Fast travel-distance estimation using overhead graph
Radu Mariescu-Istodor and Pasi Fränti
School of Computing, University of Eastern Finland, Joensuu, Finland

ABSTRACT
Shortest path can be computed in real-time for single navigational instructions. However, in complex optimisation tasks, lots of travel-distances (lengths of shortest paths) are needed and the total workload becomes a burden. We present a fast method for estimating the travel-distance from one location to another by using an overhead graph that stores the ratio between the bird-distance and the travel-distance for few representative locations. The travel-distance is then estimated for any two locations using the corresponding value between their nearest nodes in the graph. We test the method within an optimization setting where the goal is to relocate health services so that the travel-distance of patients is minimised. We build the overhead graph using road network information from Open Street Map and experiment with real-world data in the region of North Karelia, Finland as a part of the ongoing IMPRO project. The results show that the average estimation error is 0.5 km with a graph of 512 nodes, and the total processing time reduces from 1.2 hours to 2.9 seconds per iteration in the optimisation process. The error in the estimated travel-distances is 2%, on average, which is significantly smaller than 8% of the second best estimation method.

1. Introduction

The shortest path between two locations can be calculated using path-finding algorithms such as Floyd, Dijkstra or A* (Cormen 2009) assuming that road-network information is available in the region. On large networks, these algorithms become impractical and speed-up techniques such as contraction hierarchies (Geisberger et al. 2008) are used to generate shortcuts (precomputed distances) between selected nodes.

Using this enhanced structure, the existing path-finding algorithms perform in a matter of milliseconds. This functionality is nowadays so common that many Application Programming Interfaces (APIs) support it. Anyone can install a routing engine such as the Open Source Routing Machine (OSRM). This engine needs a road-network such as the one provided by Open Street Map (OSM). This brings some complications such as hardware requirements: a server with...
sufficient memory and a powerful processor. One also needs to update the road-network data regularly. But once these are solved, the method is fast (Huber and Rust 2016). For these reasons, many prefer the more convenient alternative: to use a routing service such as those provided by Bing or Google.\(^3\,^4\)

The shortest path itself is not useful in many applications such as optimisation tasks. There, the only thing needed is its length: the travel-distance. Moreover, in optimisation tasks millions or even billions of such travel-distances are needed, which is a computational challenge. In this paper, we propose a fast method to estimate the travel-distance without the need to compute the shortest path. The method is fast enough to optimise large problem instances in some cases in real-time using no hardware enhancements (a typical computer).

In some applications, other properties than the travel-distance are more useful like the travel-time (duration of the fastest path). There also exist others like exposure to sunlight (Li et al. 2019), travel-safety (Krumm and Horvitz 2017) and CO\(_2\) emission (Kuo and Wang 2011; Lin et al. 2014). The same path-finding algorithms can be used to obtain these other values assuming that additional contextual information exists. However, the quality of this information also affects when computing the fastest path. For example, one may use the road speed limits which are usually a part of the road network database, but a more realistic result would be obtained using predictive traffic (Cristian et al. 2019). Large-scale service providers like Bing and Google support this feature. The proposed method can be used to estimate these other values as well (Mariescu-Istodor et al. 2021). In this paper, we focus on the ability to estimate the travel-distance and explain how it can be used effectively in optimisation tasks.

It is important to note that all previously mentioned services (OSRM, Bing and Google) also provide a batch functionality with their so-called table service or distance matrix APIs. Using this functionality, multiple travel-distances or travel-times can be computed using a single request. This accomplishes two things. First, it removes the overheads caused by sending the request and receiving the response. Second, it takes advantage of the dynamic programming nature of Dijkstra’s algorithm and that subproblem solutions are likely to benefit more than one search task if the shortest paths are similar. These services are more expensive, require more hardware, parallelisation and are limited to only a few thousand pairs. Therefore, if more values are needed, they need to be called multiple times.

In this paper, we consider facility location optimisation as an example application. The goal is to find the best possible locations for some facilities (warehouses, hospitals) so that the total travel-distance is minimised. The optimisation works iteratively and constantly relocates the facilities towards a local optimum. On every iteration, new travel-distances are computed between the new facility locations and the targets. This results in a quadratic number of travel-distance calculations per iteration. The number of iterations
correlates with the quality of the solution; the higher the better. Millions or even billions of travel-distances are therefore required. If the travel-distances are slow to compute this becomes a problem as pointed out by (Hidaka and Okano 2003) where a manufacturing company with 6000 customers and 380,000 potential warehouse sites needs to be optimised for travel-distance. The travel-distance calculations are the most time-consuming part of the process and they had to subsample the data to fewer locations to be able to get result.

Our application is very similar to (Hidaka and Okano 2003). However, we provide an alternative solution and attack the core problem: speeding up the travel-distance calculations. We do this by relaxing the condition that the value must be exact and accept an estimation instead. We argue that this is acceptable for three reasons. First, the road network database is not perfect. Second, there is no guarantee that people will take the same route as used in the optimisation process. Third, since the optimisation process is NP hard, applicable algorithms will provide sub-optimal solution anyway. We can therefore compromise the exactness of the travel-distance calculations as well. This brings a major speed-up into the overall optimisation process and allows to solve the complete problem instance without subsampling. The only potential issue is how much we compromise the quality when doing this.

Many other optimisation tasks exist where travel-distance estimates would help (Tong and Murray 2012). One example is maximising coverage (Church and Murray 2018; Farahani et al. 2012). These are needed in energy-storing scenarios (Leuthold, Weigt, and von Hirschhausen 2012); or when patients need to be treated within a given time otherwise consequences can be fatal (Terkelsen et al. 2010; Berlin 2016). In general, patients living closer to health stations have a better health situation (Kelly et al. 2016). Some other models include minimising the distance to the farthest target or the average time or distance (Farahani, SteadieSeifi, and Asgari 2010). Another example is itinerary optimisation (Sengupta, Mariescu-Istodor, and Fränti 2018, 2019; Dantzig and Ramser 1959) which recently became more difficult due to a surplus of home delivery tasks caused by the currently ongoing COVID-19 pandemic. This created an urgent need for some methods to scale up to support more locations (Cristian et al. 2019; Mariescu-Istodor et al. 2021).

Travel-cost estimates are not only useful in optimisation scenarios. They can play an important role in path prediction too (Mariescu-Istodor, Ungureanu, and Fränti 2019; Krumm, Gruen, and Delling 2013), where travel-distances are measured from the user current location to every destination candidate. These values are updated as the user moves to determine if the user is heading towards a specific candidate or not.

In summary, many applications need travel-distance, time or other similar calculations. In some cases the exact values are not critical and an estimation is acceptable. Moreover, optimisation tasks require so many calculations that exact values
cannot be obtained in a reasonable amount of time and a compromise needs to be
made either by subsampling the data or speeding up the calculations using
approximate methods (Yiu and Mamoulis 2004). In this paper we focus on the later.

The simplest and perhaps the most popular approximation is the Euclidean
distance or the great circle distance when the locations are polar coordinates on
the surface of the Earth. The exact mathematical formula depends on the format
of the coordinates (L2 norm or haversine formula). For simplicity, we refer to this
as the bird-distance throughout this paper. It underestimates the travel-distance
as it does not consider the detours caused by the road-network.

Boscoe, Henry, and Zdeb (2012) found that multiplying the bird distance
values by 1.4 gives a reasonable estimate for the travel-distance in US cities. The
authors also concluded that if the application is not critical in some way and if
geography is not too influential simple multiplication by this constant can be
applied. In our case, however, facilitating access to hospitals can be critical.
Moreover, our location in Finland contains thousands of lakes, which impose
significant detours on certain regions. We will compare our method using the
bird-distance as a baseline.

Using the bird-distance is not taking into consideration the road-network
within the region. Love and Morris (1972, 1979, 1988) examined different dis-
tance estimators for rural and urban areas in the United States and Germany.
They also used a constant multiplier $k$ to the L2 norm to make it closer to the
travel-distance. This multiplier is a parameter in their case. Moreover, they
analyse a generalisation called $l_{k,p}$ norm. This generalisation encompasses
both the rectangular norm ($p = 1$) and the L2 norm ($p = 2$):

\[
l_{k,p}(x, y) = k ||x_1 - y_1|^p + |x_2 - y_2|^p ||^{1/p}, \quad k > 0, \quad p \geq 1
\]

where $x = (x_1, x_2)$ and $y = (y_1, y_2)$ are the two locations, and the parameter $k$
accounts for hills, valleys and other noise. The proposal by Love and Morris
(1988) is to rotate the coordinate system by angle parameter $\theta$ so that the roads
in the region orient with respect to the North-South direction:

\[
\begin{pmatrix}
  x_1' \\
  y_1'
\end{pmatrix}
= \begin{pmatrix}
  x_1 \\
  y_1
\end{pmatrix}
\begin{pmatrix}
  \cos \theta & -\sin \theta \\
  \sin \theta & \cos \theta
\end{pmatrix}
\]

The two parameters are fine-tuned in a preprocessing step and the distance
estimation is then performed on the transformed points $l_{k,p}(x', y')$. This method
was demonstrated to work well also in other studies, such as (Brimberg, Walker,
and Love 2007), where the superiority of the $l_p$ norm is demonstrated. An
interesting phenomenon is also discussed: within a rotation of 90 degrees,
two angles seem to exist where the error is minimised. This means that in the
preprocessing step it is enough to consider a rotation up to 45 degrees only to
find the best parameter setting, which reduces the preprocessing time.

The above method works well if the structure of the road-network displays a
consistent grid-like pattern. For example, the street structure in Manhattan can
be rotated by 60 degrees to align it almost perfectly in north-south direction. However, in many other cities, this is not possible for various reasons like when the topography interferes too much (e.g. lakes in Finland), when the city evolves without a specific plan, or when the city has a radial layout such as the city of Timisoara, Romania (see Figure 1).

In this paper, we present a novel method to estimate the travel-distance based on overhead, which is similar to the multiplier used in the above-mentioned methods. The difference is that we do not use a single constant for all distances but optimise the overhead locally. We generate an overhead graph, which contains pre-calculated travel-distance information for a small subset of representative locations. The wanted travel-distance between two points can then be estimated based on the pre-calculated overheads between their nearest representatives in the graph. This graph is usually small and can be generated by calling a Routing or Distance Matrix API.

2. Proposed estimation method

The proposed method is based on two independent components: the bird-distance (Euclidean or great circle distance) and the pre-calculated overhead graph. The bird-distance is an obvious approximation of the travel-distance, but it can seriously under-estimate the real distance. In O-Mopsi orienteering game (Fränti et al. 2017), bird-distance was found more useful than road-network because the travel is done on foot and the game instances include lots of park fields, forests and other open areas which are usually not modelled by the road-network. However, in our target application, the travel is expected to be done by cars or public transportation.

Bird-distance provides a strict lower bound for the true distance, and it can be used for the approximation as follows. If we knew that, in a given region, the
road-network distance is always $T$ times longer, we could trivially calculate the travel-distance as follows:

$$d_{\text{road}}(A, B) = T \cdot d_{\text{bird}}(A, B)$$  \hspace{1cm} (3)

where A and B are two locations. We denote the value exceeding the road network distance as as the \textit{overhead}. For example, if the overhead is 40\%, the distance is multiplied by a factor $T = 1.4$. The overhead value varies a lot from a region to another, and therefore, is not useful as a single constant. If we instead wish to estimate the time, for example, the equation remains the same: $t_{\text{road}}(A, B) = T \cdot d_{\text{bird}}(A, B)$, however, $T$ now acts also as a converter from distance to time. In this way, we can estimate other properties as well, such as sunlight exposure, travel safety or CO$_2$ emissions.

A given region is pre-processed as follows. We sub-sample the region by selecting $n$ fixed locations. We then calculate both the bird-distance ($d_{\text{bird}}$) and the road-network distance ($d_{\text{road}}$) between every pair of locations. Based on these two values, we calculate the overhead using equation (3). The overheads (T-values actually) are stored in the \textit{overhead graph}. An example is shown in Figure 2.

The number of nodes in the graph $G(N, L)$ is denoted by the parameter $n$. The number of links is $n^2$, as the travel-distance between two locations is not necessarily symmetric (due to one-way roads). The nodes can be scattered uniformly within the region, however, it is better to optimise them based on any data available in the application. We will discuss these options in the experimental section.

Given two locations, \textit{source A} and \textit{destination B}, we use the overhead graph to estimate the travel-distance as follows:

1. Calculate bird-distance between the source and destination $d_{\text{bird}}(A, B)$
2. Find their nearest nodes $NN(A)$ and $NN(B)$ in the overhead graph
3. Take the overhead factor from overhead graph as $T = \text{Graph}[NN(A), NN(B)]$

Figure 2. On the left, representative locations within a region (black dots) with three example travel paths and their distances. On the right, the three respective multipliers are shown. Links exist between all possible location pairs, we only show few to avoid congestion.
4. Calculate travel-distance estimation as \( d_{\text{road}}(A, B) = T \cdot d_{\text{bird}}(A, B) \)

In other words, we use the bird distance between the source and destination as the base of the estimation, and then scale it by multiplying by the overhead. The overhead is taken from the graph as the pre-calculated overhead value between the points, \( \text{NN}(A) \) and \( \text{NN}(B) \), that are nearest to the source and destination points \( A \) and \( B \). This estimation is not usually symmetric \( d_{\text{road}}(A, B) \neq d_{\text{road}}(A, B) \) because of the use of road network.

An example is shown in Figure 3 where the bird distance is 1.1 km and the corresponding overhead value is 50\%, which results in an estimation of \( 1.1 \times 1.5 = 1.65 \) km. The estimation is more accurate when the source and destination are close to the graph nodes \( A \) and \( B \). A special case occurs when two locations are mapped to the same node. In this case, we calculate the overhead as the minimum overhead from the node to any other.

The accuracy of the method depends on two factors: the size of the graph, and the location of the nodes. Increasing the number of nodes decreases the average distance to the nodes, which improves the accuracy of the distance estimation. However, large graphs require more memory and can slow down the method.

The nearest neighbour search is the bottleneck of the estimation. It takes \( O(n) \) using straightforward linear search. The nodes can also be indexed using kd-tree or similar structures to achieve \( O(\log(n)) \) time. In optimisation tasks, however,

![Figure 3. Components used when estimating the distance from source to destination. In our application, the source is a patient’s home and destination is the hospital where the patient needs to visit. The bird-distance between the two is 1.1 km. The two nearest nodes of the overhead graph are connected by a link with \( T = 1.5 \).](image-url)
the application data is mostly static, and we can store the nearest nodes to every location in a lookup table. Locations that change during the optimisation can be immediately paired to their nearest nodes and added to the lookup table at the cost of $O(n)$, each. This step usually amounts to a fraction of the massive amount of estimations needed and in some applications it is even zero (Mariescu-Istodor et al. 2021). This reduces the time complexity of the distance estimation to $O(1)$, in practice.

The accuracy of the method is expected to increase when adding more nodes. An extreme case is when every possible location is included in the graph and the distance to their nearest node becomes zero. Then, all estimations revert to the original values used to compute the overhead graph and the estimation is perfect. This also means that there is no need to implement a separate functionality for small problem instances where the overhead graph acts as a look-up table without any approximation error.

3. Facility location optimisation

The overhead graph is needed in several optimisation tasks in the IMPRO project. An example is to optimise location of the health centres so that the travel-cost would be minimised. Alternatively, we can minimise the number of patients whose travel time exceeds a predefined threshold (90 minutes), which is considered critical in the case of cardiovascular diseases.

We focus here on the first example. This is formulated as a clustering problem where we minimise the total travel-distance of all patients to their nearest health centre. We note that the real-world problem is more complex than this because, in reality, we cannot optimise the location freely as lots of infrastructure is already existing and moving a hospital even by 500 metres would be unrealistic. In addition, the patients are expected to travel by different means of transport (walking, car, bus, taxi) according to the model in (Leminen et al. 2018). We plan to use this as our travel-cost instead of the travel-distance, however, predicting the mode of transport is currently too time consuming and we omit it in this paper. The current optimisation is still valuable as such at providing information to decision makers, who can analyse the data and forecast future populations to see if the optimised facility locations would change due to course of time.

We experiment on patient locations from Siun Sote health organisation in North Karelia, Finland. The data was collected during several years and it contains 10,147 patients and 23 health centres in the region. Exact locations of the patients and the health centres are known. The number of visits per patient varies from 1–6, and the same patient may have visited several health stations.

The selected data were originally used for other research purposes and includes mainly patients with alcohol-use disorder (Rautiainen, Ryynänen, and Laatikainen 2018), type 2 diabetes (Toivakka et al. 2018), coronary heart disease
(Repo et al. 2018) and anticoagulation (Leminen et al. 2019). The data is therefore not an exact representation of the entire population, but its geographic distribution is rather similar, and it is useful for our study. An example of the patient distribution in Joensuu city area with three health centres are shown in Figure 4.

The optimisation relies on the travel-distances. The task involves trial and error iterations where health centre locations are repositioned. After every change, the total travel-distance is recalculated which is very time-consuming taking 1.2 hours per iteration using standard API calls. This slows down the optimisation process significantly. The travel-distance computation is the bottleneck. Hardware improvements could also be possible with better processor, parallelisation and cloud computing. However, our target audience is a small number of decision makers and algorithmic improvement is therefore more feasible.

A decision to compromise the precision of the distance calculation was made by using the overhead graph. This is acceptable because patients do not necessarily travel on the shortest path; commonly people are more concerned with time and may choose the fastest path instead. The distance estimation is therefore not critical and not needed to be perfect.

We also provide an interactive optimisation tool to the decision makers who can manually relocate certain health stations on the map to see the effect on total cost. In this scenario, the computer then performs the optimisation, or fine-tunes the one given by the user. This is very important as the decision makers need to consider things like: can a hospital be relocated, or whether it is possible to build a hospital in the given location at all. The exact but slower distance calculation would make the waiting times unacceptable. Using the

Figure 4. Patient locations and the current health centres in North Karelia (left) and Joensuu city centre (right). The patient locations are shown using a heat-map to preserve anonymity.
overhead graph, the values are generated within 3 seconds, on average, keeping the tool interactive.

4. Experiments

We study next the behaviour of the proposed method under different circumstances.

4.1. Graph creation

There are many ways to create an overhead graph depending on the data available in a particular region. If we do not know the application data beforehand, we can create uniform distribution of the nodes, but this would be too naïve (see Figure 5). A better choice is to utilise the road network and take random intersections to avoid unreachable places like the middle of lakes. We can also cluster the road network locations because random sampling tends to favour larger cities. The road network of our region has 11,756 road segments and 7,536 intersections.

In our application, we know the location of the patients for whom the optimisation is performed. Clustering these locations gives better representatives than the road network. We use Random Swap clustering algorithm (Fränti 2018) with 3,000 iterations with a fixed number of clusters.

Once the node locations are selected, we call the OSRM API (installed locally on our server) to obtain all pairwise travel-distances between the nodes and then calculate the overheads. In Figure 6, we can see sample overhead graphs when the number of nodes is increased from 16 to 64. Links that cross over the water have higher overheads, which indicate that significant detours are needed. Note: when displaying the graphs on map, we show Gabriel links.

Figure 5. Overhead graphs with 16 nodes generated in the North Karelian region using four different methods. The worst estimation is given by the uniform method. The values in the bottom-right corners are the average approximation errors.
provided by the XNN algorithm (Fränti, Mariescu-Istodor, and Zhong 2016) for clarity even though they are complete graphs having links between all nodes. We do this to avoid congestion. On average, the overhead value is 40% and the largest values are over 700%. These values are in agreement with (Boscoe, Henry, and Zdeb 2012). In North Karelia, larger values are more common due to many lakes compared to those reported in the US.

In our local implementation, the smallest graph (16 nodes) is generated fast but the larger graphs takes significantly longer time, up to several hours with 512 nodes. One reason for the slowness is that we do not perform just a single large request to the Distance Matrix API due to hardware and size limitations. Instead, we make multiple Routing API calls. However, the slowness of the graph creation does not matter much because it needs to be done only once, and this can be performed off-line.

The storage space of the graph depends on the number of nodes. Each of the \( n \) nodes requires two floating point numbers (latitude and longitude) and there are \( n^2 \) links which contain the overhead values (floating point numbers). Selected memory consumption values are given in Table 1.

### 4.2. Method comparison

The optimisation process of the health center locations is based on the Random Swap clustering algorithm which uses two iterations of K-Means (Lloyd 1982) after every swap. The most time-consuming step is the partitioning step of K-Means, which calculates distances between every data point and centroid. In

![Figure 6. Patient locations clustered with Random Swap algorithm using 16, 32 and 64 clusters; the blue centroids are the nodes of the overhead graph. Selected links from the overhead graph are shown. The links are coloured so that the higher the overhead, the redder the link. The values in the bottom-right corners are the average approximation errors.](image)

| Nodes | 2     | 4     | 8     | 16    | 32    | 64    | 128   | 256   | 512   |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Size  | 32 B  | 96 B  | 320 B | 1 KB  | 4 KB  | 16 KB | 65 KB | 258 KB| 1 MB  |

Table 1. The size required by the overhead graph with different number of nodes.
our case, this corresponds to calculating travel-distances from every patient to every health centre, which sums up to \(10,147 \times 23 = 233,381\) travel-distance estimations. In the following analysis, we do not focus on this overall optimisation process as a whole, but focus on the calculations of the 233,381 travel-distances.

First, we measure the quality of the estimation when the size of the overhead graph is varied within \(\{2, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8\}\) (see Figure 7). The quality of the estimation is measured as the absolute difference between the estimated and the real travel-distance using the road network. We report the average difference of the 233,381 estimations. The proposed method tends to balance between under and over-estimations so that the average error is close to 0.

From the results, we can see that increasing the number of nodes improves the approximation so that the average error decreases to about 500 metres when the number of nodes reaches 512. The approximation error is never bigger than the error of the bird-distance. It drops fast when more nodes are added and outperforms also Manhattan (city block) distance when 128 nodes are used. With 256 nodes, the result outperforms also the general norm proposed by (Love and Morris 1988). Their method was optimised by considering a wide range of possible \(p, k\) and \(\theta\) values. We experimentally found the best values with the Siun Sote data in North Karelia as \(p = 1.07, k = 1.08\) and \(\theta = 62\) degrees. When increasing to 512 nodes, our method achieves 50% lower error than the Love and Morris method. Decrease in the standard deviation of the error values is also observed when the nodes increases.

In the optimisation task, we are more concerned with the total travel-distance of all patients to their nearest health centre. We therefore investigate also this total error in Figure 8. Bird-distance causes 30% error. This is significantly higher than Manhattan and Love and Morris because it always underestimates. The other two methods can also overestimate in different situations, and some of the values cancel out when the total error is measured. Using the overhead graph, there is a tendency to underestimate the error until graph size 32 is reached. At that point, the error decreases down to 2% when graph size is 512, surpassing Love and Morris, which is the second best method with error of 8%.

![Figure 7. Error of the travel-distance estimation using multiple methods. The error of the proposed method is shown when varying the number of nodes of the overhead graph (blue).](image-url)
There is no other drawback of using a larger overhead graphs except for the increased memory demand. The processing times for estimates are constant for the most part (see Figure 9). It is very important to note that nearest neighbour search from a patient location to its nearest overhead graph node is not needed because the patient locations are fixed and they are already mapped to the nearest node in the overhead graph during the clustering process.

If travel-distances between new locations were needed, it would require additional $O(n)$ operations to find the nearest graph node. This happens in our optimisation task because centroids keep on moving to new locations and

![Figure 8](image1.png)

**Figure 8.** Relative error of the total travel-distance from patients to their nearest health centre. The error of the proposed method is shown when varying the number of nodes of the overhead graph (blue).

![Figure 9](image2.png)

**Figure 9.** Processing times for all methods. The number of nodes have almost no effect on the speed of the proposed method (blue).
their nearest node in the overhead graph needs to be updated accordingly. This requires \(O(n)\) time for each of the 23 centroids. With the largest graph size of \(n = 512\), this induces 11,776 operations in total. This is still just a fraction (5\%) of the total work required in each iteration, and therefore, not significant enough to require spatial indexing or other typical speedup tricks. We do, however, recommend this if the step becomes a bottleneck.

All compared methods work seemingly in constant time; Love and Morris is the slowest due to the rotation step which consists of trigonometrical functions. The true travel-times took 1.2 hours using our locally installed OSRM. We cannot use the Distance Matrix service as such because of its large size and our hardware limitations. We therefore resorted to performing 233,381 separate requests from each patient location to every health centre. This means roughly 54 requests per second. However, part of this time is due to the requests being created, sent, and the results being retrieved. The requests are made by the optimisation software, not the OSRM codebase, causing additional delays. Previous research on the topic suggests that it is possible to improve this performance significantly (Huber and Rust 2016), where the authors managed to succeed 280 requests per second on a comparable machine, but this does not amount to much in the big picture.

### 4.3. Qualitative observations

We next give some examples to demonstrate how the method behaves in different situations. First, in a city centre area (see Figure 10), the result depends on how the nodes are positioned. If they happen to be on the same road, the method will work perfectly for locations on the same or parallel roads. In other cases, it will result in an underestimation. The opposite can also happen: if the nodes are on different roads, the method can overestimate the travel-distance.

Geographic properties of the region also affect performance of the method. If a river bisects the region, the overhead value becomes unusually high (see Figure 11). This can also cause bigger over or underestimations when they happen. High overestimation happens when the points are further from each other than the corresponding graph nodes (further from the river), but also when they have similar distance as the graph nodes but both are further from the travel path between the graph nodes. In practice, these happen rarely, especially for larger graphs as other nodes are usually in the surrounding area, causing \(A\) and \(B\) to be not too far away from any node. To help the reader understand the quality of the estimation better, we made this interactive tool.

### 5. Conclusions

We have proposed a fast travel-distance estimation based on overhead graph. The estimation works in \(O(n)\) time where \(n\) is the size of the graph, or in \(O(\log(n))\)
time with spatial indexing. However, in optimisation tasks where the locations are already known, the estimation works in constant time by storing the nearest neighbour nodes in a lookup table. Using the proposed method with a graph size of $n = 512$ the processing time is 2.9 seconds, which is 10 seconds faster than the second best alternative proposed by Love and Morris, and over 1 hour faster than a typical routing algorithm (when performing 233,381 estimations).

Compared to other speed-up methods, the proposed method provides estimation error of 0.5 km, compared to the second best alternative, Love and Morris, which provides error of 1 km. The cumulated total error in the travel-distances of the patients to their nearest health centre is only 2%, compared to 8% if using Love and Morris.

The proposed speed-up method provides significant improvement for practical applications as it helps to build interactive tools for decision makers and logistics planners. As a case study, we considered facility location optimisation which requires thousands of distance calculations every iteration. Another application is itinerary optimisation where the optimal tours are re-calculated constantly due to fast-changing traffic situation and new customer orders coming in real-time.

To sum up, the proposed travel-distance estimation can provide huge speed-up for the optimisation of location-based services. In our case study, the total
processing time of one optimisation iteration was reduced from 1.2 hours to 2.9 seconds.

**Future work:** While the method is simple and effective, further accuracy could be obtained by a more complex prediction function. We also think that it would be possible to reduce the memory complexity of the method from quadratic to linear if using a hierarchical graph structure.

**Notes**

1. https://www.stnimpro.fi
2. http://project-osrm.org
3. https://docs.microsoft.com/en-us/bingmaps/rest-services/routes
4. https://cloud.google.com/maps-platform/routes
5. https://www.stnimpro.fi
6. the tool contains confidential data and is only accessible to decision makers
7. http://cs.uef.fi/mopsi/overheadGraph

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

The project was funded by the Strategic Research Council (SRC) at the Academy of Finland (IMPRO project) [Grant number 312706].
References

Berlin, C., R. Paniczak, R. Hasler, and M. Zwahlen. 2016. “Do Acute Myocardial Infarction and Stroke Mortality Vary by Distance to Hospitals in Switzerland? Results from the Swiss National Cohort Study.” BMJ Open 6 (11): e013090. doi:10.1136/bmjopen-2016-013090.

Boscoe, F. P., K. A. Henry, and M. S. Zdeb. 2012. “A Nationwide Comparison of Driving Distance versus Straight-line Distance to Hospitals.” The Professional Geographer 64 (2): 188–196. doi:10.1080/00330124.2011.583586.

Brimberg, J., H. T. Kakhki, and G. O. Wesolowsky. 2003. “Location among Regions with Varying Norms.” Annals of Operations Research 122 (1–4): 87–102. doi:10.1023/A:1026190322164.

Brimberg, J., H. T. Kakhki, and G. O. Wesolowsky. 2005. “Locating a Single Facility in the Plane in the Presence of a Bounded Region and Different Norms.” Journal of the Operations Research Society of Japan 48 (2): 135–147. doi:10.15807/jorsj.48.135.

Brimberg, J., J. H. Walker, and R. F. Love. 2007. “Estimation of Travel Distances with the Weighted ℓp Norm: Some Empirical Results.” Journal of Transport Geography 15 (1): 62–72. doi:10.1016/j.jtrangeo.2006.01.004.

Church, R. L., and A. Murray. 2018. Location Covering Models. Advances in Spatial Science. Springer International Publishing.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. Introduction to Algorithms. MIT Press.

Cristian, A., L. Marshall, M. Negrea, F. Stoichescu, P. Cao, and I. Menache (2019, November). “Multi-Itinerary Optimization as Cloud Service (Industrial Paper).” Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago, IL, 279–288.

Dantzig, G. B., and J. H. Ramser. 1959. “The Truck Dispatching Problem.” Management Science 6 (1): 80–91. doi:10.1287/mnsc.6.1.80.

Farahani, R. Z., M. SteadieSeifi, and N. Asgari. 2010. “Multiple Criteria Facility Location Problems: A Survey.” Applied Mathematical Modelling 34 (7): 1689–1709. doi:10.1016/j.apm.2009.10.005.

Farahani, R. Z., N. Asgari, N. Heidari, M. Hosseininia, and M. Goh. 2012. “Covering Problems in Facility Location: A Review.” Computers & Industrial Engineering 62 (1): 368–407. doi:10.1016/j.cie.2011.08.020.

Fränti, P., R. Mariescu-Istodor, and L. Sengupta. 2017. “O-Mopsi: Mobile orienteering game for sightseeing, exercising, and education.” ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 13 (4): 1–25.

Fränti, P. 2018. “Efficiency of Random Swap Clustering.” Journal of Big Data 5 (13): 1–29. doi:10.1186/s40537-018-0122-y.

Fränti, P., R. Mariescu-Istodor, and C. Zhong (2016, November). “XNN Graph.” In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR) (pp. 207–217). Springer, Cham.

Geisberger, R., P. Sanders, D. Schultes, and D. Daniel. 2008. “Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks.” In Experimental Algorithms, edited by C. C. McGeoch, 319–333. Vol. 5038. Heidelberg: Springer Berlin Heidelberg, Berlin. doi:10.1007/978-3-540-68552-4_24.
Hidaka, K., and H. Okano. 2003. “An Approximation Algorithm for a Large-scale Facility Location Problem.” *Algorithmica* 35 (3): 216–224. doi:10.1007/s00453-002-0996-z.

Huber, S., and C. Rust. 2016. “Calculate Travel Time and Distance with OpenStreetMap Data Using the Open Source Routing Machine (OSRM).” *The Stata Journal* 16 (2): 416–423. doi:10.1177/1536867X1601600209.

Kelly, C., C. Hulme, T. Farragher, and G. Clarke. 2016. “Are Differences in Travel Time or Distance to Healthcare for Adults in Global North Countries Associated with an Impact on Health Outcomes? A Systematic Review.” *BMJ Open* 6 (11): 11. doi:10.1136/bmjopen-2016-013059.

Krumm, J., and E. Horvitz (2017). “Risk-Aware Planning: Methods and Case Study for Safer Driving Routes.” AAAI, San Francisco, CA, February, 4708–4714.

Krumm, J., R. Gruen, and D. Delling. 2013. “From Destination Prediction to Route Prediction.” *Journal of Location Based Services* 7 (2): 98–120. doi:10.1080/17489725.2013.788228.

Kuo, Y., and C. C. Wang. 2011. “Optimizing the VRP by Minimizing Fuel Consumption.” *Management of Environmental Quality: An International Journal* 22: 440–450. doi:10.1108/147778311136054.

Leminen, A., M. Pyykönen, J. Tynkkynen, M. Tykkyläinen, and T. Laatikainen. 2019. “Modeling Patients’ Time, Travel, and Monitoring Costs in Anticoagulation Management: Societal Savings Achievable with the Shift from Warfarin to Direct Oral Anticoagulants.” *BMC Health Services Research* 19: 901. doi:10.1186/s12913-019-4711-z.

Leminen, A., M. Tykkyläinen, and T. Laatikainen. 2018. “Self-monitoring Induced Savings on Type 2 Diabetes Patients’ Travel and Healthcare Costs.” *International Journal of Medical Informatics* 115: 120–127. doi:10.1016/j.ijmedinf.2018.04.012.

Leuthold, F. U., H. Weigt, and C. von Hirschhausen. 2012. “A Large-scale Spatial Optimization Model of the European Electricity Market.” *Networks and Spatial Economics* 12 (1): 75–107. doi:10.1007/s11067-010-9148-1.

Li, X., Y. Yoshimura, W. Tu, and C. Ratti. 2019. “A Pedestrian Level Strategy to Minimize Outdoor Sunlight Exposure in Hot Summer.” *arXiv Preprint arXiv* 1910: 04312.

Lin, C., K. L. Choy, G. T. S. Ho, S. H. Chung, and H. Y. Lam. 2014. “Survey of Green Vehicle Routing Problem: Past and Future Trends.” *Expert Systems with Applications* 41 (4): 1118–1138.

Lloyd, S. 1982. “Least Squares Quantization in PCM.” *IEEE Transactions on Information Theory* 28 (2): 129–137. doi:10.1109/TIT.1982.1056489.

Love, R. F., and J. G. Morris. 1972. “Modelling Inter-city Road Distances by Mathematical Functions.” *Journal of the Operational Research Society* 23 (1): 61–71. doi:10.1057/jors.1972.6.

Love, R. F., and J. G. Morris. 1979. “Mathematical Models of Road Travel Distances.” *Management Science* 25 (2): 130–139. doi:10.1287/mnsc.25.2.130.

Love, R. F., and J. G. Morris. 1988. “On Estimating Road Distances by Mathematical Functions.” *European Journal of Operational Research* 36 (2): 251–253. doi:10.1016/0377-2217(88)90431-6.

Mariescu-Istodor, R., A. Cristian, M. Negrea, and P. Cao. 2021. “VRPDiv: A Divide and Conquer Framework for Large Vehicle Routing Problem Instances.” *ACM Transactions of Spatial Algorithms and Systems*. Submitted Manuscript.

Mariescu-Istodor, R., R. Ungureanu, and P. Fränti. 2019. “Real-time Destination Prediction for Mobile Users.” *The Advances in Cartography and GIScience of the International Cartographic Association* 2, 10. doi:10.5194/ica-adv-2-10-2019

Rautiainen, E., O. P. Ryynänen, and T. Laatikainen. 2018. “Care Outcomes and Alcohol-related Treatment Utilisation Profiles of Patients with Alcohol-use Disorder: A Prospective Cohort
Study Using Electronic Health Records." Nordic Studies on Alcohol and Drugs 35 (5): 329–343.

Repo, T., M. Tykkyläinen, J. Mustonen, T. T. Rissanen, M. Ketonen, M. Toivakka, and T. Laatikainen. 2018. “Outcomes of Secondary Prevention among Coronary Heart Disease Patients in a High-Risk Region in Finland.” International Journal of Environmental Research and Public Health 15 (4): 724. doi:10.3390/ijerph15040724.

Sengupta, L., R. Mariescu-Istodor, and P. Fränti. 2018. “Planning Your Route: Where to Start?” Computational Brain & Behavior 1 (3–4): 252–265. doi:10.1007/s42113-018-0018-0.

Sengupta, L., R. Mariescu-Istodor, and P. Fränti. 2019. “Which Local Search Operator Works Best for the Open-Loop TSP?” Applied Sciences 9 (19): 3985. doi:10.3390/app9193985.

Terkelsen, C. J., J. T. Sørensen, M. Maeng, L. Okkels Jensen, H.-H. Tilsted, S. Trautner, W. Vach, S. P. Johnsen, L. Thuesen, and J. F. Lassen. 2010. “System Delay and Mortality among Patients with STEMI Treated with Primary Percutaneous Coronary Intervention.” JAMA 304 (7): 763–771. doi:10.1001/jama.2010.1139.

Toivakka, M., A. Pihlapuro, M. Tykkyläinen, L. Mehtätalo, and T. Laatikainen. 2018. “The Usefulness of Small-area-based Socioeconomic Characteristics in Assessing the Treatment Outcomes of Type 2 Diabetes Patients: A Register-based Mixed-effect Study.” BMC Public Health 18 (1): 1–7.

Tong, D., and A. T. Murray. 2012. “Spatial Optimization in Geography.” Annals of the Association of American Geographers 102 (6): 1290–1309. doi:10.1080/00045608.2012.685044.

Yiu, M. L., and N. Mamoulis (2004). “Clustering Objects on a Spatial Network.” Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, June, 443–454.