Online Multi-Path Routing in a Maze*

Stefan Rührup[1] and Christian Schindelhauer[2]

[1] Heinz Nixdorf Institute, University of Paderborn, Germany
sr@uni-paderborn.de
[2] Computer Networks and Telematics, University of Freiburg, Germany
schindel@informatik.uni-freiburg.de

Abstract. We consider the problem of route discovery in a mesh network with faulty nodes. The number and the positions of the faulty nodes are unknown. It is known that a flooding strategy like expanding ring search can route a message linear in the minimum number of steps d while it causes a traffic (i.e. the total number of messages) of $\mathcal{O}(d^2)$. For optimizing traffic a single-path strategy is optimal producing traffic $\mathcal{O}(d+p)$, where p is the number of nodes that are adjacent to faulty nodes. We present a deterministic multi-path online routing algorithm that delivers a message within $\mathcal{O}(d)$ time steps causing traffic $\mathcal{O}(d + p \log^2 d)$. This algorithm is asymptotically as fast as flooding and nearly traffic-optimal up to a polylogarithmic factor.

1 Introduction and Overview

Sending a message is the most fundamental feature of communication networks. We consider two-dimensional mesh networks, which can be found in parallel computers, in integrated circuits, FPGAs (Field Programmable Gate Arrays) and also some kinds of wireless sensor networks. In all these networks nodes may fail or may be unavailable. A node's failure can only be noticed by its neighbors. A straight-forward approach is to regularly test the neighbors of each node, to collect this data and to distribute a map of all failed and working nodes throughout the network. We investigate scenarios where this knowledge is not available when the message is on its way. Due to the lack of global information this routing problem states an online problem.

The basic problem is that the faulty nodes are barriers to the routing algorithm and that the algorithm does not know these barriers. There is no restriction on the size and the shape of the barriers, so even labyrinths are possible. In such situation a fast message delivery can only be guaranteed by flooding the complete network, which results in a tremendous increase of traffic, i.e. the number of node-to-node transmissions. If the algorithm uses a single-path strategy, then the additional effort necessary for searching a path to the destination increases the time.

* Partially supported by the DFG Sonderforschungsbereich 376 and by the EU within 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS).

We analyze algorithms with respect to the length of the shortest path d between source and target and with respect to the number of border nodes p, which are the nodes adjacent to faulty nodes. Regarding the time, no single-path online algorithm can beat the optimal offline algorithm and in worst case scenarios it has to investigate all the barriers, i.e. a traffic proportional to the number of border nodes p is inevitable. There are single-path algorithms that use only $\mathcal{O}(d+p)$ messages in total, but they need $\mathcal{O}(d+p)$ time steps. Time-optimal algorithms are parallel multi-path algorithms (e.g. expanding ring search) with time $\mathcal{O}(d)$ and traffic $\mathcal{O}(d^2)$ in the worst case.

We are interested in optimizing time and traffic at the same time. One might expect a trade-off situation between these measures. However, our research shows that there are algorithms that approximate the offline time bound and the optimal online traffic bound by a factor of $\mathcal{O}(\sqrt{d})$ [21] at the same time. The quotient comparing to the offline time bound is called the competitive time ratio, while the quotient comparing to the traffic bound of the optimal online algorithm is called the comparative traffic ratio. Subsequent work showed that both bounds could be improved below any polynomial bound to a term of $d^{\mathcal{O}\left(\sqrt{\frac{\log\log d}{\log d}}\right)}$ [22]. We call a bound on both ratios the combined comparative ratio $\mathcal{R}_c$ (Def. 5).

Strategy	Time	Traffic	$\mathcal{R}_c$
Exp. Ring Search [9, 18]	$\mathcal{O}(d)$	$\mathcal{O}(d^2)$	$\mathcal{O}(d)$
Lucas' Algorithm [13]	$\mathcal{O}(d+p)$	$\mathcal{O}(d+p)$	$\mathcal{O}(d)$
Alternating Strategy [21]	$\mathcal{O}(d^{3/2})$	$\mathcal{O}(\min\{d^2, d^{3/2}+p\})$	$\mathcal{O}(\sqrt{d})$
Selective Flooding [22]	$d \cdot 2^{\mathcal{O}\left(\sqrt{\frac{\log d}{\log\log d}}\right)}$	$\mathcal{O}(d)+p\,d^{\mathcal{O}\left(\sqrt{\frac{\log\log d}{\log d}}\right)}$	$d^{\mathcal{O}\left(\sqrt{\frac{\log\log d}{\log d}}\right)}$
JITE (this paper)	$\mathcal{O}(d)$	$\mathcal{O}((d+p)\log^2 d)$	$\mathcal{O}(\log^2 d)$
Online Lower Bound (cf. [3])	$\Omega(d)$	$\Omega(d+p)$	$\Omega(1)$

In this paper we achieve a break-through in this line of research showing a ratio of $\mathcal{O}(\log^2 d)$. More specifically we present a deterministic algorithm that delivers the message on a multi-path route within time $\mathcal{O}(d)$ and with traffic $\mathcal{O}(d+p\log^2 d)$. This shows, that one can route a message asymptotically as fast as flooding while increasing the traffic by a factor of only $\mathcal{O}(\log^2 d)$ compared to the traffic-optimal online algorithm.

This paper is organized as follows. We continue this section by presenting related research. In the following section we describe the basic definitions and techniques more formally. In Section 3, we present the algorithm starting with an overview and the description of its components. In Section 4, we sketch the time and traffic analysis which concludes the paper.

1.1 Related Work

The problem studied in this paper has a strong relation to online search and navigation problems. These problems have been investigated in different research communities, which Angluin et al. [1] called "the online competitive analysis community" and the "theoretical robotics community". The fundamental goal

in online searching is to find a point in an unknown environment. In theoretical robotics the scenarios contain obstacles of arbitrary shape and the performance of algorithms is expressed by comparing the distance traveled by the robot to the sum of the perimeters of the obstacles [15, 1] (see also [2] for a survey and [14, 19] for an overview of path-planning and maze traversal algorithms). The competitive analysis community has studied various kinds of scenarios with restrictions on the obstacles (e.g. quadratic or convex obstacles). The performance is expressed by the *competitive ratio*, which is the ratio of the distance traveled by the robot and the length of the shortest obstacle-free path [17, 3].

Our model connects these two lines of research. Scenarios considered in online navigation with a lower bound on distance between s and t and with finite obstacle perimeters can be modeled by a faulty mesh network. We also investigate the problem of finding a path to a given point in an unknown environment, but here, the search can also be done in parallel. For robot navigation problems it is not clear how unbounded parallelism can be modeled in a reasonable way. Usually, navigation strategies are only considered for a constant number of robots. The model of a mesh network with faulty parts enables us to study the impact of parallelism on the time needed for finding the target. For the time analysis we use the competitive ratio as used by the competitive analysis community. Traffic is compared to the perimeters of the barriers which gives the *comparative traffic ratio*. This ratio expresses the amount of parallelism used by the algorithm.

Routing in faulty networks has also been considered as an offline problem. In the field of parallel computing the fault-tolerance of networks is studied, e.g. by Cole et al. [8]. The problem is to construct a routing scheme that emulates the original network. Zakrevski and Karpovski [26] investigate the routing problem for two-dimensional meshes. The model is similar to ours as they consider two-dimensional meshes under the store-and-forward model. Their algorithm needs an offline pre-routing stage, in which fault-free rectangular clusters are identified. Routing algorithms for two-dimensional meshes, that need no pre-routing stage are presented by Wu [24]. These algorithms use only local information, but the faulty regions in the mesh are assumed to be be rectangular blocks. In [25] Wu and Jiang present a distributed algorithm that constructs convex polygons from arbitrary fault regions by excluding nodes from the routing process. This is advantageous in the wormhole routing model, because it helps to reduce the number of virtual channels. We will not deal with virtual channels and deadlock-freedom as we consider the store-and-forward model.

Bose and Morin [6, 5] study the online routing problem for triangulations and plane graphs with certain properties and present constant-competitive algorithms for routing in these graphs. In triangulations, where no local minima exist, routing can be done by a greedy strategy. Such strategies are also used for position-based routing. Position-based routing is a reactive routing used in wireless networks, where the nodes are equipped with a positioning system, such that a message can be forwarded in the direction of the target (see [16] for a survey). Due to the limited range of the radio transceivers, there are local minima and messages have to be routed around void regions (an analog to the fault

regions in the mesh network). There are various single-path strategies, e.g. [11, 7, 12]. Position-based routing strategies have been mainly analyzed in a worst case setting, i.e. the void regions have been constructed such that the connections form a labyrinth. In this case the traffic-efficient single-path strategies produce as much traffic as flooding. In our analysis we take the perimeters of fault regions into account, so that we can express performance beyond the worst case point of view.

2 Basic Definitions and Techniques

A two-dimensional mesh network with faulty nodes is defined by a set of nodes $V \subseteq \mathbb{N} \times \mathbb{N}$ and a set of edges $E := \{(v, w) : v, w \in V \wedge |v_x - w_x| + |v_y - w_y| = 1\}$. There is no restriction on the size of the network, because time and traffic are analyzed with respect to the position of the given start node s and target node t. We assume a synchronized communication: Each message transmission to a neighboring node takes one *time step*. Furthermore, we assume the messages to be transported in a store-and-forward fashion and that the nodes do not fail while a message is being transported. However, there is no global knowledge about faulty nodes. Only adjacent nodes can determine whether a node is faulty.

Barriers, Borders and Traversals The network contains *active* (functioning) and *faulty* nodes. Faulty nodes neither participate in communication nor can they store information. Faulty nodes which are orthogonally or diagonally neighboring form a *barrier*. A barrier consists only of faulty nodes and is not connected to or overlapping with other barriers. Active nodes adjacent to faulty nodes are called *border nodes*. All the nodes in the neighborhood (orthogonally or diagonally) of a barrier B form the *perimeter* of B. A path around a barrier in (counter-)clockwise order is called a *right-hand (left-hand) traversal path*, if every border node is visited and only nodes in the perimeter of B are used. The *perimeter size* $p(B)$ of a barrier B is the number of directed edges of the traversal path. The *total perimeter size* is $p := \sum_{i \in \mathbb{N}} p(B_i)$. The perimeter size is the number of steps required to send a message from a border node around the barrier and back to the origin, whereby each border node of the barrier is visited. It reflects the time consumption of finding a detour around the barrier.

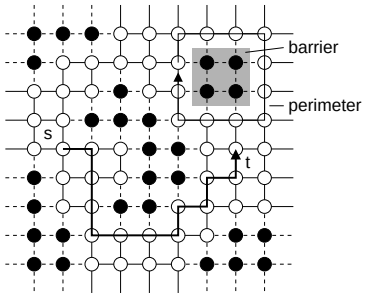

Fig. 1. Mesh network with faulty nodes (black), routing path and right-hand traversal path

The Competitive Time Ratio Time is the number of steps needed by the algorithm to deliver a message and equivalent to the length of a path a message takes. Comparing the time of the algorithm with the optimal time leads to the competitive ratio, which is well known in the field of online algorithms [4].

Definition 1. *An algorithm A has a* competitive ratio *of c, if $\forall x \in \mathcal{I} : C_A(x) \leq c \cdot C_{\text{opt}}(x)$, where $\mathcal{I}$ is the set of all instances of the problem, $C_A(x)$ the cost of algorithm A on input x and $C_{\text{opt}}(x)$ the cost of an optimal offline algorithm.*

We compare the time of the algorithm with the length d of the shortest path to the target. Note, that the shortest path uses only non-faulty nodes.

Definition 2. *Let d be the length of the shortest barrier-free path between source and target. A routing algorithm has* competitive time ratio $\mathcal{R}_t := T/d$ *if the message delivery is performed in T steps.*

The Comparative Traffic Ratio Traffic is the number of messages an algorithm needs. A comparison with the traffic of the best offline algorithm would be unfair, because no online algorithm can reach this bound. Therefore, we define a *comparative ratio* based on a *class* of instances of the problem, which is a modification of the definition given by Koutsoupias and Papadimitriou [10]:

Definition 3. *An algorithm A has a* comparative ratio $f(P)$, *if*

$$\forall p_1 \dots p_n \in P : \max_{x \in \mathcal{I}_P} C_A(x) \leq f(P) \cdot \min_{B \in \mathcal{B}} \max_{x \in \mathcal{I}_P} C_B(x),$$

where $\mathcal{I}_P$ is the set of instances which can be described by the parameter set P, $C_A(x)$ the cost of algorithm A and $C_B(x)$ the cost of an algorithm B from the class of online algorithms $\mathcal{B}$.

With this definition we address the difficulty that is caused by a certain class of scenarios that can be described in terms of the two parameters d and p. For any such instance the online traffic bound is $\min_{B \in \mathcal{B}} \max_{x \in \mathcal{I}_{\{d,p\}}} C_B(x) = \Theta(d+p)$. Note, that for any scenario one can find an optimal offline algorithm: $\max_{x \in \mathcal{I}_{\{d,p\}}} \min_{B \in \mathcal{B}} C_B(x) = d$. This requires the modification of the comparative ratio in [10] in order to obtain a fair measure. So, we use the online lower bound for traffic to define the *comparative traffic ratio*.

Definition 4. *Let d be the length of the shortest barrier-free path between source and target and p the total perimeter size. A routing algorithm has* comparative traffic ratio $\mathcal{R}_{Tr} := M/(d+p)$ *if the algorithm needs altogether M messages.*

The combined comparative ratio addresses time efficiency *and* traffic efficiency:

Definition 5. *The* combined comparative ratio *is the maximum of the competitive time ratio and the comparative traffic ratio: $\mathcal{R}_c := \max\{\mathcal{R}_t, \mathcal{R}_{Tr}\}$*

Basic Strategies
Lucas' algorithm [13] is a simple single-path strategy that works as follows: (1.) Follow the straight line connecting source and target node. (2.) If a barrier is in the way, then traverse the barrier, remember all points where the straight line is crossed, and resume step 1 at that crossing point that is nearest to the target. This algorithm needs at most $d + \frac{3}{2}p$ steps, where d is the length of the shortest barrier-free path and p the total perimeter size. This is an optimal single-path strategy [14], which matches the asymtotical lower bound for traffic.

Expanding ring search is a straightforward multi-path strategy [9, 18], which is nothing more than to start flooding with a restricted search depth and repeat flooding while doubling the search depth until the destination is reached. This strategy is asymptotically time-optimal, but it causes a traffic of $\mathcal{O}(d^2)$, regardless of the presence of faulty nodes.

We modify the previous strategy as follows: The source starts flooding without a depth restriction, but with a delay of $\sigma > 1$ time steps for each hop. If the target is reached, a notification message is sent back to the source. Then the source starts flooding a second time, and this second wave, which is not slowed down, is sent out to stop the first wave. This *continuous ring search* needs time $\sigma \cdot d$ and causes a traffic of $\mathcal{O}\left((\frac{\sigma+1}{\sigma-1}\, d)^2\right)$ (see [20, 23] for a proof). The asymptotic performance is no improvement to expanding ring search, but an area is flooded at most two times, whereas expanding ring search visits some areas $\mathcal{O}(\log d)$ times. We use this advantage for our algorithm.

3 The JITE Algorithm

The Just-In-Time Exploration (JITE) algorithm consists of two parts: *Slow Search* and *Fast Exploration*. Slow Search is a modified breadth-first search (BFS) algorithm which uses (in contrast to flooding) a path system that is generated just-in-time by Fast Exploration. This path system is similar to a quadtree-style grid subdivision and consists of the borders of quadratic subnetworks, called *frames*, and perimeters of barriers. Due to space limitations, we present the basic ideas in the following and refer to [20, 23] for a detailed description.

The algorithm starts with a search area consisting of four connected frames with s lying on the common corner (see Fig. 2). These frames are examined by Fast Exploration: Messages are sent on a *traversal path*[3] along the frame and—if the frame is intersected by a barrier—along the perimeter of the barrier (see Fig. 3). If the traversal needs too much time because of a detour caused by barriers, then the frame is subdivided. The recursive subdivision stops when a partition of a frame is *simple*, i.e. a traversal path contains only a bounded number of border nodes. This way the path system becomes denser in the proximity of barriers. Slow Search uses this path system and propagates a slowly proceeding *shoreline* (i.e. the leaves of the BFS tree) through the network. The path system is not constructed completely when Slow Search starts. The shoreline triggers the exploration of new squares in its proximity, so that the traffic-producing exploration is restricted to areas which are visited by the Slow Search.

[3] A traversal uses the well-known *right-hand rule*: By keeping the right hand always in touch of the wall, one will find the way out of the maze.

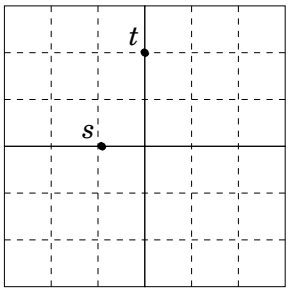

Fig. 2. Initial frames (solid) and extended search area.

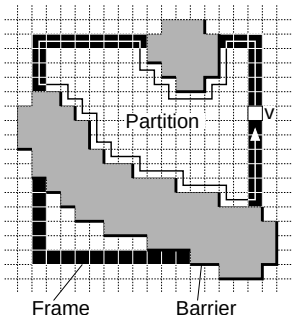

Fig. 3. Partition of a frame, defined by a right-hand traversal path

The search area is successively enlarged until t can be reached[4]. For the expansion of the search area we use the idea of continuous ring search: When the target is found, the source is notified, which sends out messages to stop the search.

As the exploration takes time, we have to slow down the shoreline, so that there is always enough time for the exploration. However, to achieve a constant slow down, the size of the neighboring squares has to be restricted, which is expressed by the *Subdivision Rule*: A simple partition of a $3g \times 3g$ frame is subdivided, (1.) if there is an orthogonally neighboring frame of size $g \times g$ or (2.) if there is diagonally neighboring frame of size $\frac{g}{3} \times \frac{g}{3}$.

Furthermore, the shoreline may enter a frame from different sides, so that a coordination mechanism is necessary in order to avoid that the same frame is explored several times. We solve this by a schedule of traversal messages that count border nodes and coordinate the concurrent exploration. This Frame Exploration Algorithm is described in detail in [20, 23].

4 Time and Traffic Analysis

In this section we present proof sketches for the time and traffic analysis. Proofs can be found in [20, 23].

4.1 Time

The constant competitive time ratio can be achieved because Fast Exploration constructs a path system that contains a constant-factor approximation of the

[4] We assume, that t also lies on a frame, i.e. $||s-t||_\infty = 3^k$ for some $k \in \mathbb{N}$. If this is not the case, then we search for any node s' with $||s'-t||_\infty = 3^k$ with the same algorithm and restart the algorithm from s'. This increases time and traffic only by a constant factor.

shortest path tree. Slow Search performs a breadth-first search on this path system which is delayed only by a constant factor. The allowed detours do not affect the linear time behaviour because of the following reason. The criterion for a simple partition in a $g \times g$-frame is that at least $4(g - \frac{g}{\gamma(t)})$ of the frame nodes is accessible and that there are at most $g/\gamma(t)$ border nodes. $\gamma(t)$ is a function of the time t when the exploration of a frame starts. We choose $\gamma(t) := \log(t)$, i.e. the allowed detours are bigger in the beginning. Summing up the allowed detours in the squares of a recursive subdivision (with frame side lengths ranging from 1 to $\log(d)$) would result in logarithmic time overhead. But this holds only for a fraction of $1/\log(d)$ of the frames. For most of the frames $g/\gamma(t)$ is bound by $\mathcal{O}(g/\log d)$. With these observations we can prove the following theorem:

Theorem 1. *Let P be the shortest path with length $|P|$ connecting s and t. The algorithm finds a path P' connecting s and t of length $\mathcal{O}(|P|)$.*

The shoreline (BFS) is slowed down by a constant factor σ and uses frames that provide a constant factor approximation of the shortest path. The exploration of new frames can be always performed in time because the Subdivision Rule guarantees that neighboring frames differ only by a constant factor in the side length.

Corollary 1. *Let d be the length of the shortest path connecting s and t. The algorithm finds a path connecting s and t in time $\mathcal{O}(d)$.*

4.2 Traffic

The traffic depends on the size of the path system, i.e. the number and size of the frames that are explored and subdivided by the algorithm and that constitute the search area. A quadtree-style subdivision enclosing barriers with a total perimeter size p has a size of $\mathcal{O}(p \log d)$. The traffic caused by the exploration of each frame is determined by the size of the frame and an additional detour, which is allowed in simple partitions. This detour adds an additional logarithmic factor.

We distinguish between *barrier-induced subdivisions* and *neighbor-induced subdivisions*. A barrier-induced subdivision occurs, if at least $g/\gamma(t)$ barrier nodes are inside the frame (whether they are found or not). The other subdivisions are neighbor-induced and due to the Subdivision Rule. In the traffic analysis, a barrier that causes a subdivision "pays" for the (barrier-induced) subdivision of the current square as well as for the (neighbor-induced) subdivision of the neighboring squares. This extra cost adds a constant factor to the traffic for exploring a single square.

Theorem 2. *The algorithm produces traffic $\mathcal{O}(d + p \log^2 d)$.*

Corollary 2. *The JITE Algorithm has a constant competitive time ratio and a comparative traffic ratio of $\mathcal{O}(\log^2 d)$. It has a combined comparative ratio of $\mathcal{O}(\log^2 d)$.*

5 Conclusions and Open Problems

Conclusions In this paper we present an algorithm for the routing problem in faulty meshes which can route a message asymptotically as fast as a the fastest algorithm and (up to a term of $\mathcal{O}(\log^2 d)$) with only as many messages as the number of faulty nodes obstructing the messages plus the minimal path length. This considerably improves the known factors of $\mathcal{O}(\sqrt{d})$ [21] and more previously of $\tilde{O}(d^{\sqrt{\frac{\log\log d}{\log d}}})$ [22].

This is achieved by the JITE Algorithm combining several techniques. First we use a continuously expanding search area and establish an adaptive grid of frames which is denser when many barriers are around. On this grid a slowly proceeding shoreline simulates a flooding mechanism. This shoreline triggers the Just-In-Time Exploration (JITE) of new frames that are used by the shoreline. The artful combination of these techniques lead to an algorithm which needs time $\mathcal{O}(d)$ and traffic $\mathcal{O}(d + p \log^2 d)$ where d denotes the length of the shortest path and p denotes the number of border nodes being adjacent to faulty nodes.

Open problems This gives rise to the open question whether these bounds are tight, whether there is a small trade-off between time and traffic. The routing time is delayed by a large constant factor. It seems achievable to decrease this factor without an asymptotic increase of the traffic. However, it is not clear how. Another chance of improvement could be the use of randomized algorithms, which for many other problems outperform deterministic online algorithms.

A straight-forward generalization of this problem are three-dimensional meshes with faulty nodes. The JITE Algorithm, however, in its straight-forward generalization causes a significant increase in traffic. So the question for efficient online routing in higher dimensions is wide open.

References

1. D. Angluin, J. Westbrook, and W. Zhu. Robot navigation with distance queries. *SIAM Journal on Computing*, 30(1):110–144, 2000.
2. P. Berman. On-line searching and navigation. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 232–241. Springer, 1998.
3. A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26:110–137, 1997.
4. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
5. P. Bose and P. Morin. Competitive online routing in geometric graphs. *Theoretical Computer Science*, 324(2-3):273–288, September 2004.
6. P. Bose and P. Morin. Online routing in triangulations. *SIAM Journal on Computing*, 33(4):937–951, May 2004.
7. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.
8. R. J. Cole, B. M. Maggs, and R. K. Sitaraman. Reconfiguring Arrays with Faults Part I: Worst-case Faults. *SIAM Journal on Computing*, 26(16):1581–1611, 1997.

9. D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, pages 152–181. Kluwer, 1996.
10. E. Koutsoupias and Ch. H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):300–317, 2000.
11. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. 11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.
12. F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proc. of the 6th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 24–33, 2002.
13. C. Lucas. Comments on "dynamic path planning for a mobile automation with limited information on the environment". *IEEE Transactions on Automatic Control*, 33(5):511, May 1988.
14. V. J. Lumelsky. Algorithmic and complexity issues of robot motion in an uncertain environment. *J. Complex.*, 3(2):146–182, 1987.
15. V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.
16. M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network Magazine*, 15(6):30–39, November 2001.
17. Ch. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proc. of the 16th Int. Colloq. on Automata, Languages, and Programming (ICALP'89)*, pages 610–620, 1989.
18. C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. IETF RFC 3561, July 2003.
19. N. Rao, S. Kareti, W. Shi, and S. Iyenagar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms, 1993.
20. S. Rührup. *Position-based Routing Strategies*. PhD thesis, University of Paderborn, 2006.
21. S. Rührup and Ch. Schindelhauer. Competitive time and traffic analysis of position-based routing using a cell structure. In *Proc. of the 5th IEEE Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN'05)*, page 248, 2005.
22. S. Rührup and Ch. Schindelhauer. Online routing in faulty mesh networks with sub-linear comparative time and traffic ratio. In *Proc. of the 13th European Symposium on Algorithms (ESA'05), LNCS 3669*, pages 23–34. Springer, 2005.
23. S. Rührup and Ch. Schindelhauer. Improved bounds for online multi-path routing in faulty mesh networks. Technical Report TR-RSFB-06-078, University of Paderborn, 2006.
24. J. Wu. Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels. *IEEE Transactions on Parallel and Distributed Systems*, 11:149–159, February 2000.
25. J. Wu and Z. Jiang. Extended minimal routing in 2-d meshes with faulty blocks. In *Proc. of the 1st Intl. Workshop on Assurance in Distributed Systems and Applications*, pages 49–55, 2002.
26. L. Zakrevski and M. Karpovsky. Fault-tolerant message routing for multiprocessors. In *Parallel and Distributed Processing*, pages 714–730. Springer, 1998.