PLAYING SIMPLE LOONY DOTS-AND-BOXES ENDGAMES OPTIMALLY

Kevin Buzzard
Department of Mathematics, Imperial College London, London, England
k.buzzard@imperial.ac.uk

Michael Ciere

Received: 6/27/13, Revised: 5/16/14, Accepted: 10/19/14, Published: 12/8/14

Abstract

We explain a highly efficient algorithm for playing the simplest type of Dots-and-Boxes endgame optimally (by which we mean "in such a way so as to maximize the number of boxes taken"). More precisely, our algorithm applies to any endgame made up of long chains of any length and long loops of even length (loops of odd length can show up in generalizations of Dots-and-Boxes but they cannot occur in the original version of the game). The algorithm is sufficiently simple that it can be learned and used in over-the-board games by humans. The types of endgames we solve come up commonly in practice in well-played games on a five by five board and were in fact developed by the authors in order to improve their over-the-board play.

1. Introduction

Dots-and-Boxes is a two-person pencil-and-paper game, where players take it in turns drawing lines on a (typically square) grid of dots. If a player draws the the fourth line of a square (a box), the player wins the box and then makes another move. At the end of the game the person with the most boxes wins. There are a couple of variants of the rules in the literature; we are using the "standard" rules, as set out in Chapter 16 of [3] and [1], which provide a very rich game even on a board as small as five by five (by which we mean twenty-five squares, not twenty-five dots). These are also the rules implemented on various game sites on the Internet such as `www.littlegolem.net`, `www.jijbent.nl` and `www.yourturnmyturn.com`. In brief: a player does not have to complete a box, but if they do complete a box then they must make another move. We follow Berlekamp in distinguishing between the idea of a "move" (the act of drawing one line on the board) and a "turn" (the act of drawing possibly several lines on the board, all but the last of which completes at least one box).

Although perhaps initially counterintuitive, a crucial observation of Berlekamp is that sometimes it is best not to make a box even if the opportunity to do so is available. Recall that a game is said to be played under the Normal Play Rule if "the player who completes the last legal turn of the game wins." Dots-and-Boxes is not played under the Normal Play Rule – at the end, box totals are counted up. However one can introduce the game Nimdots, which is Dots-and-Boxes but played under the Normal Play Rule. It is noted in [3] that often a player will win the Dots-and-Boxes game if and only if the same player wins the Nimdots game. Furthermore it is also not uncommon that the Nimdots optimal line of play (which is trivial to compute in a given loony endgame position) will win the Dots-and-Boxes game as well. This advice should nowadays be taken with a pinch of salt in expert-level play, but is certainly often true in endgame positions typically reached in games between amateurs.

As is explained in both [3] and [1], there is a generalization of Dots-and-Boxes called Strings-and-Coins, played on what is basically an arbitrary finite graph (the dictionary being that a box corresponds to a vertex and an undrawn line to an edge), where players take turns in cutting edges, and they claim isolated vertices. The analogous generalization of Nimdots is called Nimstring, and in this more general setting one can of course see positions containing, for example, isolated loops of any length (including odd lengths, or lengths less than 4; this cannot happen in a Dots-and-Boxes game). We will occasionally mention positions containing loops of odd length, but our main results on endgames will be obtained under the assumption that all loops have even length.

A lot has been written about Nimdots and Nimstring, which are far more amenable to analysis than Dots-and-Boxes. Nimdots is an impartial game which yields well to Sprague–Grundy (Nim) theory. We do not go into this theory here (see the references cited above for a very thorough analysis) because the emphasis of this paper is not on winning Nimdots but on winning actual Dots-and-Boxes games, especially if they are close. The sort of position that the authors are interested in is something like Figure 1. This is not a pathological position – this sort of position can easily occur as the endgame of a well-played game of of Dots-and-Boxes. Here thirty moves have been played. The assiduous reader of the references above will know that player one has achieved his Nimdots aim: the board has two long chains (we refer the reader to the references above for all basic definitions such as chains and loops) and hence player 1 will win the Nimdots game from this position. However player 2 has been smart and made some loops, it is possible and relatively easy to prove that despite winning the Nimdots game, player 1 will actually lose the Dots-and-Boxes game 13–12. This phenomenon is well-known to the authors of [3], who observe on page 569 that "experts will need to know something about the rare occasions

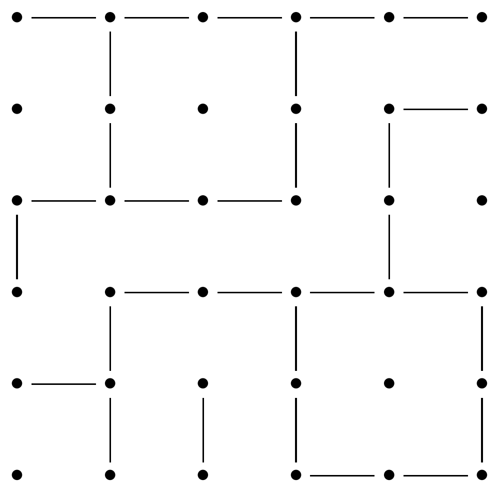

Figure 1: A typical Dots-and-Boxes endgame

when the Nimstring theory does not give the correct Dots-and-Boxes winner" and on page 577 that "Your best chances at Dots-and-Boxes are likely to be found by the Nimstring strategy." These words have not aged well however. With the advent of games sites on the Internet such as `jijbent.nl` and `littlegolem.net` over the last few years, the number of very strong Dots-and-Boxes players has skyrocketed, and one of the first things that one discovers when playing games against stronger players on these sites is that a Nimdots player, even one who has assiduously read the Dots-and-Boxes chapter of [3], stands very little chance against someone who can count a Dots-and-Boxes endgame correctly. This paper explains an optimal way of playing the simplest possible Dots-and-Boxes endgames, composed only of isolated loops and chains; even this simple situation is more subtle than one might imagine. In fact there seems to be very little in the literature about counting boxes, and more generally about the differences between Nimdots and Dots-and-Boxes. After some preliminary definitions we will list everything that we are aware of. Chapters 10 and 11 of [1] provide a good introduction to this material.

The *value* of a game (played between X and Y, with X to play) is the number of future boxes that Y will get minus the number of future boxes X will get, under best Dots-and-Boxes play. For example, the value of an n-chain for $n \geq 1$ is n, because X will have to open it and then Y will take all of it. Note that this is a very different notion to the nim-value of the game, a notion that we will not consider at all in this paper (the nim-value of any chain of length $n \geq 3$ is zero). Values are considered in Chapters 10 and 11 of [1], and also on pages 574–575 of [3]; this latter reference states a formula for the value of a game comprising only of long chains (without proof; the proof, which is not hard, is given in Theorem 10 here). Next, there is the paper [2], which makes a beautiful analysis of some very topologically complicated endgames that can occur on a very large board. Finally there is Scott's MSc thesis [4], available in the UC Berkeley library, which makes some more interesting observations about values of certain loony Dots-and-Boxes endgames.

The contribution of this paper is to give a practical algorithm useful for small-board play in a topologically simple endgame such as Figure 1. In a sense our work is closest to pages 574–575 of [3], which we take much further because we completely deal with the situation where the endgame contains only chains and loops. The presence of loops complicates matters immensely, but is crucial if one wants a practical algorithm because in high-level play on a 5×5 board, the player who realizes that he will probably be losing the Nimdots game (i.e., the chain battle) will quickly turn their attention to rigging the situation to winning the Dots-and-Boxes game regardless, typically by making loops. Our philosophy when writing the paper was to assume that the reader knows the theory as developed in [1] (which we only skim through), but then to give detailed proofs of the results we need, including results which are stated without proof in [1] and [3].

2. An Overview of the Problem Considered in This Paper

This paper considers Dots-and-Boxes positions that have reached the "loony endgame" stage – that is, in which every available move on the board not only gives away a box, but is a loony move in the sense of [3]. Recall that a loony move is (informally) a move that a player makes which gives their opponent the option to decide who makes the last move of the game. An example of a loony endgame would be a position like Figure 2. Here the only available moves are in chains of length at least 3 (a chain of length at least 3 is called a "long chain"), or loops (any loop on a Dots-and-Boxes board has length at least 4 and is hence a "long loop"). Any move that a player makes in a long chain gives their opponent the opportunity to play the "all-but-two trick" if they so desire, where they take all but two of the boxes offered and then leave the last two by playing a so-called "double-cross" move, as indicated in Figures 3 and 4.

Similarly there is an "all but four" trick that applies if a player opens a loop: their opponent can take all of the loop and play first in the remainder of the game, or (as in Figure 16 on page 554 of [3]) they can leave four boxes and force the opener to play first in the remainder of the game.

Our main results apply to what we call "simple loony endgames", which are endgames where every connected component of the game is either an isolated loop or an isolated long chain. We also assume that every loop is of even length (this is automatic for a Dots-and-Boxes position, but may not hold for a general Strings-

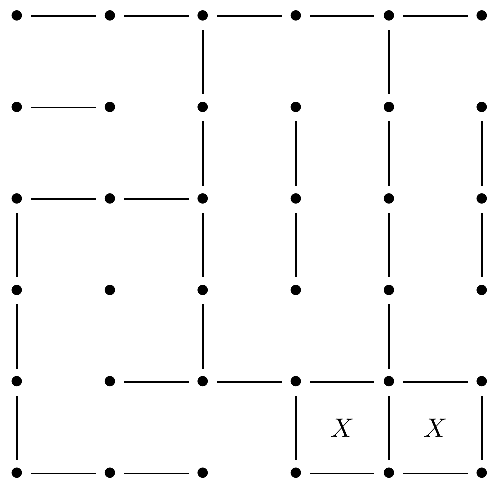

Figure 2: A loony endgame

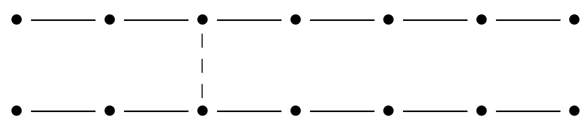

Figure 3: Player X plays the vertical move opening a chain...

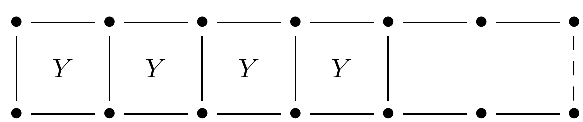

Figure 4: Instead of taking all six boxes, player Y takes four boxes but leaves the last two for X, forcing him to make another loony move elsewhere after taking his two free boxes.

and-Coins position[1]). Such positions arise commonly in high-level play. Figure 5 is an example which we will work through carefully to highlight some issues. It is a game between player X and player Y, and Y started. One checks that 21 moves have been played, and no boxes have been taken, so it is X's turn. An expert will see instantly that X has lost, but we want to talk about maximizing the number of boxes they get.

[1] There is currently, as far as we know, still no known method for *efficiently* analyzing a general loony Strings-and-Coins position consisting of only isolated long loops and chains; the only method we know is to do a brute force search down the game tree. The point of this paper is to show how one can get away with much less if all loops have even length.

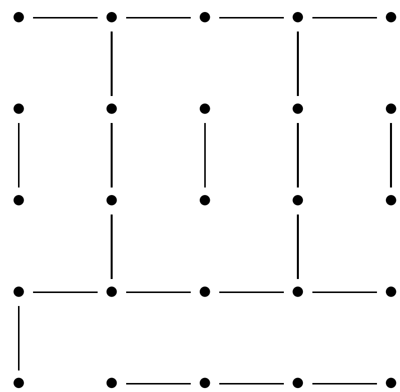

Figure 5: A simple loony endgame

Our notation for such positions is as follows: we denote by n a chain of length n, and by n_ℓ a loop of length n, so we would call this position $3+3+4+6_\ell$, or $2\cdot 3+4+6_\ell$. If this were a game of Nimdots, then X would open one of the loony regions (a 3-chain, the 4-chain or the 6-loop) and Y would keep control (that is, if X opens a chain then Y would play the all-but-two trick, and if X opens the loop then Y would play the all-but-four trick). In either case X gets some boxes, but then has to open the next loony region. Player Y may continue in this way and ensure that they have the last complete legal turn of the game. Hence Y can win the Nimdots game, but of course this line results in a 10-6 win for X in the Dots-and-Boxes game (X gets two boxes for each chain and four for the loop). Of course Y has far better options. We shall see in this paper that one of the correct ways to play this position (there are several) is as follows:

1. X opens a 3-chain and Y takes all three boxes.
2. Y opens a 3-chain and X declines the offer, taking one box and giving Y the other two.
3. Y opens the 6-loop and X takes all six boxes.
4. X opens the 4-chain and Y takes all four boxes.

The final score is 10-8 to Y.

What is going through each of the players' heads during this exchange? At each stage of this process above, *both* players had a question to answer. The person not in control (the one opening the loony regions) had to play a loony move and so they had to decide which component of the game to open. And then the person in control had to decide whether to keep control (sacrificing some boxes in the process), or whether to grab everything on offer and lose control. *How are these decisions made optimally?*

By a "simple loony endgame" we formally mean a loony endgame each of whose components are long chains or long loops of even length – such as the position in Figure 5. In this paper we explain two algorithms. One of them, when given a simple loony endgame, quickly tells the player whose turn it is (that is, the player not in control), which loony region should be opened first. The other, when given a simple loony endgame, quickly returns its value; given this second algorithm it is then a simple matter (see Section 5) for the player in control to decide whether to remain in control or not. Hence if both players have access to both algorithms, they can play the endgame optimally. By "quickly" here we mean either "without traversing the entire game tree," or we could mean "quickly enough to be of practical use in an over-the-board game." We could even give a more formal definition of what we have here; if we are given the game as an input in a certain format, then our algorithms will terminate in time $O(1)$ *independent of the number of components on the board*; in particular we are beating the "traversing the game tree" approach hands down. To give an example of what we are doing in this paper, imagine a position consisting of a hundred 3-chains and a hundred 4-loops. The naive method for computing whether to open a 3-chain or a 4-loop would involve recursively computing the values of a position consisting of a 3-chains and b 4-loops for all $0 \leq a, b \leq 100$. However (once one has understood the notions of value and controlled value, which are about to be explained) Theorem 29 of our paper implies immediately that the player whose turn it is should open a 3-chain, and Corollary 30 implies immediately that their opponent should take all three boxes. Essentially no calculations at all are required, and certainly no iteration over all subgames.

The algorithms are given in Section 4, and the reader only interested in playing Dots-and-Boxes endgames optimally can just commit these algorithms to memory (or, even better, learn the few underlying principles behind them), and ignore the proofs that the algorithms are correct.

3. Notation and Basic Examples

We will typically let G denote a Dots-and-Boxes game (for example G could be "the opening position in a 5×5 game" or "an endgame with five 5-chains." A *loony endgame* is a game G where every available move is a loony move. A *simple loony endgame* is a game comprised only of disjoint long loops of even length and chains (for example the game in figure 5, but not the game in figure 2, as this has a chain running into a loop).

We say that the *value* $v(G)$ of the game G is the net score for *the player whose turn has just finished* if the game is played optimally (in the sense that both players are trying to maximize the number of boxes they obtain). We ignore any boxes taken prior to reaching the position G; for example the value of the empty game is zero (whatever the scores of the players are at this point). It is natural to look at the net gain of the second player rather than the first, for this convention means that the value of a loony endgame is always non-negative: if a player makes a loony move, then it is clear that their opponent will not make an overall net loss if they play optimally, because the sum of the values of their two options is non-negative and hence they cannot both be negative.

We can easily extend this notion of value to define the *value of a move*: we write $v(G; m)$ to denote the net score for the player whose turn has just finished if the move m is played by his opponent, and then the resulting game is played optimally by both players. If Z is an isolated long loop or chain in G then we denote by $v(G; Z)$ the value $v(G; m)$ for m any move in Z (all such moves are loony moves and have the same value).

Sometimes it is convenient to refer to a game as the collection of several disjoint components. We may for example write $G = H + K$, where we mean that G is the game consisting of the disjoint union of games H and K. One case where this is useful is when we want to evaluate how the value of a game changes when we slightly change one component. Say, for example, we are interested in the game $G + n$, which consists of G and a long chain of length n. Its value $v(G + n)$ is then a function of n.

A notion that was introduced by Berlekamp and has proven to be very useful is the *controlled value* of a game. In fact we use this notion only for simple loony endgames. For G a simple loony endgame, we first define the *fully controlled value* $fcv(G)$ to be the net score for the player in control, if he keeps control for the entire game, even to the extent of giving his opponent the last few boxes in the connected region that is opened last. The controller hence loses two boxes in each chain and four boxes in each loop. In particular if G consists of long (by which we mean "of length at least 3) chains of lengths $c_1, c_2, \ldots$, and long (by which we mean "of length at least 4) loops of even lengths $\ell_1, \ell_2, \ldots$, then $fcv(G) = \sum_i (c_i - 4) + \sum_j (\ell_j - 8)$.

The *controlled value* $cv(G)$ of a simple loony endgame G is equal to the sum $fcv(G) + tb(G)$, where $tb(G)$, the *terminal bonus* of G, is an integer calculated thus. The empty game has terminal bonus zero. If G is non-empty and has a chain of length at least 4, or no loops, then $tb(G) = 4$. If G has loops but no chains, then $tb(G) = 8$. The remaining case is when G has chains and loops, but all chains are 3-chains; then we set $tb(G) = 6$. The motivation behind this definition is that the controlled value of a position is the net gain for the player in control, assuming he remains in control until the last (or occasionally the last-but-one) turn of the game (we shall clarify this comment later). Note in particular that whilst $v(G)$ is a subtle invariant of G, computing $cv(G)$ is trivial for any simple loony endgame. It is easy to check that $v(G) \equiv cv(G) \bmod 2$ (as both are congruent mod 2 to the total number of boxes in G). After we have developed a little theory we prove (Lemma 15(a)) the assertion on page 86 of [1], namely that if G is a simple loony endgame and $cv(G) \geq 2$, then $v(G) = cv(G)$.

For convenience, when talking about loony endgames we will refer to the player in control as the *controller* and the other player as the *defender* from now on.

4. The Algorithms

We briefly describe the two algorithms whose correctness we prove in this paper. Both algorithms take as input a simple loony endgame. The first algorithm computes its value (and is hence the one the controller needs, to compute what to do after the defender opens a loony region in a simple loony endgame). The second returns an optimal move (and is hence the one used by the defender).

4.1. The Algorithm to Compute Values

Let G be a simple loony endgame.

- Is G empty? If so, the value is zero. If not, continue.
- Is $cv(G) \geq 2$? If so, $v(G) = cv(G)$. If not, continue.
- Are there any loops in G? If so, continue. If not, use Theorem 10 to determine $v(G)$.
- Are there any 3-chains in G? If so, continue. If not, use Corollary 22 to determine $v(G)$.
- If G has one 3-chain and every other component of G is a loop, use Corollary 24 to determine $v(G)$. If not, continue.
- Does G have just one 3-chain? If so, use Corollary 27 to compute $v(G)$, and if not, use Corollary 30.

4.2. The Algorithm Giving Optimal Moves

Here we assume G is non-empty. There will always be an optimal move in G which is either opening the smallest loop or the smallest chain (this is Corollary 3(a)).

- If G has no chains, open the smallest loop. If G has no loops, open the smallest chain. If G has both loops and chains, continue.
- If G has no 3-chains, open the smallest loop (Theorem 18). Otherwise continue.

- If G has one 3-chain but every other component of G is a loop, use Theorem 23 to determine an optimal move. Otherwise continue.
- If G has one 3-chain, use Theorem 25 to determine an optimal move. Otherwise use Theorem 29.

In fact applying the algorithms is relatively simple, but perhaps not quite as simple as we have made things appear. Let us say for example that we are trying to compute the value of a game G with loops and chains, but no 3-chains. Assume $cv(G) < 2$. Write $G = K + L$, where L is all the 4-loops and K is all the rest. The algorithm above says that one should use Corollary 22 to compute the value of G, however looking at that corollary one sees that to compute $v(G)$ one needs to compute $v(K)$, and to compute $v(K)$ we need to apply the algorithm again! However, one checks relatively easily that the number of times one needs to "start again" is at most 3 for any position, and that this can only happen in some cases where G has precisely one 3-chain and at least one 4-loop. As we indicate in the paper, any time one needs to start again, the result in the paper that one needs to compute the value of the subgame comes strictly before the location of the order to start again, so there is no way to get into an infinite loop.

To clarify, here is a worked example of a worst-case scenario. Let G be the game $3 + 4 + 100 \cdot 4_\ell + 100 \cdot 6_\ell$ (a 3-chain, a 4-chain, a hundred 4-loops and a hundred 6-loops); let us compute its value. Its controlled value is negative and odd. Our first run through the value algorithm tells us to use Corollary 27, where we see that to compute $v(G)$ we need to know $v(G\backslash 3)$. Starting the value algorithm again with $G\backslash 3$ (or just reading the remark after Corollary 27) we are led to Corollary 22(d), where we find that we need to compute $v(G\backslash 3\backslash 100 \cdot 4_\ell) = v(4 + 100 \cdot 6_\ell)$; restarting for the second time (or just reading the comments in Corollary 22(d)) we find our way to Corollary 22(c), which tells us that $v(4 + 100 \cdot 6_\ell) = 4$. Hence $v(G\backslash 3) = 4$. This means (again by Corollary 27(e)) that we need to compute $v(G\backslash 100 \cdot 4_\ell)$, which by part (d) is equal to 3 (note that we have already computed $v(G\backslash 100 \cdot 4_\ell\backslash 3)$), and hence $v(G) = 3$ and we are finished. This looks complicated but it is the worst-case scenario, and often the calculations are much easier: the bottom line is that computing the values of just a few sub-positions has given us $v(G)$. An expert could have said immediately that the value was either 1 or 3, but if a player leaves their opponent the position $G + 3$ and their opponent opens a 3-chain, the player needs to know which possibility is the right one in order to decide whether to double-cross or not.

5. When to Lose Control

We briefly recall how the controller uses the algorithm which computes the values of simple loony endgames, to answer the question of whether to stay in control or not. This argument is standard – see for example page 82 of [1].

Consider a position $G + n$, that is, a game G plus an n-chain, with $n \geq 3$. Let the players of this game be X and Y, and let it be X's move. Say X opens the n-chain. Then Y takes all but two of this chain (giving him a net score so far of $n - 2$) and then has to decide whether to take the last two boxes and then to move first in G, or whether to give the two boxes to X and then move second in G. Recall that $v(G)$ is the net score for the second player if the game is played optimally. If Y keeps control (and the game is then played optimally) then Y's net gain will be $n-2+(v(G)-2)$; if Y relinquishes control, Y's gain will be $n-2+(2-v(G))$. The larger of these two numbers is $n - 2 + |v(G) - 2|$, so the critical question is whether $v(G) \leq 2$ or $v(G) \geq 2$. If $v(G) > 2$, Y should keep control. If $v(G) < 2$, Y should relinquish control, and if $v(G) = 2$ then it does not matter. We have proven

$$v(G + n; n) = n - 2 + |v(G) - 2|$$

and may also deduce that if Y knows the value of G then they know whether to keep control or not – that is, they know how to play the n-chain optimally after X opens it.

Similarly, if n_ℓ denotes a loop of length $n \geq 4$ then

$$v(G + n_\ell; n_\ell) = n - 4 + |v(G) - 4|,$$

and faced with the position $G + n_\ell$, with $n \geq 4$, if X opens the n-loop then Y keeps control if $v(G) > 4$, loses control if $v(G) < 4$, and it does not matter what they do if $v(G) = 4$.

In particular, given our two algorithms (one saying which loony move to play in a simple loony endgame, the other computing the value of a simple loony endgame), we know how to play the endgame perfectly.

Before we start on the proof of the correctness of our algorithms, we develop some general theory. We start with a recap of the "man in the middle" proof technique from Chapter 10 of [1].

6. The Man in the Middle

The "man in the middle" is a technique explained on page 78 of [1]. In its simplest form it says that if the man in the middle (call him M) is playing two games of Dots-and-Boxes against two experts, and the starting positions are the same, but M starts one game and does not start the other, then M can guarantee a net score of zero by simply copying moves from one game into the other – the experts are then in practice playing against each other. The crucial extension of this idea is not to play the same game against the two experts, but to play very closely-related games G and H, typically identical apart from in one simple region, and in this setting M basically "copies as best they can," where this has to be made precise when one of his opponents plays in a region of one game which does not have an exact counterpart in the other. The outcome of such an argument will be a result of the form $v(G) - v(H) \leq n$, or sometimes $|v(G) - v(H)| \leq n$, when G and H are sufficiently similar. Note that we always assume our experts are playing optimally, and so, for example, if M opens a long chain against an expert then we may assume that either they take all of the chain, or play the all-but-two trick (other options, such as leaving three or more boxes, are dominated by one of the two options above).

We will now spell out one example in detail; the other results in this section are proved using the same techniques.

Example 1. If J is an arbitrary game of Dots-and-Boxes, then $|v(J+3)-v(J+4)| = 1$. To prove this, set $G = J+3$ and $H = J+4$, and then let the man in the middle, M, play both games, one as player 1 and the other as player 2. Each time an opponent plays in one of the J's, M copies in the other J. We now have to give a careful explanation as to what algorithm M will follow when the play moves out of J. When this happens, either an opponent has opened the 3-chain (in which case M opens the 4-chain in the other game), or an opponent has opened the 4-chain (in which case M opens the 3-chain). M now waits to see whether his opponent takes all, or leaves two, and then copies the choice in the other game. Note that at this point the games he's playing become the same, but M will either be a box down or a box up, depending on the choices that his opponents made. We conclude that his net loss is no more than one box, and hence $|v(J+3) - v(J+4)| \leq 1$. Now a parity argument shows that in fact $|v(J+3) - v(J+4)| = 1$.

Note finally that this argument does *not* tell us whether $v(J+3) > v(J+4)$ or $v(J+4) > v(J+3)$: both can happen, and distinguishing between the two possibilities seems in general to be a very subtle issue.

An easy generalization of the above argument gives the following lemma.

Lemma 2. *Let G be a Dots-and-Boxes game.*
(a) If $1 \leq m \leq n \leq 2$ or $3 \leq m \leq n$ then $|v(G+m) - v(G+n)| \leq n - m$.
(b) If $4 \leq m \leq n$ then $|v(G+m_\ell) - v(G+n_\ell)| \leq n - m$.

Proof. For (a), the man in the middle mimics his opponents: the inequalities say that either both chains are long or both are short, so this is possible. In (b), the man in the middle again mimics his opponents. $\square$

Corollary 3. *Let G be a Dots-and-Boxes game.*
(a) If $3 \leq m < n$ then in the game $G + m + n$, opening the n-chain is never strictly better than opening the m-chain, and similarly if $4 \leq m < n$ then opening the n-loop in $G + m_\ell + n_\ell$ is never strictly better than opening the m-loop.
(b) In a loony endgame comprising entirely of long isolated chains and long isolated loops, there will be an optimal play of the game in which all the chains are opened in order (smallest first), as are all the loops.

Proof. To prove part (a), Lemma 2(a) says $m - n \leq v(G+m) - v(G+n) \leq n - m$, and hence $n + v(G+m) \geq m + v(G+n)$ and $n - v(G+m) \geq m - v(G+n)$. Hence $v(G + m + n; n) = \max\{n - 4 + v(G + m), n - v(G + m)\} \geq \max\{m - 4 + v(G + n), m - v(G+n)\} = v(G+m+n; m)$. Hence a player opening the n-chain is always at least as good for their opponent as opening the m-chain. The same argument works for loops. Note that part (b) is immediate from part (a). □

As a consequence, we deduce that in a simple loony endgame, either opening the smallest loop or the smallest chain is optimal! This looks like a major simplification, but in fact distinguishing which of the two possibilities is the optimal one is precisely the heart of the matter.

Note also that as a consequence of this corollary, we may modify the rules of Dots-and-Boxes by making it *illegal* to open an n-chain if there is an m-chain with $n > m \geq 3$; extra assumptions like this simplify man in the middle arguments without changing the values of any games, as we have just seen.

7. Amalgamating Loops and Chains

Imagine player P is winning a big chain battle in a Dots-and-Boxes game. In amongst the game, player P can see two 5-chains, and they mentally note that these will be worth one point each at the end (as P will take three boxes and lose two, for each of them). If someone were to offer to remove those two 5-chains and replace them with a 6-chain (but without changing whose move it was, so P was still winning the chain battle), then perhaps P would say that they didn't mind much, because the two 5-chains are going to be worth two boxes in total, which is what the 6-chain will bring in. More generally, P might be happy to swap an a-chain and a b-chain for an $(a + b - 4)$-chain, at least if $a, b \geq 4$. Proposition 4 below proves that in fact if $a, b \geq 4$ then an a-chain and a b-chain are *equivalent* to an $(a + b - 4)$-chain in huge generality.

Proposition 4. *Let G be an arbitrary Dots-and-Boxes game, and say $a, b \geq 4$ and $c = a + b - 4$. Then $v(G + a + b) = v(G + c)$.*

Proof. We use the man in the middle technique. The man in the middle, M, plays two experts, playing the game $G+a+b$ with one and $G+c$ with the other, and starting in precisely one of these games. We assume the experts play optimally. If we can show that M always at least breaks even on average (that is, his net gain on one board is at least his net loss on the other, always), regardless of which of the games he starts, then we have proved the result. The work is in deciding how to play when one of M's opponents plays a move not in G and hence that cannot immediately be mirrored. The easiest case is when an opponent opens the a-chain or the b-chain (WLOG the a-chain). M then doubledeals them (that is, he keeps control), making a net gain of $a-4$ and leaving the games $G+b$ and $G+c$, whose values are known by Proposition 2(b) to differ by at most $c-b=a-4$, so we are finished in this case.

The fun starts when one of the experts opens the c-chain. This move lets M take $c-2$ boxes and then M has a decision to make – whether to keep control or not. If one of these decisions scores v (not counting the $c-2$ boxes), then the other scores $-v$, so $v(G+c;c)$ is clearly at least $c-2$. But we are assuming the experts are playing optimally, so we may deduce $v(G+c) \geq c-2$. Again by Proposition 2(c) we have $|v(G+c)-v(G+b)| \leq c-b$ (note that $c=b+(a-4) \geq b$) and hence $v(G+b) \geq b-2 \geq 2$. Hence if M opens the a-chain in $G+a+b$, then we may assume that his opponent keeps control. Now M has just made a loss of $a-4$ but it is his move in $v(G+b)$ and he now opens the b-chain. When his opponent takes the first $b-2$ boxes of this chain, we see that M has in total lost $a+b-6$ boxes in this game, and his opponent has to decide whether or not to keep control in a game whose other component is G. On the other hand M took $c-2=a+b-6$ boxes in the $G+c$ game, and M has to decide himself whether or not to keep control in this game, so the positions have become identical and M has a net score of zero, and the proof is now complete. $\square$

An easy induction on n gives the following result:

Corollary 5. *If G is a game, if $n \geq 1$, if $c_i \geq 4$ for $1 \leq i \leq n$, and if $c = 4+\sum_i(c_i-4)$, then $v(G+c)=v(G+c_1+c_2+\ldots+c_n)$.*

Now say m is a move in the game G, say c and the c_i are as in the previous corollary, and let $H_1=G+c$ and $H_2=G+c_1+c_2+\ldots+c_n$. We can also regard m as a move in H_1 and H_2.

Lemma 6. *The move m is optimal in H_1 if and only if it is optimal in H_2.*

Proof. Capturing free isolated boxes is always optimal, so the result is obvious if m is the capture of an isolated free box. Indeed, we may assume that G has no free boxes on offer.

Next say m is a non-loony move in G. Then playing m has a cost s (the number of free boxes one's opponent can take after m) and, after playing m in G and then removing these boxes, we are left with a game G'. Set $H_1' = G' + c$ and $H_2' = G' + c_1 + c_2 + \ldots$. Then m is optimal in H_1 if and only if $v(H_1) = s - v(H_1')$, and similarly for H_2. However $v(H_1) = v(H_2)$ and $v(H_1') = v(H_2')$ by the preceding corollary, and the result follows.

Finally, assume m is a loony move, resulting in a handout of s boxes and a potential doubledeal involving $d \in \{2, 4\}$ boxes (depending on whether m was in a chain or a loop). Let G' denote the game obtained from G after all the available boxes are taken, and let H_1' and H_2' be as before. Then m is optimal in H_1 if and only if $v(H_1) = s - |d - v(H_1')|$, and again by the previous corollary this is if and only if $v(H_2) = s - |d - v(H_2')|$, which is if and only if m is optimal in H_2. □

Although less useful in practice on a small board, arguments like the above can also be done for loops just as easily (with only trivial modifications to the proofs). For example:

Proposition 7. *If $m, n \geq 8$ and $r = m + n - 8$ then $v(G + m_\ell + n_\ell) = v(G + r_\ell)$.*

Corollary 8. *If G is a game, if $n \geq 1$, if $p_j \geq 8$ for $1 \leq j \leq n$ and if $p = 8 + \sum_j (p_j - 8)$, then $v(G + p_\ell) = v(G + (p_1)_\ell + \cdots + (p_n)_\ell)$.*

Lemma 9. *If m is a move in G, then m is optimal in $G + p_\ell$ if and only if it is optimal in $G + (p_1)_\ell + \cdots + (p_n)_\ell$.*

8. The Easiest Examples: All Loops or All Chains

We finally begin the proof of the correctness of our algorithms. In this section we deal with simple loony endgames consisting either entirely of long chains, or entirely of long loops (of even length). In this case, Corollary 3 tells us that an optimal move is opening the component with smallest size, and our task is hence to compute the value of such a game. We start with the case where G is made up entirely of chains; part (b) of the theorem below is stated without proof in the section "when is it best to lose control" in Chapter 16 of [3], and in a sense our paper starts where they leave off.

Theorem 10. *Let G be a simple loony endgame.*
(a) If $G = c_1 + c_2 + \ldots + c_n$ consists of $n \geq 1$ disjoint chains of lengths c_1, $c_2, \ldots, c_n \geq 4$, then $v(G) = cv(G) = 4 + \sum_{i=1}^{n}(c_i - 4) \geq 4$.
(b) If a simple loony endgame $G = c_1 + c_2 + \ldots + c_n$ is composed entirely of $n \geq 1$ long chains, and if $c = cv(G) = 4 + \sum_i (c_i - 4)$ is the controlled value of G, then $v(G) = c$ if $c \geq 1$, and $v(G) \in \{1, 2\}$ with $v(G) \equiv c \bmod 2$ if $c \leq 0$.

Remark 11. This theorem determines $v(G)$ uniquely for G a simple loony endgame with no loops, and furthermore (assuming we know the controlled value of the game) it does it without having to embark on a recursive procedure, computing values of various subgames. Removing the need to "go down the game tree" in this way is precisely the point of this paper.

Proof. The proof of part (a) is an easy induction on n. For part (b), induct on the number of 3-chains, noting that if $G = 3 + H$ and $c' = cv(H)$ is the analogue of the number c for the game H, then $c' = c + 1$. If $c \geq 1$ then $c' \geq 2$, so $v(H) = c' \geq 2$ by the inductive hypothesis and the controller should stay in control giving $v(G) = v(H) - 1 = c$. On the other hand if $c \leq 0$ then $c' \leq 1$ so by the inductive hypothesis $v(H) \in \{1, 2\} \leq 2$ and the controller should lose control, giving $v(G) = 3 - v(H) \in \{1, 2\}$. □

We next deal with the case of simple loony endgames which are all loops. This case is a little more complicated, because the 3-chain was the only chain that needed to be dealt with separately above, whereas we must deal with both 4-loops and 6-loops here as special cases. The following theorem is the first time we need our assumption that all loops are of even length (which will be true on a standard square Dots-and-Boxes grid but may not be true in more generalized versions of the game – without this assumption then we would have to deal with 5-loops and 7-loops as well).

Say G is a disjoint union of long loops of even lengths $\ell_1, \ell_2, \ell_3, \ldots, \ell_n$ with $n \geq 1$. Set $c = cv(G) = 8 + \sum_i (\ell_i - 8)$ (and note that c is even). Let f be the number of 4-loops in G and let s be the number of 6-loops.

Theorem 12. *With notation as above, the following hold:*
(a) if $\ell_i \geq 8$ for all i then $v(G) = c$;
(b) if $c \geq 2$ or G is empty then $v(G) = c$;
(c) if $c \leq 0$ and $f = 0$ and G is non-empty then $v(G) \in \{2, 4\}$ and $v(G) \equiv c \bmod 4$ (this congruence determines $v(G)$ uniquely);
(d) if $c \leq 0$ and $f \geq 1$ then let K denote G minus all the 4-loops (whose value is computable using (a)–(c) above). If $v(K) \equiv 2 \bmod 4$ then $v(G) = 2$. Otherwise $v(G) \in \{0, 4\}$ and $v(G) \equiv v(K) + 4f \bmod 8$.

Proof. Part (a) is an easy induction on number of loops, as in the previous theorem. Part (b) follows by induction on $f + s$. The base case is (a), and the inductive step proceeds as follows: if the length of the smallest loop is t and $G = H + t_\ell$ then we know that opening t is optimal, and hence $v(G) = t - 4 + |v(H) - 4|$. But $cv(H) = c - (t - 8) \geq 4$ so by the inductive hypothesis $v(H) = cv(H) \geq 4$ and hence $v(G) = t - 8 + cv(H) = c$.

Part (c) is proved by induction on s. The case $s = 0$ cannot occur, for then G is a disjoint union of loops of length ≥ 8, so its controlled value c must be at least 8. The game cannot be one isolated 6-loop either, because we are assuming $c \leq 0$. Hence we can write $G = 6_\ell + H$ with $cv(H) = c + 2 \leq 2$. Part (b) (if $cv(H) = 2$) and the inductive hypothesis (if not) implies $v(H) \in \{2,4\} \leq 4$ and $v(H) \equiv c+2 \bmod 4$. Hence $v(G) = v(G; 6_\ell) = 2 + |v(H) - 4| = 6 - v(H)$ is congruent to $4 - c$ and hence to $c \bmod 4$ (as c is even).

Part (d) is similar – an induction on f. Write $G = H + 4_\ell$. We have $cv(H) = 4 + c \leq 4$ so by (b) and the inductive hypothesis we have $v(H) \in \{0, 2, 4\}$ and hence $v(G) = |4 - v(H)| = 4 - v(H)$. A case-by-case check now does the job; the hard work is really in formulating the statement rather than checking its proof. □

9. A General Simple Loony Endgame: Values and Controlled Values

Recall that the value of a simple loony endgame is a number which we do not (at this stage in the argument) know how to compute efficiently. However the controlled value of such a game is very easy to compute. We are lucky then in that the value and controlled value of a simple loony endgame are by no means completely independent (for example they are congruent mod 2, because both are congruent mod 2 to the total number of boxes in the game). We give some more subtle relations between these numbers here, which turn out to be crucial; Lemma 15 and Corollary 17 will be used again and again throughout the rest of the paper.

Lemma 13. *Let G be a simple loony endgame. Then $v(G) \geq cv(G)$.*

Remark 14. This is stated without proof on page 84 of [1]; we give a proof here for completeness.

Proof. We exhibit a strategy for the controller which, against best play, gives the controller a net gain of $cv(G)$; this suffices. Because the definition of $cv(G)$ has several cases our argument must also have several cases.

The case where G is empty is clear. If G is a non-empty game and the controller simply remains in control until the last move of the game, where he takes all the boxes, then this is a play where the controller scores $fcv(G) + 4$ or $fcv(G) + 8$ depending on whether the last region opened was a loop or a chain. We deduce that $v(G) \geq fcv(G) + 4$ in all cases and that $v(G) \geq fcv(G) + 8$ if G is composed of a non-zero number of loops. This completes the proof in all cases other than those with $tb(G) = 6$.

So now let us assume that G comprises at least one loop, at least one 3-chain, and that all chains are 3-chains, so $cv(G) = fcv(G) + 6$. Here is a strategy for the controller: keep control until the last loop is opened, and then take all of it (and play optimally afterwards; this is easy by Theorem 10). We check that this guarantees a score of at least $cv(G)$. If the last loop is opened on the last move then the controller gets a score of $fcv(G) + 8 > cv(G)$. If the last loop is opened earlier, and there are $n \geq 1$ 3-chains left, then after the controller has taken all of the loop he has a net score of $fcv(G) + 8 + n \geq fcv(G) + 9$ and is about to play a loony move in a position comprising entirely of 3-chains, which has value at most 3 by Theorem 10(b), so his net gain with this line is at least $fcv(G) + 9 - 3 = fcv(G) + 6 = cv(G)$. □

Part (a) of the next lemma is mentioned on page 86 of [1].

Lemma 15. *Let G be a simple loony endgame.*
(a) If $cv(G) \geq 2$ then $v(G) = cv(G)$.
(b) If $cv(G) < 2$ then $0 \leq v(G) \leq 4$, and $v(G) \equiv cv(G) \bmod 2$.
(c) If $cv(G) < 2$ and if G has a 3-chain then $v(G) \leq v(G; 3) \leq 3$.

Proof. (a) If there are no 3-chains or 4-loops or 6-loops in G then the fully controlled value of any non-empty subgame of G is always at least 0, and so the controlled value is always at least 4. By the preceding lemma the value of the subgame is always at least 4, and hence the controller should initially remain in control as the game progresses. However the value of the empty game is $0 < 2$, and hence the controller should stay in control until the last move of the game, whereupon he takes everything and scores $fcv(G)+4$ or $fcv(G)+8$ depending on whether the final move was in a loop or a chain. The defender's optimal play is hence to ensure that the final move is in a chain, if there are any chains, and we have proved $v(G) = cv(G)$ in this case.

The general case though is more complex. We prove the result by induction on the number of components of G. The strategy of the proof then is for the defender to find a move in a component C of G such that $cv(G \backslash C) \geq 2$ if C is a chain and $cv(G \backslash C) \geq 4$ if C is a loop. If the defender can always do this then he is "flying the plane" (using the notation in [2]) without crashing; this forces the controller to stay in control at all times and this will prove the results. The annoying added technicality is that although it is easy to keep track of fully-controlled values (the fully-controlled-value function is additive), the terminal bonus function is not so well-behaved, so we need to check that we can control this "error term" at all times.

One case that we have already dealt with is when there are no chains at all. Then the result follows from Theorem 12(b). Similarly the case where there are no loops at all is Theorem 10(b).

Another simple case is when G contains a chain of length at least 4. In this case the terminal bonus is always 4 and this will not change if we remove some 3-chains, 4-loops and 6-loops. We know $v(G) \geq cv(G)$ so it suffices to find a strategy for the defender which results in a net loss for him of only $cv(G)$. The defender can start by opening all the 3-chains, 4-loops and 6-loops. For $cv(G) \geq 2$ by assumption, so if a 3-chain is opened then the controlled value goes up by 1 and hence the value of the new game is at least 3 and the controller will stay in control. Similarly if the defender opens a 4-loop or a 6-loop then the controlled value of the new game is at least 4 and hence the controller should stay in control. When all the

3-chains, 4-loops and 6-loops are opened the defender just opens all the loops and then finally all the chains, and it is not hard to check that he makes a net loss of $fcv(G) + 4 = cv(G)$.

The one remaining case is when there is at least one chain, at least one loop, and all chains are 3-chains. In this case we see that the defender can open all but one of the 3-chains and the controller must stay in control because if $G = 3 + H$ with H containing at least one 3-chain then $cv(H) = 1 + cv(G) \geq 3$, so the controller will stay in control. When there is only one 3-chain left, the defender starts opening all the not-very-long loops (that is, 4-loops and 6-loops), and as above checks easily that the controller has to stay in control, until either there are no not-very-long loops left, or there is only one loop left. The case of a 4-loop cannot occur as $cv(3+4_\ell) = 1 < 2$. The result is easily checked for $G = 3+n_\ell$ with $n \geq 6$. Our final case is a 3-chain and more than one very long loop. One checks that the defender can just pick off the smallest very long loop, because the controlled value of any game comprising a 3-chain and at least one very long loop is at least 5 so again the controller must stay in control.

(b) and (c): We prove both statements together, again by induction on the number of components of G. Of course the congruence mod 2 is automatic, as is $v(G) \leq v(G; 3)$ in part (c) and $v(G) \geq 0$ (as G is loony); the issues are proving that $v(G) \leq 4$ and that furthermore $v(G; 3) \leq 3$ if G has a 3-chain. The case of G empty is clear so let us assume G is non-empty. Then the assumption $cv(G) < 2$ implies that G must contain either a 3-chain, a 4-loop or a 6-loop; we deal with each case separately.

The easiest case is when $G = H+L$ with L a 4-loop. Then H must be non-empty, so $tb(G) = tb(H)$ and hence $cv(H) = cv(G) + 4 \leq 5$. So, $0 \leq v(H) \leq 5$ by (a) and our inductive hypothesis, and thus $v(G; L) = |v(H) - 4| \leq 4$ and hence $v(G) \leq 4$.

The next case we consider is when $G = H + 3$. Again H must be non-empty. Removing the 3-chain can sometimes make the terminal bonus increase (from 6 to 8) but we certainly have $cv(H) \leq cv(G) + 3 \leq 4$ so, by the inductive hypothesis, $v(H) \leq 4$ and hence $v(G) \leq v(G; 3) = 1 + |v(H) - 2| \leq 3$, and this proves (c).

The final case is when G has no 4-loops or 3-chains, but has at least one 6-loop. If $G = H + L$ with L the 6-loop then $cv(G) < 2$ implies H is non-empty, so one checks that $tb(G) = tb(H)$, and hence $cv(H) \leq 3$ so $v(H) \leq 4$ by (a) and our inductive hypothesis. Moreover H has no 3-loops or 4-chains, hence $v(H) \geq 2$ (as any move by the defender sacrifices at least 2 boxes before the decision of whether to keep control is made), giving $v(G) \leq v(G; L) = 2 + |v(H) - 4| \leq 4$. □

Remark 16. Part (a) of the preceding lemma becomes false if we drop the assumption that loops have even length. For example if G is the game $4 + 7_\ell + 7_\ell$, then $cv(G) = 2$ but we claim $v(G) = 4$. Indeed, $v(4 + 7_\ell) = 3$ (open the 7-loop), so $v(4 + 7_\ell + 7_\ell; 7_\ell) = 4$ (the controller should lose control because $v(4 + 7_\ell) < 4$). Similarly $v(7_\ell + 7_\ell) = 6$, so $v(4 + 7_\ell + 7_\ell; 4) = 6$ (the controller should this time keep control). We deduce that $v(4+7_\ell+7_\ell)=4$.

The preceding lemma showed how the controlled value influences the value of a simple loony endgame. This corollary of it shows how the value influences the controlled value.

Corollary 17. *Let G be a simple loony endgame.*
(a) If $v(G)\geq 5$ then $cv(G)=v(G)$.
(b) If $v(G)=4$ and if G has a 3-chain then $cv(G)=v(G)$.
(c) If $G=H+n+3$ with $n\geq 3$ and if $v(G;3)\geq 4$ then $cv(G)=v(G)=v(G;3)$.

Proof. Parts (a) and (b) follow immediately from the preceding lemma. Let us prove (c). Note first that $cv(G)\leq v(G)\leq v(G;3)$, so all we need to show is $v(G;3)=cv(G)$. We have $4\leq v(G;3)=1+|v(H+n)-2|$, and because $v(H+n)\geq 0$ we must have $v(H+n)\geq 5$. By (a) we have $cv(H+n)=v(H+n)$, and hence $v(G;3)=v(H+n)-1=cv(H+n)-1=cv(G)$ and we are done. Note that it is in the very last equality where we need the existence of the n-chain (to stop the terminal bonus from changing). $\square$

10. Simple Loony Endgames With No 3-Chains

We return to verifying the correctness of our two algorithms, this time under the assumption that our simple loony endgame G has no 3-chains. Our algorithm giving an optimal move for the defender is very simple in this situation.

Theorem 18. *In a simple loony endgame with at least one loop but no 3-chains, opening the smallest loop is an optimal move.*

Remark 19. Note that our proof *does* assume that the loops have even length. Indeed, if G is the Strings-and-Coins game $G=4+4_\ell+4_\ell+7_\ell+7_\ell$ then opening the smallest loop is not optimal. For $v(4+7_\ell+7_\ell)=4$ (see remark 16), hence $v(4+4_\ell+7_\ell+7_\ell)=0$ (as the value is non-negative but opening the 4-loop has value 0) so $v(G;4_\ell)=4$; however $v(4_\ell+4_\ell+7_\ell+7_\ell)$ is easily checked to be 2 and hence $v(G;4)=2<4$ and opening the 4-chain is better than opening the 4-loop.

Before we prove Theorem 18, we verify it in the special case where the game has one 4-chain and no other chains at all. We then deduce the result in general by using our technique of amalgamating chains developed in Section 7.

Proposition 20. *Let $G=K+L+4$ be a simple loony endgame consisting of a 4-chain, a loop L, and a (possibly empty) collection of loops K each of length at least that of L. Then opening L is an optimal move.*

Proof. Let us first deal with the case $v(G;L) > 4$. In this case we can even show that $v(G;L) = cv(G) = v(G)$. We will show this via a case-by-case check on the size of L. Recall that by definition of a simple loony endgame, L and the loops in K have even length, and hence $v(G;L)$ is even and hence at least 6.

If L has size 4, then $v(G;L) = |4 - v(K+4)| \geq 6$, and hence $v(K+4) \geq 10$, so $cv(K+4) = v(K+4) \geq 10$ by Corollary 17(a), and in particular K is non-empty. We deduce $cv(G) = cv(K+4) - 4 \geq 6$, and now using Lemma 15(a) we have $v(G) = cv(G) = cv(K+4) - 4 = v(K+4) - 4 = |4 - v(K+4)| = v(G;L)$ in this case.

If instead L is a 6-loop then we have $6 \leq v(K+4+L;L) = 2 + |v(K+4) - 4|$, so $v(K+4) = 0$ or $v(K+4) \geq 8$. However $v(K+4) = 0$ is impossible, because $K+4$ is a non-empty loony endgame with no 4-loops (recall L is the smallest loop so there are no 4-loops in K). Hence $v(K+4) \geq 8$ and the argument proceeds just as in the case of a 4-loop above, the conclusion being $v(G) = cv(G) = cv(K+4) - 2 = v(K+4) - 2 = 2 + |4 - v(K+4)| = v(G;L)$.

The final case, under the $v(G;L) > 4$ assumption, is if L is a loop of length $\ell \geq 8$. Then all loops have length at least 8, so $fcv(G) \geq 0$ and $fcv(K+4) \geq 0$, hence $cv(G)$ and $cv(K+4)$ are both at least 4 and again we see $v(G) = cv(G) = \ell - 8 + cv(K+4) = \ell - 8 + v(K+4) = v(G;L)$.

Our conclusion so far is that if $v(G;L) > 4$ then $v(G) = v(G;L)$. We now treat the remaining possibilities. Because G is a simple loony endgame, the optimal move is either in the smallest chain or the smallest loop by Corollary 3, and hence $v(G) = \min\{v(G;4), v(G;L)\}$. Note also that $v(G;4)$ and $v(G;L)$ are congruent mod 4 (because both are congruent mod 4 to the total number of boxes in G; this is because every component of G has even size and hence the value of any subgame of G is even). Hence the only way that the Proposition can fail is if $v(G) = v(G;4) = 0$ and $v(G;L) = 4$. But this cannot happen as $v(G;4) = 2 + |2 - v(K+L)| \geq 2$. □

Proof of Theorem 18. Let us first consider a game H of Dots-and-Boxes consisting entirely of loops, with m_ℓ in H the smallest loop. Let us consider the function $g : \mathbf{Z}_{\geq 4} \to \mathbf{Z}$ defined by $g(n) = v(H+n; m_\ell)$ (the value of the game $H+n$ under the assumption that the first player opens the m-loop). It follows easily from Lemma 2(a) (applied with G equal to H minus m_ℓ) that $|g(n+1) - g(n)| = 1$, and now an easy induction on n shows that $g(n) \leq g(4) + n - 4$ for $n \geq 4$.

We have seen in Proposition 20 that if $G = H + 4$ with H comprising at least one loop, then an optimal move in G is opening the shortest loop in H. We now claim that the same remains true in the game $H+n$ with $n \geq 4$. For if opening the smallest loop in H were not optimal, then opening the n-chain must be optimal, and (with g defined as above) we have

$$\begin{aligned}g(n) &= v(H+n;m_\ell)\\ &> v(H+n) = v(H+n;n)\\ &= n-2+|2-v(H)| = (n-4)+2+|2-v(H)| = n-4+v(H+4;4)\\ &\geq n-4+v(H+4;m_\ell) \text{ (by Proposition 20)}\\ &= n-4+g(4) \geq g(n),\end{aligned}$$

a contradiction. Hence we deduce that if H is non-empty and consists entirely of isolated loops of even length, then an optimal move in $H+n$ $(n \geq 4)$ is opening the shortest loop.

The theorem now follows easily from our "amalgamating chains" technique. Indeed Lemma 6 shows that if opening the shortest loop is optimal in the game $H+n$ for any $n \geq 4$, then it is optimal in the game $H+C$ where C is any non-empty collection of chains each of which has length 4 or more. □

Theorem 18 tells us in which order to open the components of any simple loony endgame G with no 3-chains: first all the loops are opened, and then all the chains. But there still remains the issue of when to lose control. We could use Theorem 18 to recursively compute the value of any simple loony endgame with no 3-chains, by applying it to all the subgames that arise and computing values of all of them in turn, and this would then give us an algorithm for playing such games optimally both as the controller and the defender. However we want to avoid any recursion at all when running our algorithms, which fortunately we can do. Before we state our algorithm for computing the value of such a simple loony endgame, here is an easy lemma which we shall use in the proof and several more times later on.

Lemma 21. *If $w \in \mathbf{Z}_{\geq 0}$, if $f \in \mathbf{Z}_{\geq 0}$, if $w-4f \leq 4$, and if v is the result of iterating the function $x \mapsto |4-x|$, f times, on input w, then we have the following formula for v: let $0 \leq d \leq 7$ be congruent to w mod 8; then $v = |4-d|$ if f is odd, and $v = 4-|4-d|$ if f is even.*

Proof. Induction on f. □

We are now ready to compute the value of a simple loony endgame with no 3-chains.

Corollary 22. *Say G is a simple loony Dots-and-Boxes endgame, with no 3-chains. Write f for the number of 4-loops in G, and set $c = cv(G)$.*
(a) If $c \geq 2$ or G is empty then $v(G) = c$.

Furthermore, if G is non-empty, then the following hold:
(b) if $c \leq 1$ and $f = 0$ and c is odd then $v(G) = 3$;

(c) If $c \leq 1$ and $f = 0$ and c is even then $v(G) \in \{2, 4\}$ and $v(G) \equiv c$ mod 4 (this congruence determines $v(G)$ uniquely). (In the remaining case, $c \leq 1$ and $f \geq 1$, so there is a 4-loop. Let K denote G minus all the 4-loops; then $v(K)$ is computable by (a)–(c).)
(d) If $c \leq 1$ and $f \geq 1$ and K is as above, then let $0 \leq d \leq 7$ be the unique integer in this range congruent to $v(K)$ mod 8. If f is odd then $v(G) = |4 - d|$, and if f is even then $v(G) = 4 - |4 - d|$.

Proof. (a) This is just Lemma 15(a). We now prove (b) and (c) simultaneously, by induction on the number of 6-loops in G, using the observations that if $G = H + 6_\ell$ then $cv(H) = cv(G) + 2 \leq 3$ and hence $v(H) \leq 4$ by Lemma 15, and that H has no 3-chains or 4-loops so $v(H) \geq 2$. To prove part (d), by Theorem 18 we know that opening all the 4-loops is an optimal line of play; $v(G)$ is hence computed from $v(K)$ by iterating the function $x \mapsto |x - 4|$ f times, so the result follows from Lemma 21. □

11. Loops, One 3-Chain, and No Other Chains

Based on the previous section one might hope that opening the smallest loop is optimal in all simple loony endgames. Unfortunately, 3-chains complicate matters immensely. For example, in the simple loony endgame $G = 3 + 6_\ell + 6_\ell + 6_\ell$ with three 6-loops and a 3-chain, opening the smallest loop (one of the 6-loops) is not optimal: it has a value of $v(G; 6_\ell) = 3 > 1 = v(G; 3)$. The remainder of this paper is devoted to a proof of the correctness of our algorithms in the cases where our simple loony endgame has at least one 3-chain. We break the argument into three cases. In this section we will fully analyze simple loony endgames that contain exactly one 3-chain and no other chains at all. In the next section we deal with simple loony endgames that contain exactly one 3-chain and also at least one chain of length 4 or more. Finally Section 13 handles the situation where there is more than one 3-chain.

For the remainder of this section then, G denotes a simple loony endgame with one 3-chain and no other chains. If G has no loops at all then the game has value 3 and the only move is to open the 3-chain, so let us for the rest of this section assume that G also contains at least one loop. In general, sometimes opening the smallest loop in G is strictly better than opening the 3-chain, sometimes it is strictly worse, and in many cases both moves are optimal. We do not compute all of the optimal moves in any given position, we are content with just finding one. We write $G = 3 + H$, where H is non-empty and composed entirely of loops. Note that Theorem 12 tells us the value of H and all of its subgames. The next theorem tells us an optimal move to play in G, and although we may need to apply Theorem 12 to compute the value of some subgames of G, it is clear from the statement of the theorem that we only need to apply it twice.

Theorem 23. *Say $G = H + 3 = K + L + 3$ is a simple loony endgame, where H is non-empty and composed entirely of disjoint loops, the smallest of which is L. The following strategy for playing G is optimal. If $v(H) = 2$ then open the 3-chain. If $v(H) \neq 2$, if L is a 6-loop and and $v(K) = 2$, then open the 3-chain. In all other cases, open L.*

Proof. We know by Corollary 3 that the optimal move in G is to open either L or the 3-chain. Note also that $cv(G)$ must be odd. We first deal with the case $cv(G) \geq 3$. Then $v(G) = cv(G)$ by Lemma 15(a); however $cv(H) = cv(G)+3$ because $tb(H) = 8$ whereas $tb(G) = 6$; so $v(H) = cv(H) \geq 6$ and $v(G; 3) = v(H) - 1 = v(G) + 2$. This implies that opening L is strictly better than opening the 3-chain, and our task is to check that this is what the theorem predicts. We have seen $v(H) > 2$. Moreover, if L were a 6-loop, then either $H = L$ so $v(H\backslash L) = 0$, or $H\backslash L$ is non-empty, so $tb(H\backslash L) = tb(H) = 8$ and hence $cv(H\backslash L) = cv(H) + 2 \geq 8$, implying $v(H\backslash L) \geq 8$. Our theorem is hence correct in the case $cv(G) \geq 3$.

Let us now assume $cv(G) \leq 1$. We know $v(G)$ is odd and hence $v(G) \geq 1$. If $v(H) = 2$ then $v(G; 3) = 1 + |v(H) - 2| = 1 \leq v(G)$, and hence $v(G) = v(G; 3)$ and opening the 3-chain is optimal, as predicted.

The next case we consider is when $cv(G) \leq 1$, $v(H) \neq 2$ and L has length not equal to 6. First note that the length of L cannot be 8 or more, because this would imply that all loops in H had length 8 or more, hence $fcv(G) \geq -1$ and so $cv(G) \geq 5$, a contradiction. The only possibility is that L is a 4-loop; we then claim that opening L is optimal. For if it were not then $v(G; 3) < v(G; L)$, but $v(G; 3) = 1 + |v(H) - 2| \geq 3$ and hence $v(G; L) = v(K + 3 + L; L) \geq 5$, implying $|v(K+3)-4| \geq 5$, so $v(K+3) \geq 9$, so $cv(K+3) \geq 9$ by Corollary 17(a) and hence $cv(G) \geq 5$, a contradiction.

The next case we have to check is when $cv(G) \leq 1$, $v(H) \neq 2$, $L = 6_\ell$ is a 6-loop, and $v(K) = 2$. Then $v(H) = v(K+L; L) = 2+|4-v(K)| = 4$, and thus $v(G; 3) = 3$; however $v(G; L) = 2 + |4 - v(K + 3)| \geq 3$ (as it is odd) and thus $v(G; 3) \leq v(G; L)$ and opening the 3-chain is optimal.

It suffices then to show that if $cv(G) \leq 1$, $v(H) \neq 2$, L is a 6-loop and $v(K) \neq 2$, then opening L is optimal. We do this by contradiction. If opening L were not optimal then $v(G; 3) = v(G) < v(G; L)$. Now $cv(G) \leq 1$ and $v(G)$ is odd, so by Lemma 15(b) we must have $v(G) \in \{1, 3\}$. However $v(G; 3) = 1 + |2 - v(H)| > 1$. Hence $v(G) = v(G; 3) = 3$, so $v(G; L) \geq 5$. Thus $5 \leq 2 + |v(K + 3) - 4|$, so either $v(K + 3) = 1$ or $v(K + 3) \geq 7$. If $v(K + 3) \geq 7$ then $cv(G) \geq 5$, which is a contradiction. The case $v(K + 3) = 1$ cannot occur either; K cannot be empty, the smallest loop in K has length at least 6 and hence opening it costs at least 2, so the only way that $v(K+3)$ can be 1 is if $v(K+3; 3) = 1$, but this implies $v(K) = 2$. $\square$

The previous theorem means that we now have an efficient algorithm for computing an optimal move for any simple loony endgame with one 3-chain and no other chains; we may use this result to compute the value of any such game and hence verify the correctness of our second algorithm in the case where G has only one 3-chain and no other chains.

Corollary 24. *Say $G = 3 + H$ is a simple loony endgame, where H is non-empty and composed entirely of loops. Write f for the number of 4-loops in H, and set $c = cv(G)$.*
(a) If $c \geq 2$ then $v(G) = c$.
(b) If $c \leq 1$ and $v(H) = 2$ then $v(G) = 1$ (note that $v(H)$ can be computed using Theorem 12).
(c) If $c \leq 1$ and $v(H) \neq 2$ and $f = 0$ then $v(G) = 3$.
(d) If $c \leq 1$ and $v(H) \neq 2$ and $f \geq 1$ then let M denote G minus all the 4-loops (and note that $v(M)$ can be computed using (a)–(c) above). Let $0 \leq d \leq 7$ be the unique integer in this range congruent to $v(M)$ modulo 8. If f is odd then $v(G) = |4 - d|$, and if f is even then $v(G) = 4 - |4 - d|$.

Proof. Part (a) This is just Lemma 15(a). Moving to part (b), if $v(H) = 2$ then $v(G; 3) = 1$ and hence $v(G) = 1$ because $v(G)$ is non-negative, odd, and at most 1. For (c), the assumptions imply $v(G; 3) \geq 3$; there are no 4-loops, so if L is the smallest loop then $v(G; L) \geq 2$, and $v(G; L)$ is odd so $v(G; L) \geq 3$. Hence $v(G) \geq 3$. But $v(G) \leq 3$ by Lemma 15(c). To prove (d), write $M = N + 3$, where N, possibly empty, is composed entirely of loops of length 6 or more. If $d \in \mathbf{Z}_{\geq 0}$ we write $N + d \cdot 4_\ell$ for the game comprising the position N plus n 4-loops, so $H = N + f \cdot 4_\ell$. Now H has no chains, so an optimal way to play H is to open all the loops in order, starting with the smallest; hence if $v(N + d \cdot 4_\ell) = 2$ for some $1 \leq d \leq f$, then $v(H) = 2$ contradicting our assumptions. By Theorem 23 we see that opening the 4-loop is optimal in $N + d \cdot 4_\ell + 3 = M + d \cdot 4_\ell$ for all $1 \leq d \leq f$. In particular we can compute $v(G)$ by starting with $v(M)$ and then iterating the function $x \mapsto |4 - x|$, f times. We know $v(G) \leq 3$ by Lemma 15(c), and the result follows by Lemma 21. $\square$

12. Loops, Very Long Chains, and One 3-Chain

In the previous section we analyzed games consisting of some loops and a single 3-chain. In this section we will add long chains of length at least 4 to that situation. We begin by proving that our algorithm for predicting an optimal move is correct in this situation.

Theorem 25. *If G is a simple loony endgame that contains a single 3-chain and at least one chain of length $n \geq 4$, the following strategy is optimal: If $cv(G) \leq 1$*

and G contains a 4-loop, write $G = H + 3 + 4_\ell$ and open the 4-loop if $cv(H+3) = 4$ or if $v(H + 4_\ell) \in \{0, 4\}$. In all other cases, open the 3-chain.

Remark 26. Note that $v(H + 4_\ell)$ can be computed using Corollary 22.

Proof. As usual, the proof breaks into a number of cases; in each case we verify that the theorem gives us an optimal move in each case.

The first case we consider is the case $cv(G) \geq 2$. Then $cv(G\backslash 3) = cv(G)+1 \geq 3$, so by Lemma 15(a) we have $v(G\backslash 3) = v(G) + 1 \geq 3$, hence $v(G; 3) = v(G\backslash 3) - 1 = v(G)$ and so opening the 3-chain is optimal.

The second case is when $cv(G) \leq 1$ but G has no 4-loop. By Lemma 15(c) we have $v(G; 3) \leq 3$, but if L is any loop in G then L must have length at least 6, hence $v(G; L) \geq 2$. Because $v(G; 3)$ and $v(G; L)$ must be congruent mod 2, we deduce that $v(G; 3) \leq v(G; L)$ for any loop, and hence again opening the 3-chain is optimal.

So from now on we may assume $cv(G) \leq 1$ and G has a 4-loop; we write $G = 3 + 4_\ell + H$. Again by Lemma 15(c) we have $v(G; 3) \leq 3$.

The next case, a very easy case, is when $cv(H + 3) = 4$. By Lemma 15 we have $v(H + 3) = 4$, so $v(G; 4_\ell) = 0$ and opening the 4-loop must be optimal.

The last case we deal with is when $cv(H+3) \neq 4$. This implies that $v(H+3) \neq 4$ (if $cv(H + 3) \geq 2$ use Lemma 15(a), and if $cv(H + 3) < 2$ use Lemma 15(c)), and hence $v(G; 4_\ell) \neq 0$. Next we claim $v(G; 4_\ell) \leq 4$; for if $v(G; 4_\ell) \geq 5$ then $v(3+H) \geq 9$, so $cv(3 + H) \geq 9$ (Corollary 17), so $cv(G) \geq 5$ contradicting $cv(G) \leq 1$. We deduce $1 \leq v(G; 4_\ell) \leq 4$. Our aim is to work out which of $v(G; 4_\ell)$ and $v(G; 3)$ is the smaller, and recall that we know that these numbers are congruent modulo 2. We have $v(G; 3) = 1+|2-v(H+4_\ell)|$ and we already know that this is at most 3. But we are now finished, because if $v(H + 4_\ell) \in \{0, 4\}$ then $v(G; 3) = 3 \geq v(G; 4_\ell)$ and we open the 4-loop, but if $v(H+4_\ell) \notin \{0, 4\}$ then $v(G; 3) \in \{1, 2\}$ so $v(G; 3) \leq v(G; 4_\ell)$ and we open the 3-chain. ☐

Once again we can use this result to verify that our algorithm predicting the value of such a game is correct.

Corollary 27. *If G is a simple loony endgame with exactly one 3-chain and at least one chain of length 4 or more, and $c = cv(G)$, we can compute its value $v(G)$ as follows:*
(a) If $c \geq 2$ then $v(G) = c$.
(b) If $c \leq 1$ and c is even, the value of G is 2 except if G has a 4-loop and $cv(G\backslash 4_\ell) = 4$, in which case $v(G) = 0$.
(c) If $c \leq 1$ is odd and $v(G\backslash 3) = 2$, then $v(G) = 1$.
(d) If $c \leq 1$ is odd, $v(G\backslash 3) \neq 2$, and G has no 4-loops, then $v(G) = 3$.
(e) If $c \leq 1$ is odd and $v(G\backslash 3) \neq 2$ and G has $f \geq 1$ 4-loops, then let K denote G minus all the 4-loops (whose value can be computed using (a), (c) and (d)), and let

$0 \le d \le 7$ be congruent to $v(K)$ mod 8. If f is odd then $v(G) = |4-d|$, and if f is even then $v(G) = 4 - |4-d|$.

Remark 28. Note that $v(G\backslash 3)$ can be computed using Corollary 22.

Proof. (a) This is Lemma 15(a). For (b), we note that Lemma 15(c) implies that $v(G) \in \{0, 2\}$. Furthermore $v(G) = 0$ if and only if G has a 4-loop and $v(G\backslash 4_\ell) = 4$. By Corollary 17(b) and Lemma 15(a) this occurs if and only if $cv(G\backslash 4_\ell) = 4$. Part (c) is clear: $v(G\backslash 3) = 2$ implies $v(G; 3) = 1$. In part (d), c is odd, and hence the value of any loop move is odd. If there are no 4-loops, then the value of any loop move is at least 3. Now $v(G\backslash 3) \neq 2$ implies $v(G; 3) \ge 3$, and we conclude $v(G) \ge 3$. We conclude by using Lemma 15(c). Part (e) will follow from Lemma 21, once we have proved that opening all the 4-loops in G is an optimal line of play. Let us consider the position $K + d.4_\ell$, where $1 \le d \le f$. We prove by induction on d that opening the 4-loop is optimal, and this suffices.

There are two cases to consider. The first is when $cv(K + d \cdot 4_\ell) \ge 2$. In this case, $v(K + d \cdot 4_\ell) = cv(K + d \cdot 4_\ell)$ by Lemma 15(a). Moreover, $cv(K + (d-1) \cdot 4_\ell) = 4 + cv(K + d \cdot 4_\ell) \ge 6$, and hence $v(K + (d-1) \cdot 4_\ell) = cv(K + (d-1) \cdot 4_\ell) \ge 6$, so $v(K + d \cdot 4_\ell; 4_\ell) = v(K + (d-1) \cdot 4_\ell) - 4 = v(K + d \cdot 4_\ell)$ and hence opening a 4-loop is optimal in this situation.

The other case is when $cv(K + d \cdot 4_\ell) \le 1$. Write $K = 3 + M$ and observe that for any $e \ge 1$, the game $M + e \cdot 4_\ell$ has no 3-chains, so by Theorem 18 an optimal way to play it is to open the 4-loop. We claim $v(M + d \cdot 4_\ell) \neq 2$; for if $v(M + d \cdot 4_\ell) = 2$ then $v(M + e \cdot 4_\ell) = 2$ for all $e \ge d$ and in particular $v(G\backslash 3) = 2$, contradicting our assumptions. We next note that $cv(K + d \cdot 4_\ell) \le 1$ implies $cv(M + d \cdot 4_\ell) \le 2$ and hence $v(M + d \cdot 4_\ell) \le 4$ by Lemma 15(a) and (b). Now c is odd, so $v(M + d \cdot 4_\ell)$ is even, and hence $v(M + d \cdot 4_\ell) \in \{0, 4\}$. So by Theorem 25 opening a 4-chain is optimal in $3 + M + d \cdot 4_\ell = K + d \cdot 4_\ell$, and we are finished. □

13. Simple Loony Endgames With At Least Two 3-Chains

The last situation we need to analyze is when G has at least two 3-chains (and of course possibly other chains and loops). Theorem 29 describes an optimal strategy for such a situation. This result concludes the proof that our algorithm to compute an optimal move in a simple loony endgame is correct.

Theorem 29. *If G is a simple loony endgame with at least two 3-chains, the following strategy is optimal: If there is a 4-loop in G, write $G = H + 3 + 4_\ell$ and open the 4-loop if $cv(H + 3) = 4$ or $cv(H + 4_\ell) = 4$ or $cv(H) = 4$. In all other cases, open the 3-chain.*

Proof. As usual, we break things up into cases, and verify that the theorem predicts an optimal move in each case.

The first case we consider is when G has no 4-loop, and our task is hence to prove that opening the 3-chain is optimal. If $v(G; 3) \geq 4$ then Corollary 17(c) does the job. If however $v(G; 3) \leq 3$ then opening the 3-chain must be optimal, because the value of opening any loop is at least 2, and is congruent to $v(G; 3)$ mod 2, so must be at least $v(G; 3)$.

So we may now assume that $G = H + 3 + 4_\ell$ contains a 4-loop, and the next three cases we consider will be the three cases where the theorem tells us to open it. The easiest case is when $cv(H + 3) = 4$; then $v(H + 3) = 4$ by Lemma 15(a) and hence $v(G; 4_\ell) = 0$, so opening the 4-loop is optimal.

Before we continue with our case-by-case analysis, we observe that if $v(G; 4_\ell) \geq 5$ then $v(H+3) \geq 9$, and by repeated uses of Lemma 15(a) and Corollary 17(a) we may deduce $cv(H + 3) \geq 9$, $cv(H) \geq 10$ and $cv(H + 4_\ell) \geq 6$ (note $tb(H + 4_\ell) \geq tb(H)$). Hence $v(H + 3) \geq 9$, $v(H) \geq 10$ and $v(H + 4_\ell) \geq 6$.

The next case we consider is when $cv(H + 4_\ell) = 4$. Then $v(H + 4_\ell) = 4$ and hence $v(G; 3) = 3$. Because $v(H + 4_\ell) \leq 5$, the argument in the previous paragraph shows that $v(G; 4_\ell) \leq 4$; moreover $v(G; 4_\ell)$ is odd, and hence at most 3, and so opening the 4-loop is optimal, as predicted.

The next case we consider is when $cv(H) = 4$. The argument in the last-but-one paragraph then shows that $v(G; 4_\ell) \leq 4$ and hence $v(G; 4_\ell) \leq 3$. Furthermore $v(H) = 4$, so $v(H + 4_\ell; 4_\ell) = 0$, thus $v(H + 4_\ell) = 0$. We again deduce $v(G; 3) = 3$, so again opening the 4-loop in G is optimal.

The final case we need to consider is when $G = 3 + 4_\ell + H$, $cv(H + 3) \neq 4$, $cv(H + 4_\ell) \neq 4$ and $cv(H) \neq 4$; we need to check in this case that opening the 3-chain is optimal. If $v(G; 3) \geq 4$ then Corollary 17(c) gives the result. If however $v(G; 3) \leq 3$ then $v(G; 3) \in \{1, 2, 3\}$ and we need to rule out the case $v(G; 4_\ell) < v(G; 3)$. Because these numbers are non-negative and congruent mod 2, there are only two possibilities: the first that $v(G; 4_\ell) = 0 < v(G; 3) = 2$, and the second that $v(G; 4_\ell) = 1 < v(G; 3) = 3$. The first is easily ruled out, as $v(G; 4_\ell) = 0$ implies $v(H + 3) = 4$ and hence $cv(H + 3) = 4$ by Corollary 17(b), contradicting our assumptions. As for the second, we see that $v(G; 3) = 3$ implies $v(H + 4_\ell) \in \{0, 4\}$. The case $v(H + 4_\ell) = 4$ cannot happen because it implies $cv(H + 4_\ell) = 4$, contradicting our assumptions. The final case to deal with is $v(H + 4_\ell) = 0$; but this implies $v(H) = 4$ and hence $cv(H) = 4$, again contradicting our assumptions. The proof of the theorem is hence complete. □

Finally, we show how to use this result to compute the value of the simple loony endgames in question; this will finish the proof of correctness of our algorithm for computing values of simple loony endgames.

Corollary 30. *If G is a simple loony endgame with at least two 3-chains, we can*

compute its value as follows:
(a) If $cv(G) \geq 2$ *then* $v(G) = cv(G)$*.*
(b) If $cv(G) < 2$ *and* $cv(G)$ *is even, the value of* G *is 2 except if* G *has a 4-loop and* $cv(G\backslash 4_\ell) = 4$*, in which case* $v(G) = 0$*.*
(c) If $cv(G) < 2$ *and* $cv(G)$ *is odd then* $v(G) = 1$*.*

Proof. (a) This is Lemma 15(a). In part (b), we know $v(G)$ is even, so Lemma 15(b) and (c) imply $v(G) \in \{0, 2\}$. If G has no 4-loop then $v(G) = 0$ is impossible, and hence $v(G) = 2$ as claimed. If $G = H + 3 + 4_\ell$ has a 4-loop, then we want to use Theorem 29 to find an optimal move. Note first that $cv(H + 4_\ell)$ and $cv(H)$ are odd, so they cannot be 4, and so the only question we need to consider is whether $cv(G\backslash 4_\ell) = cv(H+3) = 4$ or not. If $cv(H+3) = 4$ then $v(G\backslash 4_\ell) = 4$, so $v(G; 4_\ell) = 0$ and hence $v(G) = 0$ as claimed. If however $cv(H + 3) \neq 4$ then Theorem 29 tells us that opening a 3-chain is optimal, and hence $v(G) > 0$, giving $v(G) = 2$ as the only possibility. In part (c), $v(G)$ is odd, and Lemma 15(b) and (c) then imply $v(G) \in \{1, 3\}$. If $v(G; 3) = 1$ then $v(G) = 1$ which is what we want. If $v(G; 3) \geq 5$ then Corollary 17(c) implies $cv(G) \geq 5$, a contradiction. The only possibility left is $v(G; 3) = 3$, from which we need to deduce $v(G) = 1$.

Now $v(G; 3) = 3$ implies $v(G\backslash 3) \in \{0, 4\}$. However $v(G\backslash 3) = 4$ is impossible, as Corollary 17(b) would then imply $cv(G\backslash 3) = 4$ which would imply $cv(G) = 3$, a contradiction.

We deduce $v(G\backslash 3) = 0$; however $G\backslash 3$ is non-empty (as it contains a 3-chain), and hence $G\backslash 3$ must contain a 4-loop. Write $G = 3+4_\ell+H$, so $v(4_\ell+H) = 0$ and hence $v(H) = 4$. But this implies $cv(H) = 4$ (Corollary 17(b) again), so $cv(H + 3) = 3$, thus $v(H + 3) = 3$ and hence $v(G; 4_\ell) = 1$, proving that $v(G) = 1$. □

Acknowledgements. The first author would like to thank the player `Carroll` at `littlegolem.net` for several helpful discussions about values; in particular the correct formulation of Theorem 12 was only made after one such discussion. The second author would like to thank Astrid Bönning, for sharing some original ideas. Part of this work was initially done by the second author for a Dutch school project, for which he would like to thank his teacher and supervisor, Bart van de Rotten.

References

[1] Elwyn Berlekamp. *The Dots and Boxes Game: Sophisticated Child's Play* A K Peters Ltd., Natick, MA, 2000.

[2] Elwyn Berlekamp and Katherine Scott. Forcing your opponent to stay in control of a loony dots-and-boxes endgame. In *More games of no chance (Berkeley, CA, 2000)*, volume 42 of *Math. Sci. Res. Inst. Publ.*, pages 317–330. Cambridge Univ. Press, Cambridge, 2002.

[3] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays, Vol. 3.* A K Peters Ltd., Natick, MA, second edition, 2003.

[4] Katherine Scott. Loony Endgames in Dots-and-Boxes. UC Berkeley MS thesis, 2000.