Contribution au calcul du skyline par réduction de l'espace candidat

Lougmiri Zekri*,Hadjer Belaicha*

*Déparetement d'Informatique Université d'Oran,
BP 1524 EL-Mnaouer, Maraval Oran, Algérie
{lougmiri, belaichahadjer}@gmail.com

Résumé. L'opérateur skyline est devenu un paradigme dans les bases de données. Il consiste à localiser Sky l'ensemble des points d'un espace vectoriel qui ne sont pas dominés. Cet opérateur est utile lorsqu'on n'arrive pas à se décider dans les situations conflictuelles. Le calcul des requêtes skyline est pénalisé par le nombre de points que peuvent contenir les bases de données. Dans ce papier, nous présentons une solution analytique pour la réduction de l'espace candidat et nous proposons une méthode efficace pour le calcul de ce type de requêtes

1 Introduction

Les requêtes skyline sont importantes dans les applications qui nécessitent la localisation des réponses selon plusieurs critères. Ayant un ensemble de points dans un espace vectoriel de d dimensions, un algorithme traitant ce type de requêtes doit retourner l'ensemble des points de S dits non dominés. Il est meilleur pour ce type d'algorithmes de fonctionner progressivement Kossmann et al. (2002) car les utilisateurs sont souvent impatients de recevoir des réponses.
La relation de dominance se définit comme suit Börzsonyi et al. (2001) : Soit S un ensemble de données de d dimensions (d critères) sur lequel va porter l'opérateur skyline. Soit D l'ensemble de toutes les dimensions $D = \{d_1, , d_d\}$. Soient p et q deux points de S. La relation de dominance ($\prec$) suivant D est pour $1 \leq i, j \leq d$,p domine $q \Longleftrightarrow \{\forall d_i \in D, p(i) \leq q(i)\}$ et $\{\exists d_j \in D, p(j) < q(j)\}$. Lorsque aucun point ne domine l'autre on dit qu'ils sont non dominés ou concurrents. L'opérateur skyline renvoie l'ensemble des points concurrents, suivant toutes les dimensions D : $SkyD(S) = \{p \in S / \nexists q \in S : q \prec p\}$
Dans ce papier, nous présentons une solution analytique pour déduire l'ensemble de points candidats afin d'éviter de parcourir l'ensemble S entièrement. Nous donnons un nouveau théorème pour l'élimination des points non candidats. Notre méthode est basée sur le tri Tan et al. (2001) et à la différence avec ce travail, où les tests entre les points balayent tout l'ensemble S, nous donnons des théorèmes pour la déduction des points les plus évidents et qui constituent les premières solutions à présenter. Nous montrerons que la combinaison de DC Divide-and-Conquer avec notre méthode fournit des résultats meilleurs que lorsqu'il est appliqué tout seul.
Le reste de ce papier se présente comme suit. La section 2 présente les travaux liés à cette problématique. La section 3 donne notre approche. La section 4 présente les résultats des expérimentations. La conclusion et les travaux futurs sont donnés dans la section 5.

2 Travaux liés

Börzsonyi et al. (2001) était le premier travail ayant adapté l'optimisation au sens de Pareto dans les bases de données. Intuitivement, le calcul du skyline consiste à comparer chaque point p avec tous les autres et si aucun point ne le domine alors p est un point skyline. L'algorithme BNL Börzsonyi et al. (2001) utilise cette technique directe. Il met en mémoire une liste candidate et teste à chaque fois si un nouveau point p domine un ou plusieurs points déjà insérés. Si c'est le cas, il est inséré et l'ensemble des points dominé est écarté sinon il passe au point suivant. Cette méthode peut être utilisée facilement et ne requiert pas de prétraitement, sauf qu'elle est gourmande en mémoire et en temps de calcul. DC Börzsonyi et al. (2001) divise l'entrée en plusieurs partitions et détermine le skyline de chaque partition. Par la suite, les skyline sont fusionnés et les points dominés sont écartés. Cette méthode est meilleure que BNL mais souffre des multiples duplications lors de la fusion. Notons que ni BNL ni DC ne fonctionnent en on-line. L'algorithme Bitmap Tan et al. (2001) consiste à encoder dans des vecteurs bitmap toutes les informations de chaque point selon le nombre de points distincts sur chaque axe. La comparaison des vecteurs bitmap se fait par la suite. Bitmap est progressive mais nécessite beaucoup d'opérations et de codage en commençant par la détermination des points distincts dans chaque axe car il y aura beaucoup de tests dupliqués. Index Tan et al. (2001) consiste à trier les données sur chacun des d axes dans un ordre croissant. Afin de déterminer le skyline, les points sont testés de façon circulaire. Le problème est que la récupération des coordonnées des points peut prendre du temps. Cette méthode est bien adaptée pour les applications on-line, elle retourne aussitôt les premiers points, sauf que les auteurs ne déduise pas les skyline induit par le tri. NN Kossmann et al. (2002) est un algorithme qui utilise les R-Trees pour indexer les données. Il partitionne l'espace selon chaque axe selon le point le proche voisin de l'origine. NN est progressif et est efficace dans un espace à deux dimensions, mais il souffre du problème de duplications des éliminations pour 3 dimensions et plus. A partir de 4 dimensions, il devient difficile de l'appliquer. Les auteurs proposent différentes techniques pour remédier à ces problèmes. Branch and Bound Skyline Papadias et al. (2003) exploite les R-Tree, la méthode de Branch and Bound et NN afin de calculer, en on-line, le skyline. Son plus grand problème est qu'il souffre de requêtes redondantes. Yuan et al. (2005) proposent Skycube. Il calculent les skyline fils de toutes les combinaisons possibles des points dans le treillis. Lorsqu'ils passent au niveau supérieur ou inférieur du treillis ils fusionnent ces fils.

3 La méthode DCRD

Notre méthode DCRD pour Divide-and-Conquer for Reduced Data est une méthode analytique qui détermine l'espace candidat en se basant sur le tri. L'utilisation directe de l'espace Pareto est simple pour un espace à 2 dimensions, mais au-delà de 3 dimensions, il faut ajouter des méthodes efficaces pour calculer cet espace. Nous donnons un nouveau théorème qui permet de déduire cet espace. Dans Index, les tests de dominance se font entre tous les points sans l'exploitation de la concurrence induite par le tri. Par exemple, il est impossible qu'un point A, ayant la valeur minimale unique sur un axe X, soit dominé par un autre point. Ce qui nous mène à donner le théorème 1.

Théorème 1 : Existence d'un seul point minimal sur un axe i. Soit E un espace défini par un ensemble D de dimensions $\{d_1, d_2,..., d_d\}$ et soit un ensemble de données S. Pour tout

point $p \in S$ tel qu'il existe un axe i où : $\forall q \in S, p[i] < q[i]$ alors p est un point skyline.
Preuve. Suite à la discussion précédente, sur l'axe i, il est impossible qu'un autre point puisse dominer le point p, puisqu'aucune valeur sur cet axe ne sera inférieure à celle de p. Si pour tous les autres axes, le point q domine p, il sera impossible qu'il le domine sur i, d'où p est soit concurrent avec q soit le domine.
Définition du conflit entre les points ayant des valeurs minimales sur le même axe. Il est fréquent que deux points ou plus aient la même valeur minimale sur un axe. Ainsi, il faut résoudre ce conflit de dominance avant de passer au calcul du skyline définitif. Ainsi, nous donnons le théorème 2 suivant :
Théorème 2 : Existence de plusieurs points minimaux sur un axe i
Soient p et q deux points de S tel qu'il existe un axe i avec $p[i] = q[i]$ et p et q sont les points minimaux sur l'axe i. Si p domine q pour tout sous-espace E' défini par $D' = D - d_i$ alors p est un point skyline sinon p et q sont concurrents.
Preuve. Ceci revient à résoudre le conflit entre p et q dans les autres sous-espaces. Le test montrera soit la dominance soit la concurrence entre eux sur les autres axes.

3.1 Phase 1 : Tri des données et déduction des premiers points skyline

- Sur chaque dimension d_i ($d_i \in D$) trier les valeurs de S par ordre croissant ;
- Extraire l'ensemble Sky des premiers points skyline en utilisant les théorèmes 1 et 2 ;

3.2 Phase 2 : Réduction de l'espace de données

Cette phase consiste à limiter l'espace de données en filtrant l'ensemble de données selon deux points appelés Min_{sys} et Max_{sys}, autrement dit, on détermine l'espace de dominance dans lequel se trouvent tous les points candidats. Min_{sys} est le même que le point idéal de Pareto. Le point Max_{sys} est un point virtuel qui permet de délimiter cet espace. Il sert à éliminer un espace important non utile. En deux dimensions, il se confond au point nadir mais, à plus de dimensions, ils sont différents. Le point nadir est un point pour lequel la fonction à optimiser est maximale **?** et ce n'est pas notre cas car Max_{sys} est dominé.
D'une manière analytique, nous déterminons les points Min_{sys} et Max_{sys} à partir des coordonnées des points skyline déduits de la phase précédente : Min_{sys} et Max_{sys} possède chacun d composantes. Chaque composante $Min_{sys}[i]$ (resp. $Max_{sys}[i]$) de Min_{sys} (resp. Max_{sys}) est la valeur minimale(resp. maximale) de toutes les composantes des points de Sky.
Calcul de l'espace des candidats. Dans cette étape, l'ensemble des données candidats est réduit à un hypercube. Pour l'obtenir, nous énonçons et appliquons le théorème3 suivant :
Théorème 3 : Réduction de l'espace
Soit p un point de S. Si p possède d ou $(d-1)$ composantes (attributs) supérieures aux d ou $(d-1)$ composantes correspondantes du point Max_{sys}, où d est le nombre de dimensions, alors p est éliminé.
Preuve. a) Soit $p \in S$. Si $\forall i$, $p[i] > Max_{sys}[i]$ alors p est éliminé. Ceci est trivial. Dans ce cas, p appartient à un espace dont l'origine est Max_{sys} et sera ainsi dominé.
b) Nous démontrons maintenant que si p possède $(d-1)$ composantes supérieures aux $(d-1)$ composantes équivalentes de Max_{sys} alors p est éliminé. Pour simplifier, on suppose que la d^{ime} composante de p est inférieure à son équivalente de Max_{sys}, autrement dit, $p[d] < Max_{sys}[d]$ et pour $1 \leq i \leq d-1, p[i] > Max_{sys}[i]$.

contribution au calcul du skyline par reduction de l'espace candidat

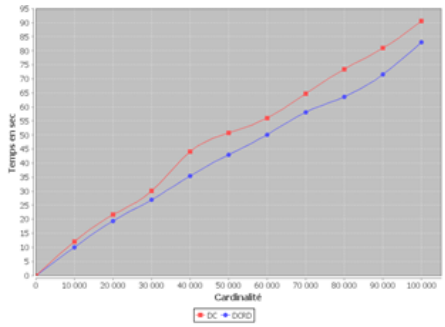

FIG. 1 – *comparaison selon la cardinalité pour le type anticorrélé*

Si les données sont de type corrélé Börzsonyi et al. (2001) alors $Max_{sys} = Min_{sys}$. Ainsi, il y a un seul point skyline. D'où on a : soit $p = Max_{sys}$, soit il est dominé et éliminé.
Pour le reste des types(anti-corrélés, indépendants ou réels), d'une part, on a $Min_{sys} \neq Max_{sys}$ et plus précisément, $Min_{sys} < Max_{sys}$. Formellement, $\forall i, Min_{sys}[i] < Max_{sys}[i]$. D'une autre part, $p[d] < Max_{sys}[d]$ or $Min_{sys}[d] < Max_{sys}[i][d]$ et $Min_{sys}[d] < p[d]$ car la d^{ime} composante de Min_{sys} est obtenue à partir d'un point $q \in Sky$ où la valeur de q est minimale sur l'axe d. Ainsi, $\forall m \in Sky, 1 \leq i \leq d-1, m[i] \leq Max_{sys}[i] < p[i]$ et pour $m = q$, on a $(m[d] = Min_{sys}[d]) < p[d]$. p ne peut dominer q sur l'axe d. Donc, il y a au moins un point m appartenant au skyline actuel qui domine p.

3.3 Phase 3 : Calcul du skyline final

A l'issue de la phase 2, l'espace déduit contient les points candidats. Il ne reste que de les comparer entre eux et éliminer les points dominés, nous appliquons ainsi DC.

4 Expérimentations

Les expérimentations ont été réalisées dans une machine dotée d'Intel Core i5 2,50 GHz, et de 4 Go de RAM, sous Windows 7, 64 bits. Le programme est écrit sous Java. MySQL 5.1.41 est utilisé comme système gestion des bases de données.
Nous avons utilisé les mêmes bases de données synthétiques de Börzsonyi et al. (2001) Kossmann et al. (2002). Il s'agit de trois types de données : corrélées, anti-corrélées et indépendantes. Nous avons calculé le temps nécessaire en secondes pour retourner le skyline en tenant compte de la dimensionalité et la cardinalité.
Comparaison entre DC et DCRD dans le type anti-corrélé. En fixant la dimension à 5 et en variant la cardinalité de 10000 à 100000 points. La figure 1 montre que pour ce type de données, DCRD a rendu les résultats dans des temps meilleurs puisqu'il y a eu des éliminations de points inutiles.
Pour 10000 points et en variant la dimension de 1 à 10, nous remarquons sur la figure 6 que DCRD a commencé à rendre les réponses rapidement à partir de d=6. On déduit ainsi que le

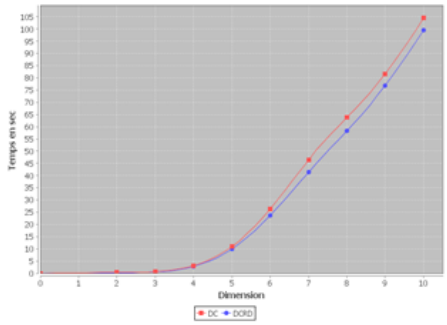

FIG. 2 – *comparaison selon la dimension pour le type anticorrélé*

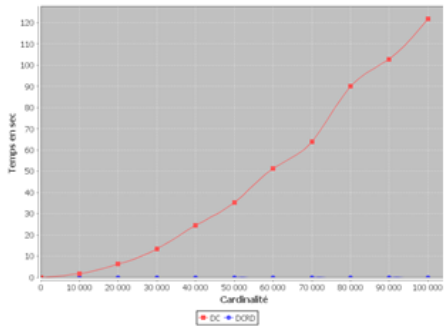

FIG. 3 – *comparaison selon la cardinalité pour le type corrélé*

nombre d'attributs ou de dimensions a un effet sur les temps de réponse puisque ce nombre est proportionnel avec le nombre de tests de dominance.

Comparaison entre DC et DCRD dans le type corrélé Ce type de données est intéressant et facile à manier ; même un algorithme naïf pourrait présenter de bonnes performances. En comparant ces deux méthodes sur la cardinalité et en fixant le nombre d'attributs à 5, nous remarquons sur la figure 3 que DCRD a dépassé de loin DC. Ceci est dû au nombre important de points qui ont été éliminés. La faiblesse de DC est qu'il exécute des tests de dominance sur tout l'ensemble de données d'une façon aveugle.

De même, en comparant DC et DCRD selon la dimension, nous avons fixé la cardinalité à 10000 et nous avons varié le nombre d'attributs. Puisque le nombre de points éliminé est important, la figure 4 montre que DCRD a été très rapide alors que DC a consommé plus de temps. Ceci est toujours le cas puisque DC ne fait aucun traitement préalable et exécute des partitions sur tout l'ensemble d'entrée. Ceci a un effet sur les temps de réponse.

contribution au calcul du skyline par reduction de l'espace candidat

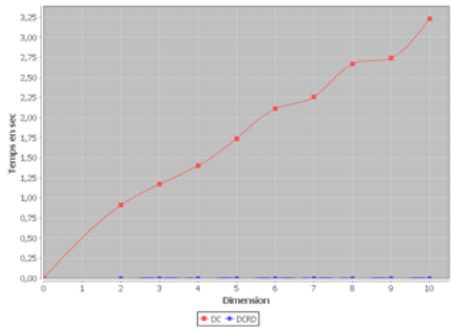

FIG. 4 – *comparaison selon la dimension pour le type corrélé*

5 Conclusion

Dans ce papier, nous avons proposé une méthode analytique pour réduire l'ensemble de données en entrée afin de diminuer le nombre explosif de tests de dominance. Pour ce faire, nous avons prouvé un nouveau théorème. Cette méthode exploite le tri afin de retourner les premiers points skyline sans passer par des tests inutiles. La combinaison de notre méthode avec DC a montré que notre proposition est de meilleure performance.
Actuellement, nous travaillons sur le maintien des points skyline lors des insertions des nouveaux points. Nous travaillons aussi sur la répartition du calcul sur les systèmes pair-à-pair.

Références

Börzsonyi, S., D. Kossmann, et K. Stocker (2001). The skyline operator. *ICDE 2001*, 421–430.

Kossmann, D., F. Ramsak, et S. Rost (2002). Shooting stars in the sky: An online algorithm for skyline queries. *VLDB 2002*, 275–286.

Papadias, D., Y. TAO, G. Fu, et B. Seeger (2003). *An optimal and progressive algorithm for skyline queries*. *SIGMOD 2003*, 467–478.

Tan, K., P. Eng, et B. Ooi (2001). Efficient Progressive Skyline Computation. *VLDB 2001*, 23–28.

Yuan, Y., X. Lin, Q. Liu, W. Wong, X. Yu et Q. Zhang (2005). *Efficient computation of the skyline cubes*. *VLDB 2005*, 241–252.

Summary

This paper presents our approach for computing the skyline. We propose to reduce the set of points in order to minimizing the dominance tests. We combine the DC algorithm with our reducing method. By comparing our method with traditional DC algorithm, we show that our proposition presents better performance