A coin-moving game on graphs

Florian Galliot

A coin-moving game on graphs

Florian Galliot

21 June 2019

Master's thesis under the direction of
Sylvain Gravier (Institut Fourier) & Isabelle Sivignon (GIPSA-lab)

M2 ORCO, Université Grenoble-Alpes, France

This work has been partially supported by the LabEx PERSYVAL-Lab
(ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Contents

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Introduction

We study a one-player game that is played on an undirected simple graph $G = (V, E)$, referred to as the *game graph* or *board*. Throughout the game, there will be coins sitting on some of the vertices of G (at most one per vertex). In the unlabelled version of the game, the coins are undistinguishable and define a *configuration*, i.e. a finite subset $C \subseteq V$ where we see each element of C as a coin sitting on the corresponding vertex. In the labelled version, all coins wear a distinct number and define a *labelled configuration*, i.e. a configuration with the additional information of which coin sits where. A legal *move* consists of moving a single coin c to a free position such that c has at least two other coins adjacent to it after it is moved. This is called the *2-adjacency* restriction. Given two (potentially labelled) configurations A and B, we want to know whether the puzzle $A \xrightarrow{?} B$ is solvable: starting from A, is it possible to reach B using only legal moves? In the positive case, we would like an explicit winning sequence of moves. We are also interested in complexity aspects of this problem.

Instances of this game, or rather a variation with tightly packed coins that can only be *slid* in the plane without collision, appear in the litterature as early as the 1950s in [2] and [3]. Figure 1 features a couple of classic puzzles on the triangular grid as well as a more rare puzzle on the square grid. Such examples also appear in [4] and [5] among others, but it is not until 2002 that general puzzles with these rules have been studied, in the article [1] that serves as foundation for the present master's thesis. The authors give a characterisation for solvable puzzles on the triangular grid, and address a large family of puzzles on the square grid, providing polynomial time solving algorithms. Their work does not seem to have been developed since. Another coin-moving game on the square grid, with different rules but similar methods and also polynomial time algorithms for a large array of cases, is studied in [6]. Such games fall into the category of reconfiguration problems in graphs, where more recent but less related works include [7], [8], [9], [10] (unlike our game, the configurations must maintain a given property additionally to the movement rule).

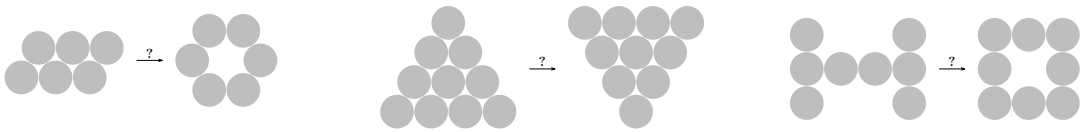

Figure 1. In the first two puzzles, the coins have to stay in the triangular grid. In the third puzzle, they have to stay in the square grid. The left puzzle is solvable in 2 moves or 3 slides. The middle puzzle is solvable in 3 moves/slides. The right puzzle is solvable in 4 moves/slide.

In this master's thesis, we aim at furthering the study of the game that is made in [1]. We will start by regrouping the (few) results we know that hold for general graphs. We will then talk about the triangular grid, where the game is entirely solved, using alternative methods compared to that of [1]. Finally, the major part will address the square grid, where the authors obtained a strong result about a large family of puzzles but also left numerous areas to inspect: we will complete a few of their results and introduce new tools to study some unexplored cases.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

1 General observations

This section deals with notions and results that hold for any game graph G.

Notation 1.1. Let us first introduce some useful notations and vocabulary:

- Moving a coin c from its current position p_o to a legal destination p_d may be denoted indifferently by $c : p_o \mapsto p_d$, $c \mapsto p_d$ or $p_o \mapsto p_d$. If p_o also has two occupied neighbours, the move is said to be *reversible*, in the sense that it could immediately be undone by moving c back. Note that, in the unlabelled case, a coin c is simply an element of V representing its location (a coin c is nothing but an occupied position p). Typically, the notation c is used to designate an occupied position, and the notation p is used for a position that is either unoccupied or not necessarily occupied.
- The neighbourhood of a position $p \in V$ is denoted by $N(p)$.
- If there is a single move m from A to B, we write $A \overset{m}{\mapsto} B$ or simply $A \mapsto B$.
- The *graph of configurations* for the unlabelled (resp. labelled) game, denoted by $\mathscr{G}$ (resp. $\mathscr{G}_l$), is the digraph where the vertices are the configurations and there is an arc (A, B) if and only if $A \mapsto B$. Given A and B with $|A| = |B|$, our problem is to find out whether there is a directed path from A to B in $\mathscr{G}$ (resp $\mathscr{G}_l$), which we denote by $A \overset{\mathscr{G}}{\to} B$ (resp. $A \overset{\mathscr{G}_l}{\to} B$) or simply $A \to B$.
- By extension, if $|A| > |B|$, then $A \to B$ means that there exists a directed path from A to some $B^+ \supset B$ where $|B^+| = |A|$. We will also use B^+ as a standard notation for a superset of B.

1.1 Necessary conditions

It is easy to spot a few conditions that are necessary for a puzzle $A \overset{?}{\to} B$ to be solvable.

Proposition 1.2. *Let $A \neq B$ be configurations such that $|A| = |B|$. If $A \to B$, then:*

- *There exist two coins in A that share an unoccupied neighbour.*
- *There exists a coin in B that has at least two neighbours in B. In other words, B contains a connected component of size at least 3.*

Proof. No move would be possible from A if the first condition was not satisfied, and the last moved coin when going from A to B necessarily has two coins adjacent to it. □

Proposition 1.3. *Let $A \neq B$ be configurations. Suppose there is a coin $b \in B$ such that all connected components of $B \setminus \{b\}$ are of size 1. Then $A \to B$ if and only if $A \mapsto B$.*

Proof. Suppose that there exists a sequence of moves $A = A_0 \overset{m_1}{\longmapsto} A_1 \overset{m_2}{\longmapsto} \ldots \overset{m_T}{\longmapsto} A_T = B$. Since $B \setminus \{b\}$ consists of all isolated coins, b is the only coin in B that can have two neighbours in B,

2 This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

therefore the final move m_T is necessarily of the form $p_T \mapsto b$. Same goes for $A_{T-1}\backslash\{p_T\} = B\backslash\{b\}$, so the penultimate move m_{T-1} is necessarily of the form $p_{T-1} \mapsto p_T$. Going back in time like this until the beginning, we notice that the coin that ends up at position b is the only one that is being moved throughout, occupying positions $p_1, \dots, p_T, b$ successively. In particular, the winning move $p_1 \mapsto b$ was legal from the beginning, hence $A \mapsto B$. □

1.2 Span

Notation 1.4. Let C be a configuration, we denote by $\mathrm{Adj}(C) \subseteq V \setminus C$ the set of all positions outside C that have at least two neighbours in C.

Definition 1.5. The *span* of a configuration C, denoted by $\mathrm{span}(C)$, is the set of all positions that could be reached from C if we had unlimited coins to add to the board at will in valid positions. In other words, $\mathrm{span}(C)$ is the limit of the non-decreasing sequence of configurations $(C_i)_{i\geq 0}$ defined recursively by $C_0 = C$ and $C_{i+1} = C_i \cup \mathrm{Adj}(C_i)$.

The following properties of the span are elementary. As a corollary, we get a new and simple (yet important) necessary condition for a puzzle $A \xrightarrow{?} B$ to be solvable.

Proposition 1.6. *Let C, C_1, C_2, C_3 be configurations, then:*

- $\mathrm{span}(\mathrm{span}(C)) = \mathrm{span}(C)$.
- *If* $C_1 \subseteq C_2$ *then* $\mathrm{span}(C_1) \subseteq \mathrm{span}(C_2)$.
- *If* $\mathrm{span}(C_1) = \mathrm{span}(C_2)$ *then* $\mathrm{span}(C_1 \cup C_3) = \mathrm{span}(C_2 \cup C_3)$.

Corollary 1.7. *The span never increases during moves: if* $A \to B$ *then* $\mathrm{span}(A) \supseteq \mathrm{span}(B)$.

Proof. Clearly, any configuration obtained from A is included in $\mathrm{span}(A)$, hence the result by monotonicity of the span. □

We now design an algorithm to compute finite spans. This will suppose that we have a theoretical argument, about either the game graph or the specific configuration, that ensures the finiteness of the span (otherwise the algorithm obviously never terminates). See Appendix A.1.1.

Proposition 1.8. *Let C be any configuration with finite span S, then S can be computed in $O(|S|\Delta^2)$ time where Δ is the maximum degree of a vertex in G.*

Proof. Let $(C_i)_{i\geq 0}$ as in the definition of span. Since S is finite, we actually get a finite sequence $(C = C_0, \dots, C_s = S)$. We start by computing $A = \mathrm{Adj}(C)$ in $O(|C|\Delta^2)$ time: we initialize $A = \varnothing$, and for each coin $c \in C$ and each path $c \text{ — } p \text{ — } q$ in G, if $p \notin C \cup A$ and $q \in C$ then we add p to A. We can then compute $\mathrm{Adj}(C_1)$ the same way, but for this it suffices to go through the positions in A: indeed, any new position that we might add to the span at this

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

stage will have at least one neighbour in A (if it had two neighbours in C, then it would be in A already). We can then compute $\mathrm{Adj}(C_2)$ by going through all the positions in $\mathrm{Adj}(C_1)$, etc. until we cannot discover any new position at which point the span is fully known. The running time is $O(|C|\Delta^2 + |\mathrm{Adj}(C)|\Delta^2 + |\mathrm{Adj}(C_1)|\Delta^2 + \ldots + |\mathrm{Adj}(C_{s-1})|\Delta^2) = O(|S|\Delta^2)$. □

1.3 Minimal/minimum configurations

Definition 1.9. Let C be a configuration. We say C is *span-minimal*, or simply *minimal*, if the removal of any coin in C decreases the span. Otherwise, we may say C is *minimal+k* where k is the maximum number of coins that can be removed from C without reducing the span. We say C is *minimum* if there is no configuration C' with same span as C such that $|C'| < |C|$.

Proposition 1.10. *Any move played from a minimal configuration A will decrease the span.*

Proof. This is clear, since removing a coin decreases the span and then re-placing it anywhere in $\mathrm{Adj}(A)$ does not increase it. □

Proposition 1.11. *Let C be a minimal (resp. minimum) configuration. Then any subconfiguration of C is also minimal (resp. minimum).*

Proof. Suppose that $C' \subseteq C$ is not minimal, i.e. $\mathrm{span}(C') = \mathrm{span}(C' \setminus \{c\})$ for some $c \in C'$. Then $\mathrm{span}(C) = \mathrm{span}(C' \cup (C \setminus C')) = \mathrm{span}((C' \setminus \{c\}) \cup (C \setminus C')) = \mathrm{span}(C \setminus \{c\})$, therefore C is not minimal either. The proof is the same in the minimum case. □

The next result, although less fundamental, will be useful during our study of the square grid.

Proposition 1.12. *Consider a sequence of moves $A = A_0 \overset{c_1 \mapsto p_1}{\longmapsto} A_1 \overset{c_2 \mapsto p_2}{\longmapsto} A_2 \mapsto \ldots$ such that the span is preserved throughout and no coin is moved twice in a row. Then, for all t such that $A_t \setminus \{c_{t+1}\}$ is minimal, we have $c_{t+2} \in N(p_{t+1})$. In particular, if A_t is minimal+1 for all t, then each moved coin is necessarily a neighbour of the coin that has been moved just before.*

Proof. Suppose $A_t \setminus \{c_{t+1}\}$ is minimal and let $c \in A_{t+1} \setminus N(p_{t+1})$, we show that the removal of c from A_{t+1} reduces the span. Since $A_{t+1} \setminus \{c\}$ contains two distinct neighbours of p_{t+1}, we have $\mathrm{span}(A_{t+1} \setminus \{c\}) = \mathrm{span}((A_{t+1} \setminus \{c\}) \setminus \{p_{t+1}\})$. Moreover $A_{t+1} \setminus \{p_{t+1}\} = A_t \setminus \{c_{t+1}\}$ is minimal, therefore $\mathrm{span}(A_{t+1} \setminus \{c\}) = \mathrm{span}((A_{t+1} \setminus \{p_{t+1}\}) \setminus \{c\}) \subsetneq \mathrm{span}(A)$. □

4 This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

1.4 An equivalent variation of the unlabelled game

Let us consider the following variation of the unlabelled game. Here, we may stack several coins at the same position, and we also have a reserve R of available coins besides the coins on board. There are three types of legal actions:

1. *Move*: Move a coin c from its current position p_o to any position p_d that has at least two occupied neighbours, regardless of whether p_d is already occupied (here p_o itself counts as on occupied neighbour if it hosts at least one coin other than c).
2. *Take*: Take a coin c off the board and add it to the reserve. This is denoted by $c \mapsto R$.
3. *Drop*: Take a coin from the reserve and place it at any position p that has at least two occupied neighbours. This is denoted by $R \mapsto p$.

Note that a *move* is exactly equivalent to a *take* immediately followed by a *drop*, but it is still helpful to view a *move* as an action of its own. Finally, let us define *reversible* actions: reversible *moves* are defined as in the original game, a *drop* is always reversible, and a *take* $c \mapsto R$ is said to be reversible if c has at least two occupied neighbours.

In this new game, a configuration is a pair (A, r) where:

- $A : V \to \mathbb{N}$ indicates how many coins sit at each position. However, in the absence of stacking, we will simply see A as a standard configuration i.e. a finite subset of V.
- $r = |R|$ is the number of coins in the reserve.

Let $\mathscr{G}'$ be the corresponding graph of configurations.

We now show that the alternative game, which is a priori easier than the original game, is actually equivalent to it. This facilitates our study of the original game, since we can freely give ourselves the more permissive set of rules above.

Proposition 1.13. *Let A and B be configurations such that $|A| \geq |B|$, then $A \xrightarrow{\mathscr{G}} B$ if and only if $(A, 0) \xrightarrow{\mathscr{G}'} (B, |A| - |B|)$. Moreover, there exists a simple algorithm in $O(|A| + T)$ time to convert a sequence of T actions from $(A, 0)$ to $(B, |A| - |B|)$ in $\mathscr{G}'$ into a sequence of at most T moves from A to B^+ in $\mathscr{G}$.*

Proof. The direction from $\mathscr{G}$ to $\mathscr{G}'$ is clear: we just follow a sequence of *moves* from A to B^+ in $\mathscr{G}$, and at the end we *take* all coins in $B^+ \setminus B$. Let us show the other direction. Let $(a_t)_{1 \leq t \leq T}$ be a sequence of actions from $(A, 0)$ to $(B, |A| - |B|)$ in $\mathscr{G}'$. We proceed in two steps:

1. We first transform this into a sequence of all *moves*, with the following remark. Since stacking coins is allowed, we can postpone the first *take* to just before the first *drop*, and then pair them to form a *move*. Similarly, we pair the second *take* with the second *drop*, etc. If $|A| > |B|$, then we simply ignore the last (unpaired) $|A| - |B|$ *takes*.
2. We get a sequence of *moves* $(m_t = (o_t \mapsto d_t))_{1 \leq t \leq T'}$ from $(A, 0)$ to $(B^+, 0)$ in $\mathscr{G}'$, and we have to eliminate any stacking of coins that may occur. This is easily done: we go through this sequence in order, but whenever a move m_t is illegal for $\mathscr{G}$ (i.e. the destination d_t is presently occupied), we ignore it and replace the next move with origin d_t (if there is one) by a move with the same destination but origin o_t instead, which effectively swaps the roles of the coins at positions o_t and d_t.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

As for the statement on the complexity, we refer to the algorithm CLEAN from Appendix A.1.2, which computes the above transformation. In this pseudocode, C and P are arrays that are defined in the remark that follows the present proposition. We see that the algorithm only goes through the sequence twice (step 1, then step 2), addressing each t in constant time. Removing the empty moves is also done in $O(T)$ time, and lines 16 to 18 are executed in $O(|A|)$ time. □

Remark. The arrays C and P are of size $|V|$ and should be defined outside of the algorithm:

- C is the current configuration: $C[p] = 1$ if there is a coin at position p and $C[p] = 0$ otherwise. It is initialised as all zeros.
- P contains the information of which coin plays the role of which during step 2: if $P[p] = p'$, then the next move with origin p will be replaced by a move with same destination but origin p'. It is initialised as the identity.

The true running time of the algorithm CLEAN should thus be $O(|V| + T)$. However, once C and P are initialised, any instance of the algorithm on the same graph G runs in a further $O(|A| + T)$ time and can be followed by a reset of C and P (i.e. all zeros for C and the identity for P) in $O(|A| + T)$ time also, so that C and P can be re-used for any number of calls to the algorithm. Therefore, C and P can be created once and for all at the beginning of the study of a given graph G (even: a given $|V|$), and it makes sense to separate their initialisation cost $O(|V|)$ from the rest. Future complexities will not take this cost into account.
As for the triangular and square grids, we simulate the infinite vertex set V using a finite subset of positions W inside of which all the moves will take place. Identifying positions with their coordinates in Z^2, it is sensible to define W as a (large) rectangle and to implement C and P as matrices: this way, accessing $C[p]$ and $P[p]$ from the coordinates (i, j) of p is done in constant time, which is crucial not only in CLEAN but also in other algorithms we may encounter. As before, C and P only need to be initialised once and for all at a cost $O(|W|)$ to study any number of sequences of moves, as long as everything happens inside W.

An important consequence of the previous proposition is that coins can never "get in the way" in the unlabelled version of the game: suppose a sequence of moves could be made provided we initially remove certain coins, then it can also be made without taking these coins off the board or even moving them.

Corollary 1.14. *Let A and B be configurations such that $|A| \geq |B|$, and let C be a configuration disjoint from A. If $A \xrightarrow{\mathscr{G}} B$, then $A \cup C \xrightarrow{\mathscr{G}} B \cup C$.*

Proof. Apply the same sequence of moves that gives $A \xrightarrow{\mathscr{G}} B$: because of the additional coins C, some stacking may occur, and we get $A \cup C \xrightarrow{\mathscr{G}'} B \cup C$ (indeed, the coins in C never move throughout, even though they might get other coins stacked onto them during the process). Proposition 1.13 concludes. □

Note that obstacles do matter a lot in the labelled game. The transition from $\mathscr{G}'$ to $\mathscr{G}$ does not behave well with labels, because it is not possible to swap the roles of two coins. Allowing to *take* and *drop* labelled coins would be "cheating" since it would alter the game, deceivingly

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

making some puzzles appear solvable when they are not. For instance, consider $G = K_n$ ($n \geq 4$) and A a labelled configuration where all vertices are occupied, then no move is possible from A in the actual game, whereas any labelled configuration would be reachable from A if we were allowed to *take* and *drop* coins.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

2 The triangular grid

In this section, $G = (V, E)$ is the infinite triangular grid. In the main version of the game, V has a fixed ordering, however we will also consider variations of the game where the end configuration B can be reached up to any translation and/or rotation of the coins. In graphical representations of this game, instead of the coins sitting on vertices of the triangular grid, we will see them as sitting on faces of the dual hexagonal grid which is equivalent.

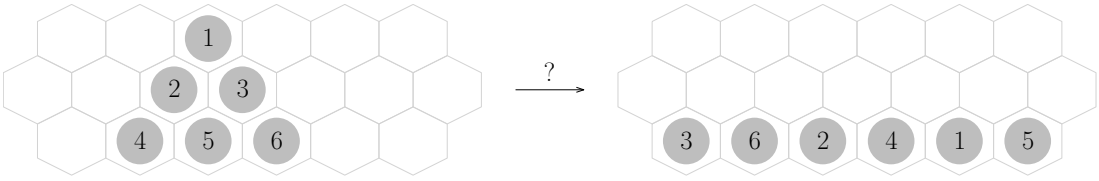

Figure 2. An example of a labelled puzzle on the triangular grid.

2.1 Unlabelled game

We first consider the unlabelled version of the game, and we will then explain how to adapt if the coins are labelled. Remember that Proposition 1.13 allows us to play the game $\mathscr{G}$ or $\mathscr{G}'$ indifferently, which we will put to use. This is a different approach from that of [1], where the authors solve both the unlabelled and labelled versions at the same time, explaining how to transport any specific coin to a desired location without emulation through other coins. The core of the reasoning will be the same however, namely triangle manoeuvring.

2.1.1 Triangle manoeuvring

Definition 2.1. A *triangle* is a set of three pairwise adjacent positions/coins.

A simple but crucial observation is that a triangle of coins can be moved all over the board. For a possible implementation of the game graph and triangle manoeuvring, see Appendix A.2.

Lemma 2.2. *For all triangles T_1 and T_2, we have $T_1 \to T_2$. Moreover, let d be the distance between T_1 and T_2, then this can be done in $O(d)$ moves that can be found in $O(d)$ time.*

Proof. As apparent on Figure 2, there is a one-to-one correspondence between the triangles and the vertices of the hexagonal grid (each vertex is the center of a triangle). Therefore, by the connectivity of the hexagonal grid, it suffices to check the case where T_1 and T_2 are neighbours in that sense. This is straightforward, since each of the three neighbouring triangles

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

of $T_1 = \{a, b, c\}$ is obtained by moving either a, b or c across the other two. As for the result on complexity, it is easily obtained by considering the algorithms from Appendix A.2.1 for instance. □

Note that, as long as the starting configuration A allows at least one move, it is always possible to form a triangle of coins in at most two moves. Indeed, after the first move, we have two adjacent coins and we can bring a third one adjacent to both if needed. In particular, the notion of span is irrelevant to the case of the triangular grid.

Notation 2.3. The following notations will be useful in the next part.

- If $C = \{c\} \subset V$, we define positions $p_1(C)$, $p_2(C)$ and $p_3(C)$ as in Figure 3, and we define the triangles $T(C) = \{c, p_1(C), p_2(C)\}$ and $T'(C) = \{p_1(C), p_2(C), p_3(C)\}$.

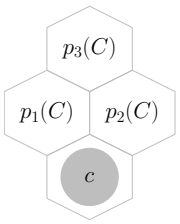

Figure 3. $C = \{c\}$.

- If $C = \{c_1, c_2\} \subset V$ where c_1 and c_2 are adjacent, we define positions $p_1(C)$, $p_2(C)$ and $p_3(C)$ as in Figure 4, and we define the triangles $T(C) = \{c_1, c_2, p_1(C)\}$ and $T'(C) = \{p_1(C), p_2(C), p_3(C)\}$.

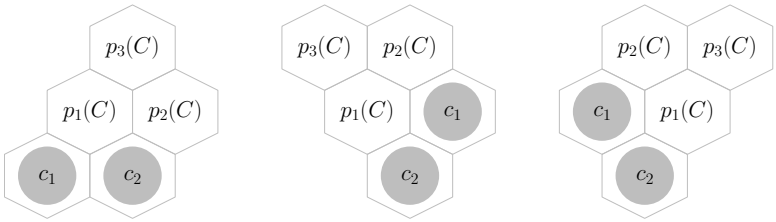

Figure 4. $C = \{c_1, c_2\}$, three possible shapes.

2.1.2 A full solution

Triangle manoeuvring allows us to transport coins anywhere on board, and build practically anything as a result. The following theorem gives an exact characterisation of solvable puzzles on the triangular grid, and provides a polynomial time algorithm to identify them and find a winning sequence of moves. It is remarkable that the set of three necessary conditions given by Propositions 1.2 and 1.3 turns out to be sufficient. See Appendix A.2.2 for a possible implementation and Figure 6 for an illustration of how the algorithm works.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Theorem 2.4. *Let $A \neq B$ be configurations with same cardinality n. We suppose that A allows at least one move. Then $A \to B$ if and only if at least one of the following four conditions is satisfied:*

(i) B contains a triangle.
(ii) B contains a path of 4 coins.
(iii) B contains a connected component of size at least 3 and another one of size at least 2.
(iv) $A \mapsto B$.

The same result holds for variations of the game where translations and/or rotations are allowed, with the same proof. Moreover, denoting by d_B (resp. d_{AB}) the maximum distance between two coins in B (resp. a coin in A and a coin in B), there exists an algorithm running in $O(d_{AB} + nd_B)$ time that decides whether the puzzle $A \xrightarrow{?} B$ is solvable and finds a winning sequence of $O(d_{AB} + nd_B)$ moves in the positive case.

Proof. We start by proving that it is necessary that one of these conditions is satisfied in order to have $A \to B$, and then we give a constructive proof that it is also sufficient.

(a) Negative case: Neither of (i), (ii), (iii) or (iv) holds.

- The first possibility is that all connected components of B are of size at most 2, in which case Proposition 1.2 ensures that $A \not\to B$.
- The second possibility is that all coins in B are isolated except for a single connected component C of size at least 3 that does not contain a triangle or a path of 4 coins. This means that C is a tree with diameter 2 i.e. a star. Let b be the center of that star, then all connected components of $B \setminus \{b\}$ are of size 1. Since $A \not\mapsto B$, we get $A \not\to B$ by Proposition 1.3.

(b) Positive case: Either (i), (ii), (iii) or (iv) holds. Since the result is obvious if (iv) holds, we will suppose that either (i), (ii) or (iii) holds.

By Proposition 1.13, it suffices to prove that $(A, 0) \xrightarrow{\mathscr{G}'} (B, 0)$. Let $(b_1, \ldots, b_e)$ be a maximal (it need not be maximum) sequence of coins in B such that, for all $i \in [\![1, e]\!]$, $B \setminus \{b_1, \ldots, b_{i-1}\}$ contains at least two coins adjacent to b_i. Note that $e \geq 1$ given our assumption. Let $B^- := B \setminus \{b_1, \ldots, b_e\}$. We show that $(A, 0) \xrightarrow{\mathscr{G}'} (B^-, e)$, which is sufficient since we can then *drop* the e coins from the reserve at positions $b_e, b_{e-1}, \ldots, b_1$ successively to obtain B.

Our new target configuration B^- has less coins than A: in a way, we now have e coins in extra to solve the puzzle. By construction, all connected components of B^- are of size 1 or 2. Moreover, let n_1 (resp. n_2) denote the number of connected components of size 1 (resp. 2) in B^-, then either $n_2 \geq 1$ or $e \geq 2$. To prove this, let us suppose that $e = 1$ and show that it implies $n_2 \geq 1$ under each of (i), (ii) or (iii):

(i) Let T be a triangle in B, then the lone removed coin necessarily comes from T which leaves a connected component of size 2.
(ii) Let P be a path of 4 coins in B, then the removal of a single coin leaves a connected component of size at least 2 (and thus exactly 2).
(iii) Let $C_3 \neq C_2$ be connected components of B such that $|C_3| \geq 3$ and $|C_2| \geq 2$, then $|C_2| = 2$ and the lone removed coin comes from C_3 which means that C_2 is left intact.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

We construct the connected components of B^- one by one, starting with those of size 1 and finishing with those of size 2. Once a connected component is constructed, we never touch it afterwards. Let $C_1, \ldots, C_{n_1+n_2}$ be the connected components of B^-, where $|C_i| = 1$ for $i \leq n_1$ and $|C_i| = 2$ for $i > n_1$. We proceed in four steps, as follows:

1 Using the fact that A allows at least one move, we put together a triangle T of coins. We then *take* all the coins on board except for the three that form T.

2 We *move* T to $T(C_1)$ using Lemma 2.2. We then leave C_1 there in its correct and final position, and *drop* a coin at $p_3(C_1)$ to form the triangle $T'(C_1)$. After that, we *move* $T'(C_1)$ to $T(C_2)$, leave C_2 there etc. until all the connected components of size 1 are constructed.

3 We *move* $T'(C_{n_1})$ to $T(C_{n_1+1})$, leave C_{n_1+1} there, *drop* two coins at $p_2(C_{n_1+1})$ and $p_3(C_{n_1+1})$ respectively, *move* $T'(C_{n_1+1})$ to $T(C_{n_1+2})$, leave C_{n_1+2} there etc. until all the connected components of size 2 are constructed.

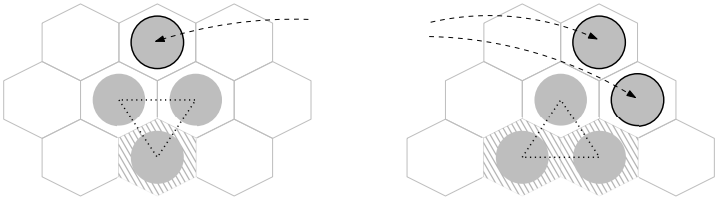

Figure 5. Illustration of steps 2 (left) and 3 (right). The triangle has been placed so as to cover C_i (shaded area). The circled coins are then brought from the reserve to reconstitute the triangle and be able to move elsewhere, while the coins in C_i stay there until the end.

4 Once B^- is fully constructed, if the final connected component $C_{n_1+n_2}$ was of size 1 (resp. 2) then we simply *take* the coin (resp. the two coins) in $T(C_{n_1+n_2}) \setminus C_{n_1+n_2}$.

Note that some stacking of coins may occur when we *move* our triangle around and/or when we *drop* coins to reconstitute it, because of the previously constructed C_i on board.

We shall verify that we do not *drop* more than the $|A| - 3$ coins we have in our reserve after step 1. Recall that $|A| = |B^-| + e = n_1 + 2n_2 + e$. There are two possibilities:

- If $n_2 = 0$, then we have seen that $e \geq 2$. We *drop* one coin $n_1 - 1$ times. The total number of dropped coins is $(n_1 - 1) \times 1 \leq n_1 + e - 3 = |A| - 3$.
- If $n_2 \geq 1$, then we *drop* one coin n_1 times then two coins $n_2 - 1$ times. The total number of dropped coins is $n_1 \times 1 + (n_2 - 1) \times 2 = n_1 + 2n_2 - 2 \leq n_1 + 2n_2 + e - 3 = |A| - 3$.

We wrap up the proof with the complexity considerations. Computing the connected components of B is done in $O(n)$ time, as well as finding a triangle or a path of length 4, therefore the four conditions (i), (ii), (iii) and (iv) can be checked in $O(n)$ time. The coins $b_1, \ldots, b_e$ can also be found in $O(n)$ time, since it suffices to go through all coins in B once. Steps 1 and 4 both feature less than n actions found in $O(n)$ time. As for steps 2 and 3, the first triangle is moved over a distance at most d_{AB}, and after that we do $n_1 + n_2 - 1 \leq n$ triangle transportations between the connected components of B. Note that the d_{AB} quantity disappears if we play up to translations. Finally, Proposition 1.13 addresses the conversion from $\mathscr{G}'$ to $\mathscr{G}$. □

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

If $|A| > |B|$, then $A \to B$ if and only if there is a way to add $|A| - |B|$ coins to the configuration B such that it satisfies (i), (ii), (iii) or (iv). Therefore:

- If $|A| \geq |B| + 2$, then $A \to B$. Indeed, we can always add one or two coins to B if needed to form a triangle.
- If $|A| = |B| + 1$, then $A \to B$ if and only if $A \supset B$ or B contains a connected component of size at least 2. Indeed, if B contains a connected component of size at least 2 then we can add a coin if needed to form a triangle, and if $A \supset B$ then there is nothing to do. In the case where $A \not\supset B$ and all coins in B are isolated however, the additional coin could only be in between two coins of B and be the only coin moved throughout, which means it is impossible to reach B.

In particular, an algorithm with the same complexity still works in the case where $|A| > |B|$.

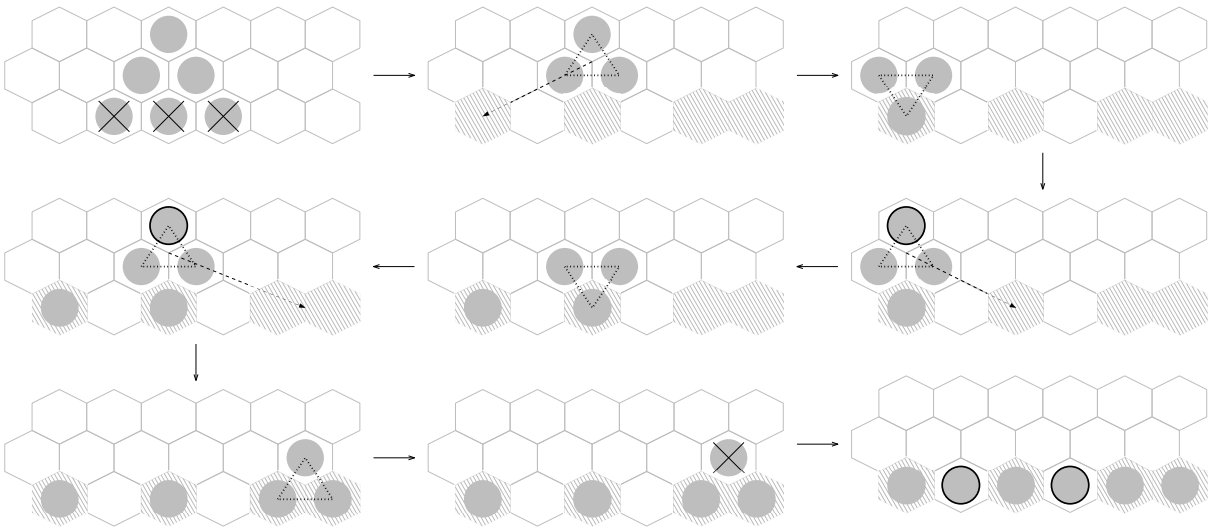

Figure 6. Solving the unlabelled version of the puzzle from Figure 2. *Dropped* coins are circled and *taken* coins are crossed out. The shaded area represents B^- ($e = 2$ here).

Remark. There is a mistake in the version of the theorem that is mentioned in [1]. The condition (ii) is replaced by the condition that B contains a connected component of size 4, which in insufficient. For a counterexample, let B be the star with 4 coins (see Figure 7): according to Proposition 1.3, B is reachable from A if and only if $A = B$ or $A \mapsto B$. Also note that this star disproves the conjecture that the authors make at the very end of the article, about the variation of the game where the coins need to be *slided* on the board: the puzzle on the right of Figure 7 is solvable according to Theorem 2.4 (condition (iii) is satisfied), but the coin at the center of the star cannot have been slided in the middle of the other three.

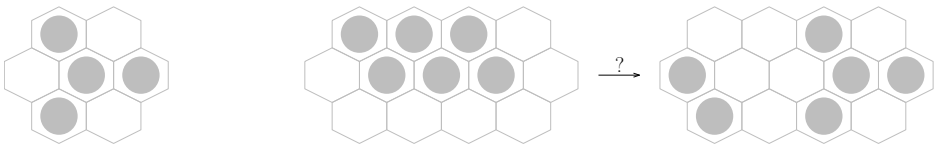

Figure 7. On the left: the star with 4 coins. On the right: a puzzle that is solvable in our game but not in the sliding version.

12 This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

2.2 Labelled game

We are also able to solve the labelled version of the game completely. We are going to separate the case of labelled puzzles with only 3 coins since they work a little differently. For labelled puzzles with at least 4 coins, we get the same result as in the unlabelled version. The idea is to first treat the puzzle as if the coins were unlabelled, observe the mistake that is made on the labels, and then start again applying a pre-emptive correction. This method is algorithmically efficient, however it is probably not the best way to proceed when playing the game in real life (even though it is technically possible). A labelled puzzle like the one from Figure 2 for instance, which is solvable according to the following theorem, is more naturally solved by directly bringing the right coins to the right places as per [1].

Theorem 2.5. *Theorem 2.4 also holds for labelled configurations with cardinality $n \geq 4$.*

Proof. Obviously, the negative case of Theorem 2.4 is still negative with labels, so we only need to address the positive case. Let A and B be labelled configurations satisfying one of the four conditions. Starting from A, we could follow the sequence of moves provided by Theorem 2.4 to put all the coins at the positions given by B, but the labels would likely be mixed up. We deal with this problem by arranging the coins at the very beginning, in a way that is designed to ensure the labels are correct at the end. Here is how we proceed:

1 Using the fact that A allows at least one move, we put together a triangle of coins. Say this triangle points upwards and covers positions p_1, p_2 and p_3 from left to right. We then stick all the other coins in A to these three, in no particular order, at the successive positions $p_4, \ldots, p_n$ defined in Figure 8. Let A_1 be the new labelled configuration formed by this agglomerate of coins, and let $\tau(i)$ be the label of the coin at position p_i in A_1.

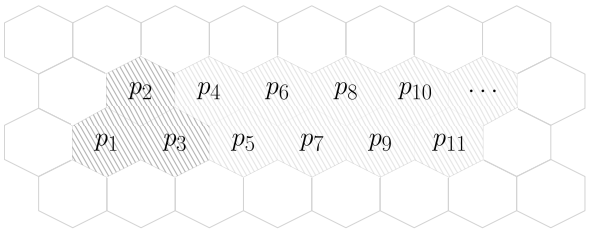

Figure 8. At the end of step 1, the coins occupy the positions $p_1, \ldots, p_n$. Any label might be at any position.

2 We run the algorithm for unlabelled puzzles from Theorem 2.4, and thus get a sequence of moves $(m_t)_{1 \leq t \leq T}$ from the positions in A_1 to those in B.

3 By tracking the movements of each coin throughout this sequence, we observe the difference $\delta \in \mathfrak{S}_n$ between where the labels would end up by applying $(m_t)_t$ starting from A_1 and where we would want them to be (i.e. where they are in B): this means that, at the position where coin i is in B, we would get coin $\delta(i)$ instead. Set $\sigma = \tau^{-1} \circ \delta \circ \tau$, and let A_2 be the labelled configuration where a coin that is at position p_i in A_1 is at position $p_{\sigma(i)}$ instead, we now perform $A_1 \xrightarrow{\mathscr{G}_1} A_2$. To prove that this is possible, the case

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

where σ is a transposition of the form $(1\ \ i)$ suffices since σ is always a product of such transpositions, and it is detailed in Figure 9. Note that this is only possible because $n \geq 4$.

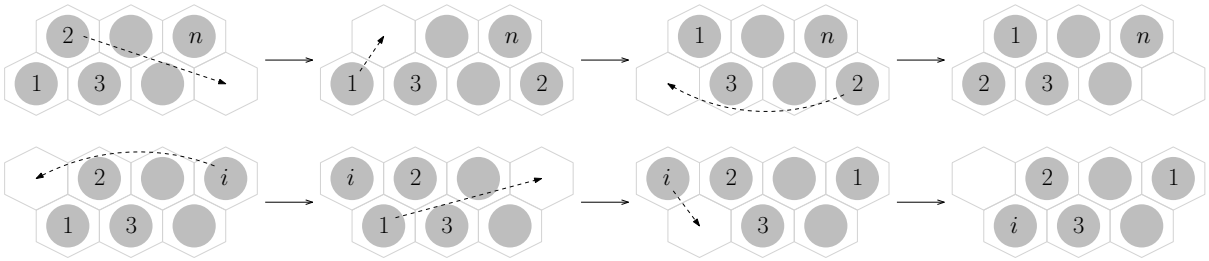

Figure 9. Permuting labelled coins. Top: $s = (1\ \ 2)$. Bottom: $s = (1\ \ i)$, $i \geq 3$.

4 The definition of σ ensures that this rearrangement nullifies the mistake δ, so that we can now apply $(m_t)_t$ starting from A_2 and get exactly B.

Step 1 is done in less than n moves found in $O(n)$ time. Computing σ and decomposing it as a product of $O(n)$ transpositions of the form $(1\ \ i)$ is done in $O(n)$ time, therefore the routines from Figure 9 ensure that step 3 is done in $O(n)$ moves found in $O(n)$ time. In conclusion, the result on complexity is the same as in Theorem 2.4. □

It is only for labelled puzzles with 3 coins that it makes a difference whether we play up to translations/rotations or not. In the main version of the game, where translations/rotations are not allowed, it will be useful to 3-colour the board as in Figure 10. Configurations themselves can be considered coloured, with each coin conceptually wearing the colour of its current position.

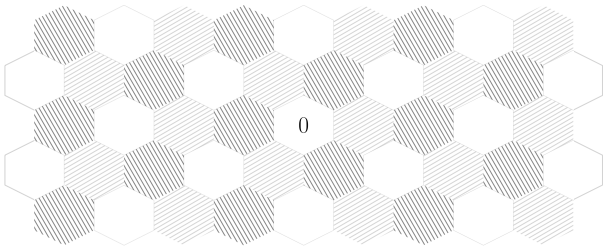

Figure 10. Our 3-colouration of the game board. The origin "0" is fixed.

Notation 2.6. We categorise labelled triangles into two classes, depending on their shape and labels (and regardless of their exact position on board), as in Figure 11. The class of a labelled triangle T is denoted by $Cl(T)$.

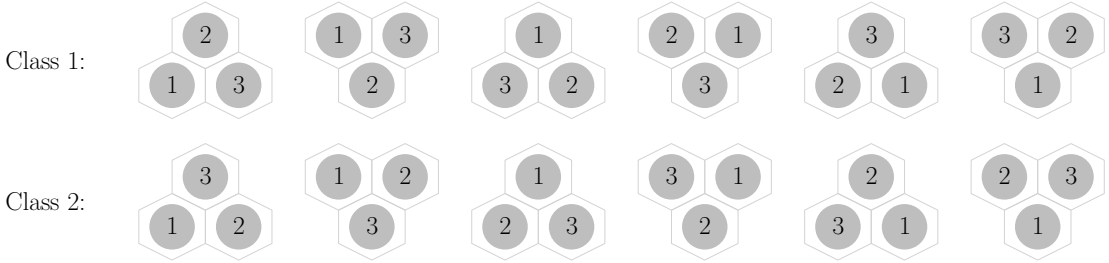

Figure 11. 12 possible shapes/labels, split into two classes.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Theorem 2.7. *Let A and B be distinct (up to translations/rotations if they are allowed) labelled configurations with exactly 3 coins. We suppose A allows at least one move.*

- *If B is not connected, then $A \not\to B$.*
- *If B is a path that is not a triangle, then $A \xrightarrow{\mathscr{G}_l} B$ if and only if $A \overset{\mathscr{G}_l}{\mapsto} B$ (up to translations/rotations if they are allowed).*
- *If B is a triangle, then:*
  - *If translations/rotations are not allowed, then $A \xrightarrow{\mathscr{G}_l} B$ if and only if $A \overset{\mathscr{G}_l}{\mapsto} B'$ in at most two moves for some $B' \in Cl(B)$ with same colours as B.*
  - *If translations are allowed, then $A \xrightarrow{\mathscr{G}_l} B$ if and only if $A \overset{\mathscr{G}_l}{\mapsto} B'$ in at most two moves for some $B' \in Cl(B)$.*
  - *If translations and rotations are allowed, then $A \xrightarrow{\mathscr{G}_l} B$.*

Proof. We can always assume that we never move the same coin twice in a row. A first remark is that, from the second move on, the coins will always form a triangle: indeed, if they form a path $c_1 — c_2 — c_3$ that is not a triangle after the first move, then c_2 has just been moved, which forces us to next bring either c_1 or c_3 adjacent to the other two, thus forming a triangle that we will keep until the end. A second remark is that the coins in this triangle will never change colour (a coin can only be sent accross the other two, which preserves its colour), and therefore the triangle will never change class since it would necessitate that at least one coin changes colour (as we can see in Figure 11).

- The case where B is not connected is straightforward by Proposition 1.2.
- Suppose B is a path that is not a triangle. Then, by the first remark above, the puzzle is solvable if and only if it is solvable in only one move.
- Suppose B is a triangle.
  - If translations/rotations are not allowed, then by the above remarks it is necessary that $A \overset{\mathscr{G}_l}{\mapsto} B'$ in at most two moves for some $B' \in Cl(B)$ with same colours as B. To prove it is sufficient, since all elements of the same class are clearly reachable from each other, we just have to verify that a triangle can be moved to any position where it would have the same shape, labels and colours. This is done by repeating routines such as that of Figure 12.

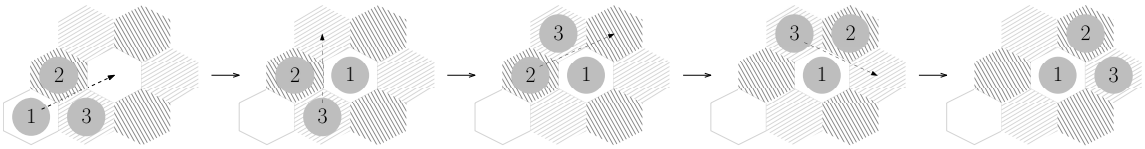

Figure 12. Translating a triangle. Preserving the shape and labels means preserving the colours.

  - If translations are allowed, they do not help overcome differences in class but they do make colours irrelevant, hence the statement from the theorem.
  - If rotations are also allowed, then there is no notion of class anymore (for instance, take Figure 11 and rotate the first element of Class 1 clockwise by 60°, we get the second element of Class 2) hence why there is no condition needed on the triangles that can be obtained after two moves. □

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

3 The square grid

In this section, $G = (V, E)$ is the infinite square grid. We will again use the (self-)dual grid for graphical representations of the game, where each position is seen as a square and coins are placed at the center of squares. It will sometimes be helpful, especially in Section 3.3, to see a configuration as the plain area on board formed by its occupied squares.

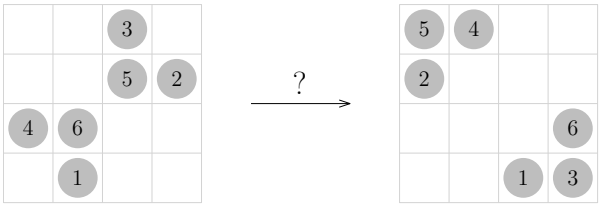

Figure 13. An example of a labelled puzzle on the square grid.

3.1 Span and canonical configurations

Definition 3.1. An $m \times n$ *rectangle* is a subset of $V \simeq \mathbb{Z}^2$ of the form $R = I \times J$ where I and J are intervals of cardinality m and n respectively. We say R is *even* (resp. *odd*) if its half-perimeter $m + n$ is even (resp. odd). The *hull* of a configuration C, denoted by $R(C)$, is the smallest rectangle containing C.

We can immediately notice that the span of any configuration is included in its hull: indeed, no position outside of the hull has two distinct neighbours inside of the hull. In particular, all configurations have bounded span, which contrasts with the case of the triangular grid. The following proposition gives a more precise description of spans on the square grid.

Proposition 3.2. *The span of any configuration C is a union of rectangles at distance at least 3 from each other.*

Proof. It is clear that the span of a connected configuration is equal to its hull, therefore each connected component of $\mathrm{span}(C)$ is a rectangle. Moreover, suppose for a contradiction that two of these components R_1 and R_2 are at distance only 2 from each other. Then there exists a position p that is a common neighbour of R_1 an R_2, therefore $p \in \mathrm{span}(C)$, which means R_1 and R_2 should be in the same connected component of $\mathrm{span}(C)$. □

Definition 3.3. Let C be a configuration. If $c, c' \in C$, a *2-path* in C between c and c' is a sequence $(c = c_0, c_1, \ldots, c_l = c')$ of coins in C such that c_i and c_{i+1} are *2-neighbours*, i.e. $\mathrm{dist}(c_i, c_{i+1}) \in \{1, 2\}$ where dist denotes the geodesic distance in the square grid. We say C is *2-connected* if for all $c, c' \in C$ there exists a 2-path between c and c'.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Definition 3.4. An 'L' of size $m \times n$ is a configuration L such that:

- L forms a 2-path between two opposite corners of an $m \times n$ rectangle R ;
- The coins in L hug two consecutive sides of R ;
- $|L| = \lceil \frac{m+n}{2} \rceil$.

We say L is *even* (resp. *odd*) if R is even (resp. odd) i.e. if $m + n$ is even (resp. odd). If $m = 1$ or $n = 1$, we may call L a *line* 'L'.

It is easy to see that 'L's of any size do exist (see Figure 14). Notice that an 'L', just like any 2-connected configuration, has a span equal to its hull (R in the definition). Consecutive coins in an even 'L' are at distance exactly 2, whereas consecutive coins in an odd 'L' are at distance exactly 2 except for a single adjacent pair. An even 'L' is entirely defined by its span and orientation, whereas for an odd 'L' we also need the localisation of the two adjacent coins.

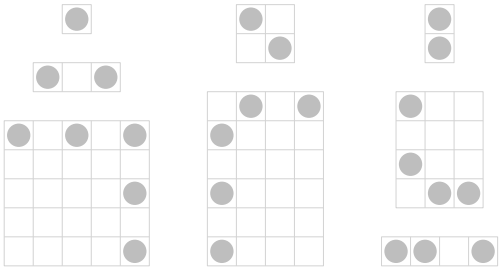

Figure 14. Examples of 'L's: even 'L's with both odd sides (left), even 'L's with both even sides (middle), odd 'L's (right).

Definition 3.5. Let R be a rectangle, the *canonical 'L'* with span R is the 'L' with span R that is oriented like the letter L, with the additional property if R is odd that the two adjacent coins are in the top-left corner (if the vertical branch is even) or bottom-right corner (if the horizontal branch is even). Let C be a configuration with span $\bigcup_{i=1}^{s} R_i$ where $R_1, \ldots, R_s$ are rectangles at distance at least 3 from each other, the *canonical configuration* associated to C is the configuration L with same span as C such that $L \cap R_i$ is the canonical 'L' with span R_i.

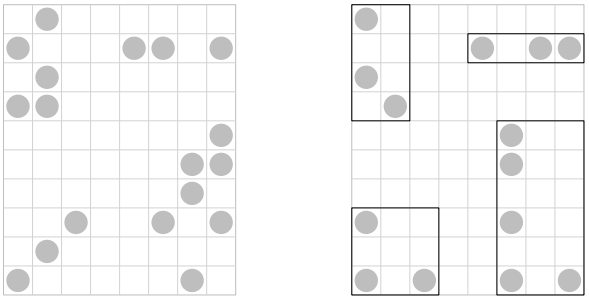

Figure 15. A configuration of coins and its associated canonical configuration.

A key information is the maximum number e of coins that can be removed from the starting configuration A without reducing the span, or at least without making it smaller than the span of the target configuration B. Those will be referred to as *extra coins*. The case $e = 0$ is trivially negative by Proposition 1.10. The essential disjunction will only be between two cases: $e = 1$ or $e \geq 2$, the former proving more difficult.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

3.2 Two extra coins: a partial solution

We first study the case where we have two or more extra coins at our disposal. If A and B have the same span, this means exactly that A is minimal+k with $k \geq 2$. We get a strong result in both the unlabelled and labelled versions of the game, however we will have to make an assumption on B that means the case of two extra coins will not be solved entirely.

3.2.1 Unlabelled game

In the unlabelled game on the square grid, we will use Theorem 1.13 again (we will not stack coins, but we will *take* and *drop* whenever we want). We may start any puzzle by *taking* as many extra coins as possible so as to use them when and where they are needed. On the triangular grid, the key was triangular manoeuvring, which allowed us to transport coins anywhere. There is no equivalent of this on the square grid where spans are bounded, however the two extra coins do make it possible to send coins anywhere inside the span while preserving it. The central result is the ability to re-orient an 'L' at will:

Lemma 3.6. *Let L_1 and L_2 be two 'L's with the same $m \times n$ span, then $(L_1, 2) \xrightarrow{\mathscr{G}'} (L_2, 2)$. Moreover, this can be done in $O(mn)$ actions that can be found in $O(mn)$ time.*

Proof. Even though it suffices to show that any 'L' can be *rotated* by $\pm$ 90°, we will also show how an 'L' can be directly *flipped* (i.e. rotated by 180°), both because it is easier and because there are interesting observations to be made in the case of odd 'L's. Using symmetries, it suffices to consider the case where L_1 is canonical.

(a) First case: m and n are odd.

- Flip: Using the fact that a 3×3 'L' can by flipped in 8 actions, we can flip an $m \times n$ 'L' in $O(mn)$ actions. See Figure 16.

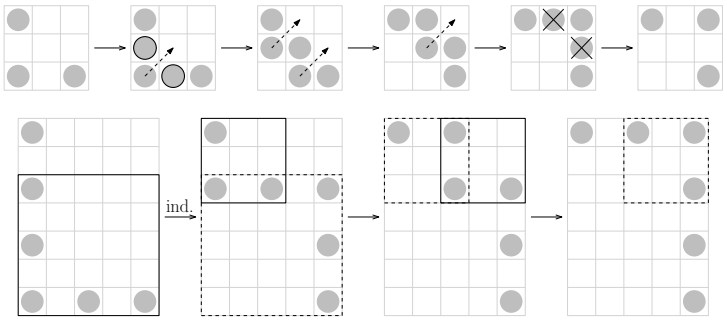

Figure 16. Flipping an $m \times n$ 'L' where m and n are odd. *Dropped* coins are circled and *taken* coins are crossed out. We simply repeat the 3×3 subroutine $O(mn)$ times. In the first step, "ind." means we use induction.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

- Rotation: Using the fact that a 3×3 'L' can be rotated in 11 actions, we can rotate an $m \times n$ 'L' in $O(mn)$ actions. See Figure 17.

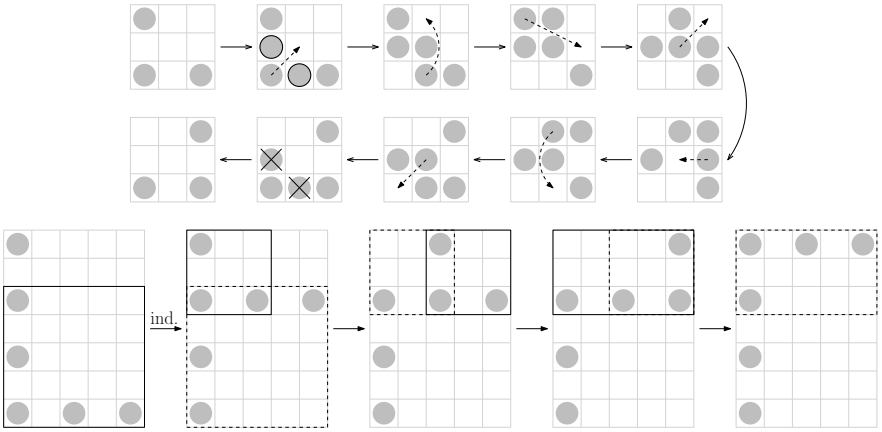

Figure 17. Rotating an $m \times n$ 'L' where m and n are odd. In total, we use $O(mn)$ 3×3 subroutines and $O(n)$ $m \times 3$ flips.

(b) Second case: $m + n$ is odd.

Let us first remark that the two adjacent coins can be moved at will inside the 'L', in $O(m+n)$ actions and using only one extra coin (see Figure 18). We will call this a *leapfrog*. In particular, this ensures that we can indeed consider only the case where L_1 is canonical.

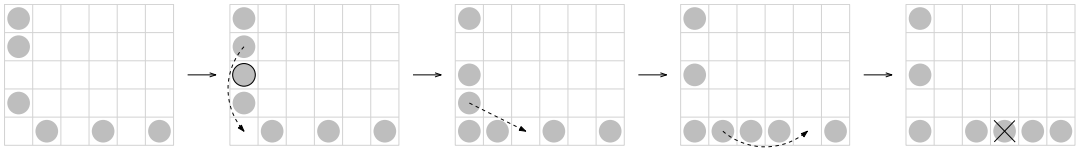

Figure 18. Moving the connected component of size 2 inside an odd 'L'.

- Flip: Using the fact that a 3×2 'L' can be flipped in 4 actions, and introducing an elementary 3×4 subroutine in 8 actions, we can flip an odd $m \times n$ 'L' in $O(mn)$ actions **using only one extra coin**. See Figure 19.

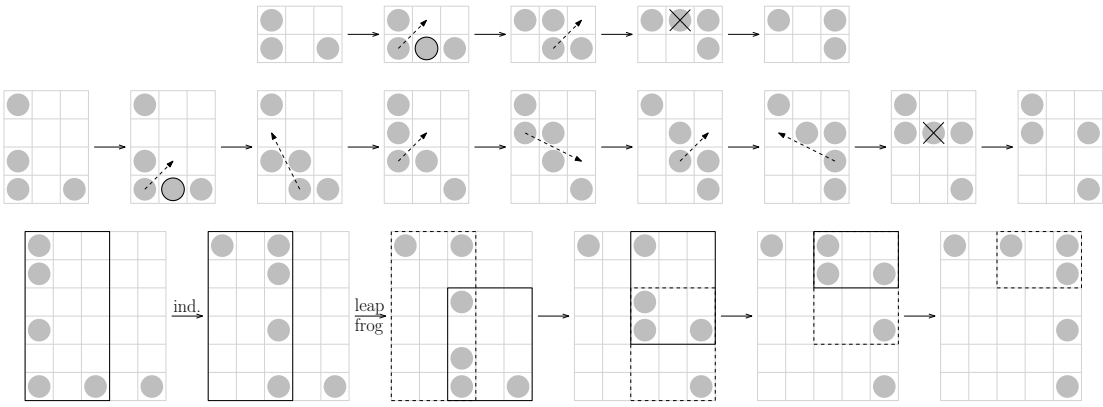

Figure 19. Flipping an odd 'L' (here the vertical branch is even). We use the 3×4 subroutine $O(mn)$ times, and the 3×2 subroutine $O(m)$ times. We also leapfrog $O(m)$ times.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

- Rotation: Using the fact that a 3×2 'L' can be rotated in 5 actions, we can rotate an $m \times n$ 'L' in $O(mn)$ actions. See Figure 20. Both extra coins are crucial this time: it is easily checked that even the 3×2 'L' cannot be rotated with just one extra coin.

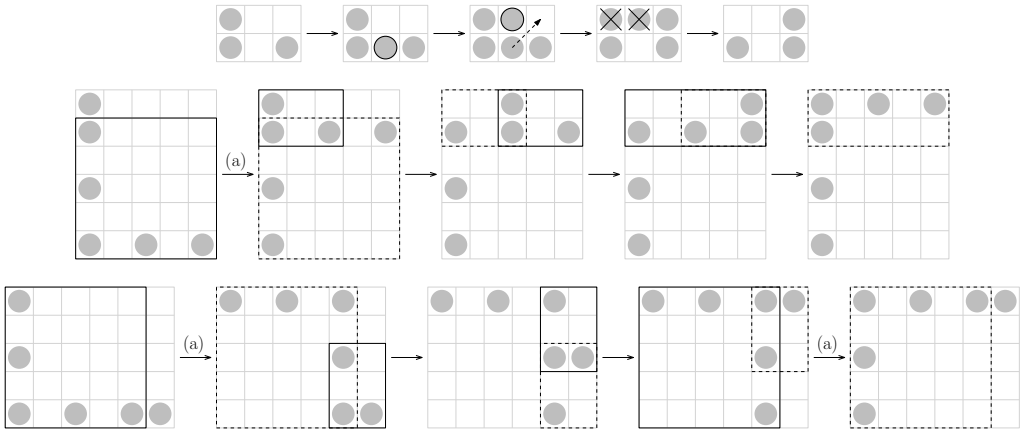

Figure 20. Rotating an odd 'L', depending on which branch is even. We use one flip, one rotation from the first case, and the 3×2 subroutine $O(m)$ or $O(n)$ times.

(c) Third case: m and n are even.

- Flip: Using the fact that a 4×2 'L' can be flipped in 5 actions, we can flip an $m \times n$ 'L' in $O(mn)$ actions. See Figure 21.

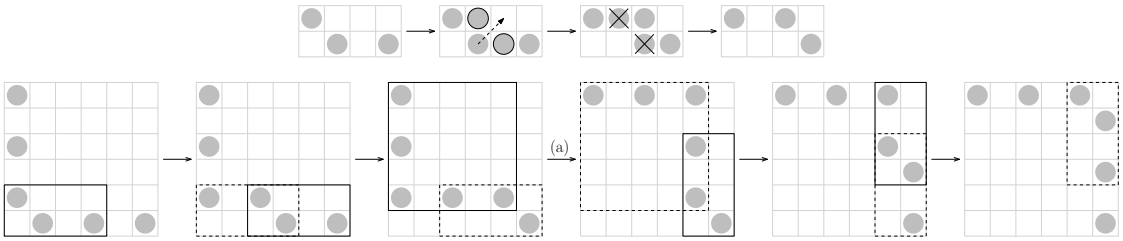

Figure 21. Flipping an 'L' where m and n are even. We use the 4×2 subroutine $O(m+n)$ times and one flip from the first case.

- Rotation: Using the fact that odd 'L's can be flipped using only one extra coin, we can rotate an $m \times n$ 'L' in $O(mn)$ actions. See Figure 22.

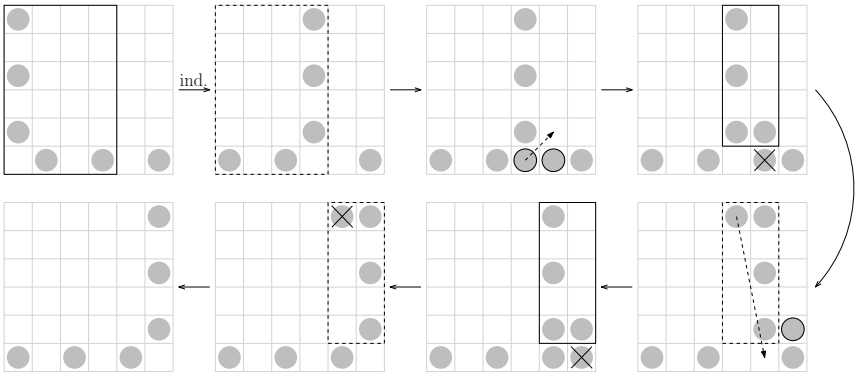

Figure 22. Rotating an 'L' where m and n are even. We flip odd $2 \times (n-1)$ 'L's $O(m)$ times, with just $O(n+m)$ total additional actions in between.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Let us finally justify the assertion on complexity. We have defined some small elementary subroutines, the biggest one being of size 3×4, each of which only consists of a few moves and can be pre-implemented. A flip/rotation/leapfrog on L_1 uses $O(mn)$ of these subroutines, with types and locations that only depend on the type and location of L_1. Therefore the time needed to find the sequence of moves from L_1 to L_2 is simply the $O(mn)$ time needed to translate each subroutine (so that it is performed at the right location) and add it to the list of moves that will be output. □

If $\mathrm{span}(A) = \mathrm{span}(B)$, the idea is to go from A to B by routing through their common associated canonical configuration. We now show that two extra coins suffice to transform any configuration (envision A) into its associated canonical configuration, in such a way that we can also travel back to the original configuration (envision B). This is also the approach used in [1], even though we use a different algorithm.

Theorem 3.7. *Let C be a configuration and L_C be the canonical configuration associated to C, then $(C, 2) \overset{\mathscr{G}'}{\longleftrightarrow} (L_C, 2 + |C| - |L_C|)$. Moreover, setting $N = |C|$, both directions can be done in $O(N^3)$ actions that can be found in $O(N^3)$ time.*

Proof. We give a constructive proof that $(C, 2) \overset{\mathscr{G}'}{\longrightarrow} (L_C, 2 + |C| - |L_C|)$, and show that our construction can easily be reversed.

The key is the ability to *increment* an 'L', i.e. use a nearby coin to make the 'L' bigger, thus increasing its length by 1 (resp. 2) if the coin closest to its hull is at distance 1 (resp. 2). An 'L' can be incremented to the side or diagonally, for a total of eight possible directions. Note that some additional coins might need to be absorbed in the process (see Figure 23, left). To *augment* an 'L' means to increment it as long as possible, i.e. increment it repeatedly until there is no coin at distance 1 or 2 from its hull, with the following priority rules: we only increment diagonally if we cannot increment to the side, and when incrementing to the side we maintain the same direction while it is possible.

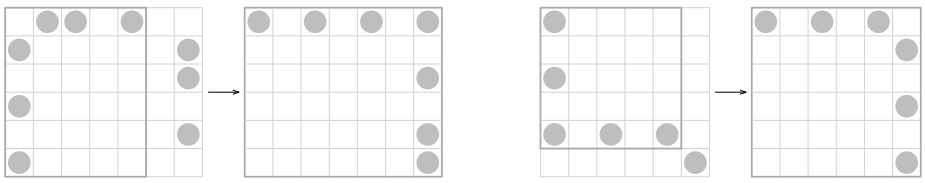

Figure 23. Incrementing an 'L'. On the left: an increment to the side, at distance 2. On the right: a diagonal increment.

In [1], the authors explain how to merge two full 'L's L_1 and L_2 at once, as well as how to deal with the possible surplus of coins inside $R(L_1 \cup L_2)$ (the difficulty being that simply *taking* these is not reversible a priori). Instead, we prefer to see this merge as L_1 progressively "swallowing" the coins in L_2, increment by increment: this way, we go over all the occupied positions on board manually, and we will show how each coin in surplus can be timely *taken* while it has two coins adjacent to it (i.e. reversibly). Our algorithm is as follows:

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

1. Initialize a set $\mathscr{L}$ of 'L's as $\mathscr{L} = \{\{c\}, c \in C\}$.
2. While there exist $L_1 \neq L_2$ in $\mathscr{L}$ such that $R(L_2)$ has a corner p_2 at distance at most 2 from $R(L_1)$:
   i. Re-orient L_2 (if needed) so that there is a coin at position p_2.
   ii. Re-orient each 'L' in $\mathscr{L} \setminus \{L_1, L_2\}$ (if needed) so that it hugs the two sides of its hull that are closest to L_1.
   iii. Augment L_1.
   iv. Update $\mathscr{L}$: replace the old L_1 by the new and enlarged L_1, and remove all 'L's that have been swallowed during the augmentation.
3. Make the configuration canonical via re-orientations and leapfrogs.

Note that during the augmentation of L_1 in step 2.iii:

- Each 'L' other than L_1 either gets entirely swallowed or remains intact. Indeed, if one coin of that 'L' gets eaten, then so will the other ones by the 2-connectedness.
- L_2 gets swallowed, due to step 2.i. The cardinality of $\mathscr{L}$ thereby decreases at each iteration of step 2, and the algorithm ends after at most $N-1$ iterations of step 2.
- Any 'L' whose hull intersects that of L_1 at any point during the augmentation of L_1 automatically gets swallowed, due to step 2.ii. In consequence, throughout the algorithm, $\mathscr{L}$ contains 'L's with pairwise disjoint hulls.

Moreover, if two rectangles are at distance at most 2 from each other, then one of them has a corner at distance at most 2 from the other. Therefore, at the end of step 2, all 'L's in $\mathscr{L}$ have hulls at distance at least 3 from each other, and the configuration obtained after step 3 is indeed canonical (with same span as C, since we will only use actions that preserve the span).

On to complexity considerations:

- Finding L_1 and L_2 can be done in $O(N^2)$ time by simply checking all pairs of 'L's in $\mathscr{L}$.
- If L is an $m \times n$ 'L', then $mn \leq \left(\frac{m+n}{2}\right)^2 \leq |L|^2$ and therefore L can be re-oriented in $O(|L|^2)$ actions. Since the sum of the cardinalities of all 'L's in $\mathscr{L}$ does not exceed N:
  - Each iteration of steps 2.i and 2.ii is done in $O(N^2)$ actions. Step 2 being iterated less than N times during the algorithm, these represent $O(N^3)$ actions in total.
  - Step 3 is done in $O(N^2)$ actions.
- We shall now address increments. Finding a nearby coin that can be used for the increment is done in $O(N)$ time (we check all positions around $R(L_1)$ at distance 1 or 2). We will see that each diagonal increment is done in $O(N^2)$ actions found in $O(N^2)$ time, and that the same holds for any number of side increments made consecutively in the same direction. Because of our priority rules for augmentations, swallowing an 'L' is done using $O(1)$ changes of direction, therefore an augmentation during which k 'L's are swallowed is done in $O(kN^2)$ actions found in $O(kN^2)$ time. Since at most $N-1$ 'L's get swallowed throughout the whole process, we obtain the complexity statement from the theorem.

It is tautological that re-orientations and leapfrogs of 'L's can be reversed. All that is left now is to explain how to increment an 'L' and why it can also be reversed.

(a) Incrementing an 'L' to the side.
We will only show how to increment an 'L' to the right. See Figure 29 for a detailed example. Whether it be at distance 1 or 2, we will proceed the same:

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

1. If needed, we re-orient L so that it hugs the top and right sides of its hull. If L is odd, we also leapfrog to put the two adjacent coins at the bottom. This is the most costly step of the increment, with $O(N^2)$ actions used.
2. Of all coins closest to L to its right, we *drag* the topmost one along L towards the bottom as in Figures 24 and 25. If we encounter another coin on the way down, we *absorb* one of the two as in Figures 26 and 27, and we continue to drag and absorb successively until the only remaining coin reaches the bottom. This is done in $O(N)$ actions, that can be found in $O(N)$ time (we drag/absorb $O(N)$ times in total, with only one square to check each time to know whether to drag or absorb and thus activate the adequate subroutine at the right location).
3. We obtain a 'Z' shape that we transform back into an 'L' as in Figure 28, in $O(N)$ actions found in $O(N)$ time.

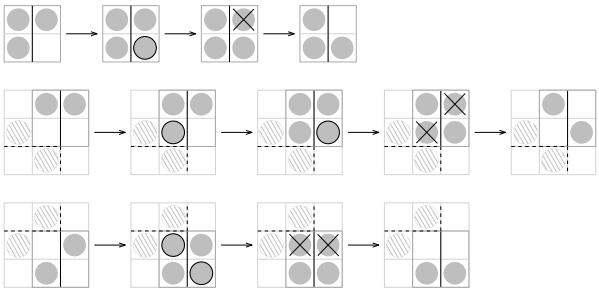

Figure 24. Dragging a coin at distance 1 towards the bottom, all cases. When there are two shaded coins, it means that the 'L' contains one of the two.

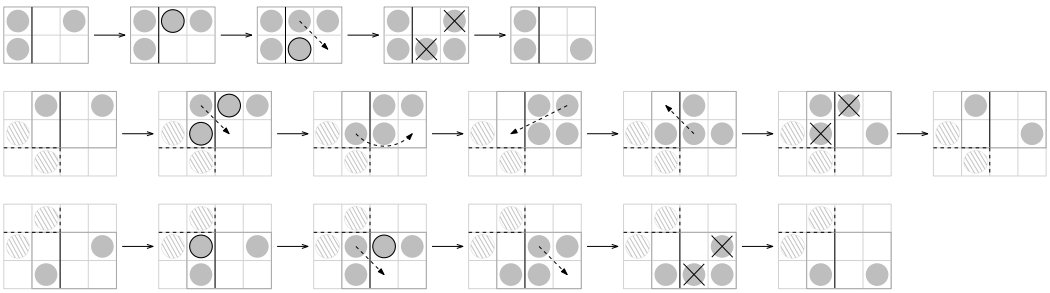

Figure 25. Dragging a coin at distance 2 towards the bottom, all cases.

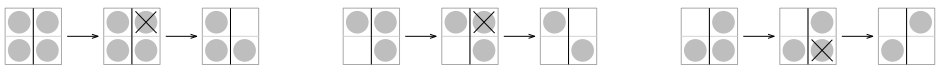

Figure 26. Absorbing a coin at distance 1, all cases. All actions are reversible.

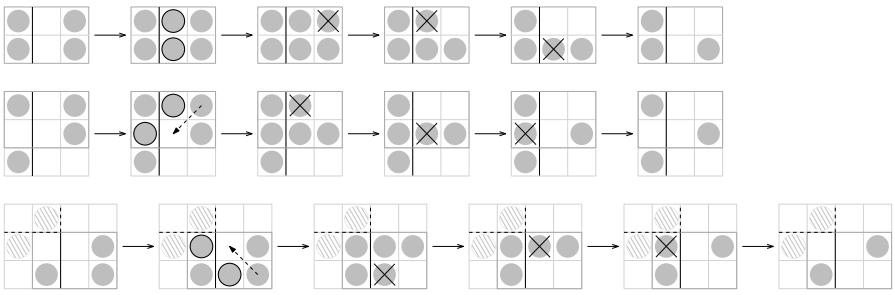

Figure 27. Absorbing a coin at distance 2, all cases. All actions are reversible.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

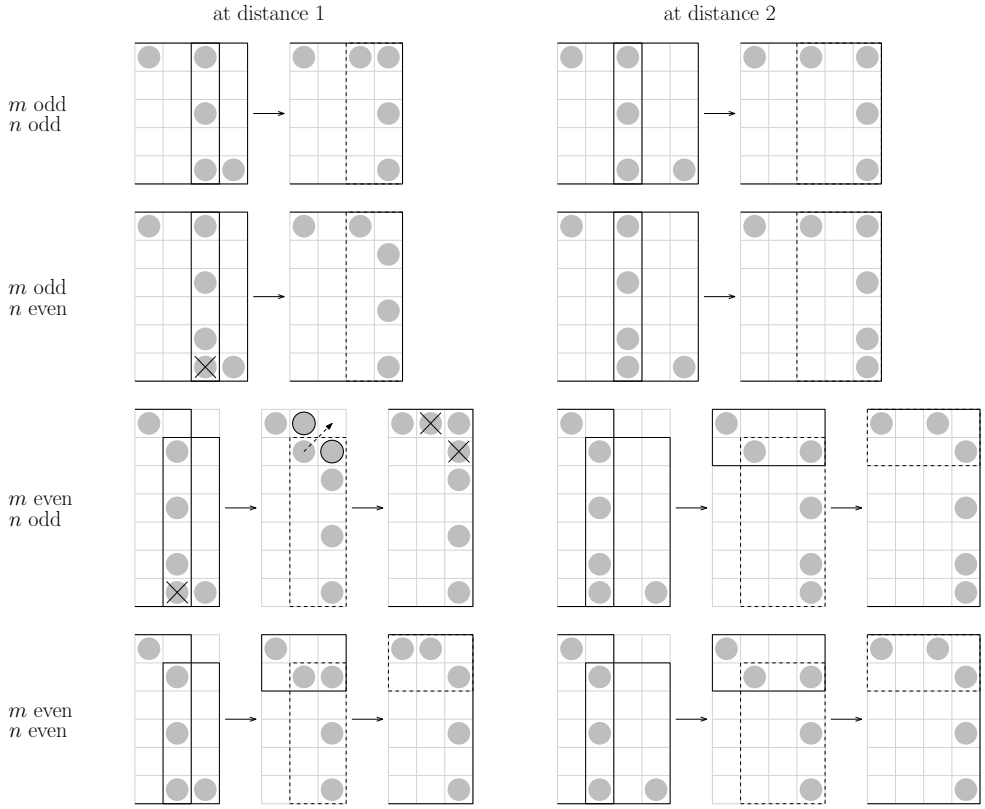

Figure 28. Finishing the increment, using a flip in $O(N)$ actions and $O(1)$ other actions. There are eight cases, depending on the parity of each branch and whether the added coin is at distance 1 or 2.

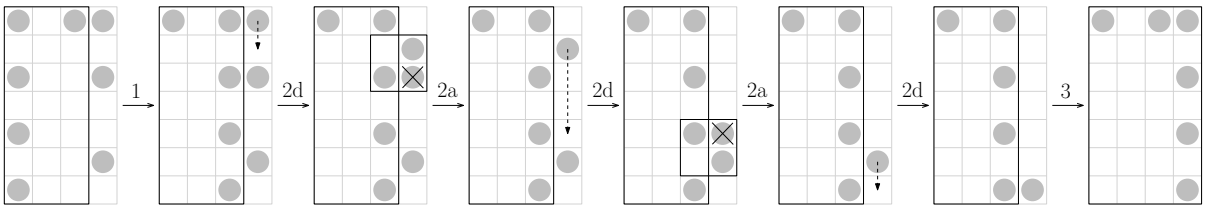

Figure 29. An example of a 3×7 'L' being incremented, to the right and at distance 1, into a 4×7 'L'. The numbers above the arrows refer to the three steps of the increment ("d"="drag", "a"="absorb").

Incrementing an 'L' to one of its sides is thus done in $O(N^2)$ actions found in $O(N^2)$ time. However, when we increment an 'L' multiple times in a row **to the same side**, we only re-orient it once (at the beginning). Indeed, as shown in Figure 28, the 'L' finishes the increment in the right orientation to be incremented again to the same side. Therefore any number of successive increments of an 'L' to the same side is still done in $O(N^2)$ actions.

Finally, let us verify that incrementing an 'L' to the side can be reversed. We know that re-orientations and leapfrogs are not an issue. The reverse of dragging towards the bottom is dragging towards the top, which is done in similar fashion (just take the symmetrics of the subroutines from Figures 24 and 25). All other actions used throughout are reversible, including absorptions as shown in Figures 26 and 27.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

(b) Incrementing an 'L' diagonally.

We will only consider the case of the bottom-right diagonal. Since we only increment diagonally when it is impossible to increment on the side, there is no coin to absorb: the only coins involved are the ones forming the 'L' and the one that we want to add.

1. If needed, we re-orient L so that it hugs the top and right sides of its hull. If L is odd, we also leapfrog to put the two adjacent coins at the bottom (resp. left) if the vertical branch is even (resp. odd).
2. We obtain a 'Z' shape that we transform back into an 'L' as in Figure 30, in $O(N)$ actions found in $O(N)$ time.

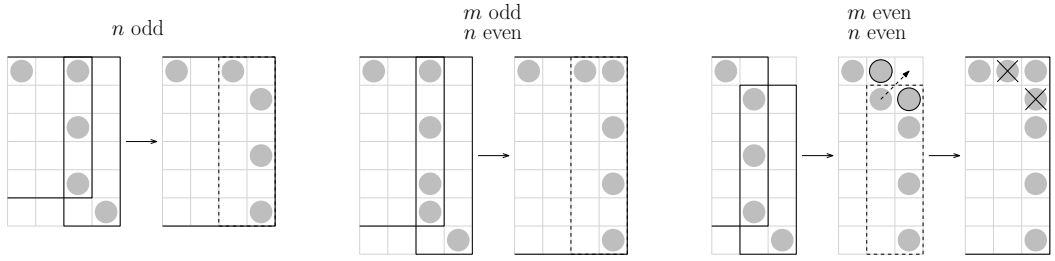

Figure 30. Incrementing an 'L' diagonally, all cases.

Incrementing an 'L' diagonally is thus done in $O(N^2)$ actions found in $O(N^2)$ time, and it can easily be reversed since it is a mixture of re-orientations/leapfrogs of 'L's and punctual reversible moves. □

Theorem 3.8. *Let A and B be configurations such that $|A| = |B|$, and suppose that:*

(a) There exist $a_1 \neq a_2$ in A such that:

(i) $\mathrm{span}(A \setminus \{a_1, a_2\}) \supseteq \mathrm{span}(B)$.

(ii) $\mathrm{span}(B) \cap R$ *is connected for each connected component R of* $\mathrm{span}(A \setminus \{a_1, a_2\})$.

(b) There exist $b_1 \neq b_2$ in B such that:

(i) b_1 has at least two neighbours in B.

(ii) b_2 has at least two neighbours in $B \setminus \{b_1\}$.

Then $A \xrightarrow{\mathscr{G}} B$. Moreover, setting $N = |A| = |B|$, there exists an algorithm that checks in $O(N^4)$ time whether these conditions are satisfied and finds a winning sequence of $O(N^3)$ moves in a further $O(N^3)$ time in the positive case.

Proof. We will see shortly (Corollary 3.21) that the span of any configuration C can be computed in $O(|C|^2)$ time. Condition (a) can thus be checked in $O(N^4)$ time, by computing $\mathrm{span}(B)$ then comparing it with $\mathrm{span}(A \setminus \{a_1, a_2\})$ that we compute for all choices of $\{a_1, a_2\}$, and in the positive case we get an explicit suitable pair $\{a_1, a_2\}$. As for condition (b), it is equivalent to B having either: a connected component of size at least 6, a connected component of size 5 that is not a star, a connected component of size 4 that is a cycle, or at least two connected components of size 3 or more. Therefore it is possible to check condition (b), and get an explicit suitable pair $\{b_1, b_2\}$ in the positive case, in $O(N)$ time.

Suppose we are in the positive case. Let L_A (resp. L_B) be the canonical configuration associated to $A \setminus \{a_1, a_2\}$ (resp. to B and $B \setminus \{b_1, b_2\}$). After *taking* the two extra coins a_1 and a_2, we apply the main direction of the previous theorem to reach L_A. By condition (a), each connected

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

component of span(L_A) contains at most one 'L' in L_B, therefore we can transform L_A into L_B by *shrinking* all 'L's. The reverse direction of the theorem allows us to reach $B \setminus \{b_1, b_2\}$, and finally condition (b) allows us to *drop* b_2 then b_1 at their correct positions to obtain B.
We now explain how to shrink an 'L'. This is achieved by *trimming* it in all four directions. For instance, trimming a canonical 'L' to the right is achieved as follows (see Figure 31):

1. If the 'L' is odd, we leapfrog so that the two adjacent coins are on the far right.
2. We make sure there is a coin c at the rightmost position that we want to keep, by *dropping* one there if needed.
3. We finish by simply *taking* all coins that are further right than c.

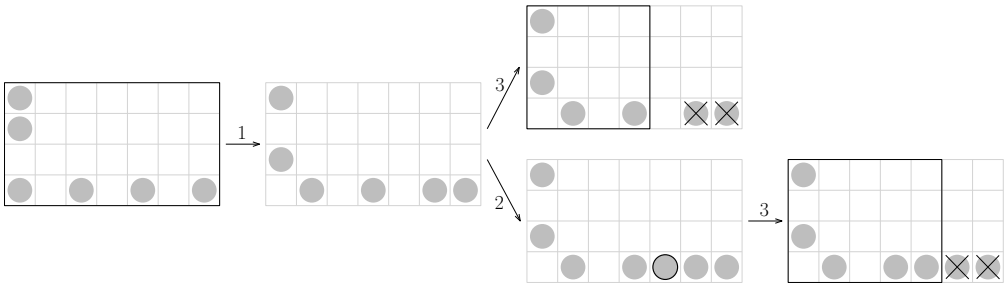

Figure 31. Trimming a 7×4 'L' to its right, making it 4×4 (top) or 5×4 (bottom). The numbers above the arrows refer to the three steps.

A canonical 'L' is ready to be trimmed at the top and to the right. We can then flip it so as to trim it at the bottom and to the left. Finally, we flip it back (and leapfrog if needed) to make it canonical again. Therefore, any 'L' L can be resized in $O(|L|^2)$ actions found in $O(|L|^2)$ time, so that the transition from L_A to L_B does not take more than $O(N^2)$ actions.
In conclusion, the winning sequence is made of $O(N^3)$ actions, which can be computed in $O(N^3)$ time once a_1 and a_2 have been found:

$$(A, 0) \xrightarrow{\mathscr{G}'} (A \setminus \{a_1, a_2\}, 2) \xrightarrow{\mathscr{G}'} (L_A, |A| - |L_A|) \xrightarrow{\mathscr{G}'} (L_B, |B| - |L_B|) \xrightarrow{\mathscr{G}'} (B \setminus \{b_1, b_2\}, 2) \xrightarrow{\mathscr{G}'} (B, 0)$$

Theorem 1.13 thus ends the proof. □

This algorithm can be applied to solve the puzzle from Figure 13 for instance (see Figure 32).

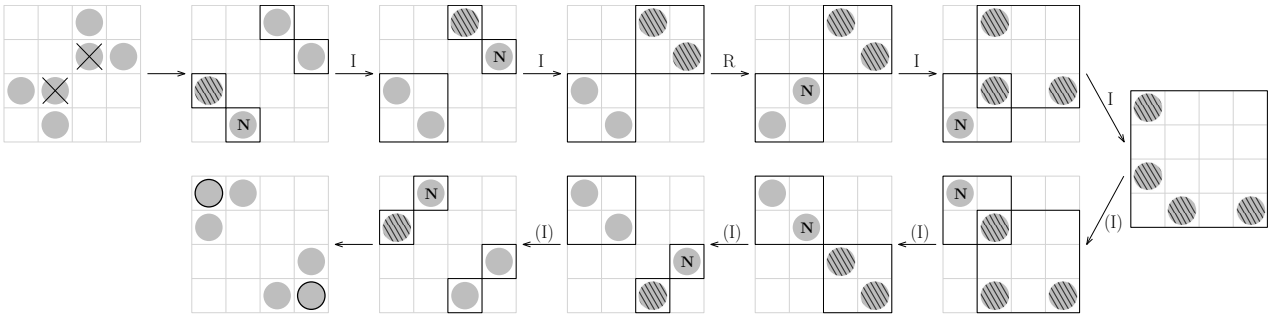

Figure 32. Solving the unlabelled version of the puzzle from Figure 13, step by step, routing through the canonical configuration on the right. The shaded coins represent the 'L' currently being augmented, i.e. L_1, whose next target is the coin marked "N". "I": increment, "(I)": reverse increment, "R": rotation of L_2.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Remark. There is a mistake in the version of the theorem that is mentioned in [1], where condition (a.ii) is omitted. This is because the span reduction from span(A) to span(B) cannot be done by simply *taking* all coins in $A \setminus \text{span}(B)$, since some of these coins could still be crucial in reaching certain parts of span(B), and we could therefore end up with a span strictly smaller than that of B. In fact, it is easy to think of solvable puzzles where $A \cap \text{span}(B)$ is empty. This is why we were careful when shrinking 'L's, making sure to safeguard certain rows and columns before removing coins. It turns out that, of all puzzles satisfying conditions (a.i) and (b) but not (a.ii), some are solvable and some are not, as illustrated by Figure 33. Concerning the puzzle on the right, splitting the coins into two dense far-apart groups proves impossible. The next proposition demonstrates this by brute force.

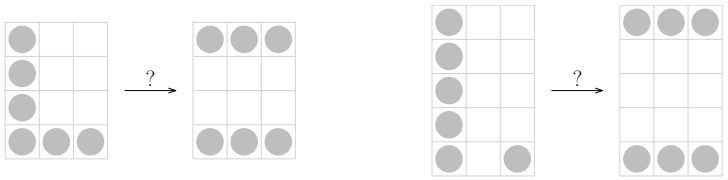

Figure 33. Two puzzles satisfying conditions (a.i) and (b) but not (a.ii). The one on the left is solvable in 12 moves. The one on the right is unsolvable according to Proposition 3.10.

Lemma 3.9. *Consider a checkerboard colouring of the square grid (it does not matter which half is black or white). If a configuration A contains at least two coins on white squares, then so does any configuration obtained from A.*

Proof. After each move, the number of coins on white squares either increases by 1, decreases by 1, or stays the same. Suppose that, at one point during the moves, there are exactly two coins on white squares: we need to show that no coin c can be moved from a white square to a black square at that moment. This is straightforward since black squares have all white neighbours, and therefore no black square can have two neighbouring coins other than c. □

Note that this lemma is a direct consequence of Proposition 1.3, since if only one coin b sits on a white square in B then $B \setminus \{b\}$ is entirely black and therefore has all connected components of size 1. However, it can be of visual help on the square grid.

Proposition 3.10. *The puzzle $A \xrightarrow{?} B$ on the right of Figure 33 is unsolvable.*

Proof. We proceed by reverse engineering, starting from B and finding all its ancestor configurations to show A is not one of them. A *backward move* consists of moving a coin that has at least two neighbouring coins to any unoccupied square on the board, with the following added rules:

1. We never step out of span(A), since such configurations could not descend from A.
2. We never move the same coin twice in a row.
3. There always needs to be a coin, other than the last moved coin, that has at least two neighbouring coins. Otherwise the descent would be over and the configuration obtained could not be A.
4. If there are exactly two coins on white squares, then we do not move any of them to a black square, because there would be no more hope of reaching A according to Lemma 3.9.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

If no move is possible under these rules, then we have reached a dead end i.e. a configuration that cannot have A as an ancestor. We could have done without Rule 4, but it does detect the use of Rule 3 one step in advance which simplifies the process a little. Figure 34 details all possibilities, up to symmetries and repetitions. In the first step for example, there are only three possible backward moves because of symmetries and Lemma 3.9.

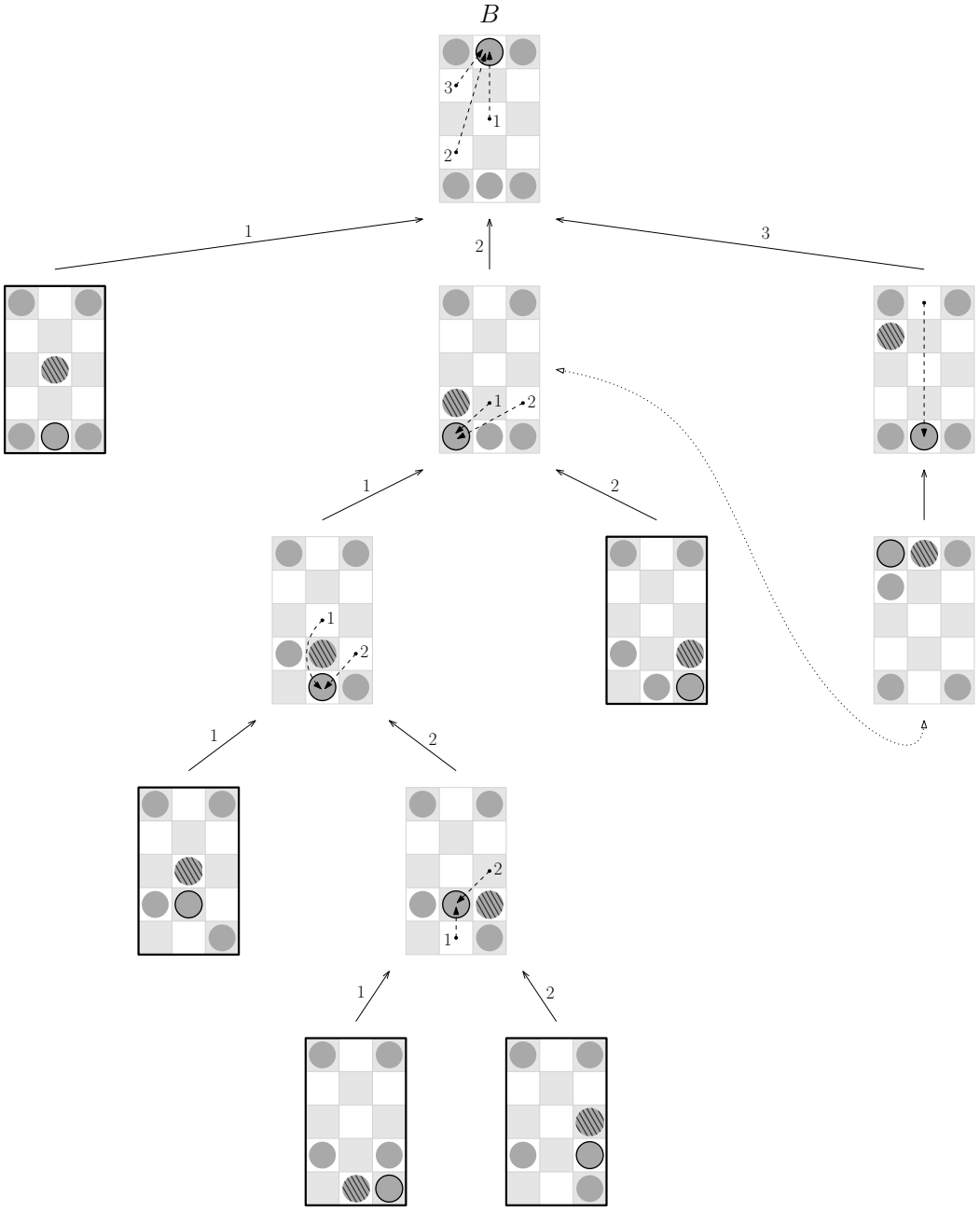

Figure 34. All possibilites for backward moves from B. The coin that we backward move is circled and the coin that has just been backward moved is shaded. Dead ends are outlined. The dotted double arrow on the right means that we recognize symmetrical configurations.

Looking at all dead ends in this diagram, we can see that neither A or any of its symmetrics is an ancestor of B, which concludes the proof and also brings out a lot of other starting configurations that would not work either. □

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Finally, puzzles satisfying conditions (a) and (b.i) but not (b.ii) also may or may not be solvable as shown in Figure 35.

Figure 35. Two puzzles satisfying conditions (a) and (b.i) but not (b.ii). The one on the left is solvable in 4 moves. The one on the right is unsolvable according to Proposition 1.3, even though the ending configuration is minimal+2 itself.

3.2.2 Labelled game

Labelled puzzles on the square grid are only briefly studied in [1]. We will go further in detail and get a result almost as strong as what we got for unlabelled puzzles.

Let A and B be labelled configurations with same cardinality N, whose unlabelled counterparts $\widetilde{A}$ and $\widetilde{B}$ satisfy the conditions of Theorem 3.8. We can run the algorithm to get a sequence of moves that gives $\widetilde{A} \xrightarrow{\mathscr{G}} \widetilde{B}$, then apply it on A and take note of the difference σ between where the labels end up and where we wanted them to be (i.e. where they are in B). The idea, much like we did for labelled puzzles on the triangular grid, is to prevent the mistake σ by grabbing an opportune moment during the moves to rearrange all the coins on board. This is most easily done when the coins are well connected. For this reason, looking at the way we perform $\widetilde{A} \xrightarrow{\mathscr{G}} \widetilde{B}$, the in-between stage when the configuration is canonical seems like a natural stopover to take advantage of. At that exact moment however, we are in the middle of a sequence in $\mathscr{G}'$, with coins in our reserve. Remember that this is incompatible with the labelled version of the game. Nevertheless, we are able to get around this problem, by slightly altering our path from $(\widetilde{A}, 0)$ to $(\widetilde{B}, 0)$ in $\mathscr{G}'$ (detailed at the very end of the proof of Theorem 3.8):

1. We do the first part of the sequence, namely $(\widetilde{A}, 0) \xrightarrow{\mathscr{G}'} (L_{\widetilde{A}}, |\widetilde{A}| - |L_{\widetilde{A}}|)$.
2. We then *drop* all the coins in our reserve, at successive positions given by Figure 36. We choose an ordering of the components of span(A), and we fill the empty positions in successive components: bottom row from left to right, then left column from bottom to top, then the remaining rectangle from left to right and bottom to top.
3. Now that our reserve is empty (and since we do not stack coins), we are back playing the original game $\mathscr{G}_l$, knowing exactly which coin sits where. We use this moment to permute all the coins we can.
4. Once this is done, we *take* all coins at the positions of the *drops* from step 2, only leaving the 'L's on board again.
5. Finally, we do the second part of the sequence, namely $(L_{\widetilde{A}}, |\widetilde{A}| - |L_{\widetilde{A}}|) \xrightarrow{\mathscr{G}'} (\widetilde{B}, 0)$.

It may look like we are blending several games here, but this is only conceptual, we are actually playing in $\mathscr{G}_l$ all the way: steps 1 and 2 combined define a sequence in $\mathscr{G}$ (and hence in $\mathscr{G}_l$ once the labels are accounted for) by Proposition 1.13, and so do steps 4 and 5 combined. With steps 1, 2, 4 and 5 already addressed, we need to explain what to do during step 3.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

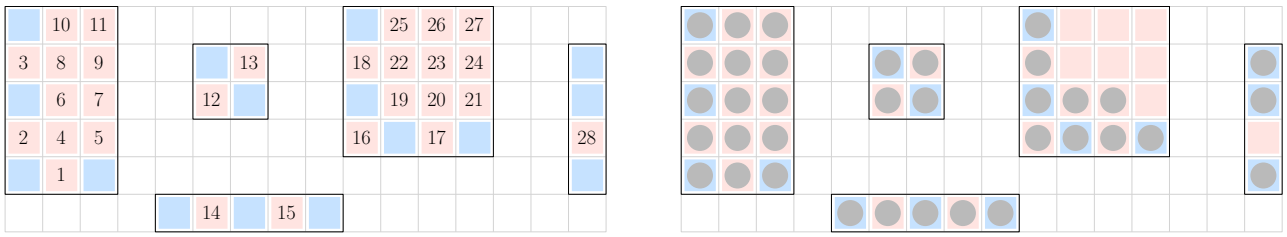

Figure 36. On the left: the ordering in which we fill in the gaps during step 2. On the right: an illustration of what we might have at the beginning of step 3 (20 coins have been successively *dropped* at the positions numbered 1 to 20).

We can immediately notice that certain coins are completely unmovable, hence the following necessary condition of solvability for labelled puzzles:

Proposition 3.11. *Suppose $L_{\widetilde{A}}$ contains a line 'L'. Let p be one end of this 'L'. Then A contains a coin c at position p, which can never be moved without decreasing the span. In consequence, if B contains a coin other than c at position p, then the puzzle $A \xrightarrow{?} B$ is not solvable.*

Proof. Since only one neighbouring position of p is in $\mathrm{span}(A)$ (or zero, in case of a 1×1 'L'), it is impossible to ever move a coin to p. This means that A necessarily contains a coin c at position p already, and after c is moved there will never be a coin at position p again. □

With that out of the way, we now seek sufficient conditions for permuting movable coins. We will focus on permuting positions instead of permuting labels, which is equivalent: in other words, we try to answer the question "Is it possible to exchange the coins at positions i and j?" rather than the question "Is it possible to exchange the coins labelled i and j?". We distinguish two types of positions: the *blue positions* that are part of the actual canonical 'L's (i.e. $L_{\widetilde{A}}$), and the *red positions* that are not. It will also be convenient to talk about *blue coins* and *red coins*, with each coin conceptually wearing the colour of the position it was on at the beginning of the current swap (unlike positions, coins are not inherently blue or red, their colours are updated after each swap). A *swap* is a basic unit of our algorithm:

– *red/red swap*: exchanging two given red coins; any other coin that moves during the process needs to end up back where it was at the start of the swap.
– *blue/red swap*: exchanging a given blue coin and a red coin of our choice; any other blue coin that moves during the process needs to end up back where it was at the start of the swap.

Indeed, we observe that any permutation of the coins is a combination of red/red and blue/red swaps: we can start with blue/red swaps to get the right coins on all the blue positions, and finish with red/red swaps to correct the labels on the red positions.

In what follows, we will always suppose that $N < |\mathrm{span}(A)|$, otherwise A would be congested and not a single move could me made. We introduce the following notations:

- $(C_j)_{1 \leq j \leq s}$ is our ordering of the components of $\mathrm{span}(A)$ chosen arbitrarily in step 2.
- $(p_i)_{1 \leq i \leq |\mathrm{span}(A)| - |L_{\widetilde{A}}|}$ is our ordering of the red positions defined in step 2.
- $r := N - |L_{\widetilde{A}}| \geq 2$ is the number of red coins (e.g. $r = 20$ in Figure 36). Before each swap, they occupy positions $p_1, \ldots, p_r$, while p_{r+1} is free with at least two neighbouring coins.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Proposition 3.12. *Any red/red swap is possible, in $O(1)$ moves found in $O(1)$ time.*

Proof. It suffices to show that the coin c_1 at position p_1 (the bottommost leftmost red position in C_1) can be swapped with the coin c_i at position p_i for any $i \in [\![2, r]\!]$. Say C_1 is of size $m \times n$.

- Suppose p_1 and p_{r+1} are not adjacent. It is then possible to swap c_1 with c_i in 3 moves as in Figure 37. Note that move 1 is only possible because p_1 is not a neighbour of p_{r+1}. Moves 2 and 3 are always possible, since the two neighbouring positions of p_1 (resp. p_i) that were occupied at the beginning are also occupied when move 2 (resp. move 3) happens.

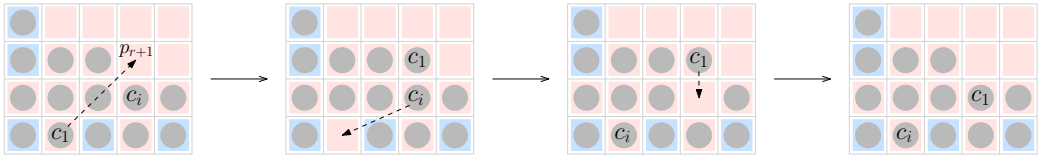

Figure 37. Swapping c_1 with c_i when p_1 and p_{r+1} are not adjacent. It is not important that p_1, p_i and p_{r+1} are in the same component.

- Suppose p_1 and p_{r+1} are adjacent. Then p_{r+1} is necessarily the top neighbour of p_1, since the others can only be blue. Moreover, we know that $m \geq 3$ and that either m or n is odd, otherwise the top neighbour of p_1 would be blue (see Figure 38).

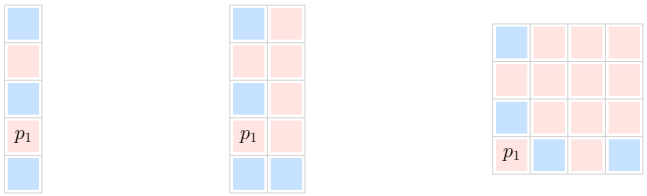

Figure 38. Position of p_1: if $m = 1$ (left), if $m = 2$ (middle), if m and n are even (right). The top neighbour of p_1 is blue in all three cases.

All neighbours of p_1 are thus known: its left neighbour is the bottom-left corner of C_1 which is blue, its right neighbour is also blue, and its top neighbour is p_{r+1} (in particular p_i is not a neighbour of p_1). There are only two cases left:

(a) Suppose p_i is not to the direct left of p_{r+1}. Then p_i is not a neighbour of p_{r+1}, because that would contradict our ordering of the red positions. We can therefore swap c_1 and c_i in 3 moves as in Figure 39. Move 1 (resp. move 2) is possible because p_i is not a neighbour of p_{r+1} (resp. of p_1).

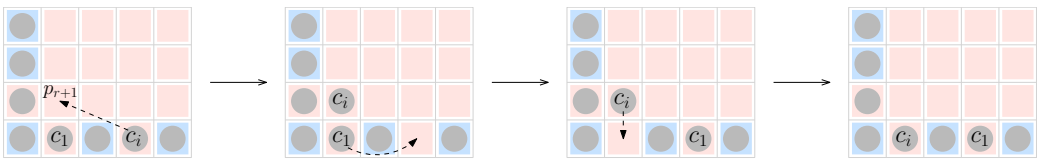

Figure 39. Swapping c_1 with c_i when p_1 and p_{r+1} are adjacent, case (a).

(b) Suppose p_i is to the direct left of p_{r+1}. Then the top neighbour of p_i is necessarily blue, and Figure 40 shows how to swap c_1 with c_i in 5 moves. All positions that appear unoccupied in Figure 40 indeed are, thanks to our ordering of the red positions.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

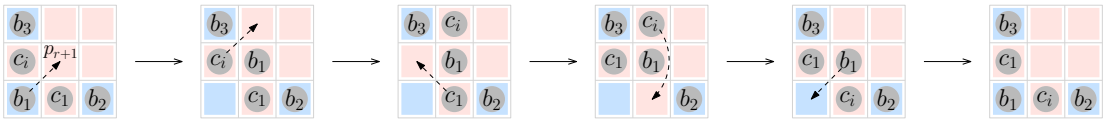

Figure 40. Swapping c_1 with c_i when p_1 and p_{r+1} are adjacent, case (b). □

Proposition 3.13. *If $r \geq 3$, then any blue/red swap is possible, as long as the blue coin c is not an end of a line 'L'. This is done in $O(|L|)$ moves found in $O(|L|)$ time if c is an end of an 'L' L, and in $O(1)$ moves found in $O(1)$ time otherwise.*

Proof. Let p be the (blue) starting position of c. We know p is either part of the bottom row or the left column of its component, so we may assume whichever since both work the same way.

- Case (I): c is not an end of an 'L'. Assume c is in the bottom row. The idea is to flank c with two coins, so as to extract it without losing span and replace it with a third coin.

  (a) Suppose c already has two coins adjacent to it. More specifically, given our ordering of the red positions, c has a coin c_1 to its left and a coin c_2 to its right (one of them might be blue in the case of an odd 'L', but this is not important).

  (i) Suppose p_{r+1} is not a neighbour of p. Then c can instantly be moved to p_{r+1}, and since $r \geq 3$ there exists a red coin $c' \notin \{c_1, c_2\}$ that can be swapped with c in 3 moves as in Figure 41.

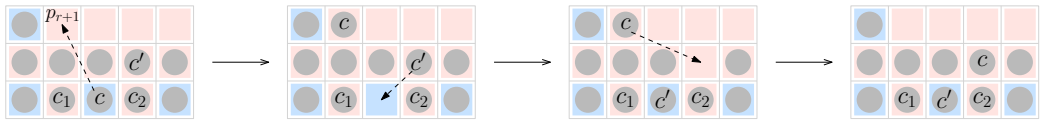

Figure 41. Blue/red swap of c and c' when $r = 3$, case (I.a.i).

  (ii) Suppose p_{r+1} is a neighbour of p (necessarily its top neighbour) and there is a red coin $c' \notin \{c_1, c_2\}$ whose position p_i is not a neighbour of p_{r+1}. Using the fact that p_i and p are not adjacent (all neighbours of p being known already), we swap c with c' in 3 moves as in Figure 42.

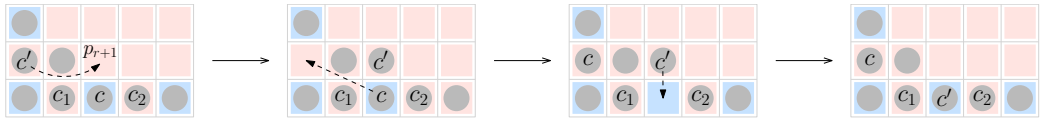

Figure 42. Blue/red swap of c and c' when $r = 3$, case (I.a.ii).

  (iii) Suppose p_{r+1} is a neighbour of p and all red coins other than c_1 and c_2 are on positions that are adjacent to p_{r+1}. Given our ordering of the red positions, the top and right neighbours of p_{r+1} are necessarily unoccupied. Therefore $r = 3$ and the positions of the three red coins are exactly known: left of p (occupied by c_1), right of p (occupied by c_2), left of p_{r+1} (occupied by some coin c'). For this to be possible, the component needs to be of size exactly 5×2. We swap c with c' in 5 moves as in Figure 43.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

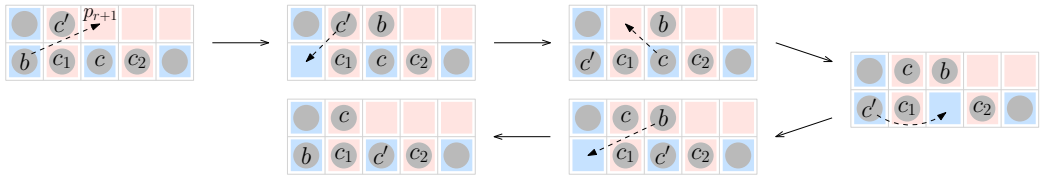

Figure 43. Blue/red swap of c and c' when $r = 3$, case (I.a.iii).

(b) Suppose c has exactly one coin c_1 adjacent to it: c has c_1 to one side and an unoccupied position p' to the other. Let c_2 and c' be red coins other than c_1: given our ordering of the red positions, there cannot be any coin in the component of p outside the left column and bottom row, hence the positions of c_2 and c' are not neighbours of p or p'. We can therefore swap c with c' in 4 moves as in Figure 44.

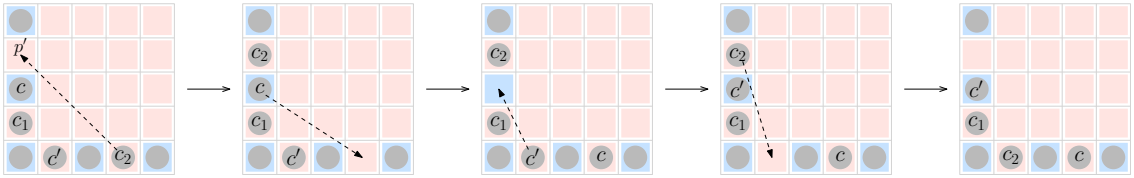

Figure 44. Blue/red swap of c and c' when $r = 3$, case (I.b).

(c) Suppose c has no coin adjacent to it. We then bring a red coin c_1 next to c, perform the subroutine from Figure 44, and bring c_1 back where it started.

- Case (II): c is one end of an $m \times n$ 'L' that is not a line 'L'. An important remark is that showing that c can be swapped with a blue coin of our choice is enough, provided it is not the other end of the 'L'. Indeed, suppose that c can be swapped with such a blue coin c_b, then we can exchange c with a red coin c_r by swapping successively: c_b with c_r (blue/red, case (I)), c with c_r (blue/blue, only c is an end), c with c_b (blue/red, case (I)).

  (a) If c has two coins adjacent to it, then we can proceed as in Figure 41.

  (b) Assume c is the top-left coin and its right neighbour is unoccupied. We will still need to flank c with two coins eventually, otherwise we could not extract c without losing span. However, the position to the immediate right of c is not accessible straightaway, so we reach it using a *scaffolding* method. Let p_0 be the position of the highest coin in the second column from the left (if there is none, we define p_0 as the bottommost position of that column), then p_0 defines the height at which we will start building:

  1. If p_0 is not in the bottom row, then p_0 and the left column are necessarily filled with coins already. If p_0 is in the bottom row, then we bring one or two red coins if needed at the positions indicated by Figure 45.

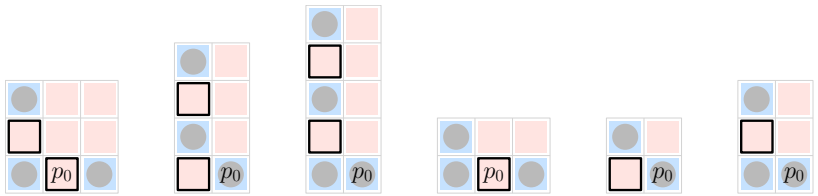

Figure 45. Where to bring red coins in step 1, all cases. The three cases on the right are only used if $n = 2$ or if $n = 3$ and $m = 2$.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

2. We climb all the way up the left branch as in Figure 46.

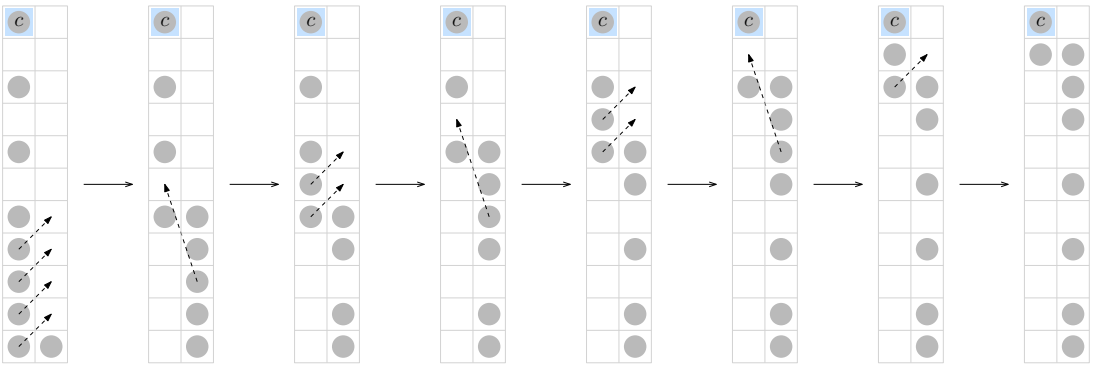

Figure 46. Building scaffolding. Only the two leftmost columns and the rows from p_0 and above are represented.

3. Now that we have two coins to the immediate bottom and bottom-right of c, we can perform the swap as in Figure 47. In the case where $n = 2$ or $n = 3$, we use the fact that the two leftmost columns contain at most two red coins in total, which guarantees the existence of a red coin c_1 from another column (or another component) that necessarily still has two neighbouring coins at this point. In the case $n \geq 4$, the coin c' that we swap c with is red if m is odd and n is even, otherwise it comes from the blue position just under p and is thus blue.

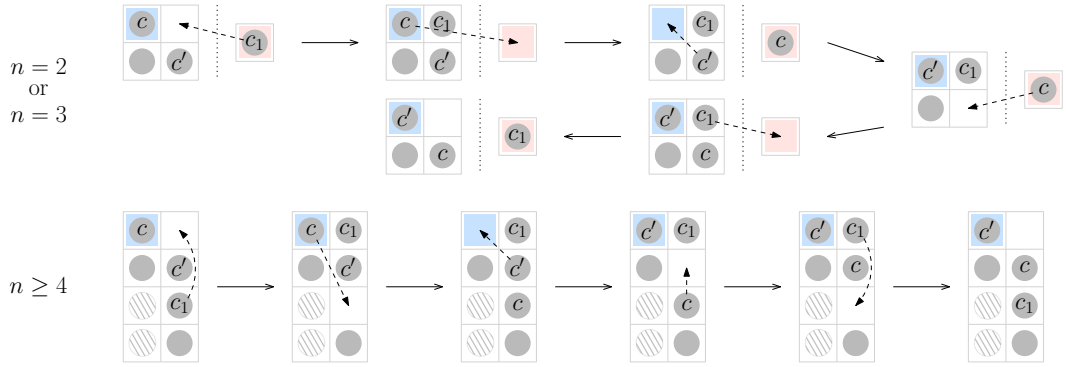

Figure 47. Blue/red swap when $r = 3$, case (II). The uncoloured squares might be red or blue depending on the exact 'L'. A shaded coin means there might or might not be a coin at this position.

4. Finally, we undo steps 1 and 2 thanks to the fact they use all reversible moves. The only coin for which the reversibility might be unclear is the one on p_0 during step 2, but it actually does not move unless we are in the leftmost situation of Figure 45, in which case we see that p_0 has (at least) two neighbouring coins at the beginning of step 2.

Notice how, in the case $n \geq 4$, we never use more than two red coins throughout the scaffolding routine: we will thus be able to use the same method when $r = 2$. □

Because of the uncertainty concerning the total number of coins, the previous two proofs were particularly tedious. We now consider the case $r = 2$, where there will be no more obstacles on board, nothing but the 'L's and the two red coins.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Proposition 3.14. *If $r = 2$, then any blue/red swap is possible, unless perhaps the blue coin is:*
- *an end of a line 'L';*
- *the middle coin of a 1×5 or 5×1 'L';*
- *one of the two coins of a 2×2 'L' (even though these two can be exchanged together).*

This is done in $O(|L|)$ moves found in $O(|L|)$ time if the blue coin is an end of an 'L' L, and in $O(1)$ moves found in $O(1)$ time otherwise.

Proof. Let c be the blue coin that we intend to swap with a red coin and let p be its starting (blue) position. Let L be the $m \times n$ 'L' that p is part of. Let c_1 and c_2 be the two red coins. In our swapping routines, we will always omit the first two (at most) moves made on the red coins to put them where we want, as well as the final two (at most) moves made on the new red coins to put them back to p_1 and p_2. Let p_0 be a red position that, once c_1 and c_2 have been placed where we want, is unoccupied and has at least two occupied neighbours (e.g. $p_0 = p_{r+1} = p_3$ if we do not change the starting positions of c_1 and c_2), then p_0 may act as a default destination for the first blue coin we move.

- Case (I): c is not an end of L.

  (a) We start off with 'L's of small sizes. Figures 48 and 49 feature 9 examples.

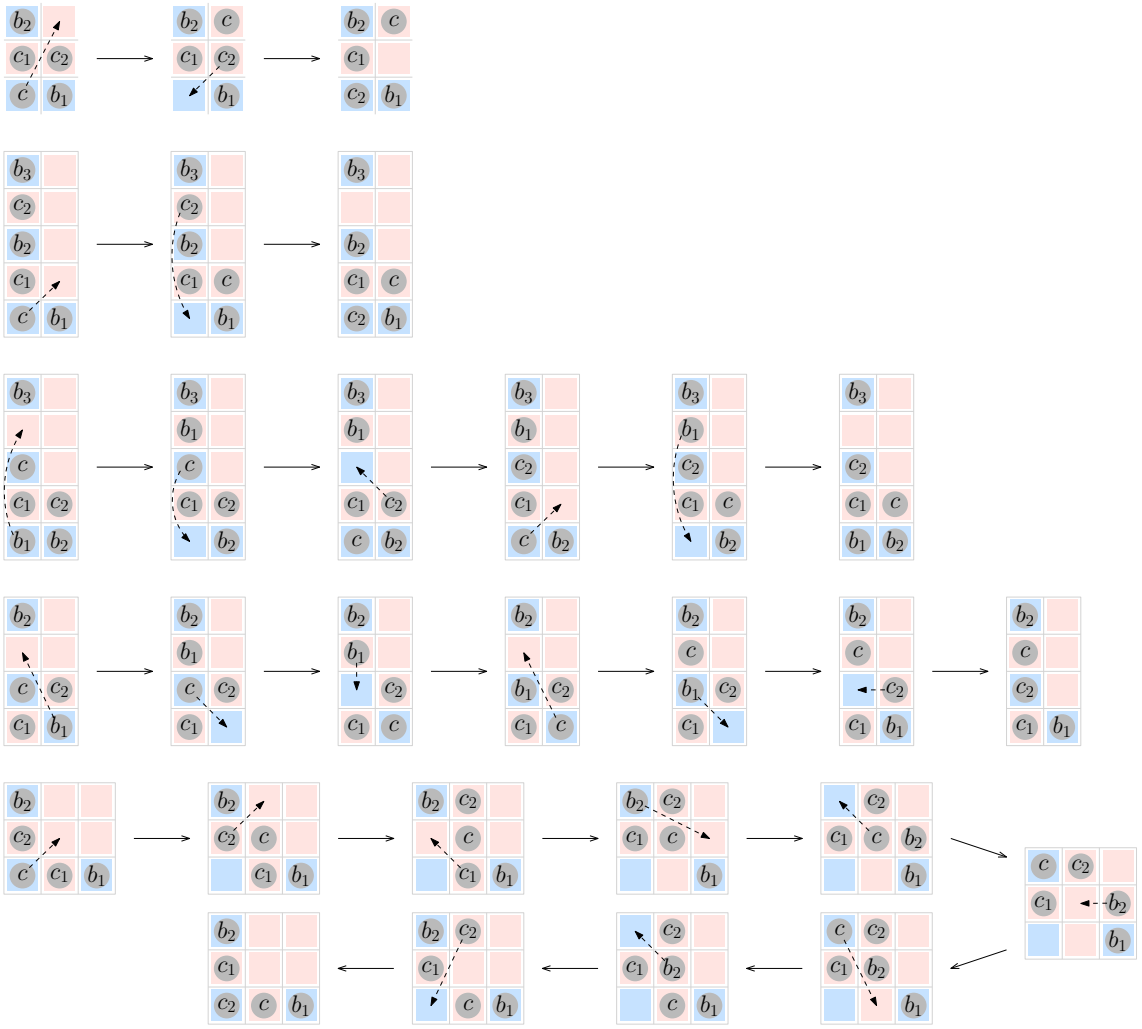

Figure 48. Blue/red swap of c and c_2 when $r = 2$, case (I), for 'L's of size: 2×3, 2×5, 2×4, 3×3.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

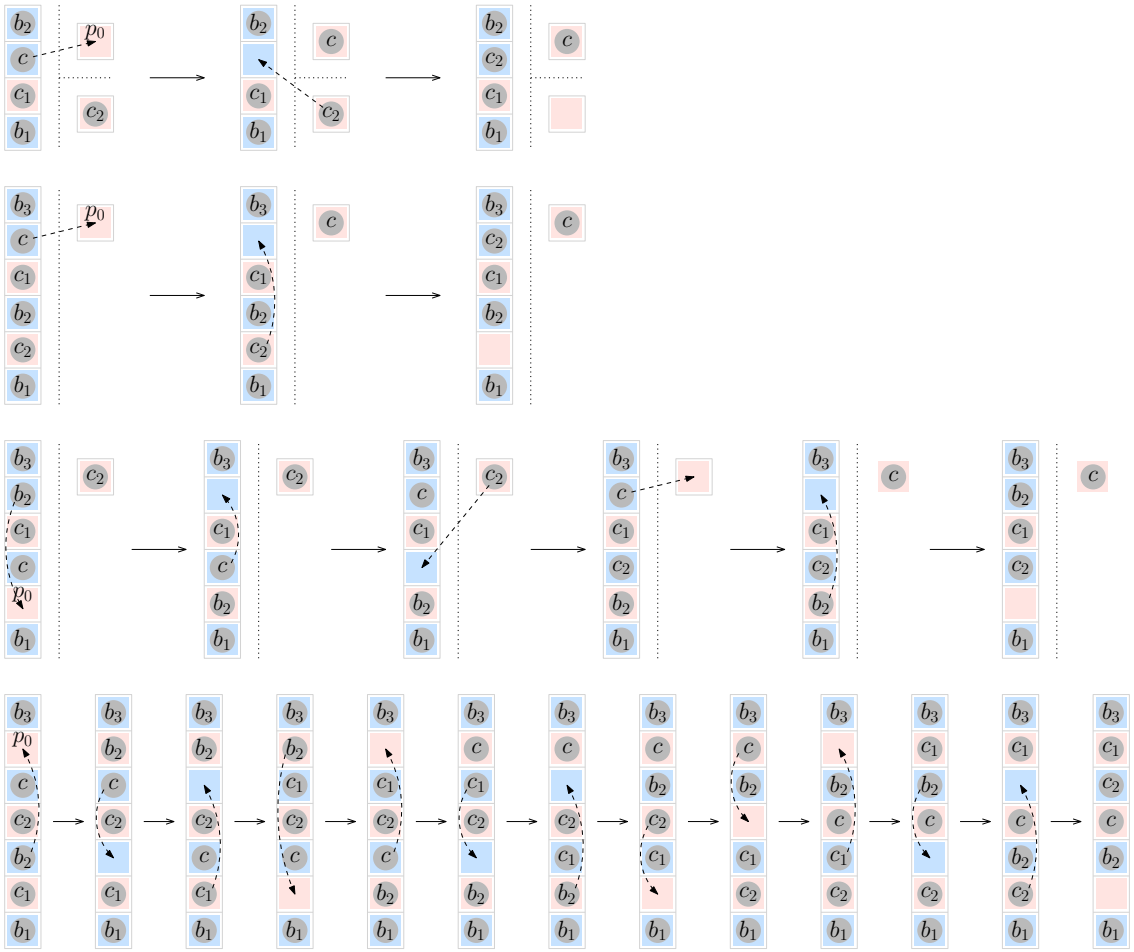

Figure 49. Blue/red swap of c and c_2 when $r = 2$, case (I), for 'L's of size: 1×4, 1×6, 1×7. Other components get involved when the 'L' is too small.

(b) It is easy to see that the previous 9 examples actually cover all possibilities, even for larger 'L's, with the notable exception of 1×5 and 5×1 'L's. Indeed, the routines for line 'L's from Figure 49 can also be applied for coins that form an angle as long as the spacing is the same (see Figure 50 for an example).

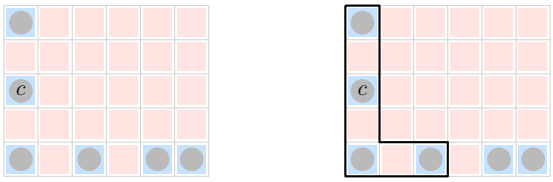

Figure 50. The moves for the 1×7 'L' can be applied to the outlined region.

- Case (II): c is an end of L, and L is not a line 'L'. Assume c is the top-left corner of L. Using the same argument as in the proof of Proposition 3.13, it suffices to show that c can be swapped with a blue coin of our choice, provided it is not the other end of L.
  (a) Suppose $n \geq 4$. This case is easy: we can use the same scaffolding method as in the proof of Proposition 3.13, since it only needed two red coins to work.
  (b) Suppose $n = 2$ or $n = 3$. Figure 51 addresses all cases, with the exception of 2×2 'L's whose two coins can nevertheless be exchanged together as in Figure 52.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

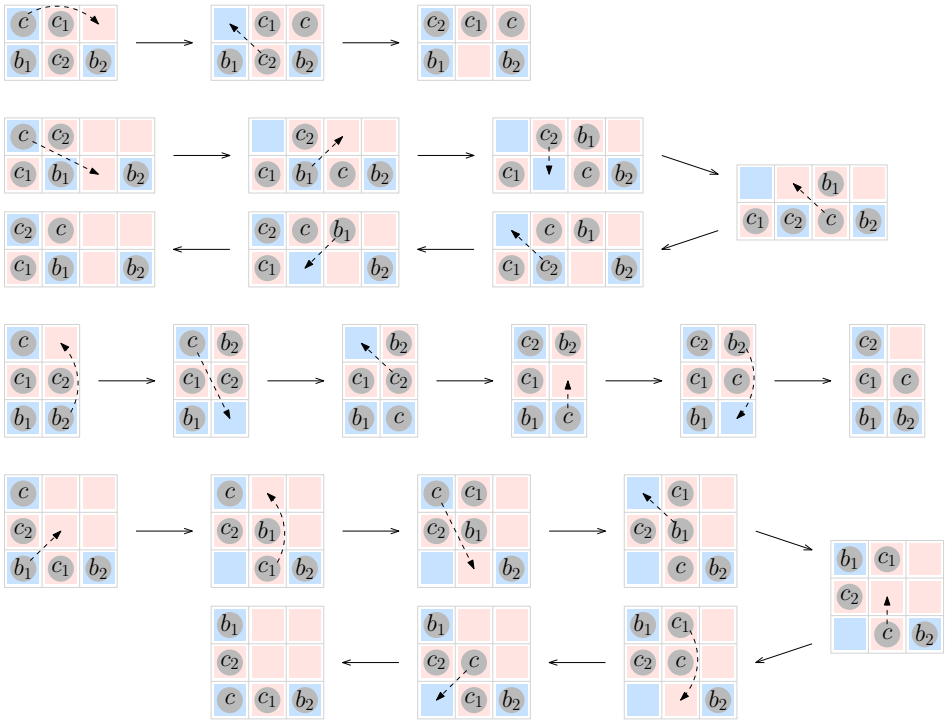

Figure 51. Blue/red swap of c and c_2 when $r = 2$, case (II), $n = 3$ or $n = 4$. For the 3×3 'L', we perform a blue/blue swap instead.

Figure 52. Exchanging the two coins in a 2×2 'L'. □

With only two red coins, we cannot hope to correct the labels in 2×2, 1×5 or 5×1 'L's in general. Figure 53 features examples of labelled configurations that are almost deadlocks, and in the next section we will see a more general negative result for 1×5 or 5×1 'L's in the case where all components are even (Proposition 3.31). However, if there is an odd component anywhere on board that contains at least one red position, then the next proposition shows that these restrictions disappear, thanks to one of the two adjacent coins having a lot of mobility and basically playing the part of a third red coin.

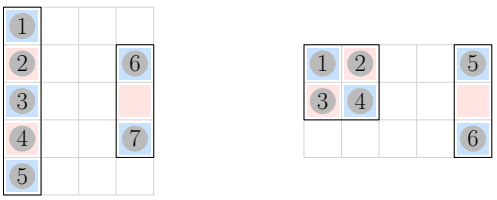

Figure 53. In the left configuration, coin 3 can be moved but only to be brought back immediately. In the right configuration, after coin 1 is moved for instance, we can only bring it back or exchange it with coin 4.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Proposition 3.15. *If $r = 2$ and at least one of the components is odd of size other than 1×2 or 2×1, then we have the same result as in the case $r = 3$ (Proposition 3.13).*

Proof. We only have to check the blue/red swaps in 2×2, 1×5 and 5×1 'L's. Using the fact that there exists a red position that has a blue coin to one side and two adjacent blue coins to another, this is easily done as per Figure 54.

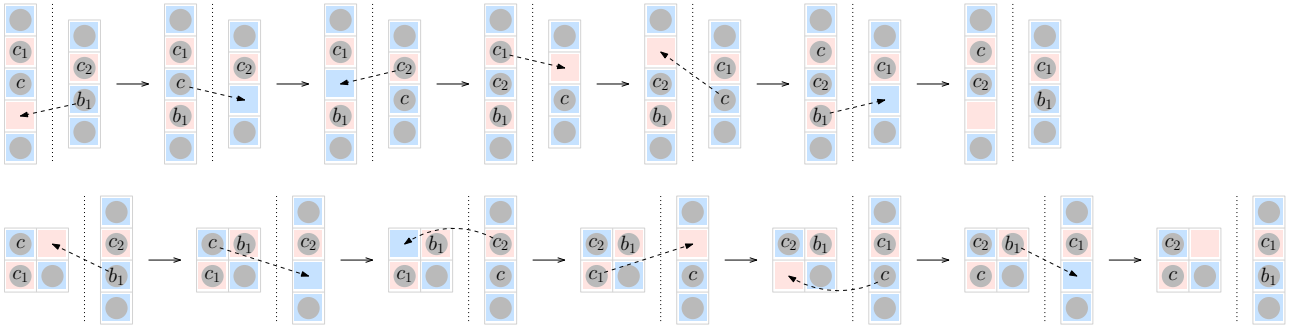

Figure 54. Blue/red swap of c and c_2 when $r = 2$ in presence of an odd component (only the missing cases). □

The following theorem sums up what we know about labelled puzzles on the square grid.

Theorem 3.16. *Let A and B be labelled configurations whose unlabelled counterparts $\widetilde{A}$ and $\widetilde{B}$ satisfy the conditions of Theorem 3.8. We set $r = |A| - |L_{\widetilde{A}}|$ as before. Suppose that $|A| < |\mathrm{span}(A)|$ and that, in step 3 of the algorithm described at the beginning of this section, the desired permutation does not involve:*

- *If $r = 2$ and all 'L's in $L_{\widetilde{A}}$ are even: end coins of line 'L's, central coins of 1×5 and 5×1 'L's, coins of 2×2 'L's (except for exchanging the two coins of the same 2×2 'L').*
- *If $r \geq 3$ or $L_{\widetilde{A}}$ contains an odd 'L': end coins of line 'L's.*

Then $A \xrightarrow{\mathscr{G}_l} B$. Moreover, the labelled puzzle is solved in a further $O(N)$ moves found in further $O(N)$ time compared with what is needed to solve the unlabelled version of the puzzle.

Proof. Only the final assertion remains to be shown. Once the algorithm for unlabelled puzzles has been run, we know what permutation we need to achieve during step 3, and we have to decompose it as a product of red/red and blue/red swaps in a clever manner, taking into account the fact that blue/red swaps involving an end of an 'L' are costlier. The ends of the bigger 'L's need to be involved in as few swaps as possible. Let $p_1, \ldots, p_k$ be the end positions of all non-line 'L's, ordered by decreasing size of the corresponding 'L's, i.e. $|L_1| \geq |L_2| \geq |L_3| \geq \ldots$ where L_i denotes the 'L' that contains p_i. We proceed as follows:

1. First of all, we correct all labels at positions $p_1, \ldots, p_k$ successively. Let us consider p_1: if the coin that needs to end up at position p_1 is currently red then we place it at p_1 at a cost (moves and time) $O(|L_1|)$, but if it is currently blue then we need to swap it with a red coin first which costs $O(|L_2|)$ in the worst case scenario, so the total cost for p_1 is $O(|L_1| + |L_2|)$. We now consider p_2: we know that p_1 will not be involved, because it has been corrected already, therefore the same reasoning yields a cost of $O(|L_2| + |L_3|)$ for p_2, etc. In total, this first step is therefore done in $O(\sum_{i=1}^{k} |L_i|) = O(N)$ moves found

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

in $O(N)$ time. We use the fact that $\sum_{i=1}^{k} |L_i| = O(N)$, which will soon become evident during our study of minimum configurations on the square grid.

2. We finish by correcting all other positions. This is done in $O(N)$ red/red and blue/red swaps, each of which is done in $O(1)$ moves found in $O(1)$ time.

Note that a systematic use of scaffolding methods (even for coins that are not ends of 'L's), as described in [1], would also work but would need $O(N^2)$ moves instead. □

Corollary 3.17. *Let A and B be labelled configurations with same span S, whose unlabelled counterparts satisfy the conditions of Theorem 3.8. Suppose that $|A| < |S|$ and that:*

- *S has no components of size 2×2, 1×5 or 5×1.*
- *For each end p of a $1 \times n$ or $m \times 1$ component of S, there is the same coin at position p in A and in B.*

Then $A \xrightarrow{\mathscr{G}_l} B$.

The puzzle from Figure 13 is an example of a solvable labelled puzzle on the square grid.

3.3 One extra coin: minimum+1 configurations

We now look at puzzles where there is only one extra coin, starting with the case where A is *minimum+1* i.e. consists of a minimum configuration with same span as A plus a single coin.

3.3.1 The virus problem

In the classic *virus problem* from folklore, a certain set C of squares in a rectangle R are initially contaminated, and at each time step all healthy squares with at least two contaminated neighbours catch the virus. The goal is to find the minimum cardinality of a set C such that R gets entirely contaminated. Since the set of all squares that end up contaminated is precisely $\text{span}(C)$, this is equivalent to finding the cardinality of minimum configurations with span R on the square grid. Given a configuration C on the square grid, a *step-by-step construction* of $\text{span}(C)$ is a sequence $(C = C_0, C_1, \ldots, C_s = \text{span}(C))$ satisfying $C_i = C_{i-1} \cup \{p_i\}$ for all i where $p_i \in \text{Adj}(C_{i-1})$, which we can see as an order of contagion of the squares. The virus problem is solved by considering the perimeter of the contaminated area.

Definition 3.18. The *perimeter* of a configuration C, denoted by $\text{Perim}(C)$, is the perimeter of the union of squares defined by C, where each square is considered unit. Equivalently, the perimeter is given by the formula $\text{Perim}(C) = 4|C| - 2|E(G[C])|$.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Lemma 3.19. *In the virus problem, the perimeter of the contaminated area never increases during the contagion. In particular, every configuration C satisfies* $\mathrm{Perim}(C) \geq \mathrm{Perim}(\mathrm{span}(C))$*, and if* $\mathrm{Perim}(C) = \mathrm{Perim}(\mathrm{span}(C))$ *then any step-by-step construction of the span is such that each square is contaminated by exactly two neighbours and not more.*

Proof. Let $(C = C_0, C_1, \ldots, C_s = \mathrm{span}(C))$ be a step-by-step construction of $\mathrm{span}(C)$. For all i, we have $\mathrm{Perim}(C_i) = \mathrm{Perim}(C_{i-1}) + 4 - 2n_i$ where $n_i \geq 2$ is the number of neighbours of p_i in C_{i-1}, hence $\mathrm{Perim}(C_i) \leq \mathrm{Perim}(C_{i-1})$ for all i. In particular $\mathrm{Perim}(C) \geq \mathrm{Perim}(\mathrm{span}(C))$, and if there is equality then $\mathrm{Perim}(C_i) = \mathrm{Perim}(C_{i-1})$ i.e. $n_i = 2$ for all i. □

Proposition 3.20. *Let R be an $m \times n$ rectangle, then minimum configurations with span R have cardinality $\lceil \frac{m+n}{2} \rceil$. In particular, canonical configurations are minimum.*

Proof. Since 'L's have cardinality $\lceil \frac{m+n}{2} \rceil$, we just have to show that any configuration C with span R satisfies $|C| \geq \lceil \frac{m+n}{2} \rceil$. By Lemma 3.19: $4|C| \geq \mathrm{Perim}(C) \geq \mathrm{Perim}(R) = 2(m+n)$. □

Corollary 3.21. *Any configuration C satisfies* $|\mathrm{span}(C)| \leq |C|^2$*. In particular, the span of a configuration C can be computed in $O(|C|^2)$ time.*

Proof. Write $\mathrm{span}(C) = \bigcup_{i=1}^{s} R_i$ where the R_i are $m_i \times n_i$ rectangles at distance at least 3 from each other. Set $k_i := |C \cap R_i|$, Proposition 3.20 ensures that $k_i \geq \lceil \frac{m_i+n_i}{2} \rceil$ hence $m_i n_i \leq \frac{(m_i+n_i)^2}{4} \leq k_i^2$. Since $|C| = \sum_{i=1}^{s} k_i$, we get $|\mathrm{span}(C)| = \sum_{i=1}^{s} m_i n_i \leq \sum_{i=1}^{s} k_i^2 \leq |C|^2$. The statement on the complexity follows by Proposition 1.8. □

3.3.2 Structure of minimum configurations

Even though the virus problem is originally only about the cardinality of minimum configurations, we now explain how the key Lemma 3.19 also provides a lot of information about their structure.

Proposition 3.22. *Let M be a minimum configuration with rectangular span R.*

- *If R is even, then* $\mathrm{Perim}(M) = \mathrm{Perim}(R)$ *and all connected components of M are of size 1.*
- *If R is odd, then: either* $\mathrm{Perim}(M) = \mathrm{Perim}(R) + 2$ *and all connected components of M are of size 1, or* $\mathrm{Perim}(M) = \mathrm{Perim}(R)$ *and all connected components of M are of size 1 except for a single one of size 2.*

Proof. If R is of size $m \times n$, we know from Lemma 3.19 that $\mathrm{Perim}(M) \geq \mathrm{Perim}(R) = 2(m+n)$.

- If R is even, then $|M| = \frac{m+n}{2}$ hence $\mathrm{Perim}(M) = 2(m+n) - 2|E(G[M])$, so $|E(G[M])| = 0$ which means all connected components of M are of size 1.
- If R is odd, then $|M| = \frac{m+n+1}{2}$ hence $\mathrm{Perim}(M) = 2(m+n) + 2 - 2|E(G[M])$, so either: $|E(G[M])| = 0$ i.e. all connected components of M are of size 1, or $|E(G[M])| = 1$ i.e. all connected components of M are of size 1 except for a single one of size 2. □

40 This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Definition 3.23. Let M be a minimum configuration and R be an $m \times n$ rectangle. We say R is *full in* M if $\text{span}(M \cap R) = R$. In particular, it implies that $|M \cap R| = \lceil \frac{m+n}{2} \rceil$.

Proposition 3.24. *In a minimum configuration, there cannot be a coin at distance 1 from an odd full rectangle.*

Proof. Suppose for a contradiction that there exists $c \in M$ at distance 1 from an odd full rectangle R. Let $m \times n$ be the size of R. Since $M \cap R$ and $(M \cap R) \cup \{c\}$ are both minimum, with respective spans R and $R(R \cup \{c\})$, we have $|M \cap R| = \lceil \frac{m+n}{2} \rceil = \lceil \frac{m+n+1}{2} \rceil = |(M \cap R) \cup \{c\}|$ because $m + n$ is odd. This is a contradiction. □

Definition 3.25. Let $P = (c_0, \ldots, c_l)$ be a 2-path on the square grid.

- We say P is *direct* if the l jumps from c_i to c_{i+1}, put together, take at most one horizontal direction (left or right) and at most one vertical direction (up or down). Each individual jump takes just one direction, apart from the diagonal jumps that take two at once. For instance, an 'L' forms a direct 2-path between its end-points. Obviously, if we orient P the other way, all directions are reversed and the definition is unchanged.
- We say P is a *shortest 2-path* if it is a shortest path from c_0 to c_l in the graph obtained from the square grid by adding edges between vertices at distance 2 from each other.

Proposition 3.26. *In a minimum configuration, direct 2-paths and shortest 2-paths coincide.*

Proof. A 2-path between c and c' is a shortest 2-path if and only if it is of length $\lceil \frac{\text{dist}(c,c')}{2} \rceil$. Moreover, a shortest 2-path is always direct. Let $P = (c_0, \ldots, c_l)$ be a direct 2-path in M, we want to show that P is a shortest 2-path. Let $R = R(\{c_0, c_l\})$: if R is of size $m \times n$ then $m + n = \text{dist}(c_0, c_l) + 2$. We know that $\text{span}(P)$ is a rectangle containing c_0 and c_l because P is a 2-path, moreover $\text{span}(P)$ is included in R because P is direct, therefore $\text{span}(P) = R$. Since P is minimum, we conclude that $|P| = \lceil \frac{m+n}{2} \rceil = 1 + \lceil \frac{\text{dist}(c_0,c_l)}{2} \rceil$ i.e. P is a shortest 2-path. □

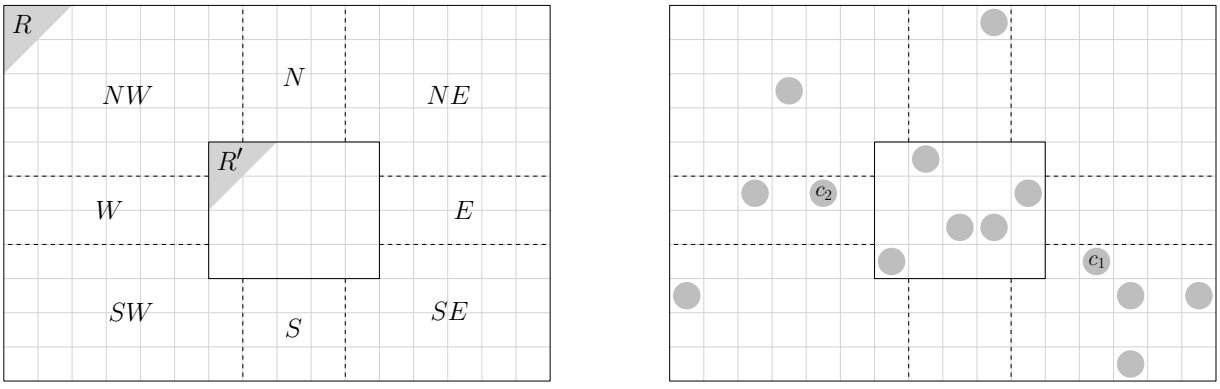

Figure 55. The 8 regions from Proposition 3.28 are shown on the left. Note that, for instance, if R' is of size $m' \times 1$ then $W = \varnothing$ and $SW \cap NW \neq \varnothing$. A possible minimum configuration M is given on the right. Here, c_1 is the exit out of R' to the bottom-right, c_2 is the exit out of R' to the left, and there is no path coming out of R' towards the top.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Definition 3.27. Let R' be a rectangle. We say a 2-path $P = (c_0, \dots, c_l)$ *comes out of* R' if P is disjoint from R' and $\mathrm{dist}(c_0, R') \leq 2$.

Proposition 3.28. *Let M be a minimum configuration with rectangular span R. Suppose that either: R is even, or R is odd and M contains a pair of adjacent coins. Then:*

- *All 2-paths in M are shortest 2-paths. In particular, the coins in M form a forest in terms of the 2-connectivity.*
- *Let $R' \subset R$ be a full rectangle in M. We define 8 regions on board as in Figure 55. Let $P = (c_0, \dots, c_l) \subseteq M$ be a 2-path coming out of R', then we have the following (and the corresponding results for the other directions hold in a symmetrical manner):*
  - *If $c_0 \in N$, then P is a straight line towards the top.*
  - *If $c_0 \in NE \setminus \{SE, NW\}$, then P goes towards the top, right or top-right.*
  - *If $c_0 \in NE \cap SE$ (necessarily R' is of size $m' \times 1$ and c_0 is to the right of R'), then P goes towards the right, top-right or bottom-right.*

  In particular, c_0 is a 2-neighbour of the position in R' that is closest to c_l. Finally, two 2-paths coming out of R' with distinct starting points cannot share a direction.

Proof. Recall that $\mathrm{Perim}(M) = \mathrm{Perim}(R)$ by Proposition 3.22.

- The first point can be proved by induction on the cardinality (number of coins). Using Proposition 3.26, it suffices to show that all 2-paths in M are direct. A 2-path of cardinality 2 is always direct, as well as any 2-path of cardinality 3 in M as shown in Figure 56.

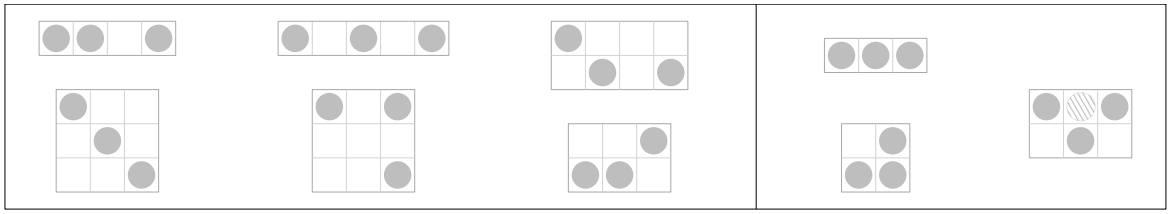

Figure 56. All possible shapes of a 2-path of cardinality 3, up to symmetries. The ones on the left are direct. The ones on the right cannot appear in M: the first two are not minimal, and the final one would contradict Lemma 3.19 since the unoccupied square in the middle of the 'V' shape can be instantly contaminated by the three coins.

Let $P = (c_0, \dots, c_l)$ be a 2-path with cardinality $l + 1 \geq 4$ and suppose that all 2-paths of cardinality at most l are direct. Since $(c_0, \dots, c_{l-1})$ (resp. $(c_1, \dots, c_l)$) is direct, if $(c_1, \dots, c_{l-1})$ takes two directions then the jump from c_0 to c_1 (resp. from c_{l-1} to c_l) cannot take any direction other than these two, and thus P is direct. Therefore the only case left is when $(c_1, \dots, c_{l-1})$ forms a straight line, say from left to right. Suppose for a contradiction that P is indirect, we can assume without loss of generality that the jump from c_0 to c_1 is made towards the bottom or bottom-right and the jump from c_{l-1} to c_l is made towards the top or top-right (it is either this or the other way around). Actually, we can see that one of these two jumps has to be diagonal, otherwise one of c_0 or c_l can be removed while preserving the span. Say the jump from c_0 to c_1 is diagonal for instance. Figure 57 shows how to start the construction of the span by creating a hollow "basin" shape, with non-empty interior since $l \geq 3$, which when filled from left to right leads to a contradiction of Lemma 3.19.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

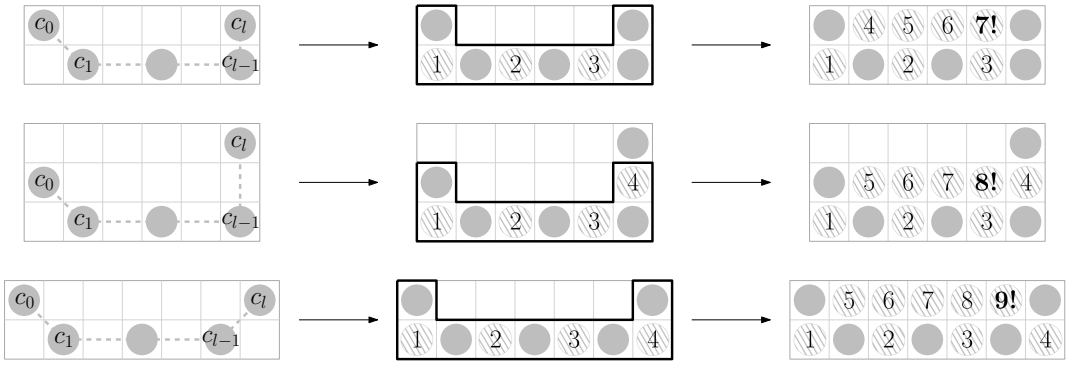

Figure 57. The numbering shows a possible construction of the span. The basin shape is outlined in the middle. The exclamation mark then highlights the square that is contaminated by three neighbours.

Finally, suppose there is a 2-cycle $(c_0, \ldots, c_l, c_0)$ in M, then by what precedes the 2-path $P = (c_0, \ldots, c_l)$ is direct hence $\text{dist}(c_0, c_l) \geq l$. Since $\text{dist}(c_0, c_l) \leq 2$, we get $l = 2$, which is impossible because there cannot be a 2-cycle of cardinality 3 in M according to Figure 56.

- Let R' be a full rectangle and $P = (c_0, \ldots, c_l) \subseteq M \setminus R'$ be a 2-path coming out of R'.
  - Suppose for a contradiction that $c_0 \in N$ and P is not a straight line towards the top. Let (c_i, c_{i+1}) be the first jump that is not strictly towards the top. If $i \geq 1$, since P is direct, this jump is towards the left, right, top-left or top-right, and Figure 58 shows how to start the construction of the span by filling R' and then creating a hollow basin shape with non-empty interior, which contradicts Lemma 3.19 as we have seen before. We now assume that $i = 0$ and discuss depending on the direction of the jump from c_0 to c_1. If it is towards the left at distance 1, right at distance 1, bottom-right or bottom-left, then c_1 can be removed while preserving the span. If it is towards the left at distance 2, right at distance 2, top-right or top-left, then either: $\text{dist}(c_0, R') = 1$, in which case c_0 can be removed while preserving the span, or $\text{dist}(c_0, R') = 2$, in which case the construction from Figure 58 still works.

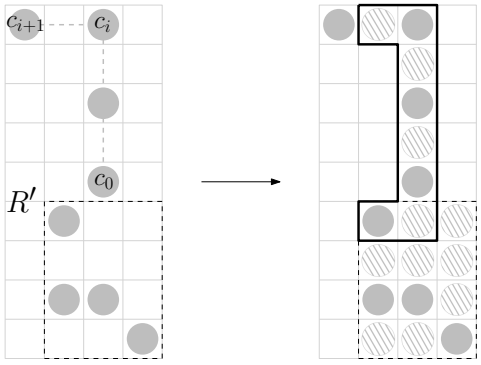

Figure 58. Case $c_0 \in N$, contradiction. The basin shape is outlined.

  - Suppose $c_0 \in NE \setminus \{SE, NW\}$. Consider the minimum configuration M' that is the same as M except for the coins in R' that we replace by an 'L' L from the bottom-left corner of R' to the top-right corner of R'. Note that M' still satisfies the conditions of this proposition, therefore all 2-paths in M' are direct, and in particular $L \cup P$ is

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

direct. Since $L \cup \{c_0\}$ goes top-right, P necessarily goes top, right or top-right.

– Suppose $c_0 \in NE \cap SE$, then $(M \cap R') \cup \{c_0\}$ is a 2-path towards the right. Since $(M \cap R') \cup P$ is direct, P necessarily goes towards the right, top-right or bottom-right.

Finally, suppose for a contradiction that there are two 2-paths P and P' coming out of R' with distinct respective starting points c_0 and c'_0, that share a direction. Say that direction is top, i.e. each of P and P' goes towards the top, top-right or top-left. We can eliminate all cases where one of c_0 or c'_0 can be removed while preserving the span. In all other cases, we can start the construction of the span by filling R' and then creating a hollow basin shape with non-empty interior as in Figure 59, hence a contradiction.

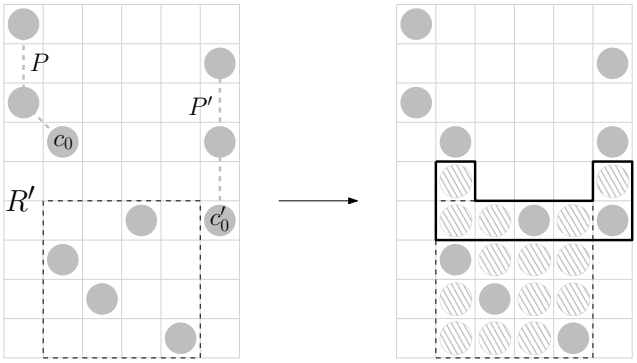

Figure 59. There cannot be two different exits out of a full rectangle towards the same direction (in this case: top). The basin shape is outlined.

□

3.3.3 Unlabelled game

We now go back to playing the game, of which we will only consider the unlabelled version. We also restrict ourselves to puzzles where the starting and ending configurations both have the same span. Let A be a fixed minimum+1 configuration with cardinality N, we want to know what configurations with same span as A are reachable from A. A trivial remark is that, if we only play moves that preserve the span, then the configuration remains minimum+1 all the way.

Proposition 3.29. *Deciding whether A is minimum+1, and finding all possibilities for the "+1" coin (i.e. the first moved coin, since we want to preserve the span) in the positive case, can be done in $O(N^3)$ time.*

Proof. Remember that spans are computed in $O(N^2)$ time according to Proposition 3.21. We first compute $\mathrm{span}(A) = \bigcup_{i=1}^{s} R_i$, where R_i is of size $m_i \times n_i$, and check that $|A| = \sum_{i=1}^{s} \lceil \frac{m_i+n_i}{2} \rceil + 1$. If it is the case, then A is either minimal or minimum+1. For each $a \in A$, we then compute $\mathrm{span}(A \setminus \{a\})$ and compare it with $\mathrm{span}(A)$: this is done in $O(N^3)$ time total. If the removal of a decreases the span for all a, then A is minimal, otherwise A is minimum+1 and any a whose removal does not decrease the span is a possibility for the "+1" coin. □

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Up to a separate study of each component of the span, we will suppose that span(A) is a single $m \times n$ rectangle R. The next proposition shows that our problem is easy if R is even. The result about labelled puzzles that we mentioned in the previous section follows as a side effect.

Proposition 3.30. *Suppose R is even, then the first coin moved from A is the only coin that can be moved throughout without reducing the span. In particular, let B be a configuration with span R, then $A \to B$ if and only if $A \mapsto B$.*

Proof. Suppose that $A \to B$. Since B is minimum+1, there exists $b \in B$ such that $B \setminus \{b\}$ is minimum with even span R. By Proposition 3.22, $B \setminus \{b\}$ contains all connected components of size 1. Proposition 1.3 concludes. □

Proposition 3.31. *Let A be a labelled configuration with exactly two more coins than the minimum. Suppose that* span(A) *has all even components, including one of size 1×5 or 5×1 that we denote by C. If A contains a coin c at the center position p of C, then there can never be a coin other than c at position p unless we reduce the span during the moves. In particular, if B has the same span as A and contains a coin other than c at position p, then the labelled puzzle $A \xrightarrow{?} B$ is not solvable.*

Proof. Consider the instant just before c is first moved: all four other positions in C are necessarily occupied at that moment (otherwise moving c would mean losing span), therefore the configuration A' formed by the coins that are not in C is minimum. We now move c to some position inside span(A) $\setminus C$, then:

- Since A' is minimum and all components are even, Proposition 3.30 ensures that c is the only coin in span(A) $\setminus C$ that can be moved without losing span.
- The four remaining coins inside C obviously cannot be moved without losing span.

In conclusion, no coin other than c can be moved until c comes back to position p. □

From now on, we will suppose that the span R is an odd rectangle. While Proposition 3.30 shows that minimum+1 configurations with even R are almost stalemates, we will see that the slightly higher density of coins with odd R allows for some non-trivial movement, even though the range of possibilities will be nothing close to what we had with two extra coins. This makes the minimum+1 case with odd R somewhat of a "minimum+1.5" situation.

Conceptually, it will help to forget about the "+1" coin, ignoring the last moved coin at all times so as to focus on the underlying minimum configuration. We now explain how this effectively transforms the original (unlabelled) game into a "pushing game". Let c_1 and c_2 be coins sharing an unoccupied neighbour p, then *pushing c_1 onto p* simply means moving c_1 by one spot to position p. This is denoted by $c_1 \mapsto p$. If c_1 and c_2 share exactly one neighbour, or if the destination is not important to us, we might simply say that we *push c_1 towards c_2*. We write $A \xrightarrow{\mathscr{P}} B$ if there is a sequence of pushes from A to B. In the minimum+1 case, the following proposition reduces the original game to the pushing game (see Figure 60 for an illustration). Note that this emulation, just like that of Proposition 1.13, does not behave well with labels.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Proposition 3.32. *Let A and B be minimum+1 configurations with the same span, then $A \xrightarrow{\mathscr{G}} B$ if and only if there exist $a \in A$ and $b \in B$ such that:*

- *a is an extra coin.*
- *b has two neighbours in B.*
- *$A \setminus \{a\} \xrightarrow{\mathscr{P}} B \setminus \{b\}$.*

Moreover, a sequence of T moves is converted from A to B is converted into a sequence of $T-1$ pushes from $A \setminus \{a\}$ to $B \setminus \{b\}$ for some a, b (and reciprocally) in $O(T)$ time.

Proof. In the original game, we can always consider that we never move the same coin twice in a row. To a sequence of moves $(c_t \mapsto p_t)_{1 \leq t \leq T}$ from A to B in $\mathscr{G}$, we associate the sequence of pushes $(c_{t+1} \mapsto p_t)_{1 \leq t \leq T-1}$ from $A \setminus \{c_1\}$ to $B \setminus \{p_T\}$ in $\mathscr{P}$. Proposition 1.12 ensures that those are well and truly pushes. □

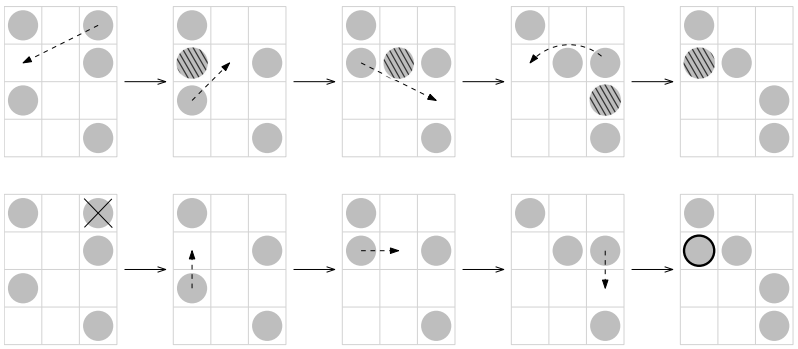

Figure 60. The top sequence shows the actual moves (the last moved coin is shaded) while the bottom sequence shows the equivalent "pushing" version. The crossed out coin at the beginning is a and the circled coin at the end is b.

There are at most N choices for the extra coin a, and at most 2 choices for b (indeed, when adding a coin c to a minimum configuration M, the only coins that can have at least two neighbours are c and one of the two that form the connected component of size 2 in M if there is one). Therefore, up to a time factor $O(N)$, the study of the minimum+1 case reduces to finding all minimum configurations M and M' such that $M \xrightarrow{\mathscr{P}} M'$.

We can go a bit further. Starting from a minimum configuration, the first push necessarily creates an adjacent pair of coins if there was not one already. After that, no loss of perimeter is allowed, so at any time it is only possible to push one of the two adjacent coins. Also note that such a push is always reversible. There are $O(N)$ possibilities for the first push. Up to a time factor $O(N^2)$, our problem thus boils down to finding all minimum configurations M and M', each containing a pair of adjacent coins, such that $M \xrightarrow{\mathscr{P}} M'$ where at all times the pushed coin is part of the adjacent pair. Let M be a fixed minimum starting configuration containing two adjacent coins. We want to know what configurations we can reach from M playing the pushing game, while preserving the span R.

We start by studying the simplest minimum configurations, namely 2-paths. During our study of configurations with two extra coins, we noticed that flipping odd 'L's actually only required one extra coin. The following lemma, which generalises this observation, is one of the most central results in this section.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Lemma 3.33. *If M is a (necessarily shortest) 2-path between two coins c and c', then $M \xrightarrow{\mathscr{P}} M'$ if and only if M' also is a 2-path between c and c', in which case this can be done in $O(mn)$ pushes found in $O(mn)$ time.*

Proof. Suppose M forms a 2-path $(c = c_0, \dots, c_l = c')$. We first show the necessity, and then we give a constructive proof of the sufficiency.

- We want to show that, after pushing one of the two adjacent coins, we still obtain a 2-path between c and c'. Say the adjacent coins are c_i and c_{i+1}, and we push c_i for example. The directness implies that $\mathrm{dist}(c_i, c_j) = 2(i-j)$ if $j < i$ and $\mathrm{dist}(c_i, c_j) = 2(j-i) - 1$ if $j > i$, thus $i \neq 0$ and c_i is necessarily pushed towards c_{i-1} since it is its only neighbour at distance exactly 2. The 2-paths $(c_0, \dots, c_{i-1})$ and $(c_{i+1}, \dots, c_l)$ are untouched. Let $\widetilde{c_i}$ be the new position of the pushed coin: where we had $\mathrm{dist}(c_{i-1}, c_i) = 2$ and $\mathrm{dist}(c_i, c_{i+1}) = 1$, we now have $\mathrm{dist}(c_{i-1}, \widetilde{c_i}) = 1$ and $\mathrm{dist}(\widetilde{c_i}, c_{i+1}) = 2$. In conclusion, the new configuration we obtain is still a 2-path $(c = c_0, \dots, c_{i-1}, \widetilde{c_i}, c_{i+1}, \dots, c_l = c')$.

- Suppose M' also forms a 2-path between c and c', we want to show that $M \xrightarrow{\mathscr{P}} M'$. We can suppose without loss of generality that c' is to the right or bottom-right of c. Let L be the 'L' with end-points c and c' that hugs the left and bottom sides of R and where the two adjacent coins are at the top-left. Since all pushes are reversible, it suffices to show that we can reach L in $O(mn)$ pushes: indeed, we would get $M \xleftrightarrow{\mathscr{P}} L \xleftrightarrow{\mathscr{P}} M'$. We prove this by induction on $|M|$. The case $|M| = 2$ is trivial. Now assume $|M| \geq 3$ and suppose the result holds for 2-paths of cardinality lesser than $|M|$.

  1. If the two adjacent coins are c_0 and c_1, then we push c_1 towards c_2. This way, we can always consider that the 2-path $(c_1, \dots, c_l)$ contains two adjacent coins.
  2. We can now apply the induction hypothesis on the 2-path $(c_1, \dots, c_l)$.
  3. To finish, there are three possibilities depending on the direction of the jump from c_0 to c_1. If it is towards the bottom, then c_0 prolongs the 'L' we obtained by induction and we just have to push c_1 up towards c_0. If it is towards the bottom-right (resp. right), then we proceed as in Figure 61 (resp. Figure 62).

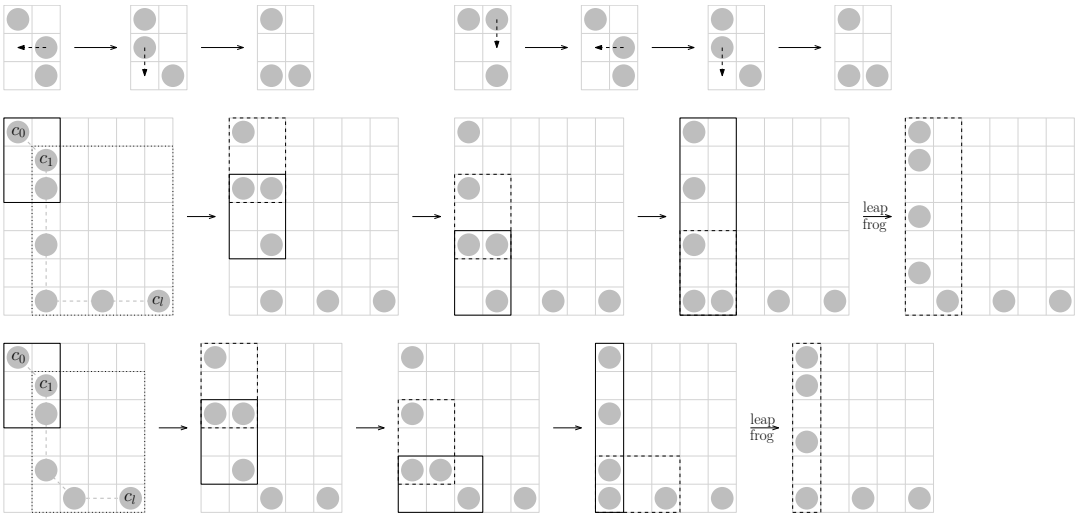

Figure 61. Case where c_1 is to the bottom-right of c_0, depending on which branch is even. We use basic 2×3 subroutines, and one leapfrog at the end.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

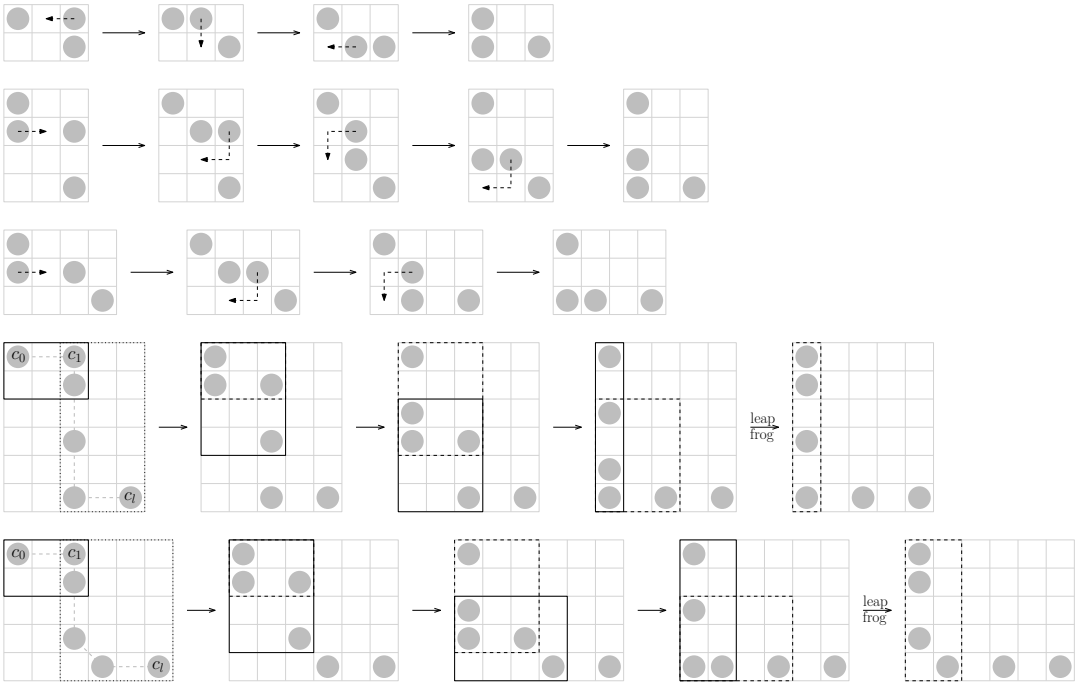

Figure 62. Case where c_1 is to the right of c_0, depending on which branch is even. We use basic 3×2, 3×4 and 4×3 subroutines, and one leapfrog at the end.

The subroutines introduced in Figures 61 and 62 all consist of $O(1)$ pushes and are used $O(mn)$ times in total. As for the leapfrogs used to put the two adjacent coins back at the top, in $O(n)$ pushes each time, they are needed whenever c_1 is to the right or bottom-right of c_0 which happens $O(m)$ times. The remaining punctual pushes do not influence the complexity, and we get a total of $O(mn)$ pushes during the transformation from M to L, hence the same from M to M'. □

We have just seen that, if M is a 2-path, then its end-points can never move and all we can do is exactly change the 2-path that is connecting them. We now try to generalise this as much as possible. When playing the pushing game on a few examples, it quickly becomes apparent that certain coins are immobile and all we do is perform successive transformations of 2-paths between some of them. This is illustrated by Figure 63: in this example, all the action seems to happen inside three specific rectangles, with no possibility other than transition from rectangle to rectangle with consecutive transformations of 2-paths.

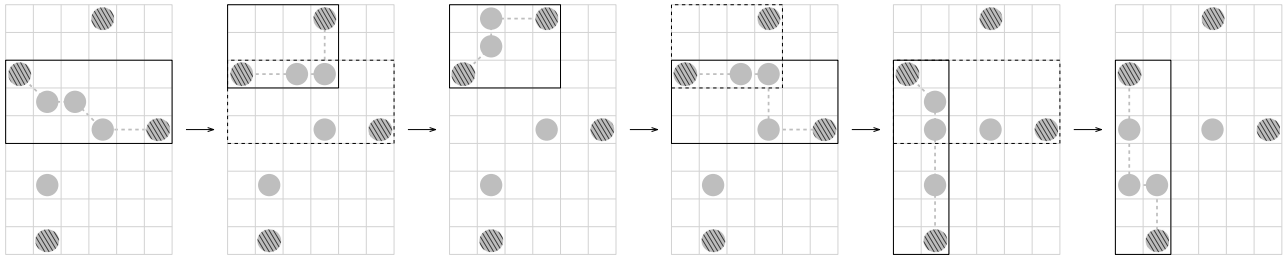

Figure 63. Playing the pushing game. The outlined rectangle shows the 2-path that is currently being transformed. The shaded coins seem unmovable.

48 This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

We start by transforming a certain 2-path inside a certain rectangle R_0, until one of the two adjacent coins becomes neighbours with the first coin of a 2-path P coming out of R_0. We can then push towards this coin and get a new 2-path, inside a new rectangle, with P prolonging one half of the previous 2-path. We then transform this new 2-path, etc.

We now introduce some vocabulary and notations in order to prove these observations rigorously.

- The immobile end coins of the different 2-paths we get during moves will be called *fixed leaves*. Here is how we construct the set $F(M)$ of fixed leaves in M (see Figure 64):

  1 Let P_0 be a maximal 2-path in M containing the two adjacent coins. Let l_0 and l_0' be its end coins, we initialise $F(M) = \{l_0, l_0'\}$ and $R' = R(\{l_0, l_0'\}) = R(P_0)$.

  2 While there exists a 2-path coming out of R': let $P = (c_0, \dots, c_f)$ be a maximal 2-path coming out of R', we add c_f to $F(M)$ and replace R' with $R(R' \cup \{c_f\})$.

  Since there are arbitrary choices made during this algorithm, we should check that this definition does not depend on them. Suppose that, at one point during step 2, P and P' are distinct maximal 2-paths coming out of R' ending at l and l' respectively, and that P is the one that is chosen. By minimality, $l' \notin R(R' \cup \{l\}) = \mathrm{span}(R' \cup \{l\})$ hence $P' \setminus R(R' \cup \{l\})$ is a 2-path coming out of $R(R' \cup \{l\})$ and ending at l'. This means l' will again be a possible choice at the next iteration, so eventually it will get chosen and added in F. A similar reasoning is also valid for the choice of P_0 in step 1.

- We would now like to define the rectangles that we intuited, inside of which the transformations of 2-paths happen. These are all odd rectangles since these 2-paths contain the two adjacent coins. Moreover, each rectangle should contain exactly two fixed leaves, at opposite corners. We therefore define the family $\mathscr{R}(M) = (R_i)_{0 \le i \le s}$ of all rectangles of the form $R(\{l, l'\})$, where $l, l' \in F(M)$, that have the property of being odd and containing no fixed leaf other than l and l'. Note that $\mathscr{R}(M)$ does not depend on $M \setminus F(M)$. We set $R_i = R(\{l_i, l_i'\})$, with $R_0 = R(\{l_0, l_0'\})$ being the rectangle from step 1.

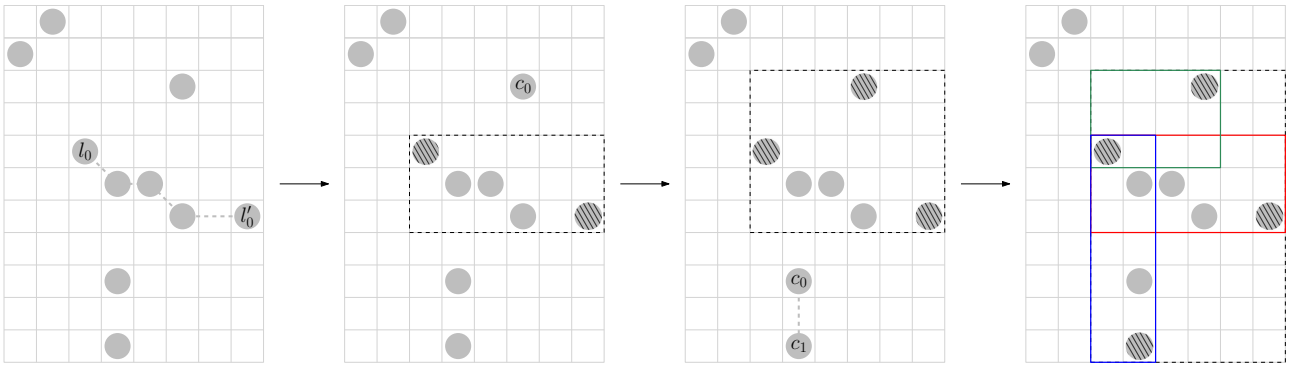

Figure 64. The search algorithm to find $F(M)$. The dashed rectangle is R' and fixed leaves found throughout are shaded. In total, we get 4 fixed leaves, and 3 rectangles represented on the right (the two coins in the top-left corner form $M \setminus \bigcup_{i=0}^{s} R_i$, they are too far from R' hence why the search terminates here).

The following proposition describes the pushing game, thus formalising the previously made remarks. The fixed leaves are indeed immobile throughout, and any sequence of pushes constitutes a series of transformations of 2-paths between pairs of fixed leaves inside the rectangles $\mathscr{R}(M)$.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Definition 3.34. We say a rectangle $R_i \in \mathscr{R}(M)$ is *active* in M if there is a 2-path of coins between l_i and l'_i in M.

Proposition 3.35. *Let $M = M_0 \overset{c_1 \mapsto p_1}{\longmapsto} M_1 \overset{c_2 \mapsto p_2}{\longmapsto} M_2 \mapsto \dots$ be a sequence of pushes from M that preserves the span. Then, at any time $t \geq 0$:*

(i) $F(M_t) = F(M) =: F$. In particular, each $l \in F$ is a leaf in M_t i.e. is of degree at most 1 in the forest M_t in terms of the 2-connectivity.

(ii) $M_t \setminus \bigcup_{i=0}^{s} R_i = M \setminus \bigcup_{i=0}^{s} R_i$. In other words, nothing ever changes outside of $\bigcup_{i=0}^{s} R_i$.

Moreover, there exists a sequence $(I_t)_{t\geq 0} \in [\![0, s]\!]^{\mathbb{N}}$ such that at any time $t \geq 0$:

(iii) R_{I_t} is active in M_t (we say R_{I_t} is "active at time t").

(iv) For all $c \in M_t \cap (\bigcup_{i=0}^{s} R_i \setminus R_{I_t})$, there exists a maximal 2-path $P = (c, \dots, l)$ coming out of some full rectangle containing R_{I_t} that satisfies $P \cap F = \{l\}$.

Finally, let $t_0 = 0$ and $t_{j+1} = \inf\{t > t_j \mid I_t \neq I_{t_j}\}$, then for all $j \geq 0$ the subsequence of pushes from M_{t_j} to $M_{t_{j+1}-1}$ is a transformation of a 2-path inside $R_{I_{t_j}}$ as in Lemma 3.33.

Proof. We may say R_{I_t} is the *current* rectangle at time t. We prove this proposition by induction on t, constructing the sequence $(I_t)_t$ as the pushes go. For $t = 0$, setting $I_0 = 0$, we see that (i), (ii), (iii) and (iv) are all true by construction of F and $(R_i)_i$. Now suppose that (i), (ii), (iii) and (iv) hold up to time t with $I_0, \dots, I_t$ already constructed. There are two cases depending on the next push, namely $c_{t+1} \mapsto p_{t+1}$.

(a) First case: $p_{t+1} \in R_{I_t}$.

This means that we stay inside of R_{I_t}, so it is natural to show that R_{I_t} is still active at time $t+1$. Since R_{I_t} is odd, we know by Proposition 3.24 that p_{t+1} is not a neighbour of a coin outside R_{I_t}, therefore c_{t+1} is pushed towards a coin that was itself inside R_{I_t}, and it is as if the coins outside R_{I_t} did not exist (see Figure 65).

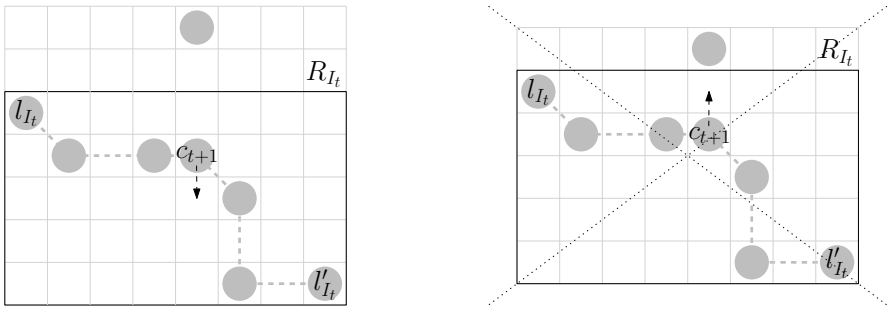

Figure 65. Pushing inside of R_{I_t}. On the left is a possible push, that simply transforms the 2-path inside R_{I_t}. On the right is an impossible situation, where the push would break the 2-path: this would necessitate a coin at distance 1 from R_{I_t}, contradicting Proposition 3.24.

We can therefore apply the "only if" direction of Lemma 3.33 inside R_{I_t} and get that R_{I_t} is still active after the push $c_{t+1} \mapsto p_{t+1}$, which allows us to set $I_{t+1} = I_t$ so that (iii) holds at time $t+1$. Moreover, (i) and (iv) at time $t+1$ follow immediately from (i) and (iv) at time t. Finally, to prove (ii) at time $t+1$, we show that $F(M_{t+1}) = F(M_t)$: in both M_t and M_{t+1}, the search algorithm may choose the 2-path inside R_{I_t} as P_0 in step $\boxed{1}$, and the rest of the algorithm will be the same in both cases since $M_t \setminus R_{I_t} = M_{t+1} \setminus R_{I_t}$.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

(b) Second case: $p_{t+1} \notin R_{I_t}$.

In this case, we push outside R_{I_t}, so we have to find a new current rectangle. We can first notice that $c_{t+1} \notin \{l_{I_t}, l'_{I_t}\}$ since the degree of l_{I_t} and l'_{I_t} is only 1. Let $c \notin R_{I_t}$ be the coin that c_{t+1} is pushed towards (see Figure 66). By construction, all coins in $M \setminus \bigcup_{i=0}^{s} R_i$ are at distance at least 3 from R_{I_t}, therefore $c \in \bigcup_{i=0}^{s} R_i$. Since (iv) holds at time t, there exists a maximal 2-path $P = (c, \dots, l)$ coming out of R_{I_t} that satisfies $P \cap F = \{l\}$. Before the push, c_{t+1} is connected to both l_{I_t} and l'_{I_t} by a 2-path on each side. After the push, as shown in Figure 66, the pushed coin is still connected by a 2-path to either l_{I_t} or l'_{I_t}, depending on which half of the 2-path joining l_{I_t} and l'_{I_t} contained the two adjacent coins. Without loss of generality, assume the pushed coin is connected to l_{I_t} by a 2-path: then P prolongs this 2-path, creating a long 2-path connecting l_{I_t} to l and containing the two adjacent coins (in particular, $\mathrm{dist}(l_{I_t}, l)$ is odd). The rectangle $R(\{l_{I_t}, l\})$ is odd and does not contain any fixed leaf other than l_{I_t} and l, which allows us to define I_{t+1} by $R_{I_{t+1}} = R(\{l_{I_t}, l\})$, and $R_{I_{t+1}}$ is active at time $t+1$. Actually, $R_{I_{t+1}}$ was already active at time t, and the push $c_{t+1} \mapsto p_{t+1}$ was a transformation of the 2-path inside it.

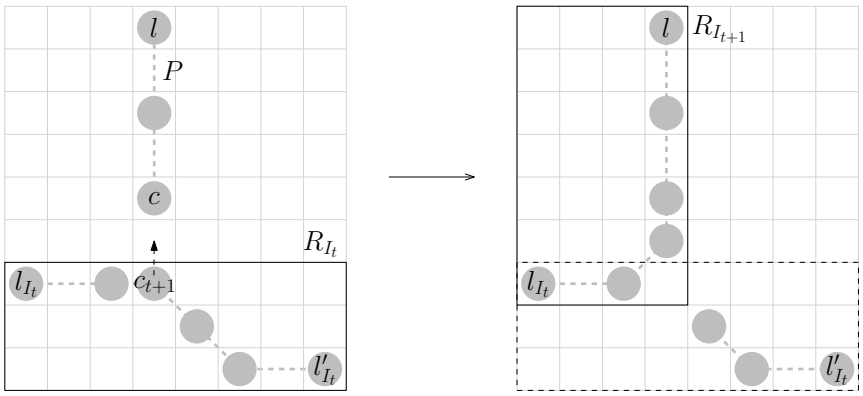

Figure 66. Pushing outside of R_{I_t}.

At this point, (i) and (iii) are proven at time $t+1$. To prove (ii) at time $t+1$, we show that $F(M_{t+1}) = F(M_t)$: in both M_t and M_{t+1}, the search algorithm may choose the 2-path inside $R_{I_{t+1}}$ as P_0 in step $\boxed{1}$ (we use the fact that $R_{I_{t+1}}$ is already active at time t), and the rest of the algorithm will be the same in both cases since $M_t \setminus R_{I_{t+1}} = M_{t+1} \setminus R_{I_{t+1}}$. Finally, we have to show that (iv) holds at time $t+1$. Let $\widetilde{c} \in M_{t+1} \cap \left(\bigcup_{i=0}^{s} R_i \setminus R_{I_{t+1}}\right)$, we distinguish two cases.

- Suppose $\widetilde{c} \in R_{I_t}$. We know exactly what the coins in $R_{I_t} \setminus R_{I_{t+1}}$ look like: they form a 2-path out of $R_{I_{t+1}}$ that ends at l'_{I_t}, with l'_{I_t} being the only fixed leaf (see Figure 66, right). Therefore, we prove (iv) at time $t+1$ by simply following this 2-path, and the maximality is ensured by the fact l'_{I_t} was of degree only 1 before the push.
- Suppose $\widetilde{c} \notin R_{I_t}$ (see Figure 67). Then $\widetilde{c} \in M_t \cap (\bigcup_{i=0}^{s} R_i \setminus R_{I_t})$, so the induction hypothesis gives us a 2-path $\widetilde{P} = (\widetilde{c}, \dots, \widetilde{l})$, coming out of some full rectangle $R^{(t)}$ containing R_{I_t}, such that $\widetilde{P} \cap F = \{\widetilde{l}\}$. The same 2-path $\widetilde{P}$ will work at time $t+1$, however $R^{(t)}$ might not contain $R_{I_{t+1}}$, which is why we define $R^{(t+1)} := R(R^{(t)} \cup R_{I_{t+1}})$. Since $R^{(t)}$ and $R_{I_{t+1}}$ intersect and are both full at time t, we know that $R^{(t+1)}$ is full at time t, therefore it is still full at time $t+1$ since the push happens inside it. In particular, since $\widetilde{c} \notin R^{(t)}$ and $\widetilde{c} \notin R_{I_{t+1}}$, this yields $\widetilde{c} \notin R^{(t+1)}$. On the other hand,

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

the fact that $R^{(t+1)} \supseteq R^{(t)}$ implies that $\text{dist}(R^{(t+1)}, \widetilde{c}) \leq 2$, so all that is left to check is that $\widetilde{P}$ is entirely disjoint from $R^{(t+1)}$. By Proposition 3.28, we know that $\widetilde{P}$ goes away from $R^{(t)}$, in particular it goes away from $l_{I_t} \in R^{(t)}$. By "going away from l_{I_t}", we mean that, when following $\widetilde{P}$ coin by coin from $\widetilde{c}$ to $\widetilde{l}$, the distance to l_{I_t} only increases. Since $l_{I_t} \in R^{(t+1)}$ and $\widetilde{c} \notin R^{(t+1)}$, this implies $\widetilde{P}$ can never enter $R^{(t+1)}$.

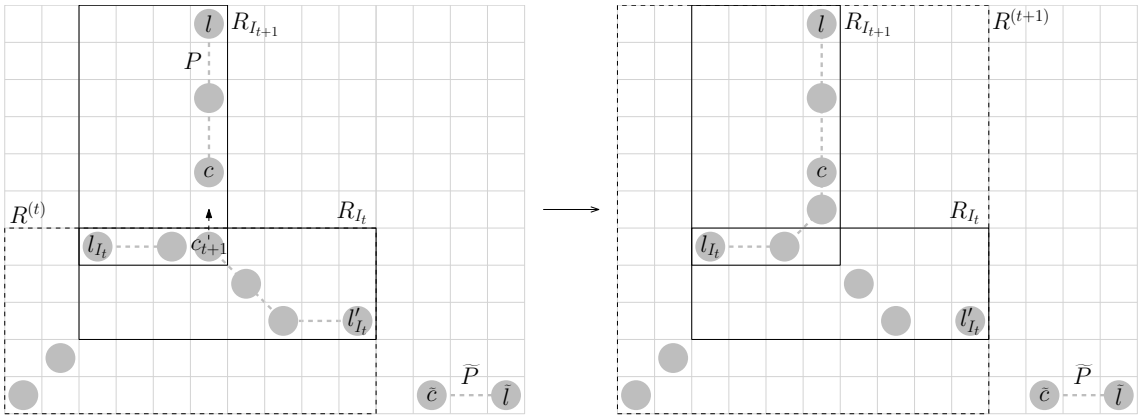

Figure 67. Time t is on the left, time $t+1$ (after the push) is on the right.

The induction is now complete. As for the final statement of the proposition, it has been shown during the proof. □

The previous proposition gives some necessary conditions of solvability for pushing puzzles, the most important result being that the fixed leaves are an invariant. Meanwhile, the next theorem states a sufficient condition.

Definition 3.36. We say a rectangle $R_i \in \mathscr{R}(M)$ is *central* in M if all the fixed leaves outside R_i are to the sides of R_i, i.e. they are in the region defined in Figure 68.

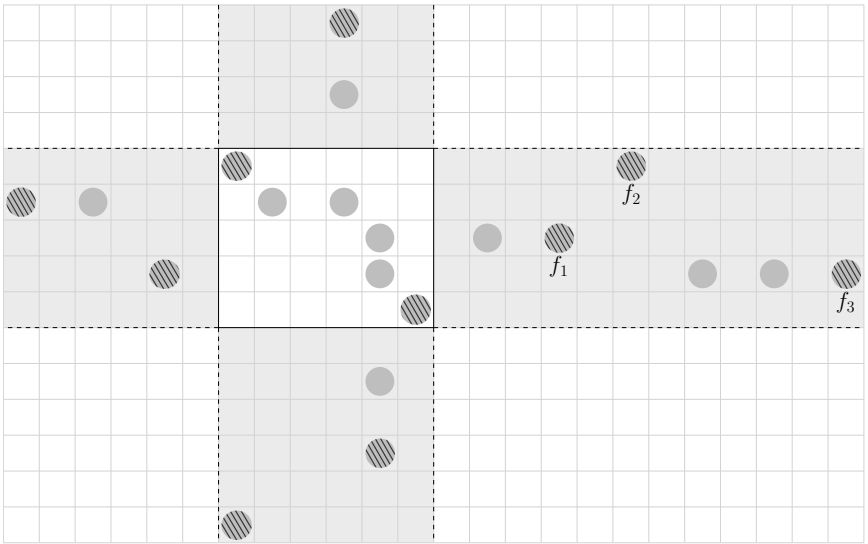

Figure 68. An example of a central rectangle: all fixed leaves (shaded) outside it are to its sides, in the area filled with grey. The fixed leaves marked f_1, f_2 and f_3 refer to the proof of Theorem 3.37.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Theorem 3.37. *Let M' be a minimum configuration with same span as M containing two adjacent coins. Suppose that $F(M') = F(M) =: F$ and that there exists $R_i \in \mathscr{R}$ such that:*

- *R_i is central in M (and in M' as a result).*
- *$M \xrightarrow{\mathscr{P}} M_1$ for some M_1 in which R_i is active (in particular: $F(M_1) = F$).*
- *$M' \xrightarrow{\mathscr{P}} M'_1$ for some M'_1 in which R_i is active (in particular: $F(M'_1) = F$).*

Then $M_1 \xrightarrow{\mathscr{P}} M'_1$ in $O(N^2)$ pushes, hence $M \xrightarrow{\mathscr{P}} M'$.

Proof. We prove that $M_1 \setminus R_i = M'_1 \setminus R_i$, by rebuilding M_1 and M'_1 based on the order in which the search algorithm finds their fixed leaves. Consider M_1 for instance. Since R_i is active in M_1, we can assume the algorithm starts there. Let f_1 be the fixed leaf found at the first iteration of step $\boxed{2}$, and suppose f_1 is to the right of R_i for instance. Then, out of all the fixed leaves that are to the right of R_i, f_1 is necessarily the leftmost one: indeed, $R(R_i \cup \{f_1\})$ would contain a fourth fixed leaf other than l_i, l'_i and f_1 otherwise. Besides, Proposition 3.28 ensures that the 2-path coming out of R_i and ending at f_1 in M_1 forms a straight line, because f_1 is to the side of R_i. We can then repeat this argument on the bigger rectangle $R(R_i \cup \{f_1\})$. Let $f_1, \dots, f_r$ be the fixed leaves that are to the right of R_i, sorted by ascending x coordinate, we obtain that f_1 is connected to R_i by a straight line, f_2 is connected to $R(R_i \cup \{f_1\})$ by a straight line, f_3 is connected to $R(R_i \cup \{f_1, f_2\})$ by a straight line, etc. as in Figure 68. The same argument is valid for the other three directions. In conclusion, all coins in M_1 outside $R_i \cup F$ are completely forced, and they are necessarily the same as in M'_1. Therefore, the only difference between M_1 and M'_1 is the 2-path inside R_i, so we can go from M_1 to M'_1 in $O(N^2)$ moves according to Lemma 3.33. Since all pushes are reversible, we finally obtain $M \xleftrightarrow{\mathscr{P}} M_1 \xleftrightarrow{\mathscr{P}} M'_1 \xleftrightarrow{\mathscr{P}} M'$. □

Theorem 3.38. *Suppose M is 2-connected, and let M' be a minimum configuration with same span as M containing two adjacent coins. Then $M \xrightarrow{\mathscr{P}} M'$ if and only if $F(M) = F(M')$, in which case we find a sequence of $O(N^2)$ pushes from M to M' in $O(N^2)$ time.*

Proof. The "only if" direction is already known. Suppose $F(M) = F(M')$. An important remark is that, by the 2-connectedness of M, the rectangles $(R_i)_i$ are all active in M. According to Theorem 3.37, it suffices to show that there is a central rectangle that can be made active via pushes from M'. Let $R_{i_0} = R(\{l_{i_0}, l'_{i_0}\})$ be a rectangle that is active in M', we can assume without loss of generality that the fixed leaves l_{i_0} and l'_{i_0} are the top-left and bottom-right corners of R_{i_0}. If R_{i_0} is central, then we have finished. Otherwise, there is a fixed leaf l that is strictly to the top-left, top-right, bottom-left or bottom-right of R_{i_0}. Actually, it is impossible that l is to the top-left or bottom-right of R_{i_0}: since there exists a 2-path between l and some coin $c \in R_{i_0} \setminus \{l_{i_0}, l'_{i_0}\}$, there would be an indirect 2-path in M, as shown in Figure 69.

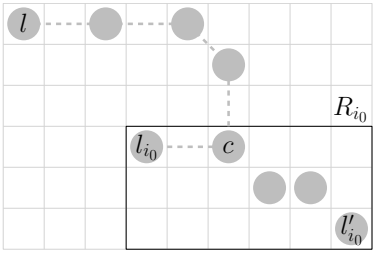

Figure 69. An impossible situation: the drawn path is indirect.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Assume l is to the top-right of R_{i_0} for instance. By Proposition 3.28 and the 2-connectedness of M, there is a 2-path P ending at l that starts at the top-right corner of R_{i_0}, and $M \cap R_{i_0}$ is necessarily an 'L' hugging the top and right sides of R_{i_0}. From there, M is almost exactly known. Other than the coins in $R(R_{i_0} \cup P)$, there can only be: a 2-path out of P to the right in a straight line towards some fixed leaf l', or a 2-path out of P to the top in a straight line towards some fixed leaf l' (and it would be impossible to have both, moreover these two situations are symmetrical, so we are only going to consider the former). In conclusion, M has one of two shapes, as described in Figure 70. We have also represented all the rectangles $(R_i)_i$ for both shapes, depending on the parity of the sides of R_{i_0}.

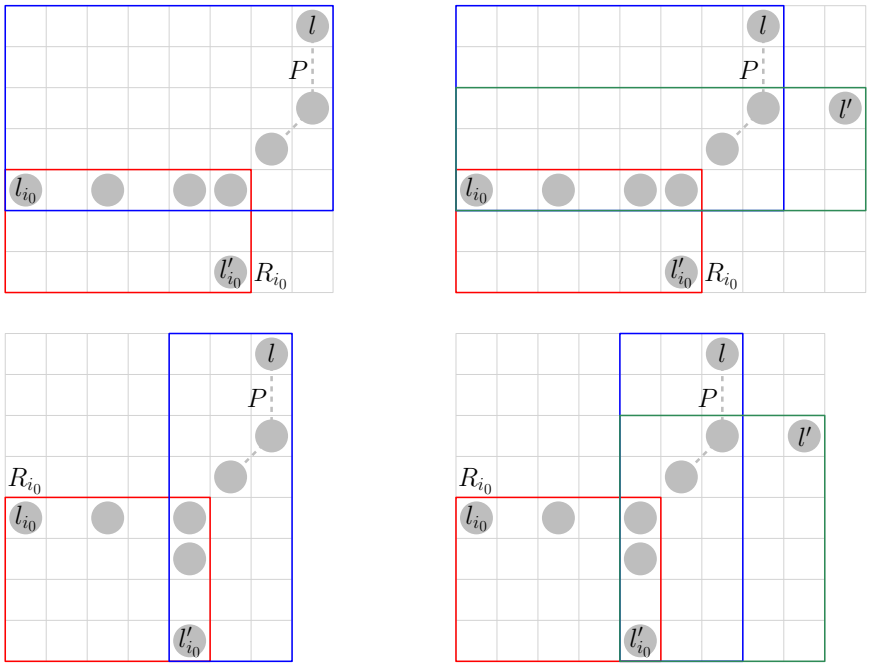

Figure 70. Left, right: two possible shapes of M. Top: case where R_{i_0} has even horizontal sides. Bottom: case where R_{i_0} has even vertical sides.

In all cases, we notice that R_{i_0} is the only rectangle in $\mathscr{R}(M) = \mathscr{R}(M')$ that is not central. We now go back to M'. Since $F(M') = F(M)$, there is a 2-path in M' coming out of the top-right corner of R_{i_0}, ending at l or l'. Therefore, from M', we simply have to transform the 2-path inside R_{i_0} into an 'L' that hugs the top and right sides using Lemma 3.33, thus making another (necessarily central) rectangle active.

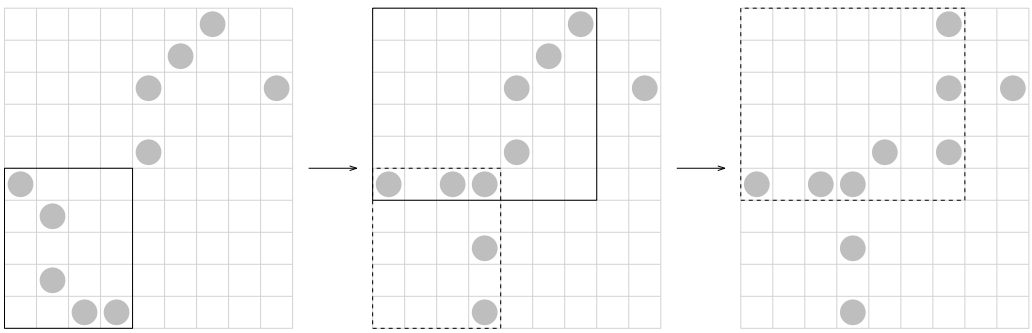

Figure 71. Going from M' (left) to M (right) in two steps: we first activate a central rectangle if needed, and then we correct the 2-path inside this rectangle.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Let us sum up the proof:

- If there is an active central rectangle in M', then we just transform the 2-path inside it to get M.
- Otherwise, let R_{i_0} be the only active rectangle in M' (with fixed leaves in the top-left and bottom-right corners for instance), then there is a 2-path P coming out of the top-right or bottom-left corner of R_{i_0}. We transform the 2-path inside R_{i_0} into an 'L' that connects to P, thus activating a central rectangle, and finally we transform the 2-path inside this new rectangle to get M (see Figure 71). □

In $\mathscr{P}$ restricted to minimum configurations with two adjacent coins and equal span, the fixed leaves thus characterise what is obtainable from a 2-connected configuration M (or, equivalently, what can reach M). However, this is not true in general, as put to light by the pushing puzzle from Figure 72. Consider a sequence of pushes starting from the left configuration. It is easy to show by induction on t that, at any time t, if R_1 (resp. R_2) is active then there is no 2-path coming out of R_1 (resp. R_2) towards l' (resp. l). In particular, R_3 can never be made active and the puzzle from Figure 72 is unsolvable.

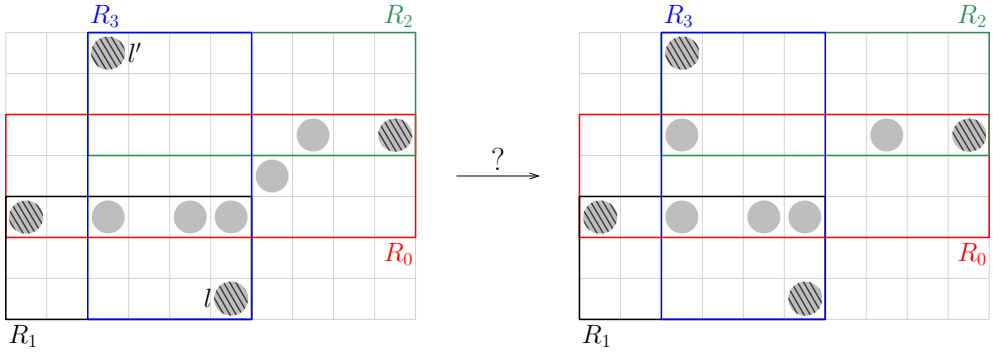

Figure 72. An unsolvable pushing puzzle where both configurations have the same fixed leaves.

During a sequence of pushes starting from M, there is always a rectangle in $\mathscr{R}(M)$ that is active, according to Proposition 3.35. However, we now see that the family $\mathscr{R}(M)$ is too big in general, in the sense that it may contain rectangles that can never be made active (e.g. R_3 in the left configuration from Figure 72). Instead of defining $\mathscr{R}(M)$ based solely on $F(M)$, we could imagine ways to account for which transitions between rectangles are actually feasible. Actually, even our set of fixed leaves is visibly too big in general. The following algorithm defines smaller alternatives for $F(M)$ and $\mathscr{R}(M)$, that we denote by $\widetilde{F}(M)$ and $\widetilde{\mathscr{R}}(M)$ respectively.

1. Let P_0 be a maximal 2-path in M containing the two adjacent coins. Let l_0 and l_0' be its end coins, we initialise $\widetilde{F}(M) = \{l_0, l_0'\}$, $\widetilde{\mathscr{R}}(M) = \{R(P_0)\}$ and $U = R(P_0)$.
2. While there exists a 2-path coming out of $R(U)$ whose first coin c_0 is a 2-neighbour of U:
   - Let $P = (c_0, \dots, c_f)$ be a maximal 2-path starting at c_0 in $M \setminus U$.
   - We add c_f to $\widetilde{F}(M)$.
   - For each $R \in \widetilde{\mathscr{R}}(M)$ that is a 2-neighbour of c_0: we have $R = R(\{l, l'\})$ for some $l, l' \in \widetilde{F}(M)$, say l is at odd distance from c_0 (i.e. l' is at even distance from c_0), we add $R(\{l, c_f\})$ to $\widetilde{\mathscr{R}}(M)$.
   - We update $U := \bigcup_{R \in \widetilde{\mathscr{R}}(M)} R$.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

In this new version, the rectangles are found at the same time as the fixed leaves. The first thing to do should be to check that $\widetilde{F}(M)$ and $\widetilde{\mathscr{R}}(M)$ are well defined i.e. do not depend on the choices made during the search, which we do not address here. We suspect that Proposition 3.35 still holds for $\widetilde{F}(M)$ and $\widetilde{\mathscr{R}}(M)$, based on the following observation. Say $l \in F(M)$ is *active* if a rectangle in $\mathscr{R}(M)$ containing l is active: during a sequence of pushes from M, the possible active rectangles containing l the first time l is made active seem to be exactly the possible active rectangles containing l at any point. If this is true, then the above algorithm does not miss any potential (future) active rectangle.

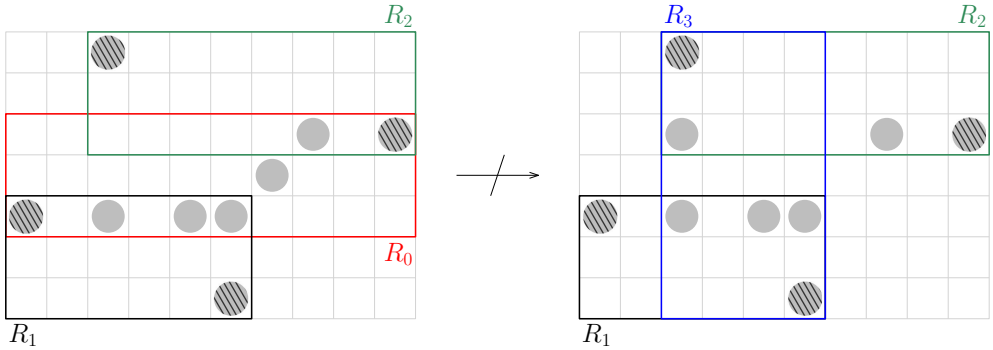

Figure 73. $\widetilde{F}$ and $\widetilde{\mathscr{R}}$ for the configurations of Figure 72. The fixed leaves are the same as in the first version, but the rectangles now highlight the dissimilarity.

The fundamental difference between the two configurations of Figure 72 appears when we consider $\widetilde{\mathscr{R}}$: the rectangles that we get are the ones that can actually become active, and we instantly see that the puzzle is unsolvable (see Figure 73). More generally, we end this section on minimum+1 configurations with the following conjecture.

Conjecture 3.39. *Let M and M' be minimum configurations with same span, each containing two adjacent coins. Then $M \xrightarrow{\mathscr{P}} M'$ if and only if the three following conditions are satisfied:*

- $\widetilde{F}(M) = \widetilde{F}(M')$.
- $\widetilde{\mathscr{R}}(M) = \widetilde{\mathscr{R}}(M')$.
- $M \setminus \left(\bigcup_{R \in \widetilde{\mathscr{R}}(M)} R\right) = M' \setminus \left(\bigcup_{R \in \widetilde{\mathscr{R}}(M')} R\right)$.

3.4 One extra coin: minimal+1 configurations

Minimal configurations are much more difficult to study compared to minimum configurations, due to the fact we have little information about their structure or even their cardinality. If a starting minimal+1 configuration A is not minimum+1, then the main hope is to make a second extra coin appear so that Theorem 3.8 applies from there. Until that happens, and unless we allow the span to decrease, the configuration remains minimal+1. By Proposition 1.12, all moves before a second extra coin appears are thus equivalent to: choosing the "+1" coin i.e. the first coin we want to move, removing it from the board, and playing the pushing game $\mathscr{P}$ on the resulting minimal (not minimum) configuration. Figure 74 shows two examples

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

of non-minimum minimal configurations, that have same span and cardinality but prove very different: for one of them, any push decreases the span, while for the other it is possible to make a second extra coin appear after a few pushes.

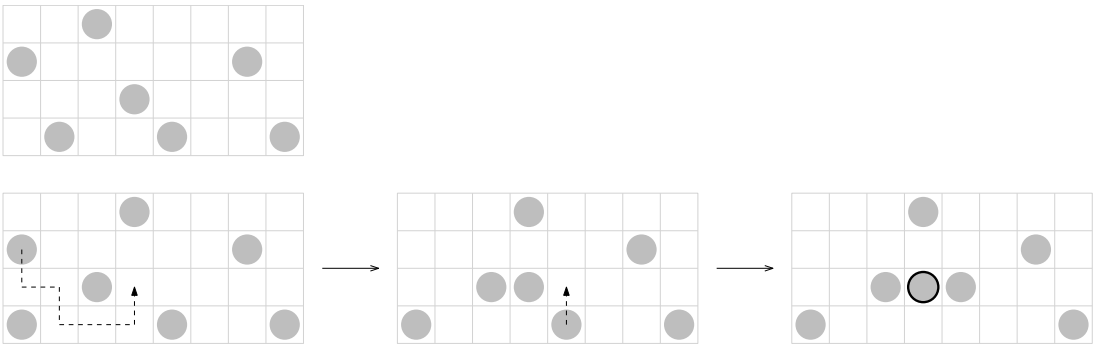

Figure 74. Two 8×4 minimal configurations of cardinality 7 (thus not minimum) on the left. The top one is a deadlock. The bottom one becomes non-minimal after 7 pushes, with the second extra coin being circled.

Therefore, one of the main interrogations about non-minimum minimal configurations is: when is it possible to make a second extra coin appear via a sequence of pushes?

We claim this is always the case for 2-paths, although the following elements of proof would need consolidation. Let $P = (c_0, \ldots, c_l)$ be a non-minimum minimal 2-path with span R.

- We can suppose that the end-points of P are at opposite corners of R, up to at most two pushes towards the sides of R as in Figure 75.

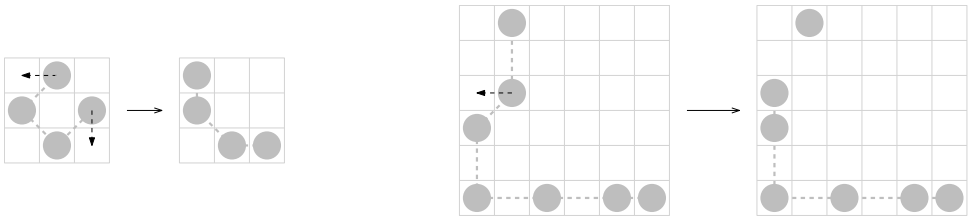

Figure 75. Pushing against the sides of the hull. Note that the new 2-path we consider might be smaller (as is the case on the right).

- Suppose P is a direct 2-path between coins at opposite corners of R. Since P is not minimum, there are necessarily at least two pairs of adjacent coins in P. We can therefore "leapfrog" as in Figure 76 and get the desired extra coin.

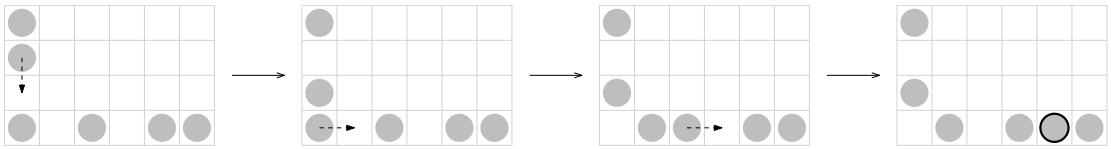

Figure 76. Retrieving an extra coin from a 2-path containing two pairs of adjacent coins.

- Suppose P is an indirect 2-path between coins at opposite corners of R, for instance c_l is to the top-right of c_0 and there is a detour towards the bottom at some point. By minimality,

there cannot be a jump towards the bottom or bottom-left in P, so the first detour starting from c_0 is necessarily a bottom-right jump (c_i, c_{i+1}) for some i. Let (c_j, c_{j+1}) be the first jump after that (i.e. $j > i$) that goes towards the top, top-left or top-right: necessarily the jump (c_{j-1}, c_j) is towards the right or bottom-right. In all cases, pushing c_i towards the right and c_j towards the left does not decrease the span, as portrayed in Figure 77 (if c_j is to the direct right of c_{j-1}, then we do not even have to push it). We now have a 2-path containing at least two pairs of adjacent coins, so we can proceed as in Figure 76 to collect an extra coin.

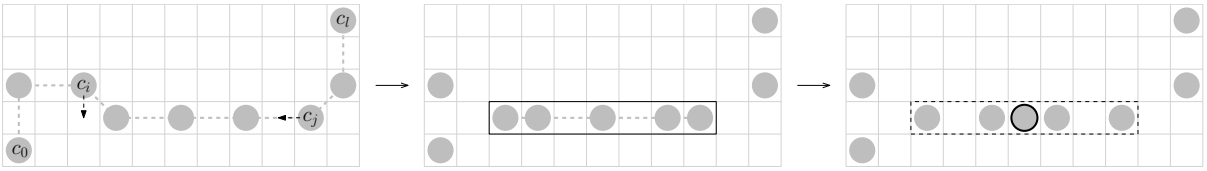

Figure 77. Retrieving an extra coin from a non-minimum minimal 2-path with end-points at opposite corners of its hull.

We do not know much about other minimal configurations. Even the case of 2-connected minimal configurations cannot be deduced from the case of minimal 2-paths, since there exist 2-connected minimal configurations that are not minimum but whose 2-paths are all minimum (see Figure 78 for an example).

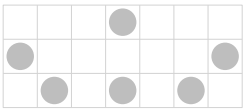

Figure 78. A 2-connected minimal configuration that is not minimum but where all 2-paths are minimum. In this example, an extra coin can be retrieved after 3 pushes.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Conclusion

We have made some contributions to the study of the coin-moving game introduced in [1], by correcting some mistakes that were made in the article as well as obtaining new results on cases that the authors had left to examine. During the process, some new questions have also arisen.

On the triangular grid, we separated the unlabelled and labelled versions of the game, using the alternative game $\mathscr{G}'$ to provide refined proofs of the results from the article. We also rectified an inaccuracy in the main theorem.

On the square grid, we gave a slightly simpler proof of the main theorem about puzzles with two extra coins. We saw that, contrary to what was claimed in the article, solvable unlabelled puzzles with two extra coins are not characterised by the inclusion of the spans. As for labelled puzzles, we improved the principles that were sketched in the article into a result comparable to that of the unlabelled version. Our main contribution is about minimum+1 configurations, based on our detailed study of the structure of minimum configurations. We have addressed 2-connected minimum configurations, and we are hopeful that our methods can lead to a characterisation of solvable unlabelled puzzles $A \xrightarrow{?} B$ when A and B are minimum+1 configurations with the same span. However, minimal+1 configurations remain obscure.

Here are a few open questions:

- About the square grid:
  - What is the correct characterisation of solvable unlabelled puzzles with two extra coins? When is it possible to break up the span into smaller components (see Figure 33)?
  - What if the starting configuration is minimum+1 but we are allowed to reduce the span?
  - What can be said when the starting configuration is minimal+1? Is there a structure to minimal configurations?
- About the game in general:
  - What is the complexity of finding a solution using the fewest moves?
  - What results can we prove for general graphs?

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

References

[1] Erik D. Demaine, Martin L. Demaine and Helena A. Verill, "Coin-Moving Puzzles", *More Games of No Chance*, pages 405–431. Cambridge University Press, 2002. Collection of papers from the MSRI Combinatorial Game Theory Research Workshop, Berkeley, California, July 24–28, 2000.

[2] Harry Langman, "Curiosa 261: A disc puzzle", *Scripta Mathematica*, 17(1–2):144, March–June 1951.

[3] Harry Langman, "Curiosa 342: Easy but not obvious", *Scripta Mathematica*, 19(4):242, December 1953.

[4] Martin Gardner, *Mathematical Carnival*, chapter 2 "Penny Puzzles", pages 12-26. Alfred A. Knopf, New York, 1975.

[5] Elwyn R. Berlekamp, John H. Conway and Richard K. Guy, *Winning Ways*, volume 2, pages 755-756. Academic Press, London, 1982.

[6] Adrian Dumitrescu and János Pach, "Pushing squares around", *Graphs and Combinatorics*, volume 22, pages 37-50, 2006.

[7] Marthe Bonamy et all., "Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs", *Journal of Combinatorial Optimization*, pages 1-12, 2012.

[8] Marthe Bonamy and Nicolas Bousquet, "Reconfiguring independent sets in cographs", *CoRR*, 1406.1433, 2014.

[9] Arash Haddadan et all., "The complexity of dominating set reconfiguration", *Lecture Notes in Computer Science*, volume 9214, pages 398-409, 2015.

[10] Marthe Bonamy et all., "The perfect matching reconfiguration problem", https://arxiv.org/pdf/1904.06184.pdf, 2019.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

A Possible algorithmic implementations

A.1 Algorithms from Section 1

A.1.1 Computing finite spans

The following algorithm computes the span of a configuration C, provided it is finite, as in the proof of Proposition 1.8.

Algorithm 1 SPAN(G, C)

```
1: S ← C
2: C_recent ← C
3: stop ← 0
4: while stop = 0 do
5:     C_new ← ∅
6:     for all c ∈ C_recent do
7:         for all p ∈ N(c) do
8:             for all q ∈ N(p) \ {c} do
9:                 if p ∉ S and q ∈ S then
10:                    S ← S ∪ {p}
11:                    C_new ← C_new ∪ {p}
12:                end if
13:            end for
14:        end for
15:    end for
16:    if C_new = ∅ then
17:        stop ← 1
18:    else
19:        C_recent ← C_new
20:    end if
21: end while
22: return S
```

A.1.2 Converting $\mathscr{G}'$ into $\mathscr{G}$

Given a starting configuration A and a sequence $(a_t)_{1\leq t\leq s}$ of actions from A to some configuration B in $\mathscr{G}'$, the algorithm CLEAN returns a sequence of moves from A to B in $\mathscr{G}$, emulating the proof of Proposition 1.13. The arrays C and P used during the algorithm are defined in the remark that follows Proposition 1.13.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Algorithm 2 CLEAN$(A, (a_t)_{1\leq t\leq T})$

$Q \leftarrow ()$ (empty queue)
for $t \leftarrow 1$ **to** T **do**
    if a_t is a *take* at o **then**
        $Q \leftarrow \text{push}(Q, o)$
    else
        if a_t is a *drop* at d **then**
            $(Q, o) \leftarrow \text{pop}(Q)$
            $m_t \leftarrow (o \mapsto d)$
        else
            $m_t \leftarrow a_t$
        end if
    end if
end for
remove all empty moves from $(m_t)_{1\leq t\leq T}$
let $(m_t)_{1\leq t\leq T'}$ be the remaining list
for all $a \in A$ **do**
    $C[a] \leftarrow 1$
end for
for $t \leftarrow 1$ **to** T' **do**
    $o \leftarrow P[o_t]$
    $d \leftarrow d_t$
    $P[o_t] \leftarrow o_t$
    if $C[d] = 1$ **then**
        $\widetilde{m}_t \leftarrow \varnothing$
        $P[d] \leftarrow o$
    else
        $\widetilde{m}_t \leftarrow m_t$
        $C[o] \leftarrow 0$
        $C[d] \leftarrow 1$
    end if
end for
remove all empty moves from $(\widetilde{m}_t)_{1\leq t\leq T'}$
return the remaining list $(\widetilde{m}_t)_{1\leq t\leq T''}$

A.2 Algorithms from Section 2

We implement the triangular grid as in Figure 79, identifying each position $p \in V$ with its coordinates $(p^{(1)}, p^{(2)}) \in \mathbb{Z}^2$. A triangle is implemented as a triplet (l, c, r) where: $r = l + (1, 0)$ and $c = l + (0, 1)$ if the triangle points upwards, or $r = l + (1, 0)$ and $c = l + (1, -1)$ if the triangle points downwards.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

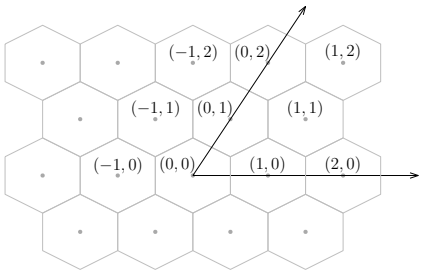

Figure 79. Implementation of the triangular grid.

A.2.1 Triangle manoeuvring

Algorithm 3 TRIANGLE_RIGHT$((l, c, r))$

$M \leftarrow (l \mapsto c + (1, 0), c \mapsto r + (1, 0))$
$T \leftarrow (l + (1, 0), c + (1, 0), r + (1, 0))$
return (M, T)

Algorithm 4 TRIANGLE_LEFT$((l, c, r))$

$M \leftarrow (r \mapsto c + (-1, 0), c \mapsto l + (-1, 0))$
$T \leftarrow (l + (-1, 0), c + (-1, 0), r + (-1, 0))$
return (M, T)

Algorithm 5 TRIANGLE_UP$((l, c, r))$

if (l, c, r) points upwards **then**
    $M \leftarrow (l \mapsto r + (0, 1), r \mapsto c + (0, 1))$
else
    $M \leftarrow (c \mapsto l + (0, 1), l \mapsto r + (0, 1))$
end if
$T \leftarrow (l + (0, 1), c + (0, 1), r + (0, 1))$
return (M, T)

Algorithm 6 TRIANGLE_DOWN$((l, c, r))$

if (l, c, r) points upwards **then**
    $M \leftarrow (c \mapsto r + (0, -1), r \mapsto l + (0, -1))$
else
    $M \leftarrow (r \mapsto l + (0, -1), c \mapsto r + (0, -1))$
end if
$T \leftarrow (l + (0, -1), c + (0, -1), r + (0, -1))$
return (M, T)

If T is a triangle, TRIANGLE_DOWN(T) returns the list of moves needed to bring T down on the board by one unit, as well as the new triangle thus obtained (same for TRIANGLE_RIGHT, TRIANGLE_LEFT, TRIANGLE_UP). If T_1 and T_2 are triangles, TRIANGLE_MOVE(T_1, T_2) returns a sequence of moves from T_1 to T_2, with no regard for coin stacking.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

Algorithm 7 TRIANGLE_MOVE$((l_1, c_1, r_1), (l_2, c_2, r_2))$

if (l_1, c_1, r_1) and (l_2, c_2, r_2) point in the same direction **then**
    $M \leftarrow ()$
    $(l_0, c_0, r_0) \leftarrow (l_1, c_1, r_1)$
else
    $M \leftarrow (l_1 \mapsto c_1 + (1, 0))$
    $(l_0, c_0, r_0) \leftarrow (c_1, r_1, c_1 + (1, 0))$
end if
$T_{curr} \leftarrow (l_0, c_0, r_0)$
if $l_0^{(2)} < l_2^{(2)}$ **then**
    for $k \leftarrow 1$ **to** $l_2^{(2)} - l_0^{(2)}$ **do**
        $(M_{part}, T_{curr}) \leftarrow$ TRIANGLE_UP(T_{curr})
        $M \leftarrow M + M_{part}$
    end for
else
    for $k \leftarrow 1$ **to** $l_0^{(2)} - l_2^{(2)}$ **do**
        $(M_{part}, T_{curr}) \leftarrow$ TRIANGLE_DOWN(T_{curr})
        $M \leftarrow M + M_{part}$
    end for
end if
if $l_0^{(1)} < l_2^{(1)}$ **then**
    for $k \leftarrow 1$ **to** $l_2^{(1)} - l_0^{(1)}$ **do**
        $(M_{part}, T_{curr}) \leftarrow$ TRIANGLE_RIGHT(T_{curr})
        $M \leftarrow M + M_{part}$
    end for
else
    for $k \leftarrow 1$ **to** $l_0^{(1)} - l_2^{(1)}$ **do**
        $(M_{part}, T_{curr}) \leftarrow$ TRIANGLE_LEFT(T_{curr})
        $M \leftarrow M + M_{part}$
    end for
end if
return M

A.2.2 Solving puzzles

Algorithm 8 BREAK(B)

$B^- \leftarrow B$
$L \leftarrow ()$
while there exists a path $c_1 - c_2 - c_3$ (not necessarily induced) in B^- **do**
    $B^- \leftarrow B^- \setminus \{c_2\}$
    $L \leftarrow L + (c_2)$
end while
return L

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.

The algorithm BREAK takes a configuration B as an input and returns a maximal sequence $(b_1, \ldots, b_e)$ of coins in B such that, for all $i \in [\![1, e]\!]$, $B \setminus \{b_1, \ldots, b_{i-1}\}$ contains at least two coins adjacent to b_i. Finally, if A and B (where $|A| = |B|$) satisfy either (i), (ii) or (iii), SOLVE(A, B) returns a sequence of moves from A to B in the original game.

Algorithm 9 SOLVE(A, B)

$L = (b_1, \ldots, b_e) \leftarrow$ BREAK(B)
let $C_1, \ldots, C_{n_1+n_2}$ be the connected components of $B^- = B \setminus \{b_1, \ldots, b_e\}$ (size 1 first, then size 2)
let $a_1, a_2, a_3 \in A$ distinct such that a_1 and a_2 have a common unoccupied neighbour p
$M \leftarrow (a_3 \mapsto p)$
if $p_1(\{a_1, p\}) \notin A \setminus \{a_3\}$ **then**
    $M \leftarrow M + (a_2 \mapsto p_1(\{a_1, p\}))$
end if
$T \leftarrow T(\{a_1, p\})$
for all $a \in A \setminus T$ **do**
    $M \leftarrow M + (a \mapsto R)$
end for
for $i \leftarrow 1$ **to** $n_1 + n_2$ **do**
    $M \leftarrow M +$ TRIANGLE_MOVE$(T_{curr}, T(C_i))$
    if $i < n_1 + n_2$ **then**
        if $i \leq n_1$ **then**
            $M \leftarrow M + (R \mapsto p_3(C_i))$
        else
            $M \leftarrow M + (R \mapsto p_2(C_i), R \mapsto p_3(C_i))$
        end if
        $T_{curr} \leftarrow T'(C_i)$
    else
        if $n_2 = 0$ **then**
            $M \leftarrow M + (p_1(C_i) \mapsto R, p_2(C_i) \mapsto R)$
        else
            $M \leftarrow M + ((p_1(C_i) \mapsto R)$
        end if
    end if
end for
for $i \leftarrow 1$ **to** e **do**
    $M \leftarrow M + (R \mapsto b_{e-i+1})$
end for
$M \leftarrow$ CLEAN(A, M)
return M

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program *Investissements d'Avenir*.