Playing Games with Algorithms: Algorithmic Combinatorial Game Theory

Erik D. Demaine

Department of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada, eddemaine@uwaterloo.ca

Abstract. Combinatorial games lead to several interesting, clean problems in algorithms and complexity theory, many of which remain open. The purpose of this paper is to provide an overview of the area to encourage further research. In particular, we begin with general background in combinatorial game theory, which analyzes ideal play in perfect-information games. Then we survey results about the complexity of determining ideal play in these games, and the related problems of solving puzzles, in terms of both polynomial-time algorithms and computational intractability results. Our review of background and survey of algorithmic results are by no means complete, but should serve as a useful primer.

1 Introduction

Many classic games are known to be computationally intractable: one-player puzzles are often NP-complete (as in Minesweeper), and two-player games are often PSPACE-complete (as in Othello) or EXPTIME-complete (as in Checkers, Chess, and Go). Surprisingly, many seemingly simple puzzles and games are also hard. Other results are positive, proving that some games can be played optimally in polynomial time. In some cases, particularly with one-player puzzles, the computationally tractable games are still interesting for humans to play.

After reviewing some of the basic concepts in combinatorial game theory in Section 2, Sections 3–5 survey several of these algorithmic and intractability results. We do not intend to give a complete survey, but rather to give an introduction to the area. Given the space restrictions, the sample of results mentioned here reflect a personal bias: results about "well-known" games (in North America), some of the results I find interesting, and results in which I have been involved. For a more complete overview, please see the full version of this paper [12].

Combinatorial game theory is to be distinguished from other forms of game theory arising in the context of economics. Economic game theory has applications in computer science as well, most notably in the context of auctions [11] and analyzing behavior on the Internet [33].

2 Combinatorial Game Theory

A *combinatorial game* typically involves two players, often called *Left* and *Right*, alternating play in well-defined *moves*. However, in the interesting case of a *combinatorial puzzle*, there is only one player, and for *cellular automata* such as Conway's Game of Life, there are no players. In all cases, no randomness or hidden information is permitted: all players know all information about gameplay (*perfect information*). The problem is thus purely strategic: how to best play the game against an ideal opponent.

It is useful to distinguish several types of two-player perfect-information games [3, pp. 16–17]. A common assumption is that the game terminates after a finite number of moves (the game is *finite* or *short*), and the result is a unique winner. Of course, there are exceptions: some games (such as Life and Chess) can be *drawn* out forever, and some games (such as tic-tac-toe and Chess) define *ties* in certain cases. However, in the combinatorial-game setting, it is useful to define the *winner* as the last player who is able to move; this is called *normal play*. If, on the other hand, the winner is the first player who cannot move, this is called *misère play*. (We will normally assume normal play.) A game is *loopy* if it is possible to return to previously seen positions (as in Chess, for example). Finally, a game is called *impartial* if the two players (Left and Right) are treated identically, that is, each player has the same moves available from the same game position; otherwise the game is called *partizan*.

A particular two-player perfect-information game without ties or draws can have one of four *outcomes* as the result of ideal play: player Left wins, player Right wins, the first player to move wins (whether it is Left or Right), or the second player to move wins. One goal in analyzing two-player games is to determine the outcome as one of these four categories, and to find a strategy for the winning player to win. Another goal is to compute a deeper structure to games described in the remainder of this section, called the *value* of the game.

A beautiful mathematical theory has been developed for analyzing two-player combinatorial games. The most comprehensive reference is the book *Winning Ways* by Berlekamp, Conway, and Guy [3], but a more mathematical presentation is the book *On Numbers and Games* by Conway [8]. See also [21] for a bibliography. The basic idea behind the theory is simple: a two-player game can be described by a rooted tree, each node having zero or more *left* branches correspond to options for player Left to move and zero or more *right* branches corresponding to options for player Right to move; leaves corresponding to finished games, the winner being determined by either normal or misère play. The interesting parts of combinatorial game theory are the several methods for manipulating and analyzing such games/trees. We give a brief summary of some of these methods in this section.

2.1 Conway's Surreal Numbers

A richly structured special class of two-player games are John H. Conway's *surreal numbers*[8], a vast generalization of the real and ordinal number systems. Basically, a surreal number $\{L \mid R\}$ is the "simplest" number larger than all Left options (in L) and smaller than all Right options (in R); for this to constitute a number, all Left and Right options must be numbers, defining a total order, and each Left option must be less than each Right option. See [8] for more formal definitions.

For example, the simplest number without any larger-than or smaller-than constraints, denoted $\{\,|\,\}$, is 0; the simplest number larger than 0 and without smaller-than constraints, denoted $\{0\,|\,\}$, is 1. This method can be used to generate all natural numbers and indeed all ordinals. On the other hand, the simplest number less than 0, denoted $\{\,|\,0\}$, is -1; similarly, all negative integers can be generated. Another example is the simplest number larger than 0 and smaller than 1, denoted $\{0\,|\,1\}$, which is $\frac{1}{2}$; similarly, all dyadic rationals can be generated. After a countably infinite number of such construction steps, all real numbers can be generated; after many more steps, the surreals are all numbers that can be generated in this way.

What is interesting about the surreals from the perspective of combinatorial game theory is that they are a subclass of all two-player perfect-information games, and some of the surreal structure, such as addition and subtraction, carries over to general games. Furthermore, while games are not totally ordered, they can still be compared to some surreal numbers and, amazingly, how a game compares to the surreal number 0 determines exactly the outcome of the game. This connection is detailed in the next few paragraphs.

First we define some algebraic structure of games that carries over from surreal numbers. Two-player combinatorial games, or trees, can simply be represented as $\{L\,|\,R\}$ where, in contrast to surreal numbers, no constraints are placed on L and R. The *negation* of a game is the result of reversing the roles of the players Left and Right throughout the game. The (*disjunctive*) *sum* of two (sub)games is the game in which, at each player's turn, the player has a binary choice of which subgame to play, and makes a move in precisely that subgame. A partial order is defined on games recursively: a game x is *less than or equal to* a game y if every Left option of x is less than y and every Right option of y is more than x.

Note that while $\{-1\,|\,1\} = 0 = \{\,|\,\}$ in terms of numbers, $\{-1\,|\,1\}$ and $\{\,|\,\}$ denote different games (lasting 1 move and 0 moves, respectively), and in this sense are *equal* in *value* but not *identical* symbolically or game-theoretically. Nonetheless, the games $\{-1\,|\,1\}$ and $\{\,|\,\}$ have the same outcome: the second player to move wins.

Amazingly, this holds in general: two equal numbers represent games with equal outcome (under ideal play). In particular, all games equal to 0 have the outcome that the second player to move wins. Furthermore, all games equal to a positive number have the outcome that the Left player wins; more generally, all positive games (games larger than 0) have this outcome. Symmetrically, all negative games have the outcome that the Right player wins (this follows automatically by the negation operation).

There is one outcome not captured by the characterization into zero, positive, and negative games: the first player to move wins. An example of such a game is $\{1\,|\,0\}$; this fails to be a surreal number because $1 \geq 0$. By the claim above, $\{1\,|\,0\} \parallel 0$. Indeed, $\{1\,|\,0\} \parallel x$ for all surreal numbers x, $0 \leq x \leq 1$. In contrast,

$x < \{1 \,|\, 0\}$ for all $x < 0$ and $\{1 \,|\, 0\} < x$ for all $1 < x$. In general it holds that a game is fuzzy with some surreal numbers in an interval $[-n, n]$ but comparable with all surreals outside that interval.

For brevity we omit many other useful notions in combinatorial game theory, such as additional definitions of summation, super-infinitesimal games $*$ and $\uparrow$, mass, temperature, thermographs, the simplest form of a game, remoteness, and suspense; see [3, 8].

2.2 Sprague-Grundy Theory

A celebrated result in combinatorial game theory is the characterization of impartial two-player perfect-information games, discovered independently in the 1930's by Sprague [39] and Grundy [25]. Recall that a game is *impartial* if it does not distinguish between the players Left and Right. The Sprague-Grundy theory [39, 25, 8, 3] states that every finite impartial game is equivalent to an instance of the game of Nim, characterized by a single natural number n. This theory has since been generalized to all impartial games by generalizing Nim to all ordinals n; see [8, 38].

Nim [5] is a game played with several *heaps*, each with a certain number of tokens. A Nim game with a single heap of size n is denoted by $*n$ and is called a *nimber*. During each move a player can pick any pile and reduce it to any smaller nonnegative integer size. The game ends when all piles have size 0. Thus, a single pile $*n$ can be reduced to any of the smaller piles $*0$, $*1$, $\ldots$, $*(n-1)$. Multiple piles in a game of Nim are independent, and hence any game of Nim is a sum of single-pile games $*n$ for various values of n. In fact, a game of Nim with k piles of sizes n_1, n_2, $\ldots$, n_k is equivalent to a one-pile Nim game $*n$, where n is the binary XOR of n_1, n_2, $\ldots$, n_k. As a consequence, Nim can be played optimally in polynomial time (polynomial in the encoding size of the pile sizes).

Even more surprising is that *every* impartial two-player perfect-information game has the same value as a single-pile Nim game, $*n$ for some n. The number n is called variously the *G-value*, *Grundy-value*, or *Sprague-Grundy function* of the game. It is easy to define: suppose that game x has k options $y_1, \ldots, y_k$ for the first move (independent of which player goes first). By induction, we can compute $y_1 = *n_1$, $\ldots$, $y_k = *n_k$. The theorem is that x equals $*n$ where n is the smallest natural number not in the set $\{n_1, \ldots, n_k\}$. This number n is called the *minimum excluded value* or *mex* of the set. This description has also assumed that the game is finite, but this is easy to generalize [8, 38].

The Sprague-Grundy function can increase by at most 1 at each level of the game tree, and hence the resulting nimber is linear in the maximum number of moves that can be made in the game; the encoding size of the nimber is only logarithmic in this count. Unfortunately, computing the Sprague-Grundy function for a general game by the obvious method uses time linear in the number of possible states, which can be exponential in the nimber itself.

Nonetheless, the Sprague-Grundy theory is extremely helpful for analyzing impartial two-player games, and for many games there is an efficient algorithm to determine the nimber. Examples include Nim itself, Kayles, and various generalizations [27]; and Cutcake and Maundy Cake [3, pp. 26–29]. In all of these examples, the Sprague-Grundy function has a succinct characterization (if somewhat difficult to prove); it can also be easily computed using dynamic programming.

2.3 Strategy Stealing

Another useful technique in combinatorial game theory for proving that a particular player must win is *strategy stealing*. The basic idea is to assume that one player has a winning strategy, and prove that in fact the other player has a winning strategy based on that strategy. This contradiction proves that the second player must in fact have a winning strategy. An example of such an argument is given in Section 3.1. Unfortunately, such a proof by contradiction gives no indication of what the winning strategy actually is, only that it exists. In many situations, such as the one in Section 3.1, the winner is known but no polynomial-time winning strategy is known.

2.4 Puzzles

There is little theory for analyzing combinatorial puzzles (one-player games) along the lines of two-player theory summarized in this section. We present one such viewpoint here. In most puzzles, solutions subdivide into a sequence of moves. Thus, a puzzle can be viewed a tree, similar to a two-player game except that edges are not distinguished between Left and Right. The goal is to reach a position from which there are no valid moves (normal play). Loopy puzzles are common; to be more explicit, repeated subtrees can be converted into self-references to form a directed graph.

A consequence of the above view is that a puzzle is basically an impartial two-player game except that we are not interested in the outcome from two players alternating in moves. Rather, questions of interest in the context of puzzles are (a) whether a given puzzle is solvable, and (b) finding the solution with the fewest moves. An important open direction of research is to develop a general theory for resolving such questions, similar to the two-player theory. For example, using the analogy between impartial two-player games described above, the notion of sums of puzzles makes sense, although it is not clear that it plays a similarly key role as with games.

3 Algorithms for Two-Player Games

Many nonloopy two-player games are PSPACE-complete. This is fairly natural because games are closely related to boolean expressions with alternating quantifiers (for which deciding satisfiability is PSPACE-complete): there exists a move for Left such that, for all moves for Right, there exists another move for Left, etc. A PSPACE-completeness result has two consequences. First, being in

PSPACE means that the game can be played optimally, and typically all positions can be enumerated, using possibly exponential time but only polynomial space. Thus such games lend themselves to a somewhat reasonable exhaustive search for small enough sizes. Second, the games cannot be solved in polynomial time unless P = PSPACE, which is even "less likely" than P equaling NP.

On the other hand, loopy two-players games are often EXPTIME-complete. Such a result is one of the few types of true lower bounds in complexity theory, implying that all algorithms require exponential time in the worst case.

In this section we briefly survey some of these complexity results and related positive results, ordered roughly chronologically by the first result on a particular game. See also [18] for a related survey. Because of space constraints we omit discussion of games on graphs, as well as the board games Gobang, Shogi, and Othello. For details on these and other games, please refer to the full version of this paper [12].

3.1 Hex

Hex [3, pp. 679–680] is a game designed by Piet Hein and played on a diamond-shaped hexagonal board; see Fig. 1. Players take turns filling in empty hexagons with their color. The goal of a player is to connect the opposite sides of their color with hexagons of their color. (In the figure, one player is solid and the other player is dotted.) A game of Hex can never tie, because if all hexagons are colored arbitrarily, there is precisely one connecting path of an appropriate color between opposite sides of the board.

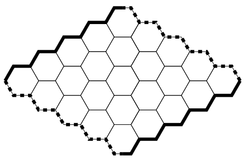


Fig. 1. A 5×5 Hex board.

Nash [3, p. 680] proved that the first player to move can win by using a strategy stealing argument (see Section 2.3). In contrast, Reisch [35] proved that determining the outcome of a general position in Hex is PSPACE-complete.

3.2 Checkers (Draughts)

The standard 8×8 game of Checkers (Draughts), like many classic games, is finite and hence can be played optimally in constant time (in theory). The complexity of playing in a general $n \times n$ board from a natural starting position, such as the one in Fig. 2, is open. However, deciding the outcome of an arbitrary configuration is PSPACE-hard [22]. If a polynomial bound is placed on the number of moves that are allowed in between jumps (which is a reasonable generalization of the drawing rule in standard Checkers [22]), then the problem is in PSPACE and hence is PSPACE-complete. Without such a restriction, however, Checkers is EXPTIME-complete [37].

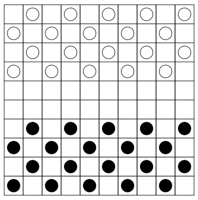


Fig. 2. A natural starting configuration for 10×10 Checkers, from [22].

On the other hand, certain simple questions about Checkers can be answered in polynomial time [22, 13]. Can one player remove all the other player's pieces in one move (by several jumps)? Can one player king a piece in one move? Because of the notion of parity on $n \times n$ boards, these questions reduce to checking the existence of an Eulerian path or general path, respectively, in a particular directed graph; see [22, 13]. However, for boards defined by general graphs, at least the first question becomes NP-complete [22].

3.3 Go

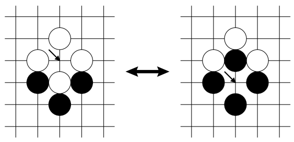


Fig. 3. A simple ko.

Presented at the same conference as the Checkers result in the previous section (FOCS'78), Lichtenstein and Sipser [32] proved that the classic oriental game of Go is also PSPACE-hard for an arbitrary configuration on an $n \times n$ board. This proof does not involve any situations called ko's, where a rule must be invoked to avoid infinite play. In contrast, Robson [36] proved that Go is EXPTIME-complete when ko's are involved, and indeed used judiciously. The type of ko used in this reduction is shown in Fig. 3. When one of the players makes a move shown in the figure, the ko rule prevents (in particular) the other move shown in the figure to be made immediately afterwards.

Recently, Wolfe [41] has shown that even Go endgames are PSPACE-hard. More precisely, a *Go endgame* is when the game has reduced to a sum of Go subgames, each equal to a polynomial-size game tree. This proof is based on several connections between Go and combinatorial game theory detailed in a book by Berlekamp and Wolfe [2].

3.4 Chess

Fraenkel and Lichtenstein [23] proved that a generalization of the classic game Chess to $n \times n$ boards is EXPTIME-complete. Specifically, their generalization has a unique king of each color, and for each color the numbers of pawns, bishops, rooks, and queens increase as some fractional power of n. (Knights are not needed.) The initial configuration is unspecified; what is EXPTIME-hard is to determine the winner (who can checkmate) from an arbitrary specified configuration.

3.5 Hackenbush

Hackenbush is one of the standard examples of a combinatorial game in *Winning Ways*; see e.g. [3, pp. 4–9]. A position is given by a graph with each edge colored either red (Left), blue (Right), or green (neutral), and with certain vertices marked as *rooted*. Players take turns removing an edge of an appropriate color (either neutral or their own color), which also causes all edges not connected to a rooted vertex to be removed. The winner is determined by normal play. Chapter 7 of *Winning Ways* [3, pp. 183–220] proves that determining the *value* of a red-blue Hackenbush position is NP-hard.

3.6 Domineering (Crosscram)

Domineering or *crosscram* [3, pp. 117–124] is a partizan game involving placement of horizontal and vertical dominoes in a grid; a typical starting position is an $m \times n$ rectangle. Left can play only vertical dominoes and Right can play only horizontal dominoes, and dominoes must remain disjoint. The winner is determined by normal play.

The complexity of Domineering, computing either the outcome or the value of a position, remains open. Lachmann, Moore, and Rapaport [31] have shown that the winner and a winning strategy can be computed in polynomial time for $m \in \{1, 2, 3, 4, 5, 7, 9, 11\}$ and all n. These algorithms do not compute the value of the game, nor the optimal strategy, only a winning strategy. We omit discussion of the related game *Cram* [3, pp. 468–472].

3.7 Dots-and-Boxes and Strings-and-Coins

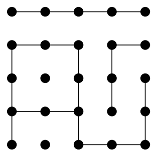


Fig. 4. A Dots-and-Boxes endgame.

Dots-and-Boxes [1], [3, pp. 507–550] is a well-known children's game in which players take turns drawing horizontal and vertical edges connecting pairs of dots in an $m \times n$ subset of the lattice. Whenever a player makes a move that encloses a unit square with drawn edges, the player is awarded a point and must then draw another edge in the same move. The winner is the player with the most points when the entire grid has been drawn. See Fig. 4 for an example of a position.

A generalization arising from the dual of Dots-and-Boxes is *Strings-and-Coins*. This game involves a sort of graph whose vertices are *coins* and whose edges are *strings*. The coins may be tied to each other and to the "ground" by strings; the latter connection can be modeled as a loop in the graph. Players alternate cutting strings (removing edges), and if a coin is thereby freed, that player collects the coin and cuts another string in the same move. The player to collect the most coins wins.

Winning Ways [3, pp. 543–544] describes a proof that Strings-and-Coins endgames are NP-hard. Eppstein [18] observes that this reduction should also apply to endgame instances of Dots-and-Boxes.

3.8 Amazons

Amazons is a game invented by Walter Zamkauskas in 1988, containing elements of Chess and Go. Gameplay takes place on a 10×10 board with four *amazons* of each color arranged as in Fig. 5 (left). In each turn, Left [Right] moves a black [white] amazon to any unoccupied square accessible by a Chess queen's move, and fires an arrow to any unoccupied square reachable by a Chess queen's move from the amazon's new position. The arrow (drawn as a circle) now occupies its square; amazons and shots can no longer pass over or land on this square. The winner is determined by normal play.

Gameplay in Amazons typically split into a sum of simpler games because arrows partition the board into multiple components. In particular, the *endgame* begins when each component of the game contains amazons of only a single color. Then the goal of each player is simply to maximize the number of moves in each component. Buro [7] proved that maximizing the number of moves in a single component is NP-complete (for $n \times n$ boards). In a general endgame, deciding the outcome may not be in NP because it is difficult to prove that the opponent has no better strategy. However, Buro [7] proved that this problem is *NP-equivalent* [24], i.e., the problem can be solved by a polynomial number of calls to an algorithm for any NP-complete problem, and vice versa.

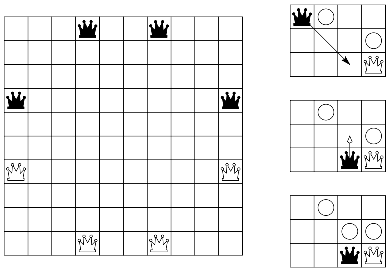


Fig. 5. The initial position in Amazons (left) and black trapping a white amazon (right).

It remains open whether deciding the outcome of a general Amazons position is PSPACE-hard. The problem is in PSPACE because the number of moves in a game is at most the number of squares in the board.

3.9 Phutball

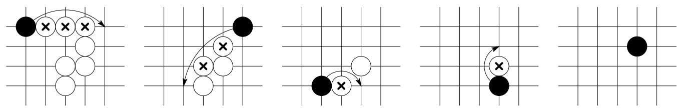


Fig. 6. A single move in Phutball consisting of four jumps.

Conway's game of *Philosopher's Football* or *Phutball* [3, pp. 688–691] involves white and black stones on a rectangular grid such as a Go board. Initially, the unique black stone (the *ball*) is placed in the middle of the board, and there are no white stones. Players take turns either placing a white stone in any unoccupied position, or moving the ball by a sequence of *jumps* over consecutive sequences of white stones each arranged horizontally, vertically, or diagonally. See Fig. 6. A jump causes immediate removal of the white stones jumped over, so those stones cannot be used for a future jump in the same move. Left and Right have opposite sides of the grid marked as their *goal lines*. Left's goal is to end a move with the ball on or beyond Right's goal line, and symmetrically for Right.

Phutball is inherently loopy and it is not clear that either player has a winning strategy: the game may always be drawn out indefinitely. One counterintuitive aspect of the game is that white stones placed by one player may be "corrupted" for better use by the other player. Recently, however, Demaine, Demaine, and Eppstein [13] found an aspect of Phutball that could be analyzed. Specifically, they proved that determining whether the current player can win in a single move ("mate in 1" in Chess) is NP-complete. This result leaves open the complexity of determining the outcome of a given game position.

4 Algorithms for Puzzles

Many puzzles (one-player games) have short solutions and are NP-complete. However, several puzzles based on motion-planning problems are harder, although often being in a bounded region, only PSPACE-complete. However, when generalized to the entire plane and unboundedly many pieces, puzzles often become undecidable.

This section briefly surveys some of these results, following the structure of the previous section. Again, because of space constraints, we omit discussion of several puzzles: Instant Insanity, Cryptarithms, Peg Solitaire, and Shanghai. For details on these and other puzzles, please refer to the full version of this paper [12].

4.1 Sliding Blocks

The Fifteen Puzzle [3, p. 756] is a classic puzzle consisting of 15 numbered square blocks in a 4×4 grid; one square in the grid is a hole which permits blocks to slide. The goal is to order the blocks as increasing in English reading order. See [29] for the history of this puzzle.

A natural generalization of the Fifteen Puzzle is the $n^2 - 1$ *puzzle* on an $n \times n$ grid. It is easy to determine whether a configuration of the $n^2 - 1$ puzzle can reach another: the two permutations of the block numbers (in reading order) simply need to match in *parity*, that is, whether the number of inversions (out-of-order pairs) is even or odd. However, to find a solution using the fewest slides is NP-complete [34]. It is also NP-hard to approximate within an additive constant, but there is a polynomial-time constant-factor approximation [34].

A harder sliding-block puzzle is *Rush Hour*, distributed by Binary Arts, Inc. Several 1×2, 1×3, 2×1, and 3×1 rectangles are arranged in an $m \times n$ grid. Horizontally oriented blocks can slide left and right, and vertically oriented blocks can slide up and down, provided the blocks remain disjoint. The goal is to remove a particular block from the puzzle via an opening in the bounding rectangle. Recently, Flake and Baum [20] proved that this formulation of Rush Hour is PSPACE-complete.

A classic reference on a wide class of sliding-block puzzles is by Hordern [29]. One general form of these puzzles is that rectangular blocks are placed in a rectangular box, and each block can be moved horizontally and vertically, provided the blocks remain disjoint. The goal is to re-arrange one configuration into another. To my knowledge, the complexity of deciding whether such puzzles are solvable remains open. A simple observation is that, as with Rush Hour, they are all in PSPACE.

4.2 Minesweeper

Minesweeper is a well-known imperfect-information computer puzzle popularized by its inclusion in Microsoft Windows. Gameplay takes place on an $n \times n$ board, and the player does not know which squares contain mines. A move consists of uncovering a square; if that square contains a mine, the player loses, and otherwise the player is revealed the number of mines in the 8 adjacent squares. The player also knows the total number of mines.

There are several problems of interest in Minesweeper. For example, given a configuration of partially uncovered squares (each marked with the number of adjacent mines), is there a position that can be safely uncovered? More generally, what is the probability that a given square contains a mine, assuming a uniform distribution of remaining mines? A different generalization of the first question is whether a given configuration is *consistent*, i.e., can be realized by a collection of mines. A consistency checker would allow testing whether a square can be guaranteed to be free of mines, thus answering the first question. A final problem is to decide whether a given configuration has a unique realization.

Kaye [30] proved that testing consistency is NP-complete. This result leaves open the complexity of the other questions mentioned above.

4.3 Pushing Blocks

Similar in spirit to the sliding-block puzzles in Section 4.1 are *pushing-block puzzles*. In sliding-block puzzles, an exterior agent can move arbitrary blocks around, whereas pushing-block puzzles embed a *robot* that can only move adjacent blocks but can also move itself within unoccupied space. The study of this type of puzzle was initiated by Wilfong [40], who proved that deciding whether the robot can reach a desired target is NP-hard when the robot can push and pull L-shaped blocks.

Since Wilfong's work, research has concentrated on the simpler model in which the robot can only push blocks and the blocks are unit squares. Types of puzzles are further distinguished by how many blocks can be pushed at once, whether blocks can additionally be defined to be *unpushable* or *fixed* (tied to the board), how far blocks move when pushed, and the goal (usually for the robot to reach a particular location). Dhagat and O'Rourke [17] initiated the exploration of square-block puzzles by proving that PUSH-$*$, in which arbitrarily many blocks can be pushed at once, is NP-hard with fixed blocks. Bremner, O'Rourke, and Shermer [6] strengthened this result to PSPACE-completeness. Recently, Hoffmann [28] proved that PUSH-$*$ is NP-hard even without fixed blocks, but it remains open whether it is in NP or PSPACE-complete.

Several other results allow only a single block to be pushed at once. In this context, fixed blocks are less crucial because a 2×2 cluster of blocks can never be disturbed. A well-known computer puzzle in this context is *Sokoban*, where the goal is to place each block onto any one of the designated target squares. This puzzle was proved PSPACE-complete by Culberson [10]. A simpler puzzle, called PUSH-1, arises when the goal is simply for the robot to reach a particular position. Demaine, Demaine, and O'Rourke [14] have proved that this puzzle is NP-hard, but it remains open whether it is in NP or PSPACE-complete.

A variation on the PUSH series of puzzles, called PUSHPUSH, is when a block always slides as far as possible when pushed. The NP-hardness of these versions follow from [14, 28]. Another variation, called PUSH-X, disallows the robot from revisiting a square (the robot's path cannot cross). This direction was suggested in [14] because it immediately places the puzzles in NP. Recently, Demaine and Hoffmann [16] proved that PUSH-1X and PUSHPUSH-1X are NP-complete. Hoffmann's reduction for PUSH-* also establishes NP-completeness of PUSH-*X without fixed blocks.

4.4 Clickomania (Same Game)

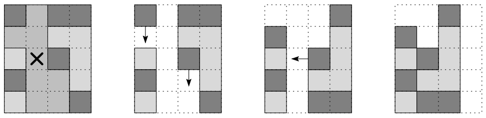


Fig. 7. The falling rules for removing a group in Clickomania. Can you remove all remaining blocks?

Clickomania or *Same Game* [4] is a computer puzzle consisting of a rectangular grid of square blocks each colored one of k colors. Horizontally and vertically adjacent blocks of the same color are considered part of the same *group*. A move selects a group containing at least two blocks and removes those blocks, followed by two "falling" rules; see Fig. 7 (top). First, any blocks remaining above created holes fall down in each column. Second, any empty columns are removed by sliding the succeeding columns left.

The main goal in Clickomania is to remove all the blocks. Biedl et al. [4] proved that deciding whether this is possible is NP-complete. This complexity result holds even for puzzles with two columns and five colors, and for puzzles with five columns and three colors. On the other hand, for puzzles with one column (or, equivalently, one row) and arbitrarily many colors, they show that the maximum number of blocks can be removed in polynomial time. In particular, the puzzles whose blocks can all be removed are given by the context-free grammar $S \to \Lambda \mid SS \mid cSc \mid cScSc$ where c ranges over all colors.

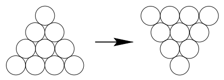
Various cases of Clickomania remain open, for example, puzzles with two colors, and puzzles with $O(1)$ rows. Richard Nowakowski suggested a two-player version of Clickomania, described in [4], in which players take turns removing groups and normal play determines the winner; the complexity of this game remains open.

4.5 Moving Coins

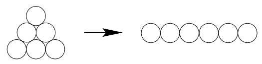
Several coin-sliding and coin-moving puzzles fall into the following general framework: re-arrange one configuration of unit disks in the plane into another configuration by a sequence of moves, each repositioning a coin in an empty position that touches at least two other coins. Examples of such puzzles are shown in Fig. 8. This framework can be further generalized to nongeometric puzzles involving movement of tokens on graphs with adjacency restrictions.

Coin-moving puzzles are analyzed by Demaine, Demaine, and Verrill [15]. In particular, they study puzzles as in Fig. 8 in which the coins' centers remain on either the triangular lattice or the square lattice. Surprisingly, their results for deciding solvability of puzzles are positive.

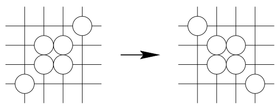
For the triangular lattice, nearly all puzzles are solvable, and there is a polynomial-time algorithm characterizing them. For the square lattice, there are more stringent constraints. For example, the bounding box cannot increase by moves; more generally, the set of positions reachable by moves given an infinite supply of extra coins (the *span*) cannot increase. Demaine, Demaine, and Verrill show that, subject to this constraint, there is a polynomial-time algorithm to solve all puzzles with at least two extra coins past what is required to achieve the span. (In particular, all such puzzles are solvable.)



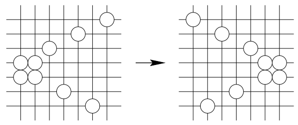
(a) Turn the pyramid upside-down in three moves.



(b) Re-arrange the pyramid into a line in seven moves



(c) Flip the diagonal in 18 moves.



(d) Invert the V in 24 moves.

Fig. 8. Coin-moving puzzles in which each move places a coin adjacent to two other coins; in the bottom two puzzles, the coins must also remain on the square lattice. The top two puzzles are classic, whereas the bottom two puzzles were designed in [15].

5 Cellular Automata and Life

Conway's *Game of Life* is a zero-player cellular automaton played on the square tiling of the plane. Initially, certain cells (squares) are marked *alive* or *dead*. Each move globally evolves the cells: a live cell remains alive if it between 2 and 3 of its 8 neighbors were alive, and a dead cell becomes alive if it had precisely 3 live neighbors. Chapter 25 of *Winning Ways* [3, pp. 817–850] proves that no algorithm can decide whether an initial configuration of Life will ever completely die out. In particular, the same question about Life restricted within a polynomially bounded region is PSPACE-complete. Several other cellular automata, with different survival and birth rules, have been studied; see e.g. [42].

6 Open Problems

Many open problems remain in combinatorial game theory. Guy [26] has compiled a list of such problems (some of which have since been solved). An example of a difficult unsolved problem is Conway's angel-devil game [9].

Many open problems also remain on the algorithmic side, and have been mentioned throughout this paper. Examples of games and puzzles whose complexities remain completely open, to my knowledge, are Toads and Frogs [19], [3, pp. 14–15], Domineering (Section 3.6), and rectangular sliding-block puzzles (Section 4.1). For many other games and puzzles, such as Dots and Boxes (Section 3.7) and pushing-block puzzles (Section 4.3), some hardness results are known, but the exact complexity remains unresolved. More generally, an interesting direction for future research is to build a more comprehensive theory for analyzing combinatorial puzzles.

References

1. E. Berlekamp. *The Dots and Boxes Game: Sophisticated Child's Play.* A. K. Peter's Ltd., 2000.
2. E. Berlekamp and D. Wolfe. *Mathematical Go: Chilling Gets the Last Point.* A. K. Peters, Ltd., 1994.
3. E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways.* Academic Press, London, 1982.
4. T. C. Biedl, E. D. Demaine, M. L. Demaine, R. Fleischer, L. Jacobsen, and J. I. Munro. The complexity of Clickomania. In R. J. Nowakowski, ed., *More Games of No Chance*, 2001. To appear.
5. C. L. Bouton. Nim, a game with a complete mathematical theory. *Ann. of Math. (2)*, 3:35–39, 1901–02.
6. D. Bremner, J. O'Rourke, and T. Shermer. Motion planning amidst movable square blocks is PSPACE complete. Draft, June 1994.
7. M. Buro. Simple Amazons endgames and their connection to Hamilton circuits in cubic subgrid graphs. In *Proc. 2nd Internat. Conf. Computers and Games*, 2000.
8. J. H. Conway. *On Numbers and Games.* Academic Press, London, 1976.
9. J. H. Conway. The angel problem. In R. J. Nowakowski, ed., *Games of No Chance*, pp. 1–12. Cambridge University Press, 1996.
10. J. Culberson. Sokoban is PSPACE-complete. In *Proc. Internat. Conf. Fun with Algorithms*, pp. 65–76, Elba, Italy, June 1998.
11. S. de Vries and R. Vohra. Combinatorial auctions: A survey. Manuscript, Jan. 2001. `http://www-m9.ma.tum.de/~devries/comb_auction_supplement/comauction.pdf`.
12. E. D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. Preprint cs.CC/0106019, Computing Research Repository. `http://www.arXiv.org/abs/cs.CC/0106019`.
13. E. D. Demaine, M. L. Demaine, and D. Eppstein. Phutball endgames are NP-hard. In R. J. Nowakowski, ed., *More Games of No Chance*, 2001. To appear. `http://www.arXiv.org/abs/cs.CC/0008025`.
14. E. D. Demaine, M. L. Demaine, and J. O'Rourke. PushPush and Push-1 are NP-hard in 2D. In *Proc. 12th Canadian Conf. Comput. Geom.*, pp. 211–219, 2000. `http://www.cs.unb.ca/conf/cccg/eProceedings/26.ps.gz`.
15. E. D. Demaine, M. L. Demaine, and H. Verrill. Coin-moving puzzles. In *MSRI Combinatorial Game Theory Research Workshop*, Berkeley, California, July 2000.
16. E. D. Demaine and M. Hoffmann. Pushing blocks is NP-complete for noncrossing solution paths. In *Proc. 13th Canadian Conf. Comput. Geom.*, 2001. To appear.

17. A. Dhagat and J. O'Rourke. Motion planning amidst movable square blocks. In *Proc. 4th Canadian Conf. Comput. Geome.*, pp. 188–191, 1992.
18. D. Eppstein. Computational complexity of games and puzzles. `http://www.ics.uci.edu/~eppstein/cgt/hard.html`.
19. J. Erickson. New Toads and Frogs results. In R. J. Nowakowski, ed., *Games of No Chance*, pp. 299–310. Cambridge University Press, 1996.
20. G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". Manuscript, 2001. `http://www.neci.nj.nec.com/homepages/flake/rushhour.ps`.
21. A. S. Fraenkel. Combinatorial games: Selected bibliography with a succinct gourmet introduction. *Electronic Journal of Combinatorics*, 1994. Dynamic Survey DS2, `http://www.combinatorics.org/Surveys/`.
22. A. S. Fraenkel, M. R. Garey, D. S. Johnson, T. Schaefer, and Y. Yesha. The complexity of checkers on an $N \times N$ board - preliminary report. In *Proc. 19th IEEE Sympos. Found. Comp. Sci.*, pp. 55–64, 1978.
23. A. S. Fraenkel and D. Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n. *J. Combin. Theory Ser. A*, 31:199–214, 1981.
24. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.
25. P. M. Grundy. Mathematics and games. *Eureka*, 2:6–8, Oct. 1939.
26. R. K. Guy. Unsolved problems in combinatorial games. In R. J. Nowakowski, ed., *Games of No Chance*, pp. 475–491. Cambridge University Press, 1996.
27. R. K. Guy and C. A. B. Smith. The G-values of various games. *Proc. Cambridge Philos. Soc.*, 52:514–526, 1956.
28. M. Hoffmann. Push-* is NP-hard. In *Proc. 12th Canadian Conf. Comput. Geom.*, pp. 205–210, 2000. `http://www.cs.unb.ca/conf/cccg/eProceedings/13.ps.gz`.
29. E. Hordern. *Sliding Piece Puzzles.* Oxford University Press, 1986.
30. R. Kaye. Minesweeper is NP-complete. *Math. Intelligencer*, 22(2):9–15, 2000.
31. M. Lachmann, C. Moore, and I. Rapaport. Who wins domineering on rectangular boards? In *MSRI Combinatorial Game Theory Research Workshop*, Berkeley, California, July 2000.
32. D. Lichtenstein and M. Sipser. GO is polynomial-space hard. *J. Assoc. Comput. Mach.*, 27(2):393–401, Apr. 1980.
33. C. H. Papadimitriou. Algorithms, games, and the Internet. In *Proc. 33rd ACM Sympos. Theory Comput.*, Crete, Greece, July 2001.
34. D. Ratner and M. Warmuth. The (n^2-1)-puzzle and related relocation problems. *J. Symbolic Comput.*, 10:111–137, 1990.
35. S. Reisch. Hex ist PSPACE-vollständig. *Acta Inform.*, 15:167–191, 1981.
36. J. M. Robson. The complexity of Go. In *Proceedings of the IFIP 9th World Computer Congress on Information Processing*, pp. 413–417, 1983.
37. J. M. Robson. N by N Checkers is EXPTIME complete. *SIAM J. Comput.*, 13(2):252–267, May 1984.
38. C. A. B. Smith. Graphs and composite games. *J. Combin. Theory*, 1:51–81, 1966.
39. R. Sprague. Über mathematische Kampfspiele. *Tôhoku Mathematical Journal*, 41:438–444, 1935–36.
40. G. Wilfong. Motion planning in the presence of movable obstacles. *Ann. Math. Artificial Intelligence*, 3(1):131–150, 1991.
41. D. Wolfe. Go endgames are PSPACE-hard. In R. J. Nowakowski, ed., *More Games of No Chance*, 2001. To appear.
42. S. Wolfram. *Cellular Automata and Complexity: Collected Papers.* Perseus Press, 1994.