SkyDist: Data Mining on Skyline Objects

Christian Böhm[1], Annahita Oswald[1], Claudia Plant[2],
Michael Plavinski[1], and Bianca Wackersreuther[1]

[1] University of Munich
[2] Technische Universität München
{boehm,oswald,plavinski,wackersreuther}@dbs.ifi.lmu.de,
plant@lrz.tum.de

Abstract. The skyline operator is a well established database primitive which is traditionally applied in a way that only a single skyline is computed. In this paper we use multiple skylines themselves as objects for data exploration and data mining. We define a novel similarity measure for comparing different skylines, called SkyDist. SkyDist can be used for complex analysis tasks such as clustering, classification, outlier detection, etc. We propose two different algorithms for computing SkyDist, based on Monte-Carlo sampling and on the plane sweep paradigm. In an extensive experimental evaluation, we demonstrate the efficiency and usefulness of SkyDist for a number of applications and data mining methods.

1 Introduction

Skyline queries are an important area of current database research, and have gained increasing interest in recent years [1,2,3,4,5]. Most papers focus on efficient algorithms for the construction of a *single* skyline which is the answer of a user's query. This paper extends the idea of skylines in such a way that multiple skylines are treated as objects for data exploration and data mining.

One of the most prominent applications of the skyline operator is to support complex decisions. As an example consider an online marketplace for used cars, where the user wants to find out offers which optimize more than one property of the car such as p (price) and m (mileage), with an unknown weighting of the single conditions. The result of such a query has to contain all offers which may be of interest: not only the cheapest offer and that with lowest mileage but also all offers providing an outstanding combination of p and m.

However, not all of these offers are equally attractive to the user. For instance, consider an offer A that has both a lower price *and* a lower mileage than another offer G and many other offers. Therefore, we say that A *dominates* G (in symbols $A \prec G$), because A is better than G w.r.t. any possible weighting of the criteria price and mileage. The *skyline* contains all objects which are *not dominated* by any other object in the database.

For the used car market, the skyline of each car model has a particular meaning: Many arbitrarily bad offers may be present in the database but only the offers in (or close to) the skyline have a high potential to find a customer. The skyline of the offers marks to some degree the fair value of a car for each mileage in the market. Therefore,

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 461–470, 2010.

the skyline characterizes a car model (or higher-order objects of other applications) in a highly expressive way.

This high expressiveness leads us to the idea of treating the skylines themselves as a measure of similarity. Figure 1 illustrates the skylines of four different car models derived from real data of an online automotive market. Car models which exhibit similar skylines (like Audi A3 1.6 and Ford Focus 1.6) may be considered as similar: A recommender system might find out that the Focus is a perfect alternative for the Audi A3. The different car models may be subject to clustering, classification, outlier detection, or other supervised or unsupervised data mining tasks, using a similarity measure which is built upon the skyline.

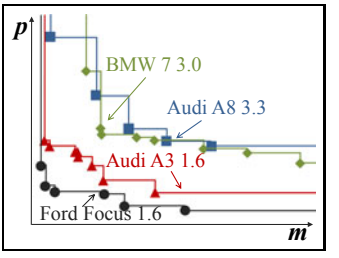


Fig. 1. Skylines of different carmodels

The remainder is organized as follows. We review related work in Section 2. Our novel distance measure for skylines is described in Section 3. We present an experimental evaluation in Section 4 and conclude the paper in Section 5.

2 Related Work

Besides introducing studies on skyline computation we briefly survey clustering algorithms which are applied to demonstrate the potential of data mining on skylines for knowledge discovery in Section 4.

Skyline computation. Many different methods for skyline computation have been emerged in recent years. Brzsnyi *et al.* [1] proposed a SQL syntax for skyline queries and developed the Block-Nested-Loops (BNL) algorithm and the Extended Divide-Conquer algorithm. The Sort-Filter-Skyline algorithm by Chomicki *et al.* [6] improves BNL by pre-sorting the entire dataset. Tan *et al.* [2] presented Bitmap and Index. Kossmann *et al.* [3] developed a nearest neighbor search technique by browsing the data set indexed by an R-tree.

Clustering. In iterative partitioning clustering k-medoid methods such as PAM [7] or CLARANS [8] aim at finding a set of k representatives among all objects that characterize the objects in the data set best. Clusters are created by assigning each object to the cluster of the medoid that is closest to that object. One of the most wide spread approaches to hierarchical clustering is the Single Link algorithm [9]. Starting with singleton clusters for each object, the algorithm merges in each step the closest pair of clusters until it ends up with the root which is one large cluster containing all objects. The hierarchy obtained by the merging order is visualized as a tree which is called dendrogram. In density-based clustering, clusters are regarded as areas of high object density which are separated by areas of lower object density. The algorithm DBSCAN [10] formalizes this idea by two parameters: $MinPts$ specifying a number of objects and ϵ specifying a volume. An object is called core object if it has at least $MinPts$ objects within its ϵ-neighborhood. DBSCAN determines a non-hierarchical, disjoint partitioning of the data set into clusters.

3 SkyDist

In this paper we describe SkyDist, a novel distance function for skyline objects. At the beginning of this section we define the essential concepts underlying SkyDist in general. Then we motivate the idea behind assigning SkyDist for 2-dimensional skylines and conclude with a generalization according n-dimensional skylines, where $n > 2$.

3.1 Theoretical Background

Consider a set of database objects, each of which is already associated with an individual skyline. For instance, each car type is associated with the skyline of the offers posted in a used car market. An effective distance measure for a pair of skyline objects should be useful in the sense that the characteristic properties of the skyline concept are suitably reflected. Whenever two skylines are similar in an intuitive sense, then SkyDist should yield a small value. In order to define such a reasonable distance measure, we recall here the central concept of the classical skyline operator, the *dominance relation* which can be built on the preferences on attributes $\mathcal{D}_1, \ldots, \mathcal{D}_n$ in a n-dimensional numeric space $\mathcal{D}$.

Definition 1 (Dominance). *For two data points u and v, u is said to **dominate** v, denoted by $u \prec v$, if the following two conditions hold:*
$\forall$ *dimensions* $\mathcal{D}_{i\in\{1,\ldots,n\}}$*:* $u.\mathcal{D}_i \leq v.\mathcal{D}_i$
$\exists$ *dimension* $\mathcal{D}_{j\in\{1,\ldots,n\}}$*:* $u.\mathcal{D}_j < v.\mathcal{D}_j$.

Definition 2 (Skyline Point / Skyline). *Given a set of data points $\mathcal{DB}$, a data point u is a **skyline point** if there exists no other data point $v \in \mathcal{DB}$ such that v dominates u. The **skyline** on $\mathcal{DB}$ is the set of all skyline points.*

Two skylines are obviously *equal* when they consist of identical points. Intuitively, one can say that two skylines are similar, whenever they consist of similar points. But in addition, two skylines may also be considered similar if they dominate approximately the same points in the data space. This can be grasped in a simple and efficient way by requiring that the one of the two skylines should not change much whenever the points of the other skyline are inserted into the first one and vice versa. This leads us to the idea to base SkyDist on the set-theoretic difference among the parts of the data space which are dominated by the two skylines. For a more formal view let us define the terms *dominance region* and *non-dominance region* of a skyline.

Definition 3 (Dominance Region of a Skyline Point). *Given a skyline point $x_{i\in\{1,\ldots,n\}}$ of a skyline $X = \{x_1, \ldots, x_n\}$. The **dominance region of** x_i, denoted by $\mathcal{DOM}_{x_i}$, is the data space, where every data point $u \in \mathcal{DOM}_{x_i}$ complies the condition $x_i \prec u$.*

Definition 4 (Dominance Region of a Skyline). *Given the set of all skyline points of a skyline X. The **dominance region of the skyline** X, denoted by $\mathcal{DOM}_X$, is defined over the union of the dominance regions of all skyline points $x_{i\in\{1,\ldots,n\}}$.*

Figure 2 illustrates this notion for two given skylines X and Y. The green and blue areas show the dominance regions of skylines X and Y, respectively.

Definition 5 (Non-Dominance Region of a Skyline). *Given a numeric space $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ and the dominance region $\mathcal{DOM}_X$ of a skyline X. The* ***non-dominance region of*** *X, denoted by $\overline{\mathcal{DOM}_X}$, is $\mathcal{D} \setminus \mathcal{DOM}_X$.*

The basic idea behind SkyDist is to determine the data space that is represented by all possible data points that are located in the dominance region of one skyline and, at the same time, the non-dominance region of the other skyline. More formally we can say that SkyDist is the volume of the distance area between two skylines which can be specified by the following equation.

$$SkyDist(X, Y) = \text{Vol}((\mathcal{DOM}_X \setminus \mathcal{DOM}_Y) \cup (\mathcal{DOM}_Y \setminus \mathcal{DOM}_X)) \quad (1)$$

In order to determine the value of the distance of two skylines X and Y based on the concept described above, we have to limit the corresponding regions in each dimension. Therefore we introduce the notion of a bounding skyline point.

Definition 6 (Bounding Skyline Point). *Given a skyline X in a n-dimensional numeric space $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$, where $x_i \in [0, 1]$. The* ***bounding skyline point*** *of skyline X in dimension i, denoted by x_{Bound_i}, is defined as follows.*

$$x_{Bound_i}.\mathcal{D}_j = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases}$$

The bounding skyline points for two 2-dimensional skyline objects X and Y are marked in Figure 2 in red color. We remark that this concept is also applicable if the domain of one or more dimensions is not bounded. In this case the affected dimensions are bounded by the highest value that occurs in either skyline object X or Y in the according dimension. The coordinates of the remaining skyline points are then scaled respectively.

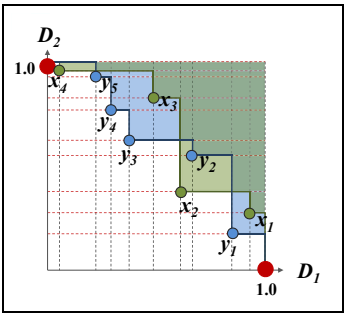
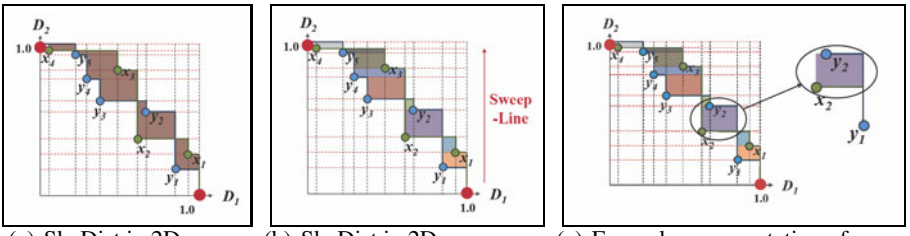


Fig. 2. Dominance regions of two skylines X and Y

3.2 SkyDist by Monte-Carlo Sampling

First we want to give an approximation of the distance of two skylines (cf. Equation 1) by a Monte-Carlo Sampling approach (MCSkyDist). SkyDist is approximated by randomly sampling points and computing the ratio between samples that fall into the region defined by Equation 1 and the ones that do not. Let us consider Figure 3(a). The region marked in red illustrates the region underlying SkyDist of two Skylines X and Y. This region is the dominance region of skyline X and simultaneously the non-dominance region of skyline Y, thus the distance of skyline X to Y. A user defined number of points is randomly sampled and the amount of samples that fall into the SkyDist region is determined. The ratio between the samples located in the SkyDist region and the ones that do not give an approximation of the distance of the two skylines. We use this technique in our experiments as a baseline for comparing a more sophisticated approach.



(a) SkyDist in 2D space. (b) SkyDist in 2D as sum of rectangles. (c) Exemplary computation of a rectangle area.

Fig. 3. SkyDist Computation in 2-dimensional space

3.3 SkyDist for 2-Dimensional Skylines

Now we describe the method of computing the exact distance of two skylines X and Y in 2-dimensional space. Let the skyline points of both skylines X and Y be ordered by one dimension, e.g. $\mathcal{D}_2$. Note that the dominance region of a skyline X is composed by the dominance regions of all of its skyline points. For the computation of SkyDist, we consider only the dominance regions that are exclusive for each skyline point. Meaning that when we look at a particular skyline point x_i, we assign the dominance region of x_i and discard all dominance regions of skyline points x_j, where $x_j.\mathcal{D}_2 > x_i.\mathcal{D}_2$.

Figure 3(a) illustrates in red the region underlying SkyDist according to skyline objects X and Y. This region can be considered as a sum of rectangles as illustrated in Figure 3(b). To calculate the region between the skylines X and Y and thus their distance we use the concept of a sweep-line approach. For this purpose we store the skyline points of both skylines X and Y in a heap structure called event point schedule (EPS), ordered by one of the two dimensions (e.g. $\mathcal{D}_2$) ascending. The general idea behind the sweep-line algorithm is that a line traverses across the plane, stopping at every point that is stored in the EPS. In our example, the sweep line moves along the axis of $\mathcal{D}_2$. Due to the ordering of the skyline points in the EPS we can determine the area of the rectangle at every stop of the sweep-line and calculate SkyDist in an incremental way. Figure 3(c) demonstrates for example the calculation process wenn the sweep-line holds on the skyline point y_2. Now we can calculate the area of the rectangle horizontal limited by the skyline points y_2 and x_2 as $(x_2.\mathcal{D}_1 - y_1.\mathcal{D}_1)(y_2.\mathcal{D}_2 - x_2.\mathcal{D}_2)$.

3.4 A Sweep-Plane Approach for the High-Dimensional Case

In this section we describe how to exactly determine the SkyDist between skylines based on a sweep-plane paradigm referred to as SPSkyDist. We consider a d-dimensional skyline as a sequence of skylines with dimensionality $(d-1)$ (see Figure 4): Here, we use $\mathcal{D}_3$ (in decreasing order) as the ordering dimension and build a sequence of 2-dimensional skylines (each in the $\mathcal{D}_1/\mathcal{D}_2$-plane).

Traversal. The event point schedule (EPS) contains all points, ordered by the last coordinate (decreasingly). In each stop of the sweeping plane, we want to obtain a valid skyline in the space spanned by the remaining coordinates. This sub-skyline is stored in

the sweep-plane status (SPS). In Figure 4, this refers to the four sub-skylines $X_1, ..., X_4$. More precisely, in each stop of the sweep-plane, this $(d-1)$-dimensional sub-skyline is updated by (1) projecting the current point of the EPS into the $(d-1)$-dimensional space, (2) inserting the projected point into the skyline in the SPS, (3) deleting those points from the skyline in the SPS which are dominated by the new point in the $(d-1)$-dimensional space, and (4) calling the traversal-algorithm recursively for the obtained $(d-1)$-dimensional skyline. In our example, the EPS has the order (A, B, C, D, E). We start with an empty SPS, and at the first stopping point, the $\mathcal{D}_1/\mathcal{D}_2$-projection of A is inserted into the SPS to obtain X_1. No point is dominated. Hence, we call the traversal algorithm for the obtained 2-dimensional skyline (A). At the next stop, the projection of B is inserted. Since the projection of B dominates the projection of A (in our figure this fact is symbolized by the canceled-out copy of A), X_2 contains only point B. After the recursive call, in the next stop, the projection of C is inserted which does not dominate object B in the skyline, and, therefore, the next skyline $X_3 = (B, C)$. Finally, D and E are inserted into the skyline in the SPS (which can be done in one single stop of the sweep-plane or in two separate stops in any arbitrary order), to obtain $X_4 = (D, C, E)$, since B is dominated by D in the $(d-1)$-dimensional projection.

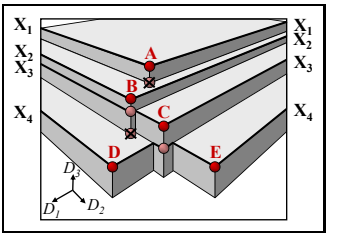


Fig. 4. A sequence of 2-dimensional skylines, forming a 3-dimensional skyline

Simultaneous Traversal for SkyDist. The computation of SkyDist(X, Y) requires a simultaneous traversal of the two skylines X and Y to be compared. That means that the EPS contains the points of both X and Y, simultaneously ordered by the last coordinate. Each of the stops of the sweep-plane corresponds either to a point of X or a point of Y, and the corresponding sub-skyline in the SPS must be updated accordingly, as defined in the last paragraph by (1) projecting the point, (2) inserting the point in the corresponding sub-skyline for either X_i or Y_i, (3) cancelling out the dominated points of the sub-skyline, and finally making the recursive call for $(d-1)$. Having developed this algorithmic template, we can easily obtain SkyDist(X, Y), because each stop of the sweep-plane defines a disklet, the thickness of which is given by the difference between the last and the current event point (taking only the coordinate which is used to order the EPS). This thickness must be multiplied with the $(d-1)$-dimensional volume which is returned by the recursive call that computes SkyDist of the $(d-1)$-dimensional sub-skylines X_i and Y_i. All volumes of all disklets of the obtained sub-skylines have to be added. This works no matter whether current and the previous event points belong to the same or different skylines. Also in the case of tie situations where both skylines have identical values in the ordering coordinates or where some points in the same skyline have identical values, our algorithm works correctly. In this case, some disklets with thickness 0 are added to the overall volume, and, therefore, the order in which the corresponding sub-skylines are updated, does not change anything.

4 Experimental Evaluation

We present an extensive experimental evaluation on synthetic and real world data. We demonstrate that SkyDist is highly effective and efficient and that data mining on skylines provides novel insights in various application domains. For skyline construction, the approach of [1] is applied.

All experiments are performed on an Asus Z92J which is equipped with an Intel Dual-Core T2050 Processor and has 2 GByte of RAM.

4.1 Efficiency

To evaluate the stability of the baseline approach MCSkyDist and the scalability of SkyDist, we generate synthetic data of various number of objects and dimensions. Unless otherwise specified, the skyline is constructed from 1,000 uniformly distributed 2-dimensional data objects and for MCSkyDist we use a sample rate of 1,000.

Accuracy of MCSkyDist w.r.t. Sample Rate. We vary the sample rate in a range of 1 to 50,000 and quantify the accuracy of SkyDist in each run. These experiments indicate that the accuracy of MCSkyDist is very robust w.r.t. the number of samples. Its results achieve a constant value even with a small sample rate. Actually with 1,000 samples the same result as using SPSkyDist can be achieved.

Runtime w.r.t. Number of Objects and Dimensionality
We use data sets with varying number of points and dimensionality and generate skylines for each data set. In most real world applications, we are interested in the skyline w.r.t a few selected dimensions. Hence, in this experiments, we focus on skylines up to dimensionality $d = 4$. All results are summarized in Table 1. In the 2- dimensional case the runtime of SkyDist remains constant even with increasing data size. This is due to the fact, that despite increasing database size the number of skyline points remains relatively constant. It has been shown in [5] that for independent distributed data the number of skyline objects is $O(\log_2 n)$.

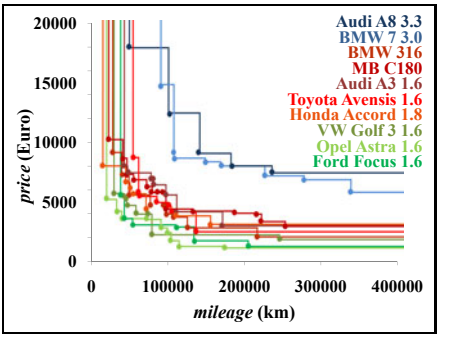


Fig. 5. Clustering of car models represented by their skyline

Also in the 3- and 4-dimensional case it is evident that considering the skyline points instead of all data points is very efficient. It takes 78 and 266 ms for SPSkyDist and MCSkyDist respectively to return the result when comparing two skylines X and Y each determined out of 10,000 data points. MCSkyDist and SPSkyDist both scale linear with increasing dimensionality. The sweep-plane approach outperforms the baseline by a factor of two which confirms the effectivity and scalability of an exact computation of SkyDist even for large data sets with more than two dimensions.

Table 1. Runtime analysis for SPSkyDist and MCSkyDist

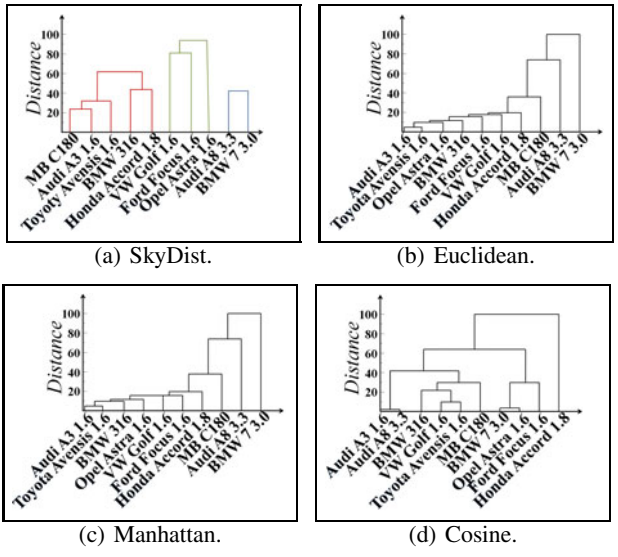
	# data points	# skyline points of skyline X	# skyline points of skyline Y	SPSkyDist	MCSkyDist
2D data	10	4	3	15 ms	47 ms
	100	5	6	16 ms	47 ms
	1000	9	5	15 ms	63 ms
	10000	7	12	15 ms	78 ms
3D data	10	5	5	31 ms	62 ms
	100	18	16	47 ms	109 ms
	1000	29	19	63 ms	125 ms
	10000	68	39	78 ms	266 ms
4D data	10	5	8	47 ms	78 ms
	100	25	36	172 ms	141 ms
	1000	80	61	203 ms	297 ms
	10000	187	151	609 ms	1078 ms

4.2 Clustering Skylines of Real World Data

In addition to synthetic data we used real world data to demonstrate the potential of SkyDist for data mining. We demonstrate that interesting reasonable knowledge can be obtained by clustering skylines with SkyDist. In particular, we focus on two case studies from different applications and we apply three different clustering algorithms (PAM, Single Link and DBSCAN) with SkyDist.

Case Study 1: Automotive Market. The data used in this experiment is obtained from the online automotive market place (`http://www.autoscout24.de`). The resulting data set comprises in total 1,519 used cars constructed in the year 2000. Thus, each data point represents a specific offer of a used car which are labeled to one of three classes *compact*, *medium-sized* and *luxury*, respectively. This information allows for an evaluation of the results. Each car model is represented by one 2-dimensional skyline using the attributes $mileage$ and $price$. Hence each skyline point represents a specific offer that provides an outstanding combination of these attributes, and therefore it is interesting for the user. PAM, DBSCAN and Single Link combined with SkyDist create an identical clustering result (cf. Figure 5) using the following parameterization: PAM ($K = 3$), DBSCAN ($\epsilon = 10$, $MinPts = 2$) and the dendrogram Single Link with a vertical cut at $maxDistance = 90$.

All algorithms produce 100% class-pure clusters w.r.t the labelling provided by the online market place. As expected the Audi A8 and BMW 7 of class *luxury* are clustered together in the blue cluster with a very low distance. Also the Mercedes Benz C180 (MB C180) and the Audi A3 belong to a common cluster (marked in red) and show a larger distance to the Toyota Avensis or the Honda Accord. These two car models are clustered together in the green cluster, whose members usually have a cheaper price. The clustering result with SkyDist is very informative for a user interested in outstanding combinations of $mileage$ and $price$ but not fixed on a specific car model. By clustering, groups of models with similar skylines become evident.



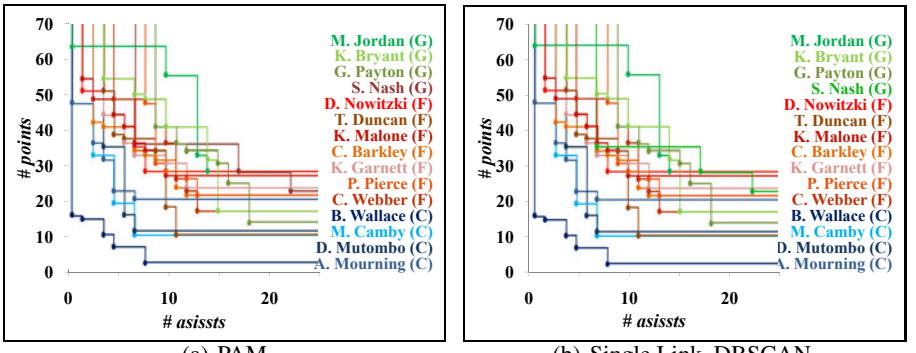
(a) SkyDist. (b) Euclidean.

(c) Manhattan. (d) Cosine.

Fig. 6. The dendrogram of Single Link using SkyDist in comparison to conventional metrics

SkyDist vs. Conventional Metrics. Conventional metrics can in principle be applied for clustering skylines. For this purpose we represent each car model as a vector by calculating the average of the skyline points of the respective car model in each dimension. Then we cluster the resulting vectors with Single Link using the Euclidean, Manhattan and Cosine distance. In contrast to clustering skylines using SkyDist (cf. Figure 6(a)), Figures 6(b), 6(c) and 6(d) demonstrates that no clear clusters are identifiable in the dendrogram and the result is not very comprehensible. In Figure 6(b) and 6(c) it can easily be seen that Euclidean and Manhattan distance lead to similar results. Both show the so called Single Link effect, where no clear clusters can be identified. Using the Cosine distance avoids this effect but does not produce meaningful clusters either. For example, the luxury car model BMW 7 has minimum distance to the Opel Astra of class *compact* but has an unexpected high distance to the luxury Audi A8.

Case Study 2: Performance Statistics of Basketball Players. The NBA game-by-game technical statistics are available via `http://www.NBA.com`. We focus on the years 1991 to 2005. To facilitate demonstration and interpretation, we select players who have played at minimum 500 games and are noted for their skills and got various awards. The players are labeled with the three different basketball positions *guard* (G), *forward* (F) and *center* (C). The individual performance skyline of each player represents the number of assists and points. We cluster the skylines using SkyDist. Single Link with a vertical cut of the dendrogram at $maxDistance = 96$ and DBSCAN ($\epsilon = 4$, $MinPts = 2$) result in the same clustering. Figure 7(b) shows that the players cluster very well in three clusters that refer to the labels G, F and C. With PAM ($K = 3$) (cf. Figure 7(a)) only Steve Nash (G) clusters into the red *forward* cluster. This can be explained by the fact, that this player has performance statistics as a player in the position forward concerning number of points and assists.



(a) PAM (b) Single Link, DBSCAN.

Fig. 7. Clustering of NBA Players represented as skylines

5 Conclusions

This is the first approach to data mining on skyline objects. Inspired by the success of the skyline operator for multi-criteria decision making, we demonstrated that the skyline of a data set is a very useful representation capturing the most interesting characteristics. Hence, data mining on skylines is very promising for knowledge discovery. As an essential building block, we proposed SkyDist, which is a novel distance function for skylines. We presented a baseline approach that approximates the distance and an exact plane sweep based computation. SkyDist can easily be integrated into many data mining techniques. Real world case studies on clustering skylines demonstrated that data mining on skylines enabled by SkyDist yields interesting novel knowledge. Moreover, SkyDist is efficient and thus scalable to large data sets.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
2. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: VLDB (2001)
3. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB, pp. 275–286 (2002)
4. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: SIGMOD Conference, pp. 467–478 (2003)
5. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: ICDE, pp. 502–513 (2005)
6. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting: Theory and optimizations. In: Intelligent Information Systems, pp. 595–604 (2005)
7. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley, Chichester (1990)
8. Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. In: VLDB, pp. 144–155 (1994)
9. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
10. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)