2 RELATED WORK

Several functions have been used in the literature to estimate the distance between two points in a plane such as the Manhattan distance [22], the Euclidean distance [6], and the ℓ_p-norm also known as the Minkowski distance [22]. More sophisticated approaches include the Multi-Layer-Perceptron (MLP) [4] that is able to distinguish between different regions. Since these approaches focus on Cartesian coordinates, the Haversine formula allows to calculate the distance between two points on a sphere [19, 20, 24]. Existing functions for distance estimation do not consider obstacles explicitly, which limits the applicability in real-world optimization problems.

Other approaches explicitly consider obstacles and calculate distances using visibility graphs [3, 5]. In visbility graphs, only nodes that can *see* each other, meaning there is no obstacle in between, are connected by an edge. While the estimated distances using this approach might be more accurate compared to state-of-the-art distance functions, their application in optimization problems such as FLP is infeasible. This is due to a constantly changing graph resulting from new depot locations which comes with a visibility check to all other nodes. In contrast, we propose to preprocess the graph in order to reduces its size and, hence, omit the need for modifications during the optimization process.

3 EXAMPLE PROBLEM DOMAIN

An example domain, where distance estimation could be applied is the facility location problem (FLP) [7, 23]. FLPs address the issue of placing depots as effectively as possible so that a number of customers can be served. Customer locations are given and depots should be placed accordingly. There are several variations of this problem including the integration of existing depots, capacities for the depots, or time constraints that have to be met when serving customers, to name a few examples. Since FLP is known to be $\mathcal{NP}$-complete, solving the problem optimally is usually not feasible, especially if the problem instances are large [23].

Since the problem is hard to solve exactly, optimization algorithms are usually used to find an approximation of the optimal solution. Nature-inspired algorithms like Genetic Algorithms [14], Particle Swarm Optimization [15], NSGA-II [9], or Ant Colony Optimization Algorithms [10] are suitable to achieve this. These algorithms apply concepts of exploration and exploitation and assess a large variety of solutions while searching for the best possible solution. A solution in the context of FLP consists of the locations of the depots that should be placed. In some cases, this can also incorporate further properties of the facilities like their size. The assessment of possible solutions means that the distance between customers and depots has to be calculated many times, significantly increasing the computation time when using shortest path algorithms like Dijkstra's algorithm operating in $O(E \log V)$ [8], based on actual road networks. For example, the road network of Germany consists of 11.5 million nodes [21]. It is also not feasible to save and reuse previously calculated distances, as the depots are moved in each solution and iteration. Even a small movement of a depot could lead to significantly worse results, as the previously used road might no longer be accessible from the new location and a new route planning needs to be triggered.

As an alternative, simple functions like the Euclidean distance or the Manhattan distance can be applied. However, those functions disregard the existence of obstacles and can only provide rough estimates. Estimates often highly deviate from the actual distances and thus jeopardize the quality of the solutions of the aforementioned optimization techniques. Even in countries with highly developed road networks, there are still natural obstacles that make it hard to use such a simplified function. In the case of obstacles like rivers or lakes, the actual distance can increase massively when compared to the estimations, if the next bridge is far away or one has to find a way around.

4 FLEXIBLE DISTANCE ESTIMATION

This section presents the idea of flexible and adaptive distance estimation. We present the general idea in Section 4.1 and then discuss the application of self-aware computing concepts in Section 4.2.

4.1 General Idea

Existing approaches mentioned in Chapter 2 either focus on solving the distance computation exactly or make use of mathematical formulas that try to approximate the distance. This results in either high computation times or strongly simplified estimates. The general idea is to find a trade-off between the accuracy and the computation time when calculating distances between two points in a road network. Note that the presented approach is only meaningful for estimating distances and not intended for actual road network-based route planning.

For this purpose, we aim to construct a simplified graph that captures (1) areas that are passable with high likelihood and (2) areas with obstacles that are generally not passable. This simplified graph is designed to be less complex compared to a road network graph and, hence, should speed up the computation time while maintaining a given level of accuracy. We identified that obstacles strongly influence the actual distance as special routes are required to cross them or to find a way around them. Thus, in the first instance, the approach focuses on water areas and bridges. It is important to note that the approach can and should be extended to also consider other types of obstacles if the use case requires it. Figure 1 illustrates our proposed approach to distance estimation.

In the first step, we create an abstract map that solely contains obstacles, such as water areas, and ways through them, for example, bridges, if they exist. This abstract version can be created directly from available information about the road network and polygons that resemble the obstacles or by extracting knowledge from existing map images. Other information like the exact location of roads, or positions and size of buildings are no longer of interest, as we assume those areas as passable.

In the second step, we place a grid above this rendered abstract map. Then, we analyze every field within this grid and decide whether a vehicle is able to drive through this field (indicated by a white field) or if it is impassable (indicated by a black field). The classification of the fields can be achieved by different methods such as rules, image processing, or neural networks, depending on the complexity of the abstracted map. The resulting black-white grid map can then be used for the construction of an undirected graph, where a node represents a field of the abstracted map. In

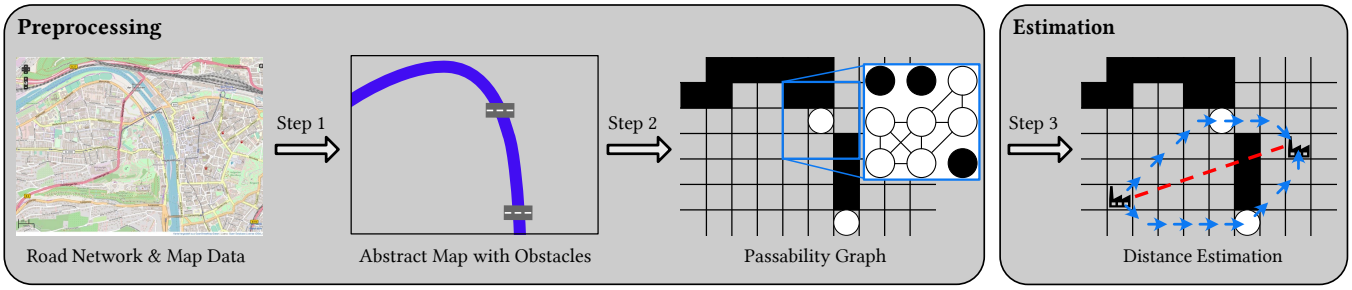

Figure 1: Steps of Flexible Distance Estimation: (1) maps or other data is used to build an abstract map, (2) a grid is used to determine the elements and connections of the passability graph, and (3) distance is calculated based on the passability graph.

case the field is white and there are also neighbors colored white, we create the node and assign edges to all available neighbors as vehicles are able to drive there. If the field is marked black, the corresponding node has no edges assigned. By applying this technique, we reduce the number of nodes in the graph and, thus, speed up the computation time. The resulting graph can then be optimized to enhance the speed of the route calculation within it, for example, by the application of Contraction Hierarchies [11]. The first two steps are considered as a preprocessing phase.

In the third and last step, we calculate the distance between locations. To do so, one selects a start and target location. These locations are mapped to the graph, meaning that we search for the closest nodes that reflect the coordinates of the start and target point best. Afterward, well-known algorithms for calculating the shortest paths in graphs can be applied to obtain a result. Such algorithms include Dijkstra [8], A* [12], and Contraction Hierarchies [11].

The proposed procedure is capable of taking obstacles into account and should generally be more precise than the regular line distance. In the simplest case, where all fields within the grid are of the same size, the distance can be estimated by the number of edges that need to be traversed multiplied by a constant factor. To further refine the estimation, one could also calculate the distance between the point where a field was entered and the point where it was exited. This can be achieved by weighing edges that are vertical or horizontal differently than diagonal edges.

The presented approach can easily be extended during the preprocessing phase. Instead of only considering areas that are classified as obstacles or passable, one could also define other regions impacting the edge weights. As an example, areas containing steep mountains are more likely to have serpentines which would increase the weight. Such behavior can be achieved by refining the classification in the second step.

4.2 LRA-M Loop Adaptation

The proposed approach for estimating distances under the consideration of obstacles works with a grid in which every field has the same size. This is disadvantageous when considering regions with smaller obstacles that would lead to large fields being marked as obstacles. However, using a higher resolution fixing this issue increases the size of the graph, and therefore slows down the required computational time. To tackle this problem, one could either find a resolution that fits both needs or the resolution could be

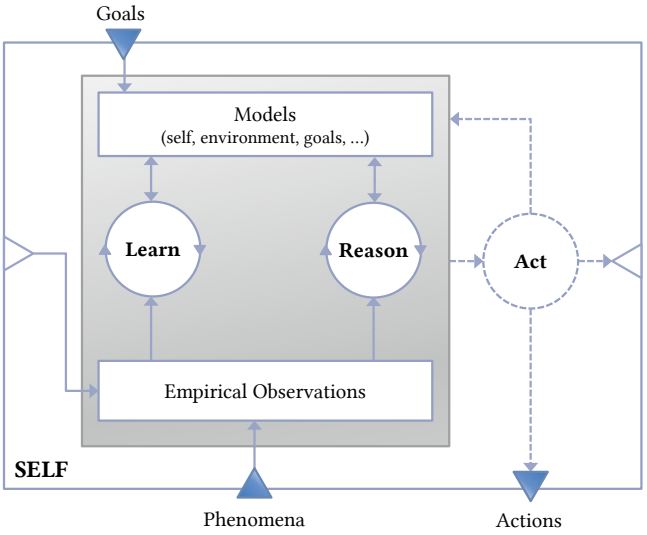

Figure 2: LRA-M Loop [17].

dynamically changed depending on the properties of the region that is represented by a field. In areas that do not contain obstacles of interest, fields could be merged so that the resulting graph comprises fewer nodes and edges. When merging fields, the edge weights of the resulting graph have to represent that merging. The simple procedure of counting the amount of edges that need to be traversed is no longer feasible in this case.

Adaptive resolutions of the graph can be achieved by integrating concepts *self-aware computing* concepts, in particular, the LRA-M loop described by Kounev et al. [17]. We map several tasks to the components of the loop displayed in Figure 2. The entire approach is considered as the *self*, including the preprocessing and the distance estimation. *Goals* can be specified by the user to define the prioritization of accuracy vs. computational time. The *Model* component keeps track of the graph that is used for the distance calculations and is regularly updated during the *Learning* phase. The *Reasoning* component calculates distances of known routes using the current model. Then, the accuracy, as well as the required computation time, are used to check if the model suits the user-specified goals. If this is not the case, the *Learning* component will update the model accordingly and learn which resolution fits best to the current area of the map as well as the given goals. This allows for faster adaptations in future executions of the approach.

Table 1: Results of the prototypical evaluation analyzing distance [m] and computation time [ms] in 30 repetitions.

Method	Distance [m]	Node Identification [ms]		Distance Calculation [ms]		Combined [ms]	
		mean	std	mean	std	mean	std
Haversine	4090	-	-	<1	-	<1	-
Manhattan	4890	-	-	<1	-	<1	-
OSM (SQL)	5146	200.567	35.145	505.633	14.771	738.2	23.057
FADE	4455	<1	-	2.233	0.183	2.333	0.3198

During the learning process in the LRA-M loop, different operations can be applied to the current model. For one, we identified a *zoom-in* operation that increases the resolution of an area. This results in more nodes and connections in the graph which also increases the accuracy but slows down the required runtime. The *zoom-out* operation inverts the zoom-in operation and reduces the resolution but accelerates the time to result. It is important that an operation is invertible to ensure that the model can always be adjusted to the specified goals.

5 PROTOTYPICAL EVALUATION

To analyze the feasibility of FADE, we prototypical implemented the general idea in Python using the network analysis package NetworkX[1]. This includes a manual extraction of an abstracted map using OpenStreetMap (OSM)[2]. Then, the prototype places a grid on the abstracted map and analyzes passability properties of the fields. Finally, the prototype receives geographic start and target coordinates, identifies the related nodes in the passability graph and executes the Dijkstra implementation of NetworkX.

In the preliminary evaluation, we extracted an abstracted map from the area around the German city Würzburg of around 6162 km^2. The original routable graph of OSM for cars contains 190,061 nodes and 211,795 edges with about 2,000 bridges and 2,000 water areas. For placing the grid, we defined a total of 178 rows and 341 columns which refers to a height of about 440 m and a width of about 230 m. The resulting graph contains 58,627 nodes and 229,213 edges which means a node reduction of about 70 % compared to the routable graph of OSM. For the evaluation, we compare the estimated distance in meters as well as the computation time in milliseconds for node identification, distance calculation, and both processes combined. Further, we compare the distance estimation to the state-of-the-art Haversine and Manhattan distance functions. We present the results of our evaluation in Table 1 which shows that Haversine distance function deviates most in terms of estimated distance compared to the ground truth calculated using pgRouting[3] on OSM, followed by FADE and the Manhattan distance. The time measurements show that the distance functions can be computed very fast and that OSM requires 200 ms, 505 ms, and 740 ms for identifying the nodes, calculating the distance, and computing both processes combined, respectively. FADE, even if in a very early prototypical state, highlights the trade-off potential as the distance estimation is better compared to the Haversine function while keeping the computation time low with zero ms for node identification, two ms for distance calculation, and two ms for the combined process.

[1]https://networkx.org
[2]https://www.openstreetmap.de
[3]https://pgrouting.org

6 DISCUSSION AND OUTLOOK

The presented approach comes with challenges that need to be solved during the implementation. First of all, different types of bridges need to be considered as some might not be built for vehicles but pedestrians or other use cases. Depending on the size of the rendered abstracted map, bridges might be too small and do not show up when applying the grid. This could result in areas being unreachable as no bridge leads in or out of the region. After the graph construction, the start and target locations need to be mapped to the graph which could result in one or both of them being within an obstacle. Finally, bridges might be passable in one direction only which needs to be considered when constructing the graph.

While the presented prototypical evaluation highlights the potential of FADE, a full-scale evaluation needs to be conducted to analyze the quality of the solutions with regards to the computation time in a statistically significant manner. Hence, the trade-off needs to be assessed by analyzing speed-up and quality metrics. Besides OSM, other distance services could be considered to further strengthen the expressive power of the results. Finally, different resolutions for the map will yield different performance results of FADE. This could be related to the current region of the planning horizon which would directly lead to the selection of different grid sizes based on the region by applying self-aware concepts.

7 CONCLUSION

The performance of optimization techniques, especially applied in logistic processes, became more and more important due to increasing transport volumes in the last years. One significant performance factor for these techniques is the exact computation and estimation of distances. In this vision paper, we propose FADE—a Flexible and Adaptive Distance Estimation—which abstracts a map to reduce the complexity of the road network graph and allow one to trade off computation time vs. result accuracy. The integration of concepts from self-aware computing allows for dynamic adaptation of given computation time and accuracy requirements. Our proposed technique can not only be applied in the proposed facility location problem but also in various logistic and general problems requiring distance information.

ACKNOWLEDGMENTS

This work was funded by Bayerisches Verbundforschungsprogramm (BayVFP) – Digitalisierung under grant No. DIK0314/02.

REFERENCES

[1] 2021. E-commerce in the time of COVID-19. https://www.oecd.org/coronavirus/policy-responses/e-commerce-in-the-time-of-covid-19-3a2b78e8. [Online; accessed 12. Jan. 2022].

[2] 2022. How e-commerce share of retail soared across the globe: A look at eight countries. https://www.mckinsey.com/featured-insights/coronavirus-leading-through-the-crisis/charting-the-path-to-the-next-normal/how-e-commerce-share-of-retail-soared-across-the-globe-a-look-at-eight-countries. [Online; accessed 12. Jan. 2022].

[3] M. Hakan Akyüz. 2018. Discretization Based Heuristics for the Capacitated Multi-Facility Weber Problem with Convex Polyhedral Barriers. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)* 8, 1 (2018), 26–42. https://doi.org/10.11121/ijocta.01.2018.00388

[4] Ethem Alpaydin, İ. Kuban Altinel, and Necati Aras. 1996. Parametric Distance Functions vs. Nonparametric Neural Networks for Estimating Road Travel Distances. *European Journal of Operational Research* 93, 2 (Sept. 1996), 230–243. https://doi.org/10.1016/0377-2217(96)00045-8

[5] Luciano Blasi, Egidio D'Amato, Massimiliano Mattei, and Immacolata Notaro. 2020. Path Planning and Real-Time Collision Avoidance Based on the Essential Visibility Graph. *Applied Sciences* 10, 16 (Jan. 2020), 5613. https://doi.org/10.3390/app10165613

[6] Jack Brimberg, John H. Walker, and Robert F. Love. 2007. Estimation of Travel Distances with the Weighted ℓp Norm: Some Empirical Results. *Journal of Transport Geography* 15, 1 (Jan. 2007), 62–72. https://doi.org/10.1016/j.jtrangeo.2006.01.004

[7] Derya Celik Turkoglu and Mujde Erol Genevois. 2020. A comparative survey of service facility location problems. *Annals of Operations Research* 292, 1 (2020), 399–468. https://doi.org/10.1007/s10479-019-03385-x

[8] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to algorithms*. MIT press.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. https://doi.org/10.1109/4235.996017

[10] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 26, 1 (1996), 29–41. https://doi.org/10.1109/3477.484436

[11] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *Experimental Algorithms*, Catherine C. McGeoch (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 319–333.

[12] Peter Hart, Nils Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107. https://doi.org/10.1109/tssc.1968.300136

[13] Kazuyoshi Hidaka and Hiroyuki Okano. 2003. An approximation algorithm for a large-scale facility location problem. *Algorithmica* 35, 3 (2003), 216–224. https://doi.org/10.1007/s00453-002-0996-z

[14] John H. Holland. 1992. Genetic Algorithms. *Scientific American* 267, 1 (1992), 66–73.

[15] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4. 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[16] Martin Kords. 2021. Transportleistung im Straßengüterverkehr 2019 | Statista. https://de.statista.com/statistik/daten/studie/2979/umfrage/entwicklung-der-transportleistung-des-strassengueterverkehrs. [Online; acc. 3. Feb. 2021].

[17] Samuel Kounev, Peter Lewis, Kirstie L Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, et al. 2017. The Notion of Self-aware Computing. In *Self-Aware Computing Systems*. Springer, 3–16.

[18] Amgad Madkour, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. 2017. A Survey of Shortest-Path Algorithms. arXiv:1705.02044 [cs.DS]

[19] Ricardo Mesquita and Pedro D Gaspar. 2020. A Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling–Simulation Results. In *2020 International Conference on Decision Aid Sciences and Application (DASA)*. IEEE, 1144–1148.

[20] Mohammad Mahdi Nasiri, Vahid Mahmoodian, Ali Rahbari, and Shabnam Farahmand. 2018. A modified genetic algorithm for the capacitated competitive facility location problem with the partial demand satisfaction. *Computers & Industrial Engineering* 124 (2018), 435–448.

[21] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. https://networkrepository.com

[22] Rizwan Shahid, Stefania Bertazzon, Merril L. Knudtson, and William A. Ghali. 2009. Comparison of Distance Measures in Spatial Analytical Modeling for Health Service Planning. *BMC Health Services Research* 9, 1 (Nov. 2009), 200. https://doi.org/10.1186/1472-6963-9-200

[23] Vedat Verter. 2011. Uncapacitated and Capacitated Facility Location Problems. In *Foundations of Location Analysis*, H. A. Eiselt and Vladimir Marianov (Eds.). Springer US, New York, NY, 25–37. https://doi.org/10.1007/978-1-4419-7572-0_2

[24] Cristian Zambrano-Vega, Génesis Acosta, Jasmin Loor, Byron Suárez, Carla Jaramillo, and Byron Oviedo. 2019. A Sales Route Optimization Mobile Application Applying a Genetic Algorithm and the Google Maps Navigation System. In *International Conference on Information Technology & Systems*. Springer, 517–527.