Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Multi-Source Spatial Entity Linkage

Isaj, Suela; Pedersen, Torben Bach; Zimányi, Esteban

Published in:
I E E E Transactions on Knowledge & Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2020.2990491

Creative Commons License
CC BY 4.0

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Isaj, S., Pedersen, T. B., & Zimányi, E. (2022). Multi-Source Spatial Entity Linkage. *I E E E Transactions on Knowledge & Data Engineering*, *34*(3), 1344-1358. https://doi.org/10.1109/TKDE.2020.2990491

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: May 01, 2024

Multi-Source Spatial Entity Linkage

Suela Isaj, Torben Bach Pedersen, *Senior Member, IEEE*, and Esteban Zimányi

Abstract—Besides the traditional cartographic data sources, spatial information can also be derived from location-based sources. However, even though different location-based sources refer to the same physical world, each one has only partial coverage of the spatial entities, describe them with different attributes, and sometimes provide contradicting information. Hence, we introduce the spatial entity linkage problem, which finds which pairs of spatial entities belong to the same physical spatial entity. Our proposed solution (*QuadSky*) starts with a time-efficient spatial blocking technique (*QuadFlex*), compares pairwise the spatial entities in the same block, ranks the pairs using Pareto optimality with the *SkyRank* algorithm, and finally, classifies the pairs with our novel *SkyEx-** family of algorithms that yield 0.85 *precision* and 0.85 *recall* for a manually labeled dataset of 1,500 pairs and 0.87 *precision* and 0.6 *recall* for a semi-manually labeled dataset of 777,452 pairs. Moreover, we provide a theoretical guarantee and formalize the *SkyEx-FES* algorithm that explores only 27 percent of the skylines without any loss in *F-measure*. Furthermore, our fully unsupervised algorithm *SkyEx-D* approximates the optimal result with an *F-measure* loss of just 0.01. Finally, *QuadSky* provides the best trade-off between *precision* and *recall*, and the best *F-measure* compared to the existing baselines and clustering techniques, and approximates the results of supervised learning solutions.

Index Terms—spatial data, entity resolution, spatial blocking, skyline-based

1 Introduction

Web data and social networks are growing in terms of information volume and heterogeneity. Almost all online sources offer the possibility to introduce locations (geo-tagged entities accompanied by semantic details). A specific type of sources whose primary focus is locations is *location-based sources*, such as Google Places, Yelp, Foursquare, etc. In contrast to cartographic data sources, locations in location-based sources have a hybrid form that stands between a *spatial object* and an *entity*. We refer to them as *spatial entities* since they are spatially located but also identified by other attributes such as the name of the location, the address, keywords, etc. Spatial entities play a key role in several systems that rely on spatial information such as geo-recommender systems, selecting influential locations, search engines using geo-preferences, etc.

However, while a spatial object is identified only by the coordinates, this is not the case for spatial entities. Different spatial entities might co-exist in the same coordinates (shops in a shopping mall), or the same entity might be located in different but nearby coordinates across different sources (e.g., "Chicago Roasthouse" appears in Yelp and Google Places with coordinates 82 meters apart). The identity of a spatial entity is the combination of several attributes. Unfortunately, the identity of a spatial entity is sometimes difficult to infer due to the inconsistencies within and among the sources; each location-based source contains different attributes; some attributes might be missing and even contradicting. For example, source A contains the spatial entity "Lygten" in (57.436 10.534) with the keywords "coffee", "tea", and "cocoa and spices", while source B contains "Restaurant Lygten" in (57.435 10.533) with the keyword "restaurant". We need a technique that can automatically decide whether these two spatial entities are the same real-world entity. The problem of finding which spatial entities belong to the same physical entity is referred to as *spatial entity linkage* or *spatial entity resolution*. We use the term entity linkage since we do not merge the entities [1].

There are several works that apply entity linkage in various fields [2], [3], [4], [4], [5], [5], [6], [7], [8], [9], [10] but only little work on spatial entities [11], [12], [13], [14], even though they are central in geo-related research. The entities in the majority of the entity linkage research refer to people; thus, the methodologies and the models are based on the similarities that two records of the same individual would reveal. Moreover, these works do not address the spatial character of spatial entities. As for the works in spatial entity integration [11], [12], [13], their main contribution is a tool rather than an algorithm. What is more, the methods propose arbitrarily attribute weights and score functions without experimentation nor evaluation. In contrast to [11], [12], [13], the skyline-based algorithm (*SkyEx*) proposed in [10] is free of scoring functions and semi-arbitrary weights, and achieves good results. However, *SkyEx* is dependent on a threshold number of skylines k, which can only be discovered through experiments, as the authors do not provide methods for estimating k. To sum up, *on the one hand, there is a growing amount of information about spatial entities, both within a single source and across sources, which can improve the quality of the geo-information; on the other hand, the spatial entity linkage problem is hard to resolve not only because of the*

- *S. Isaj and T. Bach Pedersen are with the Department of Computer Science, Aalborg University, 9100 Aalborg, Denmark. E-mail: {suela, tbp}@cs.aau.dk.*
- *E. Zimányi is with the Department of Computer and Decision Engineering, Université libre de Bruxelles, 1050 Bruxelles, Belgium. E-mail: ezimanyi@ulb.ac.be.*

Manuscript received 10 Oct. 2019; revised 3 Mar. 2020; accepted 17 Apr. 2020. Date of publication 27 Apr. 2020; date of current version 3 Feb. 2022. (Corresponding author: Suela Isaj.)
Recommended for acceptance by M. A. Cheema.
Digital Object Identifier no. 10.1109/TKDE.2020.2990491

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

heterogeneity of the data but also because of the lack of appropriate and effective methods.

In this paper, we address the problem of spatial entity linkage across different location-based sources. We significantly extend a previous conference paper [14]. As an overall solution building on [14], first, we propose a method that uses the geo-coordinates to arrange the spatial entities into blocks. Then, we pairwise compare the attributes of the spatial entities. Later, we rank the pairs according to their similarities using our novel technique, *SkyRank*. Finally, we introduce three approaches (*SkyEx-F*, *SkyEx-FES*, *SkyEx-D*) for deciding whether the pairs of compared entities belong to the same physical entity. Our contributions are: (1) we introduce *QuadSky*, a technique for linking spatial entities and we evaluate it on real-world data from four location-based sources; (2) we propose an algorithm called *QuadFlex* that organizes the spatial entities into blocks based on their spatial proximity, maintaining the complexity of a quadtree and avoiding assigning nearby points into different blocks; (3) to rank the pairs by their similarity, we propose a flexible technique (*SkyRank*) that is based on the concept of Pareto optimality; (4) to label the pairs, we propose the *SkyEx-** family of algorithms that considers the ranking order of the pairs and fixes a cut-off level to separate the classes; (5) we introduce two threshold-based algorithms: *SkyEx-F* that uses the *F-measure* to separate the classes, and *SkyEx-FES*, an optimized version of *SkyEx-F*, which provides a theoretical guarantee to prune 73 percent of the skyline explorations of *SkyEx-F*; (6) we propose *SkyEx-D*, a novel algorithm that is fully unsupervised and parameter-free to separate the classes.

Contributions 1 and 2 originate from [14], contributions 5 and 6 are new, and 3 and 4 are significantly improved compared to [14]. The work in [14] reported very good results compared to the baselines, but had the following limitation: the proposed threshold-based labeling algorithm *SkyEx* needed the threshold number of skylines k as input, and there were no proposed solutions on how to fix k, apart from experimenting with different values. We address this limitation by first modifying the original *SkyEx* in [14] as to only rank and not label the pairs, and we refer to it as *SkyRank*. Then, we delegate the classification problem to three new algorithms, namely *SkyEx-F*, *SkyEx-FES* and *SkyEx-D*. The experiments in [14] attempt to fix k using *precision*, *recall* and *F-measure*. We now formalize this rationale in our novel *SkyEx-F* algorithm. We improve further by providing a theoretical guarantee that *SkyEx-F* can be stopped before exploring the whole dataset, and propose the optimized *SkyEx-FES* that prunes 80 percent of the skyline explorations of *SkyEx-F*. Furthermore, we introduce a novel approach for estimating the number of skylines (*SkyEx-D*), which is fully unsupervised and parameter-free and closely approximates the threshold-based versions (*SkyEx-F* and *SkyEx-FES*). In the present paper, we provide a new set of experiments for *SkyEx-FES* and *SkyEx-D*, and compare with *SkyEx-F*, supervised learning and clustering techniques.

The remainder of the paper is structured as follows: first, we describe the state of the art in Section 2; then, we introduce our approach in Section 3; later, we detail the stages of our approach: the spatial blocking in Section 4, comparing the pairs in Section 5, ranking the pairs in Section 6, and estimating the kth level of skyline in Section 7; we analyze the complexity of our solution in Section 8; we provide experiments in Section 9; and finally, we conclude in Section 10.

2 Related Work

In this section, we describe some works on *entity resolution*, *spatial data integration*, and *spatial entity linkage*.

Entity Resolution. The *entity resolution* problem has been referred in the literature with multiple terms including *deduplication*, *entity linkage*, and *entity matching* [4], [15]. Entity resolution has been used in various fields such as matching profiles in social networks [2], bioinformatics data [3], biomedical data [16], publication data [4], [5], genealogical data [6], product data [4], [5], etc. The attributes of the entities are compared, and a similarity value is assigned. The decision of whether to link two entities or not is usually based on a scoring function. However, finding an appropriate similarity function that combines the similarities of attributes and decides on whether to link or not the entities is often difficult. Several works use a training set to learn a classifier [7], [8], [17], others base the decision on a threshold derived through experiments [9], [18]. Other approaches decide the include the uncertainty of a match into the decision [19]. Finally, matching the entities can also be based on the feedback of an oracle [4], [5] or of a user [5].

Spatial Data Integration. There are several works on integrating purely spatial objects. Spatial objects differ from spatial entities mainly because a spatial object is fully determined by its coordinates or its spatial shape whereas a spatial entity, in addition to being geo-located, has a well-defined identity (name, phone, categories). The works on spatial object integration aim to create a unified spatial representation of the spatial objects from single/multiple sources. Schafers at al [20] integrate road networks using rules for detect matching and non-matching roads based on the similarity in terms of the length, angles, shape, as well as the name of the street if available. The solutions in [21], [22], [23], [24] are purely spatial and discuss the integration of spatial objects originating from sensors and radars to have a better representation of the surface in 2D or even in 3D. These approaches cannot apply to spatial entities.

Spatial Entity Linkage. Accommodating the challenges of spatial entities for the entity resolution problem has been specifically addressed in [11], [12], [13], [14], [25], [26]. The work in [25] is a bridge between the works in spatial data integration and spatial entity linkage because the entities have names, coordinates, and types but similarly to spatial objects, they refer to landscapes (rivers, deserts, mountains, etc.). The method used in [25] is supervised and requires labeled data. Moreover, even the similarity of the attribute "type" is learned through a training set. Regarding [11], [12], [13], the main contribution of these works relies on designing a spatial entity matching tool rather than an integration algorithm. In [13], the spatial entities within a radius are compared with each other, and the value of the radius is fixed depending on the type of spatial entity. For example, the radius is 50 m for restaurants and hotels, but 500 m for parks. All attributes (except coordinates) are compared using the Levenshtein similarity. Since the name, the geo-data and the type of the entity are always present, they carry

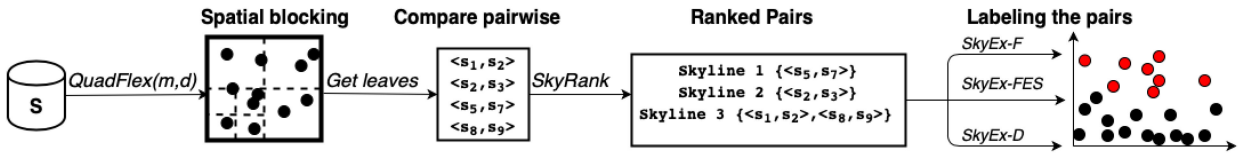


Fig. 1. QuadSky approach.

two-thirds of the weight in the scoring function whereas the weights of the website, the address and the phone number are tuned to one-third. The prototype of the spatial entity matching in [12] relies on a technique that arbitrarily uses an average of the similarity scores of all textual attributes without providing a discussing on this choice. Similarly to [11], [12], the main contribution of the work in [13] is designing a tool for spatial entity integration. The underlying algorithm considers spatial entities that are 5 m apart from each other and compares the name of the entities syntactically and the metadata related to an entity semantically. Finally, the decision is taken using the belief theory [26]. The works in [11], [12], [13] lack an evaluation of the algorithms. The work in [14] proposes a scalable spatial quadtree-based blocking technique that not only fixes the distance between the spatial entities but also controls the density of the blocks. Then, the spatial entities of the same block are compared on their name (Levenshtein), address (custom) and categories (Wu&Palmer using Wordnet). Finally, a threshold-based algorithm (*SkyEx*) is used to separate the classes. However, instead of using fixed thresholds for each attribute similarity, *SkyEx* abstracts the similarities into skylines and needs only one threshold number of skylines k to separate the classes. The authors provide experiments and evaluations, nevertheless, they lack estimation techniques for fixing k. The present paper uses the solution in [14] for the spatial blocking and the pairwise comparisons. We use the skylines for the labeling process as in *SkyEx*, but we propose three new algorithms (*SkyEx-F*, *SkyEx-FES* and *SkyEx-D*) to separate the classes, fixing k internally.

Summary. The general entity resolution approaches propose interesting solutions, but they do not consider the spatial character of a spatial entity. The majority are designed to match entities that represent individuals (profiles in social networks, authors and publications, medical records, genealogical connections, etc.) or even linking species in nature. The proposed solutions for entity resolution in individuals, either supervised or based on an experimental threshold, are learned on human entity datasets. One can not merely assume the resemblance of behaviors in a human entity dataset to a spatial entity one. The solutions in species in nature are based on domain-specific algorithms that have little to no applicability in other fields. There is little specific work in spatial entities [11], [12], [13], mostly focusing on a tool for spatial data integration rather than on the algorithm. In all these works, the scoring function is chosen arbitrarily and no evaluation provided.

3 Spatial Entity Linkage

In this section, we introduce the problem definition and our overall solution. The basic concept used in this work is a *spatial entity* such as places, businesses, etc. Spatial entities originate from location-based sources, e.g., directories with location information (yellow pages, Google Places, etc.) and location-based social networks (Foursquare, Gowalla, etc.).

Definition 1. *A spatial entity s is an entity identified uniquely within a source I, located in a geographical point p and accompanied by a set of attributes $A = \{a_i\}$.*

The attributes connected to s can be categorized as: *spatial*: the point where the entity is located, expressed in longitude and latitude; *textual*: attributes that are in the form of text such as name, address, website, description, etc.; *semantic*: attributes in the form of text that enrich the semantics behind a spatial entity, e.g., categories, keywords, metadata, etc.; *date, time or number*: other details about a spatial entity such as phone, opening hours, date of foundation, etc. An example of a spatial entity originating from Yelp can be a place named "Star Pizza" in the point (56.716 10.114), with the keywords "pizza, fast food", and with address "Storegade 31". The same spatial entity can be found again in Yelp or other sources, sometimes having the same attributes, more, less, or even attributes with contradictory values. Thus, there is a need for an approach that can unify the information within and across different sources in an intelligent manner.

Problem definition: Given a set of spatial entities S originating from multiple sources, the spatial entity linkage problem aims to find those pairs of spatial entities $\langle s_i, s_j \rangle$ that refer to the same physical spatial entity.

We propose *QuadSky*, a solution based on a quadtree data partitioning and skyline exploration. The overall approach is detailed in Fig. 1. *QuadSky* consists of four main parts: spatial blocking (*QuadFlex*), pairwise comparisons, ranking the pairs (*SkyRank*), and labelling the pairs (the *SkyEx-** family of algorithms). S contains all spatial entities. We propose *QuadFlex*, a quadtree-based solution that can perform the spatial blocking by respecting the distance between spatial entities and the density of the area. The output of *QuadFlex* is a list of leaves with spatial entities located nearby. Within the leaves, we perform the pairwise comparisons of the attributes. Then, we rank the compared pairs based on the skylines (concepts detailed in Section 6) using the *SkyRank* algorithm. In order to decide which pairs dictate a match and which not, we propose the *SkyEx-** family of algorithms (*SkyEx-F*, *SkyEx-FES*, and *SkyEx-D*) that finds which skyline level best separates the pairs that refer to the same physical spatial entity (the positives class) from the rest (the negative class). In the following sections, we detail each of the phases of *QuadSky*. We use the notations in Table 1 (We will explain them gradually during the paper).

4 Spatial Blocking

Since spatial proximity is a strong indicator of finding a match, the first step is to group nearby spatial entities in blocks. Several generic blocking techniques have been

discussed in [27], [28], but mostly based on textual attributes and not applicable to spatial blocking. We propose a quadtree-based solution (*QuadFlex*) that uses a tree data structure but also preserves the spatial proximity of spatial entities. A quadtree is a tree whose nodes are always recursively split into four children when the capacity is filled [29]. After the quadtree is constructed, the points that fall in the same leaf are nearby spatially. Hence, these leaves are good candidates to be spatial blocks. However, the existing quadtree algorithm needs to be adapted for spatial blocking.

Algorithm 1. QuadFlex Algorithm

input: A set of entities $S = \{s_i\}$, diagonal m, density d
output: The leaves QuadFlex Q $Q.leaves()$;
1: Create $Q(m, d)$ where Q has the dimensions of the bounding box of S
2: **for each** s in S **do**
3: $Q.insert(s)$ // *Insert s into the QuadFlex*
4: **end for**
return $Q.leaves()$
Method *insert (s)*
5: **if** $this.children \neq$ **then**
6: $Indexes \leftarrow getIndex(s)$ // *Find where s belongs*
7: **for each** i in $Indexes$ **do**
8: $this.child[i].insert(s)$ // *Insert s to the children it belongs*
9: **end for**
10: **end if**
11: **if** $this.diagonal > m$ **or** $this.density > d$ **then**
12: Split the current object $this$ into 4 children
13: **end if**
14: $Indexes \leftarrow getIndex(s)$
15: **for each** i in $Indexes$ **do**
16: $this.child[i].insert(s)$
17: **end for**
return
Method *getIndex (s)*
18: Let $vertical - left$ and $vertical - right$ be the lines that pass at 0.25 and 0.75 of the width of $this$, respectively
19: Let $horizontal - up$ and $horizontal - down$ be the lines that pass at 0.25 and 0.75 of the height of $this$, respectively
20: **if** s is left of $vertical - right$ and above $horizontal - down$ **then**
21: $Indexes.add(1)$ // s fits in $child[1]$
22: **end if**
23: **if** s is right of $vertical - left$ and above $horizontal - down$ **then**
24: $Indexes.add(2)$ // s fits in $child[2]$
25: **end if**
26: **if** s is left of $vertical - right$ and below $horizontal - up$ **then**
27: $Indexes.add(3)$ // s fits in $child[3]$
28: **end if**
29: **if** s is right of $vertical - left$ and below $horizontal - up$ **then**
30: $Indexes.add(4)$ // s fits in $child[4]$
31: **end if**
return $Indexes$

First, a quadtree needs a capacity (number of points) as a parameter. The capacity is not a meaningful parameter for spatial blocking, while the density of the area is a better candidate. For example, if the area is too dense (e.g., city center), even though the capacity is not reached, a further split would be more beneficial. On the contrary, two points in the countryside (e.g., a farm) might be farther apart, but they still might be the same entity. Second, a quadtree does not limit the distance between points. Even though two points might be in an area that respects the density, if they are quite distant from each other, it is not necessary to compare them. The maximal distance between two points in a child is the diagonal of the area (all quadtree children are rectangular). We used m, the diagonal of an area, as a parameter that controls the distance of points rather than comparing all distances between all spatial entities. Finally, a quadtree splits into four children, and sometimes nearby points might fall into different leaves. We modify the procedure of the assignment of the points into a child by allowing more than one assignment.

TABLE 1
Notations Used Throughout the Paper

Notation	Description
s	A spatial entity with a point p and a set of attributes $\{a_i\}$
S	A set of spatial entities $\{s_i\}$
Q	A *QuadFlex* structure used for spatial blocking
P	A set of pairs $\{\langle s_i, s_j \rangle\}$
δ_a	The similarity of a pair in terms of attribute a
$u(\langle s_i, s_j \rangle)$	The utility of a pair $\langle s_i, s_j \rangle$
$Skyline(k)$	A skyline of pairs $\{\langle s_i, s_j \rangle\}$ in the level k
K	The total number of skylines
k	A variable indicating the level of skyline
k_f	A k value fixed by *SkyEx-F* and *SkyEx-FES*
k_d	A k value fixed by *SkyEx-D*
P_k	Pairs of P associated with a skyline
P^+	A subset of pairs in P classified as positive
P^-	A subset of pairs in P classified as negative
$F1(k)$	The F-measure in the kth level of skyline
μ_d	The mean of the distances between the two classes.
$\mu_d(k)$	The function measuring μ_d in in each k level of skylines
$\mu'_d(k)$	The first derivative of $\mu_d(k)$

Fig. 2 shows the modifications that we do to the construction of the traditional quadtree for our version *QuadFlex*. The traditional quadtree divides the area of each parent into four smaller areas, the children. A point belongs only to one child. In our modification, the area will split into 4 children in the same way as a quadtree (at 0.5 of the height and 0.5 of the width of the parent), but when we assign a point to a child, we will consider including points that fall shortly outside the border in the current child, too. For example, in Fig. 2, *QuadFlex* physically splits in the same way as the quadtree, but the red dashed line shows the area that will be considered for including neighboring points. The red points are in the overlapping regions and will be included in more than one child. Algorithm 1 details the procedure for retrieving the spatial blocks with *QuadFlex*. The algorithm creates the root of the *QuadFlex* tree with the bounding box of the data and parameters m and d (line 1). Then, it inserts each spatial entity into the *QuadFlex* (line 3) and finally returns its leaves. The methods $insert(s)$ and $getIndex(s)$ are self calls on the *QuadFlex* object ($this$). The insertion procedure is similar to the traditional quadtree

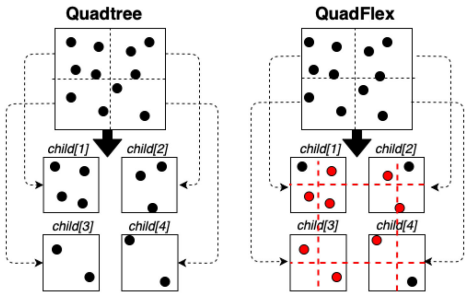


Fig. 2. QuadFlex versus quadtree.

except that the constraint is not the capacity but the diagonal of the area m (maximal distance between points) and the density of the area d. Hence, if the diagonal of the *QuadFlex* is more than the distance m or the density is larger than our defined value d (line 12), the *QuadFlex*, similarly to a quadtree, will split into four children. However, in contrast to the traditional quadtree, a spatial entity might belong to more than one child. The method $getIndex(s)$ gets the list of indexes of the children where the new point will be assigned. *Even though Q splits into 4 children in the same way as a quadtree, the lines $vertical-left$, $vertical-right$, $horizontal-up$, and $horizontal-down$ allow a logical overlap of the areas and thus, neighboring spatial entities will not be separated.*

5 Pairwise Comparisons

After the spatial blocking, we perform a pairwise comparison of spatial entities that fall in the same leaf. Next, we describe the metrics for different types of attributes.

Textual Similarity. We measure the textual similarity of spatial entities using the edit distance between the words. The Levenshtein distance [30] between string s_1 and string s_2 $d(s_1, s_2)$ is the number of edits (insertion, deletion, change of characters) needed to convert string s_1 to string s_2. We define the similarity as:

$$TextSim(s_1, s_2) = \left(1 - \frac{d(s_1, s_2)}{max(|s_1|, |s_2|)}\right). \quad (1)$$

Example 1. Let us consider "Skippers Grill" and "Skippers Grillbar". The Levenshtein distance to convert "Skippers Grill" to "Skippers Grillbar" is 3 (3 insertions). The lengths of the first and the second string are 14 and 17 respectively. So, $TextSim(\text{"Skippers Grill"}, \text{"Skippers Grillbar"}) = 1 - (3/max(14, 17) = 0.8235$.

Note here that not all textual attributes can be handled similarly. String similarity metrics are usually appropriate for attributes like names, usernames, etc. Some other textual attributes require other metrics that need to be customized. In this paper, we consider the address as a specific textual attribute. The similarity between two addresses cannot be measured with Levenshtein, Jaccard, Cosine, etc. since a small change in the address might be a giant gap in the spatial distance between the entities. For example, "Jyllandsgade 15 9480 Løkken" and "Jyllandsgade 75 9480 Løkken" have a distance of 1 and Levenshtein similarity of 0.963, but they are 650 meters apart. However, "Jyllandsgade 15 9480 Løkken" and "Jyllandsgade 15 9480 Løkken Denmark" have a distance of 8 and Levenshtein similarity of 0.772, but they are the same building. In [11], [12] the address is considered as another textual attribute. In our case, we perform some data cleaning (removing commas, punctuation marks, lowercase, etc.), and then we search for equality or inclusion of the strings. We assign a similarity of 1.0 in the case of equality, 0.9 in the case of inclusion, and 0.0 otherwise.

Semantic Similarity. The similarity of fields like categories, keywords, or metadata cannot be compared only syntactically. Sometimes, several synonyms are used to express the same idea. Thus, we need to find a similarity than considers the synonyms as well. We use Wordnet [31] for detecting the type of relationship between two words and Wu & Palmer similarity measure (wup) [32]. The semantic similarity between two spatial entities is the maximal similarity between their list of categories, keywords, or metadata. The semantic similarity of the spatial entities s_1 and s_2 is:

$$SemSim(s1, s2) = max\{wup(c_i, c_j)\}, \quad (2)$$

where $c_i \in C_1$ and $c_j \in C_2$ and C_1 is the set of keywords of s_1 and C_2 is the set of keywords s_2.

Example 2. Let us take an example of two spatial entities s_1 and s_2 and their corresponding semantic information expressed as keywords $C_1 = \{\text{"restaurant"}, \text{"italian"}\}$ and $C_1 = \{\text{"food"}, \text{"pizza"}\}$. The similarity between each pair is $wup(\text{"restaurant"}, \text{"food"}) = 0.4$, $wup(\text{"italian"}, \text{"food"}) = 0.4286$, $wup(\text{"restaurant"}, \text{"pizza"}) = 0.3333$ and wup $(\text{"italian"}, \text{"pizza"}) = 0.3529$. Finally, the semantic similarity of s_1 and s_2 is $SemSim(s1, s2) = max\{0.4, 0.4286, 0.3333, 0.3529\} = 0.4286$.

Date, Time, or Numeric Similarity. The similarity between two fields expressed as numbers, dates, times or intervals is a boolean decision (true or false). Even though the similarity of these fields relies only on an equality check, most of the effort is put in data preparation. For example, the different phone formats should be identified and cleaned from prefixes. Other data formats like intervals (opening hours) might require temporal queries for similarity, inclusion, and intersection of the intervals. In this paper, we do not compute the similarity between these attributes as we use them to construct the ground truth.

6 Ranking the Pairs

After the pairwise comparison, the pairs have n similarity values, one for each attribute. We denote as δ_a the similarity of two spatial entities for attribute a. For example, a pair $\langle s_1, s_2 \rangle$ is represented as $\{\delta_{a_1}, \ldots, \delta_{a_n}\}$. The problem that we need to solve is which $\langle s_i, s_j \rangle$ pairs indicate a strong similarity to be considered for a match. The related work solutions propose using a classifier [7], [8], [33] or experimenting with different thresholds [9], [18], [33]. We propose a more relaxed technique that uses Pareto optimality [34] for filtering the positive class. *A solution (x, y) is Pareto optimal when no other solution can increase x without decreasing y.* The points in the same Pareto frontier or skyline have the same utility. Widely used in economics and multi-objective problems, Pareto optimality is free of weights and similarity score functions. In the context of entity resolution, the skylines provide a selection of points that are better than others, but

without quantifying how much better. The pairs that refer to the same physical spatial entity (the positive class) are expected to have high values of δ, and consequently, form the first skylines. Under the assumption that the best values of δ belong to the pairs from the positive class, we label the pairs up to the kth skyline as the positive class and the rest as the negative. *To the best of our knowledge, we are the first to propose a Pareto optimal solution for detecting matches for an entity linkage problem.*

Definition 2. *An attribute a is positive discriminating if its similarity δ_a indicates a positive class rather than a negative.*

An example of a positive discriminating attribute is the similarity of name. A higher name similarity is more likely to indicate a match than a non-match. For example, the name similarity for *Mand & Bil* and *Mand og Bil* is 0.75, and for *Solid* and *Sirculus ApS* is 0.16 . Hence, the former pair has a higher probability of being a match than the second. Examples of negative discriminating attributes are the edit distance between two names. If the distance between the names is high, then the pairs are less likely to be a match.

Definition 3. *The utility of a positive discriminating attribute a, denoted as u_a, is the contribution of the attribute similarity δ_a to reveal a match, using Pareto Optimality ($\delta_a \overset{Pareto\ Optimality}{\longmapsto} u_a$).*

Each attribute similarity contributes to the labeling problem. Intuitively, a higher similarity δ_a of a has a higher utility than a lower value of δ_a. Hence, if $\delta_a(\langle s_1, s_2\rangle) > \delta_a(\langle s_3, s_4\rangle)$, then $u_a(\langle s_1, s_2\rangle) > u_a(\langle s_3, s_4\rangle)$.

Definition 4. *The utility of a pair denoted as $u(\langle s_i, s_j\rangle)$ is sum of the utilities of each of the attributes. $u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} u_{a_i}$.*

Note that the utility of a pair is not the sum of the similarities of the attributes ($u(\langle s_i, s_j\rangle) \neq \sum_{i=1}^{n} \delta_{a_i}$) but the sum of their utilities ($u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} u_{a_i}$). Nevertheless, $u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} \delta_{a_i} = \sum_{i=1}^{n} u_{a_i}$ is a specific case.

Definition 5. *A skyline of level k, $Skyline(k)$, is the collection of pairs $\langle s_i, s_j\rangle$ of equal utility such that $u_{Skyline(k)} > u_{Skyline(k+1)}$.*

Obviously, $Skyline(1)$ is the Pareto optimal frontier with the best values of δ_a. In order to continue with $Skyline(2)$, the points of $Skyline(1)$ are removed, and the frontier is calculated again. Every time we explore level k, the values in $Skyline(k)$ are the ones with the highest utility. This means that *there is no other point in a lower level that can bring a higher utility to the positive class*. This procedure continues until all the pairs are ranked according to their skyline. Algorithm 2 formalizes our proposed procedure Skyline Ranking (*SkyRank*) for ranking the pairs. The input is the set of pairs P produced from the *QuadFlex* blocking technique and the number of skyline levels k that we will explore. We find the points with the best combinations of δ that dominate the rest of the points and, consequently, have a higher utility (line 3). Then, we put these points in P_k, which keeps the explored skylines and remove them from P (line 5). We stop when all the pairs are assigned to a skyline.

After obtaining the ranking, we can assume that the pairs of the first few skylines are more likely to refer to the same physical entity than the rest.

Assumption 1. *The probability that a pair is labeled positive is inversely proportional to its skyline level.*

The assumption considers that for all $\langle s_i, s_j\rangle$and$\langle s'_i, s'j\rangle$ in P such that $\langle s_i, s_j\rangle \in Skyline(k)$, $\langle s'_i, s'_j\rangle \in Skyline(k')$ and $k < k'$, then $\langle s_i, s_j\rangle$ is more likely to be a match than $\langle s'_i, s'_j\rangle$.

7 Estimating k

In this section, we estimate the skyline level k that separates the positive from the negative class. We introduce two different methods for fixing the value of k: *threshold-based* (*SkyEx-F* and *SkyEx-FES*) and *unsupervised* (*SkyEx-D*).

Algorithm 2. Skyline Ranking (SkyRank)

input: A set of pairs $P = \{\langle s_i, s_j\rangle\}$
output: A set of pairs and their skyline $P_k = \{\langle s_i, s_j\rangle, k\}$;
1: $P_k \leftarrow \emptyset$
2: **while** $|P_k| < |P|$ **do**
3: Filter $Skyline(k) = \{\langle s_i, s_j\rangle\} \mid \forall\langle s', s''\rangle \in P - \{\langle s_i, s_j\rangle\}$, $u(\langle s_i, s_j\rangle) > u\langle s', s''\rangle\}$ // *Find the Skyline*
4: Add $Skyline(k)$ to P_k // *Move the skyline to P_k*
5: $P = P - Skyline(k)$
6: **end while**
return P_k

Algorithm 3. SkyEx-F

input: A set of pairs $P = \{\langle s_i, s_j\rangle\}$
output: A set of positive pairs P^+ and a set of negative pairs P^-;
1: $P_k \leftarrow \emptyset, F \leftarrow \emptyset$
2: **while** $|P_k| < |P|$ **do**
Lines 3-5 as Algorithm 2 ...
6: $P^+ \leftarrow P_k$
7: $P^- \leftarrow P$
8: Calculate $F1(k)$
9: Add $\langle k, F1(k)\rangle$ to F
10: **end while**
11: Find k_f such that $F1(k_f) = max(F1(k))\ \forall k \in \{1, |F|\}$
12: $P^+ \leftarrow \bigcup_{k=1}^{k_f} Skyline(k)$
13: $P^- \leftarrow P - P^+$
return P^+, P^-

7.1 SkyEx-F and SkyEx-FES

In contrast to the threshold-based methods used in entity resolution problems [12], [13], [18] where we have to find a threshold for each similarity of the attributes and then a threshold for the similarity function that aggregates the similarity scores, we have simplified our problem to only one parameter: k. We need to find the value of k that best separates the classes. As a measure of a "good model", we choose to use the *F-measure*, given that our data tends to be unbalanced [35], [36], [37]. In the context of our problem, we define true positives TP as pairs that refer to the same physical entity and are correctly labeled as positives; true negatives TN as pairs referring to different physical entities and are correctly labeled as negatives; false positive FP as pairs that do not refer to the same physical entities but are wrongly labeled as positives; FN as pairs that refer to the same physical entity but are wrongly labeled as negatives.

Thus, the precision is $p = \frac{TP}{TP+FP}$, the recall is $r = \frac{TP}{TP+FN}$ and $F - measure\ (F1) = 2\frac{p*r}{p+r}$.

The higher the k, the more unlikely it is for a pair in the kth skyline to belong to the positive class (Assumption 1). *SkyEx-F* explores the first skylines and stops at the value of $k = k_f$ that achieves the highest $F - measure$. To find k_f, we rank the pairs as in Algorithm 2, but we add some extra calculations within the loop (lines 6-9) and find the optimal k_f (line 7) in Algorithm 3. *SkyEx-F* calculates the *F-measure* for each skyline k by considering the pairs up to the kth skyline as positive and the rest as negative. We add $F1(k)$ to the set F, which keeps track of the evolution of *F-measure* while exploring more skylines. We find k_f as the value of k that achieves the highest *F-measure* in F. The pairs from the first to the k_f level of skyline are labeled as positive and the rest as negative. Note that *SkyEx-F* explores all the skylines and then, finds the threshold k_f. However, we can optimize Algorithm 3 by stopping at k_f before going through the full dataset P. Let us highlight some properties of *p* and *r*.

Property 1. *The recall is a monotonically non-decreasing function with respect to the number of skylines k.*

Proof. The recall after k skylines is $r(k) = \frac{TP(k)}{TP(k)+FN(k)}$. While we move to the next, $k+1^{th}$ skyline, we label more pairs as positive, so the probability of finding true positives TP is higher. Thus, $TP(k+1) \geq TP(k)$. As for the denominator, it is always the same despite the skyline level because the true positives are fixed in P and are independent of our labelling. This means that if we find more true positives (TP), then we automatically decrease the false negatives (FN). Hence, $TP(k+1) + FN(k+1) = TP(k) + FN(k)$. We can then show that $\frac{TP(k+1)}{TP(k+1)+FN(k+1)} \geq \frac{TP(k)}{TP(k)+FN(k)}$ so $r(k+1) \geq r(k)$. □

Property 2. *Given Assumption 1, the precision is a monotonically non-increasing function with respect to the number of skylines k.*

The precision is $\frac{TP}{TP+FP}$. However, $TP + FP$ is what our algorithm labels as positive, which means all the pairs belonging to skylines up to the kth level. According to Assumption 1, FP increase at a higher rate than TP while moving to higher k values. A proof of monotonic decreasing precision for systems that rank the results considering their relevance (like our skylines) can be found in [38].

Theorem 1. *The F-measure function with respect to the number of skylines k is increasing until a point or interval, and after that, it cannot increase again.*

Proof. Let us suppose that while moving deeper into the skylines, we found a peak point k or peak interval $[k_i, k_j]$ with $F1(k)$ as the corresponding *F-measure*. Note that for a peak interval the *F-measure* is constant. Since $F1(k)$ belongs to a peak point/interval, there exists a $F1(k+\epsilon)$ ϵ skylines after k such that $F1(k+\epsilon) < F1(k)$. Now, let us know suppose that we can find another optimum in $k+\delta$ such that $F1(k+\delta) > F1(k)$. Since $F1(k+\epsilon) < F1(k)$, consequently $F1(k+\delta) > F1(k) > F1(k+\epsilon)$. $F1 = 2\frac{p*r}{p+r}$ can be rewritten as $F1 = \frac{2}{\frac{1}{p}+\frac{1}{r}}$. So, we can rewrite: $\frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\delta)}} > \frac{2}{\frac{1}{p(k)}+\frac{1}{r(k)}} > \frac{2}{\frac{1}{p(k+\epsilon)}+\frac{1}{r(k+\epsilon)}}$. Using Property 2, $p(k+\delta) \leq p(k+\epsilon)$, so: $\frac{2}{\frac{1}{p(k+\epsilon)}+\frac{1}{r(k+\epsilon)}} \geq \frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\epsilon)}}$, which means that: $\frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\delta)}} > \frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\epsilon)}}$. According to Property 1, this inequality cannot hold, because $r(k+\delta) \geq r(k+\epsilon)$. Thus, our assumption of $F1(k+\delta) > F1(k)$ cannot hold and $F1(k)$ remains the highest value of *F-measure*. □

Algorithm 3a. SkyEx-F Early Stop (SkyEx-FES)

input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
output: A set of positive pairs P^+ and a set of negative pairs P^-;
1: $P_k \leftarrow \emptyset$, $F_{previous} \leftarrow 0$
2: **while** $|P_k| < |P|$ **do**
    Lines 3-8 as Algorithm 3...
9: **if** $F1(k) < F_{previous}$ **then**
10: **break**
11: **else**
12: $F_{previous} \leftarrow F1(k)$
13: **end if**
14: **end while**
15: $P^+ \leftarrow \bigcup_{k=1}^{k_f} Skyline(k)$
16: $P^- \leftarrow P - P^+$
  return P^+, P^-

Theorem 1 ensures that once we find the peak in the *F-measure* function, we can stop finding all the skylines and label the pairs accordingly. Consequently, we can allow Algorithm 3 to stop early. The modifications of Algorithm 3 are reflected in Algorithm 3a. We use the same procedure as in Algorithm 3, but we do not need to keep track of each of the skylines and their corresponding $F - measures$. Rather, we only keep the previous $F - measures$ in $F_{previous}$. While moving to the next skyline, we calculate the $F - measure$ and the first time we notice a drop (line 9), we stop the loop (line 10) and return both classes separated by the current k (lines 7-8)). Otherwise, we update $F_{previous}$ to the current $F - measure$ (line 12) and continue the search for the optimal k.

7.2 SkyEx-D

The methods described in the previous sections assume that the labels of the pairs are present. In this section, we assume no information about the labels, and thus, we propose a heuristic for fixing the value of k. The heuristic is based on the distance between the positive and the negative class. We refer to the k discovered by *SkyEx-D* as *distance-based k* or k_d. Our classes are not characterized by a small intra-class distance. Various patterns can reveal a positive class; for example, a similar name but different category or similar category and similar address, etc. Thus, the positive pairs, positioned in the first skylines, are scattered and do not necessarily form a cluster. However, they can still be separated from the rest, considering the distance to the negative class. Theoretically, the inter-class distance stays small when we are in the first skylines (potential positives), then starts to increase while we move into later skylines and finally falls again when we enter the deeper skylines (potential negatives). *SkyEx-D* notices the increase of the inter-class distance and sets k_d accordingly. In order to have an approximation of the inter-class distance, we use the mean and denote it as μ_d as in

$$\mu_d = \frac{\sum d(p^k, p^{-k})}{|P_k|}, \tag{3}$$

where $|P_k|$ is the number of pairs from the 1st to the kth skyline, p^k is a pair in P^k, p^{-k} is a pair in $P - P^k$, and $d(p^k, p^{-k})$ is the distance between p^k and p^{-k}.

In order to fix k_d, we monitor the value of μ_d while moving deeper into the skylines. We denote by $\mu_d(k)$ the function of μ_d with regard to k. We use the first derivative of $\mu_d(k)$, denoted as $\mu'_d(k)$, to find the points where the $\mu_d(k)$ function decreases. The intuition behind this approach is that in the beginning, the distance $\mu_d(k)$ starts increasing, which means that the first derivative has a positive slope ($\mu'_d(k) > 0$). Later, we enter the "grey area", where there is a mix of potential positives and potential negatives. This is where we need to stop because we might lose precision if we continue further. In order to find the "grey area", we note when the first derivative changes its slope to negative. In order to calculate $\mu'_d(k)$, we estimate the value of $\mu'_d(k)$ in each point k as in:

$$\mu'_d(k) = \frac{\partial}{\partial k} \approx \frac{\mu_d(k+1) - \mu_d(k)}{1}. \tag{4}$$

In order to not be sensitive to small fluctuations in $\mu'_d(k)$, we smoothen slightly $\mu'_d(k)$ with Gaussian function $((\frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}))$ using a small window. Then we monitor when $\mu'_d(k)$ decreases for the first time and we set k_d accordingly. We modify Algorithm 2 to accommodate this approach. We calculate $\mu'_d(k)$ for each point of k in line 7. Then, we have to find the first negative value of the smoothened $\mu'_d(k)$ (line 9) and fix k_d accordingly (line 10). Finally, we return the classes defined by k_d in lines 16-17.

Algorithm 4. SkyEx-D

input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
output: A set of positive pairs P^+ and a set of negative pairs P^-;
1: $P_k \leftarrow \emptyset$
Lines 2-6 as Algorithm 2...
7: Calculate $\mu'_d(k)$ in each k
8: **while** $k < k_{last}$ **do**
9: **if** $smooth(\mu'_d(k)) < 0$ **then**
10: $k_d \leftarrow k$
11: **break**
12: **else**
13: $k \leftarrow k + 1$
14: **end if**
15: **end while**
16: $P^+ \leftarrow \bigcup_{k=1}^{k_d} Skyline(k)$
17: $P^- \leftarrow P_k - P^+$
return P^+, P^-

Summary. Algorithm 4 estimates the skyline level k that best separates the positive class from the negative class. *Similarly to clustering techniques that use heuristics to estimate their parameters, SkyEx-D uses the distance of the positive class from the rest as an indicator of class separability*. However, in contrast to clustering metrics, which focus on the robustness of clusters, this is not a requirement for the *SkyEx-** family of algorithms. *The positive pairs do not show similar patterns, but rather similar utilities, which can be better captured by skylines* (see Section 9.9). Experimentally, we show that our inter-class distance approach estimates k_d very close to k_f without loosing in *F-measure*. In contrast to techniques that use a scoring function, the *SkyEx-** family of algorithms abstracts the concept of utility. Thus, no weights or similarity function is needed. Even though the positive class can be characterized by various patterns of attribute similarities, the *SkyEx-** family of algorithms can still group together the positive class based on the high utility, while a clustering technique would instead focus in grouping each pattern separately, without putting the positive-class pairs together into one cluster. Moreover, the flexibility of the *SkyEx-** family of algorithms makes it applicable to all problems where the expert knowledge on the contribution of the attributes is missing. Finally, the *SkyEx-** family of algorithms does not learn any behavior, so there is no risk of overfitting.

8 Complexity Analysis of QuadSky

In this section, we discuss the time complexity of our algorithms and of our *QuadSky* solution.

QuadFlex deals with points (not regions); thus, it behaves similarly to a point quadtree. *QuadFlex* splits the same way as a quadtree, but in contrast to the quadtree, the points can be assigned to more than one child. We construct the *QuadFlex* structure only for forming the blocks. Hence, the construction complexity is of interest to us. Let us denote by $|S|$ the number of points in S, c the smallest distance between any two points, and D_1 and D_2 the dimensions of the initial area containing all the points. Let us first estimate the depth of *QuadFlex*. The distance c of any two points p_1 and p_2 in *QuadFlex* is always less than the diagonal of the node they belong in. Given that *QuadFlex* allows neighboring points to be included in more than one child, this calculation needs to be modified. The physical diagonal of the initial (level 0) node is $\sqrt{D_1^2 + D_2^2}$. The diagonal of level i is $\frac{\sqrt{D_1^2+D_2^2}}{4^i}$. To modify the calculation, we estimate the logical diagonals if *QuadFlex* would physically expand to accommodate neighboring points, so: $c \leq \frac{\sqrt{\frac{3D_1^2}{2}+\frac{3D_2^2}{2}}}{4^i}$ Now, isolating i out of this equation results in $i \leq \log_4 \frac{\sqrt{\frac{3}{2}(D_1^2+D_2^2)}}{c} = \log_4 \sqrt{\frac{3}{2}} + \log_4 \frac{\sqrt{D_1^2+D_2^2}}{c}$. $\log_4 \sqrt{\frac{3}{2}} \approx 0.14$ so we can discard it (less than one level): $i \leq \log_4 \frac{\sqrt{D_1^2+D_2^2}}{c}$. For estimating the maximal depth, we need to add one more level (root) so the depth is estimated as $\log_4 \frac{\sqrt{D_1^2+D_2^2}}{c} + 1$. Finally, for constructing *QuadFlex*, the complexity is $O(|S|(\log_4 \frac{\sqrt{D_1^2+D_2^2}}{c} + 1))$.

SkyRank requires calculating the Pareto frontiers, which is time-consuming. In the typical case, comparing the pairs P resulting from *QuadFlex* in terms of all d dimensions has a $O(2^{|P|^d})$ time complexity [39], which is not scalable. *SkyRank* uses the method proposed in [40], which first scales down the d-dimensional domain and then pre-filters the data using a lattice. This yields a time complexity of $O(|P|^2)$ for the first skyline. For the total number of K skylines, the complexity is $O(K|P|^2)$.

SkyEx-F calculates the metrics while adding the next skyline to the positive class; thus, these calculations do not add any complexity. Finally, we perform a linear search on F to

find the skyline with the highest F-measure. The size of F is equal to K, so the complexity is $O(K|P|^2 + K)$.

SkyEx-FES stops earlier than *SkyEx-F*, avoiding a big part of the time-consuming Pareto calculations. Given that the best pairs usually are focused on the first skylines, the cut-off $k \ll K$. Moreover, according to Theorem 1, we do not need to store F, so we avoid the linear search for the best F-measure. The complexity is $O(k|P|^2)$.

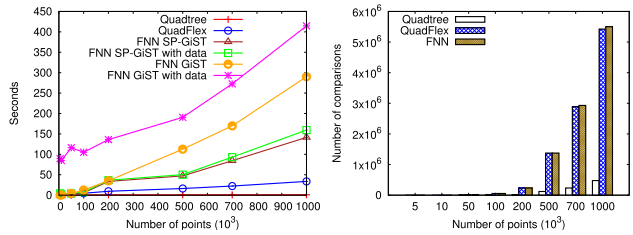
SkyEx-D uses all K Pareto calculations and then, in order to estimate the cut-off k_d, it computes the distance between the positive class and the rest. *SkyEx-D* creates a matrix where the rows are the positive class P^+ and the columns are the negative class data points $|P| - P^+$, so the complexity is $P^+ * (|P| - P^+)$. $P^+ * (|P| - P^+) = P^+ * |P| - (P^+)^2$ is the equation of a vertical parabola that opens downwards $-ax^2 + bx + c$, with the maximal value at the vertex $(-\frac{b}{2a})$. In our case, the maximum of $P^+ * (|P| - P^+)$ is at $\frac{|P|}{2}$, resulting in a maximal complexity of $\frac{|P|^2}{4}$. For each skyline k in K, the maximal complexity is $\frac{|P|^2}{4}$, thus, $K\frac{|P|^2}{4}$ for all. Note here that $K \ll |P|$ so it is far from a cubic complexity. SkyEx-D computes the mean distance μ_d for each k, which can already be done within the $\frac{|P|^2}{4}$ complexity. Then, we compute the derivative μ'_d on the means, which has a linear complexity in K. Finally, we need another partial scan until k_d ($k_d \ll K$) where the derivative μ'_d becomes negative for the first time. Hence, the total complexity is $O(K|P|^2 + K\frac{|P|^2}{4} + K + k_d) = O(\frac{5K}{4}|P| + K + k_d)$.

Summary. *QuadFlex* has $O(n \log n)$ complexity, the pairwise comparisons have a linear $O(n)$ complexity, while the *SkyEx-** family of algorithms have a quadratic complexity $O(n^2)$. However, there is a theoretical risk of a cubic complexity in *SkyEx-F* and *SkyEx-D* if the number of skylines $K = |P|$. This means that each skyline in K contains only one pair of entities, which theoretically can happen but almost never happens in practice. Thus, the algorithms have quadratic complexity in the average case. *SkyEx-D* has the highest complexity, followed by *SkyEx-F* and *SkyEx-FES*. Overall, *QuadSky* has a quadratic complexity.

9 Experiments

9.1 Dataset Description

The spatial entities that will be used in these experiments originate from four sources, namely Google Places (GP), Foursquare (FSQ), Yelp, and Krak. Krak (www.krak.dk) is a location-based source that offers information about companies, enterprises, etc. in Denmark and is also part of Eniro Danmark A / S., which publishes The Yellow Pages. The data is obtained by using the available APIs and the algorithm detailed in [41]. The dataset consists of 75,541 spatial entities where 51.50 percent comes from GP, 46.22 percent from Krak, 0.03 percent from FSQ, and 2.23 percent from Yelp (see Supp. Material Annex A for the spread of these spatial entities on the map). The dataset is 69 MB. For a 100 m blocking, there are 35,521 spatial entities that have at least one positive match in the dataset, resulting in 27,102 pairs that need to be discovered. 7,795 of these pairs are within the same source, which shows that none of these sources are free of duplicates. 3,546 of the same-source links come from GP, 3,789 from Krak, and 460 from Yelp. As for the different-source links, all the sources overlap with each other, but the highest overlap of 17,405 pairs (90 percent of different-source links) comes from Krak and GP.



(a) Execution time (b) Number of comparisons

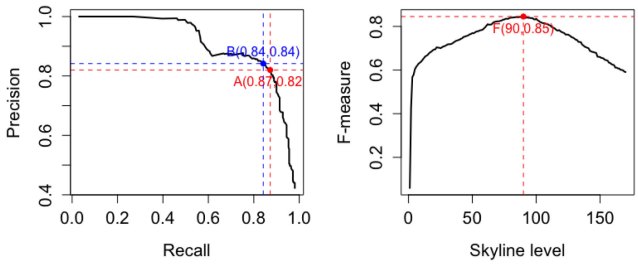
Fig. 3. Comparing quadtree, QuadFlex and FNN.

9.2 QuadFlex Performance

In this section, we compare the performance of *QuadFlex* to the quadtree, Fixed Radius Nearest Neighbors algorithm [42] (FNN), and having no index at all (No-Index). FNN finds the neighbors that fall within a fixed radius from each point. *QuadFlex* and the quadtree algorithm are implemented in Java, while FNN is run on a Postgres database (https://www.postgresql.org) using spatial indexes: GiST (optimized C implementation of B-trees and R-trees) and SP-GiST (optimized C implementation of quadtrees and k-d trees). Our dataset contains 75,541 entities in the North Denmark region (around 16 towns, 7,933 km^2), so the average density is not high, even though there are areas with high density. A high data density means more pairs to compare. To test our *QuadFlex* on different data densities, we simulate up to 1,000,000 random points from Aalborg (139 km^2). Fig. 3 shows the comparison of quadtree, *QuadFlex* and FNN in terms of execution time (Fig. 3a) and number of comparisons (Fig. 3b). The FNN versions with data are computed on the database, and then the pairs are loaded back to the java implementation. The quadtree has the lowest execution time, followed by *QuadFlex*. FNN SP-GiST is comparable and sometimes even better than *QuadFlex* for small datasets. However, when the size of the dataset increases, *QuadFlex* maintains an execution time that is eight times less than FNN GiST and 3 times less than FNN SP-GiST. FNN with SP-GiST index outperforms FNN GiST for all dataset sizes. No-Index was very inefficient, up to 848 times slower than FNN Gist with data, and up to 368,095 times slower than *QuadFlex*. Given that No-Index would have dwarfed the other curves, it is not part of Fig. 3a, but instead, refer to Fig. 2 in Supp. Material, Annex B. As for the number of comparisons, *QuadFlex* enumerates 12 times more comparisons than quadtree. Moreover, *QuadFlex* contains almost all (99.99 percent) comparisons of FNN, compared to the quadtree that contains only 10 percent of FNN. Furthermore, given that the scalability of *QuadFlex* is better than FNN, and *QuadFlex* is independent of the database implementations, the loss of around 0.01 percent of comparisons is insignificant.

9.3 SkyEx-F Results

We ran *QuadFlex* with 100 m and no density restriction, and we obtained 777,452 pairs (1426 MB). Having the same website or phone is a strong indicator of a match, so we use

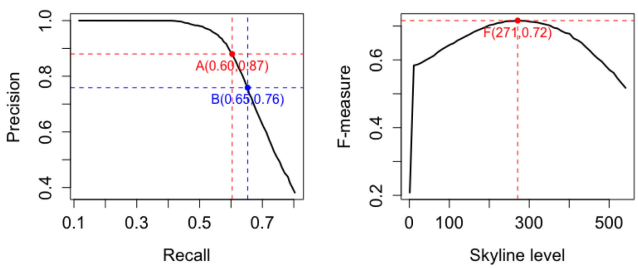


(a) precision and recall (b) F-measure

Fig. 4. SkyEx-F performance on D_{sample}

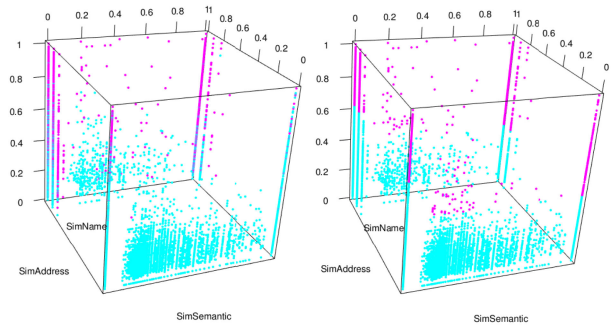
these attributes to infer the label. We refer to this labeling as *automatic labeling*. However, cases with different phone number or website but still the same entity, or same phone number but different entity might occur. Hence, we manually checked the labels of a sample of 1,500 pairs of entities (1552 kB). We will refer to the sample of manually checked pairs as D_{sample} and to the full dataset as D_{full}. Checking the labels manually on the full dataset of 777,452 pairs is unfeasible. Hence, we checked around 10,000 of the pairs, and for the rest, we rely on automatic labeling.

The results of *SkyEx-F* on D_{sample} and D_{full} are presented in Figs. 4 and 5. The curves in Figs. 4a and 5a shows the evolution of p (y-axis) and r (x-axis) while we move from one skyline to the next. *The more we explore, the more likely it is to retrieve more true positives and thus, improve the* r. However, *the more we explore and label pairs as positives, the more likely it is to increase the number of false positives, so the* p *degrades*. The algorithm explores several trade-offs; for example the points A and B are among the best. The point A with 0.87 p and 0.82 r in Fig. 4a is the same best point in terms of *F-measure* as well, so that is where *SkyEx-F* will fix k_f. Fig. 4b shows the levels of the skyline, and the value of *F-measure* achieved. The highest value is 0.85 that corresponds to $k = 90$. The evaluation on the full dataset yields lower values (*F-measure* of 0.72) compared to the sample (*F-measure* of 0.85), which might be a simple consequence of automatic labeling. Point A has 0.6 r and 0.87 p, while B offers a higher r of 0.65 but a lower p of 0.76 (Fig. 5b). To have an idea of the real classes in D_{full} and the skylines, we plotted their distribution in Fig. 6 (the actual positive classes in pink and the negative ones in sky blue). It is noticeable that the positive class pairs are allocated in the highest values of the dimensions. Despite the differences between both plots, *SkyEx-F* shows promising results in separating the positive class from the negative one with 0.6 r and 0.87 p.



(a) precision and recall (b) F-measure

Fig. 5. SkyEx-F performance on D_{full}



(a) Actual classes (b) SkyEx classes

Fig. 6. Positive (in pink) versus negative (in sky blue) classes for actual (a) and SkyEx-F (b) results.

9.4 Experimenting With Different QuadFlex Parameters

So far, we used *QuadFlex* blocking technique with 100 meters and no density restriction. In this section, we will evaluate our approach *QuadSky* for different blocking parameters.

Changing m, no Density Limit. In this experiment, we test different values of m used in *QuadFlex* for creating spatial blocks. We test m values of 1, 20, 40, 60, 80, and 100 meters. The size of the dataset for each of them is presented in Table 2. The spatially closeby points are likely to be a match. Hence, the percentage of the true positives is generally higher for smaller values of m. An interesting case is $m = 1$, where the percentage of the true positives (*TP*) is lower than $m = 20$. One would expect that points that are 1 meter apart would unquestionably be a match. However, this is not always the case. Shopping malls, buildings that host several companies, etc. are characterized by the same coordinates but not necessarily the same spatial entities. The results for different values of m are presented in Table 2 (see the precision-recall graphs for all cut-offs in Supp. Material, Annex C). For all cases, the r is higher than 0.6. The p is higher than 0.8 for all values of m, except $m = 1$, where the p is 0.67. For $m = 1$, the positive and negative class are mixed, thus *SkyEx* loses a bit in p. This is also an argument against the works that merge arbitrarily points that are 5 m apart. *Spatial proximity is not a definitive indicator of a match*.

Changing d, m ≤ 100. We experiment with different values of density d and its effect on the results. The size of the dataset, the percentage of the true positives, and the results in terms of precision, recall, and *F-measure* are in Table 3 (see the precision-recall graphs for all cut-offs in Fig. 9 in [14]). When the density is smaller, we force *QuadFlex* to split further and create smaller blocks. Thus, the number of pairs reduces. Note that, on the contrary, the percentage of the

TABLE 2
SkyEx-F Results for Different m

Meters	1	20	40	60	80	100
Total	41053	118437	226331	372553	557421	777452
% of TP	17.11%	19.88%	11.28%	7.06%	4.82%	3.49%
Prec.	0.67	0.80	0.85	0.85	0.88	0.87
Rec.	0.60	0.69	0.65	0.64	0.61	0.61
F1	0.64	0.75	0.74	0.73	0.72	0.72

TABLE 3
SkyEx-F Results for Different D

Den.	$\frac{10s}{1000m^2}$	$\frac{20s}{1000m^2}$	$\frac{30s}{1000m^2}$	$\frac{40s}{1000m^2}$	$\frac{50s}{1000m^2}$	$\frac{60s}{1000m^2}$
Total	290653	590583	711423	754195	770987	776664
% of TP	8.61%	4.57%	3.81%	3.59%	3.51%	3.49%
Prec.	0.88	0.88	0.87	0.87	0.87	0.87
Rec.	0.63	0.61	0.61	0.61	0.61	0.61
F1	0.74	0.74	0.72	0.72	0.72	0.72

TABLE 4
Skyline Explorations of *SkyEx-FES* Compared to *SkyEx-F* for Pairs That are 30, 50, 80, and 100 m Apart

Distance	30 m	50 m	80 m	100 m
Number of pairs	168193	293833	557421	777452
% of TP	14.76%	8.8%	4.82%	3.49%
***SkyEx-F*skylines**	1113	1182	1228	1228
***SkyEx-FES*skylines**	403	327	284	274

true positives (*TP*) increases. Indeed, further splits allow us to create better blocks containing a higher percentage of *TP*. However, when the density limit increases above $\frac{30s}{1000m^2}$, fewer and fewer blocks are split further, so the dataset size and the percentage of the *TP* do not vary significantly. In all the cases, the r stays above 0.61 and the p above 0.87. A slightly better p (0.88) and r (0.63) is achieved in the case of a density of $\frac{10s}{1000m^2}$ (the lowest parameter). *SkyEx-F adapts very well in finding the correct classes even when the size of blocks changes and even when the percentage of the true positives over the true negatives varies.*

9.5 SkyEx-FES Optimization

Given the theoretical guarantee in Theorem 1, we can stop *SkyEx-F* earlier as described in Algorithm 3a. We ran *SkyEx-FES* for spatial entities that are 30, 50, 80, and 100 m apart. For all the cases, *SkyEx-FES found the same k_f values as SkyEx-F exploring only 27 percent of the skylines on average.* The comparison regarding the number of iterations is shown in Table 4. For spatial entities that are 30 m, 50 m, 80 m, and 100 m apart, *SkyEx-FES* finds the optimal k_f exploring 36, 27, 23, and 22 percent of the skylines, respectively. Moreover, our theoretical guarantee that the *F-measure* function has only one optimum can also be noticed in Figs. 4b and 5b.

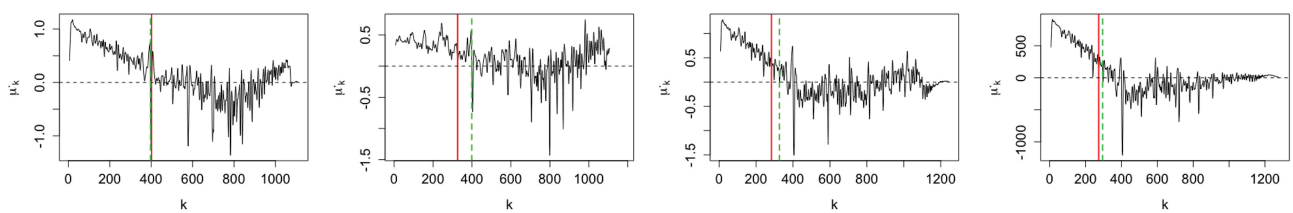
9.6 SkyEx-D Performance

In these experiments, we use *SkyEx-D* (Algorithm 4) to set k_d and evaluate our results in terms of *F-measure*. We apply *SkyEx-D* on spatial entities that are 30, 50, 80, and 100 meters apart (see the dataset details in Table 4). We calculate the first derivative (μ'_d) in each point as in Algorithm 4. The smoothed $\mu'_d(k)$ with respect to k are presented in Fig. 7. The red solid line shows the value of k_f, while the green dashed line represents k_d found by *SkyEx-D*. We note when $\mu'_d(k)$ is negative for the first time and set k_d accordingly. In the case of spatial entities that are 30 m apart (Fig. 7a), k_d is only 5 skylines apart from k_f but 73 skylines for 50 m. These values of k_d are discovered using the first derivative (Eq. 4, Section 7.2). We illustrate the trend of μ_k while increasing k, which means that we explore deeper skylines and examine more pairs that are less likely to be a match. The distance from the positive class to the negative is smaller in the beginning because the mean μ_k is biased by the close points. While we increase k, μ_k increases, meaning that the classes are becoming more and more distinguishable from one another. The high values of μ_k show a high distance between the classes. For spatial entities that are 80 m and 100 m apart, μ_k starts dropping faster than for those that are 30 m and 50 m apart (Fig. 8). This observation can be justified by the fact that closeby entities are more difficult to classify, so the "grey" area of the potential cut-off is larger. However, *SkyEx-D* detects the first decrease in μ_k from the first derivative and fixes k_d. Graphically, this point coincides with the beginning of the "grey" area. Even though k_d is sometimes fixed far from k_f (m=50), the corresponding *F-measures* are almost the same (Fig. 9). The red line in Fig. 9 corresponds to k_f and the green line to k_d. The difference in *F-measure* is 0.002 for 30 m, 0.009 for 50 m, 0.002 for 80 m, and 0.004 for 100 m. Thus, *the difference in F-measure for the classifying the pairs using k_d instead of k_f is always less than 0.01.* This means that our *SkyEx-D*, even though fully unsupervised, is almost optimal in terms of *F-measure*.

In terms of precision, recall and *F-measure*, *QuadSky* with *SkyEx* in [14], *QuadSky* with *SkyEx-F*, and *QuadSky* with *SkyEx-FES* report the same values. However, the underlying algorithms are different. *SkyEx* in [14] needs the threshold k to separate the skylines, whereas for *SkyEx-F* and *SkyEx-FES*, there is no need for specifying k because the algorithms will fix it through the skyline explorations (only 30 percent of the skylines for *SkyEx-FES*). *QuadSky* with *SkyEx-D*, being fully unsupervised, might yield different results. The optimal scenario is if it fixes k_d as the k_f.

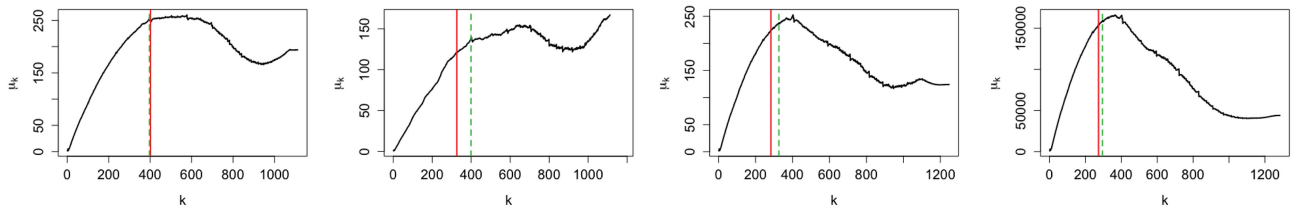
9.7 Comparison With Baselines

Even though there are several papers in spatial data integration, the works of [11], [12], [13] are the most similar to ours, as the rest of the related work considers only spatial objects, not spatial entities, or uses supervised learning techniques.



(a) m=30 (b) m=50 (c) m=80 (d) m=100

Fig. 7. Setting k_d using μ'_d.



(a) m=30 (b) m=50 (c) m=80 (d) m=100

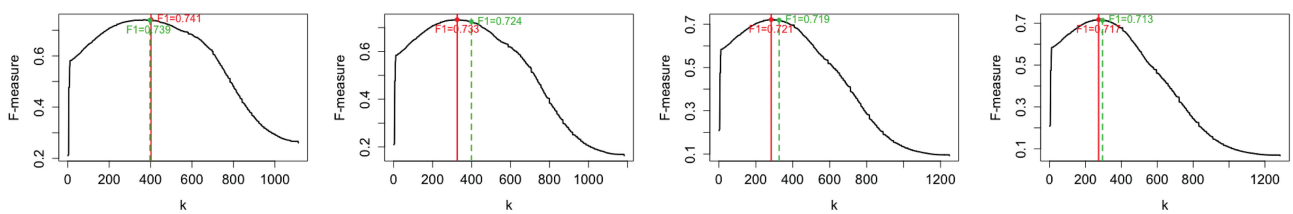
Fig. 8. $\mu_d(k)$ function with respect to k.

We will compare *QuadSky* to Berjawi *et al.* [12], Morana *et al.* [13], and Karam *et al.*[11]. Berjawi *et al.* [12] propose euclidean distance for the geographic coordinates and Levenshtein similarity for all other attributes. The similarities added together to a global similarity. The attributes mentioned in the paper are the name and the phone. However, since the phone is part of our automatic labeling, it can not be used in the algorithm as well. The authors consider pairs with $score \geq 0.75$ as a match with high confidence. We use this threshold but also try other thresholds that might yield better results (the versions with the suffix *-Flex*). We compare against two versions proposed by the authors: name + address + geographic coordinates (V1) and name + geographic coordinates (V2). Morana *et al.* [13] suggest filtering entities that share the same category or a token in the name. Then these entities are compared using the euclidean distance for the coordinates, Levenshtein for the address and name, and Resnik similarity (Wordnet) for the category. Attributes like address, phone, etc. are considered secondary, so they are given $\frac{1}{3}$ of the weight in the similarity score function, while name, category, and geographic proximity carry $\frac{2}{3}$ of the weight. The authors show top k matches for each entity to the user to decide. Karam *et al.* [11] starts with filtering spatial entities that are 5 m apart. Then, the similarity of the name is measured with Levenshtein distance, the geographic similarity with euclidean distance and the keywords are compared semantically. In order to decide which pairs to match and which not, the similarities are fused using belief theory [26].

The results using D_{full} and D_{sample} are presented in Table 5. In general, all the methods performed better in D_{sample} due to the better quality of the labels. Berjawi *et al.* (V2)[12] yields reasonable results, the second best after *QuadSky*, with an *F-measure* of 0.63 in D_{full} and 0.74 in D_{sample}. If we allow flexible thresholds, Berjawi *et al.*(V2) [12]-*Flex* in D_{full} finds the same best threshold of 0.75, whereas in D_{sample} the threshold of 0.65 yields better results, increasing the $F-measure$ from 0.74 to 0.79 (see Supp. Material, Annex D for all the thresholds and their results). To compare with Morana *et al.*[13], we tried all values k from 1 to the maximal matches for a single point (see Fig. 10 in [14]). The highest value of $F-measure$ corresponded to a p of 0.39 and a r of 0.60. The behavior of Morana *et al.*[13] in D_{sample} is similar; the best value of $F-measure$ was achieved for $k=3$ and results are similar to those in D_{full}. The work of Karam *et al.*[11] achieves the highest r of 0.73 but a very low value of p of 0.23 for D_{full}. As a result, the $F-measure$ is only 0.47. However, in D_{sample}, the method performs better overall ($F-measure$ =0.6).

The QuadSky versions provide the best trade-off between p and r, and thus, the highest $F-measure$ in both datasets. In D_{sample}, *QuadSky* with *SkyEx-F* and *QuadSky* with *SkyEx-D* achieve the best r compared to all baselines. What is more important, *QuadSky with SkyEx-D, even using an unsupervised algorithm, is still better than the threshold-based baselines.* The highest p values for both datasets is achieved by Berjawi *et al.*(V1)[12] but a very low r and poor model performance overall. In fact, models that achieve extreme values (high precision-low recall or low precision-high recall) are not a viable solution because they are either too restrictive or too flexible, and their predictability is poor. Berjawi *et al.* [12] (V2)-*Flex* assumes the same weights for all similarities, and the reported values of p and r are good. However, *the behaviors of the pairs can be of all types. QuadSky can capture these different behaviors better than a simple sum would.*

Regarding the complexity of the baselines, we cannot judge in terms of the blocking techniques because there are no details on whether the authors used an index to create the blocks. However, as we show in Fig. 3, the available FNN solutions in Postgres still do not scale as well as our *QuadFlex*. Therefore, we perform better in the blocking step. The pairwise comparison has a linear complexity for all baselines and our solution. As for the labeling, the baselines do not need the quadratic complexity induced by our skylines. Our *SkyEx-** family of algorithms run for 1 minute in D_{sample} and up to 2 hours in D_{full} with 777,452 pairs. Nevertheless, the entity linkage problem is performed offline, and consequently, even though a fast solution is preferable in



(a) m=30 (b) m=50 (c) m=80 (d) m=100

Fig. 9. *F-measure* values for different k.

TABLE 5
Comparison With the Baselines

Approach	D_{full}			D_{sample}		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Berjawi et al.(V1)[12]	**0.93**	0.26	0.41	**1.00**	0.27	0.43
Berjawi et al.(V1)[12]-*Flex*	0.87	0.50	0.63	0.79	0.42	0.55
Berjawi et al.(V2)[12]	0.73	0.56	0.63	0.97	0.60	0.74
Berjawi et al.(V2)[12]-*Flex*	0.73	0.56	0.63	0.82	0.76	0.79
Morana et al.[13]	0.39	0.60	0.47	0.33	0.60	0.43
Karam et al.[11]	0.23	**0.73**	0.35	0.54	0.68	0.60
QuadSky with *SkyEx-F*	0.87	0.60	**0.72**	0.87	**0.82**	**0.85**
QuadSky with *SkyEx-D*	0.85	0.62	0.71	0.87	**0.82**	**0.85**

general, the effectiveness is much more important, and here *QuadSky* significantly outperforms the baselines.

9.8 Comparison With Supervised Learning Techniques

In this section, we keep our *QuadSky* steps but replace the labeling of the pairs with a supervised learning technique. We decided to compare the *SkyEx-** family of algorithms with logistic regression [43], support vector machines (SVM) [44], decision trees [45], and Naive Bayes [46], which are supervised learning techniques commonly used in entity resolution problems [8], [10], [25], [33], [47]. We applied these methods on D_{full} pairs that are at most 30 meters apart (dataset description in Table 3). We experimented with training on 75 percent of D_{full} and testing on the remaining 25 percent with 4-fold cross validation (D_{full}-D_{full}), training on 75 percent of D_{sample} and testing on the remaining 25 percent with 4-fold cross validation (D_{sample}-D_{sample}), and training on D_{sample} and testing on D_{full} (D_{sample}-D_{full}). The results are presented in Table 6. While logistic regression and SVM yield a slightly higher *F-measure* of 0.76 in D_{full}-D_{full}, our algorithms, which do not build their model on labeled data, have almost the same *F-measures* (0.74 for *SkyEx-F* and *SkyEx-D*) in D_{full}-D_{full}. For the manually labeled dataset in D_{sample}-D_{sample}, our algorithms perform the second best (*F-measure* of 0.84), after the decision trees. *SkyEx-F* and *SkyEx-D* outperform the logistic regression, SVM, and the Naive Bayes, which yield *F-measures* of 0.81, 0.81, and 0.72, respectively. Having a large training set as in D_{full}-D_{full} is unrealistic in most real cases. Thus, we tried a more realistic scenario, where one would prepare a small manually labeled training set, and then, test the trained model on the full data (D_{sample}-D_{full}). *In this (most realistic) case, SkyEx-F and SkyEx-D outperform all supervised methods by 0.03-0.05 in F-measure, showing the main weakness of supervised models, namely that the D_{sample} model is not representative enough when applied to D_{full}.*

TABLE 6
Comparison With Supervised Learning

Method	D_{full} -D_{full}			D_{sample}-D_{sample}			D_{sample}-D_{full}		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
Log. reg.	0.83	0.70	**0.76**	0.80	0.83	0.81	0.70	0.72	0.71
SVM	**0.88**	0.67	**0.76**	0.81	0.80	0.81	0.71	0.70	0.71
Dec. Tree	**0.88**	0.66	0.75	**0.93**	0.82	**0.87**	0.65	0.74	0.69
Naive B.	0.71	**0.77**	0.74	0.63	**0.85**	0.72	0.62	**0.77**	0.69
SkyEx-F	0.80	0.69	0.74	0.87	0.82	0.84	0.80	0.69	**0.74**
SkyEx-D	0.81	0.68	0.74	0.87	0.82	0.84	**0.81**	0.68	**0.74**

TABLE 7
Comparing SkyEx-D to Clustering Techniques

Method	D_{full}			D_{sample}		
	Prec.	Rec.	F1	Prec.	Rec.	F1
K-means	0.28	0.96	0.44	0.62	0.92	0.74
K-medoids	0.28	0.96	0.44	0.62	0.92	0.74
Hierarchial	0.62	0.11	0.19	0.23	0.91	0.36
DBSCAN	0.23	**1.00**	0.37	0.26	**1.00**	0.42
SkyEx-D	**0.81**	0.68	**0.74**	**0.87**	0.82	**0.84**

In general, the spatial entity linkage problem suffers from the lack of labeled data [12], [13], [14]. Consequently, the applicability of supervised learning techniques is limited. On the contrary, *SkyEx-D* is completely unsupervised and can still achieve results similar to a supervised technique. If the labeled data is present, note that supervised learning techniques build the model on the labeled data, whereas *SkyEx-F* and *SkyEx-FES* use the labels only to tune the threshold because the construction of the skylines is independent of the labels. For this reason, *in contrast to supervised learning, SkyEx-F, and SkyEx-FES do not require a big and representative training set, do not struggle with class imbalance, do not overfit the data, and their dimensionality is minimal (one skyline versus high-dimensional data).*

9.9 Comparison of SkyEx-D to Clustering Techniques

In Section 7.2, we claimed that clustering techniques would not manage to create two clusters: one for the positive-class pairs and one for the negative-class pairs. In this section, we will replace *SkyEx-D* with common clustering techniques and evaluate the formed clusters. We are comparing to distance-based clustering (k-means [48] and k-medoids [49]), hierarchial clustering [50] (agglomerative), and density-based clustering (DBSCAN [51]). The results are presented in Table 7. For k-means and k-medoids, we specified the number of clusters as 2. For the hierarchical clustering, we cut the dendrogram to create two clusters. For DBSCAN, we tried several values of minimum points and ϵ to form either two clusters, or one cluster and noise points. We report the version with the noise points in the table because it yields better results. For the labeling, we tried both versions (labeling cluster 1 as positive and the rest as negative and vice-versa) and report the best version in the table. Distance-based clustering yields the best results, having the highest recall but with a very low precision of 0.28 in D_{full}, and the second-best (after *SkyEx-D*) *F-measure* of 0.74 in D_{sample}. Hierarchical clustering achieves higher precision than distance-based but with a very low recall of 0.11 in D_{full}, while the results are reversed to a high recall of 0.91 and a low precision of 0.23 in D_{sample}. For DBSCAN, the best values were achieved when we labeled the cluster as negative, and the noise points as positive, resulting in a recall of 1.0, but a very low precision of 0.23 in D_{full} and 0.26 in D_{sample}. This means that the positive-class pairs are not dense enough to form a cluster. Our *SkyEx-D* focuses more

on the distance between the classes rather than within the classes, and thus outperforms clustering.

10 Conclusions and Future Work

Location-based sources provide rich details and semantics about spatial entities. However, identifying which pairs of spatial entities refer to the same physical entity is a challenging problem. In this paper, we addressed the problem of spatial entity linkage across multiple location-based sources. We proposed *QuadSky*, an approach that consists of a spatial blocking technique *QuadFlex*, pairwise comparisons with suitable similarity metrics for each attribute, a skyline-based ranking algorithm *SkyRank*, and the *SkyEx-** family of algorithms for classifying the pairs. *QuadFlex* arranges the spatial entities into spatial blocks with a low execution time (4-8 times less than FNN [42]) and without missing relevant comparisons (99.99 percent of FNN comparisons). *SkyEx-F* achieves 0.84 p and 0.84 r on a manually labeled dataset and 0.87 p and 0.6 r on an automatically labeled dataset. We provided a theoretical guarantee to prune 73 percent of the skyline explorations in *SkyEx-F* with the novel *SkyEx-FES* without any loss of *F-measure*. Our fully unsupervised *SkyEx-D* finds k_d very close to the optimal k_f (an *F-measure* loss of just 0.01). The *SkyEx-** family of algorithms outperforms the existing baselines in terms of $F - measure$ and approximates the results of a supervised learning solution without the need of a labeled dataset, while *SkyEx-D* yields far better results than the clustering techniques. *SkyEx-F* and *SkyEx-D* are already available in the R `skyex` package [52], together with other functions for entity linkage. In future work, we aim to study different blocking techniques that combine several attributes and extend our *SkyEx-** family of algorithms to general (non-spatial) entity resolution problems.

References

[1] D. G. Brizan and A. U. Tansel, "A. survey of entity resolution and record linkage methodologies," *Commun. IIMA*, vol. 6, 2006, Art. no. 5.

[2] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *ACM SIGKDD Explorations Newslett.*, vol. 18, 2017.

[3] C. T. Yui, L. J. Liang, W. J. Soon, and W. Husain, "A survey on data integration in bioinformatics," in *Proc. Int. Conf. Inform. Eng. Inf. Sci.*, 2011, pp. 16–28.

[4] D. Firmani, B. Saha, and D. Srivastava, "Online entity resolution using an oracle," *Proc. VLDB Endowment*, vol. 9, 2016.

[5] R. Maskat, N. W. Paton, and S. M. Embury, "Pay-as-you-go configuration of entity resolution," in *Proc. Trans. Large-Scale Data-Knowl.-Centered Syst.*, 2016.

[6] J. Efremova *et al.*, "Multi-source entity resolution for genealogical data," in *Proc. Workshop Population Reconstruction*, 2015.

[7] M. Edwards, S. M. Wattam, P. E. Rayson, and A. Rashid, "Sampling labelled profile data for identity resolution," in *Proc. IEEE Int. Conf. Big Data*, 2016, pp. 540–547.

[8] O. Goga, H. Lei, S. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 447—458.

[9] A. Panchenko, D. Babaev, and S. Obiedkov, "Large-scale parallel matching of social network profiles," in *Proc. Int. Conf. Anal. Images Social Netw. Texts*, 2015, pp. 275–285.

[10] S. Isaj, N. Seghouani, and G. Quercini, "Profile reconciliation through dynamic activities across social networks," in *Proc. Int. Conf. Advanced Inf. Syst. Eng.*, 2019, pp. 126–141.

[11] R. Karam, F. Favetta, R. Kilany, and R. Laurini, "Integration of similar location based services proposed by several providers," in *Proc. Int. Conf. Netw. Digital Technol.*, 2010, pp. 136–144.

[12] B. Berjawi *et al.*, "Representing uncertainty in visual integration," in *Proc. Int. Conf. Distrib. Multimedia Syst.*, 2014, pp. 365–371.

[13] A. Morana, T. Morel, B. Berjawi, and F. Duchateau, "Geobench: A geospatial integration tool for building a spatial entity matching benchmark," in *Proc. 22nd ACM SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2014, pp. 533–536.

[14] S. Isaj, E. Zimányi, and T. B. Pedersen, "Multi-source spatial entity linkage," in *Proc. 16th Int. Symp. Spatial Temporal Databases*, 2019, pp. 1–10.

[15] X. L. Dong and D. Srivastava, "Big data integration," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 1245–1248.

[16] P. Christen, T. Churches, and M. Hegland, "Febrl–a parallel open source data linkage system," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2004, pp. 638–647.

[17] O. Peled, M. Fire, L. Rokach, and Y. Elovici, "Entity matching in online social networks," in *Proc. Int. Conf. Social Comput.*, 2013, pp. 339–344.

[18] G. Quercini, N. Bennacer, M. Ghufran, and C. N. Jipmo, "Liaison: Reconciliation of individuals profiles across social networks," in *Proc. Advances Knowl. Discovery Manage.*, 2017, pp. 229–253.

[19] M. Magnani and D. Montesi, "A survey on uncertainty management in data integration," *J. Data Inf. Quality*, vol. 2, 2010, Art. no. 5.

[20] M. Schäfers and U. Lipeck, "Simmatching: Adaptable road network matching for efficient and scalable spatial data integration," in *Proc. ACM SIGSPATIAL PhD Workshop*, 2014, pp. 1–5.

[21] R. Abdalla, "Geospatial data integration," in *Proc. Int. Geospatial Inf. Commun. Technol.*, 2016, pp. 105–124.

[22] P. Tabarro, J. Pouliot, R. Fortier, and L. Losier, "A webgis to support GPR 3D data acquisition: A first step for the integration of underground utility networks in 3D city models," *Int. Archives Photogrammetry Remote Sensing Spatial Inf. Sci.*, vol. XLII-4/W7, pp. 43–48. 2017.

[23] S. Balley, C. Parent, and S. Spaccapietra, "Modelling geographic data with multiple representations," *Int. J. Geographical Inf. Sci.*, vol. 18, pp. 327–352, 2004.

[24] V. Walter and D. Fritsch, "Matching spatial data sets: A statistical approach," *J. Int. J. Geographical Inf. Sci.*, vol. 13, pp. 445–473, 1999.

[25] V. Sehgal, L. Getoor, and P. D. Viechnicki, "Entity resolution in geospatial data integration," in *Proc. 14th Annu. ACM Int. Symp. Advances Geographic Inf. Syst.*, 2006, pp. 83–90.

[26] A. O. Raimond and S. Mustière, "Data matching–a matter of belief," in *Proc. Headway Spatial Data Handling*, 2008, pp. 501–519.

[27] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 53–62.

[28] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative analysis of approximate blocking techniques for entity resolution," *Proc. VLDB Endowment*, vol. 9, 2016.

[29] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.*, vol. 16, 1984.

[30] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, 1966, Art. no. 707.

[31] C. Fellbaum, "Wordnet," *Theory Appl. Ontology: Comput. Appl.*, 2010.

[32] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proc. 32nd Annu. Meeting Assoc. Comput. Linguistics*, 1994, pp. 133–138.

[33] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proc. VLDB Endowment*, vol. 3, 2010.

[34] Y. Censor, "Pareto optimality in multiobjective problems," *Appl Math Optim.*, vol. 4, pp. 41–59, 1977.

[35] G. M. Weiss and H. Hirsh, "Learning to predict extremely rare events," in *Proc. Amer. Assoc. Artif. Intell. Workshop*, 2000.

[36] D. A. Cieslak and N. V. Chawla, "Learning decision trees for unbalanced data," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2008, pp. 241–256.

[37] J. Zhang, E. Bloedorn, L. Rosen, and D. Venese, "Learning rules from highly unbalanced data sets," in *Proc. 4th IEEE Int. Conf. Data Mining*, 2004, pp. 571–574.

[38] M. Gordon and M. Kochen, "Recall-precision trade-off: A derivation," *J. Amer. Soc. Inf. Sci.*, vol. 40, 1989.

[39] P. Roocks, "Relational and algebraic calculi for database preferences," 2016.

[40] M. Endres, P. Roocks, and W. Kießling, "Scalagon: an efficient skyline algorithm for all seasons," in *Proc. Int. Conf. Database Syst. Advanced Appl.*, 2015, pp. 292–308.
[41] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction," in *Proc. 16th Int. Symp. Spatial Temporal Databases*, 2019, pp. 11–20.
[42] J. L. Bentley, D. F. Stanat, and E. H. Williams Jr , "The complexity of finding fixed-radius near neighbors," *Inf. Process. Lett.*, vol. 6, pp. 209–212 , 1977.
[43] D. W. Hosmer and S. Lemeshow, "Applied logistic regression," Hoboken, NJ, USA: Wiley, 1989.
[44] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, 1995.
[45] W. A. Belson, "Matching and prediction on the principle of biological classification," *J. Royal Statist. Soc. Series C*, vol. 8, pp. 65–75, 1959.
[46] T. Bayes, "Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, FRS communicated by mr. price, in a letter to john canton, AMFR S," *Philos Trans. Roy. Soc. London B Biol. Soc.*, 1763.
[47] G. You, S. Hwang, Z. Nie, and J. Wen, "Socialsearch: Enhancing entity search with social network matching," in *Proc. 14th Int. Conf. Extending Database Technol.*, 2011, pp. 515–520.
[48] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Prob.*, vol. 1, pp. 281–297, 1967.
[49] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," 1987.
[50] T. Hastie, R. Tibshirani, and J. Friedman, "Hierarchical clustering," *EoSL*, 2009.
[51] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.
[52] S. Isaj and T. B. Pedersen, "skyex: An r package for entity linkage," in *Proc. Int. Conf. Extending Database Technol.*, 2020.

Suela Isaj received the MSc degree from the University of Tirana, 2013, and the joint MSc degree from the Université libre de Bruxelles, Université François Rabelais and CentraleSupélec, in 2017. She is currently working toward the PhD fellow degree at the Center for Data-Intensive Systems (Daisy) at Aalborg University, Denmark. Her current research is focused on location-based data extraction and integration, data mining, and data analytics.

Torben Bach Pedersen (Senior Member, IEEE) is a professor with the Center for Data-Intensive Systems (Daisy) at Aalborg University, Denmark. His research concerns big data analytics on multidimensional data. He is an ACM distinguished scientist, and a member of the Danish Academy of Technical Sciences.

Esteban Zimányi is a professor and the director of the Department of Computer and Decision Engineering (CoDE), Université libre de Bruxelles. He is editor-in-chief of the Journal on Data Semantics published by Springer. His current research interests include data warehouses, spatio-temporal databases, geographic information systems, and Semantic Web.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.