# Creating Graph Partitions for Fast Optimum Route Planning

INGRID FLINSENBERG, MARTIJN VAN DER HORST, JOHAN LUKKIEN
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
THE NETHERLANDS

JACQUES VERRIET
Siemens VDO Automotive
P.O. Box 8807, 5605 LV Eindhoven
THE NETHERLANDS

*Abstract:* We investigate fast optimum route planning in large, real-world road networks for car navigation systems. We show how graph partitioning can be used to increase the speed of planning optimum routes. Creating a graph partition with future route planning in mind leads to a non-standard graph partitioning problem. In particular, the quality of a partition, indicated by the objective value, is assumed to represent the execution time of the route planning process. We present an efficient approximation algorithm for creating graph partitions suited for fast optimum route planning. We study the relation between the objective value and the number of edges evaluated by the route planning algorithm, which is an objective measure of the route planning speed. Experiments show that the best partition according to the objective value does not lead to the fastest route planning process. We present a new objective value and show that better partitions result in faster route planning for our new objective value.

*Key-Words:* Graph partitioning, Approximation algorithms, Route planning, Real-world road networks, Car navigation systems.

## 1 Introduction

An increasing number of car manufacturers are offering a navigation system as one of the possible features of their cars. The key components of such a navigation system are positioning, guidance and route planning. In this paper, we focus on the route planning functionality of a car navigation system.

Users of such systems are becoming increasingly demanding: drivers do not want to wait for their route to be planned, and they expect the system to provide, for example, the absolutely fastest route. Furthermore, the size of the available road networks is increasing. CDs or DVDs containing the road network of entire Europe are now becoming available. So optimum routes have to be planned on very large road networks, in very little time.

Because of the increasing size of available road networks and higher demands on route quality and planning speed, planning optimum routes in little time is a continuing challenge for companies developing car navigation systems.

For a car navigation system, a Dijkstra-like route planning algorithm [2, 6] is not fast enough to plan optimum routes in large real-world road networks. Because of the high demands on planning speed, the route planning process has to be speeded up, which can be done by pre-processing the roadgraph. Jung and Pramanik [9] and Flinsenberg [4, 5] describe a graph partitioning approach to speed up the planning process. They divide the roadgraph into a number of disjunct subgraphs that are connected by a boundary graph.

Flinsenberg [4] showed that a road network can be divided into several disjunct parts, called *cells*, to increase the route planning speed of a car navigation system. This results in a graph partitioning problem that is different from standard graph partitioning problems in literature. We focus on the graph partitioning approach described by Flinsenberg [4, 5]. We give a graph partitioning algorithm for her approach and study the relationship between the graph partitioning problem and the efficiency of her route planning algorithm. Flinsenberg [4] aims at minimizing the number of edges in the searchgraph because this is expected to represent the speed of the route planning algorithm used for planning an optimum route. The speed of a route planning algorithm can be measured in the number of evaluated edges, see Flinsenberg [5]. So, we do not want to minimize the number of edges in the searchgraph, but the number of evaluated edges by the route planning algorithm. These numbers are typically not the same. Therefore, we study the number of evaluated edges by the route planning algorithm of Flinsenberg [5], and compare this number with the number of edges in the searchgraph. We show that the graph partition minimizing the number of edges in the

searchgraph does not lead to the fastest route planning algorithm. We then introduce a new partitioning problem for which we show that better partitions result in faster route planning.

Graph partitioning has been studied extensively, for many different applications. Berry and Goldberg [1] compare different algorithms for computing graph partitions. Falkner et al [3] study partitioning the nodes of a graph into $k$ disjoint subsets of specified sizes with the objective of minimizing the total weight of the edges connecting nodes in distinct subsets of the partition. They present a numerical study on the use of eigenvalue-based techniques to find upper and lower bounds for this problem, based on graphs of several thousands of nodes. Huang et al [8] compare alternative graph clustering solutions for storing data in square blocks that minimize the number of I/O operations. They compare spatial-partition clustering, two-way partitioning and approximate topological clustering. Krishnan et al [11] study graph partitioning for Internet-like structures. They require that the partitions are connected and show this leads to quite a different problem. Monien and Diekmann [12] study bisection techniques for minimizing the number of connections between subgraphs. Pothen [13] studies the same problem but compares several techniques. Graph partitioning is also studied by Schloegel et al [15] and Karypis and Kumar [10]. Partitioning of flat terrain into polygons is studied by Rowe and Alexander [14].

All these graph partitioning problems are fundamentally different from our graph partitioning problem. There are three important differences between the problems studied in literature so far and ours. First of all, our problem requires the partitioning of a graph into an unknown number of subgraphs, while the number of subgraphs is assumed to be known beforehand in other papers. Secondly, the objective value of a graph partition is much more complicated for our problem. Finally, the graph partitioning algorithm has to be applied to very large real-world road networks. These road networks typically contain millions of nodes and edges.

This paper is organized as follows. We introduce cell-partitions in Section 2, and the algorithm used for computing cell-partitions in Section 3. We discuss the implementation of this algorithm in Section 4. In Section 5 we discuss the comparison of the number of evaluated edges and the number of edges in the searchgraph. It turns out that the number of edges in the searchgraph is not an accurate estimation of the number of edges evaluated by the route planning algorithm. In Section 6, we study a variant of the number of edges in the searchgraph as optimization criterion for our partitioning problem. In Section 7, we show that this variant does accurately represent the number of evaluated edges by the route planning algorithm of Flinsenberg [5]. Section 8 presents the conclusions.

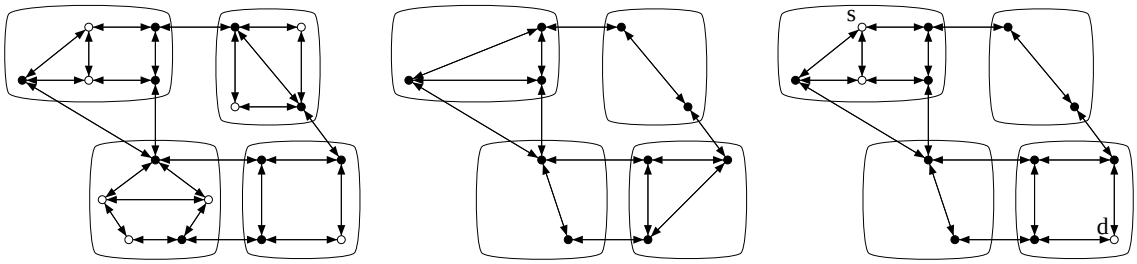
# 2 Cell-Partitions

A road network can be represented as a graph, in which the edges represent the road segments, and the nodes represent the junctions. Since there may exist parallel and circular roads, we do not exclude parallel edges or loops. Because one-way roads have to be modeled as well, every edge in a road network is directed. A road network can thus be represented by a directed multi-graph. For an edge $e$ from node $u$ to node $v$, let $\delta_1(e)$ denote start node of the edge, and $\delta_2(e)$ the end node of the edge, i.e. $\delta_1(e) = u$ and $\delta_2(e) = v$ . Formally, we define a *roadgraph* as a tuple $G = (N, E, w)$, where $N$ denotes the set of nodes, $E$ the set of edges, and $w(e)$ the non-negative cost associated with edge $e$. A *route* in a roadgraph is a sequence of adjacent edges, $p = (e_1, \ldots, e_k)$, where $e_i \in E$ and $\delta_2(e_i) = \delta_1(e_{i+1})$ for $i = 1, \ldots, k-1$. The cost of route $p$, the *route-cost*, is equal to $c(p) = \sum_{i=1}^{k} w(e_i)$. A route with minimum cost from *start node* $s$ to *destination node* $d$ is called a *minimum cost route* or an *optimum route* from $s$ to $d$.

Let $G = (N, E, w)$ be a roadgraph, then a cell-partition of $G$ is a set $\{C_1, \ldots, C_k\}$ where $C_i = (N_i, E_i, w)$ is a roadgraph induced by the node set $N_i$, such that $N_i \cap N_j = \emptyset$, for every $i \neq j$, and $\cup_{i=1}^{k} N_i = N$. See for example the cell-partiton in Figure 1(a). A cell is represented by the graph contained in a block. The edges of a cell $C$ are called *internal edges*, the nodes that only have adjacent nodes in $C$ are called *internal nodes*, and nodes that also have adjacent nodes outside cell $C$ are called *boundary nodes*. Note that in Figure 1(a), the boundary nodes are black and the internal nodes are white. The *boundary graph* $B$ of a cell-partition $\{C_1, \ldots, C_k\}$ of graph $G$ consists of all edges connecting different cells.

After the roadgraph has been partitioned into a number of cells, Flinsenberg [4] computes the optimum route cost between every pair of boundary nodes of a single cell. The optimum routes are represented by edges that are added to the boundary graph. These edges are called *route edges*. The boundary graph including all route edges of the cell-partition in Figure 1(a) is given in Figure 1(b). Specifically, Flinsenberg [4] adds two directed edges between every pair of boundary nodes of a single cell. The set of route edges of a single cell thus forms a directed clique. So $n(n-1)$ route edges are created for a cell with $n$ boundary nodes. The *searchgraph*, denoted by $G_S$, consists of the cells containing the start and destination node, all boundary edges and the route edges of all cells, except the cells containing the start and destination node. The searchgraph of the cell-partition in Figure 1(a) for start node $s$ and destination node $d$ is given in Figure 1(c).

Using the notation in Table 1, Flinsenberg [4] introduces the partitioning problem (1), which aims at minimizing the average size of the searchgraph.




(a) *A cell-partition.*

(b) *Boundary graph and route edges.*

(c) *Searchgraph for start node s and destination d.*

Figure 1: *Creating a searchgraph.*

| | |
|---|---|
| $k$ | Number of cells in cell-partition $\{C_1,\ldots,C_k\}$. |
| $n$ | Number of nodes of roadgraph $G$. |
| $n_i$ | Number of nodes in cell $C_i$. |
| $m_i$ | Number of edges in cell $C_i$. |
| $r_i$ | Number of route edges of cell $C_i$. |
| $b_i$ | Number of boundary nodes of cell $C_i$. |
| $m_B$ | Number of boundary edges in the partition. |

Table 1: *Used notation.*

$$\min \sum_{i=1}^{k} \left\{ \frac{n_i}{n}(2-\frac{n_i}{n})m_i + (1-\frac{n_i}{n})^2 r_i \right\} + m_B \qquad (1)$$

Because a route edge is created between each pair of boundary nodes, we have $r_i = b_i(b_i-1)$. Flinsenberg [5] shows that turn-restrictions can be used to further increase the route planning speed by storing the optimum route cost between every pair of boundary nodes in only $2b_i$ route edges for a cell $C_i$ with $b_i$ boundary nodes, instead of creating $b_i(b_i-1)$ route edges. As a result, the size of the searchgraph that is used for planning the optimum route changes. Therefore, also the optimization problem for creating a cell-partition changes. Specifically, we have $r_i = 2b_i$ instead of $r_i = b_i(b_i-1)$. We call the value of (1) for a particular cell-partition, the *objective value.* The partitioning problems thus consist of finding a cell-partition with minimum objective value.

## 3 Merging-Algorithm

The cell-partitioning problem (1) with $r_i = b_i(b_i-1)$ is *NP*-hard, see Flinsenberg [4]. Similarly it can be shown that the cell-partitioning problem is also *NP*-hard for $r_i = 2b_i$. Therefore we develop an approximation algorithm to solve these problems. Unlike other partitioning problems in literature, the number of cells in which the graph has to be partitioned is unknown. Therefore, the approximation algorithm also has to determine the number of cells of the cell-partition.

Our partitioning algorithm, called the Merging-Algorithm, is a greedy algorithm that starts by creating $n$ cells that each consist of a single node. Then in each step, it repeatedly selects two cells and merges these two cells into a new cell. This process continues until only a single cell remains, containing the entire graph. This process is called a *run.* During a single run the best found cell-partition is stored. We have randomized the selection of the two cells that are merged together in each step, thereby covering a different part of the state space with each run. To find a good partition the Merging-Algorithm consists of several runs.

In order to create good cell-partitions we choose the two cells to merge according to priority function $\rho$. A good cell-partition typically contains only a few boundary nodes per cell. We also like to keep the cells roughly equal in size, so that the partition does not contain one very large cell and a lot of very small cells. This is convenient for a car navigation system because it means there is not too much difference in handling different cells.

Define the *priority* of merging cells $C_i$ and $C_j$ by $\rho(i,j)$. We use:

$$\rho(i,j) = \frac{m_B(i,j)(1+b_i+b_j-b_{ij})}{n_i n_j} random(1,1.01),$$

with $m_B(i,j)$ the number of boundary edges between cell $C_i$ and $C_j$, $b_{ij}$ the number of boundary nodes of the merged cell, and $random(\alpha,\beta)$ a random (real) number between $\alpha$ and $\beta$. This priority function is based on the observation that cells that have many edges between them should be merged. Furthermore, if we can choose between merging two small cells with many boundary edges between them and two large cells, we should merge the two small cells, because relatively they have the most connecting boundary edges. So, we divide by the number of nodes in each cell to favor small cells. As a result, we are also more likely to end up with cells of approximately equal sizes. Furthermore, we multiply by the reduction in the number of

boundary nodes (plus 1) to favor cell-partitions with only a few boundary nodes. Note that the number of boundary nodes of the merged cell can never be larger than the number of boundary nodes of cells $C_i$ and $C_j$ together. So the priority value cannot be negative. By adding 1, we prevent the priority value from becoming zero (so $\rho(i,j) > 0$ for all cells $C_i$ and $C_j$). The multiplication with a random number is done to randomize the algorithm and achieve an efficient implementation.

## 4 Implementation

To make a practical analysis of the objective value, we implemented the Merging-Algorithm using a few standard data structures and techniques, like: linked lists, double linked lists, priority queues, reference counts and calculating the change of the objective value after each merge, instead of calculating it anew each time. The details of the implementation are beyond the scope of this article, a detailed description can be found in van der Horst's master's thesis [7].

It is possible to implement the Merging-Algorithm with a worst-case running time of $O(e + n \times M \times (B + \log e))$, where $e$ denotes the number of edges in the roadgraph, $M$ the maximum number of neighboring cells a cell has during a run and $B$ the maximum number of boundary nodes a cell has during a run. There are graphs for which $M$ and $B$ equal $n-1$ and $n$ respectively, but when partitioning road networks, $M$ and $B$ are generally small compared to $n$.

To get an indication of the running time, we used the algorithm to partition the map of the Netherlands. This map consists of roughly 800,000 nodes and 1,100,000 edges. A dual Pentium III 1.4 Ghz with 2 Gb of working memory was used to run the program and partition this graph. The program required 4 minutes of processing time and 400Mb of working memory to perform 10 runs and find a partition containing 524 cells with an objective value (1) of 40,163 using $r_i = 2b_i$. This partition is shown in Figure 2.

## 5 Validation of the Objective Value

The objective value of a partition indicates the expected number of edges in the searchgraph. The goal of partitioning a graph into cells however, is to speed up route planning. The speed of a route planning algorithm can be measured in the number of evaluated edges (Flinsenberg [5]), also called *expansions.* Therefore the actual quality of a partition is indicated by the average number of expansions required to plan a route.

The number of edges in the searchgraph is clearly an upper bound for the number of expansions to plan a route. So the objective value does say something about the quality of a partition, but it is unclear whether it actually corresponds to the quality.

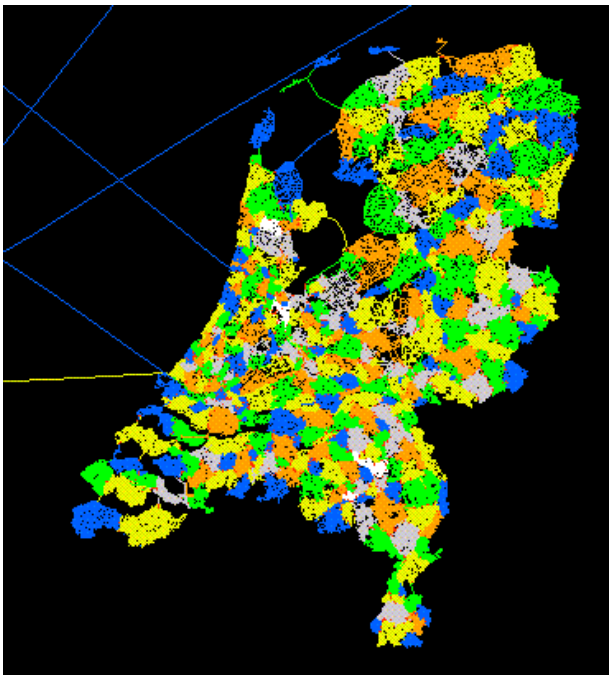


Figure 2: *Cell-partition of the Netherlands.*

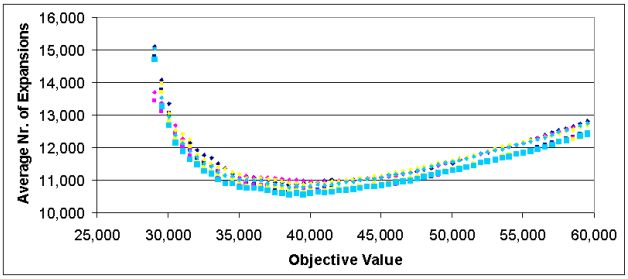



Figure 3: *Relationship between the objective value and the number of expansions.*

The relationship between the quality and the objective value is examined by planning routes on the road network of the Netherlands. First, the Merging-Algorithm is used to generate multiple partitions. This is done by creating categories represented by the values 40,000, 40,500, 41,000, ..., 60,000. The first partition encountered during a run that fits a category is saved. A partition is said to fit a category if the difference between its objective value and the representative value of the category is less than 50.

Subsequently, the $A^\star$ algorithm [6] is used to plan 2,000 routes through each of the generated partitions. The average number of expansions is plotted in Figure 3. The figure contains the plots of partitions generated during five runs. Each run is represented by a different color. The diamonds indicate the average number of expansions for planning fastest routes, (i.e. routes minimizing the driving time), while the squares indicate the average number of expansions for planning shortest routes (i.e. routes minimizing the driving distance).

Figure 3 clearly shows that the minimum average number of expansions does not coincide with the minimum objective value. The best partition according to the objective value has a value of approximately 40,500 and requires on average circa 16,500 expansions to plan a route. But clearly there is a better partition with an objective value of about 45,000 that requires only 14,500 expansions. The objective value is therefore not an appropriate measure of the quality of a partition.

## 6 Weighted Objective Value

The partition with the higher objective value from Figure 3 requires fewer expansions on average to plan an optimum route because it has smaller cells. When the $A^\star$-algorithm starts planning in the start cell it encounters edges with roughly the same cost. But as soon as the algorithm reaches the edge of the cell it suddenly encounters the route edges, which have a much higher cost than the edges inside the cell. These high costs combined with a comparetively low decrease in the heuristic estimator generally cause the expected cost of a route via route edges to be higher than the expected cost of a route via an internal edge of the start cell. Therefore the $A^\star$ algorithm continues examining edges in the start cell, often until there are no edges left to be examined. This causes, on average, 60% of all expansions to occur inside the start cell.

Because the partition with the higher objective value has smaller cells, the $A^\star$ algorithm is able to leave the start cell faster than in the partition with the "optimum" value. This can be corrected by assigning a lower weight to parts of the objective value function that do not indicate the size of the cells. We call this new function the *weighted objective value*:

$$\min \sum_{i=1}^{k} \left\{ \frac{n_i}{n}(2-\frac{n_i}{n})m_i + \alpha(1-\frac{n_i}{n})^2 r_i \right\} + \alpha m_B. \quad (2)$$

Note that for $\alpha = 1$ the weighted objective value (or w-value for short) is the same as the objective value (1). $\alpha$ indicates the weight of the boundary graph and route edges compared to the size of the cells.

Since the route planning algorithm frequently examines a large portion of the edges in the start or destination cell the $\alpha$ should indicate the fraction of the boundary graph and route edges that are evaluated by the algorithm. This fraction can be estimated by $\alpha = \frac{SEA}{SM}$, where $SEA$ indicates the surface area of the expected search area and $SM$ the surface area of the entire map.

The expected search area ($SEA$) can be seen as an ellipse with a distance between the foci equal to the average route length, and the sum of the distances from any point on the ellipse to the foci equal to the average route length multiplied by a detour factor. This leads to $SEA = \frac{1}{4} \times \pi \times ARL^2 \times c \times \sqrt{c^2-1}$ with $ARL$ the average route length and $c$ the detour factor.

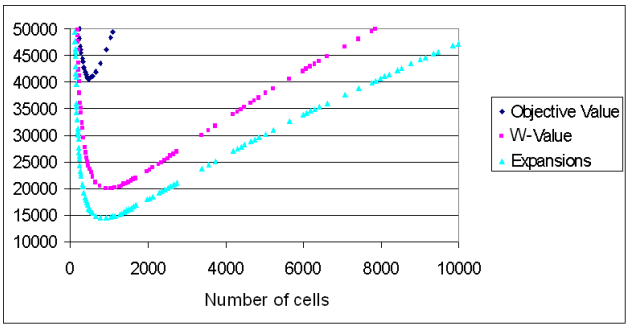



Figure 4: *Objective value, w-value and the average number of expansions*

## 7 Validation of the W-Value

To validate the w-value a new experiment is conducted. Sample partitions are generated, and routes are planned on those sample partitions. To get a better idea of the way the objective value, w-value and the number of expansions behave, samples were generated from an entire run.

The value of $\alpha$ is calculated as described in the previous section. The 2,000 routes used for planning have an average length of 122 km. The detour factor is set to 1.3, based on the experience of Siemens VDO. This results in $SEA = 12,623$ square kilometers. The surface area of the Netherlands is 37,938 square kilometers, so $\alpha \approx 0.33$.

Figure 4 shows a part of the samples and the objective value, w-value and number of expansions belonging with each sample. When projecting the minimum of all three series to the horizontal axis, those of the w-value and the average number of expansions are clearly the closest together.

Experimenting with the samples showed that for $0.28 < \alpha < 0.40$ the w-value leads the algorithm to select the optimal partition from among the samples, so the $\alpha \approx 0.33$ is a good choice. This does, of course, not mean that the w-value actually results in the algorithm selecting the best partition encountered during the run. To examine this finer sampling might be necessary.

However, the weighted objective value clearly results in a partition of better quality than the unweighted objective value. Even if the partition with the minimum number of expansions and the partition with the minimum w-value are not exactly the same, there is still little difference between average number of expansions required to plan a route. This can be seen in Figure 4, because the partitions close to the optimum have a similar number of expansions as the optimum itself. So we can conclude that partitions with a lower w-value require less expansions.

## 8 Conclusions

We have presented the Merging-Algorithm, which is a polynomial-time approximation algorithm for creating cell-partitions that allow fast optimum route planning. The Merging-Algorithm is capable of partitioning a real-world road network with more than a million edges (in an unknown number of cells), in just a few minutes. We have also shown that the best cell-partition according to the original cell-partitioning problem of Flinsenberg [4] does not give the fastest route planning results. Therefore, we developed a new cell-partitioning criterion for which we have shown that a partition that is better according to this criterion, also leads to faster route planning. This new criterion is based on information on characteristics of the road network and the route planning algorithm. The Merging-Algorithm can again be used to create a cell-partition according to our new criterion. We have shown that this leads to significantly faster route planning.

For our new criterion, better partitions result in faster route planning. Therefore, increasing the quality of the found partitions leads to faster route planning. The Merging-Algorithm is an approximation algorithm with unknown accuracy. To establish its accuracy, the value of optimum cell-partitions is an interesting subject for futher research. Different approximation algorithms can be studied as well to determine the quality of the results. The introduction of multi-level partitioning could further increase the route planning speed. We intend to study the influence of the introduction of multiple levels to our partitioning problem. Also different route planning algorithms may influence not only the route planning time, but also the relation between our new partitioning criterion and the number of edges evaluated by the route planning algorithm. Currently, we are introducing time-dependent route planning into our model and working on its integration with cell-partitions.

*References:*


[1] Jonathan Berry and Mark Goldberg. Path optimization and near-greedy analysis for graph partinioning: An empirical study. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, January 1995.

[2] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[3] Julie Falkner, Franz Rendl, and Henry Wolkowicz. A computational study of graph partitioning. Technical Report CORR-92-25, University of Waterloo, Canada, August 1992.

[4] Ingrid Flinsenberg. Graph partitioning for route planning in car navigation systems. In *Proceedings of the 11'th IAIN World Congress, Smart Navigation - Systems and Services* -, Berlin, Germany, October 2003.

[5] Ingrid Flinsenberg. Using turn restrictions for faster route planning with partitioned road networks. In *Proceedings of the 10th Saint Petersburg International Conference on Integrated Navigation Systems*, pages 198–206, Saint Petersburg, Russia, May 2003.

[6] G. Galperin. On the optimality of A$^\star$. *Artificial Intelligence*, 8(1):69–76, 1977.

[7] M.G. van der Horst. Optimal route planning for car navigation systems. Master's thesis, Technische Universiteit Eindhoven, October 2003.

[8] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. *Optimizing Path Query Performance: Graph Clustering Strategies*, volume 1. Elsevier Science (Pergamon), 2000.

[9] Sungwon Jung and Sakti Pramanik. An efficient path computation model for hierarchically structured topological road maps. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1029–1046, September/October 2002.

[10] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[11] Rajesh Krishnan, Ram Ramanathan, and Martha Steenstrup. Optimization algorithms for large self-structuring networks. In *INFOCOM : The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*, volume 1, pages 71–78, March 1999.

[12] Burkhard Monien and Ralf Diekmann. A local graph partitioning heuristic meeting bisection bounds. In *8th SIAM Conference on Parallel Processing for Scientific Computing*, March 1997.

[13] Alex Pothen. Graph partitioning algorithms with applications to scientific computing. In D.E. Keyes et al, editor, *Parallel Numerical Algorithms*, pages 323–368. Kluwer Academic Publishers, Netherlands, 1997.

[14] Neil C. Rowe and Robert S. Alexander. Finding optimal-path maps for path planning across weighted regions. *International Journal of Robotics Research*, 19(2):83–95, February 2000.

[15] Kirk Schloegel, George Karypis, and Vipin Kumar. *Graph Partitioning for High Performance Scientific Simulations*, pages 491–541. Morgan Kaufmann Publishers Inc., 2003.