Randolph's Robot Game is NP-complete!

Birgit Engels* Tom Kamphans†

Abstract

We introduce a new type of movement constraints for a swarm of robots in a grid environment. This type is inspired by Alex Randolphs board game *Ricochet Robot* and may be used to model robots with very limited abilities for self localization: We assume that once a robot starts to drive in a certain direction, it does not stop its movement until it hits an obstacle wall or another robot. We show that the question, whether a given cell can be reached is NP-complete for arbitrary environments.

A Java applet for simulating robot swarms moving with these constraints can be found in
`http://www.geometrylab.de/RacingRobots/`

Keywords: Robot navigation, cellular environments, navigation errors, robot swarms, NP-completeness.

1 Introduction

Robot motion planning has received a lot of attention both in computational geometry and in robotics; see, for example, the surveys [3, 16, 11], or the books [15, 19, 6].

In this paper, we consider a quite simple model for the robots and their environment: The robots are short sighted and the surrounding is subdivided by a rectangular integer grid; that is, the robots move in a cellular environment, similar to a chessboard or squared writing paper.

Environments with a grid structure were considered in different settings. Icking et al. [12] studied the exploration problem (also known as covering) of a simple grid polygon (i.e., a polygon with no obstacles inside). They gave a lower bound of $\frac{7}{6}$ and a $\frac{4}{3}$-competitive exploration strategy for this problem. The case of a polygon with obstacles was considered by Icking et al. [10]—see also [14]—and independently by Gabriely and Rimon [9]. Itai et al. [13] showed that the corresponding offline problem is NP-hard. Betke et al. [4] and Albers et al. [1] studied the piecemeal exploration problem, where the robot has to return to the start cell every now and then. Cellular environments were also considered from a more practical point of view; see, for example, Moravec and Elfes [17].

Swarms of robots have been studied intensely. See, for example, Bruckstein et al. [5]. Arkin et al. [2] considered the *freeze-tag problem* (i.e., the question how to 'wake up' an initially inactive swarm of robots).

Figure 1: The board game *Ricochet Robots* by Alex Randolph.

Another interesting model for robots moving around in cellular environments was inspired by the board game *Ricochet Robots* by Alex Randolph [18], see Figure 1. The interesting part of the game are the rules to move a robot: A robot can move in one of the four directions (north, east, south, or west), but once it has chosen a direction it continues to move in this direction until it hits an obstacle or another robot. Thus, it is often necessary to move robots that serve as guides to stop the movement of another robot on an appropriate cell. See, for example, Figure 2: The task is to move the robot $\otimes$ to the cell marked with $\diamond$. To permit this movement, the robot $\oplus$ has to move to a, so three moves are necessary to solve the task.

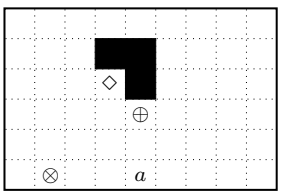

Figure 2: Example: the robot $\oplus$ has to move to a to allow the robot $\otimes$ a movement to $\diamond$.

This model can be used for a swarm of robots. Each of them has a very restricted orientation: Even if the robots have a map of their environment, a robot that touches a wall knows only, which wall in the environment it touches, but as soon as the robot leaves the

*Universität zu Köln, Zentrum für angewandte Informatik (ZAIK), 50831 Köln, Germany. engels@zpr.uni-koeln.de

†University of Bonn, Institute of Computer Science I, 53117 Bonn, Germany. kamphans@cs.uni-bonn.de

wall it has no chance to locate itself. Therefore, it continues its movement until it hits another wall or another robot. However, the robots are able to communicate with each other, or all of them are controlled by the same computer. Apart from the best strategy to solve Randolph's game, an interesting question is, whether there is an upper bound for the number of robots, such that every cell can be reached by at least one robot. In this paper we show that the reachability problem in arbitrary environments is NP-complete.

2 Preliminaries

We assume that the robot's environment is subdivided by a rectangular integer grid. We call a reachable basic block in the environment a *cell*, and the set of all cells that can be reached by the robots a *grid polygon*, or polygon for short. An unreachable block is called an *obstacle* or *hole*.

We assume that the environment is populated by a system R of N robots $r_1, \ldots, r_N$. The robots start either from a common start cell s or from a given start configuration $S = (s_1, \ldots, s_N)$ inside the polygon. The sensors of a robot provide the information, which of the four neighbors of the currently occupied cell belong to the polygon and which ones do not. Furthermore, the robot is able to recognize neighboring cells occupied by another robot. A robot can enter an unoccupied polygon cell, but once it started to drive in a certain direction, it will continue the movement until it hits a wall (i.e., an obstacle) or another robot. We assume that in any point of time at most one robot moves.

In spite of the very basic sensors, the robots are either able to communicate with each other or they are steered by a common controlling unit. However, the robot system R provides enough memory to store a map of the environment and some additional information.

3 Reachability in Arbitrary Polygons

Let the *Reachability Problem* be defined as follows: Given an arbitrary grid polygon P, a system of N *Randolph robots* $r_1, \ldots, r_N$, a start configuration $S = (s_1, \ldots, s_N)$, and a target cell t. Is one of the robots able to reach t—probably with the help of the other robots?

In this section, we show that the Reachability Problem is NP-complete by reducing the well known satisfiability problem with three literals per clause (3SAT) to the Reachability Problem. Let us recall the 3SAT Problem: Given a boolean expression, α, consisting of m clauses $C_1, \ldots, C_m$ over n variables $X_1, \ldots, X_n$, where each clause consists of three literals; that is, $\alpha = C_1 \wedge \ldots \wedge C_m$ with $C_i = (L_{i1} \vee L_{i2} \vee L_{i3})$ and $L_{ij} \in \{ X_\ell, \neg X_\ell;\ 1 \leq \ell \leq n \}$. Is there a truth assignment for $X_1, \ldots, X_n$ such that α is fulfilled?

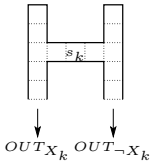

Figure 3: A fork polygon.

Given an expression α, we construct a polygon, $P(\alpha)$. In the following, we give a brief description of the construction.[1] The robot system R consists of $n+1$ robots, $r_0, r_1, \ldots, r_n$, where n is the number of variables in the 3SAT instance. The robot r_0 plays a special role: it starts on a special start cell s_0 and its purpose is to reach t. Note, that r_0 is the only robot that may reach t, if t is reachable at all. Each of the other robots r_k, $1 \leq k \leq n$, represents a variable X_k and its truth assignment. Thus, we call these robots *literal robots*. A literal robot r_k starts at a cell s_k in a special fork-shaped (sub-)polygon called *fork polygon* fp_k, see Figure 3. As soon as r_k leaves the fork polygon, it is impossible for r_k to return to s_k and to enter the other vertical passage. This property ensures the consistency of the truth assignment for the variables in α. By convention, we assign $X_k := 1$ if the robot enters the left-hand vertical passage of the corresponding fork polygon, and $X_k := 0$ otherwise. We denote the former passage with X_k*-passage* and the latter with $\neg X_k$*-passage* of fp_k. We have:

Lemma 1 *The truth assignment of the variables that is derived from the movement of the literal robots is consistent.*

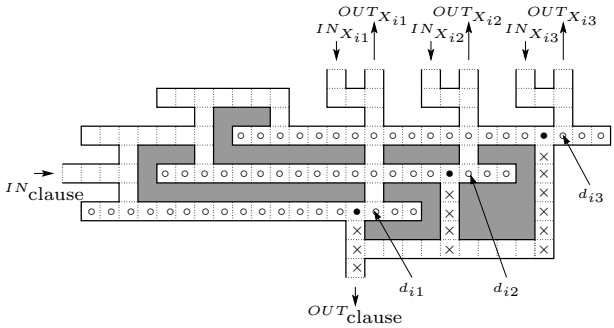

Figure 4: A clause polygon.

For every clause C_i in α we construct a corresponding (sub-)polygon called the *clause polygon* cp_i, see

[1] For more details and proofs see our technical report [8], where we discuss also lower bounds on the number of robots needed to reach every cell. For example, it is easy to see that three robots are necessary and sufficient to reach every cell in a rectangular environment.

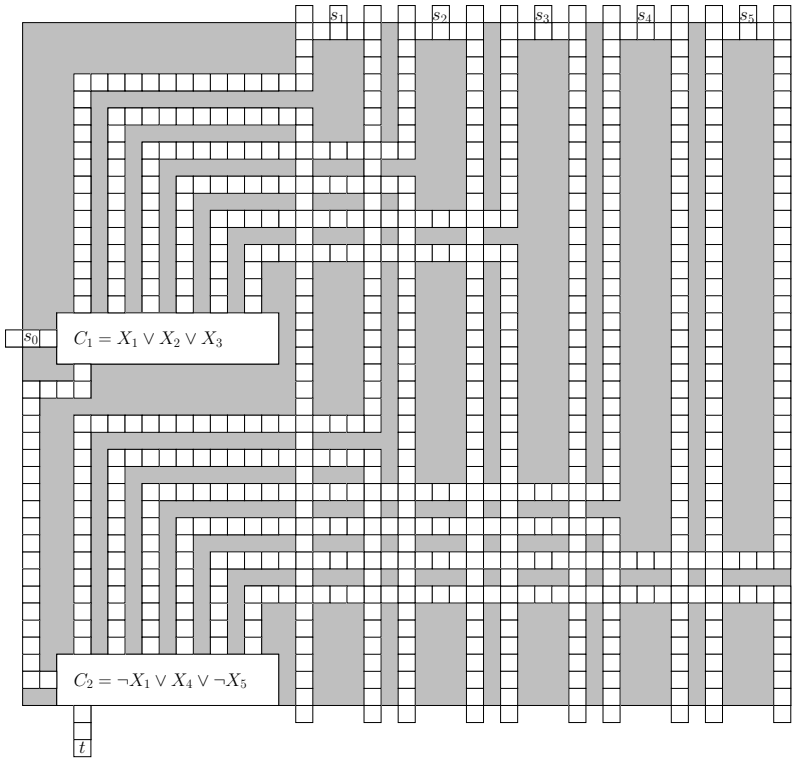

Figure 5: Construction example for $P(\alpha)$ corresponding to $\alpha = C_1 \wedge C_2 = (X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee X_4 \vee \neg X_5)$.

Figure 4. The robot r_0 may pass a clause polygon if and only if the clause polygon is visited by at least one of the *literal robots* corresponding to the literals in C_i. This, in turn, is only possible, if the clause C_i in α is fulfilled. More precisely, a robot r_k may enter cp_i, if it represents $X_k = 1$ and X_k is a literal in C_i, or if it represents $X_k = 0$ and $\neg X_k$ is a literal in C_i. Moving south into cp_i, a robot r_k stops on a cell d_{ij} marked with an arrow in Figure 4. Now, imagine that the robot r_0 moves from IN_{clause} to the horizontal passage marked with $\circ$ where r_k is positioned. The robot r_0 stops in front of r_k and can change its direction to enter the vertical passage marked with $\times$ in Figure 4 leading to OUT_{clause}. Note that r_0 cannot pass the clause polygon without further help of other robots. The main property of the clause polygons is the following:

Lemma 2 *The robot r_0 may pass a clause polygon from IN_{clause} to OUT_{clause} if and only if the corresponding clause in α can be fulfilled.*

In the last step of the construction, we arrange and combine the clause and fork polygons to one polygon $P(\alpha)$: We arrange the clause polygons one beneath the other on the left-hand side of $P(\alpha)$ and the fork polygons side by side on top of $P(\alpha)$, each of them with sufficient space for the connections, see the example in Figure 5. Then we consecutively connect all clause polygons by a passage. More precisely, for $1 \leq i < m$ we connect OUT_{clause} of cp_i and to IN_{clause} of cp_{i+1}. Further, we connect IN_{clause} of cp_1 to the start cell s_0 of r_0 and OUT_{clause} of cp_m to the target cell t. Thus, r_0 has to pass consecutively all clause polygons.

Now we connect the fork polygons to the clause polygons: First, we extend the X_k-passages and the $\neg X_k$-passages of the fork polygons to the south until they reach the last clause polygon. Thus, we have $2n$ vertical corridors parallel to the column of clause polygons. Each of these corridors ends in a blind alley. Then we add connections from this bus structure to the clause polygons: If X_k is the jth literal in the clause C_i of α, we divert r_k from the X_k-passage via $IN_{X_{ij}}$ through cp_i and via $OUT_{X_{ij}}$ back to the X_k-passage. Remark that we add an obstacle cell to the X_k-passage between the horizontal connections to the clause polygon. Analogously, if $\neg X_k$ is the jth literal, we connect $IN_{X_{ij}}$ and $OUT_{X_{ij}}$ to the $\neg X_k$-passage. Note that the passages are mostly separated by obstacles—the only exceptions are crossings of connecting passages. But these crossings do not matter, a literal robot r_k stays in its X_k- or $\neg X_k$-passages, after it left the fork polygon. Further, r_0 cannot reach one of these passages.

Altogether, we arrived at the basic idea of our proof: If t is reachable by r_0, r_0 must pass every clause polygon. At the same time r_0 reaches a clause polygon

cp_i, at least one literal robot r_k must enter cp_i. This corresponds to the conditions to fulfill α: The truth assignment derived from the literal robots that enter the clause polygons ensures that there is at least one literal fulfilled in every clause. On the other hand, any given truth assignment that satisfies α fulfills at least one literal for every clause of α; thus, for every clause polygon in $P(\alpha)$ there is at least one literal robot able to enter it. Further, all clauses are connected in a serial way, which corresponds to the conjunction of clauses in α. Thus, we can state:

Lemma 3 *There is a truth assignment that fulfills α, if and only if t is reachable by R in $P(\alpha)$.*

It is easy to see that we need a construction time in $O(mn)$, but this time is still polynomial in the size of α. Further, we can construct a nondeterministic Turing machine that chooses nondeterministically a sequence of movements to reach t and verifies the reachability of t in polynomial time, see [7] for more details. Altogether, we have:

Theorem 4 *The Reachability Problem is NP-complete.*

4 Summary

We considered robot swarms moving in cellular environments under a new type of movement constraint and showed that the question, whether a cell can be reached by a robot, is NP-complete for arbitrary environments. A Java applet for simulating robot swarms moving under our constraints can be found in

`http://www.geometrylab.de/RacingRobots/`

Interesting open problems are, whether there is a constant lower bound for polygons without holes and without corridors of width 1, and whether there is an efficient algorithm for the Reachability Problem in this type of polygons.

References

[1] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32:123–143, 2002.

[2] E. M. Arkin, M. A. Bender, S. P. Fekete, J. S. B. Mitchell, and M. Skutella. The freeze-tag problem: how to wake up a swarm of robots. In *Proc. 13th Annu. ACM-SIAM Symp. Disc. Algor.*, pages 568–577, 2002.

[3] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms*. Springer-Verlag, 1998.

[4] M. Betke, R. L. Rivest, and M. Singh. Piecemeal learning of an unknown environment. *Machine Learning*, 18(2–3):231–254, 1995.

[5] A. M. Bruckstein, M. Lindenbaum, and I. A. Wagner. Distributed covering by ant-robots using evaporating traces. *IEEE Trans. Robot. Autom.*, 15:918–933, 1999.

[6] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005.

[7] B. Engels. Navigation in Gitterumgebungen für verteilte Robotersysteme mit eingeschränkter Sensorik. Diplomarbeit, Universität Bonn, August 2005. http://www.geometrylab.de/RacingRobots/.

[8] B. Engels and T. Kamphans. Randolphs robot game is NP-complete! Technical Report 005, Department of Computer Science I, University of Bonn, 2005. http://web.informatik.uni-bonn.de/I/publications/ek-rrgin-05.pdf.

[9] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comput. Geom. Theory Appl.*, 24:197–224, 2003.

[10] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring an unknown cellular environment. In *Abstracts 16th European Workshop Comput. Geom.*, pages 140–143. Ben-Gurion University of the Negev, 2000.

[11] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In H. Bunke, H. I. Christensen, G. D. Hager, and R. Klein, editors, *Sensor Based Intelligent Robots*, volume 2238 of *Lecture Notes Comput. Sci.*, pages 245–258, Berlin, 2002. Springer.

[12] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring simple grid polygons. In *11th Internat. Comput. Combin. Conf.*, volume 3595 of *Lecture Notes Comput. Sci.*, pages 524–533. Springer, 2005.

[13] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.

[14] T. Kamphans. *Models and Algorithms for Online Exploration and Search*. PhD thesis, University of Bonn, to appear.

[15] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[16] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[17] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 116–121, 1985.

[18] A. Randolph. Ricochet robots. Board Game, german edition by Abacus Games, Dreieich, 1999.

[19] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.