



HAL
open science

Ricochet Robots game: complexity analysis Technical Report

Samuel Masseport, Benoit Darties, Rodolphe Giroudeau, Jorick Lartigau

► **To cite this version:**

Samuel Masseport, Benoit Darties, Rodolphe Giroudeau, Jorick Lartigau. Ricochet Robots game: complexity analysis Technical Report. 2019. hal-02191102

HAL Id: hal-02191102

<https://hal.science/hal-02191102v1>

Preprint submitted on 23 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Ricochet Robots game: complexity analysis

Technical Report

Samuel Masseport^{1,2}, Benoît Darties¹, Rodolphe Giroudeau¹, Jorick Lartigau²

¹ LIRMM - University of Montpellier - CNRS - Montpellier, France
² Pickio SAS - Montpellier, France

{samuel.masseport, darties, rgirou}@lirmm.fr
 {samuel.masseport, jorick.lartigau}@pikcio.com

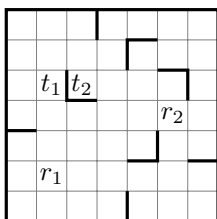
Abstract

This paper investigates the Ricochet Robots game problem from a complexity standpoint. The problem consists in moving robots in a grid game board in horizontal or vertical direction only, to reach specific target tiles. Once a robot starts moving in a direction, it cannot be stopped until being blocked by a wall or another robot. We show that the optimization problem corresponding to this game is Poly- \mathcal{APX} -hard. We also show that the decision problem is PSPACE-complete when we consider an arbitrary number of robots. In such a context, several lower bounds are introduced, exploring some classic complexity hypothesis ($\mathcal{P} \neq \mathcal{NP}$, \mathcal{ETH} , ...).

1 Introduction

Ricochet Robots [2, 7, 8, 9] is a puzzle board game designed by A. Randolph, in which a player must move playing pieces (robots) in an arbitrary grid to a given set of selected locations, with less moves as possible. The game board is a grid containing walls and colored robots. A wall can be placed between two tiles and robots are on the tiles. Robots can move on the grid according to horizontal or vertical directions only. A robot moving in a direction will not stop until it hits a wall or another robot. Each step consists in selecting both a robot and a direction the robot will follow. To solve the puzzle, the player needs to reach a configuration where all target tiles are covered by a robot of the corresponding color. Also, it is often necessary to move robots that serve as guides to stop the movement of another one on an appropriate tile (see Figure 1 for an example). Several robots cannot move at the same time. If a robot has to move, it has to wait for the end of the movement of the previous one. In the original game, the board is a square containing four robots and a specific target tile per robot - four colors being used to distinguish pairs of robots and target tiles -. Hence the player must move each robot to its corresponding target tile.

Ricochet Robots game can be categorized as a sliding game like the PushPush game studied by Demaine [6] or the Atomix game studied by Holzer [12] or Huffner [13]. Icking [14, 15] considers the exploration problem of a grid polygon with or without obstacles inside it. Engels [7] studies the solvability of Ricochet Robots with n uncolored robots and one target tile, and proves that this problem is \mathcal{NP} -hard. Hesterberg [11] studies the parameterized complexity of Ricochet Robots and Atomix. Gebser [8, 9] uses



In this grid r_1 cannot be placed on the target tile t_1 without the help of r_2 . If the robot r_2 reach the target tile t_2 with the two movements \uparrow then \leftarrow and stop to move, then r_1 will never reach t_1 . A solution possible of this instance is to execute there following moves: r_2 : \leftarrow ; r_1 : $\uparrow, \leftarrow, \downarrow, \rightarrow$; r_2 : $\uparrow, \rightarrow, \downarrow$.

Figure 1: Example of an instance of Ricochet Robots. To solve it, the robot r_1 must be placed on the target tile t_1 and r_2 must be on t_2 at the same time.

Problem	Complexity	Reduction from	Proof
GR($k, 1, 1, 1$)	Polynomial	-	Hesterberg [11]
MR($n, 1, n, 1$)	Poly- \mathcal{APX} -hard	MAXIMUM INDEPENDENT SET	Theorem 3
GR($n, 1, 1, 1$)	PSPACE-complet	TOKEN SLIDING (TS)	Theorem 4

Table 1: Problem complexity classification, where n is a variable and k a constant.

Ricochet Robots game as a benchmark for answer set programming while Butko [2] proposes to study how humans try to solve Ricochet Robots.

This article proposes to generalize the problem of Ricochet Robots game in Section 2. In Section 3 we prove that the corresponding optimization problem with n robots and n target tiles is Poly- \mathcal{APX} -hard and the Section 4 improves the result of [7] and show that the problem with n robots is PSPACE-complete. We present future works and perspectives in Section 5.

2 Modelization

The original game of Alex Randolph implies four different colored robots and one colored target tile. In this paper, we extract two problems from this original configuration, respectively a decision problem and an optimization problem.

2.1 Decision problem

We define the decision problem GENERALIZED REACHABILITY (GR) as a generalization of the *Reachability Problem* introduced by Engels and Kamphans [7].

GENERALIZED REACHABILITY (GR)

Input: Given an arbitrary grid polygon P , a set R of n robots r_1, \dots, r_n with a given starting position on the grid, a set T of m target tiles t_1, \dots, t_m and two functions $robotColor(r_i)$ and $tileColor(t_i)$ which return the color of the robot r_i and the color of the target tile t_i .

Output: Can we reach a configuration such that each target tile $t_i \in T$, t_i is covered by a robot $r_j \in R$ and $tileColor(t_i) = robotColor(r_j)$ (i.e. a configuration in which each colored target tile is reached by a robot of the same color)?

We denote by GR(n, c_r, m, c_t) the GENERALIZED REACHABILITY (GR) problem composed by n robots of c_r different colors and m target tiles of c_t different colors.

2.2 Optimization problem

The optimization problem MAXIMUM REACHABILITY (MR) corresponding to GENERALIZED REACHABILITY (GR) problem, is defined as follows:

MAXIMUM REACHABILITY (MR)

Input: Given an instance I of GENERALIZED REACHABILITY (GR) and K a set of tiles such that $\forall t_i \in T$, if t_i is covered by a robot $r_j \in R$ and $tileColor(t_i) = robotColor(r_j)$ then $t_i \in K$ (i.e. K is the set of tiles reached by a robot of the corresponding color).

Output: Maximize $|K|$.

We denote by MR(n, c_r, m, c_t) the MAXIMUM REACHABILITY (MR) problem with n robots of c_r different colors and m target tiles of c_t different colors.

2.3 Problems classification

Currently, the complexity of GENERALIZED REACHABILITY (GR) and MAXIMUM REACHABILITY (MR) is given by Table 1.

Theorem 1. *In terms of complexity, $\forall n > 1$, the problem GR($n, 1, 1, 1$) is at least as hard as GR($n-1, 1, 1, 1$).*

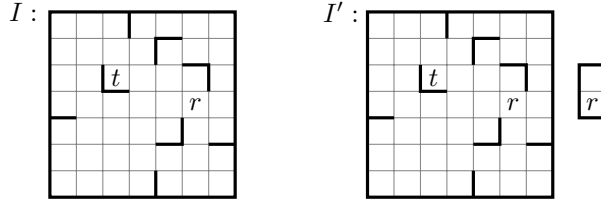


Figure 2: Example of the instance I' of $\text{GR}(2, 1, 1, 1)$ obtained from the instance I of $\text{GR}(1, 1, 1, 1)$ by adding a robot in an isolated area.

Proof. For all instances I of $\text{GR}(n-1, 1, 1, 1)$, it is possible to create an instance I' in polynomial-time of $\text{GR}(n, 1, 1, 1)$ by adding a robot in an isolated area (see example Figure 2). Then it exists a reduction from $\text{GR}(n-1, 1, 1, 1)$ to $\text{GR}(n, 1, 1, 1)$ ($\forall n > 1$). \square

It seems that the problems are more difficult when we add target tiles, robot colors or target tile colors but we do not have reductions to prove it. However, adding target tiles, robot colors or target tile colors to a positive instance can return a negative instance whereas remove one of them cannot transform a positive instance to a negative.

3 Maximum reachability problem with n robots

This section is devoted to showing some new complexity results according several complexity hypothesis.

First, let us recall the definition of strict reduction and \mathcal{S} -reduction of Crescenzi [5].

Let us consider two **NPO** problems Π and Π' . Moreover, let $\Pi'' \in \{\Pi, \Pi'\}$, we denote by $OPT_{\Pi''}$ the value of an optimal solution of Π'' and by $m_{\Pi''}(x'', y'')$ the cost of a solution y'' of an instance x'' of Π'' , $m_{\Pi''}(x'', y'')$ is also the metric used to determine which solution is considered optimal.

Given an instance x of a problem Π and a feasible solution y of x , we define the performance ratio of y with respect to x as:

$$R_{\Pi} = \max \left\{ \frac{m_{\Pi}(x, y)}{OPT(x)}, \frac{OPT(x)}{m_{\Pi}(x, y)} \right\}$$

Strict reduction consists of polynomial-time computable functions f and g such that, for each instance x of Π , $f(x)$ returns an instance of Π' and for each feasible solution y' of $f(x)$, $g(y')$ returns a feasible solution of x . A reduction (f, g) from Π to Π' is said to be a strict reduction if, for any instance x of Π and for any y' feasible solution of Π' , the following holds:

$$R_{\Pi}(x, g(y')) \leq R_{\Pi'}(f(x), y').$$

There is a \mathcal{S} -reduction from Π to Π' if:

1. For any instance x of Π , $OPT_{\Pi'}(f(x)) = OPT_{\Pi}(x)$.
2. For any instance x of Π and for any y' feasible solution of Π' , $m_{\Pi}(x, g(y')) = m_{\Pi'}(f(x), y')$.

\mathcal{S} -reduction is a special case of strict reduction, then a \mathcal{S} -reduction implies a strict reduction. Note that \mathcal{S} -reduction implies AP-reduction, then preserves the membership in Log-APX and Poly-APX classes.

In the following, we present a \mathcal{S} -reduction from the classical MAXIMUM INDEPENDENT SET problem to $\text{MR}(n, 1, n, 1)$.

MAXIMUM INDEPENDENT SET

Input: Graph $G = (V, E)$.

Output: Find an independent set of vertices $V' \subseteq V$, for V' of the maximal cardinality *i.e.* a largest set $V' \subseteq V$ such that no two vertices in V' are joined by an edge in E .

This problem is proved Poly-APX -complete by Bazgan [1]. Let us consider the function $N(v_i)$ which returns the set of neighbour vertices of the vertex v_i .

We define some polygons with particular properties. The first one, called v_i -CHOICE polygon (Figure 3), is a gadget in which robots have their initial location. This gadget admits two outputs and a robot needs to choose one of them (*i.e.* when a robot reaches an output, it cannot take the other one because it will be stuck in tiles tagged by \times).

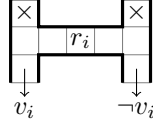


Figure 3: v_i -CHOICE polygon, tiles tagged by \times corresponding to tiles where robot is stuck if it tries to come from an output to reach another one.

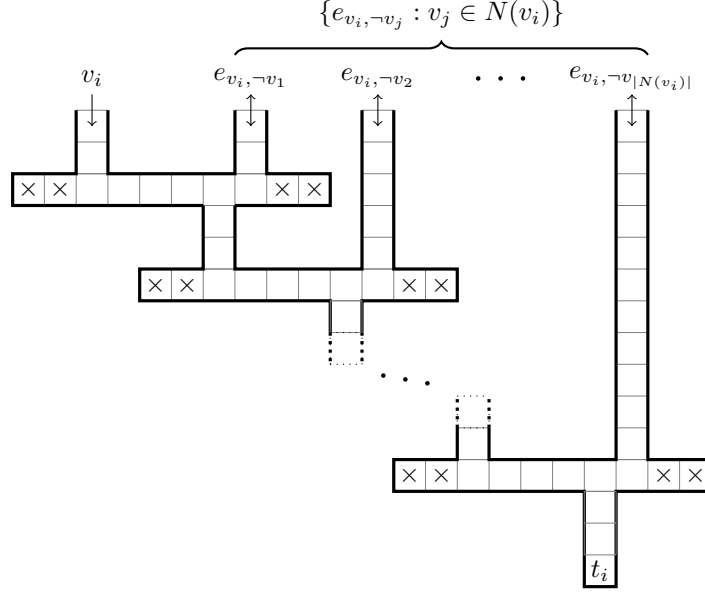


Figure 4: A v_i -VERTEX polygon. Only the robot which comes from the v_i input can reach the target tile t_i and it needs the help of robots which come from all $e_{v_i, -v_j}$ inputs, otherwise it will be stuck in tiles tagged by \times .

Lemma 1. *Each robot r_i can reach at most one v_i -CHOICE polygon output.*

Proof. When r_i outputs of its v_i -CHOICE polygon, it cannot comeback to choose the other one because it will be stuck in tiles tagged by \times . \square

The second gadget polygon called v_i -VERTEX polygon Figure 4 is created in order to verify the absence of all neighbors of v_i in the final solution when we try to add v_i to the solution. In this gadget, only the robot coming from v_i can reach the target tile t_i if and only if $|N(v_i)|$ others inputs have been reached by a robot (otherwise it will be stuck on the tiles tagged by \times). Note that a robot which reaches an $e_{v_i, -v_j}$ input can help the robot which comes from v_i before to go out of the gadget in the same way where it comes from. A robot which inputs a v_i -VERTEX polygon cannot reach another input than its to get output.

Lemma 2. *The robot r_i can reach the target tile t_i of the v_i -VERTEX polygon if and only if $\forall v_j \in N(v_i)$, the input $e_{v_i, -v_j}$ is reached by a robot.*

Proof. If $\forall v_j \in N(v_i)$, the input $e_{v_i, -v_j}$ is reached by a robot, then r_i can rely on them one after another to reach the target tile t_i . Considering an input $e_{v_i, -v_j}$ which is not reached by a robot, then r_i will be stuck in tiles tagged by \times and will not be able to reach the target tile t_i . \square

The next gadget polygon is the $\neg v_i$ -ROUTER polygon. In this one, the robot which input in $\neg v_i$ can reach any output (while traveling through counterclockwise). Another particularity, when a robot comes from an output, it can reach any others (but it cannot reach the input $\neg v_i$).

Remark 1. In the following construction, a robot r_i can reach the target tile t_i if and only if it crosses the v_i -VERTEX polygon. To cross v_i -VERTEX polygon a robot r_i needs to be helped by all of its neighbors (in the corresponding graph G). A robot which helps another one to reach a target tile cannot reach one.

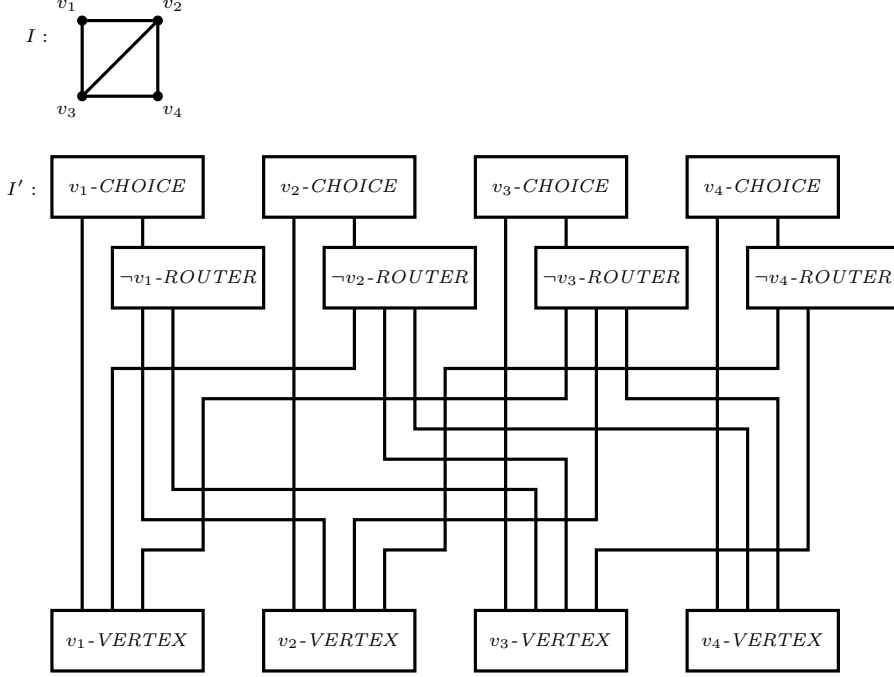


Figure 6: Example of the instance I' of $\text{MR}(n, 1, n, 1)$ obtained by [Construction 1](#) from the instance I of [MAXIMUM INDEPENDENT SET](#).

reach the target tile t_i if and only if all inputs of the v_i -*VERTEX* are reached by a robot, in other words, if $\forall v_j \in N(v_i)$, the corresponding robot r_j has reached the $\neg v_j$ -*ROUTER* polygon. Thus, r_i reaches the target tile t_i and according to [Lemma 1](#) its neighbors cannot reach it. If $\exists r_l$ a robot which can reach a target tile such that $v_l \notin V'$. Thus, $\exists v_l \in V$ such that none of its neighbors are in V' and $v_l \notin V'$ then $\{v_l\} \cup V'$ is an admissible solution of size $k + 1$ then V' is not a maximal solution. Thus, if I admits a maximal solution of size k , I' admits a maximal solution of size k .

- Reciprocally, we suppose that I' possesses a maximal solution of size $|K| = k$, then we prove that I admits a maximal solution of size k :

If I' admits a maximal solution of size k , then k robots have reached a target tiles. According to [Lemma 2](#), r_i can reach the target tile t_i of v_i -*VERTEX* polygon if and only if $\forall v_j \in N(v_i)$, the input $e_{v_i, \neg v_j}$ is reached. According to [Construction 1](#), $\forall v_j \in N(v_i)$, the input $e_{v_i, \neg v_j}$ can be reached if and only if $\neg v_j$ -*ROUTER* polygon is reached and then if r_j has chosen the $\neg v_j$ output of the v_j -*CHOICE* polygon. Then the k target tiles reached by a robot correspond to a set V' of k vertices such that any of their neighbors are not in the set (*i.e.* an independent set of size k). If $\exists v_l \in V \setminus V'$ a vertex such that $\{v_l\} \cup V'$ is an independent set, then $\exists t_l \in T \setminus K$ which can be reached by a robot r_l because none of its neighbors are in K and then they can help it to cross the v_l -*VERTEX* polygon and reach the target tile t_l . Then K is not a maximal solution (contradiction). Therefore if I' admits a maximal solution of size k , I admits a maximal solution of size k .

All maximum solutions are a maximal solutions then I admits a maximum solution of size $|V'| = k$ if and only if I' admits a maximal solution of size $|K| = k$. From previous arguments, there is a polynomial-time reduction from [MAXIMUM INDEPENDENT SET](#) to $\text{MR}(n, 1, n, 1)$. \square

Theorem 3. $\text{MR}(n, 1, n, 1)$ is *Poly-APX-hard*.

Proof. To prove this we show that exists a \mathcal{S} -reduction from [MAXIMUM INDEPENDENT SET](#) problem to $\text{MR}(n, 1, n, 1)$. Let Π the [MAXIMUM INDEPENDENT SET](#) problem, Π' the $\text{MR}(n, 1, n, 1)$ problem, I an instance of Π , I' an instance of Π' , V' a solution of I and K a solution of I' . We have:

- A function $f(I)$ which constructs I' in polynomial-time ([Construction 1](#)).
- A function $g(K)$ which constructs V' in polynomial-time ($V' = \{v_i : t_i \in K\}$).

And:

1. For any instance I of Π , $OPT_{\Pi}(I) = OPT_{\Pi'}(f(I))$ (Theorem 2).
2. For any instance I of Π and for any K feasible solution of Π' , $m_{\Pi}(I, g(K)) = m_{\Pi'}(f(I), K)$ (Theorem 2).

The previous reduction being an \mathcal{S} -reduction from MAXIMUM INDEPENDENT SET to MR($n, 1, n, 1$) and considering MAXIMUM INDEPENDENT SET Poly- \mathcal{APX} -complete (Bazgan [1]) this reduction implies that MR($n, 1, n, 1$) is Poly- \mathcal{APX} -hard. \square

Let us define k -RICOCHET for some corollaries.

k -RICOCHET (k -GENERALIZED REACHABILITY (GR))

Input: Given an instance I of GENERALIZED REACHABILITY (GR) and K a set of tiles such that $\forall t_i \in T$, if t_i is covered by a robot $r_j \in R$ and $tileColor(t_i) = robotColor(r_j)$ then $t_i \in K$ (i.e. K is the set of tiles reached by a robot of the corresponding color).

Question: Can we reach a configuration such that $|M| \geq k$ (i.e. a configuration in which at least k colored target tiles are reached by a robot of the corresponding color)?

Parameter: k

The previous reduction can also be seen as an FPT-reduction from INDEPENDENT SET parameterized by the standard parameter to k -RICOCHET parameterized by the standard parameter and then implies that k -RICOCHET is $W[1]$ -hard.

We know that for all $\epsilon > 0$, approximating INDEPENDENT SET to within $n^{1-\epsilon}$ is NP-hard [10, 22].

INDEPENDENT SET not having a $n^{o(k)}$ -time algorithm [3] implies that MR($n, 1, n, 1$) does not have a $n^{o(k)}$ -time algorithm (unless $\mathcal{W}[1] = \mathcal{FPT}$).

Construction 1 also implies subexponential lower bounds for our problems based on the widely believed complexity-theoretic hypothesis known as the ‘‘Exponential-Time Hypothesis¹’’ (ETH, see [16, 19, 21]).

The following results are straightforward since by [4], we know that INDEPENDENT SET does not have $f(k)n^{o(k)}$ (resp. $n^{o(k)}$) time algorithm, unless \mathcal{ETH} fails (unless $\mathcal{W}[1] = \mathcal{FPT}$, [3]).

Corollary 1.

1. It is NP-hard to approximate within $n^{1-\epsilon}$ for MR($n, 1, n, 1$).
2. There exists an FPT-reduction from INDEPENDENT SET parameterized by the standard parameter to k -ricochet parameter by the standard parameter.
3. Assuming \mathcal{ETH} , k -RICOCHET cannot be solved in $f(k)n^{o(k)}$ time algorithm, and,
4. Assuming $\mathcal{W}[1] \neq \mathcal{FPT}$, k -RICOCHET cannot be solved in $n^{o(k)}$ time algorithm.

4 General reachability problem with n robots

In this section we show that GR($n, 1, 1, 1$) is PSPACE-complete. We consider the problem of TOKEN SLIDING (TS) proved PSPACE-complete by [17].

Kamiński et al. [18] define reconfiguration problems as follows: ‘‘Given two feasible solutions x, y of I , the aim is to find a reconfiguration sequences s_1, \dots, s_k such that $s_1 = x$, $s_k = y$, and for each s_i (for $1 < i < k$) is a feasible solution of I , and the transition between s_i and s_{i+1} is allowed by the reconfiguration rule’’.

TOKEN SLIDING (TS) (also called independent set reconfigurability problems) can be defined as follows:

TOKEN SLIDING (TS) (Token Sliding)

Input: A graph $G = (V, E)$ and two independent sets A and B in G .

Output: Is it possible to reconfigure A into B via a sequence S of independent sets (numbered by $s_1, \dots, s_{|S|}$) such that each of which results from the previous one by withdraw a vertex and adding one of its neighbours?

Let us consider $n = |V|$, $k = |A| = |B|$ and the function $N(v_i)$ which returns the set of neighbour vertices of the vertex v_i .

We define some polygons with particular properties. The first, denoted $\neg v_i$ -ROUTER polygon (Figure 7, similar to Figure 5 with more inputs), is a gadget in which an incoming robot in $\neg v_i$ can reach

¹The ETH states that there is a constant $c > 1$ such that n -variable 3-SAT cannot be solved in $O(c^n)$ time.

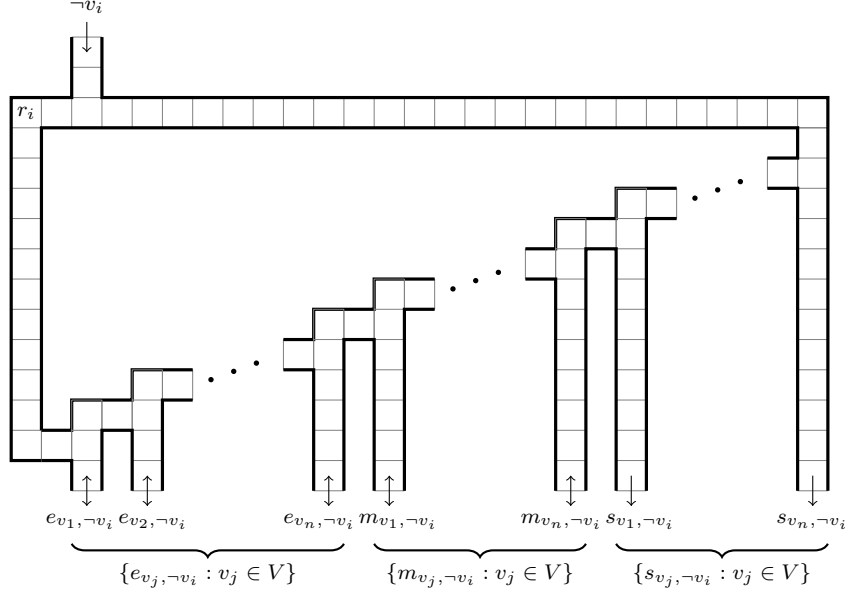


Figure 7: The $\neg v_i$ -ROUTER polygon. The robot which comes from the $\neg v_i$ input can reach any output while traveling through counterclockwise. The robot r_i start in the tile tagged by r_i if the corresponding vertex $v_i \notin A$.

any output (while traveling through counterclockwise). Another particularity, when a robot comes from an output, it can reach any others (but it cannot reach the input $\neg v_i$).

The second gadget polygon is denoted v_i -ROUTER polygon (Figure 8). It has the same properties as the $\neg v_i$ -ROUTER polygon (Figure 7) but it has fewer outputs.

Lemma 3. *A robot in a $\neg v_i$ -ROUTER polygon (see Figure 7) (resp. v_i -ROUTER polygon Figure 8) can reach any output (no matter where it comes from) but it cannot reach the input $\neg v_i$ (resp. v_i).*

The next gadget is the v_i -DISPATCHER polygon. In this one, the robot which comes in through the input d_{v_i} can reach only one output between the n available outputs. Figure 9 illustrates two examples of v_i -DISPATCHER polygon (with $n = 3$ and $n = 4$ outputs). Note that the number of outputs of the v_i -DISPATCHER polygon is totally scalable ($\forall n \geq 1$).

Lemma 4. *Each robot r_i can reach at most one v_i -DISPATCHER polygon output.*

Proof. When r_i outputs of an v_i -DISPATCHER polygon, it cannot comeback to choose another one because it will be stuck in tiles tagged by \times . \square

The following polygon denoted v_i -VERTEX polygon, described by Figure 4 is created in order to verify the absence of all neighbors of v_i in the current independent set when we try to slide a token on v_i . In this gadget, only the robot coming from v_i can reach the output v'_i if and only if the $|N(v_i)|$ others inputs have been reached by a robot (otherwise it will be stuck on the tiles tagged by \times).

The v_i -VERTEX polygon is the same as the Figure 10 but with an output v'_i rather than a target tile. Obviously the Lemma 2 is applicable to this one.

Remark 1. A robot which reaches an $e_{v_i, \neg v_j}$ input in the v_i -VERTEX polygon can help the robot which comes from v_i before to go out of the gadget in the same way where it comes from. A robot which inputs a v_i -VERTEX polygon cannot reach another input than its to get output (if it try, it will be stuck on the tiles tagged by \times).

The next gadget polygon is the v_i -MOVING polygon Figure 11. This gadget is used to synchronize the position of all other robots before to start the sliding of the vertex v_i .

Lemma 5. *In the v_i -MOVING polygon Figure 11, the robot which input in Mv_i can reach the output d_{v_i} if and only if $\forall j$ such that $v_j \in V \setminus \{v_i\}$, the input m_{v_i, v_j} or $m_{\neg v_i, v_j}$ are reached by a robot.*

Proof. The robot coming from the input Mv_i must reach the bottom of the gadget to reach the output. For that, at each floor, this robot need to lean on another one to reach the floor below (else it is stuck on tiles tagged by \times). \square

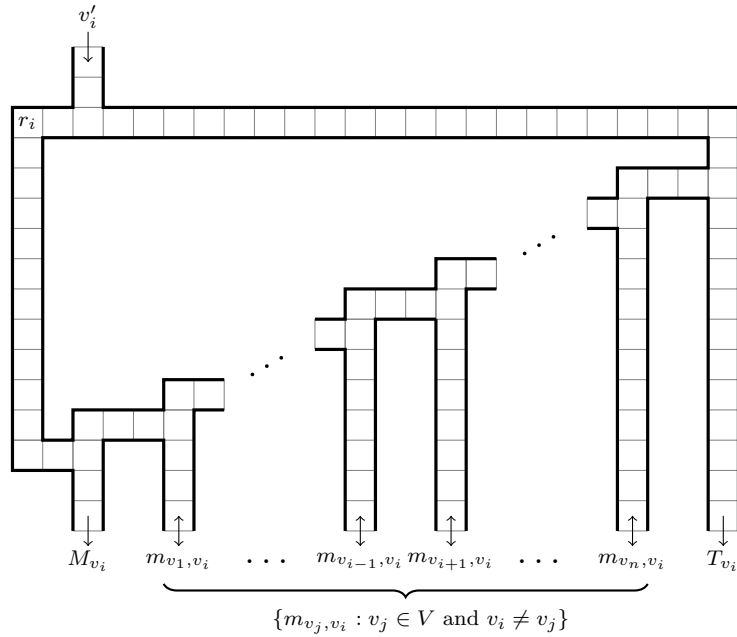


Figure 8: A v_i -ROUTER polygon. The robot which comes from the v'_i input can reach any output while traveling through counterclockwise. The robot r_i start in the tile tagged by r_i if the corresponding vertex $v_i \in A$.

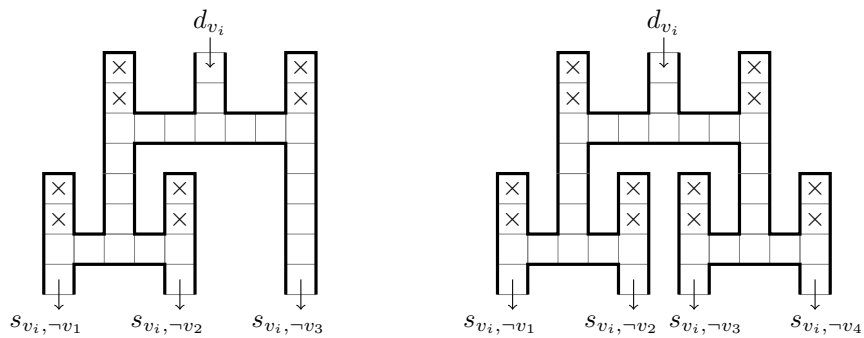


Figure 9: Example of a v_i -DISPATCHER polygons with $n = 3$ outputs on the left and $n = 4$ on the right. Tiles tagged by \times correspond to tiles where robots are stuck if they try to come from an output to reach another one. Obviously the output number of v_i -DISPATCHER polygons is totally scalable.

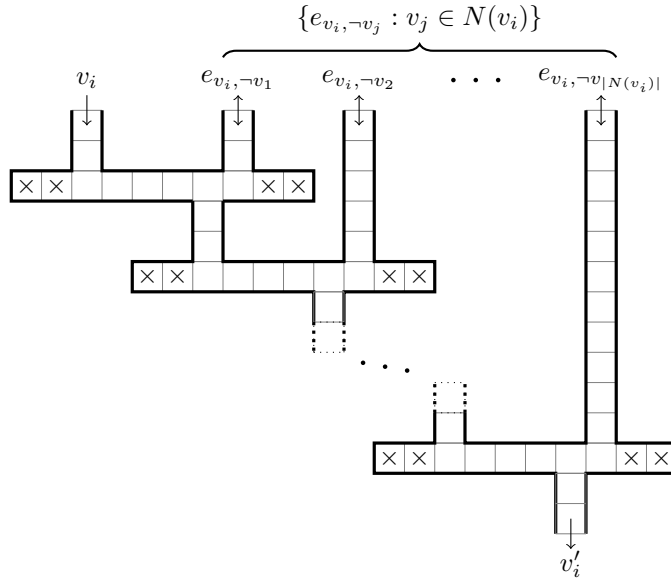


Figure 10: A v_i -*VERTEX* polygon. Only the robot which comes from the v_i input can reach the output v'_i and it needs the help of robots which come from all $e_{v_i, \neg v_j}$ inputs, otherwise it will be stuck in tiles tagged by \times .

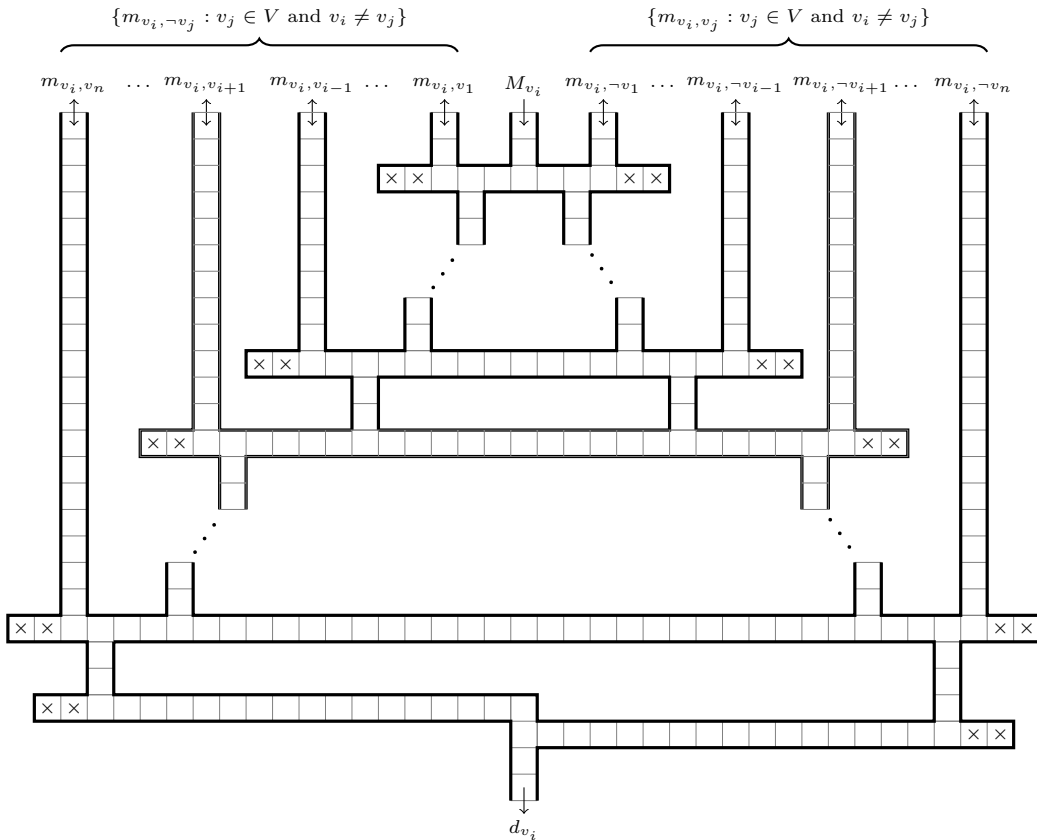


Figure 11: The v_i -*MOVING* polygon. A robot r_i which comes in from the input M_{v_i} can reach the output d_{v_i} if and only if at each floor another robot r_j help it from the input m_{v_i, v_j} or $m_{v_i, \neg v_j}$. Otherwise it is stuck on tiles tagged by \times . By the future construction, it is impossible to have a robot which come from the input m_{v_i, v_j} and another from the input $m_{v_i, \neg v_j}$ at the same time.

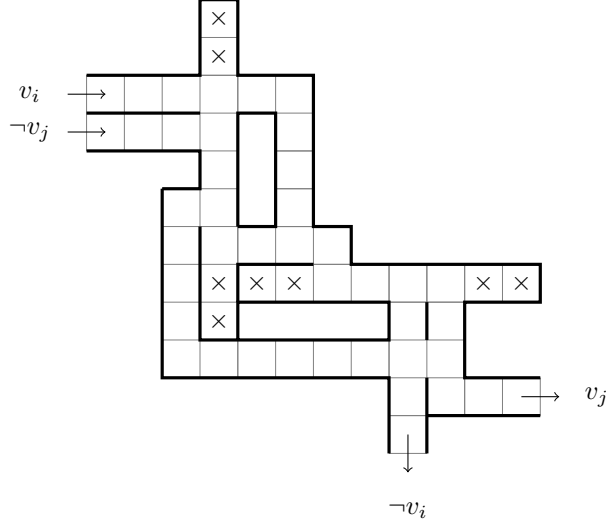


Figure 12: The $(v_i, \neg v_j)$ -SLIDING polygon. Considering a robot r_i coming from the input v_i and another one r_j coming from $\neg v_j$, according to Lemma 6, r_i can only reach the output $\neg v_i$ and r_j can only reach the output v_j .

Remark 2. A robot which reaches an m_{v_i, v_j} input or an $m_{v_i, \neg v_j}$ in the v_i -MOVING polygon can help the robot which comes from M_{v_i} before to go out of the gadget in the same way where it comes from. By the future construction it is impossible to have a robot in m_{v_i, v_j} and in $m_{v_i, \neg v_j}$ at the same time, then the robots cannot go out by another way than they come from (excepted the robot coming from M_{v_i} which can reach the output d_{v_i}).

The $(v_i, \neg v_j)$ -SLIDING polygon Figure 12 is used to simulate the sliding of a token from the vertex v_i to the vertex v_j .

Lemma 6. To reach the outputs of the $(v_i, \neg v_j)$ -SLIDING polygon (see Figure 12), considering the robot r_i coming from the input v_i and another robot r_j coming from the input $\neg v_j$. The robot r_i can only reach the output $\neg v_i$ and the robot coming from $\neg v_j$ can only reach the v_j output.

Proof. If a robot is alone in the gadget, it is stuck on tiles tagged by \times . Considering a robot r_i comes in from the input v_i and a robot r_j comes in from the input $\neg v_j$. They need to make the following movements to reach the output (otherwise they will be stuck on the gadget or one of them will not go out): $r_i : \rightarrow, \downarrow, \leftarrow$; $r_j : \rightarrow, \downarrow, \leftarrow, \downarrow, \rightarrow, \uparrow$; $r_i : \rightarrow, \downarrow$; $r_j : \downarrow, \rightarrow$. With these movements r_i reaches the output $\neg v_i$ and r_j reaches the output v_j , they cannot exchange their output. \square

The last gadget polygon is the TARGET polygon (Figure 13) which contains the target tile.

Remark 3. To reach the target tile, k robots need to reach the TARGET polygon. A robot which reaches the TARGET polygon cannot leave it.

Remark 4. In the following construction, $\forall v_i \in V$ there is a robot r_i . At each step, for each robot r_i , if v_i is in the current independent set then r_i is in gadget v_i -ROUTER, else it is in the $\neg v_i$ -ROUTER polygon. To slide a token from v_i to v_j , the corresponding robot r_i need to reach the v_i -MOVING polygon to verify the presence of all other (to prevent more robots from moving at the same time). Then r_i reaches the v_i -DISPATCHER polygon before to reach the corresponding $(v_i, \neg v_j)$ -SLIDING polygon and r_j joins it. After this one, r_i go to the $\neg v_i$ -ROUTER polygon and r_j reaches the v_i -VERTEX polygon to verify if all neighbours of v_j are not in the current independent set (*i.e.* if the new solution is an independent set). If the new set is always an independent set, v_j reaches the v_j -ROUTER. At each step we have an independent set of size k and when all robots corresponding to all vertices of B are in there v_i -ROUTER they can reach the TARGET polygon and reach the target tile.

Construction 2. Let I be an instance of TOKEN SLIDING (TS) with a graph $G = (V, E)$ with $|V| = n$ and two independent sets of k vertices A and B . In order to construct an instance I' of $\text{GR}(n, 1, 1, 1)$ we consider the following gadgets:

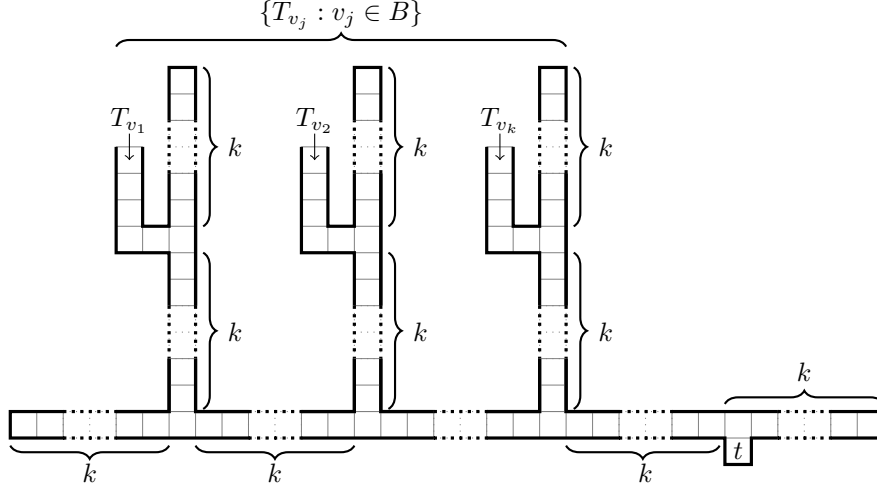


Figure 13: The *TARGET* polygon. It contains the unique target tile t which is reachable if and only if there are k robots in the polygon.

- $\forall v_i \in V$ we consider the robot $r_i \in R$.
- All n robots $r_i \in R$ have the same color.
- $\forall r_i \in R$ (and then $\forall v_i \in V$) we construct a v_i -ROUTER polygon, a $\neg v_i$ -ROUTER polygon, a v_i -MOVING polygon, a v_i -VERTEX polygon with $|N(v_i)|$ inputs of type $e_{v_i, \neg v_j}$ and a v_i -DISPATCHER polygon with $|N(v_i)|$ outputs.
- $\forall (v_i, v_j) \in E$ we construct a $(v_i, \neg v_j)$ -SLIDING polygon and a $(v_j, \neg v_i)$ -SLIDING polygon.
- $\forall r_i \in R$, if $v_i \in A$, r_i starts in the corresponding v_i -ROUTER polygon (see Figure 8) else it starts in the $\neg v_i$ -ROUTER polygon (see Figure 7).
- The unique target tile t is contained by the *TARGET* polygon with $k = |A| = |B|$.

All gadget polygons are connected as follows:

- **Outputs connections of the $\neg v_i$ -ROUTER polygons:** for each edge $(v_j, v_i) \in E$, the output $e_{v_j, \neg v_i}$ (resp. $s_{v_j, \neg v_i}$) is connected to the corresponding input $e_{v_j, \neg v_i}$ (resp. $\neg v_i$) of the v_j -ROUTER (resp. $(v_j, \neg v_i)$ -SLIDING) polygon. If $(v_j, v_i) \notin E$ then the outputs $e_{v_j, \neg v_i}$ and $s_{v_j, \neg v_i}$ are closed by a wall. Each $m_{v_j, \neg v_i}$ output is connected to the input $m_{v_j, \neg v_i}$ of the corresponding v_j -MOVING polygon.
- **Outputs connections of the v_i -VERTEX polygons:** v_i' output is connected to the v_i' input of the corresponding v_i -ROUTER polygons.
- **Outputs connections of the v_i -ROUTER polygons:** the M_{v_i} output is connected to the M_{v_i} input of the corresponding v_i -MOVING polygon. Each m_{v_j, v_i} output is connected to the m_{v_j, v_i} input of the corresponding v_j -MOVING polygon. If $v_i \in B$ the output T_{v_i} is connected to the T_{v_i} input of the *TARGET* polygon else this output is closed by a wall.
- **Outputs connections of the v_i -MOVING polygons:** the output d_{v_i} is connected to the input d_{v_i} of the corresponding v_i -DISPATCHER.
- **Outputs connections of the v_i -DISPATCHER polygons:** Each output $s_{v_i, \neg v_j}$ is connected to the input v_i of the corresponding $(v_i, \neg v_j)$ -SLIDING.
- **Outputs connections of the $(v_i, \neg v_j)$ -SLIDING polygons:** the output $\neg v_i$ (respectively v_j) is connected to the input $\neg v_i$ (respectively v_j) of the corresponding $\neg v_i$ -ROUTER polygon (respectively v_j -ROUTER polygon).

The Figure 14 is an general example of the Construction 2. To give an intuition to the reader, we can define level of the construction in the Figure 14 as follows:

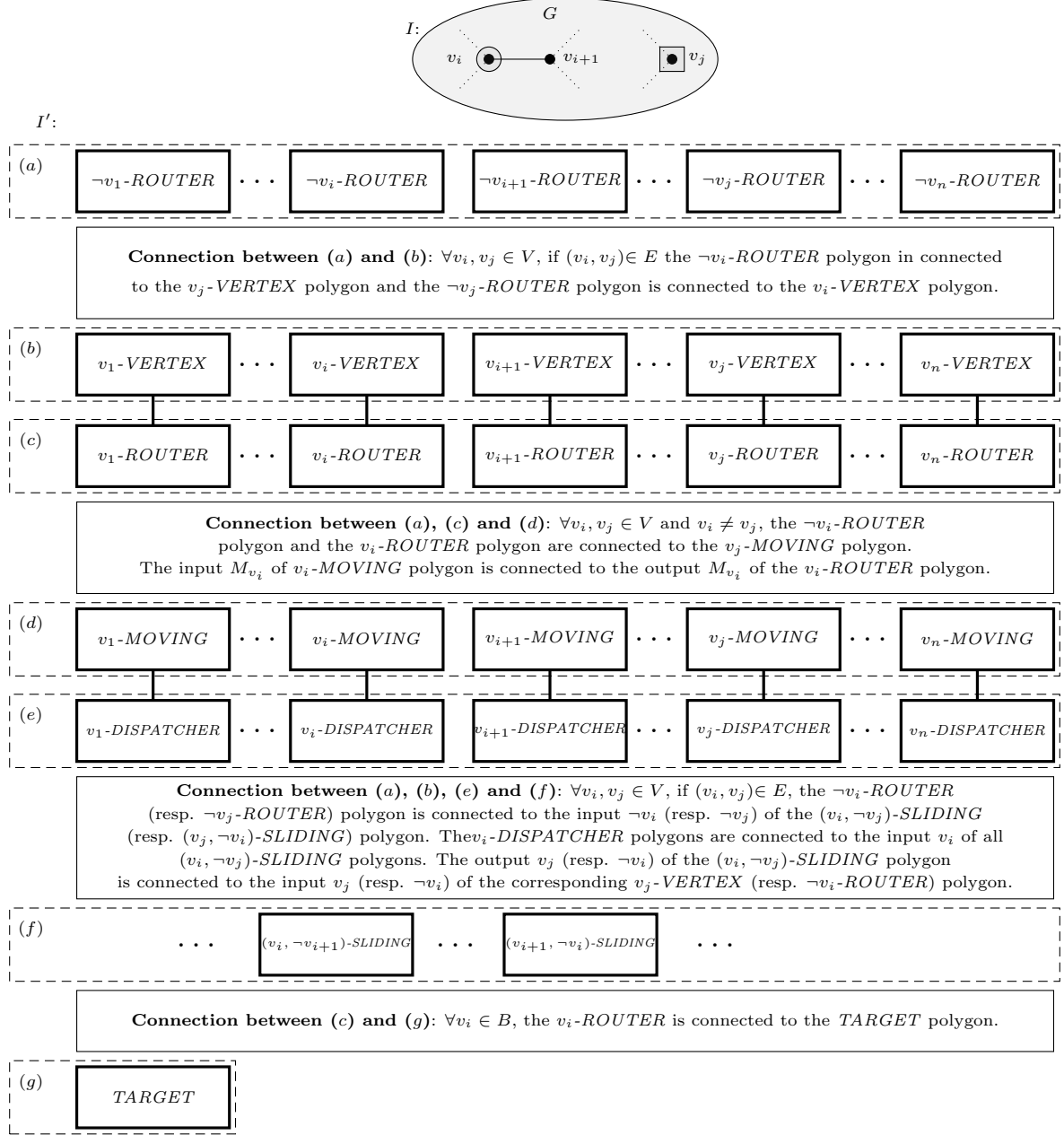


Figure 14: Example of an instance I' obtained by Construction 2 from the instance I of TOKEN SLIDING (TS). In I , vertices surrounded by circles are vertices in the current independent set (initially A) and vertices surrounded by squares are vertices of the final independent set B .

- (a): if a robot is in an $\neg v_i$ -ROUTER polygon, then v_i is not in the current independent set (*i.e.* the level (a) corresponds to the set of vertices which are not in the current independent set).
- (b): is the level to test if the sliding is legal (*i.e.* if the new set is always an independent set).
- (c): contrary to the level (a), the level (c) corresponds to the current independent set.
- (d): verify that more than one token cannot move at the same time. If a robot cross this level, no other can cross it while the sliding is not ended and legal.
- (e): when a robot r_i cross the level (d), the robot can choose where to move the token and take the corresponding polygon (*e.g.* if the token move from v_i to v_j , v_i cross the v_i -MOVING polygon and choose to reach the output corresponding to the $(v_i, \neg v_j)$ -SLIDING polygon in the v_i -DISPATCHER).
- (f): is used to execute the sliding of the token, after it, v_i which was in the current independent set go to the $\neg v_i$ -ROUTER polygon and v_j which was not in the current independent set go to the level (b) to verify if the new set is always an independent set and if it is, it reaches the v_j -ROUTER.

Remark 5. In an instance obtained by [Construction 2](#), to leave a v_i -ROUTER polygon and access to a $\neg v_i$ -ROUTER polygon, the robot r_i need to take the output M_{v_i} to access to the v_i -MOVING polygon. According to [Lemma 5](#) and [Lemma 8](#), r_i need the help of others robots to reach the output d_{v_i} and then the v_i -DISPATCHER polygon. After r_i reaches an $s_{v_i, \neg v_j}$ output and come in the corresponding $(v_i, \neg v_j)$ -SLIDING polygon by the input v_i . According to [Lemma 6](#), r_i need to be helped by another robot r_j which comes from the $\neg v_j$ input to reach the $\neg v_i$ output and then reach the $\neg v_i$ -ROUTER polygon.

Remark 6. In an instance obtained by [Construction 2](#), to leave a v_j -ROUTER polygon and access to a $\neg v_j$ -ROUTER polygon, the robot r_j need to take an output $s_{v_i, \neg v_j}$ to access to the $(v_i, \neg v_j)$ -SLIDING polygon by the input $\neg v_j$. According to [Lemma 6](#), a robot r_i need to reach the same gadget to help r_j to accesses to the output v_j and reach the corresponding input of the v_j -VERTEX polygon. According to [Lemma 2](#) and to [Construction 2](#), for each $v_k \in N(v_j)$, the corresponding robot r_k must help r_j to cross the v_j -VERTEX polygon and reach the v_j -ROUTER polygon.

Remark 7. In [Remark 5](#), the robot r_i comes from the v_i -ROUTER polygon and go to the $\neg v_i$ -ROUTER while r_j do the opposite way (from $\neg v_j$ -ROUTER to v_j -ROUTER). According to [Remark 5](#), in an instance obtained by [Construction 2](#), a robot r_i can leave the v_i -ROUTER polygon to go to the $\neg v_i$ -ROUTER polygon if and only if another robot r_j leave the $\neg v_j$ -ROUTER polygon to go to the v_j -ROUTER polygon.

Lemma 7. *In an instance obtained by [Construction 2](#), the v_i -ROUTER polygon and the $\neg v_i$ -ROUTER polygon can be reachable by the corresponding robot r_i but not by others.*

Proof. Let us consider the robot r_i in the v_i -ROUTER polygon. If r_i try to reach the $\neg v_j$ -ROUTER polygon, it need to reach the output $\neg v_j$ of an $(v_j, \neg v_k)$ -SLIDING polygon According to [Remark 1](#), [Remark 2](#) and [Lemma 6](#) when the robot r_i try to move from the v_i -ROUTER polygon to the $\neg v_i$ -ROUTER polygon or vice versa, the robot r_i cannot reach another gadget polygon than those described in the [Remark 5](#) (excepted the TARGET polygon). \square

Lemma 8. *In an instance obtained by [Construction 2](#), a robot r_i that comes from the input M_{v_i} in a v_i -MOVING polygon can reach the d_{v_i} output if and only if $\forall v_j \in V \setminus \{v_i\}$, the robot r_j can reach the v_j -ROUTER polygon or the $\neg v_j$ -ROUTER polygon.*

Proof. According to [Lemma 5](#), the output d_{v_i} can be reach if and only if there is a robot to help it at each level. By construction, each input m_{v_i, v_j} is connected with an output of the v_j -ROUTER polygon and the opposite input $m_{v_i, \neg v_j}$ is connected with an output of the $\neg v_j$ -ROUTER polygon. Then to reach the output d_{v_i} , $\forall v_j \in V \setminus \{v_i\}$, the robot r_j can reach the v_j -ROUTER polygon or the $\neg v_j$ -ROUTER polygon. \square

Lemma 9. *In an instance obtained by [Construction 2](#), when a robot r_i leave the v_i -ROUTER polygon and cross the v_i -MOVING polygon, no other robot r_k can cross a v_k -MOVING polygon while r_i has not reach the $\neg v_i$ -ROUTER polygon and then while another robot r_j leave the $\neg v_j$ -ROUTER polygon to reach the v_j -ROUTER polygon (according to [Remark 7](#)).*

Proof. If a robot r_i goes on the v_i -MOVING polygon and another robot r_j goes in the v_j -MOVING polygon, according to Lemma 5 and to Lemma 8 r_i (resp. r_j) cannot reach the output d_{v_i} (resp. d_{v_j}) because it need the help of the robot r_j (resp. r_i). If r_j help r_i to reach the output d_{v_i} of the v_i -MOVING polygon before to try to cross the v_j -MOVING polygon it will be stuck in it as long as r_i has not reach the $\neg v_i$ -ROUTER polygon. \square

Theorem 4. $\text{GR}(n, 1, 1, 1)$ is PSPACE-complete.

Proof. The argument to say that the problem is in PSPACE is close to the argument given by Hüffner [13] for the ATOMIX problem. Given an instance of $\text{GR}(n, 1, 1, 1)$ with an arbitrary grid polygon P consists of $|C|$ tiles, a non-deterministic Turing machine can solve it by repeatedly applying a move up to reach a final configuration (i.e. a configuration where a robot have reached the target tile). The number of configurations is limited to $\binom{|C|}{n}$ and then the machine can announce that there is no solution after having applied more moves without finding a solution. Since a configuration of $\text{GR}(n, 1, 1, 1)$ is encodable in polynomial space, $\text{GR}(n, 1, 1, 1)$ is in NPSPACE and then according to Savitch theorem [20], $\text{GR}(n, 1, 1, 1)$ is in PSPACE.

Now, we have to prove that there is a polynomial-time reduction TOKEN SLIDING (TS) to $\text{GR}(n, 1, 1, 1)$. Considering an instance I of TOKEN SLIDING (TS) and the corresponding instance I' of $\text{GR}(n, 1, 1, 1)$ obtained by Construction 2:

- Assume that there exists a positive solution for the instance I of TOKEN SLIDING (TS) then we construct a positive solution for the instance I' of $\text{GR}(n, 1, 1, 1)$ in the following way:

Let S be a solution of I . Then S is a sequence of independent sets (numbered by $s_1, \dots, s_{|S|}$) such that s_{l+1} is obtained by an exchange of vertex in s_l with one of its neighbors in G and where $s_1 = A$ and $s_{|S|} = B$. By construction $\forall v_i \in V$, if $v_i \in A$, the corresponding robot r_i starts in the v_i -ROUTER polygon else it starts in the $\neg v_i$ -ROUTER polygon. This starting position corresponding to the configuration A . Considering two independent sets $s_l \in S$ and $s_{l+1} \in S$, where s_{l+1} is the configuration obtained by the sliding of a token along the edge (v_i, v_j) (e.g. from v_i to v_j) in the configuration s_l . If $v_i \in s_l$ then the corresponding robot r_i is in the v_i -ROUTER polygon else it is in the $\neg v_i$ -ROUTER polygon. To reach the configuration corresponding to s_{l+1} the robot r_i (corresponding to v_i) need to reaches the $\neg v_i$ -ROUTER polygon and the robot r_j (corresponding to the vertex v_j) need to reaches the v_j -ROUTER polygon. In order to do that, r_i executes movements described in Remark 5 and r_j executes movements described in Remark 6. Since s_{l+1} is an independent set, $\forall v_h \in N(v_j)$, the corresponding robot r_h can reach the $\neg v_h$ -ROUTER polygon then r_j can reaches the v_j' output of the v_j -VERTEX polygon and accesses to the v_j -ROUTER. By repeating these movements, we can pass from $s_1 = A$ to $s_{|S|} = B$. Then we have k robots such that $\forall v_i \in B$, the corresponding robot r_i is in the v_i -ROUTER polygon. By Construction 2 these k robots can reach the TARGET polygon and one of them can reach the target tile. Then, if I admits a positive solution, I' admits a positive solution too.

- Reciprocally, we suppose that there exists a positive solution for the instance I' of $\text{GR}(n, 1, 1, 1)$ then we construct a positive solution for the instance I of TOKEN SLIDING (TS) in the following way:

The instance I' admits a solution then k robots can reach the target polygon (according to Remark 3). Then, according to Lemma 7 and Lemma 9 $\exists C$ a set of configurations such that $\forall c_l \in C$, $\forall r_i \in R$, r_i is in the v_i -ROUTER polygon or in the $\neg v_i$ -ROUTER polygon. Note that c_1 corresponds to the starting position ($\forall v_i \in V$, if $v_i \in A$, the corresponding robot r_i starts in the v_i -ROUTER polygon else it starts in the $\neg v_i$ -ROUTER polygon) and $c_{|C|}$ is the configuration where $\forall v_i \in V$, if $v_i \in B$, the corresponding robot r_i is in the v_i -ROUTER polygon else it is in the $\neg v_i$ -ROUTER polygon. According to Lemma 9 two robots cannot cross an v_j -MOVING polygon at the same time. From Lemma 2 and to Construction 2, to access to the v_j -ROUTER polygon, r_j have to cross the v_j -VERTEX polygon and then $\forall v_h \in N(j)$, the corresponding robot r_h is in the $\neg v_h$ -ROUTER polygon. Then we can consider the set S such that $\forall c_l \in C$, we can consider the set $s_l \in S$ such that $\forall r_i \in R$, if r_i is in the corresponding v_i -VERTEX polygon in the configuration c_l , then $v_i \in s_l$. So $\forall c_l \in C$, according to Lemma 2 and to Construction 2, the set s_l is an independent set. By construction, a robot r_i can move from the v_i -ROUTER polygon to the $\neg v_i$ -ROUTER polygon and another robot r_j from the $\neg v_j$ -ROUTER polygon to the v_j -ROUTER polygon if the corresponding $(v_i, \neg v_j)$ -SLIDING polygon exist and then is the edge $(v_i, v_j) \in E$. Then $\forall c_l \in C$

we have an independent set $s_l \in S$ and to pass from s_l to s_{l+1} by withdraw a vertex and adding one of its neighbour in G (i.e. a single token slide along an edge) and where $s_1 = A$ and $s_{|S|} = B$. Then the set of independent set S corresponding to C is a solution for the instance I of TOKEN SLIDING (TS).

Since [Construction 2](#) can be done in polynomial time and from previous arguments, $\text{GR}(n, 1, 1, 1)$ is PSPACE-complete. \square

5 Conclusion

In this article, we proposed some complexity results for the Ricochet Robots game. We have proven the following:

- the optimization problem corresponding to Ricochet Robots is Poly- \mathcal{APX} -hard ([Section 3](#)),
- determine if an instance of Ricochet Robots is solvable or not is PSPACE-complete ([Section 4](#)).

The next step is to categorize MAXIMUM REACHABILITY (MR) problem more precisely in the hierarchy of approximation algorithms. We can also investigate on the minimum number of moves to solve the GENERALIZED REACHABILITY (GR) problem.

References

- [1] Cristina Bazgan, Bruno Escoffier, and Vangelis Th Paschos. Completeness in standard and differential approximation classes: Poly-(d) apx-and (d) ptas-completeness. *Theoretical Computer Science*, 339(2-3):272–292, 2005.
- [2] Nicolas Butko, Katharina A Lehmann, and Veronica Ramenzoni. Ricochet robots—a case study for human complex problem solving. *Proceedings of the Annual Santa Fe Institute Summer School on Complex Systems (CSSS’05)*, 2005.
- [3] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005.
- [4] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006.
- [5] Pierluigi Crescenzi. A short guide to approximation preserving reductions. In *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference*, pages 262–273. IEEE, 1997.
- [6] Erik D Demaine, Michael Hoffmann, and Markus Holzer. Pushpush-k is pspace-complete. In *Proceedings of the 3rd International Conference on FUN with Algorithms*, pages 159–170. Citeseer, 2004.
- [7] Birgit Engels and Tom Kamphans. Randolphins robot game is np-hard! *Electronic Notes in Discrete Mathematics*, 25:49–53, 2006.
- [8] Martin Gebser, Holger Jost, Roland Kaminski, Philipp Obermeier, Orkunt Sabuncu, Torsten Schaub, and Marius Schneider. Ricochet robots: A transverse asp benchmark. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 348–360. Springer, 2013.
- [9] Martin Gebser, Roland Kaminski, Philipp Obermeier, and Torsten Schaub. Ricochet robots reloaded: A case-study in multi-shot asp solving. In *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation*, pages 17–32. Springer, 2015.
- [10] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.
- [11] Adam Hesterberg and Justin Kopinsky. The parameterized complexity of ricochet robots. *Journal of Information Processing*, 25:716–723, 2017.
- [12] Markus Holzer and Stefan Schwoon. Assembling molecules in atomix is hard. *Theoretical computer science*, 313(3):447–462, 2004.

- [13] Falk Hüffner, Stefan Edelkamp, Henning Fernau, and Rolf Niedermeier. Finding optimal solutions to atomix. In *Annual Conference on Artificial Intelligence*, pages 229–243. Springer, 2001.
- [14] Christian Icking, Thomas Kamphans, Rolf Klein, and Elmar Langetepe. Exploring an unknown cellular environment. In *EuroCG*, pages 140–143, 2000.
- [15] Christian Icking, Tom Kamphans, Rolf Klein, and Elmar Langetepe. Exploring simple grid polygons. In *International Computing and Combinatorics Conference*, pages 524–533. Springer, 2005.
- [16] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.
- [17] Takehiro Ito, Erik D Demaine, Nicholas JA Harvey, Christos H Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011.
- [18] Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical computer science*, 439:9–15, 2012.
- [19] Daniel Lokshantov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.
- [20] Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.
- [21] GerhardJ. Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization — Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–207. Springer Berlin Heidelberg, 2003.
- [22] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

A Representation

In order to simplify our scheme and to simplify the reader’s understanding, gadgets are represented by a rectangle and are connected by ”wires”. A wire corresponds to a line of free spaces surrounded by walls on both sides. A wire can be bent and two wires can cross them by preserving the planarity of the grid (Table 2). Note that two robots which crossing each other cannot exchange their way (in other words, the robot which gets in vertically cannot reach an horizontal output and vice versa).


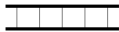




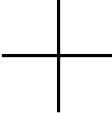
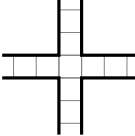
	Representation	Corresponding grid
wire		
cutted wire		
bended wire		
crossed wires		

Table 2: Wires representations.