Fast Approximation of Reach Hierarchies in Networks

Joris Maervoet
KU Leuven, Department of Computer Science
CODeS & iMinds-ITEC
Gebroeders De Smetstraat 1, 9000 Gent, Belgium

Patrick De Causmaecker
KU Leuven, Department of Computer Science
CODeS & iMinds-ITEC
Etienne Sabbelaan 53, 8500 Kortrijk, Belgium

Greet Vanden Berghe
KU Leuven, Department of Computer Science
CODeS & iMinds-ITEC
Gebroeders De Smetstraat 1, 9000 Gent, Belgium

ABSTRACT

The reach of an arc in a network can intuitively be described as an indication of the maximum length of the shortest paths of the digraph that pass through this arc. This concept captures the natural hierarchy of any type of network, in an accurate and comprehensive manner. Traditional reach approximation algorithms compute upper bounds to these reaches and require computation of a partial shortest path tree rooted in all vertices of the network. Tailored for route computation enhancement, these methods yield exact reaches in the low reach spectrum, whereas higher reaches are kept set to infinity.
The present paper introduces an iterative method for generating lower bounds to the reaches of the arcs in a network. This method is suitable for situations where it is more important to know the order of magnitude of the high than these of the low reaches and where very low or variable calculation times are required. An experiment in an attractiveness-weighted cycling network of considerable size shows that the iterative method steadily approximates the arc reaches for a low number of iterations. The approximation algorithm has been formulated for arc reaches in a digraph but can easily be adapted to a vertex reach version and works in undirected graphs as well.

Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Graph Theory—*Graph algorithms, Approximation algorithms*

Keywords

Network hierarchies, approximation algorithm, arc reach, shortest paths

General Terms

Algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SIGSPATIAL'14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA
Copyright 2014 ACM 978-1-4503-3131-9/14/11 ...$15.00
http://dx.doi.org/10.1145/2666310.2666474

1. INTRODUCTION

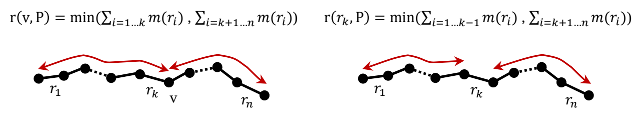
The present paper is dedicated to automated discovery of the *natural hierarchy* in road networks. Basically, this process involves assigning grades of importance to the vertices and/or the arcs of a network. A well-known example of the natural hierarchy in a time-weighted network is the classification of the network's roads in correspondence with the roads' *functional road class.* This hierarchy comprises a limited number of levels, ranging from 0 to n, where level n refers to motorways, level $n-1$ to arterial or national roads, and so on, down to level 0 referring to unclassified or local roads. In this type of hierarchy, each arc r of a digraph $G = (V, A)$ belongs to one of the $n+1$ levels: $level(r) \in [0, n]$.

The reach of a vertex or arc, introduced by Gutmann [2], is a concept capturing the natural hierarchy of any type of graph, in an accurate and comprehensive manner. The reach of a vertex/arc can intuitively be described as a label indicating the length of the shortest paths that pass through this vertex/arc. These labels enable a very drastical reduction of the search space of a corresponding shortest path algorithm. As far as the authors are aware, an all-pair-shortest-path (APSP) algorithm, which is computationally expensive, is required in order to determine all reaches of a graph. Both Gutmann [2] and Goldberg et al. [1] proposed a faster method for the calculation of the reaches' upper bounds. Tailored for route computation enhancement, these methods yield exact reaches in the low reach spectrum, whereas higher reaches are kept set to ∞.

Apart from routing computation enhancement, automated hierarchy discovery in networks has several other application areas such as

- tools and techniques for analysis of general networks,
- traffic analysis and urban planning, and,
- generation of reference networks for tourism and leisure.

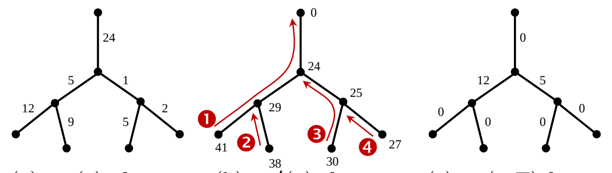
Suppose, for instance, a connected 2-level hierarchy inherent to a cycling network where the weights model the ratio of the arc's length to its cycling attractiveness. It is likely that the subgraph of the highest level, denoted S_1, will contain roads many cyclists need to take although these roads are less attractive for cycling. A verification of the road infrastructure of the level 1 roads can support urban planners in the identification of required road infrastructure improvements. Moreover, the hierarchy could provide vital information for planning a *national cycle highway.* In the domains of tourism and leisure, S_1 could serve as a reference network of



r(v, P) = min(∑_{i=1…k} m(r_i) , ∑_{i=k+1…n} m(r_i))
r_1
r_k
v
r_n
r(r_k, P) = min(∑_{i=1…k−1} m(r_i) , ∑_{i=k+1…n} m(r_i))
r_1
r_k
r_n

(a) Vertex reach of v (b) Arc reach of r_k

Figure 1: Two reach definitions in a shortest path P, consisting of arcs r_1 to r_n, given a reach metric $m : A \to \mathbb{R}^+$.



(a) $m(r)$ for any arc r in T (b) $m'(v)$ for any vertex v in T (c) $r_T(r,T)$ for any arc r in T

Figure 2: Collection of reach numbers in a shortest path tree. The subcaptions indicate the legend of the numbers printed in black. The numbers printed in hollow in Figure 2b indicate the order of the four main iterations of the collection procedure.

a hardcopy themed recreation guide, or, as a visual overlay aiding website visitors to plan highly attractive routes.

The present paper presents an iterative algorithm to approximate the arc reaches of a network. It has two advantages over the algorithms proposed by Gutmann [2] and Goldberg et al. [1] in the above mentioned application domains.
(1) The algorithm generates a reach approximation for both arcs of high and low reach. This is preferable in the discussed application domains where it is often more important to know the orders of magnitude of the high than those of the low reaches. (2) The algorithm is suitable when very low or variable calculation times are required, since it improves its approximations over each iteration, involving a shortest path tree calculation, whereas the other algorithms require a partial shortest path tree computation in every vertex of the network. Such a situation occurs for instance in dynamically changing networks where travel time weigths are determined by traffic-related conditions.
Adversely, the approximations generated by this algorithm are lower bounds to the reaches. Therefore, resulting hierarchies cannot be used for enhancing the computation of exact shortest paths.

2. APPROXIMATION ALGORITHM

Preliminaries. A reach hierarchy assigns a separate degree of importance ($\mathbb{R}^+$) to every arc or vertex in the network. Given a reach metric $m : A \to \mathbb{R}^+$, Gutmann [2] defined the reach of a vertex v in a shortest path P, as the minimum of $\sum_{r \in P_1} m(r)$ and $\sum_{r \in P_2} m(r)$, where P_1 and P_2 denote the subpaths of P prior and posterior to v. This is shown in Figure 1a. The reach of vertex v in the digraph G is the maximum of the reaches of a vertex v in any shortest path P in G.
The analogous definition of the reach of an arc r in a shortest path P, denoted $r_P(r,P)$, is shown in Figure 1b. The reach of arc r in a shortest path tree T or a digraph G, denoted respectively as $r_T(r,T)$ and $r_G(r,G)$, is the maximum of the reaches of a vertex v in any shortest path P, respectively appearing in T or G. When no shortest paths in G containing r exist, $r_G(r,G) = 0$.

Algorithm. The iterative algorithm introduced in the present section generates approximations $r_{app}(r)$ of the arc reach $r_G(r,G)$ of any arc $r \in A$ in a digraph $G = (V,A)$. The main algorithm is based on generating a number of shortest path trees rooted in random vertices. It can be outlined by the following steps.

```
1. Initialize $V_{visited} := \{\}$
2. Initialize for any $r \in A$: $r_{app}(r) := 0$
3. Do $maxiter$ times
   (a) Pick a random vertex $v_{root} \in V \backslash V_{visited}$
   (b) $V_{visited} := V_{visited} \cup \{v_{root}\}$
   (c) Calculate the shortest path tree $T$ rooted in $v_{root}$
       using Dijkstra's algorithm, registering $m'(v) :=$
       $m_{acc}(P)$ for every path $P$ in $T$ that starts in $v_{root}$
   (d) Update $r_{app}(r) := max(r_{app}(r), r_T(r,T))$ for every
       arc $r$ in the shortest path tree $T$ (procedure
       detailed below)
```

The values of $m'(v)$ collected in step 3c support the efficient update of reach approximation values in step 3d. When the reach metric m equals the native arc weight metric, $m'(v)$ is already implicitly collected by Dijkstra's algorithm.

The algorithm realizes a stepwise increase of the approximative values, preserving a lower bound of $r_G(r,G)$. It can be shown [3] that $r_{app}(r) = r_G(r,G)$ for every arc r when the algorithm iterates over all the shortest path trees in G. This is, as indicated in Section 1, not the most efficient method to calculate the exact reaches of the arcs in G.
Step 3d of the main algorithm consists of the following steps.

```
1. Initialize $V'_{visited} := \{\}$
2. Collect the list $L$ of leaf vertices $v_l$ in $T$, in descending
   order of $m'(v_l)$
3. For each vertex $v_l$ in $L$ do
   (a) $v_b := v_l$
   (b) While $v_b \notin V'_{visited} \wedge v_b \neq v_{root}$
       i. Given $r = (v_a, v_b)$, which is the arc in $T$ of
          which $v_b$ is the destination vertex, update

          $r_{app}(r) := max(r_{app}(r), min(m'(v_a), m'(v_l) - m'(v_b)))$

       ii. $V'_{visited} := V'_{visited} \cup \{v_b\}$
       iii. $v_b := v_a$
```

As illustrated in Figure 2, each arc in the tree is only visited once, during an iteration started in leaf vertices v_l for which $m'(v_l)$ is maximal. During this visit, each $r_{app}(r)$ is updated to $r_T(r,T)$, shown [3] to be equal to $min(m'(v_a), m'(v_l) - m'(v_b))$.

Joint reach variant. Since in many common transportation network representations, reverse arcs correspond to the same geometry and have the same weights, it could be interesting from a practical point of view (cf. storage and visualization), to store one single reach (approximation) for every pair of reverse arcs. In this case, where $G^e = (V,E)$ is the equivalent undirected graph of $G = (V,A)$, $r^e_G(e,G^e)$ refers to the joint reach, which is the maximum value of the original reaches, and is approximated by $r^e_{app}(e)$.

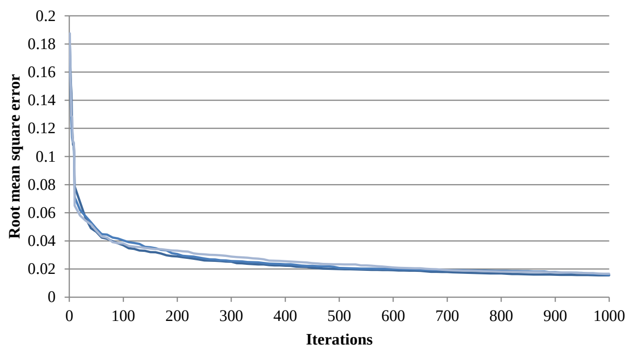


Figure 3: RMSE of the joint reach approximations as a function of the number of iterations performed for three random seeds.

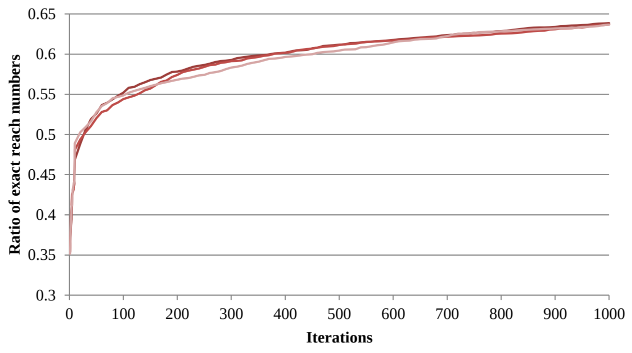


Figure 4: ER of the joint reach approximations as a function of the number of iterations performed for three random seeds.

3. EXPERIMENT

The goal of the present experiment is to study the convergence of the reach approximation algorithm on a real road network of considerable size. The network of interest is an attractiveness-weighted cycling network in Luxembourg, enabling a recreational point-to-point navigation service offered by the company RouteYou. In this network, the arc weights $w(r)$ equal $l(r)/a(r)$ where $l(r)$ refers to the physical length of the corresponding road segment and $0 < a(r) \leq 1$ to its attractiveness. The company determines attractiveness by means of a linear combination of a set of parameters referring to an arc's scenic, physical and traffic-related context. These parameters have been extracted from a third-party geographical dataset. When a new outdoor activity mode is launched, the scalars of the linear combination are fine-tuned during a manual point-to-point navigation sensitivity analysis in the company's labs. The network of interest consists of 44289 vertices.

Methods. The basic setting of this experiment involves a comparison of the set of reach approximations with the set of *exact* reaches in the network, after the execution of n iterations by the joint variant of the approximation algorithm presented in Section 2. This comparison is based on two measures:

- the root-mean-square error (RMSE) of the joint reach approximations, defined as

$$RMSE = \sqrt{\frac{\sum_{e\in E}(r^e_G(e,G^e) - r^e_{app}(e))^2}{|E|}} / \max_{e\in E}(r^e_G(e,G^e))$$

- and, the exactness ratio (ER), which is the ratio of the edges $e \in E$ for which $r^e_G(e,G^e) = r^e_{app}(e)$.

This approach has been repeated for three different seeds initializing a pseudorandom number generator required by the random vertex selection in step 3a of the approximation algorithm.

Next, an analysis has been made of the 3-level hierarchy, obtained from discretizing all reach hierarchies generated in the previous phase of the experiment. Discretization is realized by a straightforward algorithm transforming the reaches into a set of 3 discrete levels. The T-version of this algorithm selects the edges with a reach higher than a specified threshold value. Next, both loose ends and small loops connected to only one edge are repetitively removed in the resulting network. The P-version of the algorithm utilizes a percentile threshold instead of an absolute one.
Both versions were executed on every reach hierarchy obtained from n approximation iterations. The following (percentile) threshold settings

$$T_1 = 26444.65; T_2 = 50624.96; P_1 = 0.20; P_2 = 0.10$$

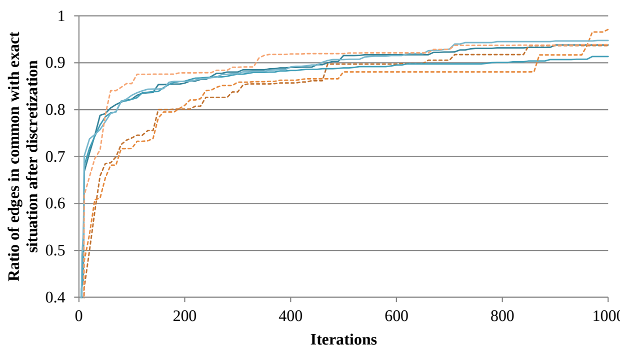
were selected and refer to the same set of edges in the exact reach hierarchy. The edges-in-common rate (EICR) is defined as

$$EICR = \frac{|E_l \cap E^R_l|}{max(|E_l|, |E^R_l|)}$$

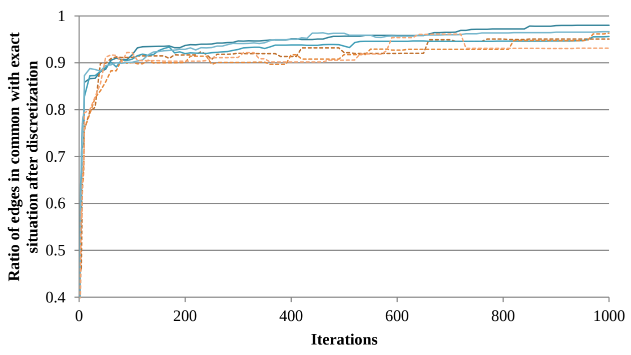
where E_l and E^R_l refer to the edges in level l obtained from discretization of respectively an approximate and the exact reach hierarchy.

Results. Figure 3 shows a steady decline of the RMSE for the first 1000 iterations. The next figure depicts a steady incline of the ER for the same iteration scope. It should be noted that $r^e_G(e,G^e) = 0$ for 19.55% of the edges. In Figure 5, the EICR for E_1 and E_2, resulting from both versions of the discretization algorithm, is shown as a function of the original n approximation iterations. For 2 out of 3 seeds, the T-version shows a steeper incline for E_1 than for E_2, underlied by a smaller population of edges of highest $r^e_{app}(e)$-value. For the T-version, the EICR is monotonically increasing since S_1 and S_2 are monotonically increasing subsets of their *exact* equivalents. Since the P-version implies variable thresholds T_l, S_1 and S_2 are not monotonically increasing and are not necessarily subsets of their exact equivalents. Table 1 provides an overview of the evolution of the RMSE and the ER of the approximated reach hierarchy and the EICR of the corresponding discrete hierarchies.

In general, the results give evidence of a steady approach of the reach approximations towards the exact $r^e_G(e,G^e)$-values during the first 1000 iterations. About 1000 shortest path tree calculations are sufficient to generate a reach hierarchy with an RMSE-deviation of 1.6% and containing 63.7% approximations corresponding to $r^e_G(e,G^e)$. For higher numbers of iterations, the RMSE decline becomes less apparent and the approximation algorithm is less effective.



(a) T-version



(b) P-version

Figure 5: EICR of the edge sets E_1 (solid line) and $E_2 \subseteq E_1$ (dashed line) resulting from applying the discretization algorithm to a reach hierarchy approximation, as a function of the number of original approximation iterations for three random seeds.

Table 1: Summary of the experiment results. The prefixes T- and P- refer to the T- and P-versions of the discretization algorithm. The minimum and maximum value is provided for experiments based on any of the three random seeds.

Itera.	RMSE	ER	T-EICR S_1	T-EICR S_2	P-EICR S_1	P-EICR S_2
100	0.03697 - 0.04037	0.54401 - 0.55209	0.82213 - 0.83070	0.71731 - 0.85546	0.90111 - 0.91699	0.90072 - 0.92186
1000	0.01548 - 0.01661	0.63675 - 0.63865	0.91339 - 0.94759	0.93641 - 0.97114	0.95627 - 0.98010	0.93131 - 0.96339
5000	0.00702 - 0.00797	0.71642 - 0.72048	0.97046 - 0.97861	0.98366 - 0.99285	0.97861 - 0.98498	0.94968 - 0.98683
10000	0.00486 - 0.00561	0.76417 - 0.78488	0.98168 - 0.98708	0.98723 - 0.99515	0.98878 - 0.99174	0.97517 - 0.98683
20000	0.00258 - 0.00300	0.84158 - 0.86597	0.99354 - 0.99608	0.99234 - 1.00000	0.99492 - 0.99534	0.97607 - 0.99542
30000	0.00160 - 0.00197	0.89980 - 0.92410	0.99608 - 0.99682	1.00000	0.99577 - 0.99577	0.99542 - 0.99974
40000	0.00060 - 0.00087	0.97121 - 0.98179	0.99682 - 0.99693	1.00000	0.99682 - 0.99693	0.99974 - 1.00000
44289	0.00000	1.00000	1.00000	1.00000	1.00000	1.00000

Although not guaranteed, the discretization algorithm easily yields hierarchies where both subgraphs are connected. A steady incline during the first 1000 iterations, very similar to the RMSE's incline, is perceived for the EICR of subgraphs arisen from T-discretization, although high thresholds (leading to small subgraphs) seem to generate more variation in the results due to differences in the random root vertex selection order. For a few hundreds of iterations, the EICR values are considerably higher and more stable for P- than for T-discretization. After 100 iterations, both S_1 and S_2 have more than 90% of edges in common with their exact equivalents. This fact confirms the assumption of linear correlation between the $r^e_{app}(e)$-values after a low number of iterations and their corresponding reach $r^e_G(e, G^e)$. For higher numbers of iterations, the P-version does not produce significantly better results than the T-version.

4. CONCLUSION

The contribution of the present paper is a method iteratively increasing the lower bounds of the reaches of the arcs in a digraph. It is based on shortest path tree calculation. Experiments showed that these lower bounds steadily approach the arcs' (joint) reaches after a low number of iterations. In a network of 44289 vertices, an ER of 63.7% was reported after 1000 iterations. For higher numbers of iterations, the algorithm is less susceptible to quick improvement of the lower bounds. Whereas the algorithm approximates arc reaches in a digraph, a similar algorithm of similar performance can be formulated for computing vertex reaches.
In addition, a straightforward algorithm transformed (the lower bounds of) the arcs' reaches into a set of $n+1$ discrete levels. The experiment indicated that it is feasible in practice to obtain connectedness for the subgraphs S_l containing arcs of the levels l when the base network is connected. When the joint reach variant of the algorithm is applied to a set of lower bounds generated in 100 iterations for $n = 2$, the subgraphs S_1 and S_2 have already more than 90% of arcs in common with their exact equivalents.

Acknowledgments. This work is supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX. We would like to thank Pascal Brackman and Benjamin Van Bever from RouteYou for the data provision and part of the problem specification and Wim Vancroonenburg and Koen Decroix for the interesting discussions with regard to the present paper.

5. REFERENCES

[1] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A*: Efficient point-to-point shortest path algorithms. In *Proceedings of the eighth Workshop on Algorithms Engineering and Experiments*, volume MSR-TR-200, pages 129–143. Society for Industrial and Applied Mathematics, 2006.

[2] R. J. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In L. Arge, G. F. Italiano, and R. Sedgewick, editors, *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, January 10, 2004*, pages 100–111. SIAM, 2004.

[3] J. Maervoet, G. Vanden Berghe, and P. De Causmaecker. Fast approximation of reach hierarchies in networks: Related work, proofs and coherence. Technical report, KU Leuven, Department of Computer Science, 2014.