre de critères du Skyline et/ou a la taille de l'entrée, au delà des quelles l'algorithme n'est plus utilisable. [8]

II.6.1.2 Progressivité :

Les volumes des bases de données sont souvent énormes. Un simple parcours de la base peut prendre un temps non négligeable. C'est pourquoi nous souhaitons que nos algorithmes calculent et affichent les résultats trouvés le plus rapidement possible, au fur et à mesure qu'ils sont découverts. Idéalement, les premiers résultats devraient être envoyés quasiment instantanément. [8]

II.6.1.3 Équité :

L'algorithme ne devrait pas favoriser les points qui sont particulièrement bons dans une seule dimension. [7]

II.6.1.4 Incorporation des préférences :

Les utilisateurs devraient pouvoir déterminer l'ordre selon lequel les points Skylines sont rapportés. [7]

II.6.1.5 Universalité :

L'algorithme devrait s'appliquer à n'importe quel ensemble de données, ou distribution et dimensionnalité, en utilisant une certaine structure d'index.

-Les algorithmes qui vont être détailles maintenant peuvent se classer en deux catégories. Les premiers n'utilisent pas de structure d'indexation particulière et n'ont donc pas besoin d'un prétraitement pour fonctionner. Les seconds regroupent tous ceux qui sont bases sur des index. Ils les exploitent pour améliorer l'efficacité du calcul et modifier leur comportement, vis-à-vis par exemple de leur progressivité. La contrepartie est toutefois qu'il faut disposer de l'index nécessaire, ou bien le cas échéant de devoir le calculer, avant de pouvoir d'utiliser ces algorithmes. [8]

II.6.2 Les algorithmes sans index :

II.6.2.1 BNL (Block-Nested-loup) :

Cet algorithme consiste a :

•Comparer chaque point **p** à chaque autre point, et rapporte **p** en tant qu'élément skyline S s'il n'est pas dominé.

•balayer le fichier de données et maintenir une liste des points skylines candidats dans la mémoire central.

Au début la liste contient le premier point de repères, alors que pour chaque point suivant p, là il existe trois cas :

1- **p** : dominé par le point dans la liste (p→ jeté).

2- **p** : domine n'importe quel point dans la liste (p→ inséré, tous les points dominés par p→laissé tomber)

3- **p** : ni dominé, ni domine n'importe quel point dans la liste (p→ inséré, sans laisser tomber tout point)

• Si la liste devient plus grande que la mémoire centrale :

-Les points qui ni dominent ni sont dominés seront ajoutés au fichier temporaire →passage multiples de l'algorithme.

•Applicabilité large (peut être employé pour n'importe quel dimensionnalité sans indexation ou assortir le fichier de données). [2]

II.6.2.2 Divide and conquer (D&C) :

• L'algorithme DC divise l'ensemble de données en plusieurs séparations, de sorte que chaque séparation est ajustée dans la mémoire.

•Le Skyline partiel des points dans chaque séparation est calculé en utilisant un algorithme de mémoire centrale.

•L'espace de données est divisé en 4 séparations S_1, S_2, S_3, S_4 respectivement avec les skylines partiels {a, c, g}, {d}, {I}, {m, k}.

-Tous les points skylines de S_3 doivent apparaître dans le Skyline final.

-Tous les points de S_2 sont dominés immédiatement.

-Chaque point skyline dans le S_1 est comparé seulement aux points skyline de s3

-Chaque point skyline dans S_4 est comparé seulement aux points skyline de s3

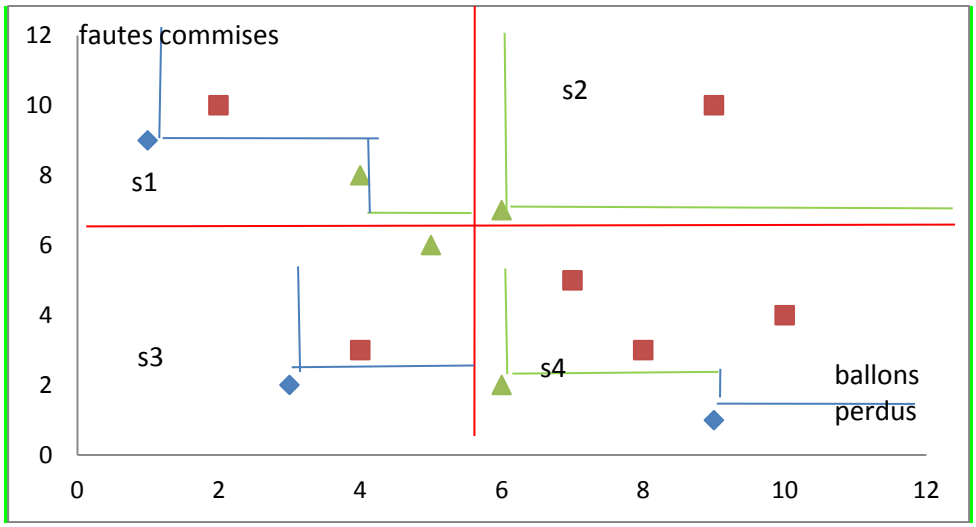


Figure II.2 : Divide and Conquer.

•Le DC est efficace seulement pour les petits ensembles de données

- Si l'ensemble de données entiers est ajustés dans la mémoire, l'algorithme exige seulement une application d'algorithme skyline de La mémoire centrale.
- Pour des grands ensembles de données, le processus de division exige la lecture et l'écriture d'ensemble de données entières au moins une fois, ainsi engager le cout significatif d'entrée-sortie.

•Cet algorithme n'est pas approprié au traitement en direct parce qu'il ne peut pas rapporter n'importe quel skyline jusqu'à la phase de division accomplit. [2,4]

II.6.2.3 Sort First Skyline (SFS) :

•C'est une variation de bloc Nested loop.

•SFS assortit l'ensemble de données entier selon une fonction de préférence (monotone).

•Les points candidats sont insérés dans la liste dans l'ordre croissant de leurs scores.

-Les points qui ont des scores inférieurs sont susceptibles de dominer un grand nombre de points.

•Cet algorithme a un comportement progressif.

-le triage préalable assure qu'un point p dominant un autre p' doit être visité avant p', par conséquent nous pouvons immédiatement produire les points insérés à la liste comme points skylines.

•Il doit balayer le fichier de données entier pour renvoyer un skyline complet, car chaque point skyline Peut avoir un score très grand et apparaître à la fin de la liste assortie (par exemple un point '**a**' a le 3eme plus grand score pour la fonction de préférence $f = 0.fautes\ Commises + 1.ballons\ perdus$). [4]

Point	score
K	**1**
I	**2**
M	**2**
N	**3**
H	**3**
L	**4**
F	**5**
G	**6**
D	**7**
c	**8**
a	**9**
e	**10**
b	**10**

Table II.2 : Les approches de SFS

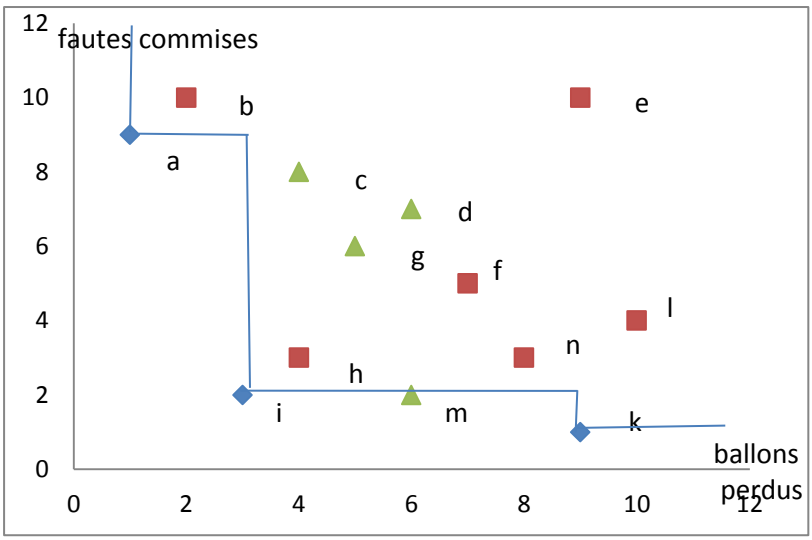


Figure II.3 :L'algorithme SFS.

II.6.3 Les algorithmes avec index :

II.6.3.1 Bitmap :

•Code bitmap (codage dans les carte binaire) toute l'information requise à décider si un point est un point Skyline.

•Un point de repères $\boldsymbol{p} = (p_1, p_2, \dots, p_d)$ est tracé à m-bits vecteur

- **d** : nombre de dimensions
- **m** : nombre total des valeurs distinctes au-dessus de tous dimensions
- $\boldsymbol{k_i}$: nombre total des valeurs distinctes sur la i-eme dimension

Si **pi** est le j_i *eme* plus petit nombre sur le i-eme axe alors il est représenté par $\boldsymbol{k_i}$ bits où l'extrême gauche ($\boldsymbol{k_i}$-j_i +1) bits est 1, et les autres 0.

Id	Coordonnés	Représentation bitmap
a	(1,9)	(1111111111,1100000000)
b	(2,10)	(1111111110,1000000000)
c	(4,8)	(1111111000,1110000000)
d	(6,7)	(1111100000,1111000000)
e	(9,10)	(1100000000,1000000000)
f	(7,5)	(1111000000,1111110000)
g	(5,6)	(1111110000,1111100000)
h	(4,3)	(1111111000,1111111100)
i	(3,2)	(1111111100,1111111110)
k	(9,1)	(1100000000,1111111111)
l	(10,4)	(1000000000,1111111000)
m	(6,2)	(1111100000,1111111110)
n	(8,3)	(1110000000,1111111100)

Table II.3 : Les approches de Bitmap.

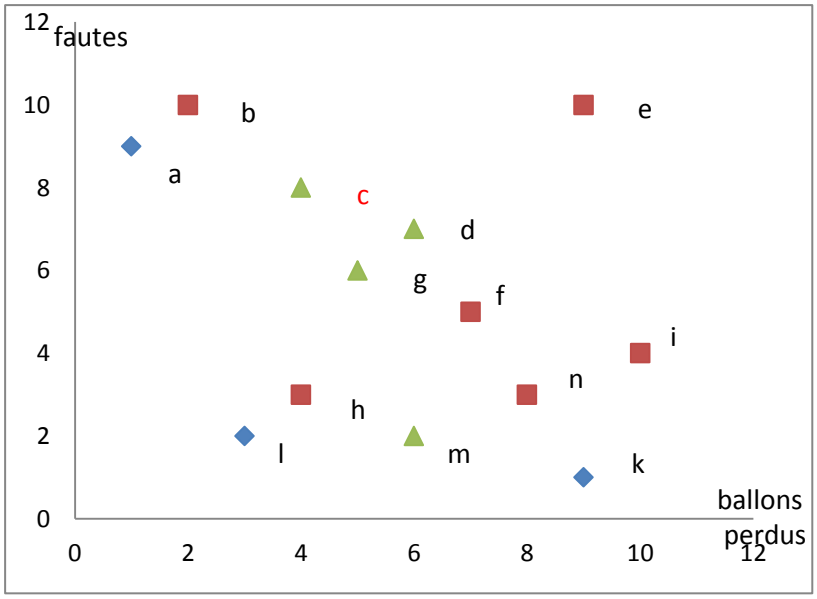


Figure II.4 L'algorithme Bitmap.

-Comment algorithme décide si un point appartient a l'ensemble des skylines ?

Par exemple : c (1111111000, 1110000000)

Id	coordonnés	Représentation bitmap
a	(1,9)	(1111111111,1100000000)
b	(2,20)	(1111111110,1000000000)
c	(4,8)	(1111111000,1110000000)
d	(6,7)	(1111100000,1111000000)
e	(9,10)	(1100000000,1000000000)
f	(7,5)	(1111000000,1111110000)
g	(5,6)	(1111110000,1111100000)
h	(4,3)	(1111111000,1111111100)
i	(3,2)	(1111111100,1111111110)
k	(9,1)	(1100000000,1111111111)
l	(10,4)	(1000000000,1111111000)
m	(6,2)	(1111100000,1111111110)
n	(8,3)	(1110000000,1111111100)

Table II.4 : Exemple de Bitmap.

La création des bits-strings Cx et Cy par la juxtaposition des bits correspondants :

-Le 1er résultat (Cx&Cy= 001000011000) indique les points qui dominent c

abcdefghiklmn

•S'il y a plus qu'un seul 1, le point considéré n'est pas un Skyline

•L'efficacité se fonde sur la vitesse au niveau du bit des opérations

•Les résultats premiers sont rapidement retournés

•En outre le calcul des skylines entier est cher.

- Pour chaque point inséré, l'algorithme doit rechercher les bitmap de chaque point afin d'obtenir la juxtaposition.

•La consommation de l'espace peut être prohibitive, si le nombre des valeurs distinctes est grand

•L'algorithme n'est pas approprié aux ensembles de données dynamiques :

-les insertions peuvent changer les rangs des valeurs attribuées. [9]

II.6.3.2 Index :

•L'algorithme Index organise un ensemble de points **d**-dimensionnels en d listes.

•Un point $\boldsymbol{p} = (p_1, p_2, \dots, p_d)$ est assigné à la **i**-eme liste ($1 \leq i \leq d$), si et seulement si son pi du même rang sur le i-eme axe est le minimum parmi toutes les dimensions

List 1		List 2	
a(1,9)	Minc=1	k(9,1)	Minc=1
b(2,10)	Minc=2	i(3,2), m(6,2)	Minc=2
c(4,8)	Minc=4	h(4,3), n(8,3)	Minc=3
g(5,6)	Minc=5	l(10,4)	Minc=4
d(6,7)	Minc=6	f(7,5)	Minc=5
e(9,10)	Minc=9		

Table II.5 : Les approches Index.

•Cet algorithme Groupe dans la **i**-eme liste les points qui ont la même **i**-eme coordonnée (c-à-d. Minc).

-regroupement dans la liste 2 :{k}, {i, m}, {h, n}, {l}, {f}.

•Ensuite il Charge la première série et manipule chaque liste et celle avec le minimum Minc

⇨

List 1		List 2	
a(1,9)	Minc=1	k(9,1)	Minc=1
b(2,10)	Minc=2	i(3,2), m(6,2)	Minc=2
c(4,8)	Minc=4	h(4,3), n(8,3)	Minc=3
g(5,6)	Minc=5	l(10,4)	Minc=4
d(6,7)	Minc=6	f(7,5)	Minc=5
e(9,10)	Minc=9		

Table II.6 : première étape de l'algorithme index.

•Ici {**a**} et {**k**} ayant Minc=1 identique, donc l'algorithme manipule le groupe de la liste 1.

- **a** est un point ajouté à la liste des skylines

•le Minc de {**b**} =2, donc l'algorithme manipule le groupe {**k**} de la liste 2

-k → non dominé par **a**, donc k → s'est inséré dans la liste des skylines.

• le groupe manipulé suivant est {**b**} de la liste 1

List 1		List 2	
a(1,9)	Minc=1	k(9,1)	Minc=1
b(2,10)	Minc=2	i(3,2), m(6,2)	Minc=2
c(4,8)	Minc=4	h(4,3), n(8,3)	Minc=3
g(5,6)	Minc=5	l(10,4)	Minc=4
d(6,7)	Minc=6	f(7,5)	Minc=5
e(9,10)	Minc=9		

Table II.7 : deuxième étape de l'algorithme index.

Mais **b** est dominé par a (déjà dans la liste des skylines)

•Il procède au groupe {**i**, **m**}

List 1		List 2	
a(1,9)	Minc=1	k(9,1)	Minc=1
b(2,10)	Minc=2	i(3,2), m(6,2)	Minc=2
c(4,8)	Minc=4	h(4,3), n(8,3)	Minc=3
g(5,6)	Minc=5	l(10,4)	Minc=4
d(6,7)	Minc=6	f(7,5)	Minc=5
e(9,10)	Minc=9		

Table II.8 : troisième étape de l'algorithme index.

i domine **m** → **i** est ajouté au liste des points Skylines

•L'algorithme peut rapidement renvoyer des points skylines en haut des listes.

•L'ordre que les points skylines sont retournés est fixe, ne soutenant pas les préférences définies par l'utilisateur.

•Les listes calculées pour des dimensions de **d** ne peuvent pas être employées à rechercher les points skylines sur n'importe quel sous-ensemble des dimensions.

- pour des requêtes sur des dimensions arbitraires, un nombre exponentiel de listes doit être pré – calculé. [9]

II.6.3.3 Nearest Neighbor (NN) :

•NN emploie les résultats de la recherche de plus proche voisin pour diviser l'univers de données périodiquement.

- Il trouve le point avec la distance minimum (mindist) depuis le début des axes (point **i**, mindist=5).
- Tous les points dans la région de dominance de **i** peuvent être taillés.

•Les séparations (1-3), (1-2) sont insérées dans une liste to-do

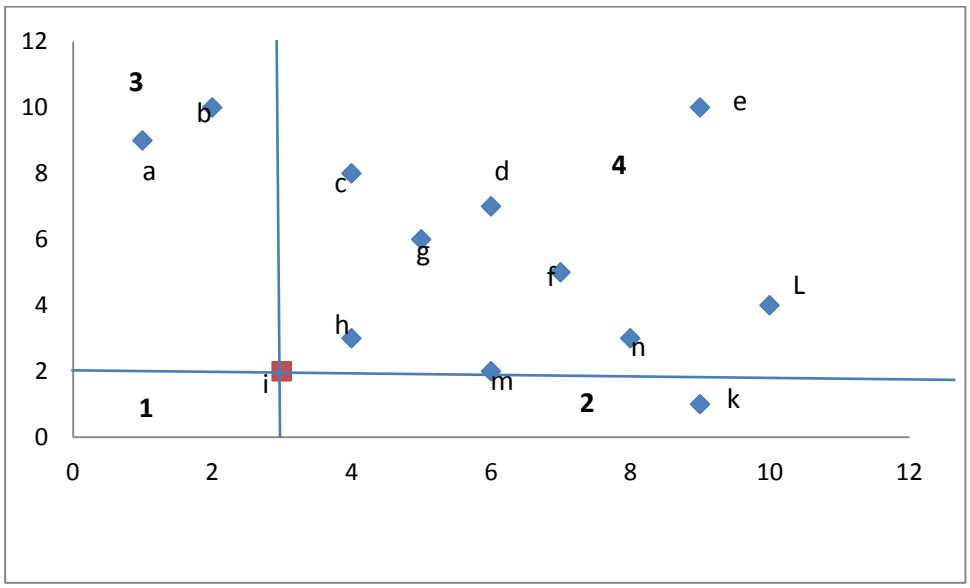


Figure II.5 l'algorithme NN découverte du point i.

•Tandis que la liste to-do n'est pas vide, NN enlève une des séparations de la liste et répète périodiquement le même processus.

-Le Point **a** cause l'insertion des séparations (5-7), (5-6).

-quand une séparation est vide, il n'est pas subdivisé plus loin

-le point **k** cause l'insertion des séparations (7-8), (7-9). [9]

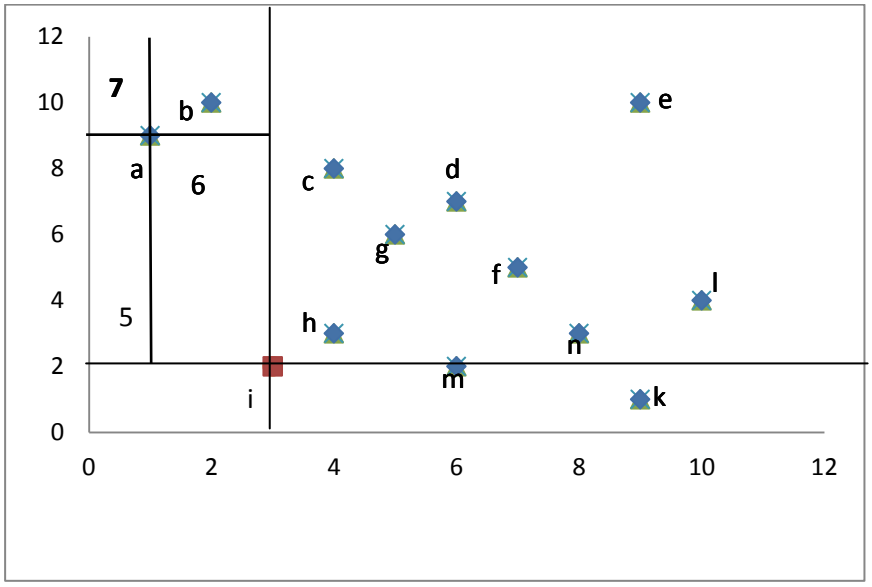


Figure II.6 : l'algorithme NN découverte du point a.

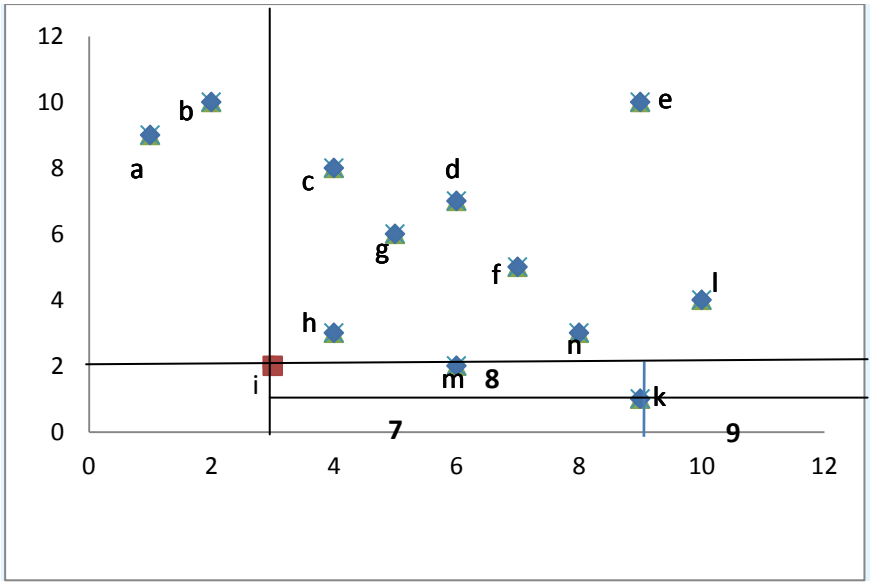
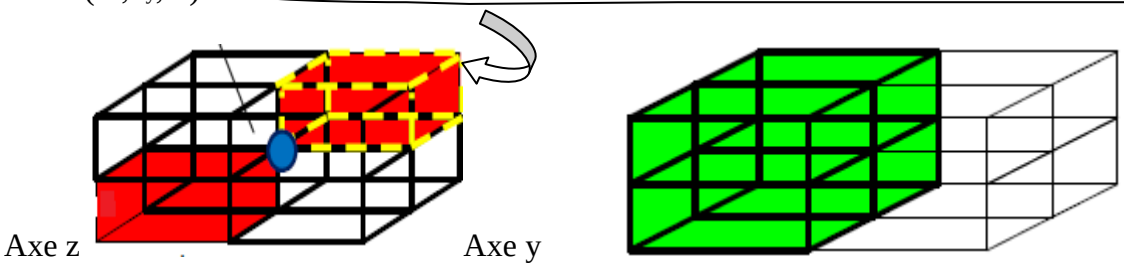


Figure II.7 :l'algorithme NN découverte du point k.

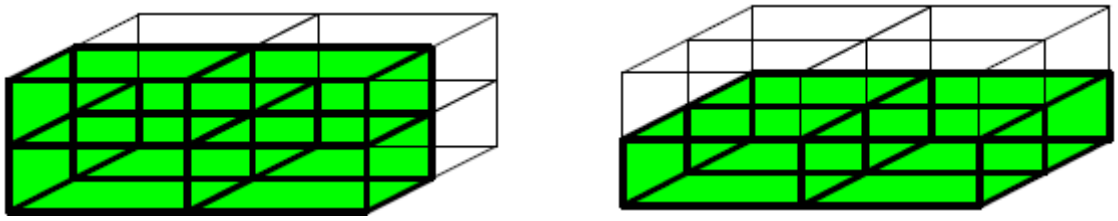
Pour la dimensionnalité **d**, chaque point Skyline mène à **d** plus de requêtes

(n_x,n_y,n_z) dominé par **n** ne sera pas recherché, tout le reste sera recherché deux fois



1er point skyline

1 ère requête $[0,n_x)[0,\infty)[0,\infty)$

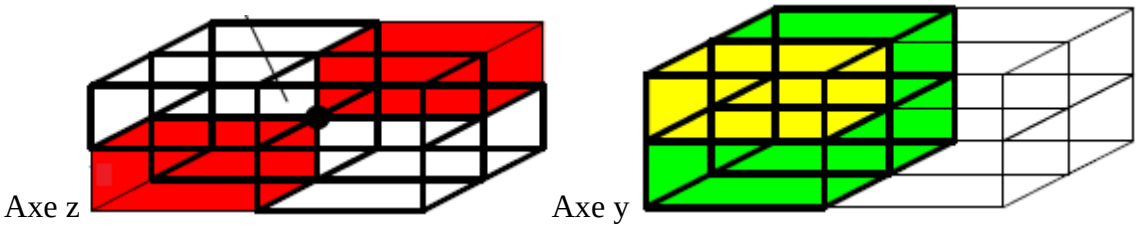


2eme requête skyline $[0, n_x)[0,\infty)[0,\infty)$

3 eme requête skyline $[0, n_x)[0,\infty)[0,\infty)$

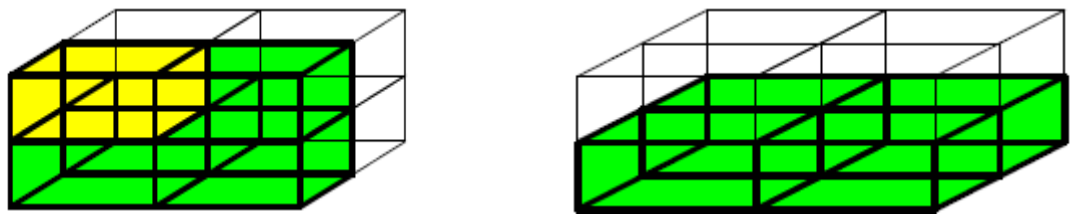
•Ayant besoin pour l'élimination en double, si la dimensionnalité d > 2

(n_x,n_y,n_z)



1er point skyline

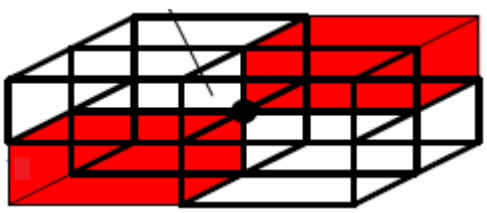
1 ère requête skyline $[0,n_x)[0,\infty)[0,\infty)$



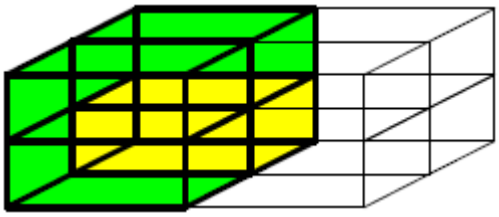
2 eme requête skyline $[0, \infty)[0,n_y)[0,\infty)$

3 eme requête skyline $[0, \infty)[0, \infty)[0,n_z)$

(n_x,n_y,n_z)



Axe z

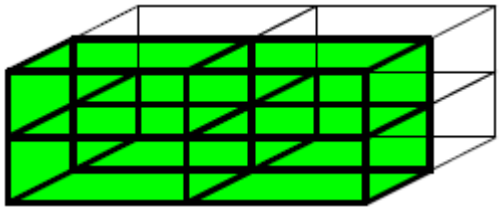


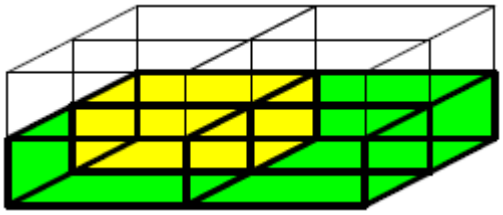
Axe y

Axe x

1er point skyline

1 ère requête skyline [0,n_x)[0,∞)[0,∞)

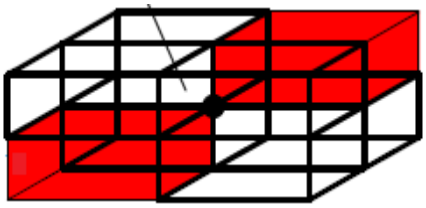




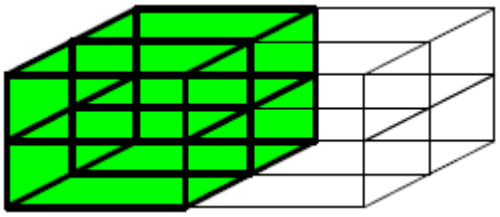
2 eme requête skyline [0, ∞)[0,n_y)[0,∞)

3 eme requête skyline [0, ∞)[0,n_y)[0,∞)

(n_x,n_y,n_z)



Axe z

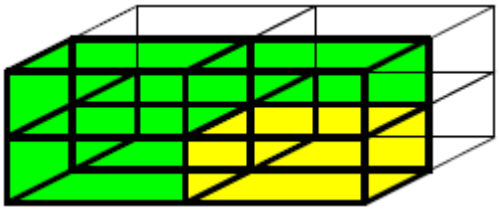


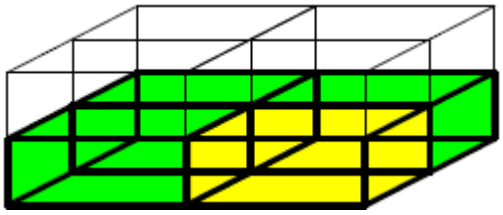
Axe y

Axe x

1er point skyline

1ere requête skyline [0,n_x)[0,∞)[0,∞)





2eme requête skyline [0,n_x)[0,∞)[0,∞)

3eme requête skyline [0,n_x)[0,∞)[0,∞)

Figure II.8 : partitionnement de NN pour trois dimensions.

-La manipulation des requêtes vides gaspille beaucoup de temps.

-Grandes conditions de mémoire centrale dans le pire des cas il pourrait être l'ordre de l'ensemble de données ! [9]

II.6.3.4 Branch and Bound Skyline (BBS) :

Le principe de cette méthode est de découper (branch) l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global (principe de séparation). L'algorithme de recherche forme ainsi un arbre dont chaque nœud représente une partie de l'espace. Ensuite chaque nœud est évalué de façon à déterminer sa borne (bound) inferieur (inferieur dans le cas d'une minimisation, supérieur dans le cas d'une maximisation) en fonction d'un critère d'évaluation. Si cette borne n'est pas meilleure que la solution courante alors la recherche sur ce nœud est stoppée, sinon la séparation continue. [7]

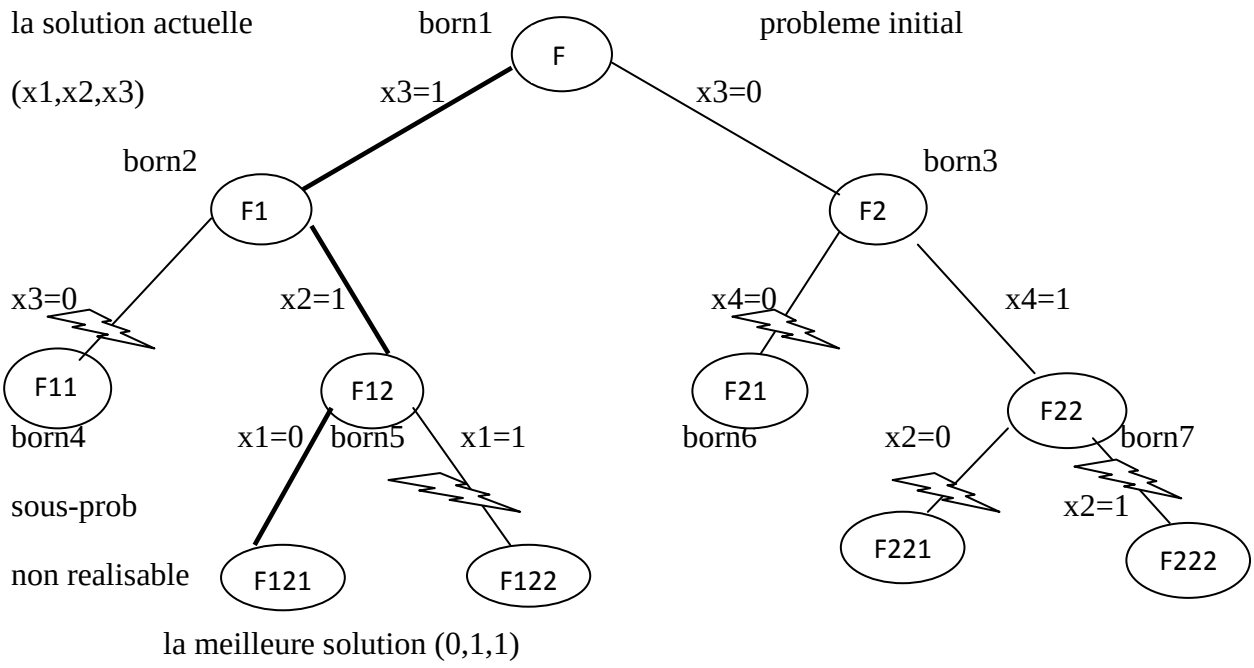


Figure II.9 : illustration de la construction de l'arbre de recherche d'un algorithme de type B&B Skyline

• Critique :

L'efficacité de cette technique dépend essentiellement du choix des critères de séparation et d'évaluation. Un bon choix permettra à l'algorithme de réduire l'arbre de recherche en évitant la construction de certaines branches dans le pire des cas, un mauvais choix peut amener l'algorithme à des espaces de petites dimensions.

Cette méthode est essentiellement utilisée sur des problèmes discrets. Couplée avec une méthode d'analyse d'intervalle, elle peut être également utilisée dans le cadre de problèmes continus avec l'avantage de ne pas exiger la continuité de la fonction et d'éviter que la propagation d'erreurs numériques empêche la validation d'une solution. [7]

Chapitre III :

L’algorithme BBS

III.1 Introduction :

La méthode de recherche des skylines par séparation et évaluation (Branch & Bound) est une méthode générique de résolution exacte de problèmes d'optimisation, C'est une méthode constructive basée sur une recherche arborescente.

Elle consiste à énumérer les solutions possibles d'une manière intelligente, en ce sens que, en utilisant certaines heuristiques, cette technique arrive à éliminer des solutions (ou nœud ou rectangle) partielles qui ne donnent pas des résultats satisfaisant

De ce fait, on arrive souvent à obtenir la solution recherchée en temps raisonnable et dans la pire des cas, on retombe toujours sur l'énumération explicite de toutes les solutions du problème.

Remarque :

Notre objectif est d'implémenter l'algorithme branch and bound utilisant une structure d'arborescence R-Tree **pré établi.**

III.2 Principe de l'algorithme Branch and Bound « BBS » :

L'idée de l'algorithme de séparation et évaluation « BBS » est de construire un arbre appelé « R-Tree » qui exprime implicitement toute les solutions du problème, en se basant sur deux concepts : la séparation qui consiste à diviser l'espace des solutions en sous-problèmes pour les optimiser chacun individuellement, et l'évaluation basée sur la notion de bornes, calculées et comparées (relation de dominance dans le cas multi objectif) à une solution déjà trouvée permet d'éliminer des branches de l'arborescence sans les parcourir. Le problème initial est la racine correspondant à une solution partielle vide, les feuilles correspondent à des solutions réalisables. Les autres nœuds correspondent à des solutions partielles. Une borne est attribuée à chaque nœud. Durant l'exploration, si une branche est moins bonne que (dominée par) la solution courante, on arrête de l'explorer. Si une solution est meilleure que la solution courante, elle la remplace. Tout sous-problème non réalisable sera coupé. [6]

III.3 L'arbre R-Tree :

Le R-Tree représente une hiérarchie d'intervalles (ou boîtes) à N dimensions imbriquées. Dans l'organisation d'un R-Tree, on peut dissocier :

-les nœuds internes

-les feuilles

Le modèle d'indexation du R-Tree possède également les propriétés suivantes :

– Chaque nœud contient entre m et M entrées (tailles des tableaux avec $2 \leq m \leq M/2$), mis à part le nœud racine. La valeur minimale est là pour prévenir une dégénération de l'arbre et assurer un stockage efficace. Si le nombre d'entrées passe sous le seuil m alors le nœud est supprimé et l'arbre réorganisé.

- Le nœud-racine contient au moins deux entrées.

- L'arbre est équilibré en hauteur, ce qui signifie que toutes les feuilles se situent au même niveau.

La recherche dans un R-Tree s'effectue en testant, à partir de la racine et pour chaque nœud d'un même niveau, l'intersection de ses boîtes avec la boîte de recherche.

En conséquence, tous les nœuds fils qui intersectent la boîte de recherche sont visités. [2]

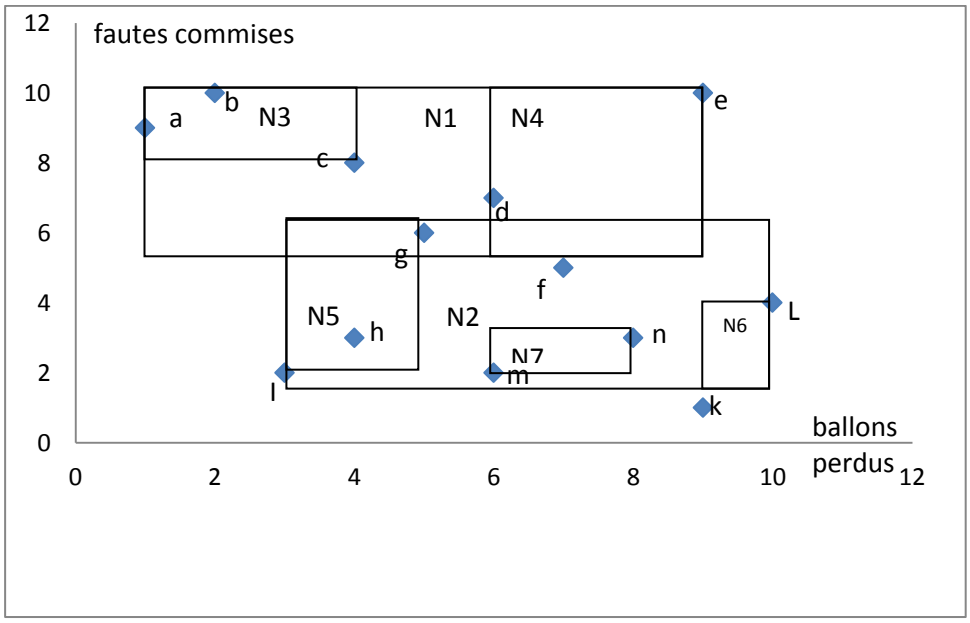
III.3.1 Exemple d'indexation des points dans un R-Tree :

- L'algorithme BBS assure que tous les points sont indexés dans un R-Tree.

  - R-Tree : structures de données d'arbre pour indexer des données multidimensionnelles

  - des points voisins sont groupés en MBR (Minimum Bounding Rectangle)

  - une requête qui n'intersecte pas un MBR, ne peut intersecter aucun de ses points. **[15]**



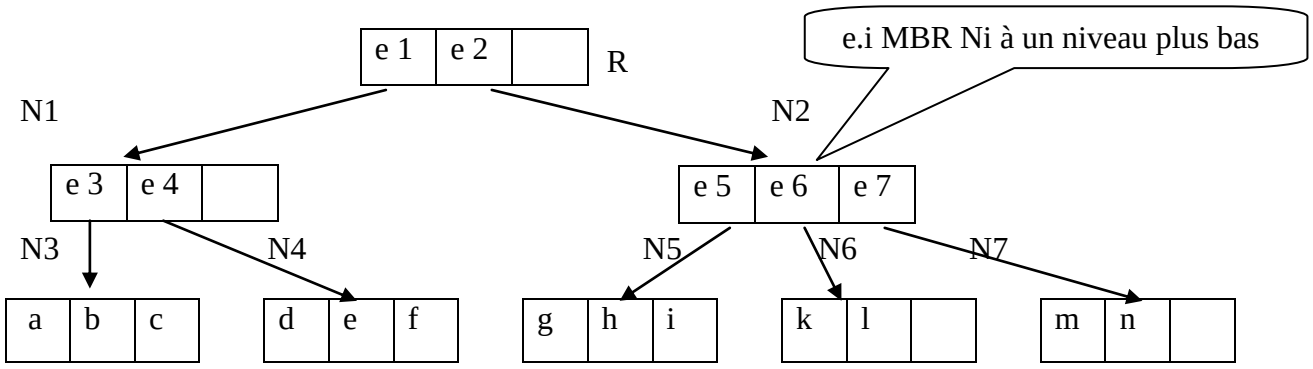


Figure III.1 :Un exemple de R-Tree.

III.4 Technologies utilisées :

III.4.1 Le langage Java :

Parmi tous les langages existants, notre choix a porté sur le langage Java. Celui-ci correspond aux objectifs que nous avons préalablement fixés pour notre projet de fin d'étude. Connu par sa modernité et sa grande performance dans plusieurs domaines de programmation il permettra sans doute de répondre à nos attentes dans le développement de notre mémoire. [15]

Définition :

Java est un langage de programmation et une plate-forme informatique crée par Sun Microsystems en 1995. Il s'agit de la technologie sous-jacente qui permet l'exécution de programmes modernes et performants, notamment dans la construction des utilitaires, des jeux et des applications professionnelles. Java est utilisé sur plus de 850 millions d'ordinateurs de bureau et plus d'un milliard de périphériques dans le monde, dont des périphériques mobiles et des systèmes de diffusion télévisuelle. [14]

III.4.2L'environnement de développement Netbeans :

Les environnements de développement intégrés (EDI), sont des logiciels regroupant un ensemble d'outils nécessaires au développement logiciel dans un (ou plusieurs) langage(s) de programmation.

Parmi tous les environnements de développement existant notre choix a porté sur NetBeans à la faveur de sa rapidité à mettre en place une application web qui est l'un des points forts de ce dernier.

Définition de NetBeans :

C'est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Développement and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web). [13]

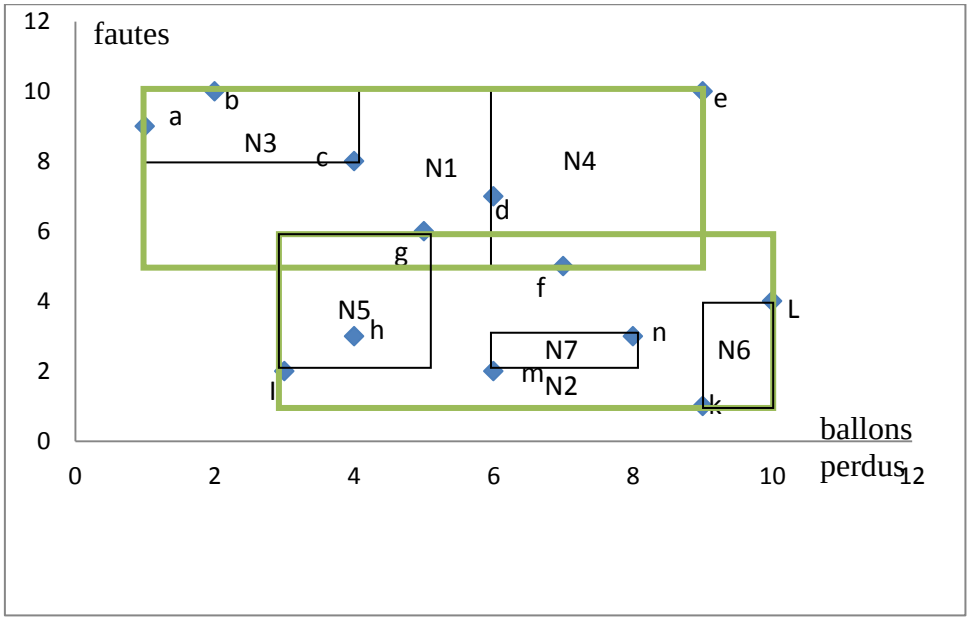
III.5 Conception :

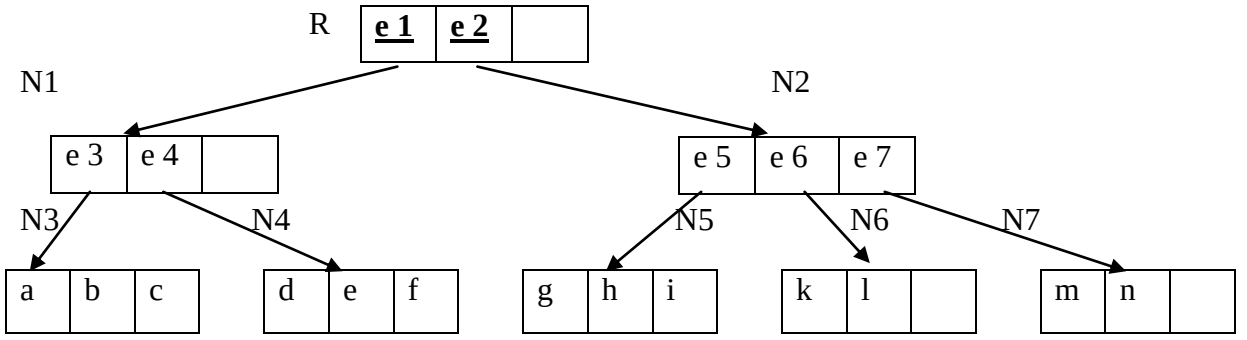
III.5.1 Le pseudo code de l'algorithme BBS :

1. S= Ø // la liste des points skylines.
2. **Insérer** toutes les entrées de la racine de la base R dans la pile P.
3. **Tant que** P n'est pas vide
   - **3.1 Supprimer** la tête e de P
   - **3.2 si** e est dominé par un autre point de S **alors**
     - **3.2.1** supprimer l'entrée e
   - **3.3 Sinon** // e n'est pas dominé
     - **3.3.1 si** e est une entrée intermédiaire **alors**
       - **3.3.1.1 pour** chaque e_i fils de e **faire**
         - **3.3.1.1.1 Si** e_i n'est pas dominé par un point de S **alors**
           - **3.3.1.1.1.1 Insérer** e_i dans la pile p
         - **3.3.1.1.2 Sinon** //e est un point skyline
           - **3.3.1.1.2.1 insérer** e_i dans S
         - **3.3.1.1.3 Fin si**
       - **3.3.1.2 fin pour**
     - **3.3.2 Fin si**
   - **3.4 Fin si**
4. **fin tant que**
5. **fin**

III.5.2 Exemple d'un BBS :

•Chaque entrée de tas maintient la mindist du MBR.

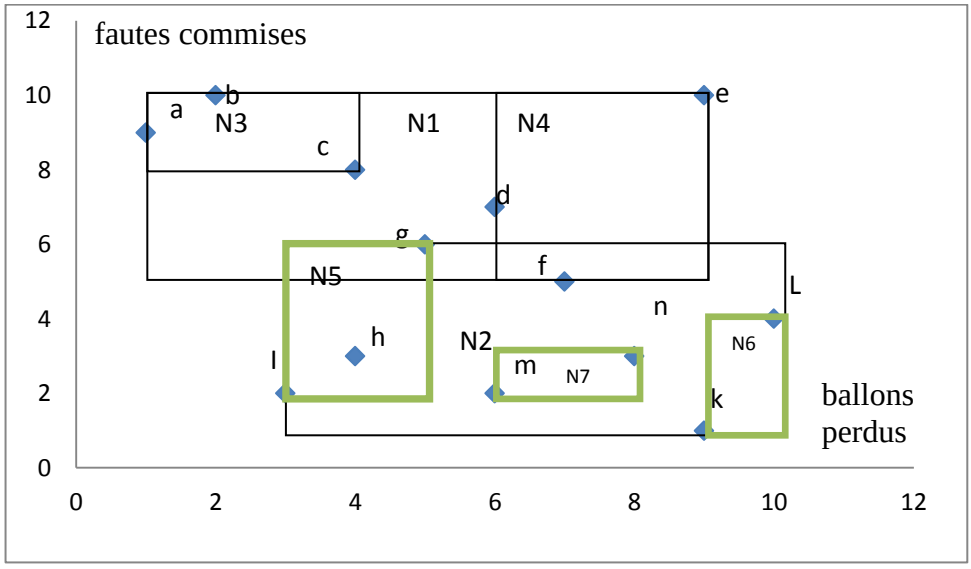


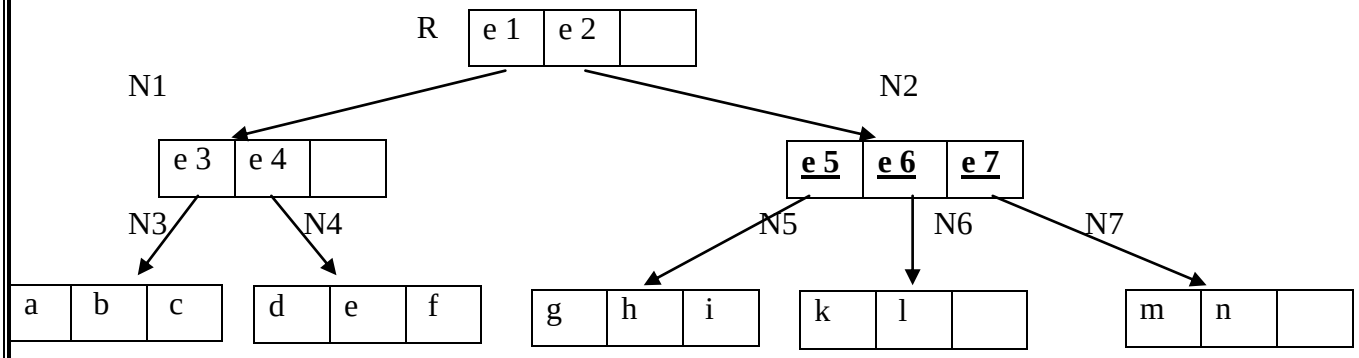


Action	Contenu de tas	S
Acces root	<e2, 4><e1, 6>	Ø

•Entrées de processus en un ordre montant de leur mindist.

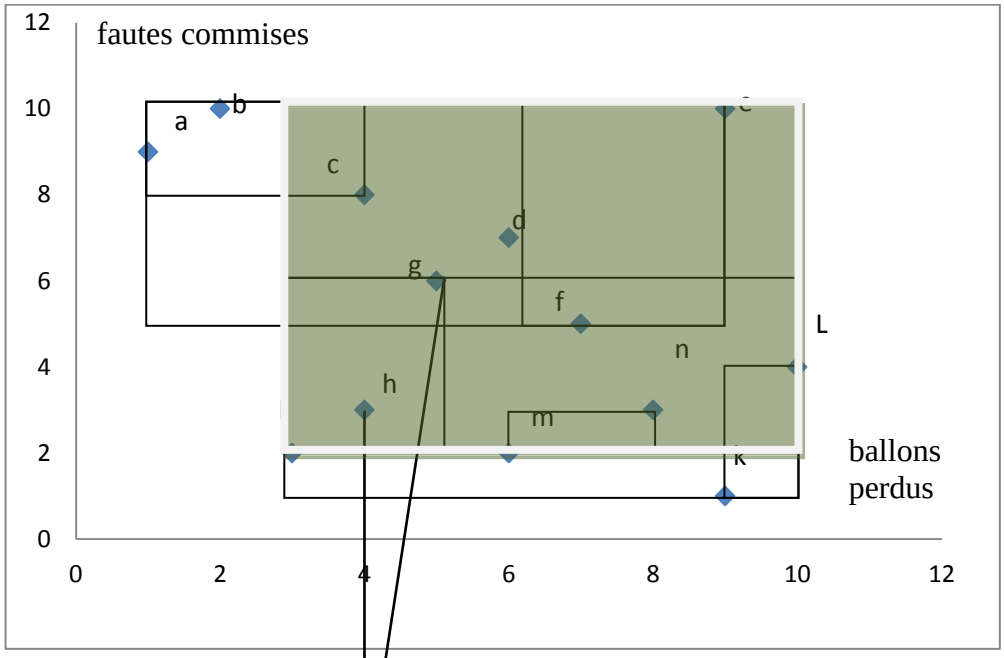
Figure III.2 : Découverte des rectangles N1 et N2.



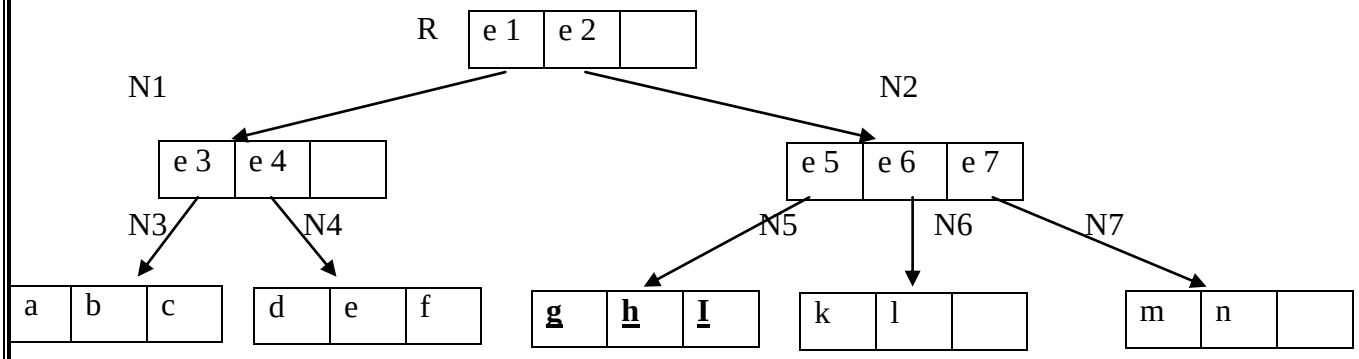


Action	Contenu de tas	S
Acces root	<e2,4><e1,6>	Ø
Expend e2	<e5,5><e1,6><e7,8><e6,10>	Ø

Figure III.3: Découverte des rectangles N5, N6 et N7.



L es points g et h sont dominés



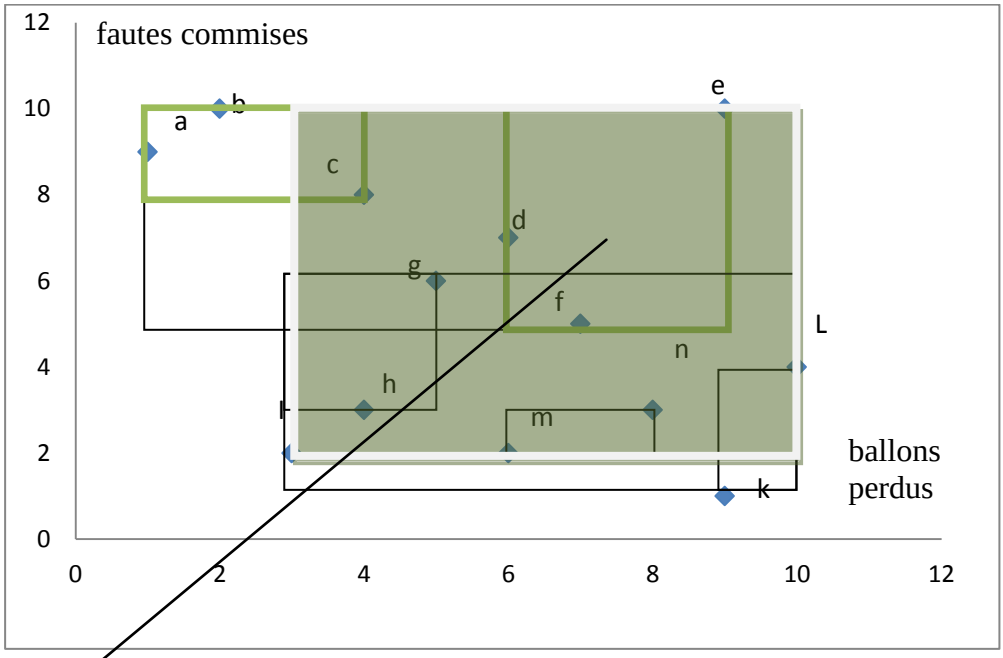
Action	Contenu de tas	S
Acces root	<e2,4><e1,6>	Ø
Expend e2	<e5,5><e1,6><e7,8><e6,10>	Ø
Expend e5	<i,5><e1,6><e7,8><e6,10>	{i}

I 1er point skyline (ix +iy) < mindist e1

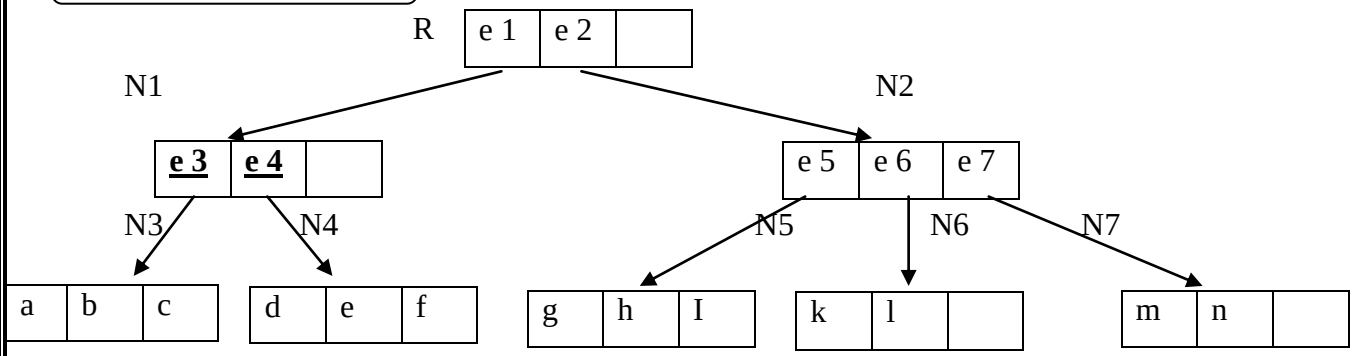
- ici le NN s'arrêterait car (e1)> (ix+iy)

- mais le BBS procédera parce que N1 peut contenir des points skylines.

Figure III.4 : Premiers points skyline « I ».

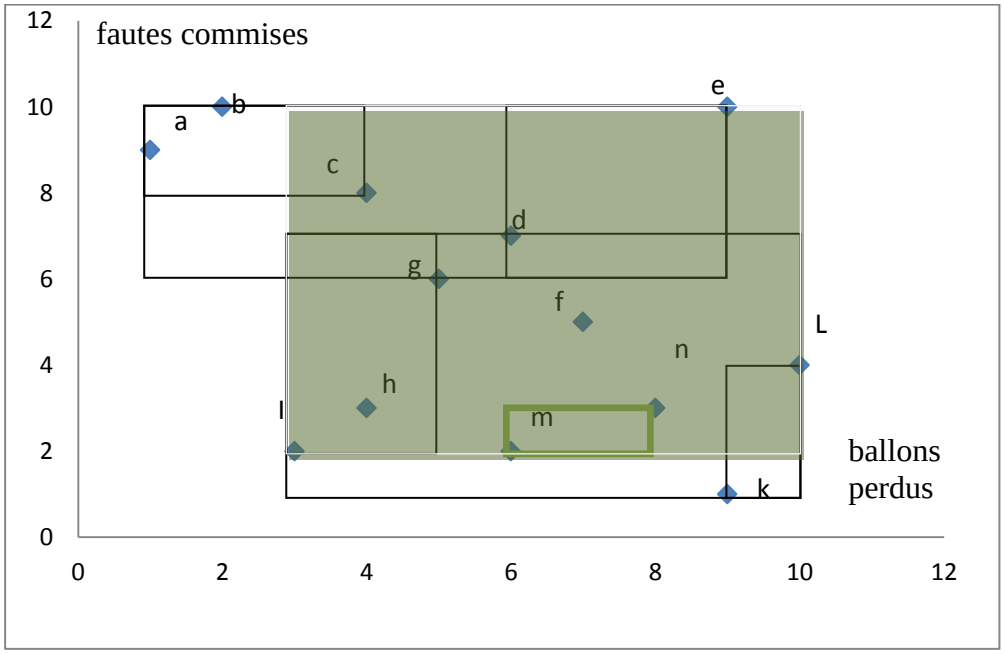


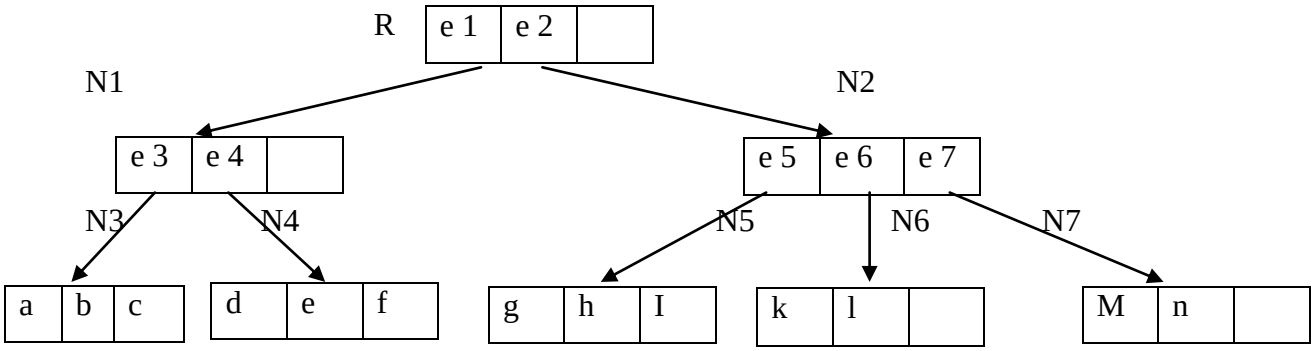
MBR e4 est dominé



Action	Contenu de tas	S
Acces root	<e2,4><e1,6>	Ø
Expend e2	<e5,5><e1,6><e7,8><e6,10>	Ø
Expend e5	<i,5><e1,6><e7,8><e6,10>	{i}
Expand e1	<e7,8><e3,9><e6,10>	{i}

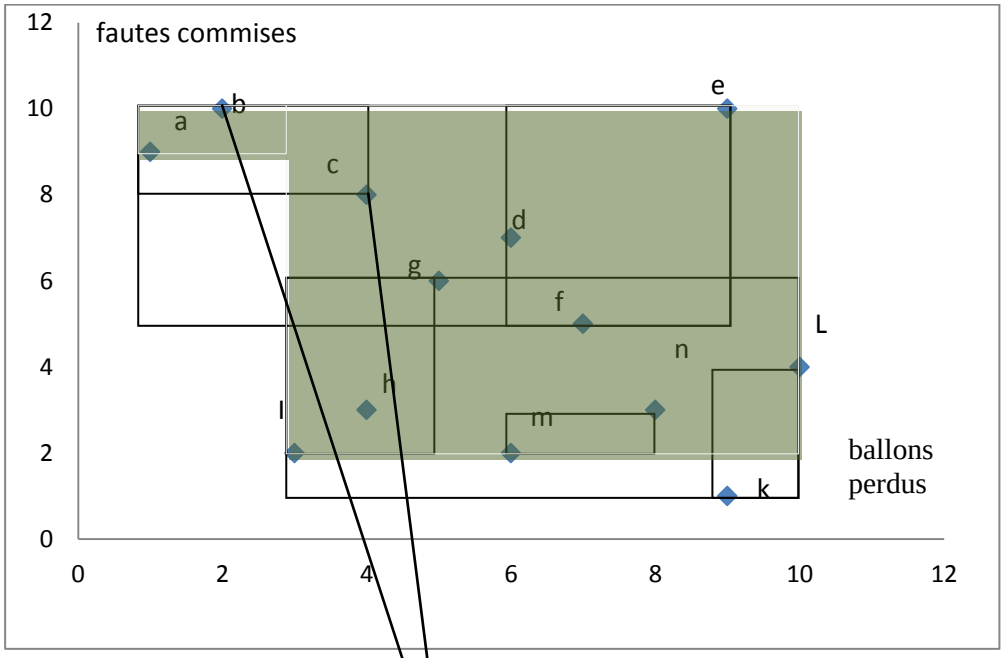
Figure III.5 : Découverte des rectangles N3 et N4.



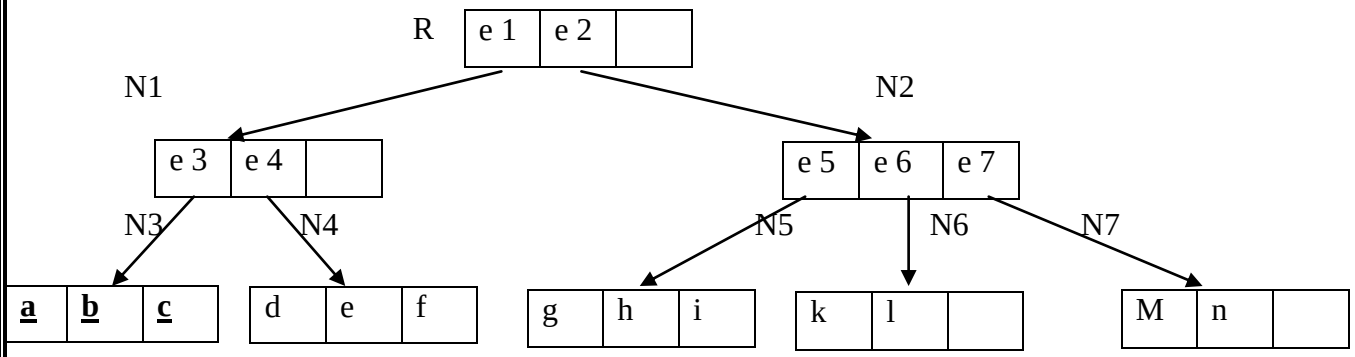


Action	Contenu de tas	S
Acces root	<e2,4><e1,6>	Ø
Expend e2	<e5,5><e1,6><e7,8><e6,10>	Ø
Expend e5	<i,5><e1,6><e7,8><e6,10>	{i}
Expand e1	<e7,8><e3,9><e6,10>	{i}
Remove e7	<e3,9><e6,10>	{i}

Figure III.6 : Elimination des rectangles dominés



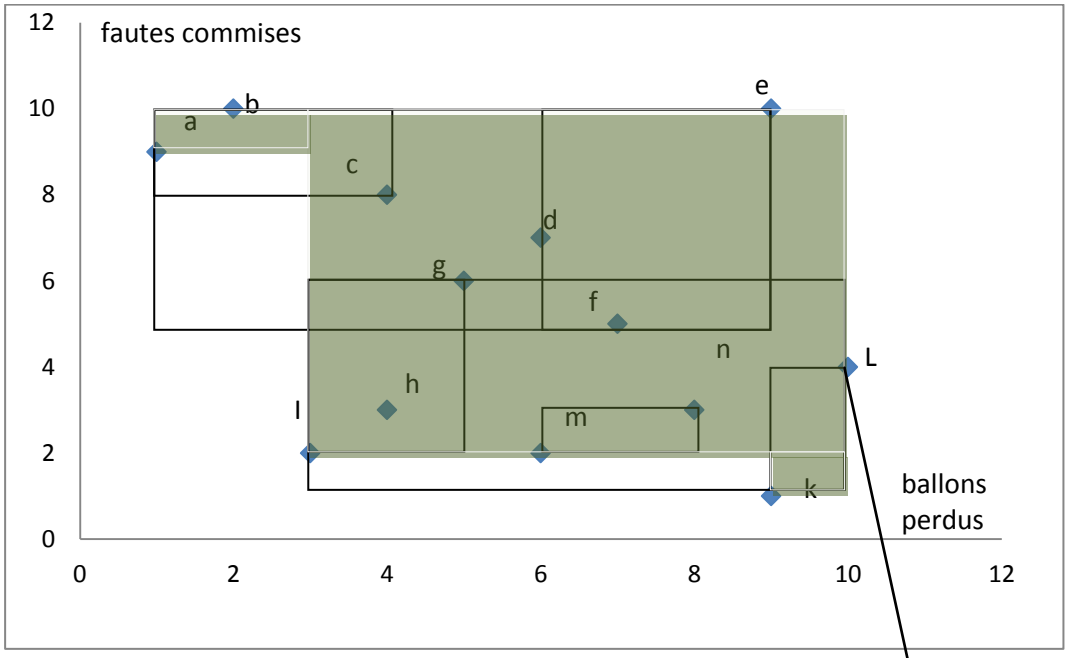
Les points b et c sont dominés



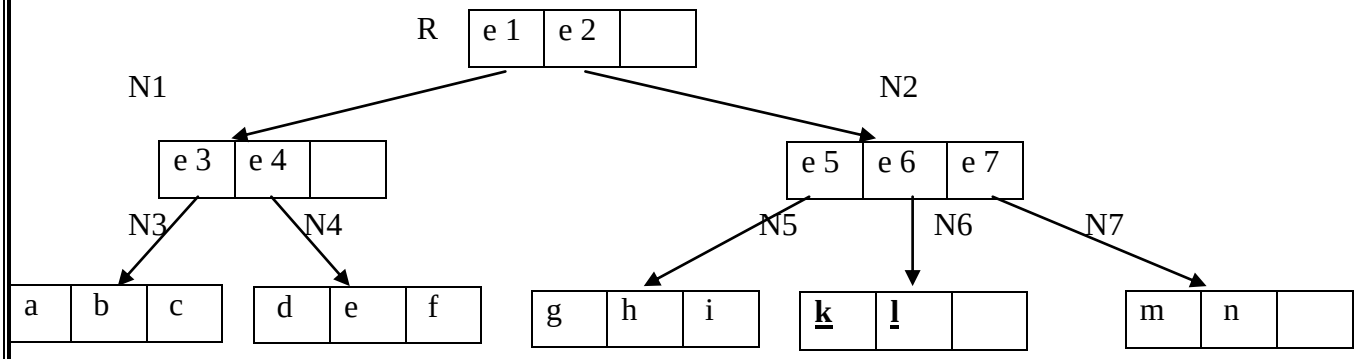
Action	Contenu de tas	S
Acces root	<e2,4><e1,6>	Ø
Expend e2	<e5,5><e1,6><e7,8><e6,10>	Ø
Expend e5	<i,5><e1,6><e7,8><e6,10>	{i}
Expand e1	<e7,8><e3,9><e6,10>	{i}
Remove e7	<e3,9><e6,10>	{i}
Expend e3	<a,10><e6,10>	{i,a}

a : 2eme point skyline (a_x+a_y) <mindist (e6)

Figure III.7 : Deuxième point skyline « a ».



Le point L est dominé



Action	Contenu de tas	S
Acces root	<e2,4><e1,6>	Ø
Expend e2	<e5,5><e1,6><e7,8><e6,10>	Ø
Expend e5	<i,5><e1,6><e7,8><e6,10>	{**i**}
Expand e1	<e7,8><e3,9><e6,10>	{**i**}
Remove e7	<e3,9><e6,10>	{**i**}
Expend e3	<a,10><e6,10>	{**i,a**}
Expand e6	<k,10>	{**i,a,k**}

{i, a, k} liste finale des skylines

Figure III.8 : troisième point skyline « k ».

III.6 Les Fenêtres :

Dans notre projet on a implémenté notre algorithme sur la version JAVA Netbeans 8.0.1

III.6.1 L'exécution :

Nous commençons d'abord par charger les données de la première base en cliquant sur l'item charger donnée1 ou en appuyant directement sur la touche D, après nous cliquons sur le jmenu Application, Puis sur l'item « Branch&Bound S » ou bien directement en appuyant sur Shift+S.

Et pour Exécuter l'algorithme de nouveau avec une autre base il faut d'abord cliquer sur vider, après sur charger donnée2.

Dans ce qui suit maintenant nous présentons les interfaces graphiques de notre prototype :

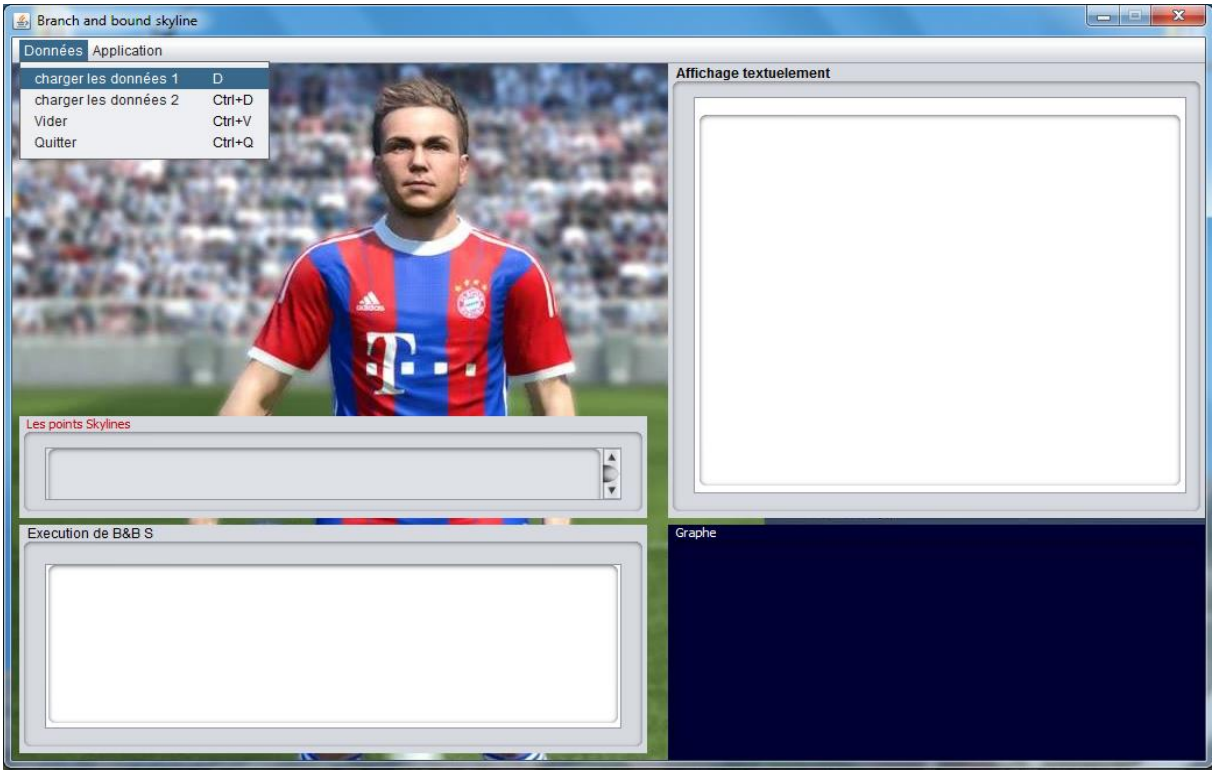


Figure III.9 : L'exécution de l'item « charger les données 1 »

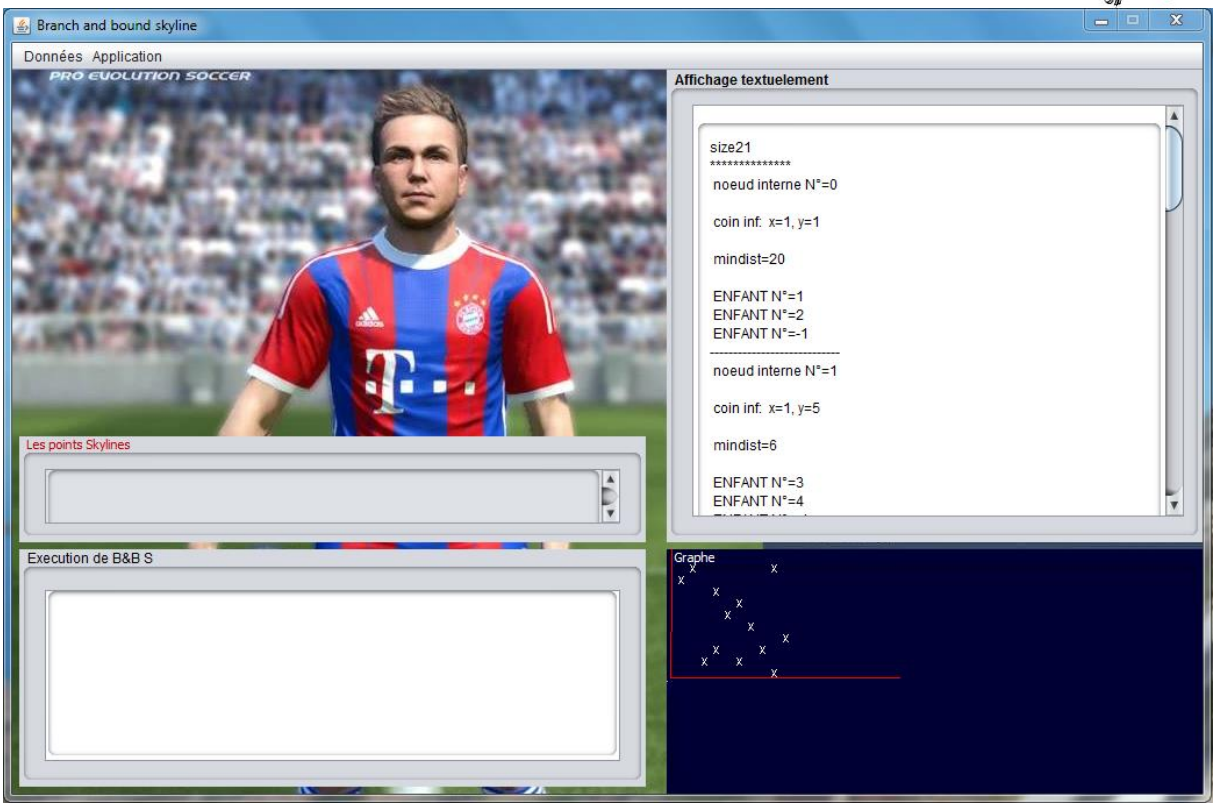


Figure III.10 : Résultat de l'exécution de « charger les données 1 »

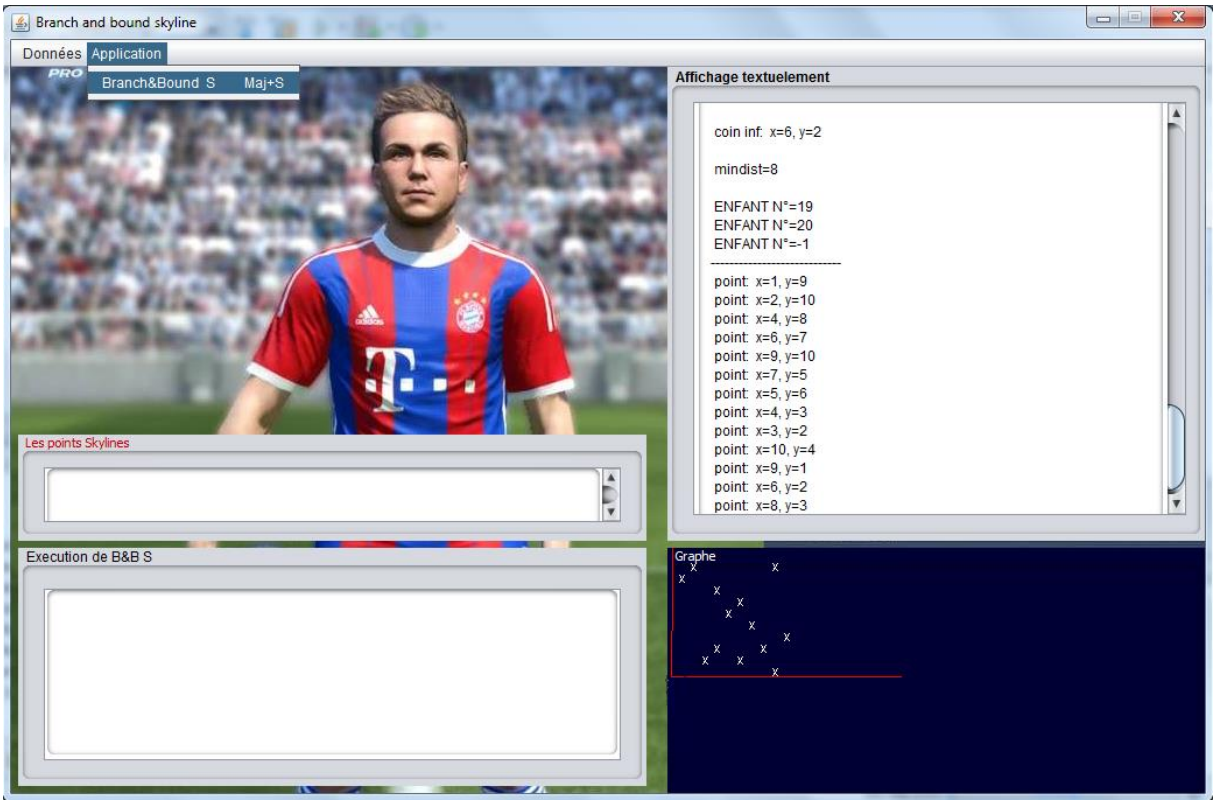


Figure III.11: L'exécution de l'item « Branch&Bound S »

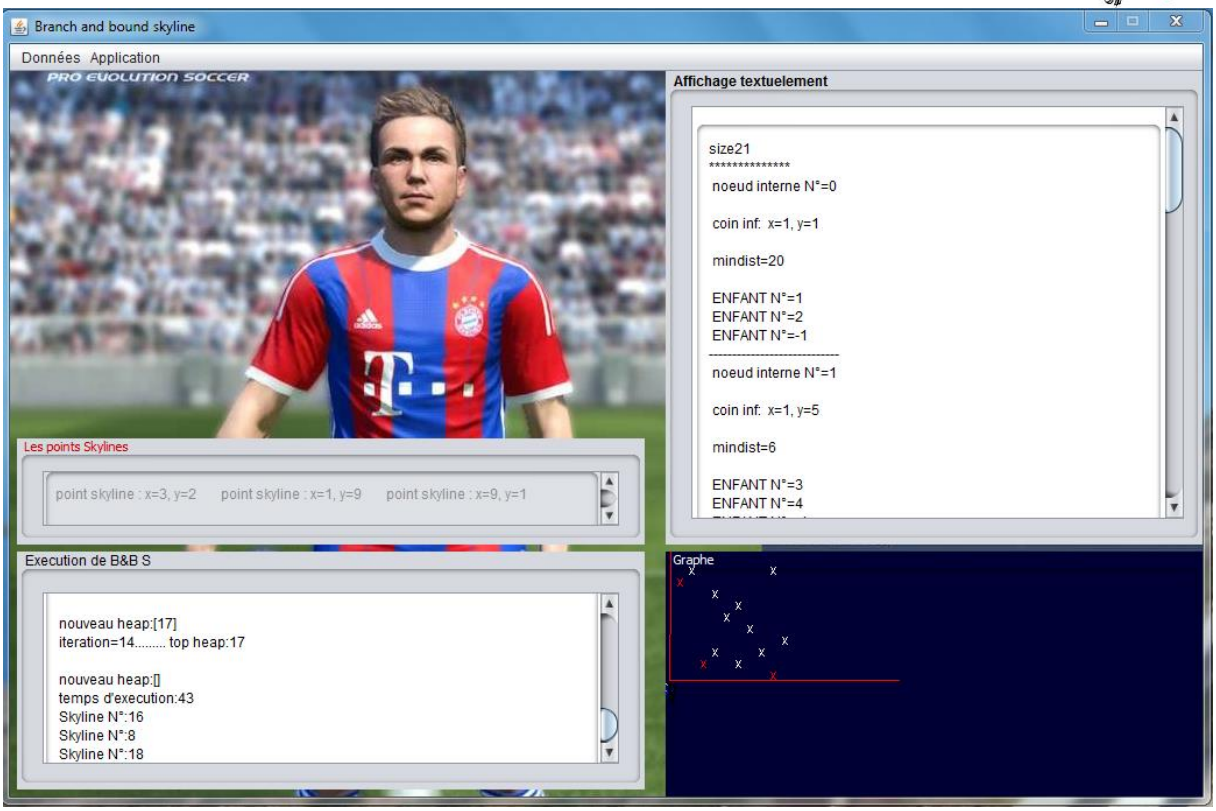


Figure III.12 : Résultat de l'exécution de l'item « Branch&Bound S »

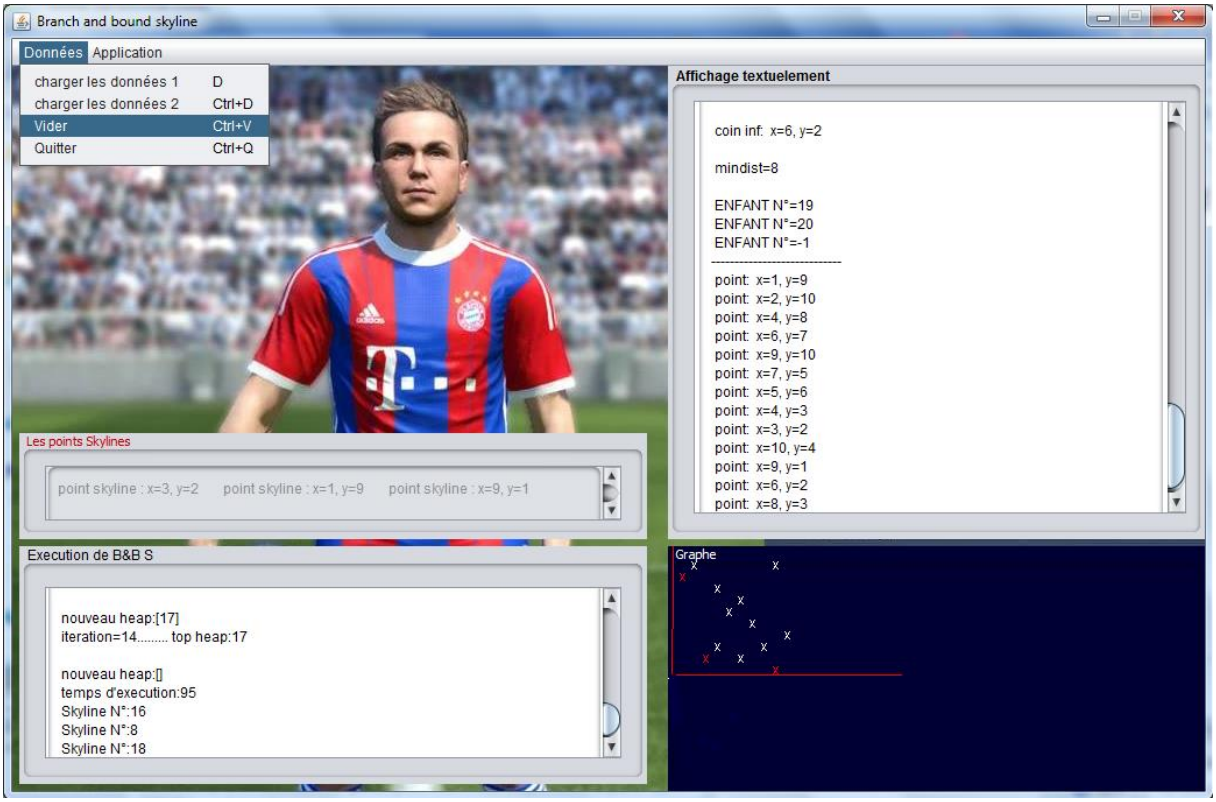


Figure III.13 : L'exécution de l'item « Vider »

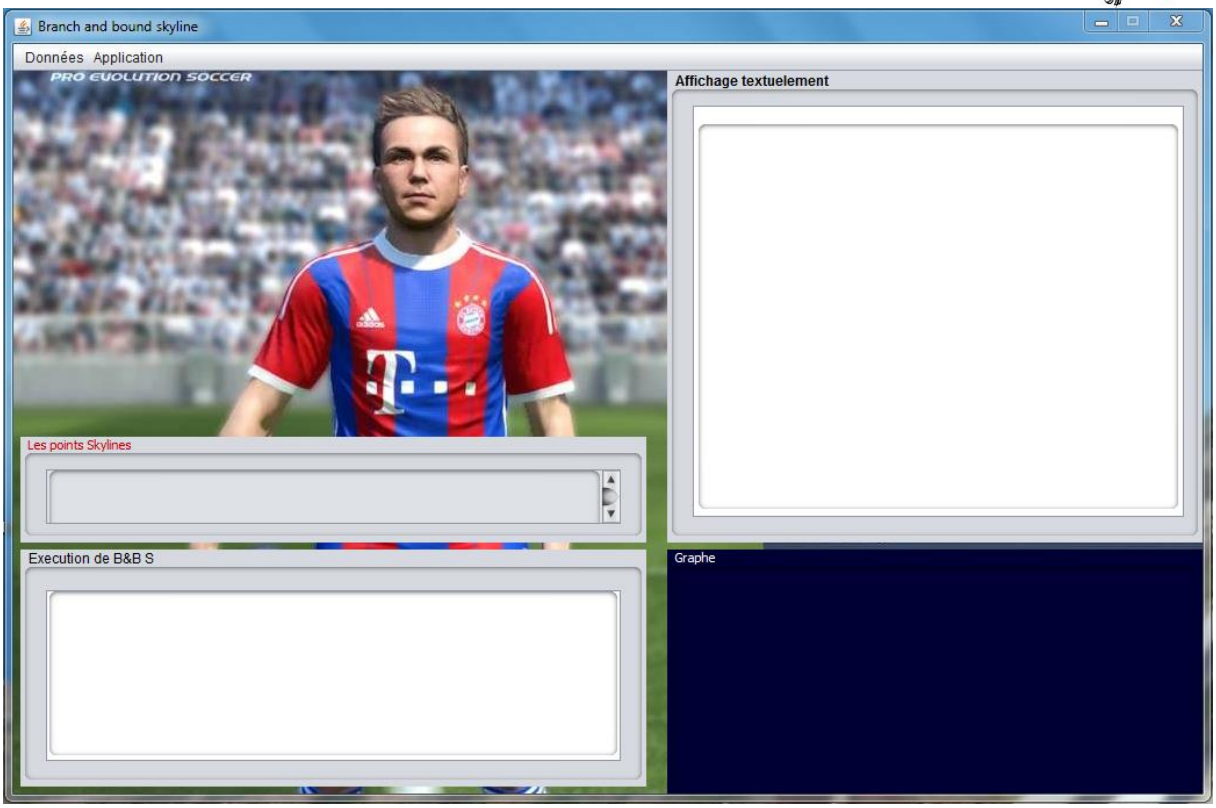


Figure III.14 : Résultat de l'exécution de l'item « Vider »

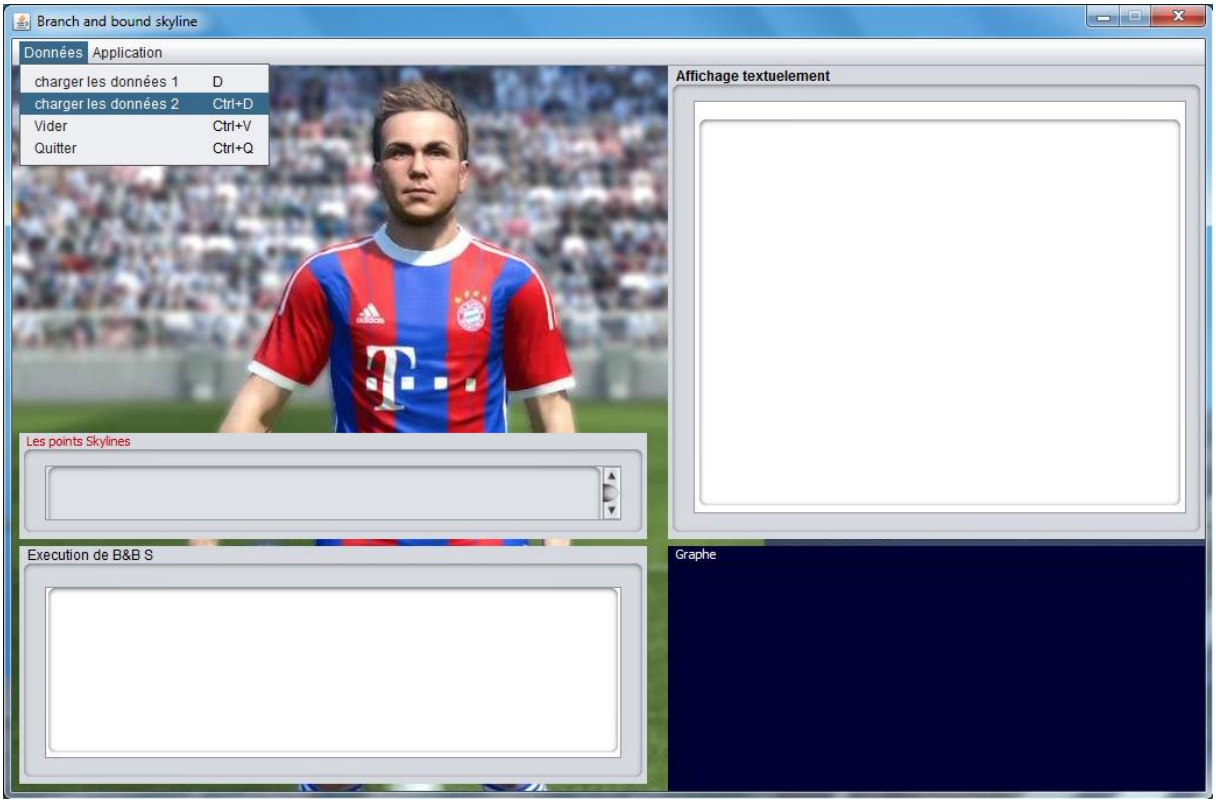


Figure III.15 : L'exécution de l'item « charger les données 2 »

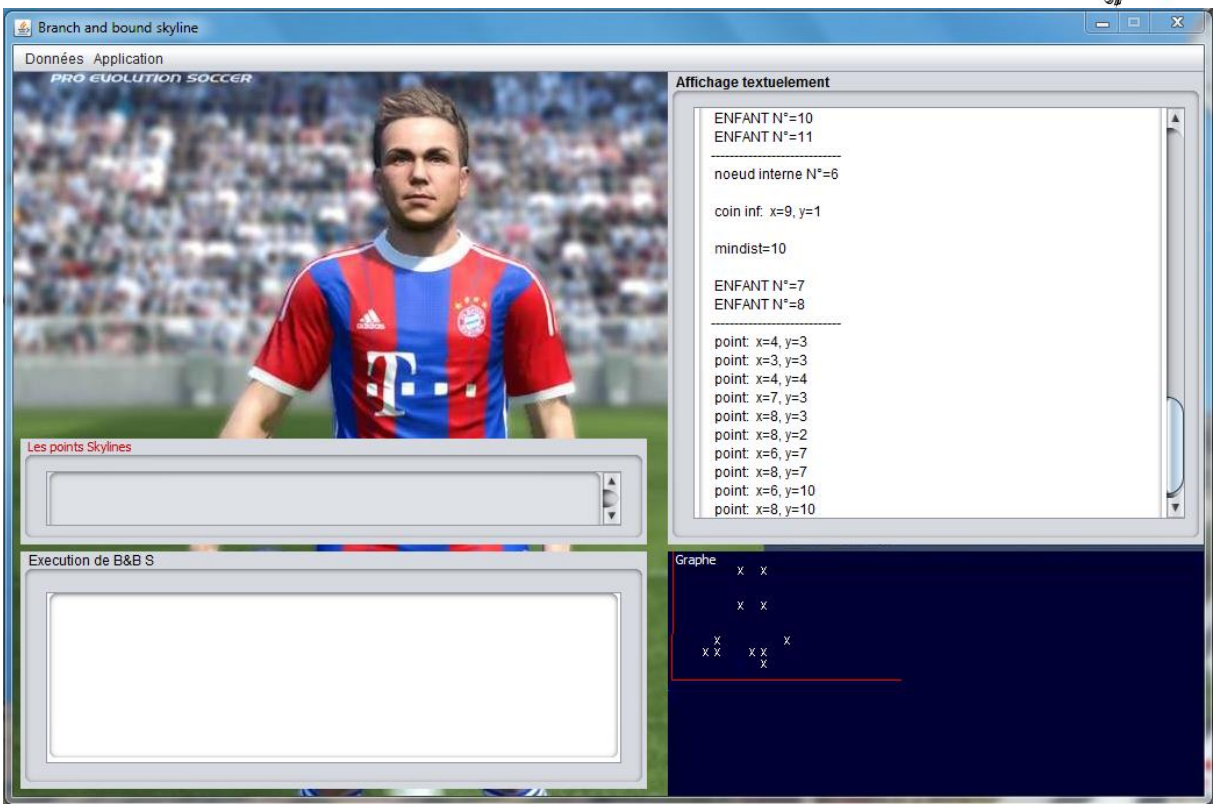


Figure III.16 : Résultat de l'exécution de l'item « Charger les données 2 »

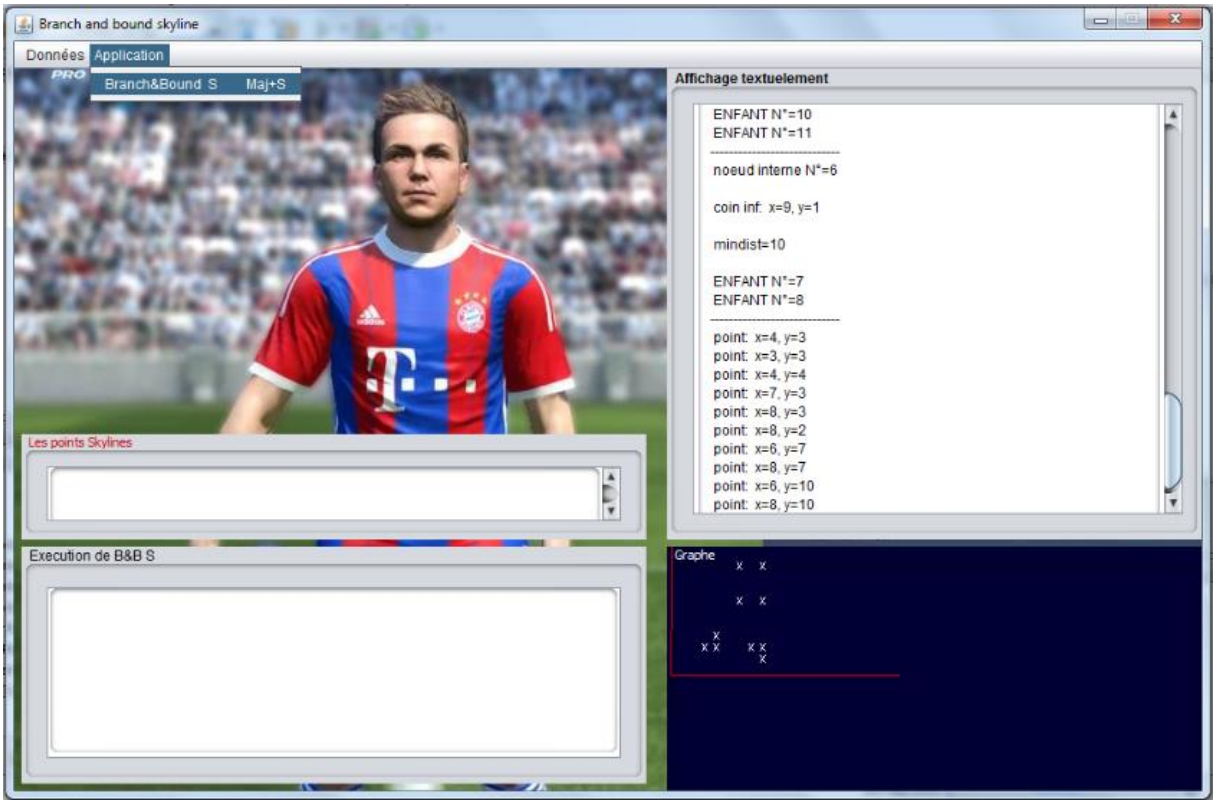


Figure III.17: L'exécution de l'item « Branch&Bound S »

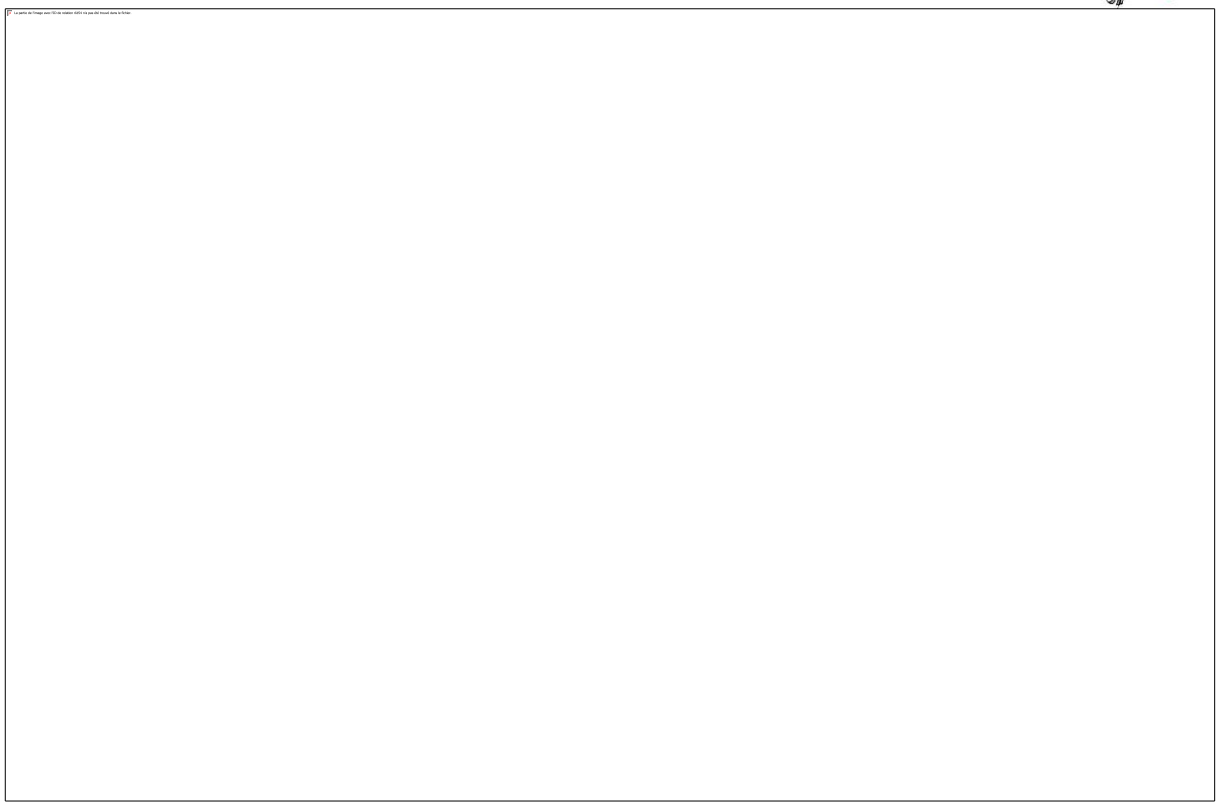


Figure III.18 : Résultat de l'exécution de l'item « Branch&Bound S »

III.6.2 Expérimentation :

Nous avons mené une expérience pour évaluer la performance de notre application. Le but de cette application est de démontrer comment on peut obtenir des points meilleurs. Nous avons développé notre prototype sous NetBeans IDE de Sun Microsystems, la machine d'expérimentation possède les caractéristiques suivantes :

La machine hp 620, processeur: pentium® duel-core CPU T4500 @ 2.30GHz 2.30 GHZ, mémoire installée: 4 Go, Système d'exploitation est Windows 7(64 bits).

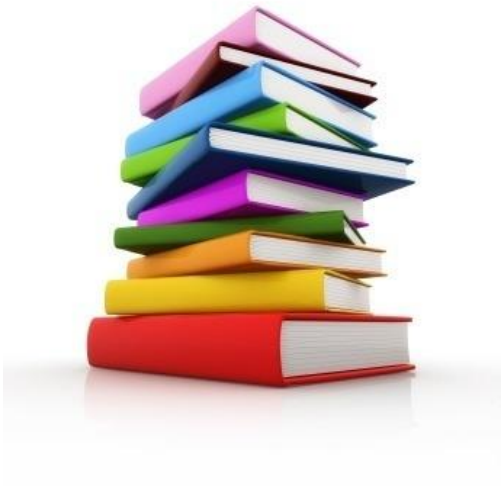
III.6.2.1 Tableau de comparaison :

Nombre de lignes	Nombre de colonnes	Nombre de skylines obtenus	Temps d'exécution
13	2	3	99
10	2	2	67

Table III.1 : La Table de comparaison

III.6.3 conclusion :

Nous avons présenté dans ce chapitre un algorithme d'optimisation multicritère qui permet de retourner les points skylines progressivement et diminuer le nombre de comparaison entre les points candidats.



Conclusion générale

Notre travail consiste à implémenter l'algorithme de recherche des éléments skylines branch&bound, ce dernier est basé sur la séparation et l'évaluation qui accélèrent la recherche des points Pareto optimaux à l'aide de la structure d'un arbre appelée R_Tree qui permet de diminuer le nombre de comparaisons, entre les éléments candidats.

Dans ce cadre nous avons développé une application basée sur cet algorithme, L'idée de base est de construire un arbre qui indexe tous les points du problème, ensuite nous parcourons cette structure en se basant sur deux concepts : la séparation qui consiste à diviser l'espace des nœuds en sous-problèmes pour les traiter de manière individuelle, et l'évaluation basée sur la notion de bornes et le score mindist.

En effet un nœud doit être comparé (selon la relation de dominance) avec des solutions déjà trouvées, si le nœud courant est déjà dominé par l'une des solutions pré existantes alors il est élagué.

En appliquant ces deux règles, nous assurons un gain considérable en terme de temps d'exécution, et surtout lorsque les collections de donnes sont anti corrélées.

Bibliographie

[1] Benameur Lamia, Contribution à l'optimisation complexe par des techniques de swarm intelligence, thèse de doctorat UNIVERSITÉ MOHAMMED V -AGDAL faculté des sciences -rabat ,13 mai 2010.

[2] Borzsonyi S, Kossmann D, and Stocker K. The skyline operator. In Proc of the 17th International Conference on Data Engineering, pages 421–430 Computer Society, 2001.

[3] Catania Band, Jain L.C,Advanved Query Processing ,ISRL 36,pp . 1-13 Springer- Verlag Berlin Heidelberg, 2013.

[4] Chomicki. J, Godfrey P, Gryz J, and Liang D, Skyline with presorting : Theory and optimizations . In Proc of Intelligent Information Systems, pages: 595–604 Springer Berlin/Heidelberg, 2005.

[5] Geoffrion A.M .Proper efficiency and the theory of vector minimization .journal of mathematical Analysis and Applications, page : 618-630,volume22, issue2,1968.

[6] Gutmann A , R-tree .A dynamic Index Structure for Spacial Searching , proceeding of ACM SIGMOD,International Conference on Management of data, page: 47-57 , Boston, Mars 1984.

[7] Messine Frédréric, Méthode d'optimisation Globale basées sur l'analyse d'intervalle pour la resolution de problems avec constraints, these de doctorat de l'université Paul Sabatier Toulouse III, 1997.

[8] Ntaflos Lefteris ,skylines, COMP 6311CSpring ,2012,CSE HKUST.

[9] Papadias Dimitris, Progressive Skyline Computation in Database Systems, ACM Transactions on Database Systems (TODS), Pages : 41-82**,** Volume 30 Issue 1, March2005.

[10] SELLIS T, Roussopoulos N, FaLoutsos C, the R-tree : A dynamic Index for Multi-Dimementional Objects , Proceeding of VLDB,1987 p 507-518 .

[11] ulungu E. L., Teghem J, the two phase method : an efficient procedure to solve bi-objective combinatorial optimization problems. in foundation of computing and decision sciences, vol.20,num 2,p149-165,1995.

[12] Zaki Mohammed. J ∗ and Ching-Jui Hsiao, Charm : An efficient algorithm for closed itemset mining. (2002).

Webographie :

[13] http://jmdoudoux.developpez.com/cours/developpons/java/chap-presentation.php

[14] http://www.java.com/fr/download/faq/whatis_java.xml

[15] http://www.techno-science.net/?onglet=glossaire&definition=11378