Searching OSM Planet with Context-Aware Spatial Relations

Daniel Bahrdt
Universität Stuttgart
Germany

Stefan Funke
Universität Stuttgart
Germany

Rick Gelhausen
Universität Freiburg
Germany

Sabine Storandt
Universität Würzburg
Germany

ABSTRACT

We consider the problem of indexing the complete OpenStreetMap planet data set (> 500GB of raw data) to support complex queries involving both text search as well as context-aware spatial relations. This requires (a) formalization of spatial relations like 'north of', 'between', 'near' depending on the context, and (b) the development of suitable data representations to integrate textual and spatial information for efficient query performance.

CCS CONCEPTS

• **Information systems** → *Information retrieval*;

KEYWORDS

OpenStreetMap, proximity queries, textual spatial search

ACM Reference Format:
Daniel Bahrdt, Stefan Funke, Rick Gelhausen, and Sabine Storandt. 2017. Searching OSM Planet with Context-Aware Spatial Relations. In *Proceedings of 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3139958.3140000

1 INTRODUCTION AND RELATED WORK

Directional relations are frequently used to select data in spatial databases (SDB) and are fundamental to spatial data queries, analysis and reasoning [1, 6]. They are not only used widely in geographic information systems, but also in areas like artificial intelligence [8], computer vision [9], and multimedia [12]. Consequently there has been a significant amount of effort to determine directional relations automatically.

A commonly used indicator for the directional relation between two regions is the direction between their centroids [11], possibly snapped to one of the 8 cardinal directions. As an advantage, this indicator is symmetric and once the centroids are (pre)computed, efficient to evaluate for a pair of regions. Yet, sometimes asymmetric answers make more sense, which the popular direction-relation matrix model [5] allows for. It subdivides the space around the bounding box of the reference region into nine direction tiles and classifies other regions according to the cell of the subdivision they lie in. [3] introduce a splitting line model to decide on the relation of two geometric entities. The goal of all these approaches is to compute a directional indicator between two given regions, whereas for our concrete application of a SDB we want to determine regions fulfilling a certain directional relation. Moreover we also need to derive the *extent* of a region for the matches, not only a direction.

[10] and [7] propose non-trivial spatial queries and respective implementations in the context of a SDB system. In [10] *nearest neighbors* and *range search* primitives, amongst others, are implemented both for Euclidean and network connectivity constraints. For example, for the nearest neighbor primitive, they employ geometric search structures and a network traversal to collect matching results. [7] on the other hand design rather complex surrounder queries with the main application domain being in location based services. The reported experiments are of rather small scale (a database of a few thousand elements). While these approaches yield exact nearest neighbors or report exactly all entities in a given range, we want to emphasize that they are hardly feasible for large data-sets. Queries like "All McDonald's branches world-wide which do *not* have a Burger King branch *within* 2km radius" would have to be executed on each McDonald's branch world-wide (more than 30,000).

Our contribution

This work is based on the freely-available OpenStreetMap (OSM) data set. In [2] we have proposed an efficient framework to access all entities of the OSM planet data set with their implicit and explicit informations, supporting substring search on the key-value tags as well as all common set operations like intersection, union, or difference. This work builds upon this framework, extending it to allow for the efficient support of adaptive spatial relations. We provide actual semantics of spatial relations based on context and their implementation in our framework. Experiments as well as a web-based demonstrator on www.oscar-web.de show the efficiency of our approach even on a planet scale with hundreds of millions of elements.

2 OSM CELL ARRANGEMENTS

The key concept behind OSCAR is a so-called *cell arrangement of OSM regions*. Consider a set of simple polygons $\mathcal{P}$, each corresponding to an OSM region. Each single polygon $P \in \mathcal{P}$ divides the plane into two regions, the *interior* of the polygon and the *exterior*. The set $\mathcal{P}$ naturally induces a subdivision of the plane into *cells*, we call this the *cell arrangement* $\mathcal{A} = \mathcal{A}(\mathcal{P})$ of $\mathcal{P}$. All points within a single cell $c \in \mathcal{A}$ have the property that they behave identically with respect

SIGSPATIAL'17, November 7–10, 2017, Los Angeles Area, CA, USA

ACM ISBN 978-1-4503-5490-5/17/11.
https://doi.org/10.1145/3139958.3140000

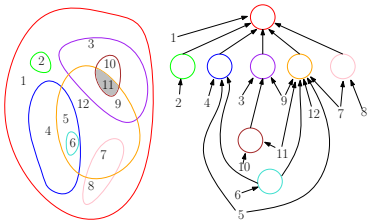

Figure 1: Left: Cell arrangement – Points in the gray cell 11 have the property of being contained in the brown, purple, orange, and red region, but not in the others. Right: Corresponding inclusion-DAG.

to containment in the set of polygons, in particular inheriting the same information associated with the respective OSM regions.

The efficiency of OSCAR is based on the fact that the number of cells in our cell arrangement $\mathcal{A}$ is considerably smaller than the total number of items. The basic idea for our query data structure is then as follows:

(1) create cell arrangement $\mathcal{A}$ induced by OSM regions
(2) associate with each cell $c \in \mathcal{A}$ all tags of OSM regions containing c
(3) build search structure on 'tagged' cells and individually tagged nodes

So for a single query string such as *Stuttgart* our data structure returns all cells which have an associated tag with substring 'Stuttgart' as well as all individual nodes whose tag set contains the substring 'Stuttgart'. If several search terms are specified as part of a complex search query, the set operations take place at the cell level as long as possible. Only at the very end, a conversion to items takes place.

The inclusion-relation of the regions creating the cell arrangement induces a natural clustering which we use for presenting the result and calculation of candidate regions and items. See Figure 1, left, for a cell arrangement with 12 cells induced by 8 regions. In Figure 1, right, we see the induced Directed Acyclic Graph – we call it *Inclusion-DAG*. There is a directed edge (v, w) from a cell/region node v to a region node w in the DAG, if and only if the cell/region v is fully contained in the region w and there exists no other region $w' \subset w$ in which it is fully contained. For a cell the set of regions it is contained in can be easily determined via breadth- or depth-first search in the inclusion-DAG starting from the respective node.

Refining the OSCAR Cell Complex

For implementations of Adaptive Spatial Relations, it is desirable to have a cell arrangement with well-shaped cells. This is typically not given for the cell arrangement induced by the OSM regions, e.g., many cells have very irregular shapes. Hence we perform a refinement step to make cells of the arrangement look nice, both in terms of shape, topology, and complexity. We first compute a *Constrained Delaunay Triangulation (CDT)*[4] of the original arrangement. Keeping these triangles as cells would increase the complexity of the search structure too much, so in a second step we merge triangles to form larger, yet simple and nice cells. See Figure 2 for a schematic of the refinement process. While the refinement process increases the complexity of the arrangement, the blow-up is still limited and allows for the easy implementation of adaptive spatial relations.

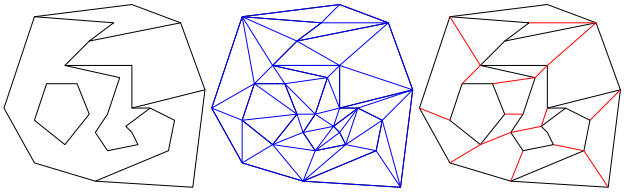

Figure 2: Schematic of the refinement process. Left: Original Cell Arrangement. Center: Constrained Delaunay Triangulation thereof. Right: After merging of triangles.

3 ADAPTIVE SPATIAL RELATIONS VIA A REFINED CELL COMPLEX (OSCAR-RCC)

In the following we define adaptive spatial relations through polygons of interest. Inspecting all items within such a polygon is particularly costly. Yet, the cell-centered representation and processing in OSCAR strongly suggests also a cell-based implementation of adaptive spatial relations. In particular larger regions of interest will benefit considerably from cell-based processing.

For generality let us assume a given decomposition of the map into tiles/cells and a data structure $\mathcal{D}$ which for a given query Q:

- computes all tiles containing a match for Q
- for a given tile C known to contain matches for Q, outputs all matches

Note that depending on the implementation of $\mathcal{D}$, computing the set of tiles containing at least one match might be much cheaper than outputting all matches. OSCAR provides such a tiling, but the following scheme applies to any cell- or tile-based representation.

For some query Q (e.g., searching for the string *Burger King*) the assumed data structure $\mathcal{D}$ can compute all tiles containing at least one match, and for each tile also output the actual matches, if necessary (at higher cost).

Neighborhood. The probably most natural spatial relation is about proximity, but there are considerable differences in terms of semantics depending on the objects referenced. The extent of the reference object (the Eiffel Tower or the city of Paris) typically also determines the extent of the region of interest. To keep things simple, we compute a minimum enclosing rectangle for the reference object and let d be the extent of the *smaller* dimension (breadth or width). Then we define as neighborhood of a reference object all locations which have distance at most $\max(d, 500m)$ to the reference object. We use the maximum of d and 500m to also define neighborhoods of 0- or 1-dimensional entities (e.g., a mailbox is typically not considered to have any real 'extent').

A simple cell-based approximation of the neighborhood semantics as defined before distinguishes between two cases. If the extent of the reference object is small, we simply define as relevant neighborhood the cell containing the reference object and neighboring cells within the desired distance. This often leads to a larger region of interest than intended, but a precise pruning with respect to distance can be part of a post-processing step. For larger reference objects, we use the occupied cells to estimate the extent and then determine the respective cells.

Cardinal Directions. A user issuing a query like "*hotels north of the Empire State Building*" probably expects accommodations

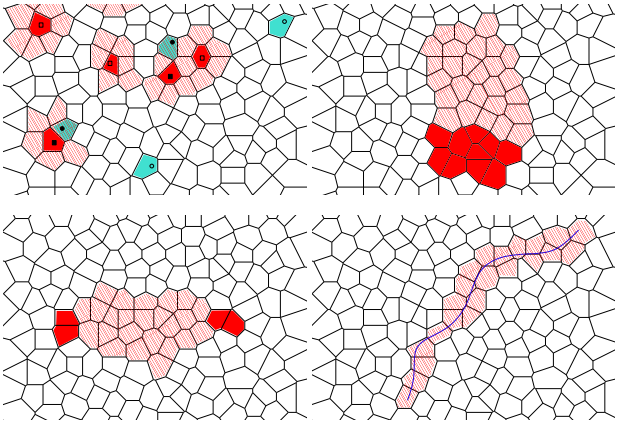

Figure 3: Relations *Nearby/SpatialJoin, NorthOf, Between* and *AlongPath*.

within a distance of 500m north of the Empire State Building. For larger entities, a larger region is of interest – "*hotels north of Paris*" probably refers to a region up to a distance of 20km north of Paris.

We formalize this as follows: For 0-dimensional reference objects a simple triangle in the cardinal direction is used. As we cannot infer the size of the region of interest from the extent of the reference object and use a triangle of height 500m. If additional context information is available, the size of this triangle can easily be adjusted.

For 1-dimensional reference objects, we first determine the northernmost point of the polygonal line as well as the midpoint of the bounding box. We then construct a temporary point with latitude of the northernmost point and longitude of the center of the bounding box. We then pass this point to the cardinal direction function to create an intermediate triangle. Finally, the region of interest is the convex hull of this triangle together with the polygonal line.

For 2-dimensional reference objects we first create a bounding box and an isosceles trapezoid of twice the height of the bounding box, the base is equal to the horizontal axis of symmetry of the bounding box and a parallel opposing side of double that length.

We can compute the result of a query "*hotels north of Paris*" based on cells by first determining the cells/tiles that are (fully or partially) covered by the city limits of Paris (via a search for "*Paris*"). We then approximate the extent of Paris by inspection of the respective tiles (computationally inexpensive, as there are only few tiles). Using this approximate extent we mark tiles north of these tiles, see Figure 3, for a schematic illustration.

Betweenness. As for cardinal directions, the region referred to by a 'between' relation depends on the reference regions and in particular their distance apart.

To compute polygons bounding the region of interest for the betweenness relation we again distinguish between the dimension of the reference objects. If we need the region between two 0-dimensional reference objects, we construct a diamond shape with two opposing corners being the two reference objects and a width of half the distance between the two reference objects. In all other cases we construct the convex hull of the bounding boxes of both reference objects. We use the area of the convex hull without the bounding boxes as region of interest, as well the part of the bounding boxes that is formed by connecting the convex hull edges between the bounding boxes to the next nodes of the reference object on the bounding box border in the respective direction (which must be well-defined).

We can use this polygon to simply mark all cells between the reference objects, see Figure 3.

Path Corridors. When looking for 'hotels along the route from Stuttgart to Berlin' one typically refers to a much smaller area than in 'hotels between Stuttgart and Berlin'. So given a (precomputed) path we define as the area of interest as all locations with distance of at most 1 kilometer from the route. In our implementation we simply mark all cells traversed by the route, see Figure 3.

We want to note, though, that for all discussed relations other toolboxes than the ones described here (or with different parameter choices) could be plugged in our framework as well.

Spatial Joins. A considerably more complex query arises in the following scenario: Assume we are looking for residential areas where both kindergarten and school are nearby. Querying for "*kindergarten with school within* 500 *m*" should yield all kindergarten for which there exists a school not too far away. The neighborhoods of such kindergartens then presumably constitute desirable residential areas. The challenge here is that there are more than 300k tagged kindergartens in the OSM data set.

How could we use our query structure $\mathcal{D}$ together with the tiling to efficiently compute the result set? We first search for *school* resulting in a set of tiles containing schools and mark neighboring tiles depending on our notion of 'nearby'. Then we search for *kindergarten* but only outputting the respective matches in tiles that were marked previously.

Some queries have much higher nesting depths, e.g., "*supermarkets which have an ATM, a pharmacy, and a car repair shop nearby*".

4 EXPERIMENTAL EVALUATION

In the following we present some benchmarks to outline the performance of OSCAR. The hardware specifications of the test environments together with information about the data set as well as an overview of resource usage of the pre-processing stage and the impact of the cell refinement processes are listed in Table 1.

We have picked a number of representative queries with increasing complexity and analyzed their query processing. Figure 4 lists the queries, their sizes and computation time. It also shows the impact of loading data from disk which usually increases the query time by a factor of 10. The cell time is the time it takes to do all the set operations. The subgraph time is the time needed to compute the subgraph of the region hierarchy. The former is of course higher for queries with many and large set operations - the latter takes longer for results with many different regions. This is especially true for all single query terms since these just do a look-up in our text search data structure. Queries 1,2,13 and 14 mostly select single items and as a consequence the number of cells does not differ very much from the number of items. The creation of the subgraph of the Inclusion-DAG does of course take longer since the items are scattered around the world. Queries 3,7 and 9 select a mixture of items and cells with *paris* selecting many full-match cells of Paris. Query 4 has large intermediate results with a small result mostly reduced to items in Paris. Query 5 adds

Data set statistics		Cell size			Hardware statistics		Pre-processing ressource usage		
# Items	500M	Avg. unrefined	353		Query	Preprocessing		db	ts
# Cells	2.78M	Avg. refined	189	CPU	2 x Intel E5-2630v2	2 x Intel E5-2650v4	Time [h:m]	15:14	34:17
Database	64.5 GB	Max unrefined	2.13M	RAM	384 GiB	768 GiB	Ram [GiB]	17.8 GB	39.8 GB
Text search	33.4 GB	Max refined	578k	Disk	2 SSD raid0	3 HDD raid0	Load avg. [%]	1516	298
Index	40.4 GB								

Table 1: From left to right: Information about the dataset with all tags stored, prefix search for tags, substring search for important tags. Impact of the cell refinement. Statistics of our test environments. Resource usage to compute the data base (db) and the text search (ts) with 32 GiB allocated for out-of-memory algorithms.

the *:north-of* operation to query 4 returning some full-match cells in Paris. Adding *@tourism:attraction* to query 5 further reduces the number of items while not increasing the computation much more since the former query mostly consists of full-match cells. Queries 10 and 11 use the region query to query for Munich and Germany. The results of these are fed to the between operator to get the rather large region between Munich and Frankfurt/Main. This operation takes quite some time which is mostly due to the not yet fully optimized polygon-query. Finally, query 15 uses the cell dilation operator turning partial-matched cells into fully-matched cells and intersecting these with another set of partial-match cells.

Moverover, the query times can be improved by using multiple threads and our implementation in fact parallelizes the execution of the set operations. Consider for example the query *@highway @building @amenity* where each operand selects a large amount of partial-match cells. Computing the set operations should in theory scale linearly in the number of processors with a speed-up of 1, however, in practice we get an efficiency ratio of 0.5 to 0.7:

Threads [1]	1	2	4	8	12	16
Time [ms]	2165	1450	830	470	330	290
Efficiency	1	0.74	0.65	0.58	0.55	0.47

5 CONCLUSIONS

In this paper we have shown that it is possible to develop efficient data representations and algorithms that allow for complex string search with spatial relation queries even for massive data sets like OSM planet. The key ingredient for efficiency is the cell-centered processing within the data structure with well-shaped cells that facilitate the implementation of adaptive spatial relations. Our web-based demonstrator can answer queries on a global scale which even involve adaptive spatial relations not available in common search engines like Google/Bing Maps. Future work includes the handling of dynamic update of the underlying data set as well as more parallelization to reduce the time for precomputation.

Acknowledgements This work was in part supported by the Deutsche Forschungsgemeinschaft (DFG) within the priority program 1894: Volunteered Geographic Information: Interpretation, Visualization and Social Computing.

No.	Cells	Items	Query
1	79.9k	259k	*@tourism:hotel*
2	54.8k	205k	*@tourism:attraction*
3	41.4k	4.23M	*paris*
4	1	2	*eiffel tour paris @tourism:attraction*
5	8	3.43k	*:north-of (eiffel tour paris @tourism:attraction)*
6	1	3	*@tourism:hotel :north-of (eiffel tour paris @tourism:attraction)*
7	1.16k	491k	*Munich*
8	14	15.8k	*:north-of Munich*
9	714	209k	*Frankfurt am Main*
10	261k	50.7M	*#Germany*
11	824	487k	*#Germany #Munich*
12	3.90k	946k	*("Frankfurt am Main") <-> (#Munich #Germany)*
13	63.7k	175k	*@amenity:kindergarten*
14	208k	836k	*@amenity:school*
15	48.5k	505k	*(%@amenity:kindergarten @amenity:school) + (@amenity:kindergarten %@amenity:school)*

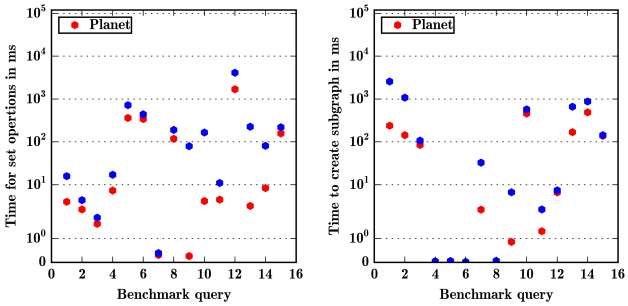

Figure 4: Top: Result sizes for various queries issued on the planet data set. Bottom: The same queries with hot cache (red) and cold cache (blue) timings for cell (left) and subgraph (right) computation. Note that the total time of all queries except one is well below the 1 second mark.

REFERENCES

[1] Alia I. Abdelmoty and Chris B. Jones. 1997. Towards Maintaining Consistency of Spatial Databases. In *Proc. 6th Int. Conference on Information and Knowledge Management (CIKM '97)*.
[2] Daniel Bahrdt and Stefan Funke. 2015. OSCAR: OpenStreetMap Planet at Your Fingertips via OSm Cell ARrangements. In *WISE (1) (LNCS)*, Vol. 9418. Springer.
[3] Kevin Buchin, Vincent Kusters, Bettina Speckmann, Frank Staals, and Bogdan Vasilescu. 2011. A Splitting Line Model for Directional Relations. In *Proc. 19th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*.
[4] L. P. Chew. 1987. Constrained Delaunay Triangulations. In *Proc. 3rd Annual Symposium on Computational Geometry (SCG '87)*.
[5] Roop Kishor Goyal. 2000. *Similarity Assessment for Cardinal Directions Between Extended Spatial Objects*. Ph.D. Dissertation. University of Maine. AAI9972143.
[6] Roop K. Goyal and Max J. Egenhofer. 2001. Similarity of Cardinal Directions. In *Proc. 7th Int. Symp. on Advances in Spatial and Temporal Databases (SSTD '01)*.
[7] Xi Guo, Baihua Zheng, Yoshiharu Ishikawa, and Yunjun Gao. 2011. Direction-based Surrounder Queries for Mobile Recommendations. *The VLDB Journal* 20, 5 (Oct. 2011).
[8] A. Klippel, C. Dewey, M. Knauff, K.-F. Richter, D. R. Montello, C. Freksa, and E.-A. Loeliger. 2004. Direction Concepts in Wayfinding Assistance Systems. In *Proc. Workshop on Artificial Intelligence in Mobile Systems (AIMS'04) (SFB 378)*.
[9] Pascal Matsakis and Laurent Wendling. 1999. A New Way to Represent the Relative Position Between Areal Objects. *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 7 (July 1999).
[10] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. 2003. Query Processing in Spatial Network Databases. In *Proc. 29th Int. Conference on Very Large Data Bases - Vol.29 (VLDB '03)*.
[11] D. J. Peuquet and Ci-Xiang Zhang. 1987. An Algorithm to Determine the Directional Relationship Between Arbitrarily-shaped Polygons in the Plane. *Pattern Recogn.* 20, 1 (Jan. 1987).
[12] Stephen S.-T. Yau and Qing-Long Zhang. 2004. On Completeness of Reasoning about Planar Spatial Relationships in Pictorial Retrieval Systems. *Commun. Inf. Syst.* 4, 3 (2004).