Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Birgit Engels[1] **Tom Kamphans**[2]

On the Complexity of Randolph's Robot Game

Technical Report 005
December 2005

[1]Zentrum für angewandte Informatik, University of Cologne, Germany
[2]Universität Bonn, Institut für Informatik, Abt. I, Römerstr. 164, D-53117 Bonn

Abstract

We introduce a type of movement constraints for a swarm of robots in a grid environment, which is inspired by Alex Randolph's board game *Ricochet Robot* and new to the field of robot motion planning. This type of movement 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 give some lower bounds on the number of robots needed to reach every cell. Especially, it is easy to see that three robots are necessary and sufficient to reach every cell in a simple rectangular environment. Further, we consider the question, whether a certain cell can be reached is in arbitrary environments.

A Java applet for simulating robot swarms moving with these constraints can be found in

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

Key words: Robot navigation, unknown environment, navigation error, robot swarms, NP-hardness, PSPACE-completeness.

1 Introduction

Robot motion planning has received a lot of attention both in computational geometry and in robotics; see, for example, the surveys [16, 29, 21, 31, 5, 13, 26, 18], or the books [30, 25, 32, 9].

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. [20] 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. [17, 19] —see also [23]—and independently by Gabriely and Rimon [12]. Itai et al. [22] showed that the corresponding offline problem is NP-hard. Betke et al. [6] 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 [27, 10], or Zelinsky et al. [8].

Swarms of robots have been studied intensely. See, for example, the works of Bruckstein et al. [7] or Amato et al. [2]. Arkin et al. considered the *freeze-tag problem* (i.e., the question how to 'wake up' an initially inactive swarm of robots) [4] and the dispersion of a swarm of robots [3].

Another interesting model for robots moving around in cellular environments was inspired by the board game *Ricochet Robots* by Alex Ran-

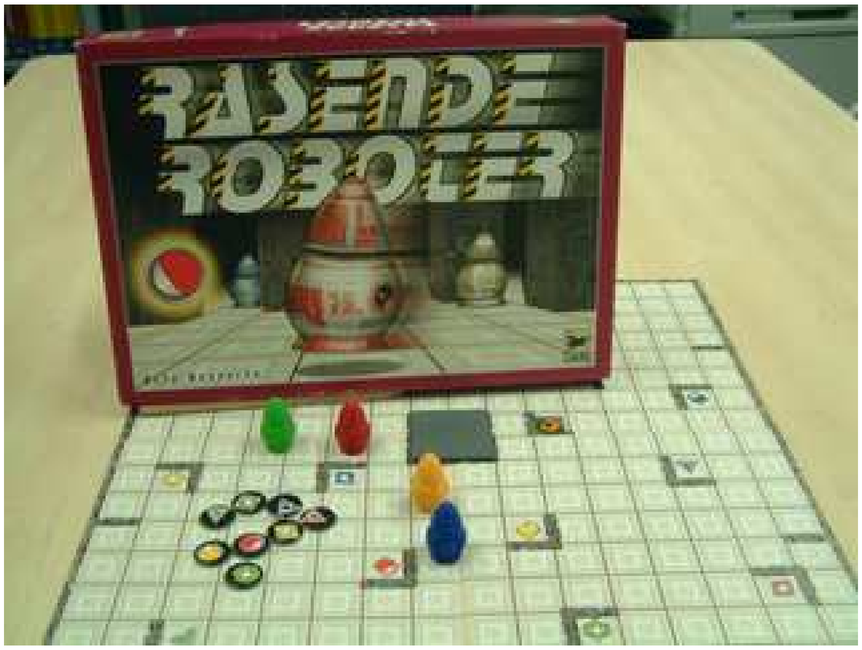


Figure 1: The board game *Ricochet Robots* by Alex Randolph.

dolph [28], see Figure 1. The game consists of four robots in different colors, several marker chips, and a game board showing some obstacles and some cells marked with different symbols. Initially, the robots are placed randomly on the board and the markers are hidden. In every turn one of the markers is drawn. The players try to figure out the smallest number of moves that are necessary to move the robot in the revealed color to the cell with the revealed symbol. 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. Of course, the player that finds the minimal number of movements wins the turn.

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 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

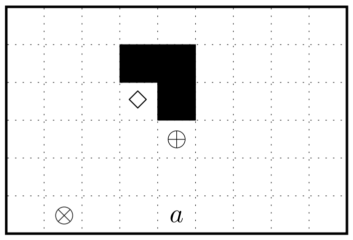


Figure 2: Example: the robot $\oplus$ has to move to a to allow the robot $\otimes$ a movement to $\diamond$.

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 report, we show—after a formal definition for the robots and their environment in Section 2— upper bounds for some rather simple configurations in Section 3.

The major contribution is the NP-hardness proof for the reachability problem in arbitrary environments that we give in Section 4. Moreover, this result can be extended to PSPACE-completeness by an analogy to the game *which* was independently analysed by Hüffner et al. [15], and Holzer and Schwoon [14].

2 Preliminaries

We assume that the robot's environment is subdivided by a rectangular integer grid, see Figure 3.

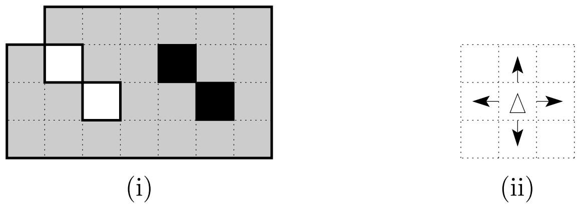


Figure 3: (i) Polygon with 23 cells and one hole (black cells) inside, (ii) the robot can determine which of the 4 adjacent cells are free, and enter an adjacent free cell.

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 call a polygon *simple*, if it has no holes inside.

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[1], 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. Further, 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 Upper bounds on the Number of Robots

If there are passages of width 1 in the environment, we can "trap" robots. See, for example, Figure 4(i): The polygon includes a corridor of width 1, and we need $\lceil \frac{k}{2} \rceil + 1$ robots starting in s before t is reached—one robot in a and $\lceil \frac{k}{2} \rceil$ robots to "fill" the corridor from the left or from the right. Note that we need one robot located in the cell b before another robot is able to reach a, but the robot in b can be used again after a is occupied.

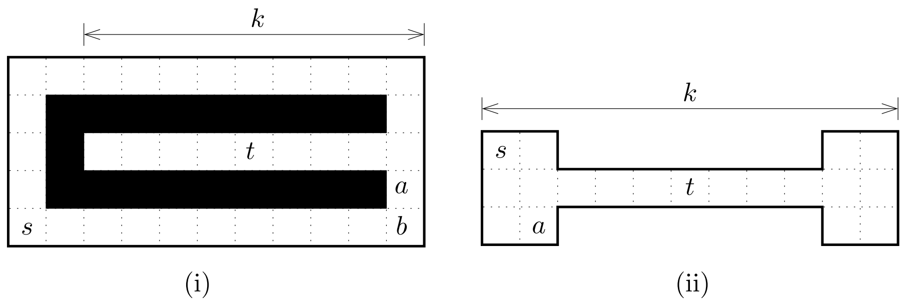


Figure 4: In both cases, $\lceil \frac{k}{2} \rceil + 1$ robots are necessary to reach t: one located in a, and $\lceil \frac{k}{2} \rceil$ to "fill" the corridor.

Even if we have a simple polygon, corridors of width 1 may cause the need for an arbitrary number of robots, see Figure 4(ii): We need also $\lceil \frac{k}{2} \rceil + 1$ robots starting in s to occupy t. But what happens, if the polygon does not have such narrow passages? Are there still polygons that need an arbitrary number of robots, or is there an upper bound? So far, only the following upper bound is known.

[1]In this case, we imagine the robots enter the polygon successively through a door in the wall or in the floor.

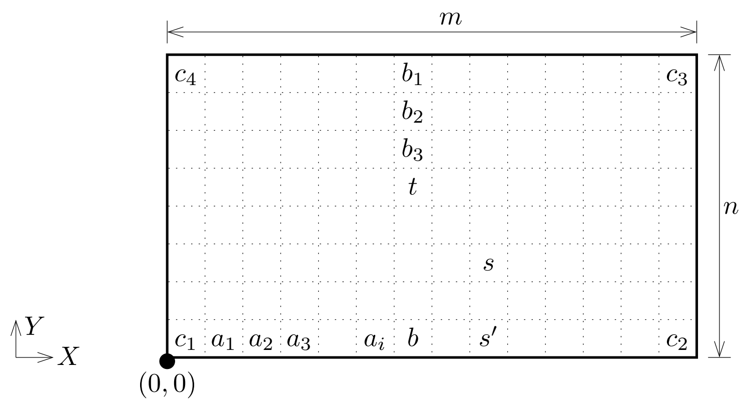


Figure 5: t can be reached with $O(n+m)$ steps using 3 robots.

Lemma 1 *Given a rectangle of size $m \times n$, $m, n > 1$, without holes inside, every cell can be reached using at most three* Randolph robots *with $O(m+n)$ moves.*

Proof. Consider an arbitrary start cell, s, and an arbitrary target, $t = (x, y)$, see Figure 5. W.l.o.g. we assume that $x \leq \lceil \frac{m}{2} \rceil$ and $y \geq \lceil \frac{n}{2} \rceil$ holds; otherwise, we choose another cell than c_1 in the following.

We reach t using the following strategy. In the first stage, we place one robot to mark the column containing t: The first robot moves from s via s' to c_1 with 2 moves, the second robot moves from s to a_1. Now, the first robot moves via c_4, c_3 and c_2 to a_2, the second robot moves a similar path to a_3. This continues, until eventually one of the two robots has reached the cell a_i. With the help of the robot in a_i we can proceed vertically to t: The other robot moves via c_3, c_2 and b to b_1. Now, we need a third robot that moves via s' and b to b_2, and the robot located on b_1 is now free to move to b_3. We proceed in this way until we reach t. It is easy to see that we use no more than $4\lceil \frac{m}{2} \rceil$ moves to occupy the cells a_i and b, and no more than $4\lceil \frac{n}{2} \rceil$ further moves to reach t, yielding $O(n+m)$ moves. $\square$

So far, we were not able to give an upper or lower bound for the number of robots needed in more complex environments (without holes inside and without corridors of width 1). Although it is possible to increase the lower bound slightly by introducing "traps" that have to be guarded by additional robots (marked with $\oplus$ in Figure 6), it is not possible to repeat this construction in a recursive way to show an unbounded number of needed robots. We can add traps to the traps, but as the number of robots increases, we have enough robots to "rescue" the guards inside a trap that are no longer needed if a guard is placed in $\oplus$.

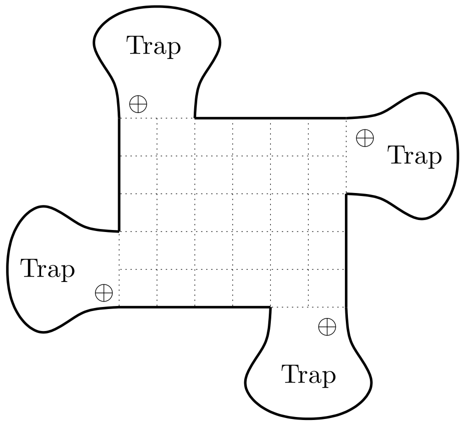


Figure 6: An attempt to increase the lower bound.

4 Reachability in Arbitrary Polygons

Now, we consider the following problem, the *Reachability Problem*:

> 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-hard by reducing the well known satisfiability problem with three literals per clause (3SAT) [24] 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:
>
> $$\alpha = C_1 \wedge \ldots \wedge C_m$$
>
> $$\text{with } C_i = (L_{i1} \vee L_{i2} \vee L_{i3}) \text{ and } L_{ij} \in \{ X_\ell, \neg X_\ell;\ 1 \le \ell \le n \}\,.$$
>
> Is there a truth assignment for $X_1, \ldots, X_n$ such that α is fulfilled?

Let us first give a brief outline on the reduction: We construct a polygon, $P(\alpha)$, depending on α. Our robot system, R, consists of $n+1$ robots, $r_t = r_0, r_1, \ldots, r_n$, where n is the number of variables in the 3SAT instance. Each robot r_i, $1 \le i \le n$, represents the variable X_i and its truth assignment. Thus, we call these robots *literal robots*. Each of them starts in a special fork-shaped (sub-)polygon called *fork polygon* fp_k, see Figure 7, and chooses the truth assignment of X_i immediately after leaving its start cell. The robot r_0 plays a special role: it starts on a special start cell, $s = s_0$, and its purpose is to reach t. Thus, we refer to this robot as r_t. Note, that r_t is the only robot that may reach t, if t is reachable at all. Further, we construct for every clause C_i in α a corresponding (sub-)polygon called the *clause polygon* cp_i, see Figure 8. The robot r_t 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. We use a bus-like structure to connect the fork polygons to the clause polygons and to construct $P(\alpha)$.This structure ensures that a clause polygon can be visited only by robots that have chosen the adequate truth assignment. For convenience, we use corridors of width 1 in our constructions. It is easy to widen these corridors, so our results hold even if we do not allow corridors of width 1.

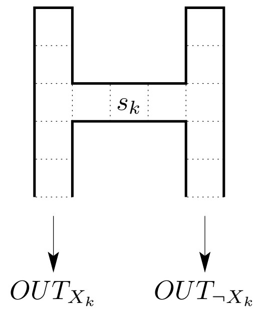


Figure 7: A fork polygon.

Fork Polygons

We start our more precise description with the fork polygons, see Figure 7. Starting from s_k in fp_k, a literal robot r_k may move to the left or to the right several times without further effect. But as soon as it moves up or down in one of the vertical passages, it needs another robot (or an obstacle) to allow a change in its direction of movement on one of the cells where the horizontal and vertical corridors intersect. The bus structure ensures that no other robot r_ℓ, $\ell \neq k$, may enter fp_k; thus, it is impossible for the robot r_k to return to s_k and to enter the other vertical passage. This property of the fork polygon ensures the consistency of the truth assignment for the variables in α: As mentioned earlier, a literal robot r_k is identified with one variable X_k. Now, we are able to assign a truth value to X_k corresponding to the vertical corridor that r_k enters. 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. Altogether, we have

Lemma 2 *The truth assignment of the variables that is derived from the movement of the literal robots is consistent.*

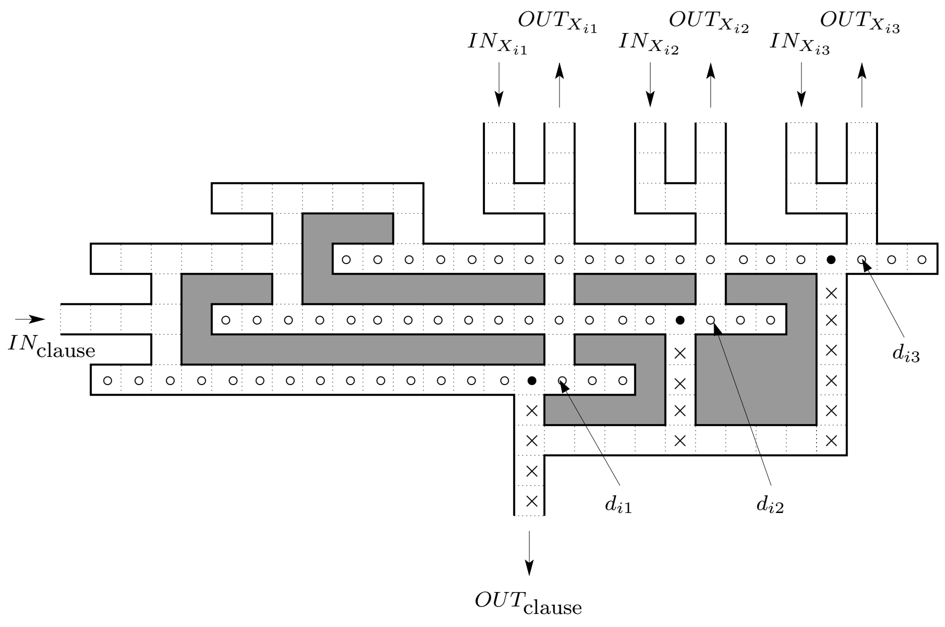


Figure 8: A clause polygon.

Clause Polygons

Now, let us consider the clause polygons. A clause polygon cp_i, $1 \leq i \leq m$, has eight connections to other parts of $P(\alpha)$. The connection between the subpolygons of $P(\alpha)$ ensures that only r_t may enter a clause polygon via IN_{clause} and leave via OUT_{clause}. Moreover, a literal robot r_k may enter the clause polygon via IN_{ij}, $1 \leq j \leq 3$, if and only if the variable X_k occurs in the corresponding clause C_i and the robot has chosen the correct truth assignment. 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 8. Now, r_k is able to enter one of the horizontal passages marked with $\circ$. Note that r_k is not able to leave the clause polygon without the help of at least two other robots once it left the cell d_{ij} in a horizontal direction. The bus structure ensures that these cells can be reached only by the robot r_t and at most one variable robot (see Lemma 4); thus, r_k is trapped if it leaves d_{ij} in a horizontal direction.

Now, let us assume that r_k does not leave d_{ij}, and r_t moves from IN_{clause} of our clause polygon to the same corridor, where r_k is already positioned. The robot r_t will stop in front of r_k and can change its direction to enter the vertical passage marked with $\times$ in Figure 8 leading to OUT_{clause}. The basic property of the clause polygons is the following:

Lemma 3 *The robot r_t may pass a clause polygon from IN_{clause} to OUT_{clause} if and only if the corresponding clause in α can be fulfilled.*

Proof. First, we show that r_t cannot pass the clause polygon without further help of other robots: Entering cp_i through IN_{clause}, the robot r_t may reach one of the three horizontal passages marked with $\circ$ in Figure 8; all of them provide a connection to OUT_{clause} by the vertical passages marked with $\times$. But r_t is not able to enter the latter passages because of our special motion behavior: There is no obstacle in one of the cells d_{ij} (marked with an arrow) that would allow r_t to alter its moving direction.

Only if d_{ij} is occupied by another robot r_k, the robot r_t may stop on one of the cells marked with $\bullet$ and reach OUT_{clause}. But a literal robot r_k reaches d_{ij} only if X_k is a literal in C_i and r_k has chosen the correct a truth assignment corresponding to the type of the literal; that is, if and only if C_i is fulfilled. $\square$

Connections

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 9. 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_t and OUT_{clause} of cp_m to the target cell t. Thus, r_t 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 end in a blind alley.[2] 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:[3]

Lemma 4 *A literal robot, r_k, stays in its X_k- or $\neg X_k$-passages, after it left the fork polygon. Further, r_t cannot reach one of these passages.*

[2]Regarding the construction time, we would like a single reading of α to be sufficient for the building of $P(\alpha)$. Since we cannot determine which of the clause polygons has to be connected to which passage, the extension up to the last clause polygon guarantees the access to every passage for every clause.

[3]Recall that no two robots move simultaneously.

Proof. All literal robots r_k start from different cells s_k, and enter different vertical passages leading to clause polygons and dead ends. Only the clause polygons connect two different passages, because no robot is able to stop at a crossing between connecting passages and, thus, no robot can change its direction in such a crossing in order to enter another passage. Entering a clause polygon cp_i by $IN_{X_{ij}}$, a robot r_k moves to *one* of the horizontal passages marked with $\circ$ in Figure 8. Inside a clause polygon, a robot r_k may meet only the robot r_t. However, it is easy to see that r_k cannot enter another horizontal passage or possibly even leave cp_i via OUT_{clause} or IN_{clause}, and r_t cannot leave cp_i via one of the $OUT_{X_{ij}}$. Further, there is only one configuration such that r_k can return to its passage by $OUT_{X_{ij}}$ and r_t can leave cp_i through OUT_{clause}. This holds for every r_k; thus, no literal robot can leave the passage it has chosen (i.e., either the X_k- or the $\neg X_k$-passage). $\square$

The second important property of our connections concerns the reachability of t:

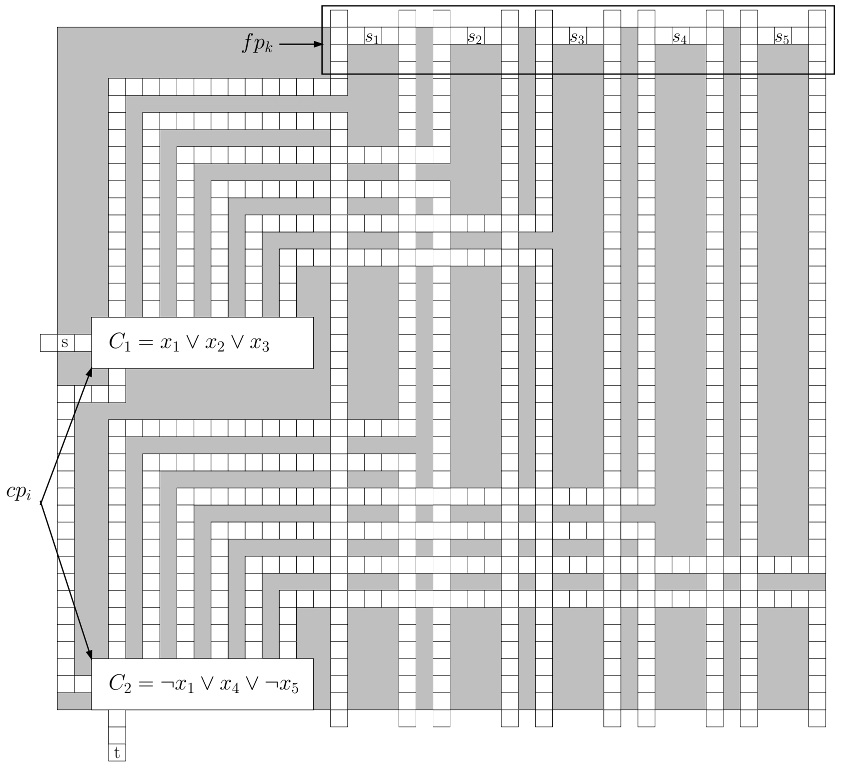
Lemma 5 *The robot r_t is the only robot that may reach t.*

Proof. To reach t, a robot must at least pass the last clause polygon in our construction, because there is no other connection to t. As we have seen earlier, no literal robot r_k can leave a clause polygon through OUT_{clause}; thus, only r_t may reach t. $\square$

Example

Let us consider the example shown in Figure 9: The left part of the construction consists of two clause polygons,[4] cp_1 and cp_2, corresponding to the clauses C_1 and C_2 of α. Both clause polygons are connected by a passage from OUT_{clause} of cp_1 to IN_{clause} of cp_2. The start cell s_0 for r_t is attached to IN_{clause} of cp_1 and the target cell t accordingly to OUT_{clause} of cp_2. The upper right part of the figure shows the five fork polygons corresponding to the variables $X_1, \ldots, X_5$ and containing the start cells s_k for the five literal robots $r_1, \ldots, r_5$. Between these components we have the bus-like connecting passages: The first literal in C_1 is X_1. Thus, we add grid cells to connect the X_1-passage to cp_1 via $IN_{X_{11}}$ and $OUT_{X_{11}}$. Analogously, $IN_{X_{12}}$ and $OUT_{X_{12}}$ are connected to the X_2-passage and so on. Note that $IN_{X_{21}}$ and $OUT_{X_{21}}$ are connected to the $\neg X_1$-passage, because the literal X_1 is negated in C_2.

[4]For convenience, both clause polygons are shown as black boxes. The interior of these boxes is shown in Figure 8.



fp_k
s_1
s_2
s_3
s_4
s_5
s
$C_1 = x_1 \vee x_2 \vee x_3$
cp_i
$C_2 = \neg x_1 \vee x_4 \vee \neg x_5$
t

Figure 9: Construction example for $P(\alpha)$ corresponding to $\alpha = C_1 \wedge C_2 = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee \neg x_5)$.

Truth Assignment

The preceding explanations lead to the basic idea of our proof: If t is reachable by r_t, r_t must pass every clause polygon. At the same time r_t 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. Further, all clauses are connected in a serial way, which corresponds to the conjunction of clauses in α. Thus, we can state:

Lemma 6 *There is a truth assignment that fulfills α, if and only if t is reachable by R in $P(\alpha)$.*

Proof. If t is reachable at all, by Lemma 5 it is reached by r_t. To reach t, r_t has to pass consecutively every clause polygon of $P(\alpha)$, because the clause polygons are connected in a serial way. By Lemma 3 (and Lemma 4), r_t can pass a clause polygon if and only if the corresponding clause in α is fulfilled. 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. Altogether, r_t is able to pass every clause polygon and reach t. If t is not reachable, there is at least one clause polygon cp_i that is not reached by one of the literal robots; thus, the clause C_i—and likewise α—cannot be fulfilled. □

Running Time

So far, we did not observe the running time needed for the construction of $P(\alpha)$. It is easy to see that this construction takes time linear in the number of cells in $P(\alpha)$, which, in turn, is polynomial in the size of α: There are m clause polygons in $P(\alpha)$—one for every clause in α— and n fork polygons, where n is the number of variables in α. The number of cells used for a clause and a fork polygon as well as a connection between two clause polygons is constant. Thus, we have $O(m + n)$ cells so far. Concerning the connecting passages of $P(\alpha)$, we have two vertical passages (X_k- and $\neg X_k$-passage) of length $O(m)$ for every variable in α in the bus-like structure, which are interrupted by horizontal passages leading to and coming from the clause polygons. Every clause polygon provides six of these passages. The vertical segments of these passages are bound by a constant, but the horizontal segments depend on the width of the whole construction and, in turn, by the number of variables. Thus, we have to charge a construction time in $O(mn)$ for the connection between fork and clause polygons, but this time is still polynomial in the size of α. Altogether, we have:

Theorem 7 *The Reachability Problem is NP-hard.*

5 Complexity of the Reachability Problem

Despite the proof of Theorem 7 the actual complexicity of the Reachability Problem stated by Randolph's robot game remained open until the discovery of some analogy to the game *Atomix*, a solitaire game invented by Günter Krämer in 1990. The task is to assembly the initially scattered atoms of a special target molecule in a grid polygon similar to the environment of the Randolph robots.

Inspired by the motion behaviour of matter in free space, the single atoms move according to the same rules as the Randolph robots. The main differences between the task of an Atomix player and the solution to our Reachability Problem are the following:

- Instead of one given target cell to be reached by any of the robots, there is a target cell for each atom—implicitly given by the target molecule.

- The target cells for the atoms are no fixed cells in the grid polygon, but are given relative to the other atoms due to the neighboring relation defined by the target molecule.
- The atoms may be of a different type (Hydrogen, Oxygen, Carbon, etc.) and therefore not exchangable with regard to their target cells.

Although these may be considered major differences, the results of Hüffner et al. [15] as well as Holzer and Schwoon [14] showing that Atomix is PSPACE-complete can be fully applied to our Reachability Problem.

To proof the PSPACE-hardness of Atomix, Holzer and Schwoon construct an Atomix instance simulating a finite automaton by employing devices which resemble our fork and clause polygons and also make extensive use of blind alleys. Furthermore these automaton instances can be combined to a reduction of the nonemptiness intersection problem for n finite automata which is known to be PSPACE-complete.

We make a few remarks regarding the differences between the Atomix definition and the Reachability Problem mentioned above: Holzer and Schwoon use an artificial target molecule consisting of only two types of atoms (2 hydrogen atoms and n oxygen atoms). The hydrogen atoms are placed in a simple subpolygon called *reaction chamber*, which they cannot leave. Thus the position of the target molecule and therefore the specification of the target cells is fixed to some extend. Moreover all remaining atoms which still need to reach their target cells are of the same type and therefore resemble a swarm of Randolph robots. To gain a Reachability Problem with a single target cell from the constructed Atomix instance, we can—in terms of Atomix—simply take a water molecule (H=O=H) for target molecule, but improve the interior of the reaction chamber in such a way that all other oxygen atoms (or rather Randolph robots in this context) are needed to enable one of the oxygen atoms to reach its target position.

In terms of Randolph's robot game, we consider the grid polygon from the reduction of Holzer and Schwoon and exchange all oxygen atoms at their initial positions with Randolph robots to get a swarm $R = (r_1, \ldots, r_n)$ and a start configuration $S = (s_1, \ldots, s_n)$. We abandon the hydrogen atoms. Afterwards we place a subpolygon $P(\alpha)$ into the reaction chamber, where

$$\alpha = (x_1 \vee x_1 \vee x_1) \wedge (x_2 \vee x_2 \vee x_2) \wedge \ldots \wedge (x_{n-1} \vee x_{n-1} \vee x_{n-1})$$

and $P(\alpha)$ is constructed as explained in Section 4.[5] Thus, we get an instance of the Reachability Problem with a swarm of n Randolph robots and a single target cell which can be reached by r_n if and only if all n robots reach their seperate entrances to the reaction chamber. As the latter condition is also required for the reduction presented by Holzer and Schwoon and the

[5] $P(\alpha)$ can of course be simplified due to the special instance α; nevertheless our construction of $P(\alpha)$ satisfied the requirements.

properties of all other parts of their construction depend only on the motion behaviour of atoms in Atomix—which is equivalent to the one of Randolph robots—,we can adopt the PSPACE-hardness result for Randolph's robot game.

Furthermore after Hüffner et al. presented a simple proof for Atomix $\in$ PSPACE (which applies to the Reachability Problem as well), Holzer and Schwoon constructed an instance of Atomix which works as a pseudo n-bit counter and therefore gives an example for an instance with an exponentially long solution. Applying the same extensions to their construction as above this also yields for the Reachability Problem with Randolph robots. Finally we state:

Theorem 8 *The Reachability Problem in arbitrary polygons is PSPACE-complete.*

6 Summary

We considered robot swarms moving in cellular environments under a certain type of movement constraint resembling he movement behaviour of the atoms in the Atomix game. For Alex Randolph's robot rame we showed independantly that there are polygons with C cells where we need $\Theta(C)$ robots to reach every cell, if we allow corridors of width 1. Further, we gave an upper bound of 3 robots for rectangles without holes. Our main result is that the question, whether a cell can be reached by a robot, is NP-hard for arbitrary environments. We extended this result by modification of the proofs by Hüffner et al. [15] and Holzer and Schwoon [14] to the PSPACE-completeness statement for the Reachability Problem.

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] N. M. Amato, O. B. Bayazit, and J.-M. Lien. Swarming behavior using probabilistic roadmap techniques. In E. Sahin and W. M. Spears, editors, *Internat. Workshop Swarm Robotics*, volume 3342 of *Lecture Notes Comput. Sci.*, pages 112–125, Berlin Heidelberg, 2004. Springer.

[3] E. M. Arkin, M. A. Bender, S. Fekete, T.-R. Hsiang, and J. S. B. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *Proc. 5th Workshop Algorithmic Found. Robot.*, pages 77–94, 2002.

[4] 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.

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

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

[7] 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.

[8] J. Byrne, R. Jarvis, S. Yuta, and A. Zelinsky. Planning paths of complete coverage of an unstructured environment by a mobile robots. In *Internat. Conf. Adv. Robotics*, pages 553–538, 1993.

[9] 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.

[10] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, 1989.

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

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

[13] D. Halperin, L. E. Kavraki, and J.-C. Latombe. Robot algorithms. In M. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 21, pages 21.1–21.21. CRC Press LLC, 1999.

[14] M. Holzer and S. Schwoon. Assembling molecules in Atomix is hard. *Theoret. Comput. Sci.*, 303(3):447–462, 2004.

[15] F. Hüffner, S. Edelkamp, H. Fernau, and R. Niedermeier. Finding optimal solutions to atomix. In *Proc. German Conf. Artif. Intell.*, volume 2174 of *Lect. Notes Comput. Sci.*, pages 229–243. Springer, 2001.

[16] Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Comput. Surv.*, 24(3):219–291, 1992.

[17] 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.

[18] 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.

[19] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring grid polygons online. Technical Report 001, Department of Computer Science I, University of Bonn, December 2005. http://web.informatik.uni-bonn.de/I/publications/ikkl-egpol-05.pdf.

[20] 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.

[21] C. Icking and R. Klein. Competitive strategies for autonomous systems. In H. Bunke, T. Kanade, and H. Noltemeier, editors, *Modelling and Planning for Sensor Based Intelligent Robot Systems*, pages 23–40. World Scientific, Singapore, 1995.

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

[23] T. Kamphans. *Models and Algorithms for Online Exploration and Search.* Dissertation, University of Bonn, 2005.

[24] R. Karp. Reducibility among combinatorical problems of computer computaions. In E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 88–104. Plenum Press, New York, 1972.

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

[26] 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.

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

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

[29] N. S. Rao, S. Kareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, July 1993.

[30] J. T. Schwartz and C. Yap. *Advances in Robotics Vol. I: Algorithmic and geometric aspects of robotics.* Lawrence Erlbaum Associates, 1987.

[31] M. Sharir. Algorithmic motion planning. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 40, pages 733–754. CRC Press LLC, Boca Raton, FL, 1997.

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