Playing Games with Algorithms: Algorithmic Combinatorial Game Theory*

Erik D. Demaine† Robert A. Hearn‡

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, and Constraint Logic, which provides a framework for showing hardness. 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 (assuming P $\neq$ NP): one-player puzzles are often NP-complete (as in Minesweeper) or PSPACE-complete (as in Rush Hour), 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.

We begin by reviewing some basics of Combinatorial Game Theory in Section 2, which gives tools for designing algorithms, followed by reviewing the relatively new theory of Constraint Logic in Section 3, which gives tools for proving hardness. In the bulk of this paper, Sections 4–6 survey many of the algorithmic and hardness results for combinatorial games and puzzles. Section 7 concludes with a small sample of difficult open problems in algorithmic Combinatorial Game Theory.

Combinatorial Game Theory is to be distinguished from other forms of game theory arising in the context of economics. Economic game theory has many applications in computer science as well, for example, in the context of auctions [dVV03] and analyzing behavior on the Internet [Pap01].

---

*A preliminary version of this paper appears in the *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 2136, Czech Republic, August 2001, pages 18–32. The latest version can be found at http://arXiv.org/abs/cs.CC/0106019.

†MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, edemaine@mit.edu

‡Neukom Institute for Computational Sciece, Dartmouth College, Sudikoff Hall, HB 6255, Hanover, NH 03755, USA, robert.a.hearn@dartmouth.edu

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 [BCG04, pp. 14–15]. 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. A new introductory book on the topic is *Lessons in Play* by Albert, Nowakowski, and Wolfe [ANW07]; the most comprehensive reference is the book *Winning Ways* by Berlekamp, Conway, and Guy [BCG04]; and a more mathematical presentation is the book *On Numbers and Games* by Conway [Con01]. See also [Con77, Fra96] for overviews and [Fra07] for a bibliography. The basic idea behind the theory is simple: a two-player game can be described by a rooted tree, where each node has zero or more *left* branches corresponding to options for player Left to move and zero or more *right* branches corresponding to options for player Right to move; leaves correspond to finished games, with the winner 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*[1] [Con01, Knu74, Gon86, All87], 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 [Con01] for more formal definitions.

For example, the simplest number without any larger-than or smaller-than constraints, denoted

[1]The name "surreal numbers" is actually due to Knuth [Knu74]; see [Con01].

$\{|\}$, is 0; the simplest number larger than 0 and without smaller-than constraints, denoted $\{0\ |\}$, is 1; and the simplest number larger than 0 and 1 (or just 1), denoted $\{0, 1\ |\}$, is 2. 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 $\{0 \mid 1\}$, 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.

Surreal numbers form a field, so in particular they are totally ordered, and support the operations of addition, subtraction, multiplication, division, roots, powers, and even integration in many situations. (For those familiar with ordinals, contrast with surreals which define $\omega - 1$, $1/\omega$, $\sqrt{\omega}$, etc.) As such, surreal numbers are useful in their own right for cleaner forms of analysis; see, e.g., [All87].

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; see Table 1 for formal definitions. Two-player combinatorial games, or trees, can simply be represented as $\{L \mid R\}$ 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. (Numeric) equality is defined by being both less than or equal to and more than or equal to. Strictly inequalities, as used in the definition of less than or equal to, are defined in the obvious manner.

Note that while $\{-1 \mid 1\} = 0 = \{|\}$ in terms of numbers, $\{-1 \mid 1\}$ 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 $\{-1 \mid 1\}$ 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). Examples of zero, positive, and negative games are the surreal numbers themselves; an additional example is described below.

There is one outcome not captured by the characterization into zero, positive, and negative games: the first player to move wins. To find such a game we must obviously look beyond the surreal numbers. Furthermore, we must look for games G that are incomparable with zero (none of $G = 0$, $G < 0$, or $G > 0$ hold); such games are called *fuzzy* with 0, denoted $G \parallel 0$.

An example of a game that is not a surreal number is $\{1 \mid 0\}$; there fails to be a number strictly between 1 and 0 because $1 \geq 0$. Nonetheless, $\{1 \mid 0\}$ is a game: Left has a single move leading to game 1, from which Right cannot move, and Right has a single move leading to game 0, from which

Let $x = \{x^L \mid x^R\}$ be a game. • $x \leq y$ precisely if every $x^L < y$ and every $y^R > x$. • $x = y$ precisely if $x \leq y$ and $x \geq y$; otherwise $x \neq y$. • $x < y$ precisely if $x \leq y$ and $x \neq y$, or equivalently, $x \leq y$ and $x \ngeq y$. • $-x = \{-x^R \mid -x^L\}$. • $x + y = \{x^L + y, x + y^L \mid x^R + y, x + y^R\}$. • x is *impartial* precisely if x^L and x^R are identical sets and recursively every position ($\in x^L = x^R$) is impartial. • A one-pile Nim game is defined by $*n = \{*0, \ldots, *(n-1) \mid *0, \ldots, *(n-1)\}$, together with $*0 = 0$.

Table 1: Formal definitions of some algebra on two-player perfect-information games. In particular, all of these notions apply to surreal numbers.

Left cannot move. Thus, in either case, the first player to move wins. The claim above implies that $\{1 \mid 0\} \parallel 0$. Indeed, $\{1 \mid 0\} \parallel x$ for all surreal numbers x, $0 \leq x \leq 1$. In contrast, $x < \{1 \mid 0\}$ for all $x < 0$ and $\{1 \mid 0\} < x$ 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. Another example of a game that is not a number is $\{2 \mid 1\}$, which is positive (> 0), and hence Right wins, but fuzzy with numbers in the range $[1, 2]$.

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 [BCG04, Con01].

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 [Spr36] and Grundy [Gru39]. Recall that a game is *impartial* if it does not distinguish between the players Left and Right (see Table 1 for a more formal definition). The Sprague-Grundy theory [Spr36, Gru39, Con01, BCG04] 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 [Con01, Smi66].

Nim [Bou02] 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 $*0$, $*1$, …, $*(n-1)$. 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, …, n_k is equivalent to a one-pile Nim game $*n$, where n is the binary XOR of n_1, n_2, …, 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 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 [Con01, Smi66].

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 [GS56b]; and Cutcake and Maundy Cake [BCG04, pp. 24–27]. 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.

The Sprague-Grundy theory seems difficult to generalize to the superficially similar case of misère play, where the goal is to be the first player unable to move. Certain games have been solved in this context over the years, including Nim [Bou02]; see, e.g., [Fer74, GS56a]. Recently a general theory has emerged for tackling misère combinatorial games, based on commutative monoids called "misère quotients" that localize the problem to certain restricted game scenarios. This theory was introduced by Plambeck [Pla05] and further developed by Plambeck and Siegel [PS07]. For good descriptions of the theory, see Plambeck's survey [Plaa], Siegel's lecture notes [Sie06], and a webpage devoted to the topic [Plab].

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 4.1. 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 4.1, 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 the 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 as a tree, similar to a two-player game except that edges are not distinguished between Left and Right. With the view that the game ends only when the puzzle is solved, the goal is then 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, and losing terminal positions can be given explicit loops to themselves.

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.

3 Constraint Logic

Combinatorial Game Theory provides a theoretical framework for giving positive algorithmic results for games, but does not naturally accommodate puzzles. In contrast, negative algorithmic results—hardness and completeness within computational complexity classes—are more uniform: puzzles and games have analogous prototypical proof structures. Furthermore, a relatively new theory called Constraint Logic attempts to tie together a wide range of hardness proofs for both puzzles and games.

Proving that a problem is hard within a particular complexity class (like NP, PSPACE, or EXPTIME) almost always involves a reduction to the problem from a known hard problem within the class. For example, the canonical problem to reduce from for NP-hardness is Boolean Satisfiability (SAT) [Coo71]. Reducing SAT to a puzzle of interest proves that that puzzle is NP-hard. Similarly, the canonical problem to reduce from for PSPACE-hardness is Quantified Boolean Formulas (QBF) [SM73].

Constraint Logic [DH08] is a useful tool for showing hardness of games and puzzles in a variety of settings that has emerged in recent years. Indeed, many of the hardness results mentioned in this survey are based on reductions from Constraint Logic. Constraint Logic is a family of games where players reverse edges on a planar directed graph while satisfying vertex in-flow constraints. Each edge has a weight of 1 or 2. Each vertex has degree 3 and requires that the sum of the weights of inward-directed edges is at least 2. Vertices may be restricted to two types: AND vertices have incident edge weights of 1, 1, and 2; and OR vertices have incident edge weights of 2, 2, and 2. A player's goal is to eventually reverse a given edge.

This game family can be interpreted in many game-theoretic settings, ranging from zero-player automata to multiplayer games with hidden information. In particular, there are natural versions of Constraint Logic corresponding to one-player games (puzzles) and two-player games, both of bounded and unbounded length. (Here we refer to whether the length of the game is bounded by a polynomial function of the board size. Typically, bounded games are nonloopy while unbounded games are loopy.) These games have the expected complexities: one-player bounded games are NP-complete; one-player unbounded games and two-player bounded games are PSPACE-complete; and two-player unbounded games are EXPTIME-complete.

What makes Constraint Logic specially suited for game and puzzle reductions is that the problems are already in form similar to many games. In particular, the fact that the games are played on planar graphs means that the reduction does not usually need a crossover gadget, whereas historically crossover gadgets have often been the complex crux of a game hardness proof.

Historically, Constraint Logic arose as a simplification of the "Generalized Rush-Hour Logic" of Flake and Baum [FB02]. The resulting one-player unbounded setting, called *Nondeterministic Constraint Logic* [HD02, HD05], was later generalized to other game categories [Hea06b, DH08].

4 Algorithms for Two-Player Games

Many bounded-length 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, unbounded-length 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 many of these complexity results and related positive results. See also [Epp] for a related survey and [Fra07] for a bibliography.

4.1 Hex

Hex [BCG04, pp. 743–744] is a game designed by Piet Hein and played on a diamond-shaped hexagonal board; see Figure 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.

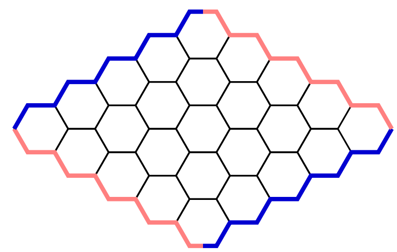


Figure 1: A 5×5 Hex board.

John Nash [BCG04, p. 744] proved that the first player to move can win by using a strategy-stealing argument (see Section 2.3). Suppose that the second player has a winning strategy, and assume by symmetry that Left goes first. Left selects the first hexagon arbitrarily. Now Right is to move first and Left is effectively the second player. Thus, Left can follow the winning strategy for the second player, except that Left has one additional hexagon. But this additional hexagon can only help Left: it only restricts Right's moves, and if Left's strategy suggests filling the additional hexagon, Left can instead move anywhere else. Thus, Left has a winning strategy, contradicting that Right did, and hence the first player has a winning strategy. However, it remains open to give a polynomial characterization of a winning strategy for the first player.

In perhaps the first PSPACE-hardness result for "interesting" games, Even and Tarjan [ET76] proved that a generalization of Hex to graphs is PSPACE-complete, even for maximum-degree-5 graphs. Specifically, in this graph game, two vertices are initially colored Left, and players take turns coloring uncolored vertices in their own color. Left's goal is to connect the two initially Left vertices by a path, and Right's goal is to prevent such a path. Surprisingly, the closely related problem in which players color *edges* instead of vertices can be solved in polynomial time; this game is known as the *Shannon switching game* [BW70]. A special case of this game is *Bridgit* or *Gale*, invented by David Gale [BCG04, p. 744], in which the graph is a square grid and Left's goal is to connect a vertex on the top row with a vertex on the bottom row. However, if the graph in Shannon's switching game has directed edges, the game again becomes PSPACE-complete [ET76].

A few years later, Reisch [Rei81] proved the stronger result that determining the outcome of a position in Hex is PSPACE-complete on a normal diamond-shaped board. The proof is quite different from the general graph reduction of Even and Tarjan [ET76], but the main milestone is to prove that Hex is PSPACE-complete for planar graphs.

4.2 More Games on Graphs: Kayles, Snort, Geography, Peek, and Interactive Hamiltonicity

The second paper to prove PSPACE-hardness of "interesting" games is by Schaefer [Sch78]. This work proposes over a dozen games and proves them PSPACE-complete. Some of the games involve propositional formulas, others involve collections of sets, but perhaps the most interesting are those involving graphs. Two of these games are generalizations of "Kayles", and another is a graph-traversal game called Edge Geography.

Kayles [BCG04, pp. 81–82] is an impartial game, designed independently by Dudeney and Sam Loyd, in which bowling pins are lined up on a line. Players take turns *bowling* with the property that exactly one or exactly two adjacent pins are knocked down (removed) in each move. Thus, most moves split the game into a sum of two subgames. Under normal play, Kayles can be solved in polynomial time using the Sprague-Grundy theory; see [BCG04, pp. 90–91], [GS56b].

Node Kayles is a generalization of Kayles to graphs in which each bowl "knocks down" (removes) a desired vertex and all its neighboring vertices. (Alternatively, this game can be viewed as two players finding an independent set.) Schaefer [Sch78] proved that deciding the outcome of this game is PSPACE-complete. The same result holds for a partizan version of node Kayles, in which every node is colored either Left or Right and only the corresponding player can choose a particular node as the primary target.

Geography is another graph game, or rather game family, that is special from a techniques point of view: it has been used as the basis of many other PSPACE-hardness reductions for games described in this section. The motivating example of the game is players taking turns naming distinct geographic locations, each starting with the same letter with which the previous name ended. More generally, Geography consists of a directed graph with one node initially containing a token. Players take turns moving the token along a directed edge. In *Edge Geography*, that edge is then erased; in *Vertex Geography*, the vertex moved from is then erased. (Confusingly, in the literature, each of these variants is frequently referred to as simply "Geography" or "Generalized Geography".)

Schaefer [Sch78] established that Edge Geography (a game suggested by R. M. Karp) is PSPACE-complete; Lichtenstein and Sipser [LS80] showed that Vertex Geography (which more closely matches the motivating example above) is also PSPACE-complete. Nowakowski and Poole [NP96] have solved special cases of Vertex Geography when the graph is a product of two cycles.

One may also consider playing either Geography game on an undirected graph. Fraenkel, Scheinerman, and Ullman [FSU93] show that Undirected Vertex Geography can be solved in polynomial time, whereas Undirected Edge Geography is PSPACE-complete, even for planar graphs with maximum degree 3. If the graph is bipartite then Undirected Edge Geography is also solvable in polynomial time.

One consequence of partizan node Kayles being PSPACE-hard is that deciding the outcome in Snort is PSPACE-complete on general graphs [Sch78]. *Snort* [BCG04, pp. 145–147] is a game designed by S. Norton and normally played on planar graphs (or planar maps). In any case, players take turns coloring vertices (or faces) in their own color such that only equal colors are adjacent.

Generalized hex (the vertex Shannon switching game), node Kayles, and Vertex Geography have also been analyzed recently in the context of parameterized complexity. Specifically, the problem of deciding whether the first player can win within k moves, where k is a parameter to the problem, is AW[$*$]-complete [DF97, ch. 14].

Stockmeyer and Chandra [SC79] were the first to prove combinatorial games to be EXPTIME-hard. They established EXPTIME-completeness for a class of logic games and two graph games. Here we describe an example of a logic game in the class, and one of the graph games; the other graph game is described in the next section. One logic game, called Peek, involves a box containing several parallel rectangular plates. Each plate (1) is colored either Left or Right except for one ownerless plate, (2) has circular holes carved in particular (known) positions, and (3) can be slid to one of two positions (fully in the box or partially outside the box). Players take turns either passing or changing the position of one of their plates. The winner is the first player to cause a hole in every plate to be aligned along a common vertical line. A second game involves a graph in which some edges are colored Left and some edges are colored Right, and initially some edges are "in" while the others are "out". Players take turns either passing or changing one edge from "out" to "in" or vice versa. The winner is the first player to cause the graph of "in" edges to have a Hamiltonian cycle. (Both of these games can be rephrased under normal play by defining there to be no valid moves from positions having aligned holes or Hamiltonian cycles.)

4.3 Games of Pursuit: Annihilation, Remove, Capture, Contrajunctive, Blocking, Target, and Cops and Robbers

The next suite of graph games essentially began study in 1976 when Fraenkel and Yesha [FY76] announced that a certain impartial annihilation game could be played optimally in polynomial time. Details appeared later in [FY82]; see also [Fra74]. The game was proposed by John Conway and is played on an arbitrary directed graph in which some of the vertices contain a token. Players take turns selecting a token and moving it along an edge; if this causes the token to occupy a vertex already containing a token, both tokens are *annihilated* (removed). The winner is determined by normal play if all tokens are annihilated, except that play may be drawn out indefinitely. Fraenkel and Yesha's result [FY82] is that the outcome of the game can be determined and (in the case of a winner) a winning strategy of $O(n^5)$ moves can be computed in $O(n^6)$ time, where n is the number of vertices in the graph.

A generalization of this impartial game, called *Annihilation*, is when two (or more) types of tokens are distinguished, and each type of token can travel along only a certain subset of the edges. As before, if a token is moved to a vertex containing a token (of any type), both tokens are annihilated. Determining the outcome of this game was proved NP-hard [FY79] and later PSPACE-hard [FG87]. For acyclic graphs, the problem is PSPACE-complete [FG87]. The precise complexity for cyclic graphs remains open. Annihilation has also been studied under misère play [Fer84].

A related impartial game, called *Remove*, has the same rules as Annihilation except that when a token is moved to a vertex containing another token, only the moved token is removed. This game was also proved NP-hard using a reduction similar to that for Annihilation [FY79], but otherwise its complexity seems open. The analogous impartial game in which just the *unmoved* token is removed, called *Hit*, is PSPACE-complete for acyclic graphs [FG87], but its precise complexity remains open for cyclic graphs.

A partizan version of Annihilation is *Capture*, in which the two types of tokens are assigned to corresponding players. Left can only move a Left token, and only to a position that does not contain a Left token. If the position contains a Right token, that Right token is *captured* (removed). Unlike Annihilation, Capture allows all tokens to travel along all edges. Determining the outcome of Capture was proved NP-hard [FY79] and later EXPTIME-complete [GR95]. For acyclic graphs the game is PSPACE-complete [GR95].

A different partizan version of Annihilation is *Contrajunctive*, in which players can move both types of tokens, but each player can use only a certain subset of the edges. This game is NP-hard even for acyclic graphs [FY79] but otherwise its complexity seems open.

The *Blocking* variations of Annihilation disallow a token to be moved to a vertex containing another token. Both variations are partizan and played with tokens on directed graph. In *Node Blocking*, each token is assigned to one of the two players, and all tokens can travel along all edges. Determining the outcome of this game was proved NP-hard [FY79], then PSPACE-hard [FG87], and finally EXPTIME-complete [GR95]. Its status for acyclic graphs remains open. In *Edge Blocking*, there is only one type of token, but each player can use only a subset of the edges. Determining the outcome of this game is PSPACE-complete for acyclic graphs [FG87]. Its precise complexity for general graphs remains open.

A generalization of Node Blocking is *Target*, in which some nodes are marked as *targets* for each player, and players can additionally win by moving one of their tokens to a vertex that is one of their targets. When no nodes are marked as targets, the game is the same as Blocking and hence EXPTIME-complete by [GR95]. In fact, general Target was proved EXPTIME-complete earlier by Stockmeyer and Chandra [SC79]. Surprisingly, even the special case in which the graph is acyclic and bipartite and only one player has targets is PSPACE-complete [GR95]. (NP-hardness of this case was established earlier [FY79].)

A variation on Target is *Semi-Partizan Target*, in which both players can move all tokens, yet Left wins if a Left token reaches a Left target, independent of who moved the token there. In addition, if a token is moved to a nontarget vertex containing another token, the two tokens are annihilated. This game is EXPTIME-complete [GR95]. While this game may seem less natural than the others, it was intended as a step towards the resolution of Annihilation.

Many of the results described above from [GR95] are based on analysis of a more complex game called *Pursuit* or *Cops and Robbers*. One player, the *robber*, has a single token; and the other player, the *cops*, have k tokens. Players take turns moving all of their tokens along edges in a directed graph. The cops win if at the end of any move the robber occupies the same vertex as a cop, and the robber wins if play can be forced to draw out forever. In the case of a single cop ($k = 1$), there is a simple polynomial-time algorithm, and in general, many versions of the game are EXPTIME-complete; see [GR95] for a summary. For example, EXPTIME-completeness holds even for undirected graphs, and for directed graphs in which cops and robbers can choose their initial positions. For acyclic graphs, Pursuit is PSPACE-complete [GR95].

4.4 Checkers (Draughts)

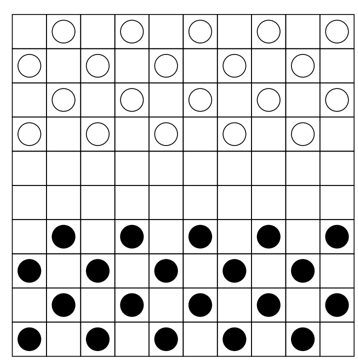


Figure 2: A natural starting configuration for 10×10 Checkers, from [FGJ+78].

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). Indeed, Schaeffer et al. [SBB+07] recently computed that optimal play leads to a draw from the initial configuration (other configurations remain unanalyzed). The outcome of playing in a general $n \times n$ board from a natural starting position, such as the one in Figure 2, remains open. On the other hand, deciding the outcome of an arbitrary configuration is PSPACE-hard [FGJ+78]. 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 [FGJ+78]), then the problem is in PSPACE and hence is PSPACE-complete. Without such a restriction, however, Checkers is EXPTIME-complete [Rob84b].

On the other hand, certain simple questions about Checkers can be answered in polynomial time [FGJ+78, DDE02]. 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 [FGJ+78, DDE02]. However, for boards defined by general graphs, at least the first question becomes NP-complete [FGJ+78].

4.5 Go

Presented at the same conference as the Checkers result in the previous section (FOCS'78), Lichtenstein and Sipser [LS80] proved that the classic Asian game of Go is also PSPACE-hard for an arbitrary configuration on an $n \times n$ board. Go has few rules: (1) players take turns either passing or placing stones of their color on positions on the board; (2) if a new black stone (say) causes a collection of white stones to be completely surrounded by black stones, the white stones are removed; and (3) a ko rule preventing repeated configurations. Depending on the country, there are several variations of the ko rule; see [BW94]. Go does not follow normal play: the winner in Go is the player with the highest score at the end of the game. A player's score is counted as either the number of stones of his color on the board plus empty spaces surrounded by his stones (area counting), or as empty spaces surrounded by his stones plus captured stones (territory counting), again varying by country.

The PSPACE-hardness proof of Lichtenstein and Sipser [LS80] does not involve any situations called kos, where the ko rule must be invoked to avoid infinite play. In contrast, Robson [Rob83] proved that Go is EXPTIME-complete under Japanese rules when kos are involved, and indeed used judiciously. The type of ko used in this reduction is shown in Figure 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.

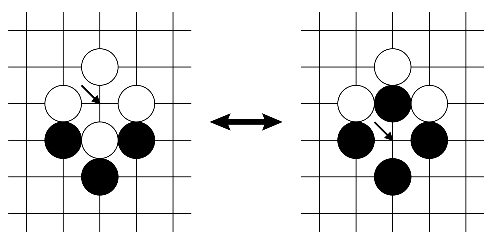


Figure 3: A simple form of ko in Go.

Robson's proof relies on properties of the Japanese rules for both the upper and lower bounds. For other rulesets, all that is known is that Go is PSPACE-hard and in EXPSPACE. In particular, the "superko" variant of the ko rule (as used in, e.g., the U.S.A. and New Zealand), which prohibits recreation of any former board position, suggests EXPSPACE-hardness, by a result of Robson for no-repeat games [Rob84a]. However, if all dynamical state in the game occurs in kos, as it does in the EXPTIME-hardness construction, then the game is still in EXPTIME, because then it is an instance of Undirected Vertex Geography (Section 4.2), which can be solved in time polynomial in the graph size. (In this case the graph is all the possible game positions, of which there are exponentially many.)

There are also several results for more restricted Go positions. Wolfe [Wol02] shows 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 [BW94]. Crâşmaru and Tromp [CT00] show that it is PSPACE-complete to determine whether a ladder (a repeated pattern of capture threats) results in a capture. Finally, Crâşmaru [Crâ99] shows that it is NP-complete to determine the status of certain restricted forms of life-and-death problems in Go.

4.6 Five-in-a-Row (Gobang)

Five-in-a-Row or *Gobang* [BCG04, pp. 738–740] is another game on a Go board in which players take turns placing a stone of their color. Now the goal of the players is to place at least 5 stones of their color in a row either horizontally, vertically, or diagonally. This game is similar to Go-Moku [BCG04, p. 740], which does not count 6 or more stones in a row, and imposes additional constraints on moves.

Reisch [Rei80] proved that deciding the outcome of a Gobang position is PSPACE-complete. He also observed that the reduction can be adapted to the rules of k-in-a-Row for fixed k. Although he did not specify exactly which values of k are allowed, the reduction would appear to generalize to any $k \geq 5$.

4.7 Chess

Fraenkel and Lichtenstein [FL81] 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.

4.8 Shogi

Shogi is a Japanese game along lines similar to Chess, but with rules too complex to state here. Adachi, Kamekawa, and Iwata [AKI87] proved that deciding the outcome of a Shogi position is EXPTIME-complete. Recently, Yokota et al. [YTK$^+$01] proved that a more restricted form of Shogi, *Tsume-Shogi*, in which the first player must continually make oh-te (the equivalent of check in Chess), is also EXPTIME-complete.

4.9 Othello (Reversi)

Othello (*Reversi*) is a classic game on an 8×8 board, starting from the initial configuration shown in Figure 4, in which players alternately place pieces of their color in unoccupied squares. Moves are restricted to cause, in at least one row, column, or diagonal, a consecutive sequence of pieces of the opposite color to be enclosed by two pieces of the current player's color. As a result of the move, the enclosed pieces "flip" color into the current player's color. The winner is the player with the most pieces of their color when the board is filled.

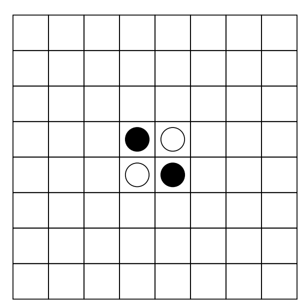


Figure 4: Initial configuration in Othello.

Generalized to an $n \times n$ board with an arbitrary initial configuration, the game is clearly in PSPACE because only $n^2 - 4$ moves can be made. Furthermore, Iwata and Kasai [IK94] proved that the game is PSPACE-complete.

4.10 Hackenbush

Hackenbush is one of the standard examples of a combinatorial game in *Winning Ways*; see, e.g., [BCG04, pp. 1–6]. 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* [BCG04, pp. 189–227] proves that determining the *value* of a red-blue Hackenbush position is NP-hard. The reduction is from minimum Steiner tree in graphs. It applies to a restricted form of hackenbush positions, called *redwood beds*, consisting of a red bipartite graph, with each vertex on one side attached to a red edge, whose other end is attached to a blue edge, whose other end is rooted.

4.11 Domineering (Crosscram) and Cram

Domineering or *crosscram* [BCG04, pp. 119–126] 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 [LMR00] 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.

Cram [Gar86], [BCG04, pp. 502–506] is the impartial version of Domineering in which both players can place horizontal and vertical dominoes. The outcome of Cram is easy to determine for rectangles having an even number of squares [Gar86]: if both sides are even, the second player can win by a symmetry strategy (reflecting the first player's move through both axes); and if precisely one side is even, the first player can win by playing the middle two squares and then applying the symmetry strategy. It seems open to determine the outcome for a rectangle having two odd sides. The complexity of Cram for general boards also remains open.

Linear Cram is Cram in a $1 \times n$ rectangle, where the game quickly splits into a sum of games. This game can be solved easily by applying the Sprague-Grundy theory and dynamic programming; in fact, there is a simpler solution based on proving that its behavior is periodic in n [GS56b]. The variation on Linear Cram in which $1 \times k$ rectangles are placed instead of dominoes can also be solved via dynamic programming, but whether the behavior is periodic remains open even for $k = 3$ [GS56b]. Misère Linear Cram also remains unsolved [Gar86].

4.12 Dots-and-Boxes, Strings-and-Coins, and Nimstring

Dots-and-Boxes 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. Most of this section is based on Chapter 16 of *Winning Ways* [BCG04, pp. 541–584]; another good reference is a recent book by Berlekamp [Ber00].

Gameplay in Dots-and-Boxes typically divides into two phases: the *opening* during which no boxes are enclosed, and the *endgame* during which boxes are enclosed in nearly every move; see

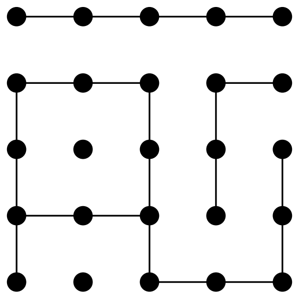


Figure 5: A Dots-and-Boxes endgame.

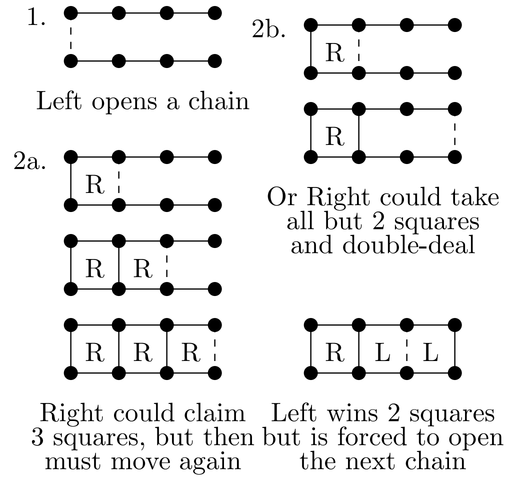


Figure 6: Chains and double-dealing in Dots-and-Boxes.

Figure 5. In the endgame, the "free move" awarded by enclosing a square often leads to several squares enclosed in a single move, following a *chain*; see Figure 6. Most children apply the greedy algorithm of taking the most squares possible, and thus play entire chains of squares. However, this strategy forces the player to open another chain (in the endgame). A simple improved strategy is called *double dealing*, which forfeits the last two squares of the chain, but forces the opponent to open the next chain. The double-dealer is said to *remain in control*; if there are long-enough chains, this player will win (see [BCG04, p. 543] for a formalization of this statement).

A generalization arising from the dual of Dots-and-Boxes is *Strings-and-Coins* [BCG04, pp. 550–551]. 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.

Another game closely related to Dots-and-Boxes is *Nimstring* [BCG04, pp. 552–554], which has the same rules as Strings-and-Coins, except that the winner is determined by normal play. Nimstring is in fact a special case of Strings-and-Coins [BCG04, p. 552]: if we add a chain of more than $n + 1$ coins to an instance of Nimstring having n coins, then ideal play of the resulting string-and-coins instance will avoid opening the long chain for as long as possible, and thus the player to move last in the Nimstring instance wins string and coins.

Winning Ways [BCG04, pp. 577–578] argues that Strings-and-Coins is NP-hard as follows. Suppose that you have gathered several coins but your opponent gains control. Now you are forced to lose the Nimstring game, but given your initial lead, you still may win the Strings-and-Coins game. Minimizing the number of coins lost while your opponent maintains control is equivalent to finding the maximum number of vertex-disjoint cycles in the graph, basically because the equivalent of a double-deal to maintain control once an (isolated) cycle is opened results in forfeiting four squares instead of two. We observe that by making the difference between the initial lead and the forfeited coins very small (either -1 or 1), the opponent also cannot win by yielding control. Because the cycle-packing problem is NP-hard on general graphs, determining the outcome of such string-and-coins endgames is NP-hard. Eppstein [Epp] observes that this reduction should also apply to endgame instances of Dots-and-Boxes by restricting to maximum-degree-three planar graphs. Embeddability of such graphs in the square grid follows because long chains and cycles (longer than two edges for chains and three edges for cycles) can be replaced by even longer chains or cycles [BCG04, p. 561].

It remains open whether Dots-and-Boxes or Strings-and-Coins are in NP or PSPACE-complete from an arbitrary configuration. Even the case of a $1 \times n$ grid of boxes is not fully understood from a Combinatorial Game Theory perspective [GN02].

4.13 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 Figure 7 (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.

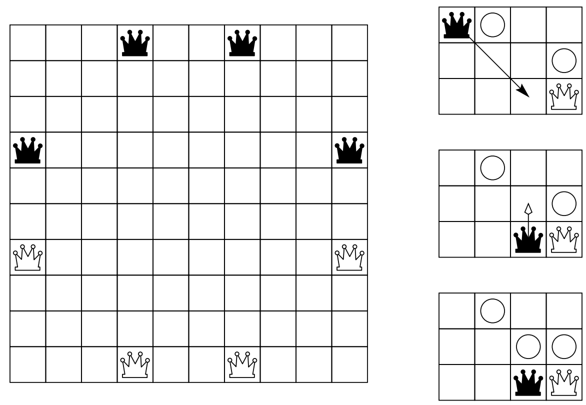


Figure 7: The initial position in Amazons (left) and an example of black trapping a white amazon (right).

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 [Bur00] 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 [Bur00] proved that this problem is *NP-equivalent* [GJ79], that is, the problem can be solved by a polynomial number of calls to an algorithm for any NP-complete problem, and vice versa.

Like Conway's Angel Problem (Section 4.16), the complexity of deciding the outcome of a general Amazons position remained open for several years, only to be solved nearly simultaneously by multiple people. Furtak, Kiyomi, Takeaki, and Buro [FKUB05] give two independent proofs of PSPACE-completeness: one a reduction from Hex, and the other a reduction from Vertex Geography. The latter reduction applies even for positions containing only a single black and a single white amazon. Independently, Hearn [Hea05a, Hea06b, Hea08a] gave a Constraint Logic reduction showing PSPACE-completeness.

4.14 Konane

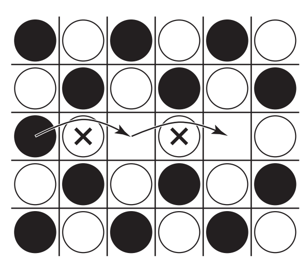


Figure 8: One move in Konane consisting of two jumps.

Konane, or Hawaiian Checkers, is a game that has been played in Hawaii since preliterate times. Konane is played on a rectangular board (typically ranging in size from 8×8 to 13×20) which is initially filled with black and white stones in a checkerboard pattern. To begin the game, two adjacent stones in the middle of the board or in a corner are removed. Then, the players alternate making moves. Moves are made as in Peg Solitaire (Section 5.10); indeed, Konane may be thought of as a kind of two-player peg solitaire. A player moves a stone of his color by jumping it over a horizontally or vertically adjacent stone of he opposite color, into an empty space. (See Figure 8.) Jumped stones are captured and removed from play. A stone may make multiple successive jumps in a single move, so long as they are in a straight line; no turns are allowed within a single move. The first player unable to move wins.

Hearn proved that Konane is PSPACE-complete [Hea06b, Hea08a] by a reduction from Constraint Logic. There have been some positive results for restricted configurations. Ernst [Ern95] derives Combinatorial-Game-Theoretic values for several interesting positions. Chan and Tsai [CT02] analyze the $1 \times n$ game, but even this version of the game is not yet solved.

4.15 Phutball

Conway's game of *Philosopher's Football* or *Phutball* [BCG04, pp. 752–755] 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 Figure 9. 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.

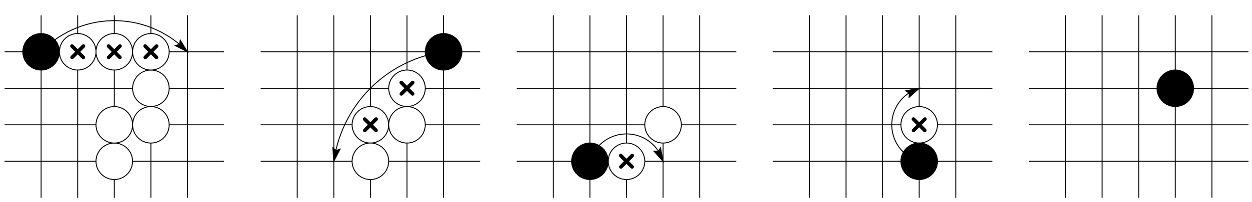


Figure 9: A single move in Phutball consisting of four jumps.

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 [DDE02] 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.

4.16 Conway's Angel Problem

A formerly long-standing open problem was Conway's *Angel Problem* [BCG04]. Two players, the Angel and the Devil, alternate play on an infinite square grid. The Angel can move to any valid position within k horizontal distance and k vertical distance from its present position. The Devil can teleport to an arbitrary square other than where the Angel is and "eat" that square, preventing the Angel from landing on (but not leaping over) that square in the future. The Devil's goal is to prevent the Angel from moving.

It was long known that an Angel of power $k = 1$ can be stopped [BCG04], so the Devil wins, but the Angel was not known to be able to escape for any $k > 1$. (In the original open problem statement, $k = 1000$.) Recently, four independent proofs established that a sufficiently strong Angel can move forever, securing the Angel as the winner. Máthé [Mát07] and Kloster [Klo07] showed that $k = 2$ suffices; Bowditch [Bow07] showed that $k = 4$ suffices; and Gács [Gác07] showed that some k suffices. In particular, Kloster's proof gives an explicit algorithmic winning strategy for the $k = 2$ Angel.

4.17 Jenga

Jenga is a popular stacked-block game invented by Leslie Scott in the 1970s and now marketed by Hasbro. Two players alternate moving individual blocks in a tower of blocks, and the first player to topple the tower (or cause any additional blocks to fall) loses. Each block is $1 \times 1 \times 3$ and lies horizontally. The initial $3 \times 3 \times n$ tower alternates levels of three blocks each, so that blocks in adjacent levels are orthogonal. (In the commercial game, $n = 18$.) In each move, the player removes any block that is below the topmost complete (3-block) level, then places that block in the topmost level (starting a new level if the existing topmost level is complete), orthogonal to the blocks in the (complete) level below. The player loses if the tower becomes instable, that is, the center of gravity of the top k levels projects outside the convex hull of the contact area between the kth and $(k+1)$st layer.

Zwick [Zwi02] proved that the physical stability condition of Jenga can be restated combinatorially simply by constraining allowable patterns on each level and the topmost three levels. Specifically, write a 3-bit vector to specify which blocks are present in each level. Then a tower is stable if and only if no level except possibly the top is 100 or 001 and the three topmost levels from bottom to top are none of $010, 010, 100$; $010, 010, 001$; $011, 010, 100$; or $110, 010, 001$. Using this characterization, Zwick proves that the first player wins from the initial configuration if and only if $n = 2$ or $n \geq 4$ and $n \equiv 1$ or $2 \pmod 3$, and gives a simple characterization of winning moves. It remains open whether such an efficient solution can be obtained in the generalization to odd numbers $k > 3$ of blocks in each level. (The case of even k is a second-player win by a simple mirror strategy.)

5 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, PSPACE-hard. Usually such puzzles occupy a bounded board or region, so they are also PSPACE-complete. A common method to prove that such puzzles are in PSPACE is to give a simple low-space nondeterministic algorithm that guesses the solution, and apply Savitch's theorem [Sav70] that PSPACE = NPSPACE (nondeterministic polynomial space). 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.

5.1 Instant Insanity

Given n cubes, each face colored one of n colors, is it possible to stack the cubes so that each color appears exactly once on each of the 4 sides of the stack? The case of $n = 4$ is a puzzle called Instant Insanity distributed by Parker Bros. In one of the first papers on hardness of puzzles and games people play, Robertson and Munro [RM78] proved that this *generalized Instant Insanity* problem is NP-complete.

The *cube stacking game* is a two-player game based on this puzzle. Given an ordered list of cubes, the players take turns adding the next cube to the top of the stack with a chosen orientation. The loser is the first player to add a cube that causes one of the four sides of the stack to have a color repeated more than once. Robertson and Munro [RM78] proved that this game is PSPACE-complete, intended as a general illustration that NP-complete puzzles tend to lead to PSPACE-complete games.

5.2 Cryptarithms (Alphametics, Verbal Arithmetic)

Cryptarithms or *alphametics* or *verbal arithmetic* are classic puzzles involving an equation of symbols, the original being Dudeney's $SEND + MORE = MONEY$ from 1924 [Dud24], in which each symbol (e.g., M) represents a consistent digit (between 0 and 9). The goal is to determine an assignment of digits to symbols that satisfies the equation. Such problems can easily be solved in polynomial time by enumerating all 10! assignments. However, Eppstein [Epp87] proved that it is NP-complete to solve the generalization to base $\Theta(n^3)$ (instead of decimal) and $\Theta(n)$ symbols (instead of 26).

5.3 Crossword Puzzles and Scrabble

Perhaps one of the most popular puzzles are *crossword puzzles*, going back to 1913 and today appearing in almost every newspaper, and the subject of the recent documentary *Wordplay* (2006). Here it is easiest to model the problem of designing crossword puzzles, ignoring the nonmathematical notion of clues. Given a list of words (the dictionary), and a rectangular grid with some squares obstacles and others blank, can we place a subset of the words into horizontally or vertically maximal blank strips so that crossing words have matching letters? Lewis and Papadimitriou [GJ79, p. 258] proved that this question is NP-complete, even when the grid has no obstacles so every row and column must form a word.

Alternatively, this problem can be viewed as the ultimate form of crossword puzzle solving, without clues. In this case it would be interesting to know whether the problem remains NP-hard even if every word in the given list must be used exactly once, so that the single clue could be "use these words". A related open problem is *Scrabble*, which we are not aware of having been studied. The most natural theoretical question is perhaps the one-move version: given the pieces in hand (with letters and scores), and given the current board configuration (with played pieces and available double/triple letter/word squares), what move maximizes score? Presumably the decision question is NP-complete. Also open is the complexity of the two-player game, say in the perfect-information variation where both players know the sequence in which remaining pieces will be drawn as well as the pieces in the opponent's hand. Presumably determining a winning move from a given position in this game is PSPACE-complete.

5.4 Pencil-and-Paper Puzzles: Sudoku and Friends

Sudoku or *Number Place* is a pencil-and-paper puzzle that became popular worldwide starting around 2005 [Del06, Hay06]. American architect Howard Garns first published the puzzle in the May 1979 (and many subsequent) *Dell Pencil Puzzles and Word Games* (without a byline); then Japanese magazine *Monthly Nikolist* imported the puzzle in 1984, trademarking the name Sudoku (single numbers); then the idea spread throughout Japanese publications; finally Wayne Gould published his own computer-generated puzzles in *The Times* in 2004, shortly after which many newspapers and magazines adopted the puzzle. The usual puzzle consists of an 9×9 grid of squares, divided into a 3×3 arrangement of 3×3 tiles. Some grid squares are initially filled with digits between 1 and 9, and some are blank. The goal is to fill the blank squares so that every row, column, and tile has all nine digits without repetition.

Sudoku naturally generalizes to an $n^2 \times n^2$ grid of squares, divided into an $n \times n$ arrangement of $n \times n$ tiles. Yato and Seta [YS03, Yat03] proved that this generalization is NP-complete. In fact, they proved a stronger completeness result, in the class of Another Solution Problems (ASP), where one is given one or more solutions and wishes to find another solution. Thus, in particular, given a Sudoku puzzle and an intended solution, it is NP-complete to determine whether there is another solution, a problem arising in puzzle design. Most Sudoku puzzles give the promise that they have a unique solution. Valiant and Vazirani [VV86] proved that adding such a uniqueness promise keeps a problem NP-hard under randomized reductions, so there is no polynomial-time solution to uniquely solvable Sudokus unless RP = NP.

ASP-completeness (in particular, NP-completeness) has been established for six other paper-and-pencil puzzles by Japanese publisher Nikoli: Nonograms, Slitherlink, Cross Sum, Fillomino, Light Up, and LITS. In a *Nonogram* or *Paint by Numbers* puzzle [UN96], we are given a sequence of integers on each row and column of a rectangular matrix, and the goal is to fill in a subset of the squares in the matrix so that, in each row and column, the maximal contiguous runs of filled squares have lengths that match the specified sequence. In *Slitherlink* [YS03, Yat03], we are given labels between 0 and 4 on some subset of faces in a rectangular grid, and the goal is to draw a simple cycle on the grid so that each labeled face is surrounded by the specified number of edges. In *Kakuro* or *Cross Sum* [YS03], we are given a polyomino (a rectangular grid where only some squares may be used), and an integer for each maximal contiguous (horizontal or vertical) strip of squares, and the goal is to fill each square with a digit between 1 and 9 such that each strip has the specified sum and has no repeated digit. In *Fillomino* [Yat03], we are given a rectangular grid in which some squares have been filled with positive integers, and the goal is to fill the remaining squares with positive integers so that every maximal connected region of equally numbered squares consists of exactly that number of squares. In *Light Up* (*Akari*) [McP05, McP07], we are given a rectangular grid in which squares are either rooms or walls and some walls have a specified integer between 0 and 4, and the goal is to place lights in a subset of the rooms such that each numbered wall has exactly the specified number of (horizontally or vertically) adjacent lights, every room is horizontally or vertically visible from a light, and no two lights are horizontally or vertically visible from each other. In *LITS* [McP07], we are given a division of a rectangle into polyomino pieces, and the goal is to choose a tetromino (connected subset of four squares) in each polyomino such that the union of tetrominoes is connected yet induces no 2×2 square. As with Sudoku, it is NP-complete to both find solutions and test uniqueness of known solutions in all of these puzzles.

NP-completeness has been established for seven other pencil-and-paper games published by Nikoli: Tentai Show, Masyu, Bag, Nurikabe, Hiroimono, Heyawake, and Hitori. In *Tentai Show* or *Spiral Galaxies* [Fri02d], we are given a rectangular grid with dots at some vertices, edge midpoints, and face centroids, and the goal is to divide the rectangle into exactly one polyomino piece per dot that is two-fold rotationally symmetric around the dot. In *Masyu* or *Pearl Puzzles* [Fri02b], we are given a rectangular grid with some squares containing white or black pearls, and the goal is to find a simple path through the squares that visits every pearl, turns $90°$ at every black pearl, does not turn immediately before or after black pearls, goes straight through every white pearl, and turns $90°$ immediately before or after every white pearl. In *Bag* or *Corral Puzzles* [Fri02a], we are given a rectangular grid with some squares labeled with positive integers, and the goal is to find a simple cycle on the grid that encloses all labels and such that the number of squares horizontally and vertically visible from each labeled square equals the label. In *Nurikabe* [McP03, HKK04], we are given a rectangular grid with some squares labeled with positive integers, and the goal is to find a connected subset of unlabeled squares that induces no 2×2 square and whole removal results in exactly one region per labeled square whose size equals that label. McPhail's reduction [McP03] uses labels 1 through 5, while Holzer et al.'s reduction [HKK04] only uses labels 1 and 2 (just 1 would be trivial) and works without the connectivity rule and/or the 2×2 rule. In *Hiroimono* or *Goishi Hiroi* [And07], we are given a collection of stones at vertices of a rectangular grid, and the goal is to find a path that visits all stones, changes directions by $\pm 90°$ and only at stones, and removes stones as they are visited (similar to Phutball in Section 4.15). In *Heyawake* [HR07], we are given a subdivision of a rectangular grid into rectangular rooms, some of which are labeled with a positive integer, and the goal is to paint a subset of unit squares so that the number of painted squares in each labeled room equals the label, painted squares are never (horizontally or vertically) adjacent, unpainted squares are connected (via horizontal and vertical connections), and maximal contiguous (horizontal or vertical) strips of squares intersect at most two rooms. In *Hitori* [Hea08c], we are given a rectangular grid with each square labeled with an integer, and the goal is to paint a subset of unit squares so that every row and every column has no repeated unpainted label (similar to Sudoku), painted squares are never (horizontally or vertically) adjacent, and unpainted squares are connected (via horizontal and vertical connections).

A different kind of pencil-and-paper puzzle is *Morpion Solitaire*, popular in several European countries. The game starts with some configuration of points drawn at the intersections of a square grid (usually in a standard cross pattern). A move consists of placing a new point at a grid intersection, and then drawing a horizontal, vertical, or diagonal line segment connecting five consecutive points that include the new one. Line segments with the same direction cannot share a point (the *disjoint model*); alternatively, line segments with the same direction may overlap only at a common endpoint (the *touching model*). The goal is to maximize the number of moves before no moves are possible. Demaine, Demaine, Langerman, and Langerman [DDLL06] consider this game generalized to moves connecting any number $k + 1$ of points instead of just 5. In addition to bounding the number of moves from the standard cross configuration, they prove complexity results for the general case. They show that, in both game models and for $k \geq 3$, it is NP-hard to find the longest play from a given pattern of n dots, or even to approximate the longest play within $n^{1-\varepsilon}$ for any $\varepsilon > 0$. For $k > 3$, the problem is in fact NP-complete. For $k = 3$, it is open whether the problem is in NP, and for $k = 2$ it could even be in P.

A final NP-completeness result for pencil-and-paper puzzles is the Battleship puzzle. This puzzle is a one-player perfect-information variant on the classic two-player imperfect-information game, *Battleship*. In *Battleships* or *Battleship Solitaire* [Sev], we are given a list of $1 \times k$ ships for various values of k; a rectangular grid with some squares labeled as water, ship interior, ship end, or entire (1×1) ship; and the number of ship (nonwater) squares that should be in each row and each column. The goal is to complete the square labeling to place the given ships in the grid while matching the specified number of ship squares in each row and column.

Several other pencil-and-paper puzzles remain unstudied from a complexity standpoint. For example, Nikoli's English website[2] suggests Hashiwokakero, Kuromasu (Where is Black Cells), Number Link, Ripple Effect, Shikaku, and Yajilin (Arrow Ring); and Nikoli's Japanese website[3] lists more.

5.5 Moving Tokens: Fifteen Puzzle and Generalizations

[2]http://www.nikoli.co.jp/en/puzzles/
[3]http://www.nikoli.co.jp/ja/puzzles/

The *Fifteen Puzzle* or *15 Puzzle* [BCG04, p. 864] is a classic puzzle consisting of fifteen square blocks numbered 1 through 15 in a 4×4 grid; the remaining sixteenth square in the grid is a hole which permits blocks to slide. The goal is to order the blocks to be increasing in English reading order. The (six) hardest solvable positions require exactly 80 moves [BMFN99]. Slocum and Sonneveld [SS06] recently uncovered the history of this late 19th-century puzzle, which was well-hidden by popularizer Sam Loyd since his claim of having invented it.

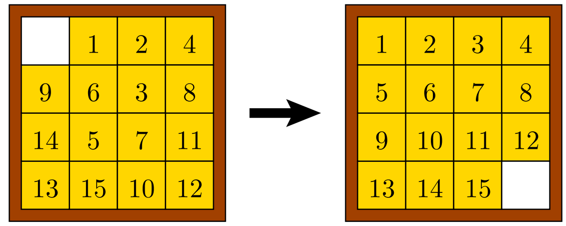


Figure 10: 15 puzzle: Can you get from the left configuration to the right in 16 unit slides?

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. See, e.g., [Arc99, Sto79, Wil74]. When the puzzle is solvable, the required numbers moves is $\Theta(n^3)$ in the worst case [Par95]. On the other hand, it is NP-complete to find a solution using the fewest possible slides from a given configuration [RW90]. It is also NP-hard to approximate the fewest slides within an additive constant, but there is a polynomial-time constant-factor approximation [RW90].

The parity technique for determining solvability of the $n^2 - 1$ puzzle has been generalized to a class of similar puzzles on graphs. Consider an N-vertex graph in which $N - 1$ vertices have tokens labeled 1 through $N - 1$, one vertex is empty (has no token), and each operation in the puzzle moves a token to an adjacent empty vertex. The goal is to reach one configuration from another. This general puzzle encompasses the $n^2 - 1$ puzzle and several other puzzles involving sliding balls in circular tracks, e.g., the Lucky Seven puzzle [BCG04, p. 865] or the puzzle shown in Figure 11. Wilson [Wil74], [BCG04, p. 866] characterized when these puzzles are solvable, and furthermore characterized their group structure. In most cases, all puzzles are solvable (forming the symmetric group) unless the graph the graph is bipartite, in which case half of the puzzles are solvable (forming the alternating group). In addition, there are three special situations: cycle graphs, graphs having a cut vertex, and the special example in Figure 11.

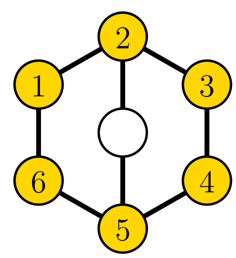


Figure 11: The Tricky Six Puzzle [Wil74], [BCG04, p. 868] has six connected components of configurations.

Even more generally, Kornhauser, Miller, and Spirakis [KMS84] showed how to decide solvability of puzzles with any number k of labeled tokens on N vertices. They also prove that $O(N^3)$ moves always suffice, and $\Omega(N^3)$ moves are sometimes necessary, in such puzzles. Calinescu, Dumitrescu, and Pach [CDP06] consider the number of token "shifts"—continuous moves along a path of empty nodes—required in such puzzles. They prove that finding the fewest-shift solution is NP-hard in the infinite square grid and APX-hard in general graphs, even if the tokens are unlabeled (identical). On the positive side, they present a 3-approximation for unlabeled tokens in general graphs, an optimal solution for unlabeled tokens in trees, an upper bound of N slides for unlabeled tokens in general graphs, and an upper bound of $O(N)$ slides for labeled tokens in the infinite square grid.

Restricting the set of legal moves can make such puzzles harder. Consider a graph with unlabeled tokens on some vertices, and the constraint that the tokens must form an independent set on the graph (i.e., no two tokens are adjacent along an edge). A move is made by sliding a token along an edge to an adjacent vertex, subject to maintaining the nonadjacency constraint. Then the problem of determining whether a sequence of moves can ever move a given token, called *Sliding Tokens*

[HD05], is PSPACE-complete.

Subway Shuffle [Hea05b, Hea06b] is another constrained token-sliding puzzle on a graph. In this puzzle both the tokens and the graph edges are colored; a move is to slide a token along an edge of matching color to an unoccupied adjacent vertex. The goal is to move a specified token (the "subway car you have boarded") to a specified vertex (your "exit station"). A sample puzzle is shown in Figure 12. The complexity of determining whether there is a solution to a given puzzle is open. This open problem is quite fascinating: solving the puzzle empirically seems hard, based on the rapid growth of minimum solution length with graph size [Hea05b]. However, it is easy to determine whether a token may move at all by a sequence of moves, evidently making the proof techniques used for Sliding Tokens and related problems useless for showing hardness. Subway Shuffle can also be seen as a generalized version of 1×1 Rush Hour (Section 5.7).

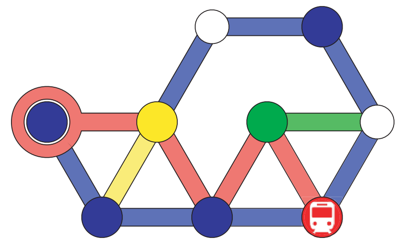


Figure 12: A Subway Shuffle puzzle with one red car, four blue cars, one yellow car, and one green car. White nodes are empty. Moving the red car to the circled station requires 43 moves.

Another kind of token-sliding puzzle is *Atomix*, a computer game first published in 1990. Game play takes place on a rectangular board; pieces are either *walls* (immovable blocks) or *atoms* of different types. A move is to slide an atom; in this case the atom must slide in its direction of motion until it hits a wall (as in the PushPush family, below (Section 5.8)). The goal is to assemble a particular pattern of atoms (a molecule). Huffner, Edelkamp, Fernau, and Niedermeier [HEFN01] observed that Atomix is as hard as the $(n^2 - 1)$-puzzle, so it is NP-hard to find a minimum-move solution. Holzer and Schwoon [HS04a] later proved the stronger result that it is PSPACE-complete to determine whether there is a solution.

Lunar Lockout is another token-sliding puzzle, similar to Atomix in that the tokens slide until stopped. Lunar Lockout was produced by ThinkFun at one time; essentially the same game is now sold as "Pete's Pike". (Even earlier, the game was called "UFO".) In Lunar Lockout there are no walls or barriers; a token may only slide if there is another token in place that will stop it. The goal is to get a particular token to a particular place. Thus, the rules are fairly simple and natural; however, the complexity is open, though there are partial results. Hock [Hoc01] showed that Lunar Lockout is NP-hard, and that when the target token may not revisit any position on the board, the problem becomes NP-complete. Hartline and Libeskind-Hadas [HLH03] show that a generalization of Lunar Lockout which allows fixed blocks is PSPACE-complete.

5.6 Rubik's Cube and Generalizations

Alternatively, the n^2-1 puzzle can be viewed as a special case of determining whether a permutation on N items can be written as a product (composition) of given generating permutations, and if so, finding such a product. This family of puzzles also includes *Rubik's Cube* (recently shown to be solvable in 26 moves [KC07]) and its many variations. In general, the number of moves (terms) required to solve such a puzzle can be exponential (unlike the Fifteen Puzzle). Nonetheless, an $O(N^5)$-time algorithm can decide whether a given puzzle of this type is solvable, and if so, find an implicit representation of the solution [Jer86]. On the other hand, finding a solution with the fewest moves (terms) is PSPACE-complete [Jer85]. When each given generator cyclically shifts just a bounded number of items, as in the Fifteen Puzzle but not in a $k \times k \times k$ Rubik's Cube, Driscoll and Furst [DF83] showed that such puzzles can be solved in polynomial time using just $O(N^2)$ moves. Furthermore, $\Theta(N^2)$ is the best possible bound in the worst case, e.g., when the only permitted moves are swapping adjacent elements on a line. See [KMS84, McK84] for other (not explicitly algorithmic) results on the maximum number of moves for various special cases of such puzzles.

5.7 Sliding Blocks and Rush Hour

A classic reference on a wide class of sliding-block puzzles is by Hordern [Hor86]. 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 usually either to move a particular block to a particular place, or to re-arrange one configuration into another. Figure 13 shows an example which, according to Gardner [Gar64], may be the earliest (1909) and is the most widely sold (after the Fifteen Puzzle, in each case). Gardner [Gar64] first raised the question of whether there is an efficient algorithm to solve such puzzles. Spirakis and Yap [SY83] showed that achieving a specified target configuration is NP-hard, and conjectured PSPACE-completeness. Hopcroft, Schwartz, and Sharir [HSS84] proved PSPACE-completeness shortly afterwards, renaming the problem to the "Warehouseman's Problem". In the Warehouseman's Problem, there is no restriction on the sizes of blocks; the blocks in the reduction grow with the size of the containing box. By contrast, in most sliding-block puzzles, the blocks are of small constant sizes. Finally, Hearn and Demaine [HD02, HD05] showed that it is PSPACE-hard to decide whether a given piece can move at all by a sequence of moves, even when all the blocks are 1×2 or 2×1. This result is best possible: the results above about unlabeled tokens in graphs show that 1×1 blocks are easy to re-arrange.

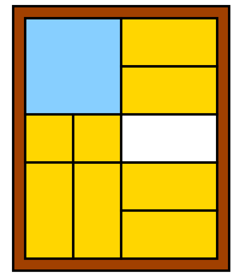


Figure 13: Dad's Puzzle [Gar64]: moving the large square into the lower-left corner requires 59 moves.

A popular sliding-block puzzle is *Rush Hour*, distributed by ThinkFun, Inc. (formerly Binary Arts, Inc.). We are given a configuration of several 1×2, 1×3, 2×1, and 3×1 rectangular blocks arranged in an $m \times n$ grid. (In the commercial version, the board is 6×6, length-two rectangles are realized as *cars*, and length-three rectangles are *trucks*.) Horizontally oriented blocks can slide left and right, and vertically oriented blocks can slide up and down, provided the blocks remain disjoint. (Cars and trucks can drive only forward or reverse.) The goal is to remove a particular block from the puzzle via a one-unit opening in the bounding rectangle. Flake and Baum [FB02] proved that this formulation of Rush Hour is PSPACE-complete. Their approach is also the basis for Nondeterministic Constraint Logic described in Section 3. A version of Rush Hour played on a triangular grid, *Triagonal Slide-Out*, is also PSPACE-complete [Hea06b]. Tromp and Cilibrasi [Tro00, TC04] strengthened Flake and Baum's result by showing that Rush Hour remains PSPACE-complete even when all the blocks have length two (cars). The complexity of the problem remains open when all blocks are 1×1 but labeled whether they move only horizontally or only vertically [HD02, TC04, HD05]. As with Subway Shuffle (Section 5.5), solving the puzzle (by escaping the target block from the grid) empirically seems hard [TC04], whereas it is easy to determine whether a block may move at all by a sequence of moves. Indeed, 1×1 Rush Hour is a restricted form of Subway Shuffle, where there are only two colors, the graph is a grid, and horizontal edges and vertical edges use different colors. Thus, it should be easier to find positive results for 1×1 Rush Hour, and easier to find hardness results for Subway Shuffle. We conjecture that both are PSPACE-complete, but existing proof techniques seem inapplicable.

5.8 Pushing Blocks

Similar in spirit to the sliding-block puzzles in Section 5.7 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 [Wil91], 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 [DO92] 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 [BOS94] strengthened this result to PSPACE-completeness. Recently, Hoffmann [Hof00] 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 NP-hard by Dor and Zwick [DZ99] and later PSPACE-complete by Culberson [Cul98]. Later this result was strengthened to configurations with no fixed blocks [HD02, HD05]. A simpler puzzle, called PUSH-1, arises when the goal is simply for the robot to reach a particular position, and there are no fixed blocks. Demaine, Demaine, and O'Rourke [DDO00a] prove that this puzzle is NP-hard, but it remains open whether it is in NP or PSPACE-complete. On the other hand, PSPACE-completeness has been established for PUSH-2-F, in which there are fixed blocks and the robot can push two blocks at a time [DHH02].

A variation on the PUSH series of puzzles, called PUSHPUSH, is when a block always slides as far as possible when pushed. Such puzzles arise in a computer game with the same name [DDO00a, DDO00b, OS99]. PUSHPUSH-1 was established to be NP-hard slightly earlier than PUSH-1 [DDO00b, OS99]; the PUSH-1 reduction [DDO00a] also applies to PUSHPUSH-1. PUSHPUSH-k was later shown PSPACE-complete for any fixed $k \geq 1$ [DHH04]. Hoffmann's reduction for PUSH-* also proves that PUSHPUSH-* is NP-hard without fixed blocks.

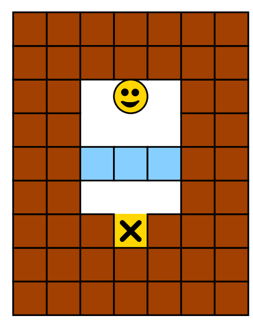


Figure 14: A PUSH-1 or PUSHPUSH-1 puzzle: move the robot to the X by pushing light blocks.

Another variation, called PUSH-X, disallows the robot from revisiting a square (the robot's path cannot cross). This direction was suggested in [DDO00a] because it immediately places the puzzles in NP. Demaine and Hoffmann [DH01] 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.

Friedman [Fri02c] considers another variation, where gravity acts on the blocks (but not the robot): when a block is pushed it falls if unsupported. He shows that PUSH-1-G, where the robot may push only one block, is NP-hard.

River Crossing, another ThinkFun puzzle (originally *Plank Puzzles* by Andrea Gilbert [Gil00]), is similar to pushing-block puzzles in that there is a unique piece that must be used to move the other puzzle pieces. The game board is a grid, with *stumps* at some intersections, and *planks* arranged between some pairs of stumps, along the grid lines. A special piece, the *hiker*, always stands on some plank, and can walk along connected planks. He can also pick up and carry a single plank at a time, and deposit that plank between stumps that are appropriately spaced. The goal is for the hiker to reach a particular stump. Figure 15 shows a sample puzzle. Hearn [Hea04, Hea06b] proves that River Crossing is PSPACE-complete, by a reduction from Constraint Logic.

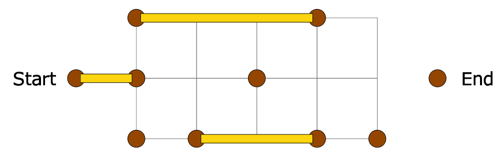


Figure 15: A River Crossing puzzle. Move from start to end.

5.9 Rolling and Tipping Blocks

In some puzzles the blocks can change their orientation as well as their position. Rolling-cube puzzles were popularized by Martin Gardner in his *Mathematical Games* columns in *Scientific American* [Gar63, Gar65, Gar75]. In these puzzles, one or more cubes with some labeled sides (often dice) are placed on a grid, and may roll from cell to cell, pivoting on their edges between cells. Some cells may have labels which must match the face-up label of the cube when it visits the cell. The tasks generally involve completing some type of circuit while satisfying some label constraints (e.g., by ensuring that a particular labeled face never points up). Recently Buchin et al. [BBD+07] formalized this type of problem and derived several results. In their version, every labeled cell must be visited, with the label on the top face of the cube matching the cell label. Cells can be labeled, *blocked*, or *free*. Blocked cells cannot be visited; free cells can be visited regardless of cube orientation. Such puzzles turn out to be easy if labeled cells can be visited multiple times. If each labeled cell must be visited exactly one, the problem becomes NP-complete.

Rolling-block puzzles were later generalized by Richard Tucker to puzzles where the blocks no longer need be cubes. In these puzzles, the blocks are $k \times m \times n$ boxes. Typically, some grid cells are blocked, and the goal is to move a block from a start position to an end position by successive rotations into unblocked cells. Buchin and Buchin [BB07] recently showed that these puzzles are PSPACE-complete when multiple rolling blocks are used, by a reduction from Constraint Logic.

A commercial puzzle involving blocks that tip is the ThinkFun puzzle *TipOver* (originally the *Kung Fu Packing Crate Maze* by James Stephens [Ste03]). In this puzzle, all the blocks are $1 \times 1 \times n$ ("crates") and initially vertical. A *tipper* stands on a starting crate, and attempts to reach a target crate. The tipper may tip over a vertical crate it is standing on, if there is empty space in the grid for it to fall into. The tipper may also move between connected crates (but cannot jump diagonally). Unlike rolling-block puzzles, in these tipping puzzles once a block has tipped over it may not stand up again (or indeed move at all). Hearn [Hea06a] showed that TipOver is NP-complete, by a reduction from Constraint Logic.

A two-player tipping-block game inspired by TipOver, called *Cross Purposes*, was invented by Michael Albert, and named by Richard Guy, at the Games at Dalhousie III workshop in 2004. In Cross Purposes, all the blocks are $1 \times 1 \times 2$, and initially vertical. One player, *horizontal*, may only tip blocks over horizontally as viewed from above; the other player, *vertical*, may only tip blocks over vertically as viewed from above. The game follows normal play: the last player to move wins. Hearn [Hea08a] proved that Cross Purposes is PSPACE-complete, by a reduction from Constraint Logic.

5.10 Peg Solitaire (Hi-Q)

The classic *peg solitaire puzzle* is shown in Figure 16. Pegs are arranged in a Greek cross, with the central peg missing. Each move *jumps* a peg over another peg (adjacent horizontally or vertically) to the opposite unoccupied position within the cross, and removes the peg that was jumped over. The goal is to leave just a single peg, ideally located in the center. A variety of similar peg solitaire puzzles are given in [Bea85]. See also Chapter 23 of *Winning Ways* [BCG04, pp. 803–841].

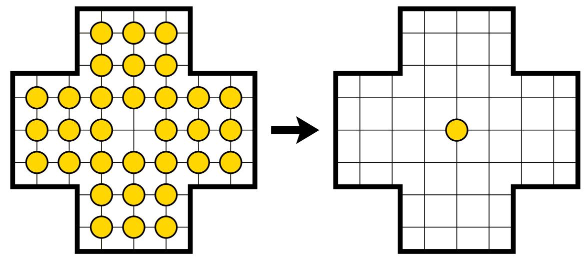


Figure 16: Central peg solitaire (Hi-Q): initial and target configurations.

A natural generalization of peg solitaire is to consider pegs arranged in an $n \times n$ board and the goal is to leave a single peg. Uehara and Iwata [UI90] proved that it is NP-complete to decide whether such a puzzle is solvable.

On the other hand, Moore and Eppstein [ME02] proved that the one-dimensional special case (pegs along a line) can be solved in polynomial time. In particular, the binary strings representing initial configurations that can reach a single peg turn out to form a regular language, so they can be parsed using regular expressions. (This fact has been observed in various contexts; see [ME02] for references as well as a proof.) Using this result, Moore and Eppstein build a polynomial-time algorithm to maximize the number of pegs removed from any given puzzle.

Moore and Eppstein [ME02] also study the natural impartial two-player game arising from peg solitaire, *duotaire*: players take turns jumping, and the winner is determined by normal play. (This game is proposed, e.g., in [Bea85].) Surprisingly, the complexity of this seemingly simple game is open. Moore and Eppstein conjecture that the game cannot be described even by a context-free language, and prove this conjecture for the variation in which multiple jumps can be made in a single move. Konane (Section 4.14) is a natural partizan two-player game arising from peg solitaire.

5.11 Card Solitaire

Two solitaire games with playing cards have been analyzed from a complexity standpoint. With all such games, we must generalize the deck beyond 52 cards. The standard approach is to keep the number of suits fixed at four, but increase the number of ranks in each suit to n.

Klondike or *Solitaire* is the classic game, in particular bundled with Microsoft Windows since its early days. In the perfect information of this game, we suppose the player knows all of the normally hidden cards. Longpré and McKenzie [LM07] proved that the perfect-information version is NP-complete, even with just three suits. They also prove that Klondike with one black suit and one red suit is NL-hard; Klondike with any fixed number of black suits and no red suits is in NL; Klondike with one suit is in $AC^0[3]$; among other results.

FreeCell is another common game distributed with Microsoft Windows since XP. We will not attempt to describe the rules here. Helmert [Hel03] proved that FreeCell is NP-complete, for any fixed positive number of free cells.

5.12 Jigsaw, Edge-Matching, Tiling, and Packing Puzzles

Jigsaw puzzles [Wil04] are another one of the most popular kinds of puzzles, dating back to the 1760s. One way to formalize such puzzles is as a collection of square pieces, where each side is either straight or augmented with a tab or a pocket of a particular shape. The goal is to arrange the given pieces so that they form exactly a given rectangular shape. Although this formalization does not explicitly allow for patterns on pieces to give hints about whether pieces match, this information can simply be encoded into the shapes of the tabs and pockets, making them compatible only when the patterns also match. Deciding whether such a puzzle has a solution was recently shown NP-complete [DD07].

A closely related type of puzzles is *edge-matching puzzles* [Hau95], dating back to the 1890s. In the simplest form, the pieces are squares and, instead of tabs or pockets, each edge is colored to indicate compatibility. Squares can be placed side-by-side if the edge colors match, either being exactly equal (*unsigned* edge matching) or being opposite (*signed* edge matching). Again the goal is to arrange the given pieces into a given rectangle. Signed edge-matching puzzles are common in reality where the colors are in fact images of lizards, insects, etc., and one side shows the head while the other shows the tail. Such puzzles are almost identical to jigsaw puzzles, with tabs and pockets representing the sign; jigsaw puzzles are effectively the special case in which the boundary must be uniformly colored. Thus, signed edge-matching puzzles are NP-complete, and in fact, so are unsigned edge-matching puzzles [DD07].

An older result by Berger [Ber66] proves that the infinite generalization of edge-matching puzzles, where the goal is to tile the entire plane given infinitely many copies of each tile type, is undecidable. This result is for unsigned puzzles, but by a simple reduction in [DD07] it holds for signed puzzles as well. Along the same lines, Garey, Johnson, and Papadimitriou [GJ79, p. 257] observe that the finite version with a given target rectangle is NP-complete when given arbitrarily many copies of each tile type. In contrast, the finite result above requires every given tile to be used exactly once, which corresponds more closely to real puzzles.

A related family of tiling and packing puzzles involve polyforms such as *polyominoes*, edge-to-edge joinings of unit squares. In general, we are given a collection of such shapes and a target shape to either tile (form exactly) or pack (form with gaps). In both cases, pieces cannot overlap, so the tiling problem is actually a special case in which the piece areas sum to the target areas. One of the few positive results is for (mathematical) *dominoes*, polyominoes (rectangles) made from two unit squares: the tiling and (grid-aligned) packing problems can be solved in polynomial time for arbitrary polyomino target shapes by perfect and maximum matching, respectively; see also the elegant tiling criterion of Thurston [Thu90]. In contrast, with "real" dominoes, where each square has a color and adjacent dominoes must match in color, tiling (and hence packing) becomes NP-complete [Bie05]. The tiling problem is also NP-complete when the target shape is a polyomino with holes and the pieces are all identical 2×2 squares, or 1×3 rectangles, or 2×2 L shapes [MR01]. The packing problem [LC89] and the tiling problem [DD07] are NP-complete when the given pieces are differently sized squares and the target shape is a square. Finally, the tiling problem is NP-complete when the given pieces are polylogarithmic-area polyominoes and the target shape is a square [DD07]; this result follows by simulating jigsaw puzzles.

5.13 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. An additional problem is to decide whether a given configuration has a unique realization.

Kaye [Kay00b] proves that testing consistency is NP-complete. This result leaves open the complexity of the other questions mentioned above. Fix and McPhail [FM04] strengthen Kaye's result to show NP-completeness of determining consistency when the uncovered numbers are all at most 1. McPhail [McP03] also shows that, given a consistent placement of mines, determining whether there is another consistent placement is NP-complete (ASP-completeness from Section 5.4).

Kaye [Kay00a] also proves that an infinite generalization of Minesweeper is undecidable. Specifically, the question is whether a given finite configuration can be extended to the entire plane. The rules permit a much more powerful level of information revealed by uncovering squares; for example, discovering that one square has a particular label might imply that there are exactly 3 adjacent squares with another particular label. (The notion of a mine is lost.) The reduction is from tiling (Section 5.12).

Hearn [Hea06b, Hea08b] argues that the "natural" decision question for Minesweeper, in keeping with the standard form for other puzzle complexity results, is whether a given (assumed consistent) instance can (definitely) be solved, which is a different question from any of the above. He observes that a simple modification to Kaye's construction shows that this question is coNP-complete, an unusual complexity class for a puzzle. The reduction is from Tautology. (If the instance is not known to be consistent, then the problem may not be in coNP.) Note that this question is not the same as whether a given configuration has a unique realization: there could be multiple realizations, as long as the player is guaranteed that known-safe moves will eventually reveal the entire configuration.

5.14 Mahjong Solitaire (Shanghai)

Majong solitaire or *Shanghai* is a common computer game played with Mahjong tiles, stacked in a pattern that hides some tiles, and shows other tiles, some of which are completely exposed. Each move removes a pair of matching tiles that are completely exposed; there are precisely four tiles in each equivalence class of matching. The goal is to remove all tiles.

Condon, Feigenbaum, Lund, and Shor [CFS97] proved that it is PSPACE-hard to approximate the maximum probability of removing all tiles within a factor of n^ε, assuming that there are arbitrarily many quadruples of matching tiles and that the hidden tiles are uniformly distributed. Eppstein [Epp] proved that it is NP-complete to decide whether all tiles can be removed in the perfect-information version of this puzzle where all tile positions are known.

5.15 Tetris

Tetris is a popular computer puzzle game invented in the mid-1980s by Alexey Pazhitnov, and by 1988 it became the best-selling game in the United States and England. The game takes place in a rectangular grid (originally, 20×10) with some squares occupied by blocks. During each move, the computer generates a tetromino piece stochastically and places it at the top of the grid; the player can rotate the piece and slide it left or right as it falls downward. When the piece hits another piece or the floor, its location freezes and the move ends. Also, if there are any completely filled rows, they disappear, bringing any rows above down one level.

To make Tetris a perfect-information puzzle, Breukelaar et al. [BDH$^+$04] suppose that the player knows in advance the entire sequence of pieces to be delivered. Such puzzles appear in *Games Magazine*, for example. They then prove NP-completeness of deciding whether it is possible to stay alive, i.e., always be able to place pieces. Furthermore, they show that maximizing various notions of score, such as the number of lines cleared, is NP-complete to approximate within an $n^{1-\varepsilon}$ factor. The complexity of Tetris remains open with a constant number of rows or columns, or with a stochastically chosen piece sequence as in [Pap85].

5.16 Clickomania (Same Game)

Clickomania or *Same Game* [BDD$^+$02] 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 Figure 17 (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.

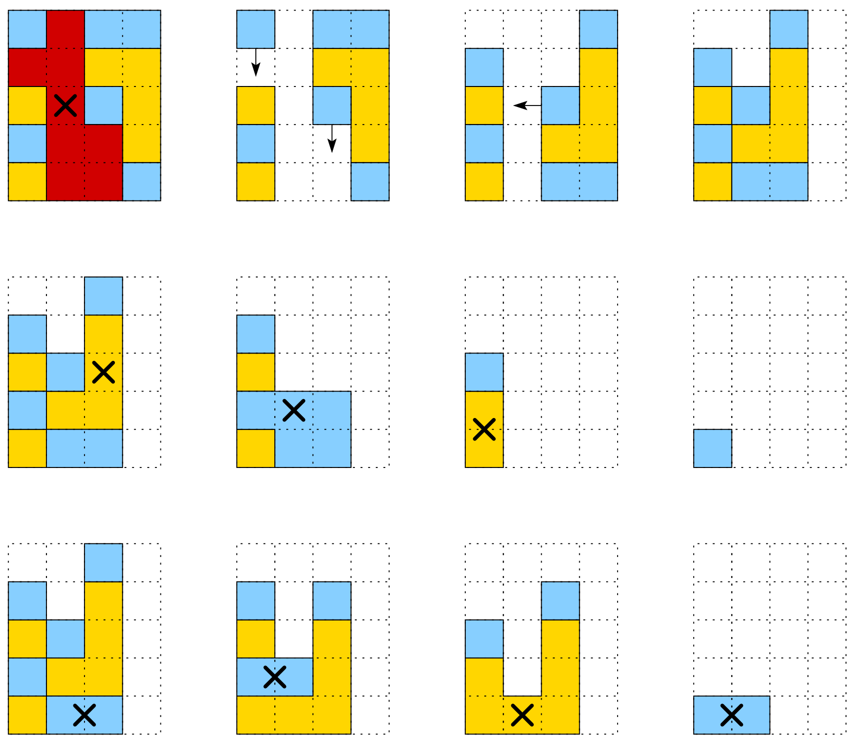


Figure 17: The falling rules for removing a group in Clickomania (top), a failed attempt (middle), and a successful solution (bottom).

The main goal in Clickomania is to remove all the blocks. A simple example for which this is impossible is a checkerboard, where no move can be made. A secondary goal is to maximize the score, typically defined by k^2 points being awarded for removal of a group of k blocks.

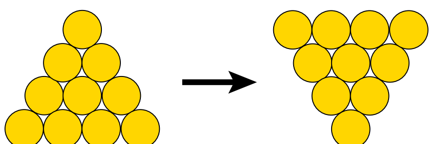
Biedl et al. [BDD$^+$02] proved that it is NP-complete to decide whether all blocks can be removed in a Clickomania puzzle. This complexity result holds even for puzzles with two columns and five colors, and for puzzles with five columns and three colors. On the other hand, for puzzles with one column (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 [BDD$^+$02], in which players take turns removing groups and normal play determines the winner; the complexity of this game remains open.

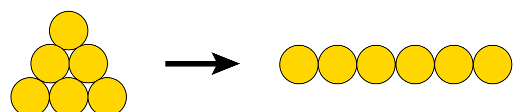
A related puzzle is called *Vexed*, also *Cubic*. In this puzzle there are fixed blocks, as well as the mutually annihilating colored blocks. A move in Vexed is to slide a colored block one unit left or right into an empty space, whereupon gravity will pull the block down until it contacts another block; then any touching blocks of the same color disappear. Again the goal is to remove all the colored blocks. Friedman [Fri01] showed that Vexed is NP-complete.[4]

5.17 Moving Coins

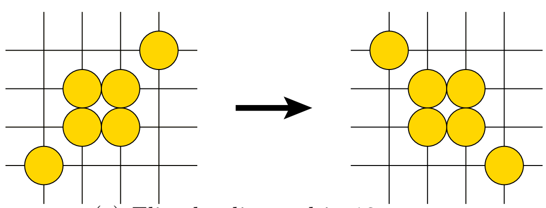
Several coin-sliding and coin-moving puzzles fall into the following general framework: re-arrange one configuration of unit disks in the plane into another configuration by a sequence of moves, each repositioning a coin in an empty position that touches at least two other coins. Examples of such puzzles are shown in Figure 18. This framework can be further generalized to nongeometric puzzles involving movement of tokens on graphs with adjacency restrictions.



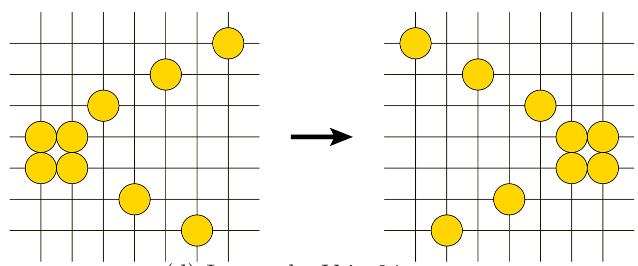
(a) Turn the pyramid upside-down in three moves.



(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.

Figure 18: 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 [DDV00].

Coin-moving puzzles are analyzed by Demaine, Demaine, and Verrill [DDV00]. In particular, they study puzzles as in Figure 18 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

[4] David Eppstein pointed out that all that was shown was NP-hardness; the problem was not obviously in NP (http://www.ics.uci.edu/~eppstein/cgt/hard.html). Friedman and R. Hearn together showed that it is in NP as well (personal communication).

puzzles with at least two extra coins past what is required to achieve the span. (In particular, all such puzzles are solvable.)

5.18 Dyson Telescopes

The *Dyson Telescope Game* is an online puzzle produced by the Dyson corporation, whimsically based on their telescoping vacuum cleaners. The goal is to maneuver a ball on a square grid from a starting position to a goal position by extending and retracting telescopes on the grid. When a telescope is extended, it grows to its maximum length in the direction it points (parameters of each telescope), unless it is stopped by another telescope. If the ball is in the way, it is pushed by the end of the telescope. When a telescope is retracted, it shrinks back to unit length, pulling the ball with it if the ball was at the end of the telescope.

Demaine et al. [DDF$^+$08] showed that determining whether a given puzzle has a solution is PSPACE-complete in the general case. On the other hand, the problem is polynomial for certain restricted configurations which are nonetheless interesting for humans to play. Specifically, if no two telescopes face each other and overlap when extended by more than one space, then the problem is polynomial. Many of the game levels in the online version have this property.

5.19 Reflection Puzzles

Two puzzles involving reflection of directional light or motion have been studied from a complexity-theoretic standpoint.

In *Reflections* [Kem03], we are given a rectangular grid with one square marked with a laser pointed in one of the four axis-parallel directions, one or more squares marked as light bulbs, some squares marked one-way in an axis-parallel direction, and remaining squares marked either empty or wall. We are also given a number of diagonal mirrors and/or T-splitters which we can place arbitrarily into empty squares. The light then travels from the laser; when it meets a diagonal mirror, it reflects by 90° according to the orientation of the mirror; when it meets a splitter at the base of the T, it splits into both orthogonal directions; when it meets a one-way square, it stops unless the light direction matches the one-way orientation; when it meets a light bulb, it toggles the bulb's state and stops; and when it meets a wall, it stops. The goal is to place the mirrors and splitters so that each light bulb gets hit an odd number of times. This puzzle is NP-complete [Kem03].

In *Reflexion* [HS04b], we are given a rectangular grid in which squares are either walls, mirrors, or diamonds. Also, one square is the starting position for a ball and another square is the target position. We may release the ball in one of the four axis-parallel directions, and we may flip mirrors between their two diagonal orientations while the ball moves. The ball travels like a ray of light, reflecting at mirrors and stopping at walls; at diamonds, it turns around and erases the diamond. The goal is to reach the target position. In this simplest form, Reflexion is SL-complete which actually implies a polynomial-time algorithm [HS04a]. If some of the mirrors can be flipped only before the ball releases, the puzzle becomes NP-complete. If some trigger squares toggle other squares between wall and empty, or if some squares contain horizontally or vertically movable blocks (which also cause the ball to turn around), then the puzzle becomes PSPACE-complete.

5.20 Lemmings

Lemmings is a popular computer puzzle game dating back to the early 1990s. Characters called lemmings start at one or more initial locations and behave deterministically according to their mode, initially just walking in a fixed direction, turning around at walls, and falling off cliffs, dying if it falls too far. The player can modify this basic behavior by applying a skill to a lemming; each skill has a limited number of such applications. The goal is for a specified number of lemmings to reach a specified target position. The exact rules, particularly the various skills, are too complicated to detail here. Cormode [Cor04] proved that such puzzles are NP-complete, even with just one lemming. Membership in NP follows from assuming a polynomial upper bound on the time limit in a level (a fairly accurate modeling of the actual game); Cormode conjectures that this assumption does not affect the result.

6 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 between 2 and 3 of its 8 neighbors were alive, and a dead cell becomes alive if it had precisely 3 live neighbors.

Many questions can be asked about an initial configuration of Life; one key question is whether the population will ever completely die out (no cells are alive). Chapter 25 of *Winning Ways* [BCG04, pp. 927–961] describes a reduction showing that this question is undecidable. In particular, the same question about Life restricted within a polynomially bounded region is PSPACE-complete. More recently, Rendell [Ren05] constructed an explicit Turing machine in Life, which establishes the same results.

There are other open complexity-theoretic questions about Life.[5] How hard is it to tell whether a configuration is a Garden of Eden, that is, cannot be the state that results from another? Given a rectangular pattern in Life, how hard is it to extend the pattern outside the rectangle to form a Still Life (which never changes)?

Several other cellular automata, with different survival and birth rules, have been studied; see, e.g., [Wol94].

7 Open Problems

Many open problems remain in Combinatorial Game Theory. Guy and Nowakowski [GN02] have compiled a list of such problems.

Many open problems also remain on the algorithmic side, and have been mentioned throughout this paper. Examples of games and puzzles whose complexities remain unstudied, to our knowledge, are Domineering (Section 4.11), Connect Four, Pentominoes, Fanorona, Nine Men's Morris, Chinese checkers, Lines of Action, Chinese Chess, Quoridor, and Arimaa. For many other games and puzzles, such as Dots and Boxes (Section 4.12) and pushing-block puzzles (Section 5.8), some hardness results are known, but the exact complexity remains unresolved. It would also be interesting to consider games of imperfect information that people play, such as Scrabble (Section 5.3, Backgammon, and Bridge. Another interesting direction for future research is to build a more comprehensive theory for analyzing combinatorial puzzles.

[5] These two questions were suggested by David Eppstein.

Acknowledgments

Comments from several people have helped make this survey more comprehensive, including Martin Demaine, Aviezri Fraenkel, Martin Kutz, and Ryuhei Uehara.

References

[AKI87] Hiroyuki Adachi, Hiroyuki Kamekawa, and Shigeki Iwata. Shogi on $n \times n$ board is complete in exponential time (in Japanese). *Transactions of the IEICE*, J70-D(10):1843–1852, October 1987.

[All87] Norman Alling. *Foundations of Analysis over Surreal Number Fields*. North-Holland Publishing Co., Amsterdam, 1987.

[And07] Daniel Andersson. HIROIMONO is NP-complete. In *Proceedings of the 4th International Conference on FUN with Algorithms*, volume 4475 of *Lecture Notes in Computer Science*, pages 30–39, 2007.

[ANW07] Michael H. Albert, Richard J. Nowakowski, and David Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory*. A K Peters, Wellesley, MA, 2007.

[Arc99] Aaron F. Archer. A modern treatment of the 15 puzzle. *American Mathematical Monthly*, 106(9):793–799, 1999.

[BB07] Kevin Buchin and Maike Buchin. Rolling block mazes are PSPACE-complete. Manuscript, 2007.

[BBD+07] Kevin Buchin, Maike Buchin, Erik D. Demaine, Martin L. Demaine, Dania El-Khechen, Sándor Fekete, Christian Knauer, André Schulz, and Perouz Taslakian. On rolling cube puzzles. In *Proceedings of the 19th Canadian Conference on Computational Geometry*, pages 141–144, Ottawa, Canada, August 2007.

[BCG04] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*. A K Peters, Wellesley, MA, 2nd edition, 2001–2004.

[BDD+02] Therese C. Biedl, Erik D. Demaine, Martin L. Demaine, Rudolf Fleischer, Lars Jacobsen, and J. Ian Munro. The complexity of Clickomania. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 389–404. Cambridge University Press, 2002. Collection of papers from the MSRI Combinatorial Game Theory Research Workshop, Berkeley, California, July 2000.

[BDH+04] Ron Breukelaar, Erik D. Demaine, Susan Hohenberger, Hendrik Jan Hoogeboom, Walter A. Kosters, and David Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry and Applications*, 14(1–2):41–68, 2004.

[Bea85] John D. Beasley. *The Ins & Outs of Peg Solitaire*. Oxford University Press, 1985.

[Ber66] Robert Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[Ber00] Elwyn Berlekamp. *The Dots and Boxes Game: Sophisticated Child's Play*. A K Peters, Wellesley, MA, 2000.

[Bie05] Therese Biedl. The complexity of domino tiling. In *Proceedings of the 17th Canadian Conference on Computational Geometry*, pages 187–190, 2005.

[BMFN99] Adrian Brüngger, Ambros Marzetta, Komei Fukuda, and Jurg Nievergelt. The parallel search bench ZRAM and its applications. *Annals of Operations Research*, 90:45–63, 1999.

[BOS94] David Bremner, Joseph O'Rourke, and Thomas Shermer. Motion planning amidst movable square blocks is PSPACE complete. Draft, June 1994.

[Bou02] Charles L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, Series 2, 3:35–39, 1901–02.

[Bow07] Brian H. Bowditch. The angel game in the plane. *Combinatocis, Probability and Computing*, 16(3):345–362, 2007.

[Bur00] Michael Buro. Simple Amazons endgames and their connection to Hamilton circuits in cubic subgrid graphs. In *Proceedings of the 2nd International Conference on Computers and Games*, volume 2063 of *Lecture Notes in Computer Science*, pages 250–261, Hamamatsu, Japan, October 2000.

[BW70] John Bruno and Louis Weinberg. A constructive graph-theoretic solution of the Shannon switching game. *IEEE Transactions on Circuit Theory CT-17*, 1:74–81, February 1970.

[BW94] Elwyn Berlekamp and David Wolfe. *Mathematical Go: Chilling Gets the Last Point*. A. K. Peters, Ltd., 1994.

[CDP06] Gruia Calinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. In *Proceedings of the 7th Latin American Symposium on Theoretical Informatics*, pages 262–273, 2006.

[CFS97] Anne Condon, Joan Feigenbaum, and Carsten Lund Peter Shor. Random debaters and the hardness of approximating stochastic functions. *SIAM Journal on Computing*, 26(2):369–400, 1997.

[Con77] J. H. Conway. All games bright and beautiful. *American Mathematical Monthly*, 84:417–434, 1977.

[Con01] John H. Conway. *On Numbers and Games*. A K Peters, Wellesley, MA, 2nd edition, 2001.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd IEEE Symposium on the Foundations of Computer Science*, pages 151–158, 1971.

[Cor04] Graham Cormode. The hardness of the Lemmings game, or Oh no, more NP-completeness proofs. In *Proceedings of the 3rd International Conference on FUN with Algorithms*, pages 65–76, Isola d'Elba, Italy, May 2004.

[Crâ99] Marcel Crâsmaru. On the complexity of Tsume-Go. In *Proceedings of the 1st International Conference on Computers and Games*, volume 1558 of *Lecture Notes in Computer Science*, pages 222–231, London, UK, 1999. Springer-Verlag.

[CT00] Marcel Crâşmaru and John Tromp. Ladders are PSPACE-complete. In *Proceedings of the 2nd International Conference on Computers and Games*, volume 2063 of *Lecture Notes in Computer Science*, pages 241–249, 2000.

[CT02] Alice Chan and Alice Tsai. $1 \times n$ Konane: a summary of results. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 331–339, 2002.

[Cul98] Joseph Culberson. Sokoban is PSPACE-complete. In *Proceedings of the International Conference on Fun with Algorithms*, pages 65–76, Elba, Italy, June 1998.

[DD07] Erik D. Demaine and Martin L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23 (Supplement):195–208, 2007. Special issue on Computational Geometry and Graph Theory: The Akiyama-Chvatal Festschrift.

[DDE02] Erik D. Demaine, Martin L. Demaine, and David Eppstein. Phutball endgames are NP-hard. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 351–360. Cambridge University Press, 2002. Collection of papers from the MSRI Combinatorial Game Theory Research Workshop, Berkeley, California, July 2000. http://www.arXiv.org/abs/cs.CC/0008025.

[DDF$^+$08] Erik D. Demaine, Martin L. Demaine, Rudolf Fleischer, Robert A. Hearn, and Timo von Oertzen. The complexity of Dyson telescopes. In *Games of No Chance III*. 2008.

[DDLL06] Erik D. Demaine, Martin L. Demaine, Arthur Langerman, and Stefan Langerman. Morpion solitaire. *Theory of Computing Systems*, 39(3):439–453, June 2006.

[DDO00a] Erik D. Demaine, Martin L. Demaine, and Joseph O'Rourke. PushPush and Push-1 are NP-hard in 2D. In *Proceedings of the 12th Annual Canadian Conference on Computational Geometry*, pages 211–219, Fredericton, Canada, August 2000. http://www.cs.unb.ca/conf/cccg/eProceedings/26.ps.gz.

[DDO00b] Erik D. Demaine, Martin L. Demaine, and Joseph O'Rourke. PushPush is NP-hard in 2D. Technical Report 065, Department of Computer Science, Smith College, Northampton, MA, January 2000. http://arXiv.org/abs/cs.CG/0001019.

[DDV00] Erik D. Demaine, Martin L. Demaine, and Helena Verrill. Coin-moving puzzles. In *MSRI Combinatorial Game Theory Research Workshop*, Berkeley, California, July 2000.

[Del06] Jean-Paul Delahaye. The science behind Sudoku. *Scientific American*, pages 80–87, June 2006.

[DF83] James R. Driscoll and Merrick L. Furst. On the diameter of permutation groups. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 152–160, 1983.

[DF97] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1997.

[DH01] Erik D. Demaine and Michael Hoffmann. Pushing blocks is NP-complete for noncrossing solution paths. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 65–68, Waterloo, Canada, August 2001. http://compgeo.math.uwaterloo.ca/~cccg01/proceedings/long/eddemaine-24711.ps.

[DH08] Erik D. Demaine and Robert A. Hearn. Constraint logic: A uniform framework for modeling computation as games. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity*, College Park, Maryland, June 2008. To appear.

[DHH02] Erik D. Demaine, Robert A. Hearn, and Michael Hoffmann. Push-2-F is PSPACE-complete. In *Proceedings of the 14th Canadian Conference on Computational Geometry*, pages 31–35, Lethbridge, Canada, August 2002.

[DHH04] Erik D. Demaine, Michael Hoffmann, and Markus Holzer. PushPush-k is PSPACE-complete. In *Proceedings of the 3rd International Conference on FUN with Algorithms*, pages 159–170, Isola d'Elba, Italy, May 2004.

[DO92] Arundhati Dhagat and Joseph O'Rourke. Motion planning amidst movable square blocks. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, pages 188–191, 1992.

[Dud24] Henry E. Dudeney. *Strand Magazine*, 68:97 and 214, July 1924.

[dVV03] Sven de Vries and Rakesh Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, Summer 2003.

[DZ99] Dorit Dor and Uri Zwick. SOKOBAN and other motion planning problems. *Computational Geometry: Theory and Applications*, 13(4):215–228, 1999.

[Epp] David Eppstein. Computational complexity of games and puzzles. http://www.ics.uci.edu/~eppstein/cgt/hard.html.

[Epp87] David Eppstein. On the NP-completeness of cryptarithms. *SIGACT News*, 18(3):38–40, 1987.

[Ern95] Michael D. Ernst. Playing Konane mathematically: A combinatorial game-theoretic analysis. *UMAP Journal*, 16(2):95–121, Spring 1995.

[ET76] S. Even and R. E. Tarjan. A combinatorial problem which is complete in polynomial space. *Journal of the Association for Computing Machinery*, 23(4):710–719, 1976.

[FB02] Gary William Flake and Eric B. Baum. *Rush Hour* is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1–2):895–911, January 2002.

[Fer74] T. S. Ferguson. On sums of graph games with last player losing. *International Journal of Game Theory*, 3(3):159–167, 1974.

[Fer84] Thomas S. Ferguson. Misère annihilation games. *Journal of Combinatorial Theory*, Series A, 37:205–230, 1984.

[FG87] Aviezri S. Fraenkel and Elisheva Goldschmidt. PSPACE-hardness of some combinatorial games. *Journal of Combinatorial Theory*, Series A, 46:21–38, 1987.

[FGJ+78] 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 *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 55–64, Ann Arbor, Michigan, October 1978.

[FKUB05] Timothy Furtak, Masashi Kiyomi, Takeaki Uno, and Michael Buro. Generalized Amazons is PSPACE-complete. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 132–137, 2005.

[FL81] Aviezri S. Fraenkel and David Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n. *Journal of Combinatorial Theory*, Series A, 31:199–214, 1981.

[FM04] James D. Fix and Brandon McPhail. Offline 1-Minesweeper is NP-complete. Manuscript, 2004. http://people.reed.edu/~jimfix/papers/1MINESWEEPER.pdf.

[Fra74] Aviezri S. Fraenkel. Combinatorial games with an annihilation rule. In *The Influence of Computing on Mathematical and Research and Education*, volume 20 of *Proceedings of the Symposia in Applied Mathematics*, pages 87–91, 1974.

[Fra96] Aviezri S. Fraenkel. Scenic trails ascending from sea-level Nim to alpine Chess. In R. J. Nowakowski, editor, *Games of No Chance*, pages 13–42. Cambridge University Press, 1996.

[Fra07] Aviezri S. Fraenkel. Combinatorial games: Selected bibliography with a succinct gourmet introduction. *Electronic Journal of Combinatorics*, 1994–2007. Dynamic Survey DS2, http://www.combinatorics.org/Surveys/. One version also appears in *Games of No Chance*, pages 493–537, 1996.

[Fri01] Erich Friedman. Cubic is NP-complete. In *Proceedings of the 34th Annual Florida MAA Section Meeting*, 2001.

[Fri02a] Erich Friedman. Corral puzzles are NP-complete. Unpublished manuscript, August 2002. http://www.stetson.edu/~efriedma/papers/corral/corral.html.

[Fri02b] Erich Friedman. Pearl puzzles are NP-complete. Unpublished manuscript, August 2002. http://www.stetson.edu/~efriedma/papers/pearl/pearl.html.

[Fri02c] Erich Friedman. Pushing blocks in gravity is NP-hard. Unpublished manuscript, March 2002. http://www.stetson.edu/~efriedma/papers/gravity/gravity.html.

[Fri02d] Erich Friedman. Spiral galaxies puzzles are NP-complete. Unpublished manuscript, March 2002. http://www.stetson.edu/~efriedma/papers/spiral/spiral.html.

[FSU93] Aviezri S. Fraenkel, Edward R. Scheinerman, and Daniel Ullman. Undirected edge geography. *Theoretical Computer Science*, 112(2):371–381, 1993.

[FY76] A. S. Fraenkel and Y. Yesha. Theory of annihilation games. *Bulletin of the American Mathematical Society*, 82(5):775–777, September 1976.

[FY79] A. S. Fraenkel and Y. Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics*, 1:15–30, 1979.

[FY82] A. S. Fraenkel and Y. Yesha. Theory of annihilation games—I. *Journal of Combinatorial Theory*, Series B, 33:60–86, 1982.

[Gác07] Peter Gács. The angel wins. Manuscript, 2007. http://arXiv.org/abs/0706.2817.

[Gar63] Martin Gardner. Mathematical games column. *Scientific American*, 209(6):144, December 1963. Solution in January 1964 column.

[Gar64] Martin Gardner. The hypnotic fascination of sliding-block puzzles. *Scientific American*, 210:122–130, 1964. Appears as Chapter 7 of *Martin Gardner's Sixth Book of Mathematical Diversions*, University of Chicago Press, 1984.

[Gar65] Martin Gardner. Mathematical games column. *Scientific American*, 213(5):120–123, November 1965. Problem 9. Solution in December 1965 column.

[Gar75] Martin Gardner. Mathematical games column. *Scientific American*, 232(3):112–116, March 1975. Solution in April 1975 column.

[Gar86] Martin Gardner. Cram, bynum and quadraphage. In *Knotted Doughnuts and Other Mathematical Entertainments*, chapter 16. W. H. Freeman and Company, 1986.

[Gil00] Andrea Gilbert. Plank puzzles, 2000. http://www.clickmazes.com/planks/ixplanks.htm.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[GN02] Richard K. Guy and Richard J. Nowakowski. Unsolved problems in combinatorial games. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 457–473. Cambridge University Press, 2002.

[Gon86] Harry Gonshor. *An Introduction to the Theory of Surreal Numbers*. Cambridge University Press, 1986.

[GR95] Arthur S. Goldstein and Edward M. Reingold. The complexity of pursuit on a graph. *Theoretical Computer Science*, 143:93–112, 1995.

[Gru39] P. M. Grundy. Mathematics and games. *Eureka*, 2:6–8, October 1939. Reprinted in *Eureka: The Archimedeans' Journal*, 27:9–11, October 1964.

[GS56a] P. M. Grundy and C. A. B. Smith. Disjunctive games with the last player losing. *Proceedings of the Cambridge Philosophical Society*, 52:527–533, 1956.

[GS56b] Richard K. Guy and Cedric A. B. Smith. The G-values of various games. *Proceedings of the Cambridge Philosophical Society*, 52:514–526, 1956.

[Hau95] Jacques Haubrich. *Compendium of Card Matching Puzzles*. Self-published, May 1995. Three volumes.

[Hay06] Brian Hayes. Unwed numbers. *American Scientist*, 94(1):12, January–February 2006.

[HD02] Robert A. Hearn and Erik D. Demaine. The nondeterministic constraint logic model of computation: Reductions and applications. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 401–413, Malaga, Spain, July 2002.

[HD05] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, October 2005. Special issue "Game Theory Meets Theoretical Computer Science".

[Hea04] Robert A. Hearn. The complexity of sliding block puzzles and plank puzzles. In *Tribute to a Mathemagician*, pages 173–183. A K Peters, 2004.

[Hea05a] Robert A. Hearn. Amazons is PSPACE-complete. Manuscript, February 2005. http://www.arXiv.org/abs/cs.CC/0008025.

[Hea05b] Robert A. Hearn. The subway shuffle puzzle, 2005. http://www.subwayshuffle.com.

[Hea06a] Robert Hearn. TipOver is NP-complete. *Mathematical Intelligencer*, 28(3):10–14, 2006.

[Hea06b] Robert A. Hearn. *Games, Puzzles, and Computation*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 2006. http://www.swiss.ai.mit.edu/~bob/hearn-thesis-final.pdf.

[Hea08a] Robert A. Hearn. Amazons, Konane, and Cross Purposes are PSPACE-complete. In R. J. Nowakowski, editor, *Games of No Chance 3*, 2008. To appear.

[Hea08b] Robert A. Hearn. The complexity of Minesweeper revisited. Manuscript in preparation, 2008.

[Hea08c] Robert A. Hearn. Hitori is NP-complete. Manuscript in preparation, 2008.

[HEFN01] Falk Hüffner, Stefan Edelkamp, Henning Fernau, and Rolf Niedermeier. Finding optimal solutions to Atomix. In *Proceedings of the Joint German/Austrian Conference on AI: Advances in Artificial Intelligence*, volume 2174 of *Lecture Notes in Computer Science*, pages 229–243, Vienna, Austria, 2001.

[Hel03] Malte Helmert. Complexity results for standard benchmark domains in planning. *Artificial Intelligence*, 143(2):219–262, 2003.

[HKK04] Markus Holzer, Andreas Klein, and Martin Kutrib. On the NP-completeness of the NURIKABE pencil puzzle and variants thereof. In *Proceedings of the 3rd International Conference on FUN with Algorithms*, pages 77–89, Isola d'Elba, Italy, May 2004.

[HLH03] Jeffrey R. Hartline and Ran Libeskind-Hadas. The computational complexity of motion planning. *SIAM Review*, 45:543–557, 2003.

[Hoc01] Martin Hock. Exploring the complexity of the ufo puzzle. Undergraduate thesis, Carnegie Mellon University, 2001. http://www.cs.cmu.edu/afs/cs/user/mjs/ftp/thesis-02/hock.ps.

[Hof00] Michael Hoffmann. Push-* is NP-hard. In *Proceedings of the 12th Canadian Conference on Computational Geometry*, pages 205–210, Fredericton, Canada, August 2000. http://www.cs.unb.ca/conf/cccg/eProceedings/13.ps.gz.

[Hor86] Edward Hordern. *Sliding Piece Puzzles*. Oxford University Press, 1986.

[HR07] Markus Holzer and Oliver Ruepp. The troubles of interior design—a complexity analysis of the game Heyawake. In *Proceedings of the 4th International Conference on FUN with Algorithms*, volume 4475 of *Lecture Notes in Computer Science*, pages 198–212, 2007.

[HS04a] Markus Holzer and Stefan Schwoon. Assembling molecules in ATOMIX is hard. *Theoretical Computer Science*, 313(3):447–462, February 2004.

[HS04b] Markus Holzer and Stefan Schwoon. Reflections on REFLEXION—computational complexity considerations on a puzzle game. In *Proceedings of the 3rd International Conference on FUN with Algorithms*, pages 90–105, Isola d'Elba, Italy, May 2004.

[HSS84] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the 'Warehouseman's Problem'. *International Journal of Robotics Research*, 3(4):76–88, 1984.

[IK94] Shigeki Iwata and Takumi Kasai. The Othello game on an $n \times n$ board is PSPACE-complete. *Theoretical Computer Science*, 123:329–340, 1994.

[Jer85] Mark R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36(2–3):265–289, 1985.

[Jer86] Mark Jerrum. A compact representation for permutation groups. *Journal of Algorithms*, 7(1):60–78, 1986.

[Kay00a] Richard Kaye. Infinite versions of minesweeper are Turing-complete. Manuscript, August 2000. http://www.mat.bham.ac.uk/R.W.Kaye/minesw/infmsw.pdf.

[Kay00b] Richard Kaye. Minesweeper is NP-complete. *Mathematical Intelligencer*, 22(2):9–15, 2000.

[KC07] Daniel Kurkle and Gene Cooperman. Twenty-six moves suffice for Rubiks cube. In *Proceedings of International Symposium on Symbolic and Algebraic Computation*, pages 235–242, 2007.

[Kem03] David Kempe. On the complexity of the reflections game. Unpublished manuscript, January 2003. http://www-rcf.usc.edu/~dkempe/publications/reflections.pdf.

[Klo07] Oddvar Kloster. A solution to the Angel Problem. *Theoretical Computer Science*, 389(1–2):152–161, December 2007.

[KMS84] Daniel Kornhauser, Gary Miller, and Paul Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, pages 241–250, 1984.

[Knu74] Donald Knuth. *Surreal Numbers*. Addison-Wesley, Reading, Mass., 1974.

[LC89] K. Li and K. H. Cheng. Complexity of resource allocation and job scheduling problems on partitionable mesh connected systems. In *Proceedings of the 1st Annual IEEE Symposium on Parallel and Distributed Processing*, pages 358–365, May 1989.

[LM07] Luc Longpré and Pierre McKenzie. The complexity of solitaire. In *Proceedings of the 32nd International Symposium on Mathematical Foundations of Computer Science*, volume 4708 of *Lecture Notes in Computer Science*, pages 182–193, 2007.

[LMR00] Michael Lachmann, Cristopher Moore, and Ivan Rapaport. Who wins domineering on rectangular boards? In *MSRI Combinatorial Game Theory Research Workshop*, Berkeley, California, July 2000.

[LS80] David Lichtenstein and Michael Sipser. GO is polynomial-space hard. *Journal of the Association for Computing Machinery*, 27(2):393–401, April 1980.

[Mát07] András Máthé. The angel of power 2 wins. *Combinatocis, Probability and Computing*, 16(3):363–374, 2007.

[McK84] Pierre McKenzie. Permutations of bounded degree generate groups of polynomial diameter. *Information Processing Letters*, 19(5):253–254, November 1984.

[McP03] Brandon McPhail. The complexity of puzzles. Undergraduate thesis, Reed College, Portland, Oregon, 2003.

[McP05] Brandon McPhail. Light Up is NP-complete. Unpublished manuscript, 2005. http://www.cs.umass.edu/~mcphailb/papers/2005lightup.pdf.

[McP07] Brandon McPhail. Metapuzzles: Reducing SAT to your favorite puzzle. CS Theory talk, December 2007. http://www.cs.umass.edu/~mcphailb/papers/2007metapuzzles.pdf.

[ME02] Cristopher Moore and David Eppstein. One-dimensional peg solitaire, and duotaire. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 341–350. Cambridge University Press, 2002.

[MR01] Cristopher Moore and John Michael Robson. Hard tiling problems with simple tiles. *Discrete and Computational Geometry*, 26(4):573–590, 2001.

[NP96] Richard J. Nowakowski and David G. Poole. Geography played on products of directed cycles. In R. J. Nowakowski, editor, *Games of No Chance*, pages 329–337. Cambridge University Press, 1996.

[OS99] Joseph O'Rourke and the Smith Problem Solving Group. PushPush is NP-hard in 3D. Technical Report 064, Department of Computer Science, Smith College, Northampton, MA, November 1999. http://arXiv.org/abs/cs/9911013.

[Pap85] Christos H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31(2):288–301, 1985.

[Pap01] Christos H. Papadimitriou. Algorithms, games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, Crete, Greece, July 2001.

[Par95] Ian Parberry. A real-time algorithm for the $(n^2 - 1)$-puzzle. *Information Processing Letters*, 56(1):23–28, 1995.

[Plaa] Thane E. Plambeck. Advances in losing. In *Games of No Chance 3*. To appear. http://arxiv.org/abs/math/0603027.

[Plab] Thane E. Plambeck. miseregames.org.

[Pla05] Thane E. Plambeck. Taming the wild in impartial combinatorial games. *INTEGERS: Electronic Journal of Combinatorial Number Theory*, 5(G05), 2005.

[PS07] Thane E. Plambeck and Aaron N. Siegel. Misère quotients for impartial games. arXiv:math/0609825v5, August 2007. http://arxiv.org/abs/math/0609825.

[Rei80] Stefan Reisch. Gobang ist PSPACE-vollständig (Gobang is PSPACE-complete). *Acta Informatica*, 13:59–66, 1980.

[Rei81] Stefan Reisch. Hex ist PSPACE-vollständig (Hex is PSPACE-complete). *Acta Informatica*, 15:167–191, 1981.

[Ren05] Paul Rendell. A Turing machine implemented in Conway's Game of Life. http://rendell-attic.org/gol/tm.htm, January 2005.

[RM78] Edward Robertson and Ian Munro. NP-completeness, puzzles and games. *Utilitas Mathematica*, 13:99–116, 1978.

[Rob83] J. M. Robson. The complexity of Go. In *Proceedings of the IFIP 9th World Computer Congress on Information Processing*, pages 413–417, 1983.

[Rob84a] J. M. Robson. Combinatorial games with exponential space complete decision problems. In *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 498–506, 1984.

[Rob84b] J. M. Robson. N by N Checkers is EXPTIME complete. *SIAM Journal on Computing*, 13(2):252–267, May 1984.

[RW90] Daniel Ratner and Manfred Warmuth. The (n^2-1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10:111–137, 1990.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[SBB$^+$07] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, September 2007.

[SC79] Larry J. Stockmeyer and Ashok K. Chandra. Provably difficult combinatorial games. *SIAM Journal on Computing*, 8(2):151–174, 1979.

[Sch78] Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16:185–225, 1978.

[Sev] Merlijn Sevenster. Battleships as a decision problem. *ICGA Journal*, 27(3):142–147.

[Sie06] Aaron N. Siegel. Misère games and misère quotients. arXiv:math/0612616v2, December 2006. http://arxiv.org/abs/math/0612616.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*, pages 1–9, Austin, Texas, 1973.

[Smi66] Cedric A. B. Smith. Graphs and composite games. *Journal of Combinatorial Theory*, 1:51–81, 1966.

[Spr36] R. Sprague. Über mathematische Kampfspiele. *Tôhoku Mathematical Journal*, 41:438–444, 1935–36.

[SS06] Jerry Slocum and Dic Sonneveld. *The 15 Puzzle*. Slocum Puzzle Foundation, 2006.

[Ste03] James W. Stephens. The Kung Fu Packing Crate Maze, 2003. http://www.puzzlebeast.com/crate/.

[Sto79] W. E. Story. Note on the '15' puzzle. *American Mathematical Monthly*, 2:399–404, 1879.

[SY83] Paul Spirakis and Chee Yap. On the combinatorial complexity of motion coordination. Report 76, Computer Science Department, New York University, 1983.

[TC04] John Tromp and Rudy Cilibrasi. Limits of Rush Hour Logic complexity. Manuscript, June 2004. http://www.cwi.nl/~tromp/rh.ps.

[Thu90] William P. Thurston. Conway's tiling groups. *American Mathematical Monthly*, 97(8):757–773, October 1990.

[Tro00] John Tromp. On size 2 Rush Hour logic. Manuscript, December 2000. http://turing.wins.uva.nl/~peter/teaching/tromprh.ps.

[UI90] Ryuhei Uehara and Shigeki Iwata. Generalized Hi-Q is NP-complete. *Transactions of the IEICE*, E73:270–273, February 1990.

[UN96] Nobuhisa Ueda and Tadaaki Nagao. NP-completeness results for NONOGRAM via parsimonious reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan, May 1996.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Wil74] Richard M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory*, Series B, 16:86–96, 1974.

[Wil91] Gordon Wilfong. Motion planning in the presence of movable obstacles. *Annals of Mathematics and Artificial Intelligence*, 3(1):131–150, 1991.

[Wil04] Anne D. Williams. *The Jigsaw Puzzle: Piecing Together a History*. Berkley Books, New York, 2004.

[Wol94] Stephen Wolfram. *Cellular Automata and Complexity: Collected Papers*. Perseus Press, 1994.

[Wol02] David Wolfe. Go endgames are PSPACE-hard. In R. J. Nowakowski, editor, *More Games of No Chance*, pages 125–136. Cambridge University Press, 2002.

[Yat03] Takayuki Yato. Complexity and completeness of finding another solution and its application to puzzles. Master's thesis, University of Tokyo, Tokyo, Japan, January 2003.

[YS03] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, E86-A(5):1052–1060, 2003. Also IPSJ SIG Notes 2002-AL-87-2, 2002.

[YTK+01] Masaya Yokota, Tatsuie Tsukiji, Tomohiro Kitagawa, Gembu Morohashi, and Shigeki Iwata. Exptime-completeness of generalized Tsume-Shogi (in Japanese). *Transactions of the IEICE*, J84-D-I(3):239–246, 2001.

[Zwi02] Uri Zwick. Jenga. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 243–246, San Francisco, California, 2002.