Requêtes skyline avec prise en compte des préférences utilisateurs pour des données volumineuses

Tassadit Bouadi*, Sandra Bringay*, Pascal Poncelet*, Maguelonne Teisseire**

*LIRMM, UMR 5506,161 rue Ada 34392 Montpellier
{bringay, poncelet}@lirmm.fr

*Cemagref, UMR TETIS, 500 rue Jean-François Breton, F-34093 Montpellier
maguelonne.teisseire@cemagref.fr

Résumé. Appréhender, parcourir des données ou des connaissances reste une tâche difficile en particulier lorsque les utilisateurs sont confrontés à de gros volumes de données. De nombreux travaux se sont intéressés à extraire des points "skylines" comme outil de restitution. La prise en compte des préférences a retenu l'attention des travaux les plus récents mais les solutions existantes restent très consommatrices en terme de stockage d'informations additionnelles afin d'obtenir des délais raisonnables de réponse aux requêtes. Notre proposition, EC^2Sky (Efficient computation of compromises), se focalise sur deux points : (1) comment répondre efficacement à des requêtes de type skyline en présence de préférences utilisateurs malgré de gros volumes de données (aussi bien en terme de dimensions que de préférences) ; (2) comment restituer les connaissances les plus pertinentes en soulignant les compromis associés aux préférences spécifiées.

1 Introduction

Appréhender, parcourir des données, reste une tâche difficile en particulier lorsque les utilisateurs sont confrontés à de gros volumes de données pouvant être comparées selon de nombreux critères.

Divers travaux Wong et al. (2008, 2009); Yuan et al. (2005); Tao et al. (2008); Jin et al. (2007); Huang et al. (2008); Su et al. (2007) se sont intéressés à l'extraction des points "skylines" comme outil de restitution dans un contexte décisionnel. Si une requête classique suffit à un touriste pour trouver un hôtel bon marché, cette recherche devient plus complexe lorsqu'il formule plusieurs critères. Les requêtes de type skyline permettent de formuler ces requêtes multi-critères et d'obtenir ainsi par exemple les dix meilleurs hôtels, bon marché et à proximité de la plage. La recherche des skylines est donc un problème d'optimisation multi-critères classique Sawaragi et al. (1985).

La prise en compte de préférences différentes sur les critères selon les utilisateurs (e.g. un voyageur préfère les hôtels de la chaîne Accord à ceux de la chaîne Hilton) a retenu l'attention des travaux les plus récents Wong et al. (2008, 2009) mais les solutions existantes restent très

consommatrices en terme de stockage d'informations additionnelles afin d'obtenir des délais raisonnables de réponse aux requêtes. Notre proposition se focalise donc sur deux points : (1) comment répondre efficacement à des requêtes de type skyline en présence de préférences utilisateurs malgré de gros volumes de données (aussi bien en terme de dimensions que de préférences) ; (2) comment restituer les connaissances les plus pertinentes en soulignant les compromis associés aux préférences spécifiées.

Dans la première partie de cet article, nous présentons les définitions préliminaires et la problématique de la gestion des préférences utilisateurs sur des attributs nominaux. Nous détaillons ensuite notre proposition et la comparons aux références du domaine.

2 Définitions préliminaires : requêtes skylines et préférences utilisateurs

Les différentes concepts seront illustrés à l'aide de la base exemple suivante, tableau 1, qui décrit un ensemble de points E correspondant à des propositions de voyage (vol plus hôtel) avec les attributs suivants : le prix du voyage, la distance de l'hôtel à la plage, le groupe de l'hôtel ainsi que la compagnie aérienne du vol.

Package ID	Prix	Distance	Groupe	Compagnie
a	1600	4	T (Tulips)	G(Gonna)
b	2400	1	T(Tulips)	G(Gonna)
c	3000	5	H(Horizon)	G(Gonna)
d	3600	4	H(Horizon)	R(Redish)
e	2300	2	T(Tulips)	R(Redish)
f	3000	3	M(Mozilla)	W(Wings)
g	3600	4	M(Mozilla)	R(Redish)
h	3000	3	M(Mozilla)	R(Redish)

TAB. 1 – *Ensembles de voyages et critères associés*

L'ensemble des points skylines obtenus à partir de l'ensemble des points E définis dans un espace à n dimensions D, noté $Sky(D,E)$, est construit à partir des points qui **dominent** tous les autres points sur au moins une dimension (ensemble $MaxSky(D,E)$) mais aussi des points qui ne sont pas meilleurs sur une dimension donnée mais qui constituent une solution **compromis** intéressante pour l'utilisateur (ensemble $CompSky(D,E)$).

C'est ainsi qu'à partir de la base exemple, nous obtenons :

$$Sky((Prix, Distance), E) = \{a, b, e\}$$

avec $MaxSky((Prix, Distance), E) = \{a, b\}$ car a et b ont les meilleures valeurs respectivement sur les dimensions 'Prix' et 'Distance' et $CompSky((Prix, Distance), E) = \{e\}$ car e est meilleur que a et b respectivement sur les deux dimensions examinées.

Les requêtes avec **préférences** sur des critères nominaux permettent à l'utilisateur de définir un ordre qui lui est propre pour les valeurs de l'attribut considéré. Dans notre exemple,

un vacancier préfère les hôtels de la chaîne Horizon, alors que d'autres préfèrent les chaînes d'hôtels Mozilla ou Tulips. Il s'agit d'un ordre total si toutes les valeurs sont ordonnées et d'un ordre partiel dans le cas contraire avec des préférences dîtes **implicites** Wong et al. (2008).

La problématique de recherche de points skylines avec préférences utilisateur consiste à calculer à la volée et de la façon la plus efficace possible les points résultats (points dominants et points compromis) selon l'ordre partiel ou total spécifié par l'utilisateur.

3 EC^2Sky : Recherche efficace de compromis skyline avec prise en compte des préférences utilisateurs

Wong et al. (2008, 2009) ont proposé deux méthodes pour la modélisation des préférences dans le cadre des skylines ("IPO-Tree" et "CST") ainsi que des propriétés intéressantes concernant la modélisation des préférences implicites. Cependant, ces deux méthodes pêchent par de nombreux problèmes de mises-à-jour des préférences. En effet, des arborescences sont construites afin de répondre rapidement à une requête mais elles deviennent très complexes et difficilement maintenables dès qu'il y a plus de trois attributs nominaux.

C'est pourquoi, dans la philosophie du compromis, nous proposons à l'utilisateur un moyen d'exprimer différentes préférences pour les attributs nominaux ainsi qu'une méthode lui permettant d'obtenir de meilleures performances tout en ne stockant que le minimum d'informations requises afin d'autoriser des mises-à-jour simples et rapides. Nous ne cherchons pas à recalculer l'ensemble des points skyline sur la totalité des dimensions car ceci est trop coûteux dans un contexte de données multidimensionnelles volumineuses.

3.1 Démarche

Comme Wong et al. (2008, 2009), nous proposons un compromis entre *matérialiser* tous les points skylines pour toutes les préférences possibles des utilisateurs et *calculer* pour chaque requête utilisateur les points skylines en fonction des préférences formulées. Nous proposons une méthode qui s'articule en trois étapes : (1) calcul des points skylines sur les attributs non nominaux ; (2) pour chaque attribut nominal, calcul des points succeptibles d'être skylines selon les différentes préférences utilisateurs de premier niveau seulement ; (3) quand l'utilisateur formule une requête, utilisation des informations stockées en (2) pour trouver les points satisfaisant sa requête.

Plus précisément, pendant l'étape 1, nous calculons tous les points skylines $Sky(D,E)$ correspondant à l'ordre défini dans l'ensemble des préférences $\wp_\emptyset$ ($\wp_\emptyset$ correspond à l'ordre partiel définissant la préférence d'indifférence sur tous les attributs nominaux de l'ensemble D). Ces points constituent l'ensemble **Global skyline point set** défini par Wong et al. (2008) et noté G.

A l'étape 2, pour chaque attribut nominal, nous calculons l'ensemble **Order-sensitive skyline point set** défini par Wong et al. (2008) et noté PC_{D_i}. Cet ensemble représente les points susceptibles de devenir des points skyline par rapport à l'ensemble des attributs nominaux dans le sous-ensemble de dimensions considéré D_i. Il s'agit de tous les points ayant des valeurs différentes de celles des points de l'ensemble G sur les attributs nominaux considérés. Pour illustration, à partir des données de la Table 1, nous avons $G = \{a,b,e\}$ et si l'on considère

l'attribut *Groupe*, $PC_{Groupe} = \{f, g, h, c, d\}$. Ensuite, toujours pour chaque attribut nominal et pour chaque préférence implicite de premier ordre, nous définissons deux ensembles de points $NewSky_{D_i}$ et $CandComp_{D_i}$. $NewSky_{D_i}$ représente l'ensemble des points de l'ensemble PC_{D_i} vérifiant le raffinement de $\wp_0$ et correspondant à la préférence implicite en question. Pour la préférence implicite $H < *$, $NewSky_{Groupe} = \{c, d\}$ car c et d dominent e sur au moins la dimension Groupe. Et $CandComp_{D_i}$ correspond à l'ensemble des points susceptibles de devenir des points skylines compromis c'est-à-dire tous les points appartenants à PC_{D_i} et ne figurant pas dans $NewSky_{D_i}$. Pour l'attribut Groupe et pour la préférence implicite $H < *$, $CandComp_{Groupe} = \{f, g, h\}$ ce qui veut dire que f, g et h dominent c sur au moins une dimension.

A l'issue de l'étape 2, nous conservons l'ensemble PC_i pour chaque dimension D_i et pour chaque préférence de premier ordre $NewSky_{D_i}$ ($CandComp_{D_i}$ étant déduit). A l'étape 3, décrite dans la section suivante, toutes ces informations sont utilisées pour calculer l'ensemble des skylines selon les préférences utilisateur spécifiées à la volée.

3.2 Requêtes skyline avec attributs nominaux

Dans un premier temps, nous allons considérer un seul attribut nominal dans notre ensemble de dimensions D. Afin de calculer les points skylines correspondants à ce type de requêtes, nous allons utiliser une propriété de Wong et al. (2008) nommée "**Merging Property**", qui permet de dériver les points skylines de toutes les préférences implicites possibles et avec n'importe quel ordre et cela en réalisant de simples opérations sur les préférences de premier ordre stockées.

S'il existe plusieurs attributs nominaux, il est nécessaire de prendre en compte tous les points compromis (ensemble *CompSky*) entre les différentes dimensions examinées et tous ceux qui peuvent être disqualifiés (ensemble *CutSky*) par l'introduction d'une nouvelle dimension.

Considérons les ensembles $D_1 = \{Prix, Distance, Groupe\}$ et $D_2 = \{Prix, Distance, Groupe, Compagnie\}$ et les préférences implicites : $H < *$ et $R < *$. $Sky(D_1, E) = \{a, b, c, d, e\}$ et $Sky(D_2, E) = \{a, b, d, e\}$. On remarque que le point c n'appartient plus à l'ensemble des skylines car il est dominé par le point d sur la dimension Compagnie car $CutSky = \{c\}$.

Pour les points compromis, si $Z = \{Groupe, Compagnie\}$ et $\wp = \{H < M < T, G < W < R\}$. $Sky(D,E)_{(Groupe, H<M<T)} = \{a, b, c, d, e\}$ et $Sky(D,E)_{(Compagnie, G<W<R)} = \{a, b, c\}$ alors que $Sky(D,E)_{(Z,\wp)} = \{a, b, c, e, f\}$ car $CompSky(D,E)_{(Z,\wp)} = \{f\}$. En effet, si on considère les deux attributs nominaux, nous trouvons un nouveau point skyline {f}.

4 Expérimentations

Nous avons conduit nos expérimentations sur un serveur ayant les caractéristiques suivantes : CPU Intel 3GHz, 16 GB de mémoire sur une plateforme Linux. L'algorithme EC^2Sky a été implémenté en language C. Nous avons utilisé le générateur de jeux de données de Borzsonyi et al. (2001) pour les attributs numériques. Trois types de jeux de données sont générés : des données indépendantes, des données corrélées et des données non-corrélées. Les attributs nominaux quand à eux ont été générés suivant une distribution de Zipfian Trenkler (1994). Par défaut, nous avons initialisé le paramètre Zipfian θ à 1. Nous avons obtenu 200 000 tuples avec

4 dimensions numériques. Nous avons fait varier le nombre de dimensions nominales de 5 à 8 et le nombre de valeurs des attributs nominaux de 2 à 5.

Lors de ces expérimentations, nous avons comparé notre proposition d'extraction de skylines EC^2Sky avec celle proposée par Wong et al. (2008) à savoir l'approche IPO-Tree, et cela en terme de temps d'exécution et de stockage mémoire.

Notre méthode s'est avérée plus performante que la méthode IPO-Tree. Ceci s'explique essentiellement par le nombre de nœuds de l'arbre IPO-Tree (égal à $O(c^m)$ (m étant le nombre d'attributs nominaux et c la cardinalité d'un attribut nominal)) alors que dans notre cas, la taille est égale à $O(c * m)$.

5 Conclusion

Dans cet article, nous nous sommes intéressés au problème de l'analyse des requêtes skyline sur des attributs nominaux avec des préférences dynamiques relatives à n'importe quel ordre partiel et en particulier au problème d'extraction de points compromis. Nous avons proposé une méthode de recherche de meilleurs compromis EC^2Sky - basée sur une matérialisation partielle des préférences implictes - qui permet de restituer les connaissances les plus pertinentes en soulignant les compromis associés aux préférences spécifiées. Les expérimentations réalisées soulignent l'efficacité de l'approche. Les perspectives de ce travail sont nombreuses. Tout d'abord, nous souhaitons appliquer notre proposition à des jeux de données réelles comme les résultats d'expérimentations biologiques (cancer et maladie dégénérative) que sont les puces ADN et proposer une exploration des motifs associés obtenus lors d'une précédente étape d'extraction. Nous souhaiterions utiliser les pathway (description fonctionnelles des interactions biologiques entre gènes) comme une source de connaissance supplémentaire correspondant aux préférences utilisateurs. Ensuite, il nous paraît intéressant de calculer des requêtes skyline dans un contexte de données hiérarchiques et agrégées. La démarche à adopter permettrait de rechercher les meilleurs compromis le long des axes. Ceci pose plusieurs problèmes, tout d'abord définir un calcul adapté au niveau de hiérarchie exploré et ensuite la sémantique de point skyline de granularité différente.

Références

Borzsonyi, S., D. Kossmann, et K. Stocker (2001). The skyline operator. In *ICDE '01 : Proceedings of the 17th International Conference on Data Engineering*, Washington, DC, USA, pp. 421–430. IEEE Computer Society.

Huang, Z., J. Guo, S.-L. Sun, et W. Wang (2008). Efficient optimization of multiple subspace skyline queries. *J. Comput. Sci. Technol. 23*(1), 103–111.

Jin, W., A. K. H. Tung, M. Ester, et J. Han (2007). On efficient processing of subspace skyline queries on high dimensional data. In *SSDBM '07 : Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, Washington, DC, USA, pp. 12. IEEE Computer Society.

Sawaragi, Y., H. Nakayama, et T. Tanino (1985). *Theory of Multiobjective Optimization*. Academic Press, Orlando.

Su, L., P. Zou, et Y. Jia (2007). Adaptive mining the approximate skyline over data stream. In *ICCS '07 : Proceedings of the 7th international conference on Computational Science, Part III*, Berlin, Heidelberg, pp. 742–745. Springer-Verlag.

Tao, Y., X. Xiao, et J. Pei (2008). Efficient skyline and top-k retrieval in subspaces. *IEEE Trans. on Knowl. and Data Eng. 19*(8), 1072–1088.

Trenkler, G. (1994). Univariate discrete distributions : N.l. johnson, s. kotz and a.w. kemp (1992) : 2nd edition. new york : John wiley, isbn 0-471-54897-9. *Computational Statistics & Data Analysis 17*(2), 240–241.

Wong, R. C.-W., A. W.-C. Fu, J. Pei, Y. S. Ho, T. Wong, et Y. Liu (2008). Efficient skyline querying with variable user preferences on nominal attributes. *Proc. VLDB Endow. 1*(1), 1032–1043.

Wong, R. C.-W., J. Pei, A. W.-C. Fu, et K. Wang (2009). Online skyline analysis with dynamic preferences on nominal attributes. *IEEE Trans. on Knowl. and Data Eng. 21*(1), 35–49.

Yuan, Y., X. Lin, Q. Liu, W. Wang, J. X. Yu, et Q. Zhang (2005). Efficient computation of the skyline cube. In *VLDB '05 : Proceedings of the 31st international conference on Very large data bases*, pp. 241–252. VLDB Endowment.

Summary

Understanding and browsing of data or knowledge is still a difficult task especially when users are confronted with large volumes of data. A considerable amount of research has focused on extracting "skyline" points as a restitution tool. Most of the recent work has taken the preferences into account, but existing solutions are not very efficient in terms of information storage, as they store additional information to improve response time to queries. Our proposal, EC^2Sky, focuses on two points (1) how to respond effectively to requests like skyline, in the presence of user preferences despite large volumes of data (both in terms of dimensions and preferences) (2) how to restore the most relevant knowledge by underlining the associated trade-offs with the specified preferences.