Algorithm for searching a shortest path obeying restricted areas without preprocessing

František Kolovský (kolovsky@students.zcu.cz), Jan Ježek (jezekjan@kma.zcu.cz)
University of West Bohemia, Faculty of Applied Sciences, Section of Geomatics, Czech Republic

Abstract

Finding the shortest path in a given road network is a well-know problem. In addition to the commonly used restrictions for the searched path, such as speed limits, there might be other constraints considered such as polygonal obstacles. This poster describes the shortest path 'obstacle-wise' search algorithm and its implementation that was submitted to the ACM SIGSPATIAL Cup 2015.
The poster depicts an overview about the implementation and evaluation of various state-of-the-art algorithms dedicated for shortest path search. This algorithms were customized to search the path that obeys specified polygonal areas (restricted zones). The resulting algorithms were extensively evaluated by measuring their performance as well as founded path quality. The test results show that the variant based on Bidirectional Dijkstra's presents the fastest exact solution.

Restricted areas

The given task requires the searched path to avoid restricted areas. Restricted areas have the form of polygons (including non-convex polygons). The well state-of-the-art search algorithms were customized in order to test whether or not a particular edge is located in the restricted zone. More concretely the *relaxation* function was extended to evaluate if a particular edge intersects a restricted area (by calling *intersectsRS*).

Function *relaxation*

1: **for** $\forall$ neighbours j **do**
2: **if** *intersectsRS*((i,j)) **then**
3: continue
4: **end if**
5: **if** $dist(i) + c(i,j) < dist(j)$ **then**
6: $dist(j) = dist(i) + c(i,j)$
7: $prev(j) = i$
8: $Q = Q \cup j$
9: **end if**
10: **end for**

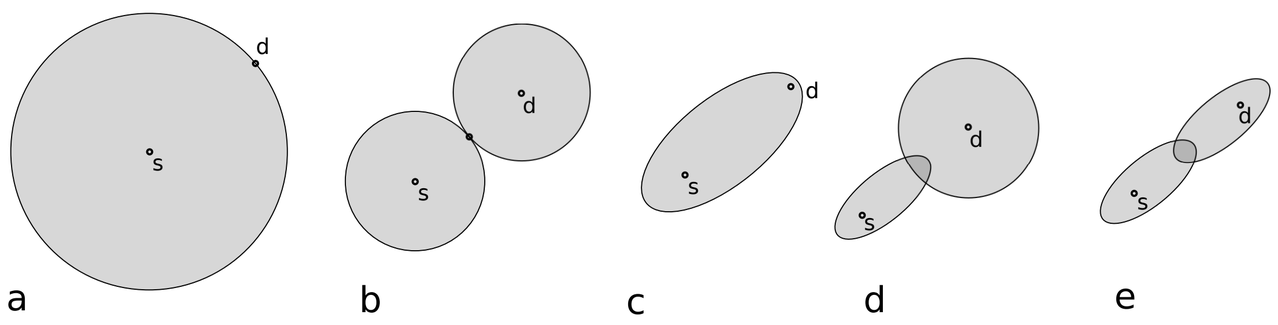
Function *intersectsRS*

1: e is input edge
2: **for** $\forall$ restricted polygons $polyg$ **do**
3: **if** bbox(e) intersect with bbox(polyg) **then**
4: **if** a intersect with $polyg$ **then**
5: **return** true
6: **end if**
7: **end if**
8: **return** false
9: **end for**

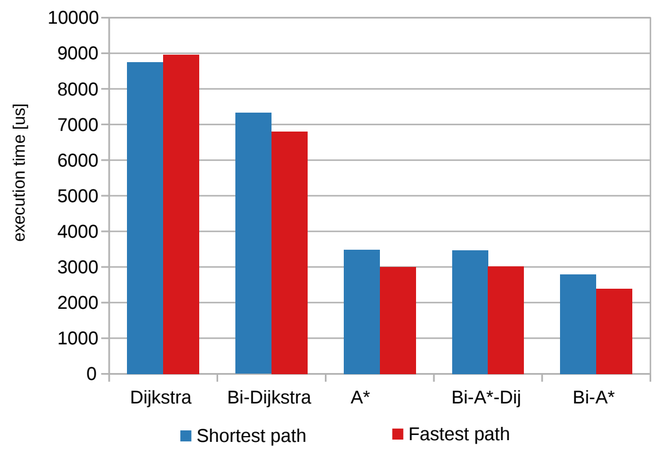
Tested Algorithms

For chosen state-of-the-art methods (**a**), Bi-directional Dijkstra (**b**), A* algorithm (**c**), Bi-directional (A* - forward search, Dijkstra - backward search) (**d**), Bi-directional A* (**c**)) we performed 100 shortest path searches between randomly generated points. Graph has 67 000 edges (m), 29 000 vertexes (n) and 5 restricted zones (every zone contains about 5 edges). Two cases were evaluated: searching for the shortest path and searching for the fastest path.

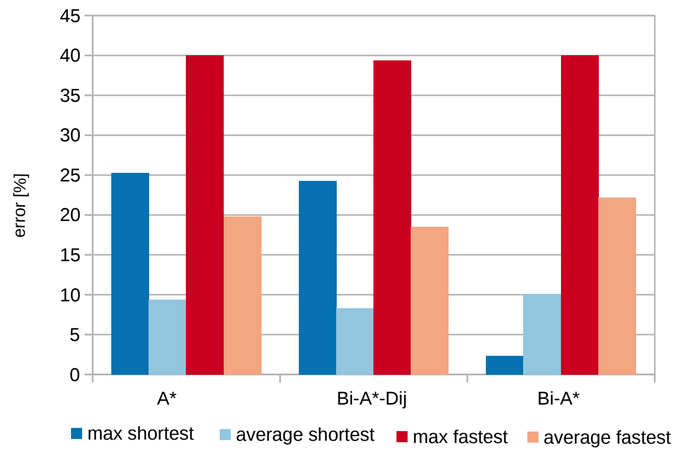
Search space of the algorithms



Preformance



Accuracy



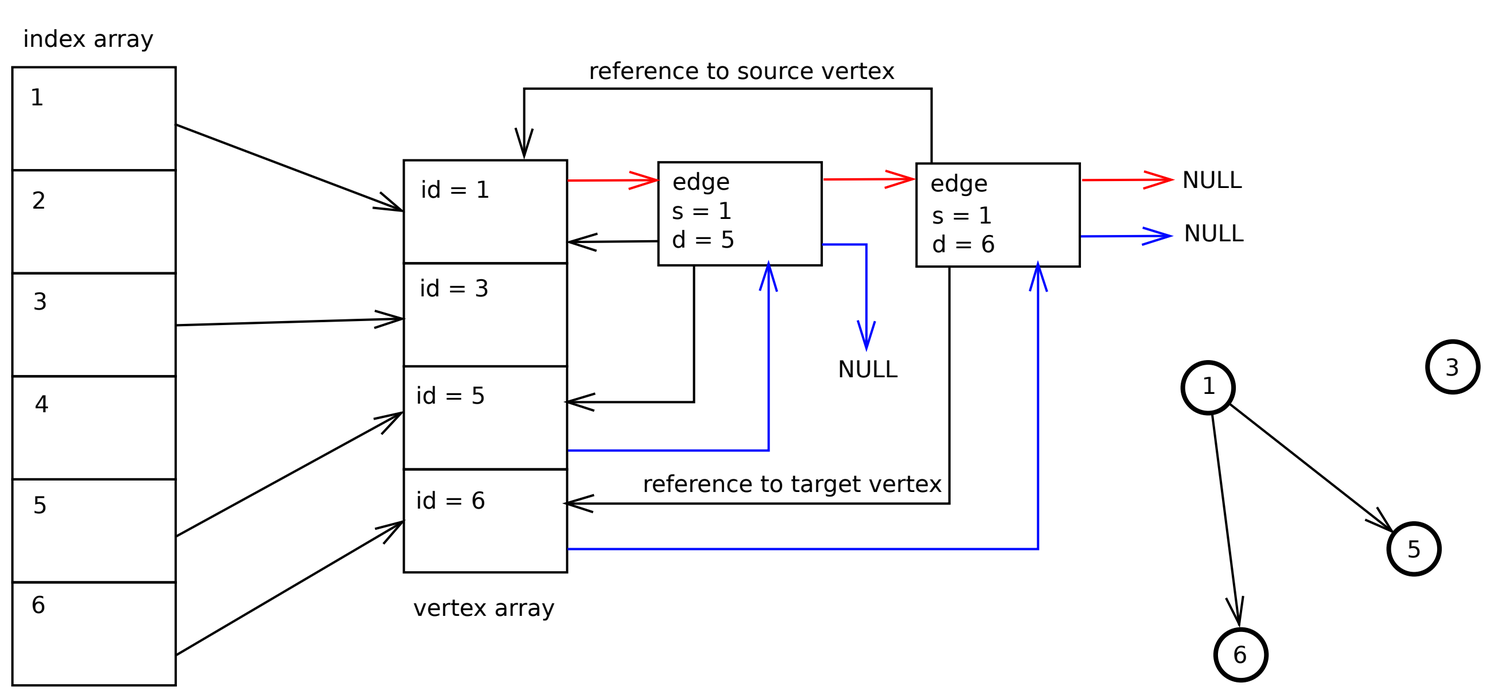
Conclusion

We customized state-of-the-art techniques focused on the shortest path search to avoid the restricted zones. We made several implementation considerations based on the size of the network, the number of restricted areas as well as expected number of given queries. By experiments we evaluate these design choices and select the one featuring the best performance while keeping the found path exact. The fastest method is Bi-A* and the slowest is Dijkstra. The solution based on Bidirectional Dijkstra's algorithm was delivered for the ACM SIGSPATIAL Cup 2015 as the fastest exact method.

Acknowledgement: This work was supported by project OpenTransportNet and LO1506 project of the Czech Ministry of Education, Youth and Sports.

Implementation

We implemented the described solution in the C programming language. Figure depicts a schema of the graph representation. The graph vertexes are stored as an array. For an efficient searching for a vertex by its *id*, there is an index array, where the *id* is used as an array's index and the item value is a pointer to the vertex array. If the range of vertexes *ids* is too wide, we get a sparse array, that might exceed the primary memory. In that case there is a fallback solution and the hash table is used. The vertex structure contains the pointer to a linked list of vertex's edges having this vertex as a source (red arrows in the schema) or as a destination (blue arrows in the schema).



References

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische mathematik, 1(1):269271, 1959.

[2] A. V. Goldberg. Point-to-point shortest path algorithms with preprocessing. In SOFSEM 2007: Theory and Practice of Computer Science, pages 88102. Springer, 2007.

[3] D. Wagner and T. Willhalm. Speed-up techniques for shortest-path computations. In STACS 2007, pages 23 36. Springer, 2007.

GISCup of 23rd ACM SIGSPATIAL Conference, 3-6. 11. 2015 in Seattle, USA.