Coin-Moving Puzzles

Erik D. Demaine* Martin L. Demaine* Helena A. Verrill†

Abstract

We introduce a new family of one-player games, involving the movement of coins from one configuration to another. Moves are restricted so that a coin can be placed only in a position that is adjacent to at least two other coins. The goal of this paper is to specify exactly which of these games are solvable. By introducing the notion of a constant number of extra coins, we give tight theorems characterizing solvable puzzles on the square grid and equilateral-triangle grid. These existence results are supplemented by polynomial-time algorithms for finding a solution.

1 Introduction

Consider a configuration of coins such as the one on the left of Figure 1. The player is allowed to move any coin to a position that is determined rigidly by incidences to other coins. In other words, a coin can be moved to any position adjacent to at least two other coins. The puzzle or 1-player game is to reach the configuration on the right of Figure 1 by a sequence of such moves. This particular puzzle is most interesting when each move is restricted to *slide* a coin in the plane without overlapping other coins.

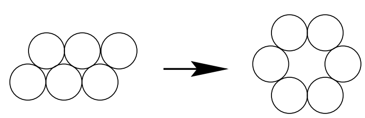

Figure 1: Re-arrange the rhombus into the circle using three slides, such that each coin is slid to a position adjacent to two other coins.

This puzzle is described in Gardner's Mathematical Games article on Penny Puzzles [7], in *Winning Ways* [1], in *Tokyo Puzzles* [6], in *Moscow Puzzles* [8], and in *The Penguin Book of Curious and Interesting Puzzles* [11]. Langman [9] shows all 24 ways to solve the puzzle in three moves. Another classic puzzle of this sort [2, 6, 7, 11] is shown in Figure 2. A final classic puzzle that originally motivated our work is shown in Figure 3; its source is unknown. Other related puzzles are presented by Dudeney [5], Fujimura [6], and Brooke [4].

*Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, email: {eddemaine, mldemaine}@uwaterloo.ca

†Institut for Matematiske Fag, Universitetsparken 5, DK-2100 København, Denmark, email: verrill@math.ku.dk

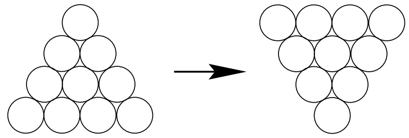

Figure 2: Turn the pyramid upside-down in three moves, such that each coin is moved to a position adjacent to two other coins.

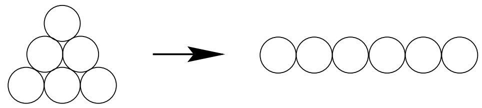

Figure 3: Re-arrange the pyramid into a line in seven moves, such that each coin is moved to a position adjacent to two other coins.

The puzzles above always move the centers of coins to vertices of the equilateral-triangle grid. Another type of puzzle is to move coins on the square grid, which appears less often in the literature but has significantly more structure and can be more difficult. The only published example we are aware of is given by Langman [10], which is also described by Brooke [4], Bolt [3], and Wells [11]; see Figure 4. The first puzzle (H $\rightarrow$ O) is solvable on the square grid, and the second puzzle (O $\rightarrow$ H) can only be solved by a combination of the two grids.

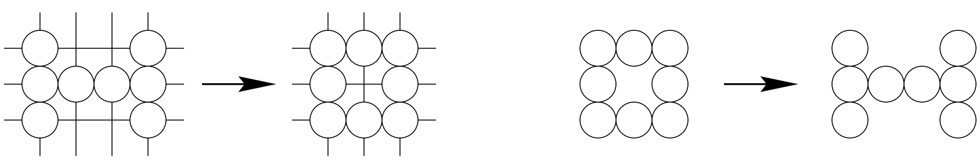

Figure 4: Re-arrange the H into the O in four moves while staying on the square grid (and always moving adjacent to two other coins), and return to the H in six moves using both the equilateral-triangle and square grids.

In this paper we study generalizations of these types of puzzles, in which coins are moved on some grid to positions adjacent to at least two other coins. Specifically, we address the basic algorithmic problem: is it possible to solve a puzzle with given start and finish configurations, and if so, find a sequence of moves. Surprisingly, we show that this problem has a polynomial-time solution in many cases. Our goal in this pursuit is to gain a better understanding of what is possible by these motions, and as a result to design new and interesting puzzles. For example, one puzzle we have designed is shown in Figure 5. We recommend the reader try this difficult puzzle before reading Section 5.3.1 which shows how to solve it. Figures 6–9 show a few of the other puzzles we have designed. The last two puzzles involve labeled coins.

This paper studies two grids in particular: the equilateral-triangle grid, and the square grid. It turns out that the triangular grid has a relatively simple structure, and nearly all puzzles are solvable. An exact, efficient characterization of solvable puzzles is presented in Section 3. The square grid has a more complicated structure, requiring us to introduce the

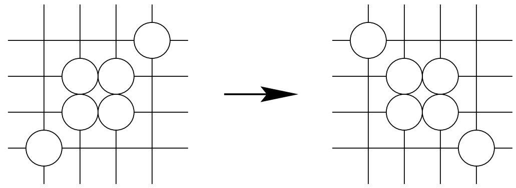

Figure 5: A difficult puzzle on the square grid. The optimal solution uses 18 moves, each of which places a coin adjacent to two others.

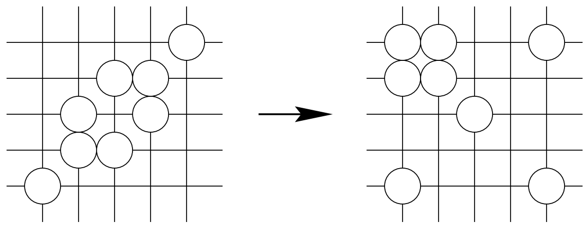

Figure 6: Another puzzle on the square grid. The optimal solution uses 24 moves, each of which places a coin adjacent to two others.

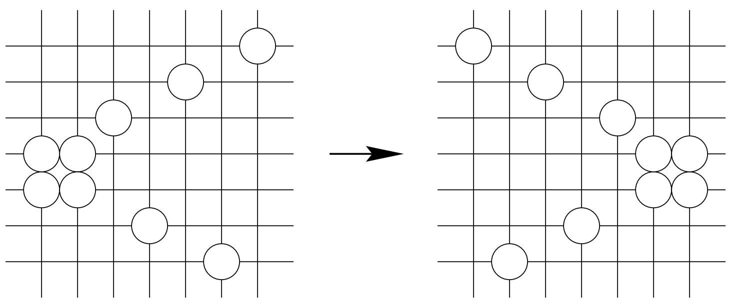

Figure 7: Another puzzle on the square grid with the same rules.

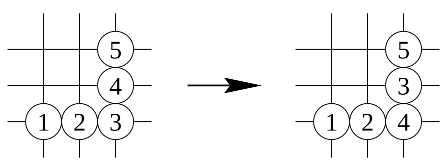

Figure 8: A puzzle on the square grid involving labeled coins. Solvable in eleven moves, each of which places a coin adjacent to two others; see Figure 31.

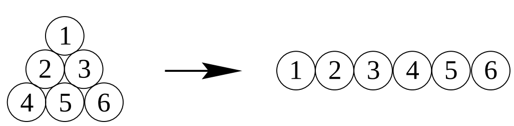

Figure 9: A puzzle on the equilateral-triangle grid involving labeled coins. Solvable in eight moves, each of which places a coin adjacent to two others.

notion of "extra coins" to give a partial characterization of solvable puzzles. This result is described in Section 5 after some general tools for analysis are developed in Section 4.

Before we begin, the next section defines a general graph model of the puzzles under consideration.

2 Model

We begin by defining "token-moving" and "coin-moving" puzzles and related concepts. The *tokens* form a finite multiset T. We normally think of tokens as unlabeled, modeled by all elements of T being equal, but another possibility is to color tokens into more than one equivalence class (as in Figure 9). A *board* is any simple undirected graph $G = (P, E)$, possibly infinite, whose vertices are called *positions*. A *configuration* is a placement of the tokens onto distinct positions on the board, i.e., a one-to-one mapping $C : T \to P$. We will often associate a configuration C with its image, that is, the set of positions *occupied* by tokens.

A *move* from a configuration C changes the position of a single token t to an unoccupied position p, resulting in a new configuration. This move is denoted $t \mapsto p$, and the resulting configuration is denoted $C \,/\, t \mapsto p$. We stress that moves are not required to "slide" the token while avoiding other tokens (like the puzzle in Figure 1); the token can be picked up and placed in any unoccupied position.

The *configuration space* (or *game graph*) is the directed graph whose vertices are configurations and whose edges correspond to feasible moves. A typical *token-moving puzzle* asks for a sequence of moves to reach one configuration from another, i.e., for a path between two vertices in the configuration space, subject to some constraints. A *coin-moving puzzle* is a geometric instance of a token-moving puzzle, in which tokens are represented by *coins*—constant-radius disks in the plane, and constant-radius hyperballs in general—and the board is some lattice in the same dimension. If a token-moving or coin-moving puzzle with source configuration A and destination configuration B is solvable, we say that A can be *re-arranged* into B, and that B is *reachable* from A. This is equivalent to the existence of a directed path from A to B in the configuration space.

This paper addresses the natural question of what puzzles are solvable, subject to the following constraint on moves which makes the problem interesting. A move $t \mapsto p$ is *d-adjacent* if the new position p is adjacent to at least d tokens other than the moved token t. (Throughout, *adjacency* refers to the board graph G.) This constraint is particularly meaningful for d-dimensional coin-moving puzzles, because then a move is easy to "perform exactly" without any underlying lattice: the new position p is determined rigidly by the d coin adjacencies (sphere tangencies).

The *d-adjacency configuration space* is the subgraph of the configuration space in which moves are restricted to be d-adjacent. Studying connectivity in this graph is equivalent to studying solvable puzzles; for example, if the graph is strongly connected, then all puzzles are solvable.

Here we explore solvable puzzles on two boards, the equilateral-triangle grid and the square grid. Because these puzzles are two-dimensional, in the context of this paper we call a move *valid* if it is 2-adjacent, and a position a *valid destination* if it is unoccupied and adjacent to at least two occupied positions. Thus a valid move involves transferring a token from some source position to a valid destination position. When the context is clear, we will refer to a valid move just by "move." A move is *reversible* if the source position is also a valid destination.

3 Triangular Grid

This section studies the equilateral-triangle grid, where most puzzles are solvable. To state our result, we need a simple definition. Associated with any configuration is the subgraph of the board induced by the occupied positions. In particular, a *connected component* of a configuration is a connected component in this induced subgraph.

Theorem 1 *On the triangular grid with the 2-adjacency restriction and unlabeled coins, configuration A can be re-arranged into a different configuration B precisely if A has a valid move, the number of coins in A and B match, and at least one of four conditions holds:*

1. *B contains three coins that are mutually adjacent (a triangle).*
2. *B has a connected component with at least four coins.*
3. *B has a connected component with at least three coins and another connected component with at least two coins.*
4. *There is a single move from A to B.*

The same result holds for labeled coins, except when there are exactly three coins in the puzzle, in which case the labelings and movements are controlled by the vertex 3-coloring of the triangular grid.

Furthermore, there is a polynomial-time algorithm to find a re-arrangement from A to B if one exists. Specifically, let n denote the number of coins and d denote the maximum distance between two coins in A or B. Then a solution with $O(nd)$ moves can be found in $O(nd)$ time.

The rest of this section is devoted to the proof of this theorem. We begin in the next subsection by proving necessity of the conditions: if a puzzle is solvable, then one of the conditions holds. Then in the following subsection we prove sufficiency of the conditions.

3.1 Necessity

Of course, it is necessary for A to have a valid move and for A and B to have the same number of coins. Necessity of at least one of the four conditions is also not difficult to show, because Conditions 1–3 are so broad, encompassing most possibilities for configuration B.

Suppose that a solvable puzzle does not satisfy any of Conditions 1–3, as in Figure 10. We prove that it must satisfy Condition 4, by considering play backwards from the goal configuration B. Specifically, a *reverse move* takes a coin currently adjacent to at least two others, and moves it to any other location. Because the puzzle is solvable, some coin in configuration B must be reverse-movable, i.e., must have at least two coins adjacent to it.

Thus, some connected component of B has at least three coins. Because Condition 2 does not hold, this connected component has exactly three coins. Because Condition 1 does not hold, these three coins are not connected in a triangle. Because Condition 3 does not hold, every other component has exactly one coin.

Hence, one component of B is a path of exactly three coins, say c_1, c_2, c_3, and every other component of B has exactly one coin, as in the left of Figure 10. Certainly at this moment c_2 is the only reverse-movable coin. We claim that after a sequence of reverse moves, c_2 will continue to be the only reverse-movable coin. If we removed c_2, then every coin would be adjacent to no others. Thus, if we reverse move c_2 somewhere, then every other coin would be adjacent to at most one other (c_2). Hence, it remains that only c_2 can be reverse moved.

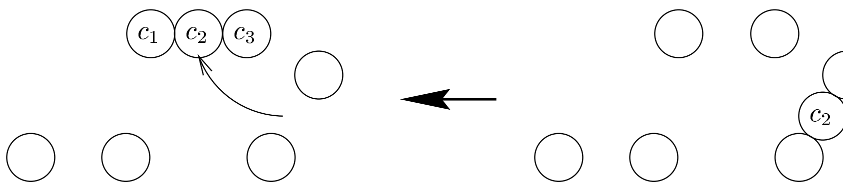

Figure 10: Reverse-moving a configuration B that does not satisfy any of Conditions 1–3.

Therefore, if we can reach A from B via reverse moves, we can do so in a single reverse move of c_2 directly to where it occurs in A. Thus Condition 4 holds, as desired.

3.2 Sufficiency

Next we prove the more difficult direction: provided one of Conditions 1–3 hold, there is a re-arrangement from A to B. (This fact is obvious when Condition 4 holds.) All three cases will follow a common outline: we first form a triangle (Section 3.2.1), then maneuver this triangle (Section 3.2.2) to transport all other coins (Section 3.2.3), and finally we place the three triangle coins appropriately depending on the case (Section 3.2.4).

3.2.1 Getting Started

It is quite simple to make some triangle of coins. By assumption, there is a valid move from configuration A. The destination of this move can have two basic forms, as shown in Figure 11. Either the move forms a triangle, as desired, or the move forms a path of three coins. In the latter case, if there is not a triangle already with a different triple of coins, a triangle can be formed by one more move as shown in the right of the figure.

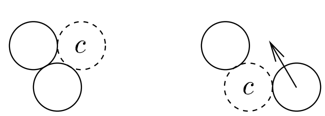

Figure 11: Two types of valid destinations for a coin c. In the latter case, we show a move to form a triangle.

This triangle T_0 suffices for unlabeled coin puzzles. However, for labeled coin puzzles, we cannot use just any three coins in the triangle; we need a particular three, depending on B.

For example, if B satisfies Condition 1, then the coins forming the triangle in B are the coins we would like in the triangle for maneuvering. To achieve this, we "bootstrap" the triangle T_0 formed above, using this triangle with the incorrect coins to form another triangle with the correct coins. Specifically, if we desire a triangle using coins t_1, t_2, and t_2, then we move each coin in the difference $\{t_1, t_2, t_3\} - T_0$ to be adjacent to appropriate coins in T_0. There are three cases, shown in Figure 12, depending on how many coins are in the difference. If ever we attempt to move a coin to an already occupied destination, we first move the coin located at that destination to any other valid destination.

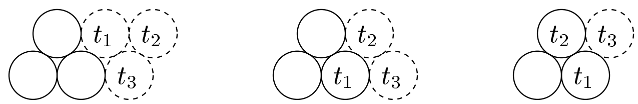

Figure 12: The three cases of building a triangle $\{t_1, t_2, t_3\}$ out of an existing triangle, depending on how many coins the two triangles share. From left to right, zero, one, and two coins of overlap.

3.2.2 Triangle Maneuvering

Consider a triangle of coins t_1, t_2, and t_3. The possible positions of this triangle on the triangular grid are in one-to-one correspondence with their centers, which are vertices of the dual hexagonal grid. Moving one coin (say t_1) to be adjacent to and on the other side of the others (t_2 and t_3) corresponds to moving the center of the triangle to one of the three neighboring centers on the hexagonal grid. Thus, without any other coins on the board, the triangle can be moved to any position by following a path in the hexagonal grid.

This approach can be modified to apply when there are additional obstacle coins; see Figure 13 for an example. Conceptually we always move one of the triangle coins, say t_i, in order to move the center of the triangle to an adjacent vertex of the hexagonal grid. But if the move of t_i is impossible because the destination is already occupied by another coin g_i, then in fact we do not make any move. There will be a triangle in the desired position now, but it will not consist of the usual three coins (t_1, t_2, and t_3); instead, t_i will be replaced by the "ghost coin" g_i. Such a triangle suffices for our purposes of transportation described in Section 3.2.3. One final detail is how the ghost coins behave: if we later need to move a ghost coin g_i, we instead move the original (unmoved) coin t_i. Thus ghost coins are never moved; only t_1, t_2, and t_3 are moved during triangle maneuvering (even if coins are labeled).

3.2.3 Transportation

Triangle maneuvering makes it easy to *transport* any other coin to any desired location. Specifically, suppose we want to move coin $c \notin \{t_1, t_2, t_3\}$ to destination position d. If d is already occupied by another coin c', we first move c' to an arbitrary valid destination; there is at least one because the triangle can be maneuvered. Now we maneuver the triangle so that the (potentially ghost) triangle has two coins adjacent to d, so that the third coin is not on d, and so that the triangle does not overlap c. This is easily arranged by examining the location of c and setting the destination of the triangle appropriately. For example, if c is within distance two of d, then there are four positions for the triangle that are adjacent

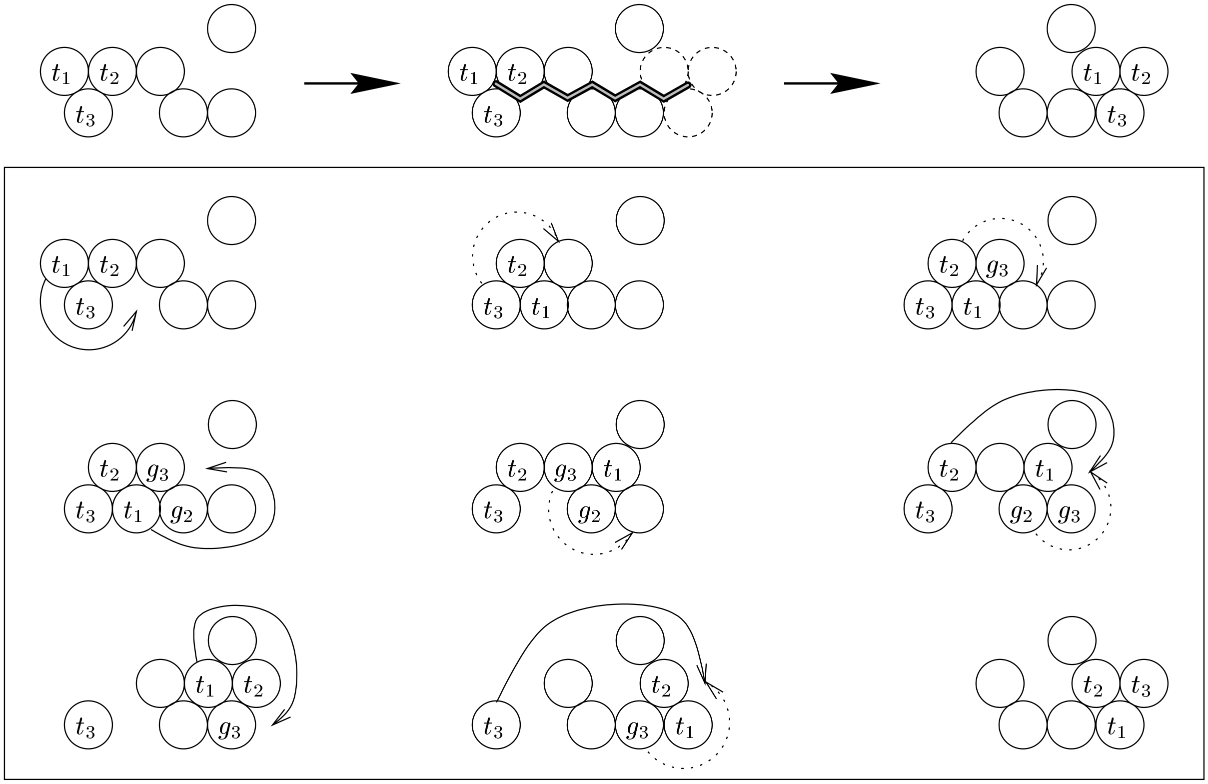

Figure 13: An example of triangle maneuvering. Dotted arrows denote conceptual moves, and solid arrows denote actual moves.

to d and do not overlap c; otherwise, the triangle can be placed in any of the six positions adjacent to d. Finally, because d is now a valid destination—it is adjacent to two coins in the triangle—we can move c to d.

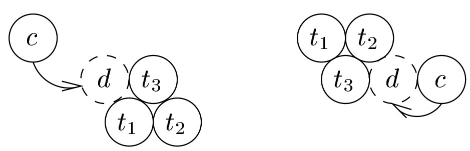

Figure 14: Transporting coin c to destination d using triangle $\{t_1, t_2, t_3\}$. In both cases, we choose the location of the triangle so that it does not overlap c.

By the properties of triangle maneuvering, this transportation process even preserves coin labels: the only actual coins moved are t_1, t_2, t_3, c, and possibly a coin at d. But any coin at position d must not have already been in its desired position, because d is c's desired position. Thus, applying the transportation process to every coin except t_1, t_2, and t_3 places all coins except these three in their desired locations.

3.2.4 Finale

Once transportation is complete, it only remains to place the triangle coins t_1, t_2, and t_3 in their desired locations. By the bootstrapping in Section 3.2.1, we are able to choose the unplaced coins $\{t_1, t_2, t_3\}$ however we like. This property will be exploited differently in the three cases.

Property 1. If there is a triangle in B, then we choose these three coins as the unplaced coins t_1, t_2, and t_3, and use them to transport all other coins. Then we maneuver the triangle $\{t_1, t_2, t_3\}$ exactly where it appears in B. Because all other coins have been moved to their proper location, in this position the triangle will not have any ghost coins.

However, it may be that the coins $\{t_1, t_2, t_3\}$ are labeled incorrectly among themselves, compared to B. Assuming there are more than three coins in the puzzle, this problem can be repaired as follows. We maneuver the triangle so that it does not overlap any other coins but is adjacent to at least one coin c; for example, there is such a position for the triangle just outside the smallest enclosing hexagon of the other coins. Refer to Figure 15. Now two coins of the triangle, say t_1 and t_2, are adjacent to three other coins each: each other, t_3, and c. Thus we can move t_1 to any other valid destination, and then move t_2 or t_3 to replace it. Afterwards we can move t_1 to take the place of t_2 or t_3, whichever moved. This procedure swaps t_1 and either t_2 or t_3. By suitable application, we can achieve any permutation of $\{t_1, t_2, t_3\}$, and thereby achieve the desired labeling of the triangle.

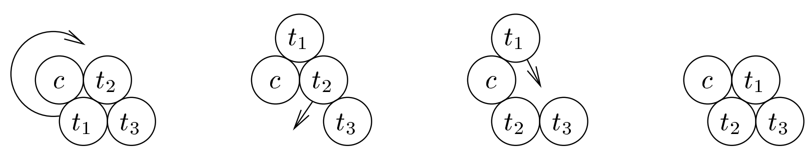

Figure 15: Swapping coins t_1 and t_2 in a triangle, using an adjacent coin c.

Property 2 but not Property 1. Refer to Figure 16. If there is not a triangle in B, but there is a connected component of B with at least four coins, then there is a path in B of length four, (p_1, p_2, p_3, p_4). From B we reverse move p_2 so that it is adjacent to p_3 and p_4. If this position is already occupied by a coin c, we first reverse move c to any other unoccupied position. Now p_2, p_3, and p_4 are mutually adjacent, so we have a new destination configuration B' with Property 1. As described above, we can re-arrange A into B'. Then we undo our reverse moves: move p_2 back adjacent to p_1 and p_3, and move c back adjacent to p_3 and p_4. This procedure re-arranges A into B.

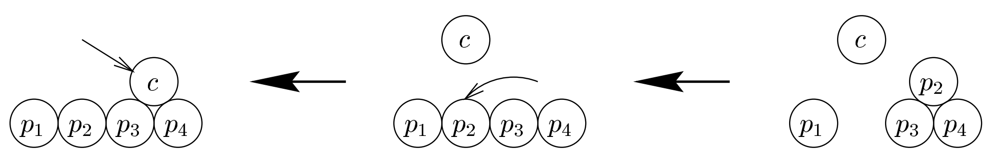

Figure 16: Reverse moving a configuration B with Property 2 into a configuration with Property 1.

Property 3 but not Property 1. This case is similar to the previous one; refer to Figure 17. There must be a path in B of length three, (p_1, p_2, p_3), as well as a pair of adjacent coins, (q_1, q_2), in different connected components of B. If both positions adjacent to both q_1 and q_2 are already occupied, we first reverse move one such coin (call it c) to an arbitrary unoccupied position. This frees up a position adjacent to q_1 and q_2, to which we reverse move p_2. Now $\{q_1, q_2, p_2\}$ form a triangle, so Property 1 holds, and we can reach this new configuration B' from A. Then we undo our reverse moves: move p_2 back adjacent to p_1 and p_3, and move c back adjacent to q_1 and q_2. This procedure re-arranges A into B.

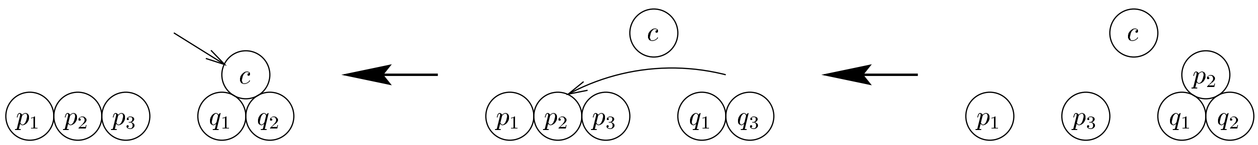

Figure 17: Reverse moving a configuration B with Property 3 into a configuration with Property 1.

This concludes the proof of Theorem 1.

4 General Tools

In this section we develop some general lemmas about token-moving puzzles. Although we only use these tools for the square grid, in Section 5, they apply to arbitrary boards and may be of more general use.

4.1 Picking Up and Dropping Tokens

First we observe that additional tokens cannot "get in the way":

Lemma 1 *If a token-moving puzzle is solvable, then it remains solvable if we add an additional token with an unspecified destination, provided tokens are unlabeled. This result also holds if all moves must be reversible.*

Proof: A move can be blocked by an extra token e at position p because p is occupied and hence an invalid destination. But if ever we encounter such a move of a token t to position p, we can just ignore the move, and swap the roles of e and t: treat e as the moved version of t, and treat t as an extra token replacing e. Thus, any sequence of moves in the original puzzle can be emulated by an equivalent sequence of moves in the augmented puzzle. We are not introducing any new moves, only removing existing moves, so all moves remain reversible if they were originally. $\square$

This proof leads to a technique for emulating a more powerful model for solving puzzles. In addition to moving coins as in the normal model, we can conceptually *pick up* (remove) a token, and later *drop* (add) it onto any valid destination. At any moment we can have any number of tokens picked up. While a token t is conceptually picked up, we emulate any moves to its actual position p as in the proof of Lemma 1: if we attempt to move another token t' onto position p, we instead reverse the roles of t and t'. To drop a token onto a desired position p, we simply move the actual token to position p if it is not there already.

Of course, this process may permute the tokens. Nonetheless we will find this approach useful for puzzles with labeled tokens.

One might instead consider the emulation method used implicitly in Section 3.2.2 for triangular maneuvering: move original coins instead of ghost coins. This approach has the advantage that it preserves the labels of the coins. Unfortunately, the approach makes it difficult to preserve reversibility as in Lemma 1, and so is insufficient for our purposes here.

4.2 Span

The *span* of a configuration C is defined recursively as follows. Let $d_1, \ldots, d_m$ be the set of valid destinations for moves in C. If $m = 0$, the span of C is just C itself. Otherwise, it is the span of another configuration C', defined to be C with additional tokens at positions $d_1, \ldots, d_m$. If this process never terminates, the span is defined to be the limit, which exists because it is a countable union of finite sets.

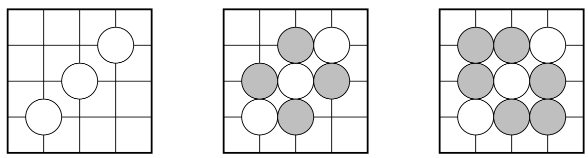

Figure 18: In this example, the span is the smallest rectangle enclosing the configuration.

The span of a configuration lists all the positions we could hope to reach, or more precisely, the positions we could reach if we had an unlimited number of extra tokens that we could drop. In particular, we have the following:

Lemma 2 *If configuration A can be re-arranged into configuration B, then* $\operatorname{span} A \supseteq B$ *and thus* $\operatorname{span} A \supseteq \operatorname{span} B$.

In other words, valid moves can never cause the span of the current configuration to increase. Thus the most connected we could hope the configuration space to be is the converse of Lemma 2: for every pair of configurations with $\operatorname{span} A \supseteq \operatorname{span} B$, A can be re-arranged into B. In words, we want that every configuration A can be re-arranged into any configuration B with the *same or smaller span.*

We call a configuration *span-minimal* if the removal of any of its tokens reduces the span. Span-minimal configurations are essentially the "skeleta" that keep configurations with the same span reachable. One general property of span-minimal configurations is the following:

Lemma 3 *If a configuration is span-minimal, any move will reduce the span.*

Proof: Suppose to the contrary that there is a move $t \mapsto p$ that does not reduce the span of a span-minimal configuration C. In particular, p must be a valid destination position in the subconfiguration $C - t$, because t does not count in the d-adjacency restriction. Hence, adding a new token at position p to the configuration $C - t$ has no effect on the span of C. But this two-step process of removing token t and adding a token at position p is equivalent to moving

t to p, so $\operatorname{span}(C \,/\, t \mapsto p) = \operatorname{span}(C - t)$. But we assumed that $\operatorname{span}(C \,/\, t \mapsto p) = \operatorname{span} C$, and hence $\operatorname{span} C = \operatorname{span}(C - t)$, contradicting that C is span-minimal. $\square$

Under the 2-adjacency restriction, a *chain* is a sequence of tokens with the property that the distance (in the board graph G) between two successive tokens is at most 2. We will use chains as basic "units" for creating a desired span.

Notice that the notion of span is useless for the already analyzed triangular grid: provided there is a valid move, the span of any configuration is the entire grid. Thus, for the triangular grid, a configuration is span-minimal precisely if it has no valid moves. For the square grid, however, the notion of span and span minimality is crucial.

4.3 Extra Tokens

As described in the previous section, we can only re-arrange configurations into configurations with the same or smaller span. Unfortunately, the converse is not true. Indeed, the key problem situations are span-minimal configurations; by Lemma 3, such configurations immediately lose span when we try to move them. Hence, any two distinct span-minimal configurations with the same span cannot reach each other. An example on the square grid is that the two opposite diagonals of a square are unreachable from each other, as shown in Figure 19.

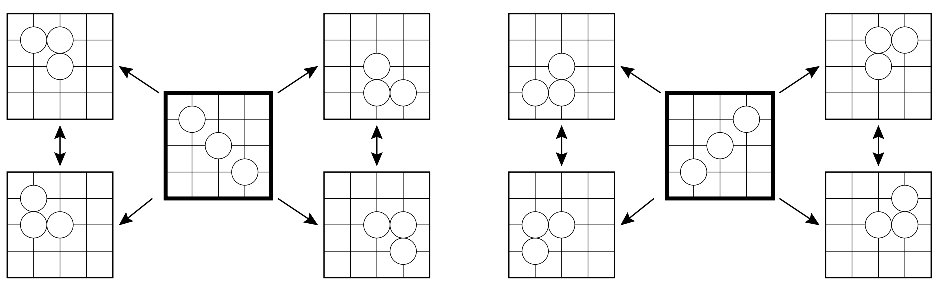

Figure 19: Subgraph of configuration space reachable from full-span configurations (outlined in bold) with no extra coins.

Thus we explore the notion of *extra tokens*, a set of tokens whose removal does not reduce the span of the configuration. Lemma 2 and Figure 19 shows that we need at least one extra token. In fact, the two opposite diagonals on the square grid shown in Figure 19 are difficult to reach from each other; as shown in Figure 20, even one extra token is insufficient. What is surprising is that a small number of extra tokens seem to be generally sufficient to make the configuration space strongly connected. We prove this for the square grid in the next section.

5 Square Grid

This section analyzes coin-moving puzzles on the square grid, using the tools from the previous section. In particular, we show that with just two extra coins, we can reach essentially every configuration on the square grid with the same or smaller span. The only restriction

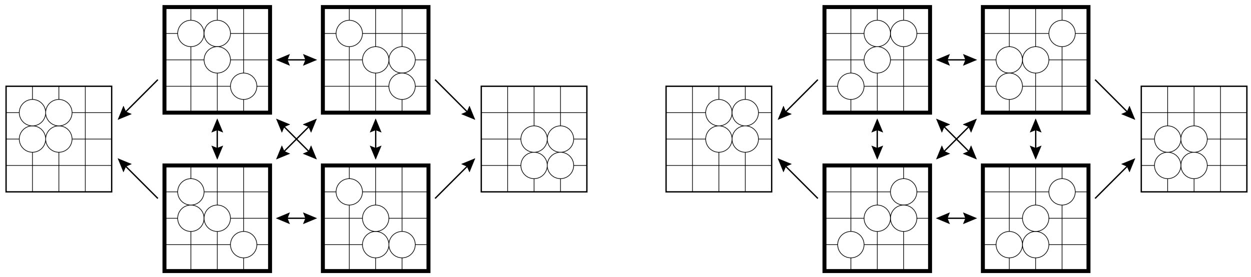

Figure 20: Subgraph of configuration space reachable from full-span configurations (outlined in bold) with one extra token.

is that the extra coins can only be destined for positions that are adjacent to at least two other coins.

Theorem 2 *On the square grid with the 2-adjacency restriction and unlabeled coins, configuration A can be re-arranged into configuration B if there are coins e_1 and e_2 such that* $\mathrm{span}(A - \{e_1, e_2\}) \supseteq \mathrm{span}(B - \{e_1, e_2\})$ *and each e_i is adjacent to two other coins in B (excluding e_1 or e_2). Furthermore, there is an algorithm to find such a re-arranging sequence using $O(n^3)$ moves and $O(n^3)$ time, where n is the number of coins.*

We prove this theorem by showing that every configuration (in particular, A and B) can be brought to a canonical configuration with the same span via a sequence of (mostly) reversible moves. As a consequence, we can move from any configuration A to any other B by routing through this canonical configuration.

Our proof uses the model of picking up and dropping coins, which can be emulated as described in Section 4.1. However, we must be careful how we pick up and drop coins, so that the resulting moves are reversible. For example, initially we pick up the extra coins e_1 and e_2, and then drop them temporarily wherever needed. For re-arranging the source configuration A into the canonical configuration, this step may not result in reversible moves, but fortunately this is not necessary in this case. For re-arranging the destination configuration B into the canonical configuration, however, reversibility is crucial, and is guaranteed by the condition in the theorem of each e_i being adjacent to at least two other coins.

5.1 Basics

We begin with some preliminary lemmas. A *rectangle* is the full collection of coins between two x coordinates and two y coordinates. The *half-perimeter* of a rectangle is the number of distinct x coordinates plus the number of distinct y coordinates over all coins in the rectangle. The *distance* between two sets of coins is the minimum distance between two coins from different sets.

Lemma 4 *For the square grid, the span of any configuration is a disjoint union of (finite) rectangles with pairwise distances at least* 3.

Lemma 5 *For each rectangle (connected component) of the span, say with half-perimeter h, there must be at least $\lceil h/2 \rceil$ coins within that rectangle in the configuration.*

The following beautiful proof of this lemma has been distributed among several people, but its precise origin is unknown. We first heard it from Martin Farach-Colton, who heard it from Peter Winkler, who heard it from Pete Gabor Zoltan, who learned of it through the Russian magazine *Kvant* (around 1985–1987).

Proof: Consider how the (full) perimeter changes as we compute the span of the coins within the rectangle. Initially we have n coins, say, so the perimeter is at most $4n$. Each coin that we add while computing the span satisfies the 2-adjacency restriction, so the perimeter never increases. In the end we must have a rectangle with perimeter $2h$. Hence $4n \geq 2h$, i.e., $n \geq h/2$, and because n is integral, $n \geq \lceil h/2 \rceil$. □

5.2 Canonical Configuration

Observe that a chain has span equal to its smallest enclosing rectangle. We define an *L* to be a particular kind of chain, starting and ending at opposite corners of the rectangular span, and arranged along two edges of this rectangle, with the property that it has the minimum number of coins. See Figure 21 for examples. More precisely, if the half-perimeter of the rectangle (along which the L is arranged) is $2k$, then there must be precisely k coins, every consecutive pair at distance exactly two from each other. And if the half-perimeter is $2k+1$, then there must be precisely $k+1$ coins, every consecutive pair at distance exactly two from each other, except the last pair which are distance one from each other. In general, for half-perimeter h, an L has $\lceil h/2 \rceil$ coins.

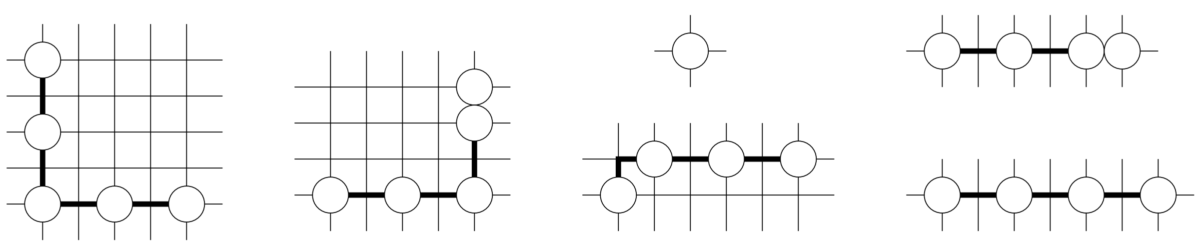

Figure 21: Examples of L's.

While L's can have any orientation, the *canonical L* is oriented like the letter L, starting at the top-left corner, continuing past the lower-left corner, and ending at the bottom-right corner.

Given a configuration, or more precisely, given its span and the number of coins in each connected component of the span, we define the *canonical configuration* as follows. Refer to Figure 22 for examples. Within each connected component (rectangle) of the span, say with half-perimeter h, we arrange the first $\lceil h/2 \rceil$ coins into the canonical L. (Lemma 5 implies that there are at least this many coins to place.) Any additional coins are placed one at a time, in the leftmost bottommost unoccupied position.

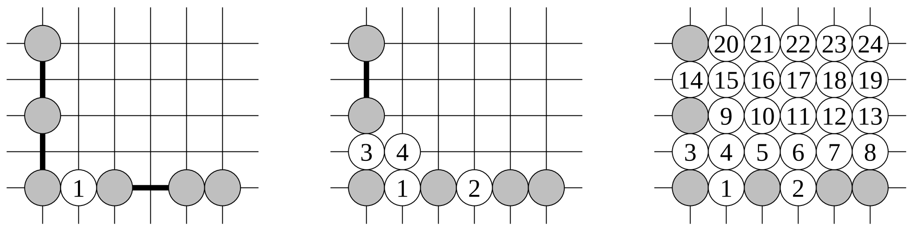

Figure 22: Examples of the canonical configuration of k coins within a rectangular span. (Left) One coin in addition to the canonical L. (Middle) Four coins in addition to the canonical L. (Right) All 24 additional coins.

This definition of the canonical configuration is fairly arbitrary, but it has the useful property that each successive position for an additional coin is a valid destination, given the previously placed additional coins. This allows us to focus on forming the canonical L, and then picking up all additional coins and dropping them in the order shown on the right of Figure 22.

5.3 Canonicalizing Algorithm

The main part of proving Theorem 2 is to show an algorithm for converting any configuration into the corresponding canonical configuration, using a sequence of (mostly) reversible moves. We will apply induction (or, equivalently, recursion) on the number of coins. That is, we assume that any configuration with fewer coins can be re-arranged into its canonical configuration.

For now, we assume that there are no extra coins in addition to e_1 and e_2. For if there were such a coin, we could immediately pick it up. Then we have a simpler configuration: it has one fewer coin. Thus we can apply the induction hypothesis, and re-arrange the remaining coins into their canonical configuration. Finally we must drop the previously picked-up coin in the appropriate location. This aspect is somewhat trickier than it may seem: if we are not careful, we may make an irreversible move. We delay this issue to Section 5.4.

The overall outline of the algorithm is as follows:

1. Initialize the set of L's to be one for each coin.
2. Until the configuration is canonical:
   (a) Pick two L's whose bounding rectangles are distance at most two from each other.
   (b) Re-orient the L's so that the L's themselves are distance at most three from each other.
   (c) Merge the two L's.

Normally, each iteration of Step 2 decreases the number of L's by one, so the algorithm would terminate in at most n iterations. However, at any time we may find an extra coin in addition to e_1 and e_2, and pick it up. Fortunately, this operation can only split one L into at most two L's. Thus we can charge the cost of creating an extra L to the event of picking up an extra coin, which can happen at most n times. Hence, the total number of iterations of Step 2 is $O(n)$.

In the following two sections, we describe how Steps 2(b) and 2(c) can be done in $O(n^2)$ moves each. These bounds result in a total of $O(n^3)$ moves. The running time of the algorithms will be proportional to the number of moves.

5.3.1 Re-orienting L's

There are eight possible orientations for an L, depending at which corner it starts, and whether it hugs the top edge or bottom edge of the rectangular span. We will only be concerned with four different types of orientations, depending on whether it looks like the letter L rotated 0, 90°, 180°, or 270°. In other words, we are not concerned with the parity issue of which corner might have two adjacent coins.

It is relatively easy to *flip* an L about a diagonal, using two extra coins. Figure 23 shows how to do this in a constant number of moves for an L consisting of three coins. Figure 24 shows how to use these subroutines to flip an L of arbitrary size. Basically, we use the flips of three-coin L's to "bubble" the kink in the L up to the top, repeatedly until it is all the way right. The total number of moves is $O(n^2)$, and they can easily be computed in $O(n^2)$ time.

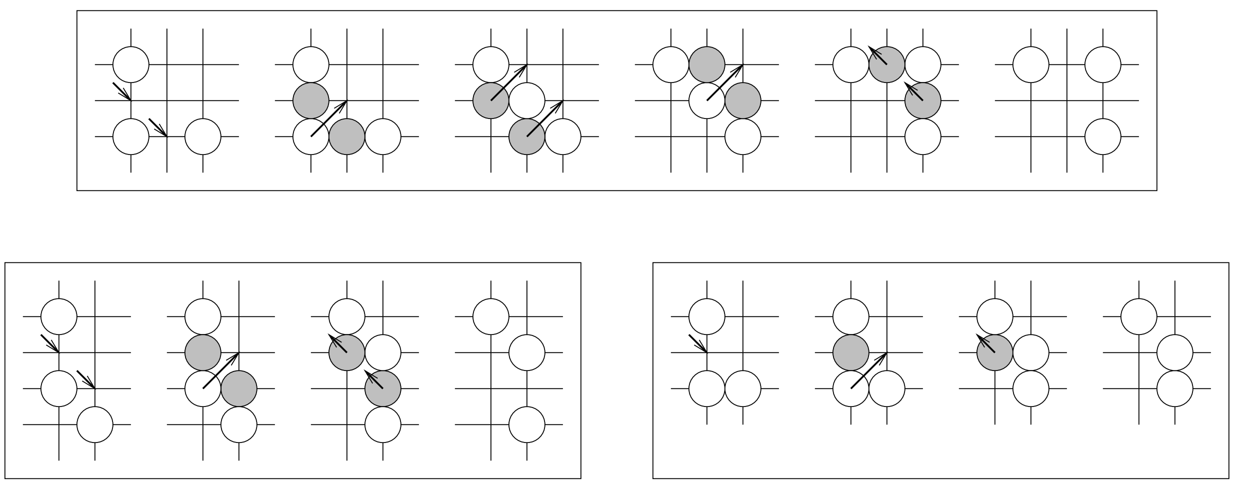

Figure 23: Flipping an L consisting of 3 coins. Extra coins are shaded.

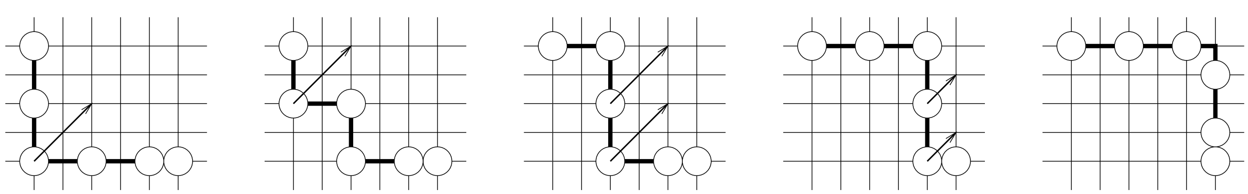

Figure 24: Flipping a general L, using the subroutines in Figure 23.

The more difficult re-orientation to perform is a *rotation* of an L by $\pm 90°$. Perhaps one of the most surprising results of this paper is that this operation is possible with two extra coins. One way to do it for a square span, shown in Figure 25, is to convert the L into a diagonal, and then convert more and more of the diagonal into a rotated L. This is the basis for our "diagonal-flipping" puzzle in Figure 5.

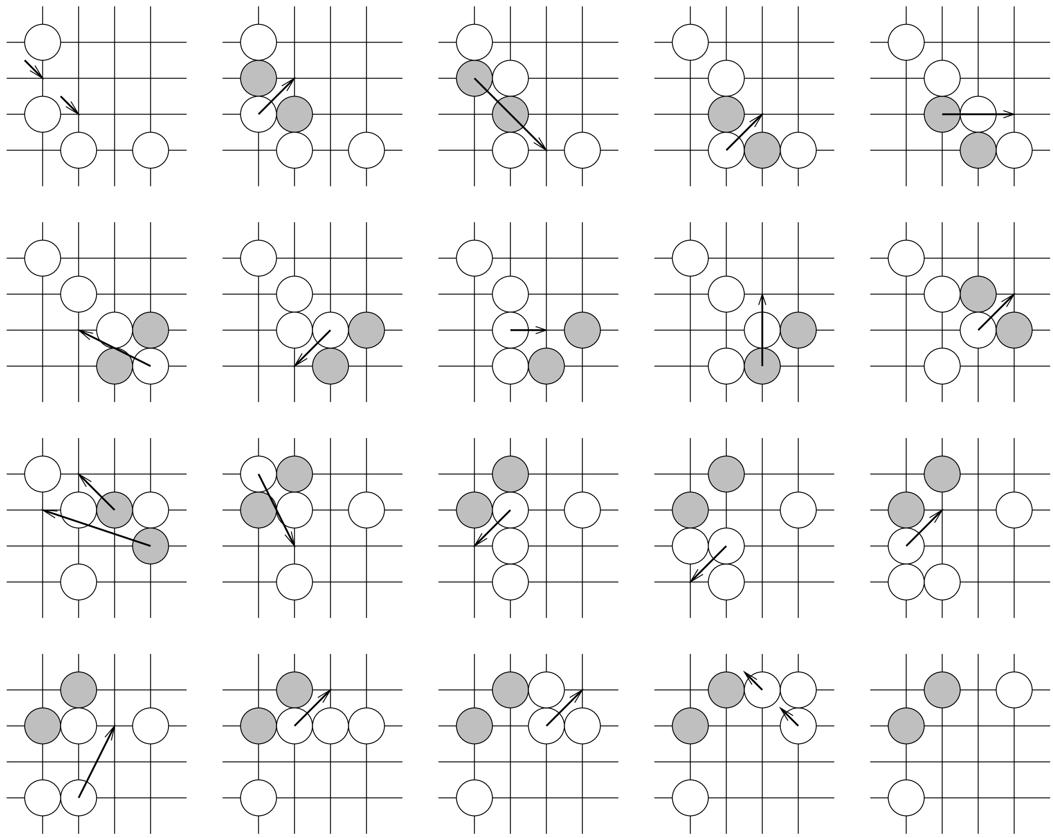

Figure 25: One method for rotating an L with a square span. Although this example places the extra coins in the final configuration, this is not necessary.

A simpler way to argue that L's can be rotated is shown in Figure 26. Assume without loss of generality that the initial orientation is the canonical L. First we apply induction to the subconfiguration of all coins except the top-left coin. Thus all rows except the third row contain at most one coin each, assuming the L consists of at least three rows. Now we apply local operations in 3×3 or 3×2 rectangles (similar to Figure 23) to move the top-left coin to the far right. (We cannot perform this left-to-right motion in one step using induction, because there may be only three rows, and hence all coins may be involved in this motion.) Finally we flip the L in the top three rows, as described above, thereby obtaining the desired result. Again the number of moves and computation time are both $O(n^2)$. Note that the same approach of repeated local operations applies when the L consists of only two rows.

5.3.2 Merging L's

Consider two L's L_1 and L_2 whose bounding rectangles R_1 and R_2 are distance at most two from each other. Equivalently, consider two L's such that $\text{span}(L_1 \cup L_2)$ has a single

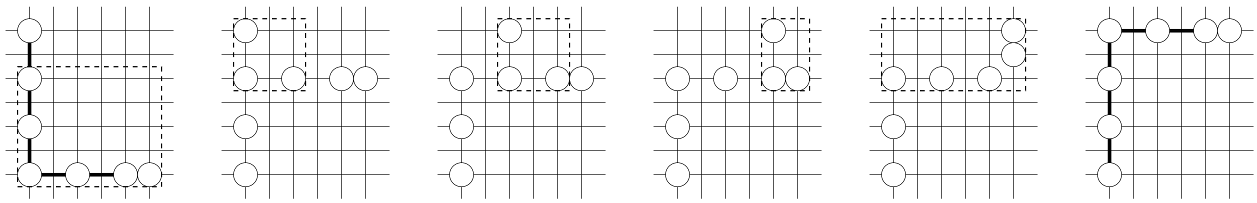

Figure 26: A general method for rotating an L. The first step is to apply induction, and the remaining steps apply subroutines similar to Figure 23.

connected component. This section describes how to merge L_1 and L_2 into a single L. This step is the most complicated part of the algorithm, not because it is difficult in any one case, but because there are many cases involved.

First suppose that the rectangles R_1 and R_2 overlap. We claim that one of the L's, say L_1, can be re-oriented so that one of its coins is contained in the other L's bounding rectangle, R_2. This coin is therefore in the span of the L_2, and hence redundant, so as described above we can apply induction and finish the entire canonicalization process.

To prove the claim, there are three cases; see Figure 27. If a corner of one of the bounding rectangles, say R_1, is in the other bounding rectangle, R_2, then we can re-orient L_1 so that one of its end coins is at that corner of R_1 and hence in R_2; see Figure 27(a)). Otherwise, we have rectangles that form a kind of "thick plus sign" (Figure 27(b–c)); we distinguish the two rectangles as according to whether they form the *horizontal stroke* or *vertical stroke* of the plus sign. If the vertical stroke has width at least two (Figure 27(b)), then that rectangle already contains a coin of the other L, because that L cannot have two empty columns. Similarly, if the horizontal stroke has height at least two, then that rectangle already contains a coin of the other L, because that L cannot have two empty rows. Finally, if both strokes are of unit thickness, and there is not already a coin in their single-position intersection (Figure 27(c)), then we can splice and redefine the L's, so that one L is formed by the top half of the vertical stroke and the left half of the horizontal stroke, and the other L is formed by the bottom half of the vertical stroke and the right half of the horizontal stroke, and then we have the first case in which the bounding rectangles share a corner.

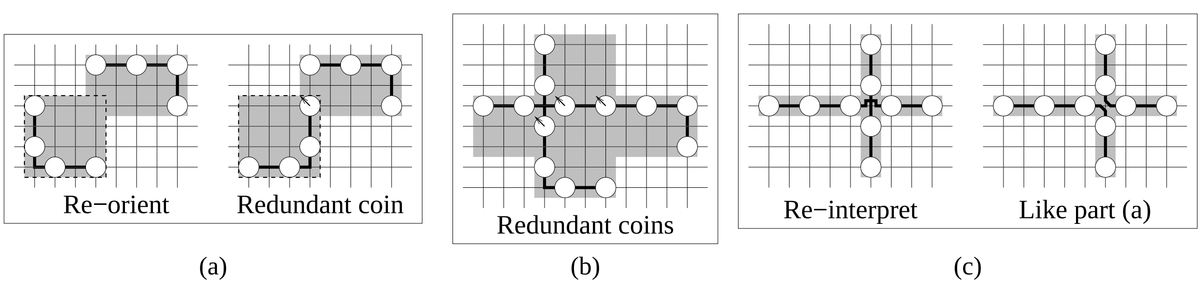

Figure 27: Merging two L's with overlapping bounding rectangles (shaded). (a) The corner of one L is contained in the other L's bounding rectangle. (b) A "thick plus sign" in which at least one stroke has thickness more than 1. (c) A plus sign in which both strokes have thickness 1.

Now suppose that the rectangles R_1 and R_2 do not overlap. Hence, either they share no x coordinates or they share no y coordinates. Assume by symmetry that R_1 and R_2 share no

x coordinates. Assume again by symmetry that R_1 is to the left of R_2. A *leg* is a horizontal or vertical segment/edge of an L. Re-orient L_1 so that its vertical leg is on the right side, and re-orient L_2 so that its vertical leg is on the left side. Now L_1 and L_2 have distance at most three from each other; the distance may be as much as three because of parity.

We consider merging L_1 with each leg of L_2 one at a time. In other words, we merge L_1 with the nearest leg of L_2 within distance three of L_1, then we merge the result with the other leg of L_2. The second leg can be treated in the same way as the first leg, by induction. Thus there are two cases: either the first leg is the horizontal leg of L_2, or it is the vertical leg of L_2. We first show how the latter case reduces to the former case.

If the vertical leg of L_2 is the first leg, it can have only one coin within the y range of R_1 (by the assumption that extra coins are picked up). We can add this coin separately, as if it were a short horizontal leg of its own L. This leaves a portion of the vertical leg of L_2 outside of the y range of R_2. Thus what remain of R_1 and R_2 do not share any y coordinates, so we can rotate the picture 90 degrees and return to a horizontal problem.

This argument reduces merging two L's to at most three merges between an L and a horizontal leg. Still several cases remain, as illustrated in Figure 28. Case 1 is when the horizontal leg is aligned with a coin in the L. Case 2 is when they are out of alignment. Case 3 is a special case occurring at the corner of the L, where the horizontal legs are aligned but distances are higher than in Case 1. The above three cases are subdivided into subcases (a) and (b), depending on how close the horizontal leg is to the L. Finally, Case 4 is when the L and horizontal leg do not share x or y coordinates.

By the procedures in Figure 28, in all cases, the merging can be done in $O(1)$ flips and rotations of L's, $O(1)$ leapfrogs, and $O(n)$ shuffles. In total, $O(n^2)$ moves are required to merge an L and a horizontal leg, or equivalently to merge two L's.

5.4 Final Sweep

Thus far we have shown how to reversibly re-arrange a configuration (A or B) into the canonical configuration, using two extra coins. However, during this process, we may have picked up extra coins, and now need to drop them appropriately. In reality, these coins sit in arbitrary locations on the board. For re-arranging the source configuration A into the canonical configuration, the moves need not be reversible, so we can simply drop the extra coins in the canonical order, as in Figure 22. For re-arranging the destination configuration B into the canonical configuration, we need to effectively drop these coins by a sequence of reverse moves.

More directly, starting from the canonical configuration, we need to show how to distribute the extra coins to arbitrary locations on the board. We can achieve this effect by making a complete sweep over the board. More precisely, we flip the L as in Section 5.3.1, which has the effect of passing over every position on the board with the operations shown in Figure 23. During this process, we will pass over the extra coins; at this point we treat them as if they were picked up, applying the emulation in Section 4.1. Then we flip the L back to its original orientation. On the way back, whenever we apply an operation in Figure 23 and pass over the desired destination d for one of the extra coins, we move the extra coin to d while there are at least two adjacent coins from the L. By monotonicity of the flipping

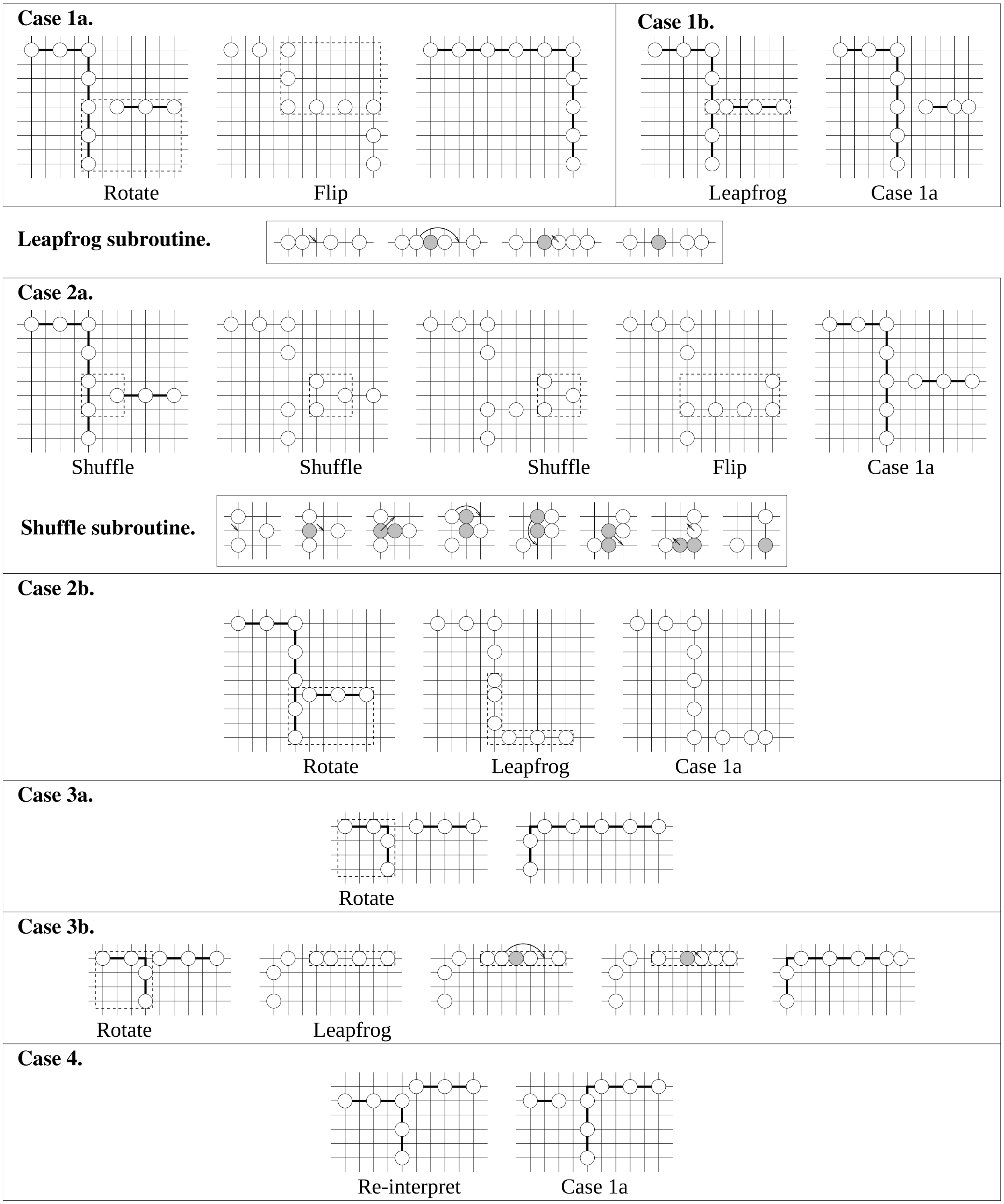

Figure 28: Merging an L and the horizontal leg of a nearby L.

process, this extra coin will not be passed over later by the flip, so once an extra coin is placed in its desired location, it remains there.

5.5 Reducing Span

Now that we know any configuration can be brought to the corresponding canonical configuration with a sequence of (mostly) reversible moves, it follows immediately that any configuration can be re-arranged into any configuration with the same span. More generally, if we are given configurations A and B satisfying $\operatorname{span} A \supseteq \operatorname{span} B$, we can first pick up all coins in $A - \operatorname{span} B$, then reversibly re-arrange both configurations into the same canonical configuration. Putting these two sequence of moves together, we obtain a re-arrangement from $A - \operatorname{span} B$ to B with some coins missing. Then we simply drop the previously picked up coins in the appropriate positions to create B.

Note that these moves need not be reversible, because we are only concerned with the direction from A to B. Indeed, the moves cannot be made reversible, because the span cannot increase (Lemma 2).

This concludes the proof of Theorem 2.

5.6 Lower Bound

The bound on the number of moves in Theorem 2 is in fact tight:

Theorem 3 *The "V to diagonal" puzzle in Figure 29 requires $\Theta(n^3)$ moves to solve.*

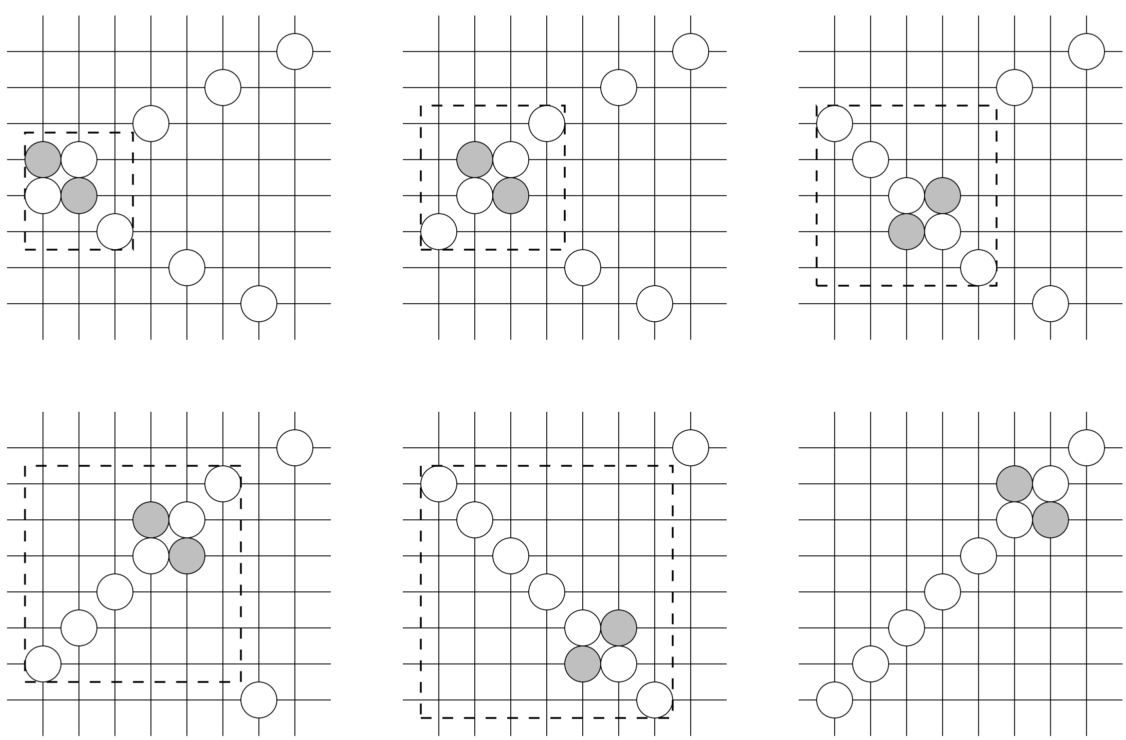

Figure 29: Re-arranging the V-shape in the upper left into the diagonal in the lower right requires repeated rotations of diagonals as in Figure 5 (or repeated rotations of L's).

Proof: We claim that re-arranging the V shape into a diagonal effectively requires repeated "diagonal flipping." At any time, only one component of coins can be actively manipulated (drawn with dotted lines in the figure); all other coins are isolated from movement. Thus we must repeatedly re-arrange the active component so that it can reach the nearest isolated coin. More specifically, we must re-arrange the active component into a chain starting at the corner of the bounding rectangle that is near the isolated coin, and ending at the opposite corner of the bounding rectangle of the active component. These two corners alternate for each isolated coin we pick up, and that is the sense in which we must "flip a diagonal." It is fairly easy to see that each diagonal flipping of a chain with k coins takes $\Omega(k^2)$ time. In total, the puzzle requires $\Theta(\sum_{k=1}^{n} k^2) = \Theta(n^3)$ moves. □

This theorem is the motivation for the puzzle in Figure 7.

5.7 Labeled Coins

We conjecture that Theorem 2 holds even when coins are labeled, subject to a few constraints. The idea is that permutation of the coins is relatively easy once we reach the canonical configuration. Examples of methods for swapping coins within one L are shown in Figure 30. The top figure shows how to swap a pair of coins when the canonical configuration is nothing more than a canonical L. The middle figure shows how to perform the same swap when there are four additional coins. Note that swapping the corner coin 3 works in exactly the same way; indeed, this method works whenever the coins to be swapped have two other coins adjacent to them, and there is another valid destination. The bottom figure shows how to swap one of the end coins, which is more difficult. This last method begins with moving the bend of the L toward the end coin, and then works locally on the coins $1, 2, 3$.

One obvious constraint for these methods is that if there are no valid moves, permutation is impossible. Also, if the bounding rectangle of an L has width or height 1, then the two end coins of the L cannot be moved. Subject to these constraints, Figure 30 proves that the coins in a single connected component of the span, other than the extra coins e_1 and e_2, can be permuted arbitrarily.

It only remains to show that a coin can be swapped with e_1 or e_2, which implies that coins between different connected components of the span can be swapped. We have not proved this in general yet, but one illustrating example is the puzzle in Figure 8, whose solution is shown in Figure 31. The idea is that coins 2 and 4 are e_1 and e_2, and so we succeed in swapping e_2 with coin 3. A slight generalization of this approach may complete a solution to the labeled coins.

5.8 Fewer Extra Coins

We have shown that the configuration space is essentially strongly connected provided there is a pair of extra coins, i.e., the removal of these two coins does not reduce the span. This section summarizes what we know about configurations without this property.

If we have a span-minimal configuration with no extra coins, Lemma 3 tells us that every move decreases the span. With an overhead of a factor of n^2, we can simply try all possible moves, in each case obtaining a configuration with smaller span, which furthermore must have an extra coin (the moved coin). Now we only need to recursively check these configurations.

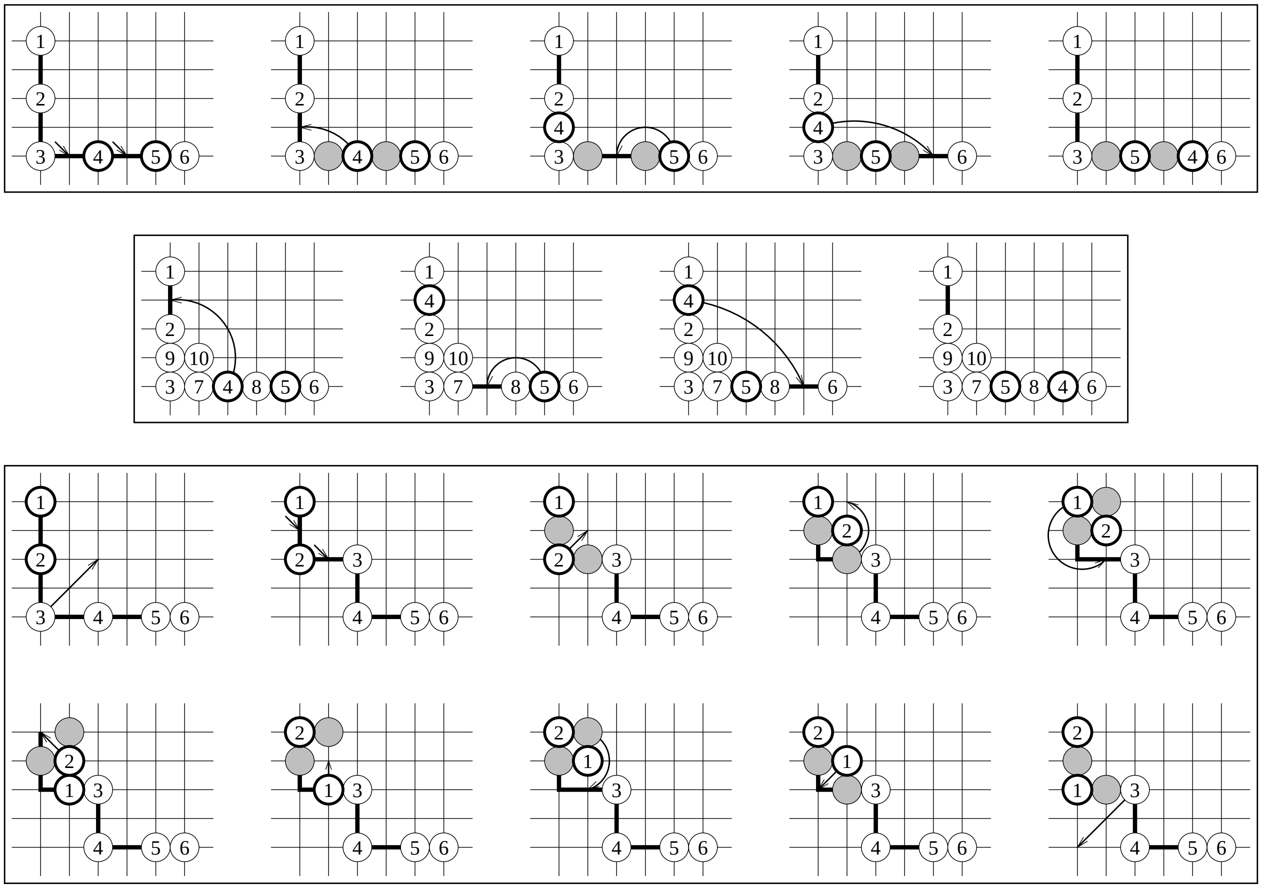

Figure 30: Three cases of swapping coins in the canonical configuration. The coins to be swapped have a thick outline.

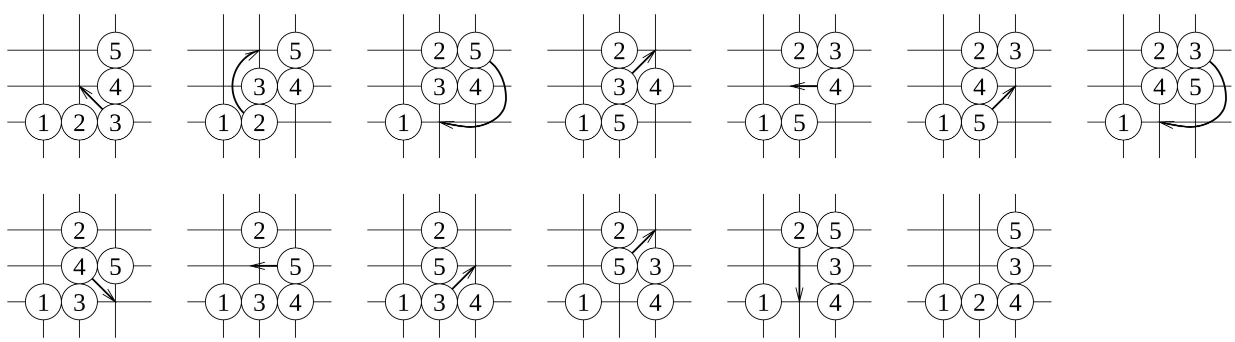

Figure 31: Solution to the puzzle in Figure 8.

Unfortunately, the situation is trickier with one extra coin. The key difficulty is that multiple coins could individually be considered extra, but no pair of coins is extra. In other words, there may be two coins such that removing either one does not reduce the span, but removing both of them reduces the span. Two simple examples are shown in Figure 32.

This difficulty makes "one" extra coin surprisingly powerful. For example, using one extra coin, an L with odd parity can be flipped, although it cannot be rotated, and an L

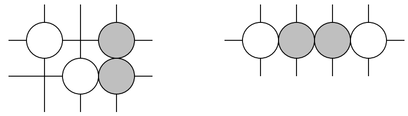

Figure 32: The shaded coins are individually extra, but do not form a pair of extra coins suitable for Theorem 2.

with even parity cannot be flipped or rotated. In Figure 33 we exploit this property to make an interesting solvable puzzle initially with no pair of extra coins; it takes significant work before a pair of extra coins appears.

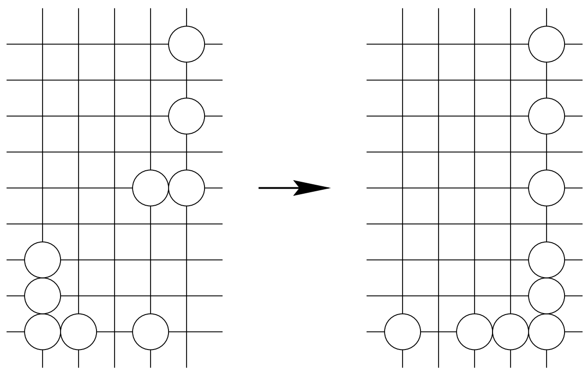

Figure 33: A puzzle on the square grid with no initial pair of extra coins.

6 Conclusion

We have begun the study of deciding solvability of coin-moving puzzles and more generally token-moving puzzles. We gave an exact characterization of solvable puzzles with labeled coins on the equilateral-triangle grid. By introducing the notion of a constant number of extra coins, we have given a tight theorem characterizing solvable puzzles on the square grid. Specifically, we have shown that any configuration can be re-arranged into any configuration with the same or smaller span using two extra coins, and that this is best possible in general. The number of moves is also best possible in the worst case.

Several open questions remain:

1. What is the complexity of solving a puzzle using the fewest moves?

2. How do our results change if moves are forced to be *slides* that avoid other coins? We conjecture that Theorem 1 still holds for unlabeled coins.

3. Can we extend our results on the square grid to the hypercube lattice in any dimension?

4. Can we combine Theorems 1 and 2 to deal with a mix of the square and equilateral-triangle lattice, like the second puzzle in Figure 4?

5. Can we prove similar results for general graphs?

Acknowledgments

We thank J. P. Grossman for writing a program to find optimal solutions to the puzzles in Figures 5 and 6.

References

[1] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. A solitaire-like puzzle and some coin-sliding problems. In *Winning Ways*, volume 2, pages 755–756. Academic Press, London, 1982.

[2] Brian Bolt. Invert the triangle. In *The Amazing Mathematical Amusement Arcade*, amusement 53, page 30. Cambridge University Press, Cambridge, 1984.

[3] Brian Bolt. A two touching transformation. In *Mathematical Cavelcade*, puzzle 20, page 10. Cambridge University Press, Cambridge, 1991.

[4] Maxey Brooke. *Fun for the Money*. Charles Scriber's Sons, New York, 1963. Reprinted as *Coin Games and Puzzles* by Dover Publications, 1973.

[5] Henry Ernest Dudeney. "The four pennies" and "The six pennies". In *536 Puzzles & Curious Problems*, problems 382–383, page 138. Charles Scribner's Sons, New York, 1967.

[6] Kobon Fujimura. "Coin pyramids," "Four pennies," "Six pennies," and "Five coins". In *The Tokyo Puzzles*, puzzles 23 and 25–27, pages 29–33. Charles Scribner's Sons, New York, 1978.

[7] Martin Gardner. Penny puzzles. In *Mathematical Carnival*, chapter 2, pages 12–26. Alfred A. Knopf, New York, 1975.

[8] Boris A. Kordemsky. A ring of disks. In *The Moscow Puzzles*, problem 117, page 47. Charles Scribner's Sons, New York, 1972.

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

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

[11] David Wells. "Six pennies," "OH-HO," and "Inverted triangle". In *The Penguin Book of Curious and Interesting Puzzles*, puzzle 305, 375, and 376, pages 101–102 and 125. Penguin Books, 1992.